Thorough Power Analysis on Falcon Gaussian Samplers and Practical Countermeasure

Xiuhan Lin[1], Shiduo Zhang[2], Yang Yu[2,3,4(✉)], Weijia Wang[1,4], Qidi You[5,6], Ximing Xu[7], and Xiaoyun Wang[1,2,3,4]

[1] School of Cyber Science and Technology, Shandong University, Qingdao, China
xhlin@mail.sdu.edu.cn
[2] Institute for Advanced Study, Tsinghua University, Beijing, China
{zsd,yu-yang,xiaoyunwang}@mail.tsinghua.edu.cn
[3] Zhongguancun Laboratory, Beijing, China
[4] State Key Laboratory of Cryptography and Digital Economy Security, China
wjwang@sdu.edu.cn
[5] State Key Laboratory of Space-Ground Integrated Information Technology, China
[6] Space star Technology Co., Ltd, China, youqd@spacestar.com.cn
[7] China Mobile Internet, xuximing@chinamobile.com

Abstract. Falcon is one of post-quantum signature schemes selected by NIST for standardization. With the deployment underway, its implementation security is of great importance. In this work, we focus on the side-channel security of Falcon and our contributions are threefold.

First, by exploiting the symplecticity of NTRU and a recent decoding technique, we dramatically improve the key recovery using power leakages within Falcon Gaussian samplers. Compared to the state of the art (Zhang, Lin, Yu and Wang, EUROCRYPT 2023), the amount of traces required by our attack for a full key recovery is reduced by at least 85%.

Secondly, we present a complete power analysis for two exposed power leakages within Falcon's integer Gaussian sampler. We identify new sources of these leakages, which have not been identified by previous works, and conduct detailed security evaluations within the reference implementation of Falcon on Chipwhisperer.

Thirdly, we propose effective and easy-to-implement countermeasures against both two leakages to protect the whole Falcon's integer Gaussian sampler. Configured with our countermeasures, we provide security evaluations on Chipwhisperer and report performance of protected implementation. Experimental results highlight that our countermeasures admit a practical trade-off between efficiency and side-channel security.

Keywords: Lattice-Based Cryptography · Side-Channel Analysis · Falcon Signature Scheme · Gaussian Sampler · NTRU

1 Introduction

In 2022, the US NIST announced the first batch of PQC algorithms to be standardized: Kyber [SAB+22] for public-key encryption and key establishment and

Dilithium [LDK+22], Falcon [PFH+22] and SPHINCS+ [HBD+22] for digital signatures. Among three signature standards, Falcon has competitive overall performance in particular the smallest communication cost (added sizes of a public key and a signature). This makes Falcon an attractive option for quantum-safe embedded systems.

For real-world deployment in embedded systems, implementation security is of great importance. Adversaries can exploit additional information given through side-channels, e.g. execution time, power consumption and electromagnetic radiations of the chips, to assist cryptanalysis and to mount possibly devastating attacks. Such physical attacks are nowadays the major threat to cryptographic embedded devices. For this, the latest NIST status report on PQC standardization process makes particular mention of side-channel analysis and notes that *"It is NIST's hope and expectation that more such work will continue, especially with regard to protecting the implementations of the algorithms announced for standardization"*.

The side-channel security of Falcon is considered as a notably challenging topic. Falcon is a lattice signature scheme based on the GPV hash-and-sign framework [GPV08]. Its signing procedure relies on sophisticated lattice Gaussian sampling and the way the secret key is used in signing is rather opaque. This complicates *identifying*, *exploiting* and *sealing* side-channel leakages. In addition, Falcon signing algorithm requires extensive floating-point operations that are notorious targets for side-channel attacks. While the reference implementation of Falcon can provably resist against timing attacks [HPRR20], the implementation does not come with protection against other types of side-channel attacks, e.g. power analysis. Some recent works [KA21,GMRR22,ZLYW23] have demonstrated side-channel vulnerabilities of Falcon implementations. Nevertheless, many operations of Falcon's algorithms still require closer scrutiny and systematic countermeasures are not well studied.

1.1 Related Works

Earlier side-channel attacks against lattice signatures targeted the Fiat-Shamir type constructions [BHLY16,EFGT17,PBY17,BDE+18]. This spurred the developments of constant-time Gaussian sampling [ZSS20,KRR+18] and masking [BBE+18,BBE+19,MGTF19,GR19].

Side-channel security of hash-and-sign lattice signatures, especially Falcon, greatly lags the Fiat-Shamir case in both attacks and protections. Fouque et al. presented a theoretical timing attack against the round-1 implementation of Falcon in [FKT+20]. The identified leakage has been provably patched [HPRR20]. The signing procedure of Falcon has two operations that leak secret information: pre-image computation and integer Gaussian sampling. Karabulut and Aysu demonstrated an electromagnetic analysis attack targeting the multiplication between two floating-point numbers on Falcon's pre-image computation [KA21]. With regard to Gaussian samplers, Guerreau et al. proposed the first power analysis attacks against Falcon [GMRR22] using the leakage in the base sam-

pler. Later, Zhang et al. [ZLYW23] improved this attack and identified another power leakage with respect to the sign flip in Falcon's integer Gaussian sampler.

Regarding to side-channel protections of Falcon, very few works are in the literature to our knowledge. Besides the isochronous implementation [HPRR20], a very recent work [CC24] provided the first masking floating-point multiplication and addition, which protects Falcon's pre-image computation against the attack of [KA21]. For power analysis on Gaussian samplers, only initial countermeasures were discussed along with corresponding attacks [GMRR22,ZLYW23] and a thorough treatment remains largely unexplored.

1.2 Contributions

The goal of this work is to give a better understanding of the side-channel security of Falcon from the aspects of both attack and defense. Our contributions are mainly threefold.

1. In Section 4, we improve the key recovery in the power analysis attack of [ZLYW23]. We make use of the symplecticity of NTRU to combine leakages at multiple positions, which refines the accuracy of statistical learning. Then a recent decoding technique [Pre23,LSZ+24] is applied to correct approximation errors. By this, we gain a substantial improvement: the number of required traces for successful key recovery is reduced by $\geq 85\%$.
2. Section 5 gives a complete analysis of the half Gaussian leakage and the sign leakage that are two crucial power leakages involved in Falcon's integer Gaussian samplers. We scrutinize each correlative component and identify some new sources of power leakages in the reference implementation of Falcon. Furthermore, quantitative side-channel evaluations are carried out on Chipwhisperer.
3. In Section 6, we propose practically effective and easy-to-implement countermeasures against both half Gaussian leakage and sign leakage. We implement the countermeasures in portable C and then perform side-channel evaluations on Chipwhisperer. Our countermeasures lower the classification accuracy of the template attack [CRR03] from 100% down to $\lesssim 58\%$ (resp. $\lesssim 62\%$) for the half Gaussian leakage (resp. sign leakage), which makes key recovery impractical. We evaluate the performance of the protected implementation on an Intel Core i5-1135G7 CPU. The overhead on the signing speed of our countermeasures is around $3.5\times$. To the best of our knowledge, our countermeasures are the first one protecting the whole Falcon Gaussian sampler from both half Gaussian leakage and sign leakage.

The source code is available at `https://github.com/lxhcrypto/FalconAnalysis` for sanity check and reproduction.

2 Preliminaries

Notations. We use bold lowercase letters for (row) vectors and denote by b_i the i-th entry of the vector $\mathbf{b}$. Let $\|\mathbf{b}\|$ (resp. $\|\mathbf{b}\|_1$ and $\|\mathbf{b}\|_\infty$) denote the Euclidean

norm (resp. ℓ_1-norm and ℓ_∞-norm) of $\mathbf{b} \in \mathbb{R}^n$. We use bold uppercase letters for matrices and denote by $\mathbf{b}_i$ the i-th row of the matrix $\mathbf{B}$. Let $\lfloor u \rceil$ be the operation rounding the real number u to the closest integer. This is naturally extended to $\mathbf{u} \in \mathbb{R}^n$ by taking rounding coefficient-wisely. We write $y \leftarrow D$ when the random variable y is drawn from the distribution D. Let $y \sim D$ denote the random variable y distributed over D and $D(x)$ denote the probability of $y = x$.

Lattices. Given $\mathbf{B} = (\mathbf{b}_1, \cdots, \mathbf{b}_n) \in \mathbb{R}^{n \times m}$ of full rank, the lattice generated by $\mathbf{B}$ is $\mathcal{L}(\mathbf{B}) = \{\sum_{i=1}^{n} x_i \mathbf{b}_i \mid x_i \in \mathbb{Z}\}$ and $\mathbf{B}$ is called a basis.

Gaussians. Let $\rho_{\sigma,\mathbf{c}}(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{c}\|^2}{2\sigma^2}\right)$ be the Gaussian function over $\mathbb{R}^n$ with standard deviation $\sigma > 0$ and center $\mathbf{c} \in \mathbb{R}^n$. Let $D_{\mathcal{L},\sigma,\mathbf{c}}$ be the discrete Gaussian over a lattice $\mathcal{L}$ defined by $D_{\mathcal{L},\sigma,\mathbf{c}}(\mathbf{u}) = \frac{\rho_{\sigma,\mathbf{c}}(\mathbf{u})}{\sum_{\mathbf{v} \in \mathcal{L}} \rho_{\sigma,\mathbf{c}}(\mathbf{v})}$ for any $\mathbf{u} \in \mathcal{L}$. When $\mathcal{L} = \mathbb{Z}$, $D_{\mathbb{Z},\sigma,c}$ is called integer Gaussian that is of particular interest. The half integer Gaussian is defined by $D^+_{\mathbb{Z},\sigma,c}(u) = \frac{\rho_{\sigma,c}(u)}{\sum_{v \in \mathbb{N}} \rho_{\sigma,c}(v)}$ for any $u \in \mathbb{N}$.

NTRU. Let $\mathcal{R} = \mathbb{Z}[x]/(x^n + 1)$ with n a power of 2. In a typical NTRU scheme, the secret is a pair of short $(f, g) \in \mathcal{R}^2$ and the public key is $h = g/f \bmod q$. The NTRU lattice defined by h is $\mathcal{L}_{NTRU} = \{(u, v) \in \mathcal{R}^2 \mid u + vh = 0 \bmod q\}$. One special basis of $\mathcal{L}_{NTRU}$, called NTRU trapdoor, is $\mathbf{B}_{f,g} = \begin{pmatrix} g & -f \\ G & -F \end{pmatrix}$ where $F, G \in \mathcal{R}$ such that $fG - gF = q$. We write $\mathbf{B}_{f,g}$ as $\mathbf{B}$ when the context is clear.

2.1 Falcon Signature Scheme

Falcon is an efficient instantiation of the GPV hash-and-sign framework [GPV08] over NTRU lattices. Specifically, the key pair of Falcon consists in the NTRU trapdoor basis $\mathbf{B}_{f,g}$ and public key $h = g/f \bmod q$. For compactness, Falcon chooses (f, g) such that $\|(f, g)\| \approx 1.17\sqrt{q}$ as per [DLP14]. In this paper, we mainly focus on the parameter set of Falcon-512 for the security level NIST-I, in which $n = 512$ and $q = 12289$.

The signing procedure of Falcon is described in Algorithm 1. Its signing is essentially to compute short $(s_1, s_2) \sim D_{(c,0)+\mathcal{L}(\mathbf{B}),\sigma}$ such that $s_1 + s_2 h = c \bmod q$ where $c = \mathsf{H}(r \| msg)$ and r is the salt. Falcon samples it by the fast Fourier sampler ffSampling [DP16]. The ffSampling sampler works on the FFT domain and takes the so-called Falcon tree T, i.e. the Gram-Schmidt Orthogonalization (GSO) of $\mathbf{B}$, as the input. We omit the details of FFT and Falcon tree as they are not necessary for understanding our work. The acceptance bound of signatures is $\lfloor \beta^2 \rfloor$ such that $\beta = 1.1 \cdot \sigma\sqrt{2n}$ where $\sigma = 1.17\sqrt{q} \cdot \eta_\epsilon(\mathcal{R}^2)$ and $\eta_\epsilon(\mathcal{R}^2)$ is the smoothing parameter.

Algorithm 1: Sign

Input: Message msg, NTRU basis $\mathbf{B}_{f,g}$ and acceptance bound $\lfloor \beta^2 \rfloor$
Output: A valid signature (r, s) of msg

$r \xleftarrow{\$} \{0,1\}^{320}$, $c \leftarrow \mathsf{H}(r \| msg)$
$\mathbf{t} \leftarrow (\mathsf{FFT}(c), \mathsf{FFT}(0)) \cdot \mathsf{FFT}(\mathbf{B}_{f,g})^{-1}$ ▷ pre-image computation
do
  do
    $\mathbf{z} \leftarrow \mathsf{ffSampling}(\mathbf{t}, \mathsf{T})$ ▷ trapdoor sampling
    $\mathbf{s} \leftarrow (\mathbf{t} - \mathbf{z}) \cdot \mathsf{FFT}(\mathbf{B}_{f,g})$ ▷ $\mathbf{s} \sim D_{(c,0)+\mathcal{L}(\mathbf{B}),\sigma}$
  while $\|\mathbf{s}\|^2 > \lfloor \beta^2 \rfloor$
  $(s_1, s_2) \leftarrow \mathsf{invFFT}(\mathbf{s})$
  $s \leftarrow \mathsf{Compress}(s_2)$
while $s = \perp$
return (r, s)

Floating-point representation. In the generation of Falcon signature and Falcon tree, involved polynomials are represented in the FFT domain, whose coefficients are complex numbers. The real and imaginary part of complex numbers are floating-point values in "double precision" (also called "binary64" format). As per the reference implementation of Falcon, it follows the IEEE-754 standard and uses unsigned 64-bit integer type to encode floating-point values with 53-bit precision. The full 64-bit is divided into three parts: the sign s is specified by the first most significant bit, the following 11 most significant bits represent the exponent e and the remaining 52 least significant bits denote the mantissa m. However, the mantissa m in effect takes 53-bit where 53rd bit remains 1 omitted in the storage. These three parts (s, e, m) assemble into the floating-point value which is given as follows:

$$x = (-1)^s \cdot 2^{e-1023} \cdot (1 + m \cdot 2^{-52}).$$

In the 64-bit floating-point number, the exponent e contains 2046 values from 1 to 2046. More precisely, when $e = 2047$, the value x is either infinity or the erroneous value (also known as NaN). Similarly, when $e = 0$, the value x is either a zero or subnormal. In this format, the mantissa m is an integer such that $m \in [2^{52}, 2^{53})$. In latter discussion, we write the floating-point representation as FPR and Hamming weight as HW for simplicity.

3 Falcon's Integer Gaussian Samplers and Their Leakages

This work focuses on the side-channel security with respect to Falcon's integer Gaussian samplers. Let us first briefly introduce the Gaussian samplers and existing power leakages.

3.1 Algorithmic Descriptions

Falcon uses the ring-efficient Klein-GPV algorithm [DP16] converting lattice Gaussian sampling into a series of integer Gaussian samplings. To deal with variable Gaussian parameters, Falcon implements the integer Gaussian sampler based on rejection sampling and a fixed base sampler. To recap, there are three levels of samplers in Falcon as illustrated in Figure 1. Our work targets the integer sampler SamplerZ and the base sampler BaseSampler.

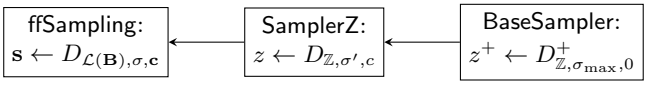


Fig. 1: Flowchart of Falcon's Gaussian samplers.

SamplerZ To sample from $D_{\mathbb{Z},\sigma',c}$ with different (σ', c), SamplerZ first draws $z^+ \sim D^+_{\mathbb{Z},\sigma_{\max},0}$ using the base sampler, then computes a bimodal Gaussian $z \leftarrow b + (2b-1)z^+$ via a random bit b, and finally applies rejection sampling to guarantee the correct output distribution. The full algorithmic description is presented in Algorithm 2. In Falcon-512, $\sigma_{\min} = 1.2778$ and $\sigma_{\max} = 1.8205$. In addition, the reference implementation of SamplerZ is provably resistant against timing attacks [HPRR20].

Algorithm 2: SamplerZ

Input: Center c and standard deviation $\sigma' \in [\sigma_{\min}, \sigma_{\max}]$ that are in FPR
Output: An integer $z \in \mathbb{Z}$ such that $z \sim D_{\mathbb{Z},\sigma',c}$

$r \leftarrow c - \lfloor c \rfloor$ ▷ $r \in [0, 1)$
$ccs \leftarrow \sigma_{\min}/\sigma'$
while *(1)* **do**
  $z^+ \leftarrow$ BaseSampler()
  $b \xleftarrow{\$} \{0, 1\}$
  $z \leftarrow b + (2b-1)z^+$
  $x \leftarrow \frac{(z-r)^2}{2\sigma'^2} - \frac{(z^+)^2}{2\sigma^2_{\max}}$ ▷ With the arithmetics of FPR
  if BerExp$(x, ccs) = 1$ **then** ▷ reject sampling
    return $z + \lfloor c \rfloor$
  end if
end while

Algorithm 3: BerExp

Input: x and $ccs \geq 0$ that are in FPR
Output: 1 with probability $\approx ccs \cdot \exp(-x)$

$s' \leftarrow \lfloor x/\ln(2) \rfloor$ ▷ $s' \in \mathbb{Z}^+$
$r' \leftarrow x - s' \cdot \ln(2)$ ▷ $r' \in [0, \ln(2))$
$s' \leftarrow \min(s', 63)$
$z' \leftarrow ((\mathsf{ApproxExp}(r', ccs) \ll 1) - 1) \gg s'$ ▷ $z' \approx 2^{64-s'} \cdot ccs \cdot \exp(-r')$
$i \leftarrow 64$
do ▷ `This loop is not constant-time`
  $i \leftarrow i - 8$, $y \xleftarrow{\$} \{0,1\}^8$
  $w \leftarrow y - ((z' \gg i)$ & 0xFF$)$
while $((w = 0)\ and\ (i > 0))$
return $[\![w < 0]\!]$

In Algorithm 3, given inputs $x = s' \cdot \ln(2) + r'$ and $ccs \geq 0$, BerExp first decomposes x into s' and r', then computes $z' \approx 2^{64} \cdot ccs \cdot \exp(-x)$ and performs lazy Bernoulli sampling to accept z with probability $2^{-64} \cdot z' \approx ccs \cdot \exp(-x)$. The subroutine ApproxExp computes approximate $\exp(-x)$ by polynomial approximation [ZSS20].

BaseSampler The base sampler BaseSampler is implemented with table-based approach. More concretely, it uses the reverse cumulative distribution table (RCDT) that consists of 18 items computed in 72-bit precision. To sample from $D^+_{\mathbb{Z},\sigma_{\max},0}$, BaseSampler draws a 72-bit random value u and determines the output via successively comparing u and each table item RCDT$[i]$. Algorithm 4 shows the algorithmic description.

Algorithm 4: BaseSampler

Input: -
Output: An integer $z^+ \sim D^+_{\mathbb{Z},\sigma_{\max},0}$

$u \xleftarrow{\$} \{0,1\}^{72}$
$z^+ \leftarrow 0$
for $i = 0, \cdots, 17$ **do**
  $z^+ \leftarrow z^+ + [\![u < \mathsf{RCDT}[i]]\!]$
end for
return z^+ ▷ $z^+ \in \{0, \cdots, 18\}$

3.2 Power Leakages

In context of power analysis, Falcon's integer Gaussian samplers currently exist two kind of leakages, known as half Gaussian leakage and sign leakage. Two leakages can be exploited to mount key recovery attacks [GMRR22,ZLYW23]. Next, we briefly recall these two leakages.

Half Gaussian Leakage Within the Falcon reference implementation, Guerreau et. al [GMRR22] first observed the visible differences in power consumption by practical simple power analysis in BaseSampler. More concretely, the different Hamming weight of the comparison $[\![u < \mathsf{RCDT}[i]]\!]$ (line 4, Algorithm 4) leads to significant power consumption, which involves at least 8-bit power leakage. This allows to accurately distinguish $z^+ = 0$ or not. The leakage of z^+, thus called half Gaussian leakage, can be combined with statistical learning techniques to recover the signing key as shown in [GMRR22,ZLYW23]. Besides BaseSampler, the half Gaussian leakage spreads to sign flip of z, calculation of x, BerExp and operation "return" in SamplerZ. A complete analysis of this leakage is given in Section 5.1.

Sign Leakage The sign leakage was first identified and exploited in [ZLYW23] within Falcon implementation. To be specific, the sign leakage has been shown to exist in the generation of sign b, sign flip $[\![z \leftarrow b + (2b-1)z^+]\!]$ (line 6, Algorithm 2) and computation $[\![x \leftarrow \frac{(z-r)^2}{2\sigma'^2} - \frac{(z^+)^2}{2\sigma_{\max}^2}]\!]$ (line 7, Algorithm 2). In this work, we further identify this leakage within the rejection sampling BerExp and the operation "return" in SamplerZ. Detailed analysis is provided in Section 5.2.

4 Improvements on the Key Recovery of [ZLYW23]

In [ZLYW23], Zhang et al. presented a side-channel assisted key recovery attack on Falcon. They exploited the half Gaussian leakage and sign leakage to filter signatures in specific domains, and then extract an approximate signing key via statistical learning. For a full key recovery, they resorted to exhaustive enumeration to correct approximate errors. This is only applicable to very small errors, thus requires a large number of power traces.

In this section, we further make use of the symplecticity of NTRU to refine the statistical learning accuracy and combine some recent decoding technique [Pre23,LSZ+24] to refine the key recovery. We dramatically reduced the number of required traces by at least 85%. Detailed results are shown in Table 1, which compares the required traces with success rate at least 25%.

Table 1: Number of required traces by [ZLYW23] and our attack for key recovery against Falcon-512.

	Half Gaussian leakage	Sign leakage	Both two leakages
[ZLYW23]	220,000	170,000	45,000
This work	27,500	25,000	6,500
Vs.	↓ 88%	↓ 85%	↓ 86%

4.1 Refining the Learning Accuracy with NTRU Symplecticity

In [ZLYW23], the attack starts with using power leakages of the $(2n-1)$-th integer sample z_{2n-1} to filter the signature $\mathbf{s} = \sum_{i=1}^{2n} y_i \cdot \mathbf{b}_i^*$ with y_1 in the specific range, where $\mathbf{b}_i^*$ is i-th row of the GSO of the Falcon trapdoor $\mathbf{B}$. Then the attack computes an approximate $\mathbf{b}_1$ by statistical learning. We observe that the signatures filtered by the samples z_{2n}, z_1 and z_2 can be directly used to refine the approximation of $\mathbf{b}_1$ due to the algebraic properties of the Falcon basis.

Falcon uses a Gaussian sampler based on fast Fourier orthogonalization [DP16], which is a ring-efficient Klein-GPV sampler performed on the NTRU trapdoor basis in the FFT order. Hence, the samples z_{2n-1}, z_{2n}, z_1 and z_2 respectively correspond to $\mathbf{b}_1^*$, $\mathbf{b}_{n/2+1}^*$, $\mathbf{b}_{3n/2}^*$ and $\mathbf{b}_{2n}^*$. According to the symplecticity of NTRU bases [GHN06], we have

$$\frac{\mathbf{b}_1^*}{\|\mathbf{b}_1^*\|} = \frac{\mathbf{b}_{n/2+1}^*}{\|\mathbf{b}_{n/2+1}^*\|} \cdot \mathbf{P} = -\frac{\mathbf{b}_{3n/2}^*}{\|\mathbf{b}_{3n/2}^*\|} \cdot \mathbf{P} \cdot \mathbf{J} \cdot \mathbf{Q} = \frac{\mathbf{b}_{2n}^*}{\|\mathbf{b}_{2n}^*\|} \cdot \mathbf{J} \cdot \mathbf{Q}$$

where $\mathbf{P} = \begin{pmatrix} & -\mathbf{I}_{n/2} & & \\ \mathbf{I}_{n/2} & & & \\ & & & -\mathbf{I}_{n/2} \\ & & \mathbf{I}_{n/2} & \end{pmatrix}$, $\mathbf{J}$ is a $2n \times 2n$ reversed identity matrix and $\mathbf{Q} = \begin{pmatrix} -\mathbf{I}_n & \\ & \mathbf{I}_n \end{pmatrix}$. The power leakage analysis of z_{2n}, z_1 and z_2 is essentially the same with that of z_{2n-1}. Therefore, the signatures filtered by z_{2n-1}, z_{2n}, z_1 and z_2 can be used to learn a more accurate direction of $\mathbf{b}_1$, while other z_i's cannot refine the learning straightforwardly.

Basically, compared with [ZLYW23], one trace can contribute three additional leakages carrying the information of $\mathbf{b}_1$. As a consequence, nearly 4 times samples yields a better learning accuracy. Then we present how to transform the signatures with filtered z_{2n}, z_1 and z_2 into equivalent ones with filtered z_{2n-1}. Let $\mathcal{S} = \varnothing$ be the initialized filtered signature set. We assume that every signature $\mathbf{s} = (s_1, s_2)$ is determined by the exploitable power leakage, i.e. half Gaussian leakage, sign leakage or both. The transformation proceeds as follows:

- For z_{2n-1}, $\mathcal{S} = \mathcal{S} \cup \{\mathbf{s}\}$,
- For z_{2n}, $\mathcal{S} = \mathcal{S} \cup \{\mathbf{s} \cdot \mathbf{P}\}$,
- For z_1, $\mathcal{S} = \mathcal{S} \cup \{-\mathbf{s} \cdot \mathbf{P} \cdot \mathbf{Q} \cdot \mathbf{J}\}$,
- For z_2, $\mathcal{S} = \mathcal{S} \cup \{\mathbf{s} \cdot \mathbf{Q} \cdot \mathbf{J}\}$.

4.2 Correcting Errors with Decoding Technique

By refined learning, one can obtain a noisy vector $\mathbf{b}' = ((b')^{(1)}, (b')^{(2)})$ of the secret basis vector $\mathbf{b} = (b^{(1)}, b^{(2)})$ i.e. $\mathbf{b}_1 = (g, -f) \in \mathcal{R}^2$. Let $\mathbf{b}' = \mathbf{b} + \mathbf{e}$. As per experimental results, $\|\mathbf{e}\|_\infty < 1$ when a moderate number of traces are used. For this case, $\mathbf{v} = \lfloor \mathbf{b}' \rceil - \mathbf{b} = (v^{(1)}, v^{(2)}) \in \{-1, 0, 1\}^{2n}$. The error $\mathbf{v}$ can be eliminated by exhaustive search or lattice reduction, however these methods are costly unless $\mathbf{v}$ is well-bounded, say $\|\mathbf{v}\|_1 < 10$.

To improve the practicality of attacks, we make use of the decoding technique introduced by Prest [Pre23]. This method turns out to be applicable to $\mathbf{v}$ of much larger size, say $\|\mathbf{v}\|_1 > 50$, thus leads to a great reduction on the number of required traces. Specifically, Prest's decoding technique is based on a simple observation that if the half coefficients of $\mathbf{b}'$ are exactly determined, then one can recover the entire NTRU secret by solving linear equations. To this end, it suffices to recover n secret coefficients of $\mathbf{b}$.

Slightly different from Prest's technique [Pre23], we employ the probability-based method of [LSZ+24] to select the half coefficients of $\mathbf{b}'$. In this setting, we simply model $\mathbf{e} = \mathbf{b}' - \mathbf{b}$ as a $2n$-dimensional spherical Gaussian vector of standard deviation σ and center 0. For simplistic hypothesis, it's a simple model that is sufficient to further improve our above refined results. With regard to σ, it is inversely proportional to the number of signatures S used in the recovering of $\mathbf{v}$. In our setting, we can approximate $\sigma \approx K/\sqrt{S}$ with some fixed constants K, which can be experimentally obtained by curve fitting (see Figure 2).

As illustrated above, we can guess the best half coefficients of $\mathbf{b}'$ according to the fractional part of b'_i. It can be verified that the probability of $v_i = \lfloor b'_i \rceil - b_i = 0$ is $\varphi_\sigma(x_i) = \frac{\rho_\sigma(x_i)}{\rho_\sigma(x_i+\mathbb{Z})}$ (see Lemma 4 of [LSZ+24]) where known value $x_i = b'_i - \lfloor b'_i \rceil$. By exploiting such probability-based strategy, for noisy vector $\mathbf{b}'$, we can in effect select n coefficients with the smallest fractional part of b'_i, i.e. correctly rounded with the highest probability. Thus, we have the probability p_i that is computed by $\varphi_\sigma(x_i)$ for i-th coefficient of the vector $\mathbf{x} = (x_1, \cdots, x_{2n}) = \mathbf{b}' - \lfloor \mathbf{b}' \rceil$. We further sort the probability p_i in decreasing order as follows:

$$0 \leq p_1 \leq p_2 \leq \cdots \leq p_{2n-1} \leq p_{2n} \leq 1.$$

As shown above, the best half of coefficients with highest probability can be effectively guessed. Then, we have the equation $b^{(1)} + b^{(2)} \cdot h = 0 \bmod q$ and then $\lfloor (b')^{(1)} \rceil + \lfloor (b')^{(2)} \rceil \cdot h = v^{(1)} + v^{(2)} \cdot h \bmod q$. It means that we can obtain $\mathbf{v}$ by solving the linear system and recover $\mathbf{b}$ entirely.

4.3 Experimental Results

Then, we experimentally verify the effectiveness of aforementioned approach on Falcon-512 instances. We use the same power leakages as in [ZLYW23], refine the statistical learning with the algebraic properties of Falcon trapdoor and finally mount a key recovery using the decoding technique. We greatly reduce the measurement cost compared to [ZLYW23]. For the attack using solely half Gaussian leakage, one can fully recover the signing key with $27,500$ traces on 14 out of 40 instances. For the attack using solely sign leakage, one can completely recover the secret with $25,000$ on 24 out of 40 instances. When the attack simultaneously uses both two leakages, one can launch a full key recovery by decoding with $6,500$ traces on 15 out of 40 instances. Figure 2 shows more detailed experimental results.

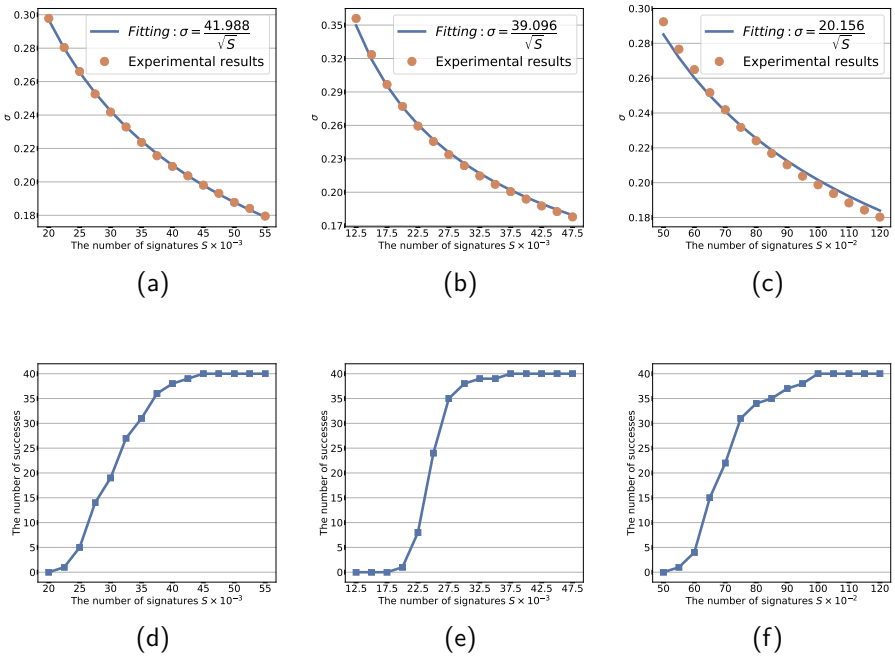


Fig. 2: Experimental results of refined key recovery attacks against Falcon-512 respectively using half Gaussian leakage (left), sign leakage (middle) and both two leakages (right). Figures (a),(b),(c) are for the measures of some constants K through curve fitting and (d),(e),(f) are for the number of successes by refined attacks. Experimental measures are over 40 instances.

5 Complete Analysis of Half Gaussian and Sign Leakages

In [GMRR22,ZLYW23], they identified some sources of half Gaussian leakage and sign leakage and gave empirical countermeasures. Unfortunately, even with their countermeasures, one can still learn the sensitive values with high probability (see Section 6.1 for more details). In this section, we further discover some new sources of two leakages by a complete power analysis on the Falcon integer Gaussian sampler SamplerZ. With our new identified leakage sources, one can also learn the sensitive values with high accuracy.

5.1 Power Analysis on Half Gaussian Leakage

The half Gaussian leakage is used to classify whether the output of BaseSampler $z^+ = 0$ or not. Besides BaseSampler, this leakage also exists in the following operations of SamplerZ including the sign flip of z, the computation of x, the

rejection sampling BerExp and the operation "return". We next analyze the half Gaussian leakage throughout SamplerZ.

Half Gaussian leakage in BaseSampler. In BaseSampler, the comparison $[\![u < \mathsf{RCDT}[i]]\!]$ (line 4, Algorithm 4) is the significant source of half Gaussian leakage first identified in [GMRR22]. More specifically, $[\![u < \mathsf{RCDT}[i]]\!]$ is completed by three successive subtractions with a carry bit. The entries of RCDT and u are split into three 24-bit limbs stored in 32-bit registers. For 32-bit two's complement, the first 8 most significant bits are regarded as "sign bits" which enhances power leakages. When $[\![u < \mathsf{RCDT}[i]]\!]$ returns true, 8 most significant bits of 32-bit register are all set 1. Otherwise, 8 most significant bits remain 0. The Hamming distance is 8 for the "sign bits". In addition, the increments of z^+ also produce half Gaussian leakage, but this part is minor compared to the leakage from $[\![u < \mathsf{RCDT}[i]]\!]$.

Half Gaussian leakage in the sign flip of z. The sign flip $[\![z \leftarrow b+(2b-1)z^+]\!]$ (line 6, Algorithm 2) also leaks the information of z^+. When $z^+ = 0$, the sign flip returns $z \in \{0, 1\}$. The dominating power leakage of the sign flip stems from $[\![(2b-1)z^+]\!]$. Regardless of the random bit b, the Hamming weight differences exist between $z^+ = 0$ and $z^+ \in \{1, \cdots, 18\}$. Thus, the largest Hamming weight gap is between $z^+ = 0$ and $z^+ = 15$, which happens rarely. In addition, $(2b-1) = -1$ further enlarges the Hamming weight difference by 32-bit two's complement, since z and z^+ are both stored in the 32-bit registers.

Half Gaussian leakage in the computation of x. Concretely, $[\![x \leftarrow \frac{(z-r)^2}{2\sigma'^2} - \frac{(z^+)^2}{2\sigma_{\max}^2}]\!]$ (line 7, Algorithm 2) is first derived by using a series of FPR arithmetics. It is split into three parts: $[\![x_1 \leftarrow \frac{(z-r)^2}{2\sigma'^2}]\!]$, $[\![x_2 \leftarrow \frac{(z^+)^2}{2\sigma_{\max}^2}]\!]$ and $[\![x \leftarrow x_1 - x_2]\!]$. We mainly focus on the analysis of $[\![x_2 \leftarrow \frac{(z^+)^2}{2\sigma_{\max}^2}]\!]$ which is the most relevant to the half Gaussian leakage. To compute x_2, z^+ is first transformed into a floating-point number with FPR for compatibility. The maximal difference in Hamming weight is 10 when $z^+ = 0$ and $z^+ \neq 0$ for $[\![(z^+)^2]\!]$. Then x_2 is fulfilled and further enlarges the gap of Hamming weight. As shown in Table 2, the difference in Hamming weight of x_2 is up to 39, which makes power consumption more significant. The computation of x_1 is indirectly related to half Gaussian leakage and takes the information of z^+ yet. It's noted that $[\![x \leftarrow x_1 - x_2]\!]$ also amplifies the half Gaussian leakage. On the whole, the computation of x involves many FPR arithmetics, including subtraction, multiplication and square, which extensively increases the Hamming weight gap and power consumption for half Gaussian leakage.

Half Gaussian leakage in BerExp. The subroutine BerExp takes x as one of inputs and its computations are also relevant to half Gaussian leakage. More precisely, the decompositions $[\![s' \leftarrow \lfloor x/\ln(2) \rfloor]\!]$ and $[\![r' \leftarrow x - s' \cdot \ln(2)]\!]$, and the computation of z', can also cause the half Gaussian leakage. However, this leakage is relatively weak in BerExp.

Half Gaussian leakage in "return". At the end of SamplerZ, it returns $[\![z+\lfloor c \rfloor]\!]$. It's clear that the "return" operation also exposes half Gaussian leakage, since z carries the information of z^+. For each sampling, the center c can be seen

Table 2: The FPR and HW of $[\![(z^+)^2]\!]$ and $[\![x_2 \leftarrow \frac{(z^+)^2}{2\sigma^2_{\max}}]\!]$ from $z^+ = 0$ to $z^+ = 6$

z^+	FPR of $[\![(z^+)^2]\!]$	HW of $[\![(z^+)^2]\!]$	FPR of x_2	HW of x_2
0	0X0000000000000000	0	0X0000000000000000	0
1	0X3FF0000000000000	10	0X3FC34F8BC183BBC2	34
2	0X4010000000000000	2	0X3FE34F8BC183BBC2	35
3	0X4022000000000000	3	0X3FF5B97D39B4333A	39
4	0X4030000000000000	3	0X40034F8BC183BBC2	27
5	0X4039000000000000	5	0X400E2C4A5E5DD55F	31
6	0X4042000000000000	3	0X4015B97D39B4333A	31

as random and thus masks partial information of z^+. Interestingly, one can still detect the half Gaussian leakage within the operation "return" on more precise devices.

5.2 Power Analysis on Sign Leakage

The sign leakage is used to distinguish $z \leq 0$ or $z \geq 1$, first exploited in [ZLYW23]. For this leakage, [ZLYW23] only gave detailed side-channel analysis towards the generation of b, the sign flip of z and the calculation of x. However, in SamplerZ, the rejection sampling BerExp and the operation "return" also reveal the sign information.

Sign leakage in the generation of b. In SamplerZ, the sign of z is determined by the random bit b. The step $[\![b \xleftarrow{\$} \{0,1\}]\!]$ (line 5, Algorithm 2) is original source of the sign leakage (discussed in [ZLYW23]). The Hamming distance is only 1 between $b = 0$ and $b = 1$. Even considering transfer on the bus or storage in the register, the magnitude of power consumption is approximately 1-bit during the generation of b.

Sign leakage in the sign flip of z. For $[\![z \leftarrow b + (2b-1)z^+]\!]$ (line 6, Algorithm 2), when $b = 0$, the sign is flipped and it outputs $[\![z \leftarrow -z^+]\!]$. Otherwise, it returns $[\![z \leftarrow 1 + z^+]\!]$. The significant power consumption and large Hamming distance exist in these two cases, which are shown in the Table 3. For 32-bit two's complement, the negative values take large Hamming weight. On the contrary, the positive ones have small Hamming weight. In particular, the maximal Hamming distance is 31 for the case $z = 1$ and the case $z = -1$. As shown in [ZLYW23], the sign leakage can be easily identified in the sign flip of z.

Sign leakage in the computation of x. The sign leakage also involves in the computation of $[\![x \leftarrow \frac{(z-r)^2}{2\sigma'^2} - \frac{(z^+)^2}{2\sigma^2_{\max}}]\!]$ (line 7, Algorithm 2). The computation of $[\![x_2 \leftarrow \frac{(z^+)^2}{2\sigma^2_{\max}}]\!]$ is irrelevant to the sign leakage, thus we mainly analyze $[\![x_1 \leftarrow \frac{(z-r)^2}{2\sigma'^2}]\!]$. Specifically, the 32-bit integer z is first converted into FPR, then minus center r, and finally the following square and multiplication are performed.

Table 3: The two's complement numbers and HW of z from $z = -3$ to $z = 4$

z	32-bit two's complement	HW
-3	b'11111111 11111111 11111111 11111101'	31
-2	b'11111111 11111111 11111111 11111110'	31
-1	b'11111111 11111111 11111111 11111111'	32
0	b'00000000 00000000 00000000 00000000'	0
1	b'00000000 00000000 00000000 00000001'	1
2	b'00000000 00000000 00000000 00000010'	1
3	b'00000000 00000000 00000000 00000011'	2
4	b'00000000 00000000 00000000 00000100'	1

Table 4 exhibits the Hamming weights of some example cases. From $z = -3$ to $z = 4$, the maximal difference in Hamming weight of $[\![(z-r)^2]\!]$ is 25 for $z = 0$ and $z = 1$, and the minimal one is 0 for $z = -1$ and $z = 3$. With respect to the results of x_1, the corresponding maximal difference in Hamming weight is 29 for $z = -1$ and $z = 3$, and the minimal one is 0 for $z = -2$ and $z = 4$. In practice, the average difference in Hamming weight is not very large between the cases $z \leq 0$ and $z \geq 1$.

Table 4: The FPR and HW of $[\![(z-r)^2]\!]$ and $[\![x_1 \leftarrow \frac{(z-r)^2}{2\sigma'^2}]\!]$ from $z = -3$ to $z = 4$

(z, r)	FPR of $[\![(z-r)^2]\!]$	HW of $[\![(z-r)^2]\!]$	FPR of x_1	HW of x_1
(-3, 0.9)	0X402E6B851EB851EB	30	0X40050D4985C1FE3B	27
(-2, 0.8)	0X401F5C28F5C28F5B	31	0X3FF5B3D1F00E2C4A	34
(-1, 0.7)	0X40071EB851EB851E	28	0X3FE0000000000000	9
(0, 0.6)	0X3FD70A3D70A3D70A	34	0X3FAFE3A76B2EF2B7	42
(1, 0.5)	0X3FD0000000000000	9	0X3FA6253443526171	29
(2, 0.4)	0X40047AE147AE147C	27	0X3FDC5894D10D4988	29
(3, 0.3)	0X401D28F5C28F5C2A	28	0X3FF42E0FF1D3B599	38
(4, 0.2)	0X402CE147AE147AE1	28	0X4003FC74ED65DE57	34

Sign leakage in BerExp. As discussed above, the input x of BerExp carries the sign leakage. Similar to the half Gaussian leakage, the sign leakage also spreads to the decomposition of x, including $[\![s' \leftarrow \lfloor x/\ln(2) \rfloor]\!]$ and $[\![r' \leftarrow x - s' \cdot \ln(2)]\!]$, the polynomial approximation ApproxExp and the right shift operation. However, BerExp exposes less information of the sign b, as the sign leakage might be weaken in the computation of x.

Sign leakage in "return". For $[\![z + \lfloor c \rfloor]\!]$, the randomly generated center c cannot entirely conceal the information of the sign b. Therefore, the operation "return" in SamplerZ can also identify the sign leakage as expected.

5.3 Practical Evaluations

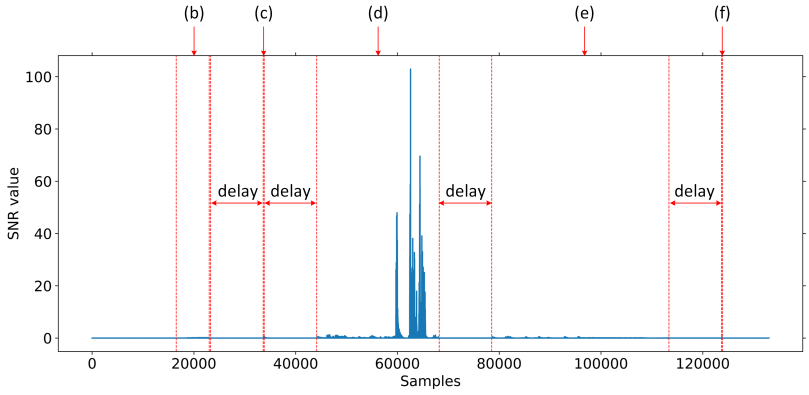
Experimental setup. We perform power analysis and run the reference implementation of Falcon (with the irrelevant delay operation used to separate different parts of leakages) on a Chipwhisperer-Lite, along with STM32F415 UFO target board (ARM Cortex-M4). The power traces of SamplerZ (see Algorithm 2) are collected by Picoscope 3206D at a real-time sampling rate of 1GSa/s, attached with a Mini-Circuits 1.9 MHz low-pass filter. In order to align the power traces, we ignore the traces in which a restart occurs, i.e. we collect the traces of SamplerZ with the acceptance rate $\approx \frac{\sigma_{\min}}{\sigma_{\max}+0.4}$ for all following power analysis experiments.

To roughly predict the existence of both two leakages, we severally collect $10,000$ traces with random centers c, standard deviation $\sigma' = 1.7$ and random seeds, then compute the Signal-to-Noise Ratio (SNR) for SamplerZ using the binary information of b and z^+, which are shown in Figure 3 and Figure 4.

Furthermore, we exploit different parts of power leakages to perform the Gaussian template attack [CRR03], which displays the magnitude of power leakage. During the profiling phase, we select the training sets where the number of traces is from 70 to $110,000$, to obtain different multivariate probability models. In the next attack phase, we collect 5659 traces (filtered in $10,000$ traces, evaluation set) and respectively perform the single-trace attack to calculate the classification accuracy for half Gaussian leakage (see Figure 5) and sign leakage (see Figure 6). In order to economize computational resources and measures, for every part of power leakages, we first choose the points (with the SNR values ≥ 0.001) of traces, then perform principal components analysis (PCA) before the first profiling and following attack phase, and then empirically select the first 65 principal components as points of interest (POIs) to launch Gaussian template attack.

Evaluation for half Gaussian leakage. Figure 3 illustrates the full SNR curve of SamplerZ and it is roughly divided into five sub-graphs for clarity. Clearly, the SNR values from x are much larger than that of other four parts, which is due to the FPR arithmetics. The zoomed-in graphs show that z and BerExp own comparably close SNR values for half Gaussian leakage. Similar cases hold for BaseSampler and "return". Figure 5 exhibits the classification accuracy with respect to the half Gaussian leakage. By consuming few traces for profiling, the leakages in x allow a perfect classification, i.e. 100% accuracy. Using the leakages in BaseSampler, the classification accuracy can achieve $\approx 98\%$ ($\approx 94\%$ accuracy in [GMRR22] with Chipwhisperer-Lite capture board). Moreover, solely using the leakages in z, the percentage of traces, which are correctly classified, is up to $\approx 83\%$. The leakages of BerExp result in the accuracy $\approx 86\%$. We finally evaluate the operation "return" and the corresponding classification accuracy is

$\approx 72\%$. To summarize, for the single-trace attack phase, the leakages in x are more significant than that of other four parts.



(a) The total SNR curve of SamplerZ for half Gaussian leakage.

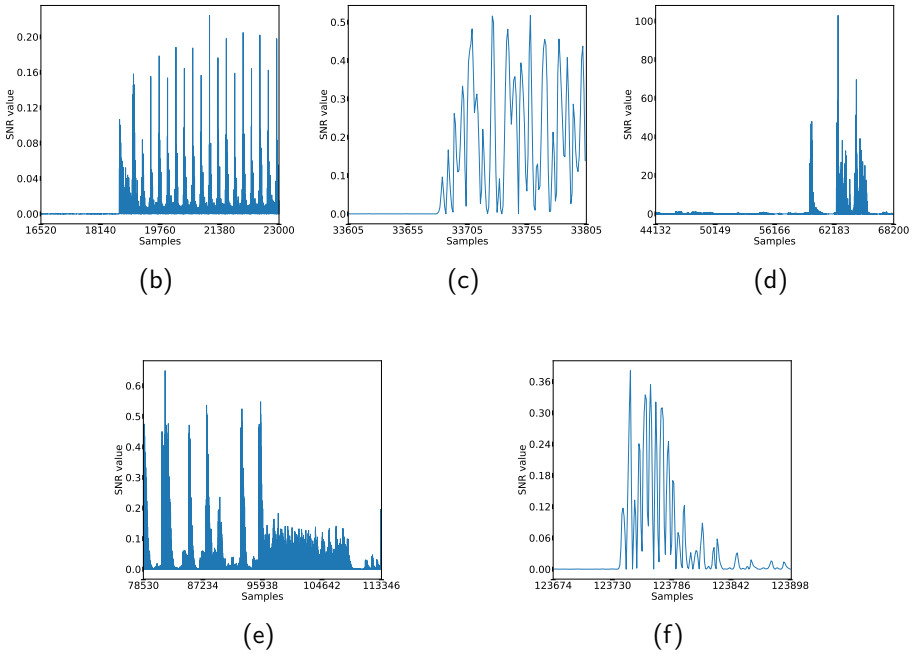
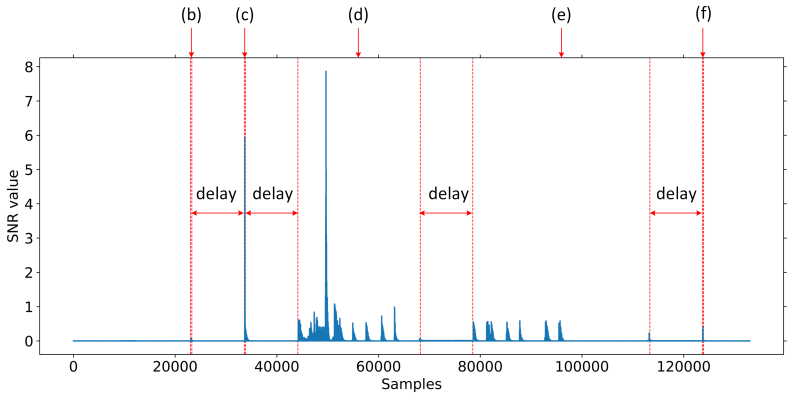


Fig. 3: SNR values of SamplerZ (see Algorithm 2) for half Gaussian leakage. The first graph (a) is the full curve and roughly split into the following five subgraphs (b), (c), (d), (e), (f), which are respectively for BaseSampler, the sign flip of z, the computation of x, the rejection sampling BerExp, the operation "return" (line 4, 6, 7, 8, 9 of Algorithm 2).



(a) The total SNR curve of SamplerZ for sign leakage.

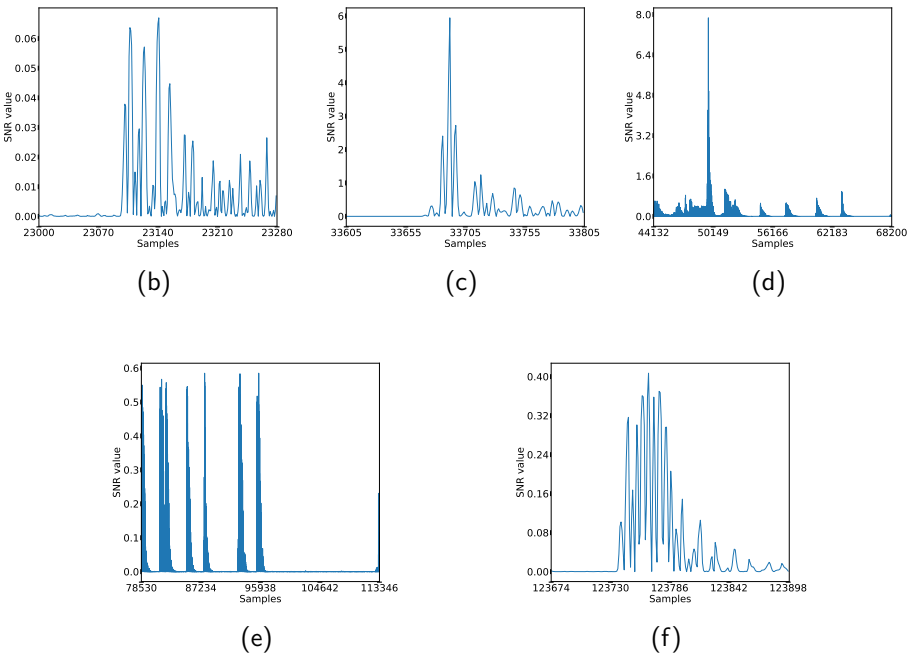


Fig. 4: SNR values of SamplerZ (see Algorithm 2) for sign leakage. The first graph (a) is the full curve and roughly split into the following sub-five graphs (b), (c), (d), (e), (f), which are respectively for the generation of b, the sign flip of z, the computation of x, the rejection sampling BerExp, the operation "return" (line 5, 6, 7, 8, 9 of Algorithm 2).

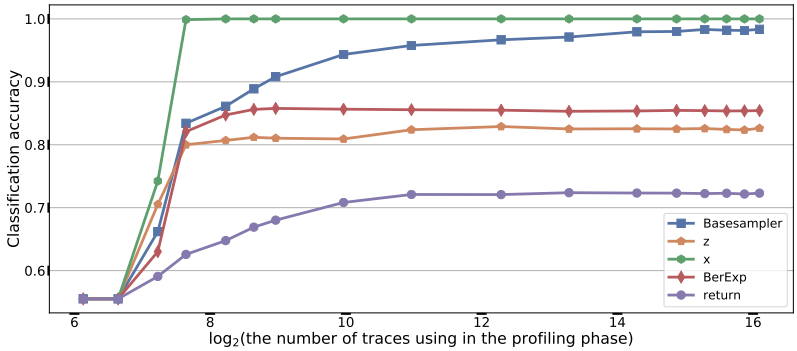


Fig. 5: The classification accuracy for SamplerZ's half Gaussian leakage.

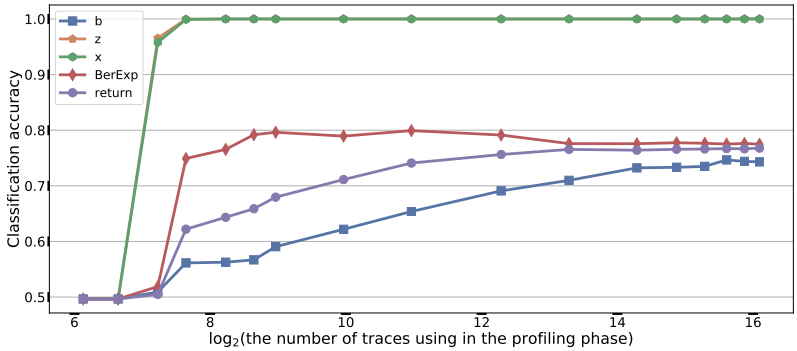


Fig. 6: The classification accuracy for SamplerZ's sign leakage.

Evaluation for sign leakage. The sign leakage was first identified and evaluated on an ARM Cortex-M4 STM32F407IGT6 microprocessor in [ZLYW23]. They also used Picoscope 3206D, equipped with 1.9 MHz filter, to collect traces. For similar template attacks, the accuracies are around 52%, 90% and 100% by respectively using these leakages in the generation of b, the sign flip of z and the computation of x. In our work, we choose a more precise and universal platform which is Chipwhisperer-Lite attached with STM32F415 UFO target board, to evaluate the sign leakage. We also compute the full SNR curve of SamplerZ with respect to the random bit b and then split it into five sub-graphs (see Figure 4). For the sign leakage, the SNR values of x are significantly smaller than that for half Gaussian leakage (see Figure 3). The sign flip of z and the computation of x have roughly close SNR values. Similar cases hold for BerExp and operation "return". By contrast, the generation of b admits the smallest SNR values by peak clusters, due to the only 1-bit Hamming weight difference. We also evaluate the classification accuracy for sign leakage by performing template attack. As shown

in Figure 6, spending few traces in profiling phase, the leakages in z or x can contribute to perfect classification (100% accuracy). Solely using the leakages in BerExp, the corresponding accuracy is around 78% for correctly determining b. Similarly, the leakages in operation "return" give almost 77% classification accuracy. By contrast, for the generation of b, the classification accuracy is at most $\approx 75\%$.

6 Countermeasures against Two Leakages

In this section, we propose effective and easy-to-implement countermeasures against both half Gaussian leakage and sign leakage on SamplerZ. Side-channel evaluations on Chipwhisperer show that our countermeasures reduce the classification accuracy of template attacks down to $\lesssim 58\%$ for the half Gaussian leakage and $\lesssim 62\%$ for the sign leakage (see comparison in Table 5). For such low accuracy, even 10 million traces is still far insufficient for practical key recovery as illustrated in [ZLYW23]. In addition, we evaluate the performance of our countermeasures on an Intel Core i5-1135G7 CPU. For the protected implementation, the overhead on signing is around 3.5× compared to the reference implementation of Falcon-512 (see more details in Table 6 and 7).

6.1 Countermeasures

Countermeasures against Half Gaussian Leakage As shown in Section 5.3, the half Gaussian leakage is more prominent than the sign leakage and is too indelible to mitigate by merely adding random noise. In [GMRR22], Guerreau et al. only proposed a countermeasure to reduce half Gaussian leakage in BaseSampler and validated the effectiveness on their experimental environment. However, to systematically mitigate the half Gaussian leakage, all sources of the leakage in SamplerZ should be taken into account, which complicates the protection.

Countermeasure for BaseSampler. While the countermeasure of [GMRR22] is claimed to have at most 1-bit leakage, advanced acquisitions and classification methods can still achieve a relatively high accuracy. We evaluate their countermeasure on Chipwhisperer-Lite along with STM32F415 UFO target board, use template attacks for classification, and finally achieve $\approx 97\%$ accuracy. Note that the leakage concentrates on the last subtraction of $[\![u < \mathsf{RCDT}[i]]\!]$. To this end, we propose a new countermeasure for this (see Algorithm 5) by using a simple trick to nearly eliminate the difference in Hamming weight. More specifically, $u_{\langle 2 \rangle}$ and $\mathsf{RCDT}[i]_{\langle 2 \rangle}$ are the 24 most significant bits of 72-bit random value u and entry $\mathsf{RCDT}[i]$, which are stored in 32-bit registers. In addition, the carry bit cc is related to the first two subtractions of $[\![u < \mathsf{RCDT}[i]]\!]$.

Similar to the countermeasure of [GMRR22], Algorithm 5 uses a different and well-selected constant value 0X1FFFFFF to let "overflow" happen when the last subtraction is negative. Thanks to this setting, the 8 most significant bits of 32-bit register are merely set to 2 when $u_{\langle 2 \rangle} < \mathsf{RCDT}[i]_{\langle 2 \rangle} + cc$ and 1 when $u_{\langle 2 \rangle} \geq \mathsf{RCDT}[i]_{\langle 2 \rangle} + cc$. Namely, our countermeasure encodes the "sign bits" with

$\{1, 2\}$ instead of $\{0, 255\}$ (for Falcon's reference implementation) or $\{0, 1\}$ (for the countermeasure in [GMRR22]). This simple trick removes the difference in Hamming weight and thus makes the classification of z^+ more difficult.

Algorithm 5: Protected BaseSampler at the last subtraction of $[\![u < \mathsf{RCDT}[i]]\!]$

Input: Two 24-bit integers $u_{\langle 2\rangle}$ and $\mathsf{RCDT}[i]_{\langle 2\rangle}$, carry bit cc
Output: 2 if $u_{\langle 2\rangle} < \mathsf{RCDT}[i]_{\langle 2\rangle} + cc$ and 1 if $u_{\langle 2\rangle} \geq \mathsf{RCDT}[i]_{\langle 2\rangle} + cc$

$bb \leftarrow$ 0X1FFFFFF
$\tilde{z}_j^+[i] \leftarrow ((bb - u_{\langle 2\rangle} + \mathsf{RCDT}[i]_{\langle 2\rangle} + cc) \gg 24)$ & 0X3
return $\tilde{z}_j^+[i]$ ▷ Table $\tilde{z}_j^+$ with $j \in \{0, 1, 2, 3\}$

Algorithm 6: Protected SamplerZ

Input: Center c and standard deviation $\sigma' \in [\sigma_{\min}, \sigma_{\max}]$ that are in FPR
Output: An integer $z \in \mathbb{Z}$ such that $z \sim D_{\mathbb{Z},\sigma',c}$

$(\tilde{t}[0], \cdots, \tilde{t}[15]) \leftarrow (2, 1, 1, 2, 2, 1, 1, 2, 2, 1, 1, 2, 2, 1, 1, 2)$
$c' \leftarrow c - \lfloor c \rfloor$, $ccs \leftarrow \sigma_{\min}/\sigma'$
$(\tilde{r}[0], \tilde{r}[1], \tilde{r}[2]) \leftarrow (0, c', 1 - c')$
while *(1)* **do**
  $t_{(4)} \xleftarrow{\$} \{0, 1\}^4$, $b' \leftarrow \tilde{t}[t_{(4)}]$
  $\tilde{z}_0^+ \leftarrow$ BaseSampler(), $\tilde{z}_1^+ \leftarrow$ BaseSampler()
  $\tilde{z}_2^+ \leftarrow$ BaseSampler(), $\tilde{z}_3^+ \leftarrow$ BaseSampler()
    ▷ New BaseSampler with countermeasure of Algorithm 5
  for $i = 0, \cdots, 3$ **do**
    $\tilde{z}^+[i] \leftarrow \lfloor c \rfloor + ((\tilde{z}_i^+[17]\ \&\ 1) + \cdots + (\tilde{z}_i^+[0]\ \&\ 1))$
    $\tilde{z}^+[i] \leftarrow 18 + 2 * \lfloor c \rfloor - \tilde{z}^+[i]$
  end for
  $t_{(2)} \xleftarrow{\$} \{0, 1\}^2$
  for $i = 0, \cdots, 18$ **do**
    $(\tilde{z}[0][i], \tilde{z}[1][i], \tilde{z}[2][i]) \leftarrow (0, \lfloor c \rfloor - i, 1 + \lfloor c \rfloor + i)$
  end for
  for $i = 0, \cdots, 18$ **do**
    $\tilde{x}[0][i] \leftarrow 0$, $\tilde{x}[1][i] \leftarrow \frac{(\tilde{z}_t^+[i] + \tilde{r}[1])^2}{2\sigma'^2} - \tilde{x}_2[i]$
    $\tilde{x}[2][i] \leftarrow \frac{(\tilde{z}_t^+[i] + \tilde{r}[2])^2}{2\sigma'^2} - \tilde{x}_2[i]$
      ▷ Precomputed entries $\tilde{z}_t^+[i] = i$ and $\tilde{x}_2[i] = \frac{i^2}{2\sigma_{\max}^2}$
  end for
  if BerExp$(\tilde{x}, ccs, \tilde{z}^+, t_{(2)}, \lfloor c \rfloor, b') = 1$ **then**
    ▷ Protected BerExp in Algorithm 7
    return $\tilde{z}[b'][\tilde{z}^+[t_{(2)}] - \lfloor c \rfloor]$
  end if
end while

Countermeasure for SamplerZ. We present a practical countermeasure for SamplerZ in Algorithm 6 that greatly mitigates the half Gaussian leakage. In

Algorithm 6, we continuously sample 4 half Gaussian tables ($\tilde{z}_0^+$, $\tilde{z}_1^+$, $\tilde{z}_2^+$, $\tilde{z}_3^+$) by calling protected BaseSampler. This roughly averages the leakage in this BaseSampler and thus lowers the classification accuracy. Less calls of this new BaseSampler with the countermeasure of Algorithm 5 could lead to higher SNR values and classification accuracy. As per Algorithm 5, the entries of $\tilde{z}_i^+$ are all encoded into $\{1, 2\}$, then we compute the increments of z^+, add the integer center $\lfloor c \rfloor$ and store them in $\tilde{z}^+$. For each sampling, we use different random integers $\lfloor c \rfloor$ to mask partial information of z^+, instead of generating new randomness. Similarly, we randomly generate 2-bit $t_{(2)}$ as the index for choosing one out of four values in table $\tilde{z}^+$.

During the sign flip of z and computation of x, we traverse all possible $(z^+, b') \in \{0, 1, \cdots, 18\} \times \{1, 2\}$ and store the intermediate results in $\tilde{z}[b']$ and $\tilde{x}[b']$. In the computation of $\tilde{x}$, both $\tilde{z}_t^+$ and $\tilde{x}_2$ involve the transformation from integer to FPR, and can be precomputed to improve performance. As for the rejection sampling BerExp, we apply similar tricks to avoid the half Gaussian leakage. More concretely, we compute all 19 possible values of z^+ in the protected BerExp (see Algorithm 7), including the decomposition of x and computation of z', and severally store them to table $\tilde{s}'[b']$, $\tilde{r}'[b']$ and $\tilde{z}'[b']$. We retrieve the proper item in table $\tilde{z}'$ as per b' and the value $[\![z^+ \leftarrow \tilde{z}^+[t_{(2)}] - \lfloor c \rfloor]\!]$, and perform lazy Bernoulli sampling. At the end of Algorithm 6, we use the same method to obtain the value $z + \lfloor c \rfloor$ in table $\tilde{z}$. Similarly, adding the random integer $\lfloor c \rfloor$ also reduces the half Gaussian leakage in operation "return". To sum up, we transfer the dominating half Gaussian leakage to the end of Algorithm 6 and Algorithm 7, which carries less information of z^+.

Algorithm 7: Protected BerExp

Input: Table $\tilde{x}$ and value $ccs \geq 0$ in FPR, table $\tilde{z}^+$, index $t_{(2)}$, integer $\lfloor c \rfloor$ and b'

Output: 1 with probability $\approx ccs \cdot \exp(-\tilde{x}[b'][\tilde{z}^+[t_{(2)}] - \lfloor c \rfloor])$

for $i = 1, \cdots, 2$ **do** ▷ The traversal for sign $b' \in \{1, 2\}$
  for $j = 0, \cdots, 18$ **do** ▷ For $z^+ \in \{0, \cdots, 18\}$
    $\tilde{s}'[i][j] \leftarrow \lfloor \tilde{x}[i][j] / \ln(2) \rfloor$
    $\tilde{r}'[i][j] \leftarrow \tilde{x}[i][j] - \tilde{s}'[i][j] \cdot \ln(2)$
    $\tilde{s}'[i][j] \leftarrow \min(\tilde{s}'[i][j], 63)$
    $\tilde{z}'[i-1][j] \leftarrow ((\mathsf{ApproxExp}(\tilde{r}'[i][j], ccs) \ll 1) - 1) \gg \tilde{s}'[i][j]$
  end for
end for
$i \leftarrow 64$, $z'' \leftarrow \tilde{z}'[b' \gg 1][\tilde{z}^+[t_{(2)}] - \lfloor c \rfloor]$
do
  $i \leftarrow i - 8$, $y \xleftarrow{\$} \{0, 1\}^8$
  $w \leftarrow y - ((z'' \gg i)\ \&\ \mathsf{0xFF})$
while $((w = 0)\ and\ (i > 0))$
return $[\![w < 0]\!]$

Countermeasures against Sign Leakage In [ZLYW23], Zhang et al. presented a countermeasure to mitigate the sign leakage from the generation of b, the sign flip of z and the computation of x in SamplerZ. Their countermeasure achieves the classification accuracy $\approx 52\%$ (evaluated on ARM Cortex-M4 STM32F407IGT6 board). We present a more complete countermeasure taking the leakage sources of BerExp and "return" operation into account.

Countermeasure for SamplerZ. Our protected SamplerZ is described in Algorithm 6. It adopts the similar idea of the sign leakage countermeasure in [ZLYW23], i.e. encoding $b \in \{0,1\}$ into $b' \in \{1,2\}$ to eliminate the Hamming weight gap. Furthermore, our countermeasure can simultaneously mitigate both sign leakage and half Gaussian leakage. More precisely, for the generation of b', we use 4-bit random value $t_{(4)}$, as an index, to read the empirically selected look-up table $\tilde{t}$ with $2^4 = 16$ entries in $\{1,2\}$ and then uniformly map the sign from $b \in \{0,1\}$ to $b' \in \{1,2\}$. With regard to the sign flip of z, we apply a similar trick to sample the value of z. Specifically, the table $\tilde{z}[b']$ is computed by $[\![-z^+ + \lfloor c \rfloor]\!]$ when $b' = 1$ or $[\![1 + z^+ + \lfloor c \rfloor]\!]$ when $b' = 2$, and the entry is selected as per $b' \in \{1,2\}$. Moreover, the sign information of z can also be reduced by adding integer $\lfloor c \rfloor$.

In the computation of x, the main source of the sign leakage is $[\![(z-r)^2]\!]$. As shown in [ZLYW23], the leakage in this part can be reduced by computing $[\![(z-c')^2 = (z^+ + \tilde{r}[b'])^2]\!]$ for all two cases of $b' = 1$ and $b' = 2$. However, we observe that $[\![\tilde{x}[i] \leftarrow \frac{(i+\tilde{r}[b])^2}{2\sigma'^2} - \frac{i^2}{2\sigma_{\max}^2}]\!]$ with $i \in \{0, \cdots, 18\}$ still exists significant sign leakage accumulated by the traversal of z^+. The classification accuracy using these leakages (evaluated on Chipwhisperer) can be up to $\approx 75\%$. To this end, we also traverse all possible $b' \in \{1,2\}$ to compute $\tilde{x}[b']$ and strip the sign leakage in computing x. Similarly, for BerExp, we perform the decomposition of x and the computation of z' by using all possible values of b' and store in table $\tilde{s}'[b']$, $\tilde{r}'[b']$ and $\tilde{z}'[b']$ (see Algorithm 7). As a result, we confine the major sign leakage to the mapping based on b', lazy Bernoulli sampling in Algorithm 7, and operation "return" of Algorithm 6, which yields a low classification accuracy.

6.2 Practical Evaluations

Based on the reference implementation of Falcon, we implement our countermeasures in portable C. This section reports on the practical evaluations on our countermeasures, including side-channel security (Table 5 for accuracy comparison) performance evaluations (Table 6 and 7 for performance comparison).

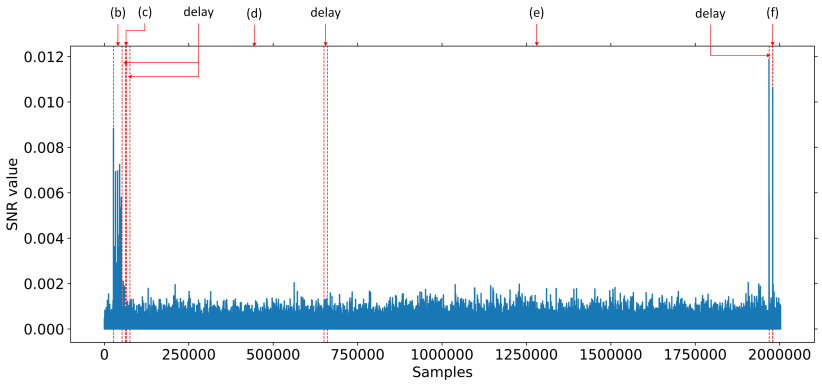
Security Evaluations Experimental setup. Our side-channel security evaluations are also performed on Chipwhisperer-Lite with STM32F415 UFO target board (ARM Cortex-M4) and experimental configuration is almost the same with that in Section 5.3. We collect $10,000$ traces to compute SNR values of protected SamplerZ (see Algorithm 6), which are shown in Figure 7 (for half Gaussian leakage) and Figure 8 (for sign leakage). Furthermore, we also demonstrate the Gaussian template attack [CRR03] against the protected SamplerZ, by

Table 5: Accuracy comparison with protected SamplerZ. The item "A/B" notes the accuracy for unprotected / protected operation

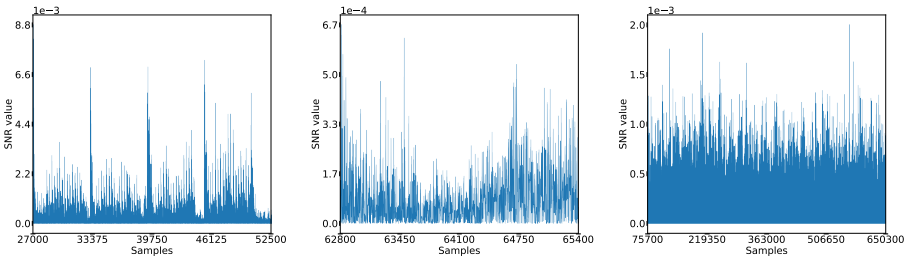
Operation	Half Gaussian leakage
BaseSampler / protected BaseSampler	98% / 57%
sign flip of z / $\tilde{z}$	83% / 50%
computation x / $\tilde{x}$	100% / 51%
BerExp / protected BerExp	86% / 54%
operation "return"	72% / 58%
Operation	**Sign leakage**
sign b / b'	75% / 62%
sign flip of z / $\tilde{z}$	100% / 51%
computation x / $\tilde{x}$	100% / 57%
BerExp / protected BerExp	78% / 57%
operation "return"	77% / 54%

empirically choosing the first 65 principal components. More precisely, we collect traces (from 70 to 110,000) for profiling and then use 5767 traces (filtered in 10,000 traces) to repeat single-trace attack to obtain the classification accuracy of protected SamplerZ for half Gaussian leakage (see Figure 9) and sign leakage (see Figure 10).

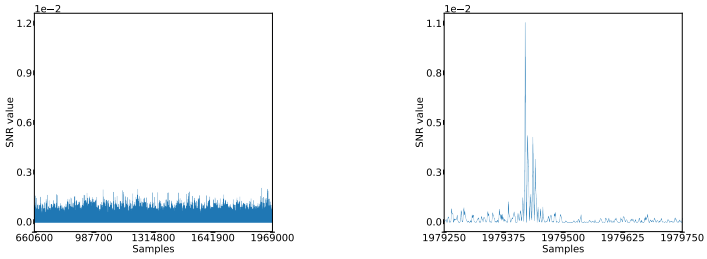
Evaluations for half Gaussian leakage. As shown in Figure 7, it still exists significant peak clusters at the begin and end of the SNR values curve. Namely, the protected BaseSampler (line 6-12, Algorithm 6) and operation "return" (line 21, Algorithm 6) can be slightly detected by half Gaussian leakage on Chipwhisperer. In addition, the rest of components in protected BaseSampler are hardly identified in the context of our experimental setup. However, the SNR value of Algorithm 6 for half Gaussian leakage is much lower than that of SamplerZ (see Figure 3). Since the true value of z should be returned, this part of leakages are difficult to experimentally remove. Therefore, by using the growing number of profiling traces, the classification accuracy for half Gaussian leakage is at most $\approx 58\%$ with different sources of leakages, as depicted by Figure 9. As per the experimental results of [ZLYW23], such imperfect accuracy for classifying $z^+ = 0$ or not leads to the dramatically increasing number of required traces. A practical key recovery can still be performed with 10 million signatures when the accuracy is 65% (see Figure 5 of [ZLYW23]). Therefore, equipped with our countermeasures, the required traces leading to full key recovery with half Gaussian leakage are much more than 10 million, which can be considered as infeasible for many real-world applications. As mentioned in [ZLYW23], one can set a counter for the maximum time of signing.



(a) The total SNR curve of protected SamplerZ for half Gaussian leakage.

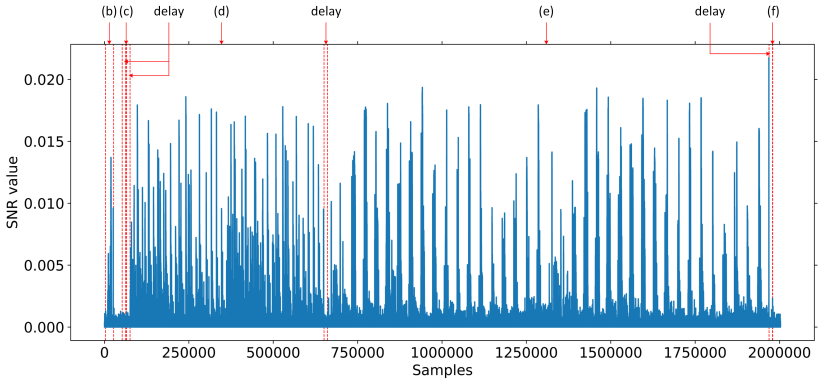


(b) (c) (d)

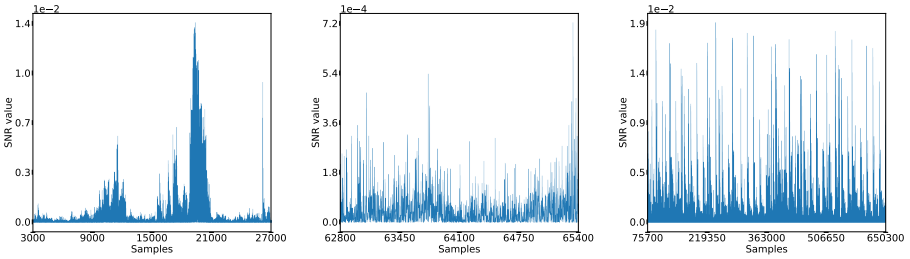


(e) (f)

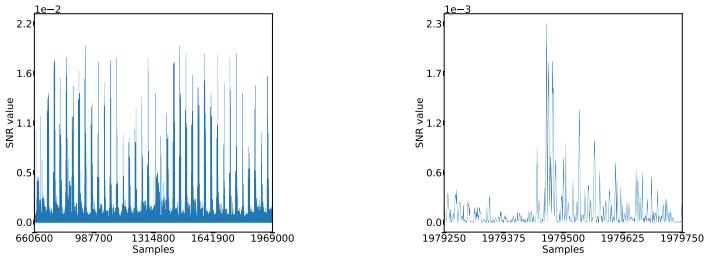
Fig. 7: SNR values of protected SamplerZ (see Algorithm 6) for half Gaussian leakage. The first graph (a) is the full curve and roughly split into the following five sub-graphs (b), (c), (d), (e), (f), which are respectively for SNR values include protected BaseSampler, the sign flip of $\tilde{z}$, the computation of $\tilde{x}$, the rejection sampling protected BerExp, the operation "return" (line 6-12, 13-15, 16-19, 20, 21 of Algorithm 6).



(a) The total SNR curve of protected SamplerZ for sign leakage.



(b) (c) (d)



(e) (f)

Fig. 8: SNR values of protected SamplerZ (see Algorithm 6) for sign leakage. The first graph (a) is the full curve and roughly split into the following five sub-graphs (b), (c), (d), (e), (f), which are respectively for SNR values include the sampling of b', the sign flip of $\tilde{z}$, the computation of $\tilde{x}$, the rejection sampling protected BerExp, the operation "return" (line 5, 13-15, 16-19, 20, 21 of Algorithm 6).

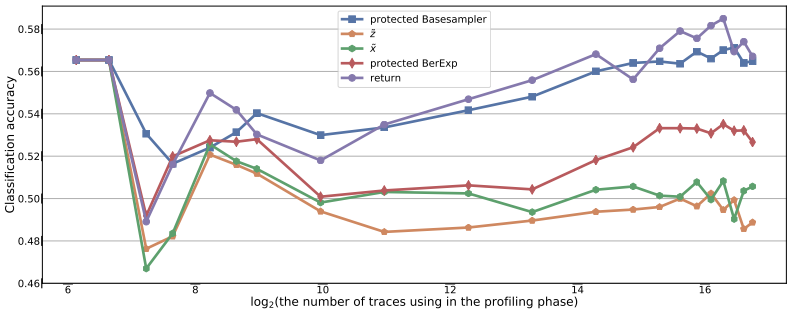


Fig. 9: The classification accuracy of protected SamplerZ for half Gaussian leakage.

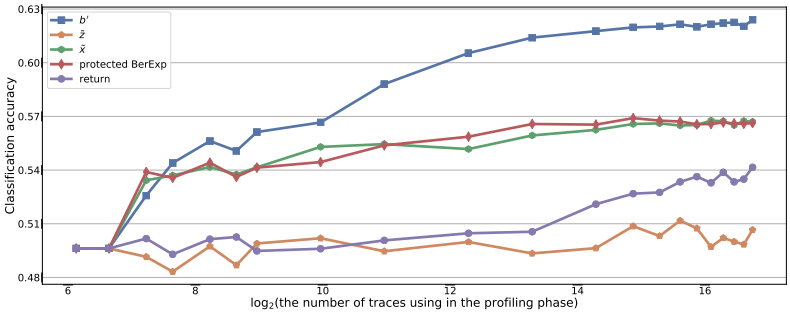


Fig. 10: The classification accuracy of protected SamplerZ for sign leakage.

Evaluations for sign leakage. For the SNR values curve in Figure 8, it still exists one single distinct peak clusters in the beginning and implies the sign leakage still persists in the generation of b' in the protected SamplerZ (line 5, Algorithm 6) even if there is no difference in Hamming weight. Based on our setup, the sign leakage in the computation of $\tilde{x}$ (line 16-19, Algorithm 6) and the rejection sampling protected BerExp (line 20, Algorithm 6) can be weakly detected as shown in the Figure 8. For operation "return" (line 21, Algorithm 6), this leakage still exists in spite of smaller SNR values. In addition, the rest of parts in Algorithm 6 are not strongly related to the sign leakage. By contrast, the SNR values of Algorithm 6 are much lower than that of SamplerZ (see Figure 4). Furthermore, we calculate the classification accuracy of the sign leakage as illustrated in Figure 10. More specifically, we can control the accuracy is increasing up to $\approx 62\%$ by using different sources of sign leakages and number of required traces for profiling. As evaluated in [ZLYW23], the inaccurate classification for the sign leakage also results in sharply increasing the number of required traces when the accuracy is $< 65\%$ (see Figure 12 of [ZLYW23]). Therefore, configured

with our countermeasures, the number of required signatures to fully recover the secret with sign leakage is much more than 10 million, which is infeasible for many real-world usecases.

Performance Evaluations The performance comparison is evaluated on an Intel Core i5-1135G7 CPU clocked at 2.4 GHz with hyper-threading disable. Compilation is executed by `Clang-10.0.0` with optimization cflags `-O0`. We employ the Falcon's benchmarking tool to measure clock time and access system counter to provide cycle counts.

We first test the performance of protected SamplerZ (Algorithm 6) and compare with the reference implementation of Falcon. As shown in Table 6, the bottlenecks stem from the calculations of $\tilde{x}$ and the protected BerExp. Our protected SamplerZ comes with about 6.0× overhead measured by clock time and 19.0× overhead for cycle counts.

Table 6: Performance comparison between SamplerZ and protected SamplerZ

	Algorithm	Clock Time (ns)	Cycles
SamplerZ	Total	2400	1299
	BaseSampler in the line 4	292	287
	generation of b in the line 5	142	18
	sign flip of z in the line 6	138	25
	calculation of x in the line 7	494	306
	BerExp in the line 8	573	428
	"return" in the line 9	138	21
Protected SamplerZ	Total	14428	24744
	sampling of b' in the line 5	146	25
	new BaseSampler in the line 6-12	745	1186
	sign flip of $\tilde{z}$ in the line 13-15	167	76
	computation of $\tilde{x}$ in the line 16-19	4132	7376
	protected BerExp in the line 20	8228	15731
	"return" in the line 21	142	26

Furthermore, in Table 7, we provide benchmarks for signing in the dynamic mode and the tree mode. We apply our countermeasures on Gaussian samplers to apportion the overheads on the whole signing process of Falcon. For Falcon-512 (Falcon-1024), dynamic signing has about 3.7× (3.3×) overhead measured by clock time and 3.5× (3.4×) overhead measured by cycle counts.

Table 7: Performance comparison with the signing dynamic (SD) and signing tree (ST) of Falcon's reference implementation

Claimed Security	Falcon-512		Falcon-1024	
	SD	ST	SD	ST
Unprotected (ms)	6.7	3.1	14.8	6.5
Protected (ms)	24.5	20.5	49.4	41.0
Vs.	3.7×	6.6×	3.3×	6.3×
Unprotected (Mcycles)	16.6	7.3	35.6	15.7
Protected (Mcycles)	58.7	49.9	119.6	99.4
Vs.	3.5×	6.8×	3.4×	6.3×

Acknowledgments

We would like to thank the anonymous reviewers for their useful suggestions and comments. Xiuhan Lin, Yang Yu and Shiduo Zhang are supported by the National Key R&D Program of China (2023YFA1009500) and the National Natural Science Foundation of China (12441104). Weijia Wang's work has been supported by the National Natural Science Foundation of China (Grant No. 62372273) and the Key Research and Development Program of Shandong Province, China (Grant No. 2024ZLGX05).

References

BBE+18. Gilles Barthe, Sonia Belaïd, Thomas Espitau, Pierre-Alain Fouque, Benjamin Grégoire, Mélissa Rossi, and Mehdi Tibouchi. Masking the GLP lattice-based signature scheme at any order. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 354–384, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Cham, Switzerland.

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. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2147–2164, London, UK, November 11–15, 2019. ACM Press.

BDE+18. Jonathan Bootle, Claire Delaplace, Thomas Espitau, Pierre-Alain Fouque, and Mehdi Tibouchi. LWE without modular reduction and improved side-channel attacks against BLISS. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part I*, volume 11272 of *LNCS*, pages 494–524, Brisbane, Queensland, Australia, December 2–6, 2018. Springer, Cham, Switzerland.

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. In Benedikt Gierlichs and Axel Y. Poschmann, editors, *CHES 2016*, volume 9813 of *LNCS*, pages 323–345, Santa Barbara, CA, USA, August 17–19, 2016. Springer Berlin Heidelberg, Germany.

CRR03. Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In Burton S. Kaliski, Jr., Çetin Kaya Koç, and Christof Paar, editors, *CHES 2002*, volume 2523 of *LNCS*, pages 13–28, Redwood Shores, CA, USA, August 13–15, 2003. Springer Berlin Heidelberg, Germany.

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

DLP14. Léo Ducas, Vadim Lyubashevsky, and Thomas Prest. Efficient identity-based encryption over NTRU lattices. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 22–41, Kaoshiung, Taiwan, R.O.C., December 7–11, 2014. Springer Berlin Heidelberg, Germany.

DP16. Léo Ducas and Thomas Prest. Fast fourier orthogonalization. In Sergei A. Abramov, Eugene V. Zima, and Xiao-Shan Gao, editors, *Proceedings of the ACM on International Symposium on Symbolic and Algebraic Computation, ISSAC 2016, Waterloo, ON, Canada, July 19-22, 2016*, pages 191–198. ACM, 2016.

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. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1857–1874, Dallas, TX, USA, October 31 – November 2, 2017. ACM Press.

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. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part III*, volume 12107 of *LNCS*, pages 34–63, Zagreb, Croatia, May 10–14, 2020. Springer, Cham, Switzerland.

GHN06. Nicolas Gama, Nick Howgrave-Graham, and Phong Q. Nguyen. Symplectic lattice reduction and NTRU. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 233–253, St. Petersburg, Russia, May 28 – June 1, 2006. Springer Berlin Heidelberg, Germany.

GPV08. Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 197–206, Victoria, BC, Canada, May 17–20, 2008. ACM Press.

GR19. François Gérard and Mélissa Rossi. An efficient and provable masked implementation of qtesla. In Sonia Belaïd and Tim Güneysu, editors, *Smart Card Research and Advanced Applications - 18th International Conference, CARDIS 2019, Prague, Czech Republic, November 11-13, 2019, Revised Selected Papers*, volume 11833 of *Lecture Notes in Computer Science*, pages 74–91. Springer, 2019.

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

HPRR20. James Howe, Thomas Prest, Thomas Ricosset, and Mélissa Rossi. Isochronous gaussian sampling: From inception to implementation. In Jintai Ding and Jean-Pierre Tillich, editors, *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020*, pages 53–71, Paris, France, April 15–17, 2020. Springer, Cham, Switzerland.

HBD+22. Andreas Hülsing, Daniel J. Bernstein, Christoph Dobraunig, Maria Eichlseder, Scott Fluhrer, Stefan-Lukas Gazdag, Panos Kampanakis, Stefan Kölbl, Tanja Lange, Martin M. Lauridsen, Florian Mendel, Ruben Niederhagen, Christian Rechberger, Joost Rijneveld, Peter Schwabe, Jean-Philippe Aumasson, Bas Westerbaan, and Ward Beullens. SPHINCS+. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

KA21. Emre Karabulut and Aydin Aysu. FALCON down: Breaking FALCON post-quantum signature scheme through side-channel attacks. In *58th ACM/IEEE Design Automation Conference, DAC 2021, San Francisco, CA, USA, December 5-9, 2021*, pages 691–696. IEEE, 2021.

KRR+18. Angshuman Karmakar, Sujoy Sinha Roy, Oscar Reparaz, Frederik Vercauteren, and Ingrid Verbauwhede. Constant-time discrete gaussian sampling. *IEEE Trans. Computers*, 67(11):1561–1571, 2018.

LSZ+24. Xiuhan Lin, Moeto Suzuki, Shiduo Zhang, Thomas Espitau, Yang Yu, Mehdi Tibouchi, and Masayuki Abe. Cryptanalysis of the Peregrine lattice-based signature scheme. In Qiang Tang and Vanessa Teague, editors, *PKC 2024, Part I*, volume 14601 of *LNCS*, pages 387–412, Sydney, NSW, Australia, April 15–17, 2024. Springer, Cham, Switzerland.

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.

MGTF19. Vincent Migliore, Benoît Gérard, Mehdi Tibouchi, and Pierre-Alain Fouque. Masking Dilithium - efficient implementation and side-channel evaluation. In Robert H. Deng, Valérie Gauthier-Umaña, Martín Ochoa, and Moti Yung, editors, *ACNS 19International Conference on Applied Cryptography and Network Security*, volume 11464 of *LNCS*, pages 344–362, Bogota, Colombia, June 5–7, 2019. Springer, Cham, Switzerland.

PBY17. Peter Pessl, Leon Groot Bruinderink, and Yuval Yarom. To BLISS-B or not to be: Attacking strongSwan's implementation of post-quantum signatures. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1843–1855, Dallas, TX, USA, October 31 – November 2, 2017. ACM Press.

Pre23. Thomas Prest. A key-recovery attack against mitaka in the t-probing model. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *PKC 2023, Part I*, volume 13940 of *LNCS*, pages 205–220, Atlanta, GA, USA, May 7–10, 2023. Springer, Cham, Switzerland.

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. avail-

able at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

SAB+22. Peter Schwabe, Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, Damien Stehlé, and Jintai Ding. CRYSTALS-KYBER. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

ZLYW23. Shiduo Zhang, Xiuhan Lin, Yang Yu, and Weijia Wang. Improved power analysis attacks on falcon. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part IV*, volume 14007 of *LNCS*, pages 565–595, Lyon, France, April 23–27, 2023. Springer, Cham, Switzerland.

ZSS20. 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.