Breaking HuFu with 0 Leakage
A Side-Channel Analysis

Julien Devevey[1], Morgane Guerreau[2], Thomas Legavre[1,3,4], Ange Martinelli[1], and Thomas Ricosset[3]

[1] ANSSI, Paris, France
{julien.devevey,ange.martinelli}@ssi.gouv.fr
[2] CryptoNext Security, Paris, France
morgane.guerreau@cryptonext-security.com
[3] Thales, Gennevilliers, France
{thomas.legavre,thomas.ricosset}@thalesgroup.com
[4] Sorbonne Université, CNRS, LIP6, Paris, France

Abstract. HuFu is an unstructured lattice-based signature scheme proposed during the NIST PQC standardization process. In this work, we present a side-channel analysis of HuFu's reference implementation.
We first exploit the multiplications involving its two main secret matrices, recovering approximately half of their entries through a non-profiled power analysis with a few hundred traces. Using these coefficients, we reduce the dimension of the underlying LWE problem, enabling full secret key recovery with calls to a small block-sized BKZ.
To mitigate this attack, we propose a countermeasure that replaces sensitive computations involving a secret matrix with equivalent operations derived solely from public elements, eliminating approximately half of the identified leakage and rendering the attack unfeasible.
Finally, we perform a non-profiled power analysis targeting HuFu's Gaussian sampling procedure, recovering around 75% of the remaining secret matrix's entries in a few hundred traces. While full key recovery remains computationally intensive, we demonstrate that partial knowledge of the secret significantly improves the efficiency of signature forgery.

1 Introduction

Unstructured lattices vs. Structured lattices. Lattice-based cryptography has become a cornerstone of post-quantum cryptography, offering strong protection against both classical and quantum adversaries. Its security is rooted in the hardness of well-studied mathematical problems, such as the Short Integer Solution (SIS) [Ajt96] and Learning With Errors (LWE) [Reg05] problems, which remain difficult even for quantum computers.

Algebraic variants of these problems allow for more efficient implementations and shorter key sizes in cryptographic primitives like Key Exchange Mechanisms (KEMs) and signature schemes. Notable examples include the standardized ML-KEM [NIS24b] and ML-DSA [NIS24a]. However, these structured vari-

ants introduce additional algebraic properties that can be exploited by adversaries. For example, attacks on Ideal-SVP provide improvements over generic attacks [CDW17,PHS19,BR20,BLNR22]. While no comparable attacks have been identified for "Module" variants, a direct reduction from generic problems has not been established.

In contrast, primitives based on unstructured lattices, though less efficient and compact, provide more conservative security guarantees. Studying these unstructured schemes provides valuable insight into the trade-offs between structured and unstructured lattices, offering a clearer understanding of their respective strengths and vulnerabilities.

Side-Channel Analysis. In both its structured and unstructured variants, lattice-based cryptography is often technical to implement and manipulates a lot of secret dependent variables. These variables are a perfect target for side-channel attacks (*SCA*), which exploit information leakages through physical data captured during the execution of the algorithm on a device. Nowadays it is the main threat to cryptographic algorithms and should be taken into account in the design phase of any modern algorithm. Indeed, while generic protections against SCA exist, they come at a non-negligible cost, which could most of the time be lowered by carefully choosing the operations performed by the algorithm. However, while being devastating on most implementations, exploiting leakage information to mount an effective cryptanalysis is often not straightforward.

The HuFu Signature Scheme. In this work, we propose to further explore SCA against unstructured schemes by targeting HuFu. The HuFu [YJL+23] signature scheme is a concrete instance of the GPV [GPV08] hash-and-sign framework, built upon the MP12 [MP12] trapdoor construction and the Peikert Gaussian sampler [Pei10]. It was a candidate in the first round of the additional signature track of the NIST PQC standardization process. Its security relies on the hardness of unstructured lattice problems, which makes it a more conservative choice security-wise compared to schemes like the NIST-standardized ML-DSA, which rely on module lattice problems. The MP12 trapdoor construction facilitates the creation of lattices with a *good* basis (serving as the secret key) and a *bad* basis (serving as the public key). The Peikert sampler then utilizes the good basis to efficiently generate Gaussian samples from the lattice, ensuring these points lie relatively close to any specified target point in the ambient space.

Although HuFu is a signature scheme, its underlying building blocks and their implementation have applications that extend well beyond digital signatures. The MP12 trapdoor construction forms the backbone of advanced cryptographic schemes, such as identity-based encryption as demonstrated in [ABB10], group signatures [dPLS18], anonymous credentials [JRS23], blind signatures [JS24]. On the other hand, the Peikert Gaussian sampler serves as a fundamental component in a wide variety of trapdoor-based lattice applications. Consequently, analyzing the side-channel vulnerabilities of HuFu offers valuable insights into the broader field of lattice-based cryptography.

HuFu was submitted to NIST's additional round of signature schemes but was cut from the competition from Round 2 onward. Issues about the signature encoding were raised[1] but quickly patched by the HuFu team.

Related Works. With the NIST standardization process coming to an end, significant attention has been given to implementation issues in lattice-based cryptography. Similar to this work, sign and zero values are common targets in horizontal attacks, as seen in [GMRR22,ZLYW23,LZY+25] for Falcon, in [KAA21,BVC+23] for ML-DSA, in [BVCV24] for ML-KEM, and in [GR24] for the more recent Hawk scheme. Exploiting such partial knowledge requires an additional cryptanalysis step, and a generic framework for evaluating the remaining strength of cryptosystems in the presence of side-channel *hints* was introduced in [DDGR20] and later improved in [DGHK23,MN23]. However, side-channel attacks on lattice-based schemes typically leverage their algebraic structure, making them difficult to extend to unstructured lattices. Nonetheless, such attacks on unstructured lattice schemes, though less common, do exist, notably against FrodoKEM [KH18,BFM+18].

1.1 Contributions

Our contributions consist of two attack paths (key recovery and signature forgery) involving side-channel analysis and lattice reduction. These attacks are illustrated in Figure 1 and can be summarized as follows[2]:

Side-Channel Analysis of HuFu. We present in Section 3 a non-profiled power analysis attack on the HuFu signature scheme. This attack targets the multiplications between the secret matrices $\mathbf{S}$, $\mathbf{E}$, and the ephemeral vector $\mathbf{z}$. Using this method, we successfully recover almost all coefficients in $\mathbf{S}$ and $\mathbf{E}$ that are equal to 0.

Key Recovery. By leveraging the recovered zero coefficients in both $\mathbf{S}$ and $\mathbf{E}$, we achieve full secret key recovery in Section 4. Each column of $\mathbf{S}$ corresponds to a secret vector $\mathbf{s}$ in an LWE instance of the form $\hat{\mathbf{A}}\mathbf{s} + \mathbf{e}$, where $\mathbf{e}$ is the corresponding column of $\mathbf{E}$. By mapping known coefficients in $\mathbf{s}$ or $\mathbf{e}$ to lower-dimensional hyperplanes containing the LWE solution, we iteratively intersect the ambient lattice with these hyperplanes. This reduces the dimension of the unknown part of $\mathbf{s}$ until it becomes solvable via lattice reduction.

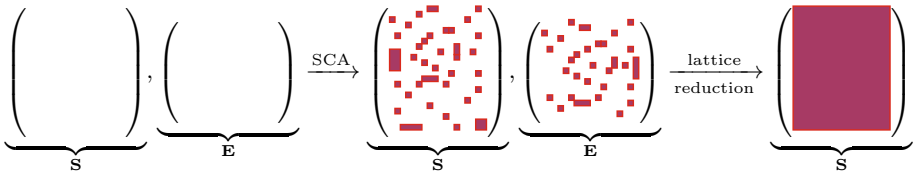
Partial Countermeasure and Enhanced SCA. We propose in Section 5 a simple countermeasure to eliminate leakage from the secret matrix $\mathbf{E}$. Consequently, we only have partial information about $\mathbf{S}$, preventing us from reproducing the initial attack that relied on coefficients from both $\mathbf{S}$ and $\mathbf{E}$. In response, we develop a deeper side-channel attack targeting the ephemeral vector $\mathbf{z}$, enabling us to recover additional coefficients in $\mathbf{S}$.

[1] `https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/Hq-wRFDbIaU/m/iLZctTiLAgAJ`

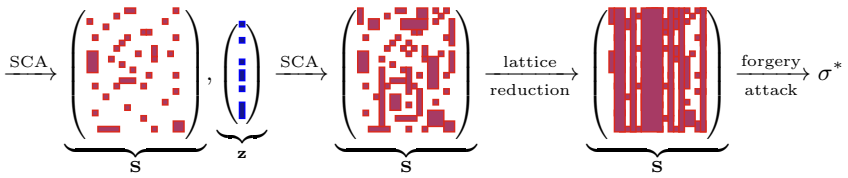
[2] All our code is available here: `https://github.com/mB64s53wFvP7637M/KF7ns9y5bf`

Signature Forgery. Using the previous key recovery attack to reconstruct a subset of the columns of $\mathbf{S}$, we demonstrate in Section 6 how this knowledge weakens the forgery security of the scheme, up to a breaking point.

Broader Applicability. Although our attacks are demonstrated on HuFu, the side-channel techniques apply to any matrix-vector multiplication implemented similarly to the HuFu reference code, where the matrix is a long-term secret with many zero entries and the vector is a (Gaussian) ephemeral secret. This is particularly relevant in lattice-based constructions using the [MP12] strategy (e.g., [DM14,CGM19,YJW23]) to sample Gaussian (inhomogeneous) SIS solutions for a public matrix $\mathbf{A}$. Specifically, this applies when leveraging a trapdoor for $\Lambda^{\perp}(\mathbf{G}) := \{\mathbf{z} \in \mathbb{Z}^m \,|\, \mathbf{Gz} = \mathbf{0}\}$, where $\mathbf{G} = \mathbf{A}[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}]$ for a small entries long-term secret matrix $\mathbf{R}$, by sampling a Gaussian $\mathbf{z}$ from $\Lambda^{\perp}(\mathbf{G})$ and computing $\mathbf{x} = [\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}]\mathbf{z} + \mathbf{p}$, with $\mathbf{p}$ ensuring a spherical Gaussian distribution on $\Lambda^{\perp}(\mathbf{A})$.



(a) Successful full key recovery attack leveraging recovered coefficients of $\mathbf{S}$ and $\mathbf{E}$ (in red) via SCA, followed by completion using lattice reduction.



(b) Successful forgery attack leveraging recovered coefficients of $\mathbf{S}$ (in red) via SCA, even when coefficients of $\mathbf{E}$ remain unrecovered. Blue entries in $\mathbf{z}$ represent coefficients whose signs are identified through SCA.

Fig. 1: Our two attack scenarios: key recovery **(a)** and signature forgery **(b)**.

Acknowledgements. We thank the anonymous reviewers for their useful remarks that helped improve the quality of this paper. This work has been partially supported by the European Union - Next Generation EU through the France Relance program and by the French government through the France 2030 program under the project RESQUE. This work has been further supported by the French Agence Nationale de la Recherche through the France 2030 program under grant agreement N. ANR-22-PETQ-0008 PQ-TLS.

2 Preliminaries

Matrices are denoted by bold uppercase letters (e.g., $\mathbf{A}$), while vectors are represented by bold lowercase letters (e.g., $\mathbf{x}$). For a vector $\mathbf{x}$, let x_i denote its i-th component. For a matrix $\mathbf{A}$, let $A_{i,j}$ refer to the entry in the i-th row and j-th column. The notation $\mathbf{0}^{m\times n}$ represents the $m \times n$ zero matrix, and $\mathbf{0}^m$ denotes the m-dimensional zero vector.

In this work, all modular reductions are centered. Namely, the modular reduction $\cdot \bmod p$ maps $\mathbb{Z}$ to the interval $[-\frac{p}{2}, \frac{p}{2}) \cap \mathbb{Z}$. For any $x \in \mathbb{R}$, let $\lceil x \rfloor$ denote the nearest integer to x, with this notation extended component-wise to vectors. Note that for any $x \in \mathbb{Z}$ and $p \in \mathbb{N}$, $\lceil x/p \rfloor = (x - (x \bmod p))/p$.

A lattice $\Lambda \subset \mathbb{R}^n$ is defined as a discrete subgroup of $\mathbb{R}^n$. Its covolume $\mathrm{Vol}(\Lambda)$ is the norm of the determinant of any of its bases. The n-dimensional Gaussian function $\rho : \mathbb{R}^n \to (0, 1]$ is given by $\rho(\mathbf{x}) := \exp(-\pi\|\mathbf{x}\|^2)$. For a positive definite matrix $\mathbf{\Sigma} = \mathbf{B}\mathbf{B}^t$, we can apply the invertible linear transformation $\mathbf{B}$ to define a Gaussian function $\rho_{\mathbf{\Sigma}}(\mathbf{x}) := \rho(\mathbf{B}^{-1}\mathbf{x}) = \exp(-\pi\mathbf{x}^t\mathbf{\Sigma}^{-1}\mathbf{x})$. For any $\mathbf{c} \in \mathrm{span}(\mathbf{\Sigma})$, the Gaussian function centered at $\mathbf{c}$ with parameter $\mathbf{\Sigma}$ is defined as $\rho_{\mathbf{\Sigma},\mathbf{c}}(\mathbf{x}) := \rho_{\mathbf{\Sigma}}(\mathbf{x} - \mathbf{c})$. The discrete Gaussian distribution centered at $\mathbf{c}$ with parameter $\mathbf{\Sigma}$ over a lattice Λ has probability mass function $D_{\Lambda,\mathbf{\Sigma},\mathbf{c}}(\mathbf{x}) := \frac{\rho_{\mathbf{\Sigma},\mathbf{c}}(\mathbf{x})}{\rho_{\mathbf{\Sigma},\mathbf{c}}(\Lambda)}$ for all $\mathbf{x} \in \Lambda$, where $\rho_{\mathbf{\Sigma},\mathbf{c}}(\Lambda) = \sum_{\mathbf{x}\in\Lambda} \rho_{\mathbf{\Sigma},\mathbf{c}}(\mathbf{x})$. Similarly, the discrete Gaussian distribution on a lattice coset $\Lambda + \mathbf{v}$, where $\mathbf{v} \in \mathrm{span}(\Lambda)$, is defined as $D_{\Lambda+\mathbf{v},\mathbf{\Sigma},\mathbf{c}}(\mathbf{x}) := \frac{\rho_{\mathbf{\Sigma},\mathbf{c}}(\mathbf{x})}{\rho_{\mathbf{\Sigma},\mathbf{c}}(\Lambda+\mathbf{v})}$ for all $\mathbf{x} \in \Lambda + \mathbf{v}$. For any positive $\varepsilon > 0$, the smoothing parameter $\eta_\varepsilon(\Lambda)$ is the smallest $s > 0$ such that $\rho(s \cdot \Lambda^*) \leq 1+\varepsilon$, where $\Lambda^* := \{\mathbf{x} \in \mathrm{span}(\Lambda) \mid \langle \mathbf{x}, \Lambda\rangle \subseteq \mathbb{Z}\}$ is the dual lattice of Λ.

For any set S, we use the notation $U(S)$ for the uniform distribution over S. We let B_1 be the centered binomial distribution with parameter 1, i.e. the distribution with mass probability $p(-1) = p(1) = 1/4$ and $p(0) = 1/2$. We let $X \leftarrow_\$ P$ denote that the random variable X follows the distribution P.

2.1 Learning With Errors

Problem Description. The *Learning With Errors* (LWE) problem involves recovering a secret vector $\mathbf{s} \in \mathbb{Z}_Q^n$ given a matrix $\mathbf{A} \in \mathbb{Z}_Q^{m\times n}$ and a vector $\mathbf{b} \in \mathbb{Z}_Q^m$, satisfying the relation $\mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e} \bmod Q$, where $\mathbf{e} \in \mathbb{Z}_Q^m$ is a short error vector with entries sampled independently from an error distribution χ. While in its original formulation [Reg05], the secret vector is assumed to be uniformly sampled at random, it is usually sampled from the error distribution, as [ACPS09] proved that this was not weakening the problem.

BKZ. To find a short vector in a d-dimensional lattice Λ, one typically uses the BKZ algorithm [SE94], which is parameterized by its block size β. BKZ with block size $d > \beta > 50$ finds a vector $\mathbf{v} \in \Lambda$ such that:

$$\|\mathbf{v}\| \leq \delta_\beta^d \cdot \mathrm{Vol}(\Lambda)^{1/d} \quad \text{and} \quad \delta_\beta \approx \left(\frac{(\pi\beta)^{1/\beta}\beta}{2\pi e}\right)^{1/(2(\beta-1))}.$$

Estimating the cost of BKZ with block size β is not a trivial task. The Core-SVP methodology considers that its cost is essentially the one of the β-dimensional SVP oracle that is used in BKZ and overlooks the polynomial overheads of the algorithm. Currently, the best classical algorithm known to solve SVP in dimension β was designed in [BDGL16] and has bit-cost 0.292β.

On the more practical side, the number of core-hours from G6K [ADH+19] are a good indicator of the cost of BKZ, as it holds records in high-dimension (i.e. up to 120) SVP-solving.

Primal Attack. Attacking an LWE instance typically involves two primary strategies: the *dual attack*, which involves solving a *Short Integer Solution* (SIS) instance, and the *primal attack*. The latter [ADPS16] is the most fundamental approach to solving the LWE problem. It interprets the LWE instance as a *Bounded Distance Decoding* (BDD) instance, where the goal is to find the closest vector to $\mathbf{b}$ in the lattice:

$$\Lambda_Q = \{\mathbf{y} \in \mathbb{Z}^m : \mathbf{y} = \mathbf{A}\mathbf{s} \bmod Q, \mathbf{s} \in \mathbb{Z}^n\}.$$

When the secret is sampled from the same distribution as the error, this approach is reduced to solving the *unique Shortest Vector Problem* (uSVP) instance defined on the following Q-ary lattice of dimension $m + n + 1$:

$$\mathcal{L}_{\text{LWE}} = \begin{pmatrix} Q\mathbf{I}_m & -\mathbf{A} & \mathbf{b} \\ \mathbf{0} & \mathbf{I}_n & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & 1 \end{pmatrix} \cdot \mathbb{Z}^{m+n+1},$$

using Kannan's embedding [Kan87] with an embedding factor of 1. This lattice has covolume Q^m and contains a remarkably short vector $\mathbf{u}$ with norm:

$$\|\mathbf{u}\| = \sqrt{\|\mathbf{e}\|^2 + \|\mathbf{s}\|^2 + 1}.$$

In the case of the primal attack, it was shown [ADPS16] that it succeeds when the block size β is such that for some non-negative integer $\ell \leq m$:

$$\sqrt{\frac{3\beta}{4(\ell + n + 1)}}\|\mathbf{s}, \mathbf{e}, 1\| \cdot \leq \delta_\beta^{2\beta-(\ell+n+1)-1} \cdot Q^{\ell/(\ell+n+1)},$$

where the $\sqrt{3/4}$ is a conservative choice, as estimated per [Duc18].

Approximate-CVP. For any matrix $\mathbf{A} \in \mathbb{Z}_Q^{m \times 2m+n}$ and vector $\mathbf{u} \in \mathbb{Z}_Q^m$, the nearest-colattice algorithm [EK20] finds a vector $\mathbf{x}$ such that $\mathbf{A}\mathbf{x} = \mathbf{u} \bmod Q$ and $\|x\| \leq B_\beta$ by calling BKZ with block size β, where:

$$B_\beta = \min_{k \leq m+n} \left(\delta_\beta^{2m+n-k} \cdot Q^{m/(2m+n-k)} \right).$$

2.2 HuFu

The HuFu signature scheme was introduced in [YJL+23] and is based on the Hash-and-Sign paradigm. This consists for lattice-based signatures in exhibiting a preimage by the trapdoored public matrix $\mathbf{A}$ of the target $H(\mu) = \mathbf{u} \in \mathbb{Z}_Q^m$ as a signature for the message μ. The shorter the preimage is, the better the security, as there are less admissible preimages.

However, the distribution of the preimage is critical, as bad choices may leak the signing key, i.e. the trapdoor stored in $\mathbf{A}$. Here, HuFu relies on the approximate preimage sampling technique from [YJW23].

Algorithms for key generation, signature and verification and HuFu-1 parameters are recalled in Figure 2. Since our analysis does not target the BlockCholesky algorithm or the compression and encoding steps, we omit their description here. For complete explanations, we refer readers to the original HuFu specification [YJL+23].

Going into more details, the HuFu signature scheme is parameterized by modulus $Q = pq$, where p is a large power of 2 and q is a small power of 2. The key generation algorithm generates m plain LWE samples $\hat{\mathbf{A}}\mathbf{S} + \mathbf{E} \bmod Q$ with secret and noise coefficients each sampled from B_1. The final public matrix $\mathbf{A}$ is $(\mathbf{Id}_m | \hat{\mathbf{A}} | p\mathbf{Id}_m - \hat{\mathbf{A}}\mathbf{S} - \mathbf{E})$.

$\mathsf{KeyGen}(1^\lambda)$

$\hat{\mathbf{A}} \leftarrow\!\!\$\ \mathbb{Z}_Q^{m \times n}$
repeat
  $(\mathbf{S}, \mathbf{E}) \leftarrow\!\!\$\ B_1^{n \times m} \times B_1^{m \times m}$
  $\mathbf{\Sigma_p} \leftarrow \sigma^2 \mathbf{I}_{n+2m} - r^2 \cdot \begin{bmatrix} \mathbf{E} \\ \mathbf{S} \\ \mathbf{I}_m \end{bmatrix} \cdot \begin{bmatrix} \mathbf{E}^t & \mathbf{S}^t & \mathbf{I}_m \end{bmatrix}$
until $\mathbf{\Sigma_p} - \bar{r}^2 \mathbf{I}_{n+2m}$ is positive definite
$\mathbf{B} \leftarrow p\mathbf{I}_m - (\hat{\mathbf{A}}\mathbf{S} + \mathbf{E}) \bmod Q$
$\mathbf{C} \leftarrow \mathsf{BlockCholesky}(\mathbf{\Sigma_p} - \bar{r}^2 \mathbf{I}_{n+2m})$
return $\mathsf{vk} = (\hat{\mathbf{A}}, \mathbf{B})$, $\mathsf{sk} = (\mathbf{E}, \mathbf{S}, \mathbf{C})$

$\mathsf{Verify}(\mathsf{vk}, \mu, \sigma)$

$\mathbf{u} \leftarrow H(\mu, \mathsf{salt})$
$\mathbf{x}_0' \leftarrow \mathbf{u} - \hat{\mathbf{A}}\mathbf{x}_1 - \mathbf{B}\mathbf{x}_2 \bmod Q$
if $\|(\mathbf{x}_0', \mathbf{x}_1, \mathbf{x}_2)\| \leq B$
  Accept
else
  Reject

$\mathsf{Sign}(\mathsf{sk}, \mu)$

$\mathbf{A} \leftarrow [\mathbf{I}_m\ \hat{\mathbf{A}}\ \mathbf{B}]$
$\mathbf{p} \leftarrow \mathsf{SampleP}(\mathsf{sk})$
$(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2) \leftarrow \mathbf{p}$
$\mathbf{c} \leftarrow \mathbf{A}\mathbf{p} \bmod Q$
$\mathsf{salt} \leftarrow\!\!\$\ \{0,1\}^{320}$
$\mathbf{u} \leftarrow H(\mu, \mathsf{salt})$
$\mathbf{v} \leftarrow \mathbf{u} - \mathbf{c} \bmod Q$
$\mathbf{e} \leftarrow \mathbf{v} \bmod p$
$\mathbf{v}' \leftarrow (\mathbf{v} - \mathbf{e})/p$
for $i = 1, \ldots, m$
  $z_i \leftarrow q \cdot \mathsf{SampleZ}_d(v_i'/q)$
$\mathbf{x}_0 \leftarrow \mathbf{E}\mathbf{z} + \mathbf{p}_0$
$\mathbf{x}_1 \leftarrow \mathbf{S}\mathbf{z} + \mathbf{p}_1$
$\mathbf{x}_2 \leftarrow \mathbf{z} + \mathbf{p}_2$
if $\|(\mathbf{x}_0 + \mathbf{e}, \mathbf{x}_1, \mathbf{x}_2)\| > B$
  goto 2
return $\sigma = (\mathsf{salt}, \mathbf{x}_1, \mathbf{x}_2)$

Fig. 2: Simplified pseudocode of HuFu. HuFu-1 uses $(m, n) = (736, 848)$, $(p, q) = (2^{12}, 2^4)$ and $B = 62521$.

A HuFu signature is a preimage of $p\mathbf{v}' = p\lfloor \mathbf{u}/p \rceil$, where $\mathbf{u} \in \mathbb{Z}_Q^m$ is the target: as $\mathbf{A}$ contains the identity matrix, this is equivalent to sampling a somewhat larger preimage of $\mathbf{u}$. This is achieved by choosing a short element $\mathbf{z} \in q\mathbb{Z}^m + \mathbf{v}'$. The signature is then $(\mathbf{Ez}, \mathbf{Sz}, \mathbf{z})$, which is still short and is a preimage of $\mathbf{v}'$. The first coordinate $\mathbf{Ez}$ is omitted from the signature, as it can be reconstructed as $\mathbf{u} - \hat{\mathbf{A}}(\mathbf{Sz}) - \mathbf{Bz}$. As it stands this signature leaks the secret matrices $\mathbf{S}$ and $\mathbf{E}$.

To solve this issue, the signer starts by sampling $\mathbf{p}$, and updates the target to $\mathbf{v} = \mathbf{u} - \mathbf{Ap} \bmod Q$. The signature is $(\mathbf{Ez}, \mathbf{Sz}, \mathbf{z}) + \mathbf{p}$, which is indeed a short preimage by $\mathbf{A}$ of $\mathbf{u}$. Once again, the first m coordinates are erased as they can be approximately recovered by the verifier.

A careful choice of the distributions of $\mathbf{z}$ and $\mathbf{p}$ makes it so that the final distribution of the signature is independent of the secret. This choice is the discrete Gaussian distribution: let $D_{q\mathbb{Z}^m+\mathbf{v}',r}$ be the output distribution of $(q \cdot \mathsf{SampleZ}_d(v_i'/q))_i$ and $D_{\mathbb{Z}^{2m+n},\mathbf{\Sigma}_p}$ be the output distribution of $\mathsf{SampleP}$ for some $r > 0$ and some positive-definite matrix $\mathbf{\Sigma}_p$. Conditioned on $\bar{r} > q\eta_\varepsilon(\mathbb{Z})$, the distribution of the signature is ε-close to a discrete Gaussian distribution centered around 0 with covariance matrix: $\mathbf{\Sigma} = \mathbf{\Sigma}_p + r^2\mathbf{S}_0^\top\mathbf{S}_0 = \sigma^2\mathbf{Id}_{2m+n}$, where $\mathbf{S}_0 = (\mathbf{E}^\top, \mathbf{S}^\top, \mathbf{Id}_m)$ and by choosing $\mathbf{\Sigma}_p$ accordingly.

The following result is a well-known fact, but we include a brief proof for completeness in Section A:

Lemma 2.1. *Let Q be a power of 2 and $m < n$ be integers, and let $\hat{\mathbf{A}} \leftarrow\!\!\$\ U(\mathbb{Z}_Q^{m\times n})$. For $0 \le k_2 < n$ and $0 < k_1 < n - k_2$, any k_1-row and $(n-k_2)$-column submatrix of $\hat{\mathbf{A}}$ contains an invertible submatrix $\hat{\mathbf{A}}_{sub} \in \mathbb{Z}_Q^{k_1\times k_1}$ with probability:*

$$p_{k_1} = \prod_{i=1}^{k_1} \left(1 - 2^{i-1-(n-k_2)}\right).$$

3 Exploiting 0 leakage

The goal of this section is to recover partial information via side-channel analysis on the private matrices $\mathbf{S}$ and $\mathbf{E}$, which we exploit in a key recovery and a signature forgery in the following sections. Namely we look for the location of the coefficients whose value is equal to zero, which we dub the "0 leakage".

Targeted Operations. We target the multiplications $\mathbf{E}\cdot\mathbf{z}$ and $\mathbf{S}\cdot\mathbf{z}$ at Lines 12 and 13 of the signature procedure described in Figure 2. $\mathbf{S}$ and $\mathbf{E}$ are long-term secrets while $\mathbf{z}$ is an ephemeral secret vector. Hence, we are not in a classical setting for a DPA, as none of the operands is known by the attacker. From a side-channel point of view, the operations involving either $\mathbf{S}$ or $\mathbf{E}$ are similar and we will only describe the recovery of $\mathbf{S}$ for conciseness. The targeted C code is shown in Listing 1, taken from the official HuFu submission package.

Experimental Setup. The power traces have been recorded with a ChipWhisperer Lite with STM32F303 target (ARM Cortex M4). Due to limitations of

our acquisition device, we executed the `mat_mul` code with matrices of smaller dimensions 8x8. However, we expect the attack to behave similarly in higher dimensions, as the traces will have identical patterns and point of leakages, and will only be longer with more individual multiplications involved. To validate the feasibility of our attack, we generated 100 datasets of 1500 power traces each. After performing the attack on each dataset separately, we aggregated the results of the 6400 coefficients on Figure 8.

```
1  void mat_mul(int16_t *C, int16_t *A, int16_t *B, int l1,
       int l2, int l3)
2  {
3    for (int i = 0; i < l1; ++i)
4      for (int j = 0; j < l3; ++j)
5      {
6        mat_element(C, l3, i, j) = 0;
7        for (int k = 0; k < l2; ++k)
8          mat_element(C, l3, i, j) += mat_element(A, l2, i,
       k) * mat_element(B, l3, k, j);
9      }
10 }
```

Listing 1: Reference C code of matrix-vector multiplication.

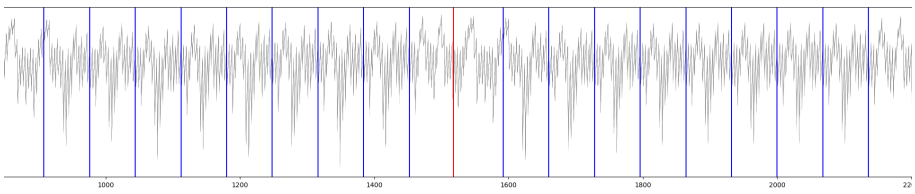


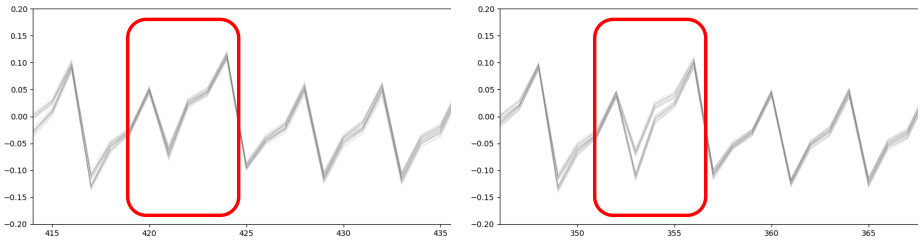
Fig. 3: Extract of power consumption during Listing 1 execution for `l2 = 8`.

Overview. The multiplication between $\mathbf{S}$ and $\mathbf{z}$ consists in $n \cdot m$ multiplications and additions of single coefficients $S_{i,j}$ and z_i. Figure 3 shows an extract of the power consumption during the multiplication between $\mathbf{S}$ and $\mathbf{z}$. The red line separates two multiplications between lines of $\mathbf{S}$ and $\mathbf{z}$, while the blue lines separate multiplications between single coefficients $S_{i,j}$ and z_i.

We exploit in our attack that $\mathbf{S}$ is ternary, i.e. the only possible values for its coefficients are $\{-1, 0, 1\}$. Consequently, multiplication between $S_{i,j}$ and z_i for some i, j can only result in three different cases:

1. 0, setting the Hamming weight to 0.
2. z_i, keeping the Hamming weight identical.
3. $-z_i$, greatly changing the Hamming weight.

We expect to see the above classification in the power consumption. In particular, $S_{i,j} = 0$ will always result in a Hamming weight of 0. For a non-zero $S_{i,j}$, we expect to see the power traces split into two groups, depending on the sign of $S_{i,j} \cdot z_i$. Those assumptions are confirmed by experimental analysis, as shown in Figure 4a and Figure 4b: a difference can be noted between samples 420-425 and samples 350-355. As we have no way to distinguish between the Cases 2 and 3, we focus on identifying only the Case 1 corresponding to $S_{i,j} = 0$.



(a) $S_{i,j} = 0$. **(b)** $S_{i,j} \neq 0$.

Fig. 4: Power consumption during the computation of $S_{i,j} \cdot z_i$.

Practical Attack. The goal is to determine whether the power traces at the selected points of interest are forming one or two clusters. To this end, we use the Kernel Density Estimator (KDE) from the Python package `sickit-learn`. To improve accuracy, we separated the power traces into 10 subsets and performed a majority voting to determine the final classification. Since we are only interested in finding the zero coefficients, we can choose conservative parameters for our KDE to drastically decrease the number of false positives.

Results. An extract of the results is shown on Table 1 (full results will be discussed in Section 5.3). From as few as 200 power traces, we are able to recover more than 90% of the $S_{i,j} = 0$. However, we must increase the number of traces to at least 600 to avoid any false positives, at which point we also get an accuracy of 97%. The results do not get much better with an increasing number of traces, and they remain steady around 98% up to 1500 power traces. By extension, we can also recover the same proportion of the $E_{i,j} = 0$.

4 Key Completion via Lattice Reduction

In this section, we exploit the side-channel information from Section 3 to recover the secret matrix $\mathbf{S}$. We focus on individual LWE instances $\hat{\mathbf{A}}\mathbf{s} + \mathbf{e} = \mathbf{b} \bmod Q$, each corresponding to a single column of the matrices $\mathbf{S}$, $\mathbf{E}$, and $p\mathbf{I}_m - \mathbf{B}$, as defined in Section 2.2 and target them separately. The challenge of leveraging

Table 1: Percentage of $S_{i,j} = 0$ successfully recovered.

Number of traces	Recovered $S_{i,j} = 0$	False positives
200	93.4%	0.12%
600	97.9%	0%
1500	98.5%	0%

partial information about the error or secret in LWE has been explored through various methods, as discussed in [DDGR20], [DGHK23], and [MN23]. In this work, we focus on the approach proposed in [MN23], which offers a framework particularly well-suited to tackling the specific challenges present in our context.

4.1 Partial Knowledge of the Secret

By utilizing side-channel information, we gain access to certain entries of $\mathbf{S}$. Building on this partial knowledge, we aim to recover individual columns $\mathbf{s}$ of $\mathbf{S}$, paving the way for a (partial) key recovery attack. To achieve this, we replace the k known coefficients of $\mathbf{s}$ in the LWE system, thereby reducing the number of unknowns by k.

Let $\tilde{\mathbf{s}}$ represent the partially known vector $\mathbf{s}$, defined as:

$$\tilde{s}_i = \begin{cases} s_i & \text{if the } i\text{-th component is known,} \\ 0 & \text{otherwise.} \end{cases}$$

Note that, in the context of Section 3, $\tilde{\mathbf{s}}$ is an all-zero vector due to the specific leakage model considered. However, we later consider cases where some nonzero s_i are known. Substituting $\tilde{\mathbf{s}}$ into the LWE relation gives:

$$\hat{\mathbf{A}}\mathbf{s} + \mathbf{e} = \mathbf{b} \bmod Q \iff \hat{\mathbf{A}}(\mathbf{s} - \tilde{\mathbf{s}}) + \mathbf{e} = \mathbf{b} - \hat{\mathbf{A}}\tilde{\mathbf{s}} \bmod Q.$$

Assume that k coordinates of $\mathbf{s}$ have been leaked. This simplifies the original LWE instance by removing corresponding columns and entries, leading to the reduced instance $(\mathbf{A}', \mathbf{s}', \mathbf{e}, \mathbf{b}')$, where:

- $\mathbf{A}'$ is derived by removing the k columns from $\hat{\mathbf{A}}$,
- $\mathbf{s}'$ consists of the remaining coordinates of $\mathbf{s} - \tilde{\mathbf{s}}$ after excluding the k-$\mathbf{0}$, and
- $\mathbf{b}'$ is $\mathbf{b} - \hat{\mathbf{A}}\tilde{\mathbf{s}}$.

This modification decreases the lattice problem's dimension by k, thereby weakening its security. However, relying solely on this method requires knowledge of a significant portion of $\mathbf{s}$ to render the attack practical (e.g., reducing dimensions below 200). Considering that the LWE error $\mathbf{e}$ also leaks information, a natural question emerges: *Can we further reduce the LWE instance dimension by exploiting partial knowledge of* $\mathbf{e}$ *alongside* $\mathbf{s}$*?*

4.2 Partial Knowledge of the Error

Knowing a coefficient of the error vector $\mathbf{e}$ reduces the dimension of the LWE solution by one, similar to knowing a coefficient of the secret vector $\mathbf{s}$. The core idea is that knowing one coordinate of $\mathbf{e}$ allows us to derive a linear combination of the components of $\mathbf{s}$. Specifically:

$$b_i - e_i = \sum_j \hat{a}_{i,j} \cdot s_j \bmod Q.$$

Since $\hat{\mathbf{A}}$ is sampled uniformly, its rows form a linearly independent generating system with high probability. Consequently, this relation reduces the dimension of the system induced by the LWE instance by one. More generally, knowing k such relations reduces the dimension by k.

Assuming that the first k coordinates of $\mathbf{e}$ are known (possibly after reordering), the LWE equation can be reformulated as:

$$(\hat{\mathbf{A}} \mid \mathbf{b}) \cdot (\mathbf{s}^\top, -1)^\top = (-\mathbf{e}_1, -\mathbf{e}_2)^\top \bmod Q,$$

where $\mathbf{e}_1 = (e_1, \ldots, e_k)$ and $\mathbf{e}_2 = (e_{k+1}, \ldots, e_m)$. Given that $\hat{\mathbf{A}}$, $\mathbf{b}$, and $\mathbf{s}$ can be decomposed (possibly after permuting columns) as:

$$(\hat{\mathbf{A}} \mid \mathbf{b}) = \begin{pmatrix} \mathbf{A}_1 & \mathbf{A}_2 & \mathbf{b}_1 \\ \mathbf{A}_3 & \mathbf{A}_4 & \mathbf{b}_2 \end{pmatrix}, \text{ and } \mathbf{s} = (\mathbf{s}_1 \mid \mathbf{s}_2)^\top,$$

where $\mathbf{A}_1 \in \mathbb{Z}_Q^{k \times k}$ is invertible with overwhelming probability, see Lemma 2.1. We can apply Gaussian elimination to the first k rows, transforming the system to the equivalent form:

$$\begin{pmatrix} \mathbf{I}_k & \mathbf{A}_1^{-1}\mathbf{A}_2 & \mathbf{A}_1^{-1}(\mathbf{b}_1 - \mathbf{e}_1) \\ \mathbf{A}_3 & \mathbf{A}_4 & -\mathbf{b}_2 \end{pmatrix} \cdot (\mathbf{s}_1 \mid \mathbf{s}_2 \mid -1)^\top = (\mathbf{0}^k, -\mathbf{e}_2)^\top \bmod Q.$$

By applying additional row operations to eliminate $\mathbf{A}_3$, the system simplifies to:

$$\begin{pmatrix} \mathbf{I}_k & \mathbf{A}_1^{-1}\mathbf{A}_2 & \mathbf{A}_1^{-1}(\mathbf{b}_1 - \mathbf{e}_1) \\ \mathbf{0}^{(m-k)\times k} & \bar{\mathbf{A}} & \bar{\mathbf{b}} \end{pmatrix} \cdot (\mathbf{s}_1 \mid \mathbf{s}_2 \mid -1)^\top = (\mathbf{0}^k, -\mathbf{e}_2)^\top \bmod Q,$$

where:

$$\bar{\mathbf{A}} = \mathbf{A}_4 - \mathbf{A}_3\mathbf{A}_1^{-1}\mathbf{A}_2, \text{ and } \bar{\mathbf{b}} = \mathbf{A}_3\mathbf{A}_1^{-1}(\mathbf{e}_1 - \mathbf{b}_1) - \mathbf{b}_2.$$

The values of $\mathbf{s}_1$ can be determined from the first k equations, leading to a new LWE instance with reduced dimension $(n-k)$ and parameters $(\bar{\mathbf{A}}, \mathbf{s}_2, \mathbf{e}_2, \bar{\mathbf{b}})$. Notably, both methods of exploiting known coordinates of $\mathbf{s}$ and $\mathbf{e}$ are specific instances of incorporating mod-Q hints, as outlined in [MN23].

4.3 Combining Insights from Secret and Error

We assume that $0 \leq k_1 \leq n$ coordinates of the secret vector $\mathbf{s}$ and $0 \leq k_2 \leq m$ coordinates of the error vector $\mathbf{e}$ are known. For the system's dimension to be

reduced by exactly $k_1 + k_2$, the linear combinations defining the leakage on $\mathbf{s}$ must remain linearly independent.

Let $\hat{\mathbf{A}}_{\text{sub}} \in \mathbb{Z}_Q^{k_2 \times n}$ represent the submatrix corresponding to the leaked coordinates of $\mathbf{e}$. If $\hat{\mathbf{A}}_{\text{sub}}$ maintains rank k_1 after excluding k_2 columns related to the leaked coordinates of $\mathbf{s}$, the resulting LWE instance will have the desired reduced dimension. This condition holds with high probability, as shown in Lemma 2.1.

For the general case $0 \leq k_2 < m$ and $0 \leq k_1 < n - k_2$, we refer to Figure 5, generated using the `leaky-estimator` from [DDGR20], to determine the BKZ block size needed to recover an entire column of $\mathbf{S}$.

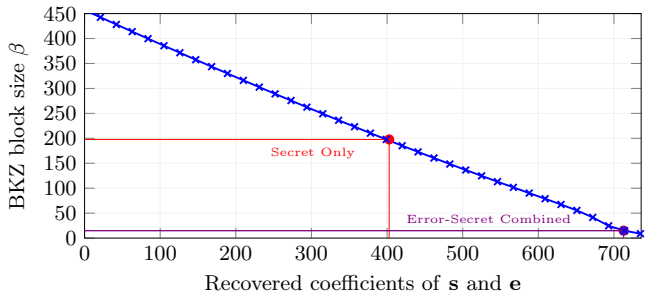


Fig. 5: Estimated evolution of the BKZ block size β as a function of the number of recovered coefficients of $\mathbf{s}$ and $\mathbf{e}$ for HuFu-1 with $(n, Q, \chi) = (848, 2^{16}, B_1)$.

When $k_1 + k_2 \geq n$, lattice reduction becomes unnecessary. Gaussian elimination can recover the secret, with the caveat that $\mathbb{Z}_Q$ is not a field.

4.4 Key Recovery with 0 Leakage

In HuFu, the LWE secrets and errors $(\mathbf{S}, \mathbf{E})$ are independently and identically distributed according to a centered binomial distribution with parameter 1. Therefore, on average, each column $\mathbf{s}$ of $\mathbf{S}$ contains $n/2$ zeros, and each column $\mathbf{e}$ of $\mathbf{E}$ contains $m/2$ zeros. By the law of large numbers, for large n and m, this is the case for all considered samples.

Secret Only. Based on Table 1, recovering more than 95% of the zeros in $\mathbf{S}$ with 100% accuracy requires 600 power traces. For HuFu-1, this is approximately 400 coefficients per column of $\mathbf{S}$. Completing the remaining coefficients necessitates BKZ with a block size around 200. Consequently, recovering the full HuFu secret key would require over 2^{77} operations, making the attack unfeasible.

Error-Secret Combined. However, Recall that the attack in Section 3 applies to $\mathbf{E}$ as well. By exploiting leakage from both $\mathbf{S}$ and $\mathbf{E}$, we can recover approximately $0.95 \cdot (n + m)/2$ coefficients per LWE instance, compared to $0.95 \cdot n/2$ when using $\mathbf{S}$ alone.

Using the same number of power traces for HuFu-1, this approach yields approximately 740 coefficients of $(\mathbf{s}, \mathbf{e})$. Consequently, the required BKZ block size reduces to 10, lowering the time complexity for full key recovery to roughly 2^{25}, which is computationally feasible within a reasonable time frame.

5 Trading 0 leakage on E for more-than-0 leakage on S

While devastating, the combined attack from Section 4 requires knowledge on both $\mathbf{S}$ and $\mathbf{E}$. We now show how to get rid of the leakage on $\mathbf{E}$.

5.1 First Countermeasure: Preventing the Leakage on E

We note that the computation $\mathbf{x}_0 = \mathbf{E}\mathbf{z} + \mathbf{p}_0$ is only ever useful to compute the norm of $(\mathbf{x}_0 + \mathbf{e}, \mathbf{x}_1, \mathbf{x}_2)$. However, the vector $\mathbf{x}_0 + \mathbf{e}$ can also be expressed as

$$\mathbf{u} - \hat{\mathbf{A}}\mathbf{x}_1 - \mathbf{B}\mathbf{x}_2.$$

Computing it this way prevents leakage on $\mathbf{E}$ here. Note that the computation of $\mathbf{x}_0$ is only there to ensure correctness and is not sensitive, as rejected signatures do not reveal any information on the private key.

Performance cost. The countermeasure involves an additional matrix-vector multiplication and and additional vector addition. We noticed an overhead in signature generation of around 5% in terms of CPU cycles on Intel architecture.

If we apply this simple countermeasure, an attacker is not able to recover information on the private matrix $\mathbf{E}$ anymore. As shown in Section 4.4, a lattice reduction with only the zero-coefficients of $\mathbf{S}$ is likely to be too costly. Hence, the goal of this section is to perform a deeper side-channel analysis to recover more information on $\mathbf{S}$. We do so in two steps.

1. Target the Gaussian sampler to gain information on the sampled vector $\mathbf{z}$.
2. Use this additional information to distinguish between the $S_{i,j} = \pm 1$.

5.2 Side-channel Analysis of the Gaussian Sampler

Contrary to the CDT samplers of similar schemes Falcon and Hawk that have been analyzed in [GMRR22,ZLYW23,GR24], the distribution table of the HuFu sampler is designed to sample both negative and positive integers. This means that, for a sample of value 0, the inner counter will have been incremented during half of the table traversal, whereas it would not have been incremented at all in more traditional samplers. As the noise on the power traces grows bigger during the execution, recovering the z_i that are close to 0 without profiling proves much more challenging in this sampler than in the ones listed earlier. For this reason, we will not target the table traversal but rather the lines 15 and 2 in Listing 2.

```
1  int c = center;
2  c = (c > 8) * (16 - 2 * c) + c;
3  z = 0;
4  for (u = 0; u < TABLE_LEN; u += 3)
5  {
6    uint32_t w0, w1, w2, cc;
7    w0 = dist0[c][u + 2];
8    w1 = dist0[c][u + 1];
9    w2 = dist0[c][u + 0];
10   cc = (v0 - w0) >> 31;
11   cc = (v1 - w1 - cc) >> 31;
12   cc = (v2 - w2 - cc) >> 31;
13   z += (int)cc;
14 }
15 return (center > 8) * (27 - 2 * z) + z - 13;
```

Listing 2: Extract of the RCDT sampler used in the online phase.

Sign Recovery. The last operation executed at Line 15 is the subtraction between an intermediary value and 13. This is done to center the Gaussian sampler around 0 and output a sample in $[-12, 12]$. Let:

$$z_{\text{CDT}} = (27 - 2 \cdot z) + z - 13 = 14 - z$$

be the final value outputted by the sampler. If the result of the final subtraction is negative, i.e. $z_{\text{CDT}} < 0$, it induces a spike in power consumption as shown on Figure 6c, that is visible with bare-eyes and is detected with a simple threshold.

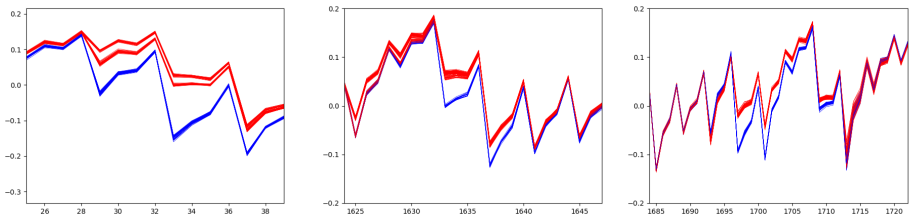
Identifying Zeros. The multiplication step `(center > 8) * (27 - 2 * z)` shows a similar leakage. However, it only reveals the sign of `27 - 2 * z` if `center` is greater than 8. Otherwise, the intermediary result targeted here will be 0 independently of the value of z. We then know when $27 - 2z < 0$, i.e. $z > 13$. From last paragraph, we also know when $z_{\text{CDT}} \geq 0$, i.e. $z \leq 14$. Hence, we isolate the case $z = 14$, i.e. $z_{\text{CDT}} = 0$. We thus identify the z_{CDT} that are zero when `center` is greater than 8, which happens with probability 0.5.

Leakage on `center`. Now we need to gain information on the `center` variable. As shown on Figure 6a, power consumption reveals whether the intermediary result $16 - 2 \cdot c$ at Line 2 is negative, implying that `center` is greater than 8. A simple threshold can be used to separate the power traces into two groups.

Results. We generated a dataset of 100,000 executions of the sampler. We were able to recover with 100% accuracy:

- for each execution, if z_{CDT} is negative or not,
- for each execution, whether `center` is greater than 8,
- for executions where `center` > 8, if z_{CDT} is 0.

The final value z_i is computed as $z_i = Q \cdot z_{\text{CDT}} - c$ with $c < Q$. Thus, while $z_{\text{CDT}} < 0 \implies z_i < 0$, the converse is not true. We discard any z_{CDT} such that $z_{\text{CDT}} \geq 0 \wedge c \leq 8$ as the sign of the corresponding z_i cannot be determined with certainty. Experimentally, we can keep around 63% of the z_i. Note that when we cannot determine the sign of some z_i, we do not discard the complete power trace as there may be other interesting values in the vector $\mathbf{z}$. We simply ignore this particular index i.

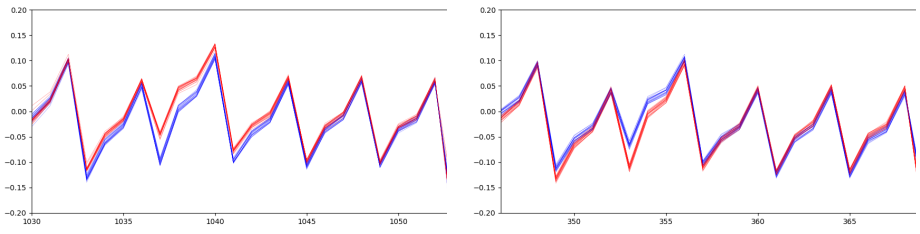


(a) Computation of c. **(b)** First multiplication. **(c)** Final subtraction.

Fig. 6: Power traces corresponding to negative (resp. non-negative) intermediary results are shown in blue (resp. red).

5.3 Recovering the Sign of $S_{i,j}$

With the knowledge of the sign of z_i, we can further analyze the power traces from Section 3 to distinguish between the 1 and -1 coefficients of $\mathbf{S}$. As described in Section 3, the power traces can be split into two groups when $S_{i,j}$ is non-zero. Those two groups depends on the sign of $S_{i,j} \cdot z_i$, hence on the sign of z_i for a fixed $S_{i,j}$. The power traces can thus be grouped depending on the sign of z_i.



(a) $S_{i,j} = -1$. **(b)** $S_{i,j} = 1$.

Fig. 7: Power traces in red (resp. blue) correspond to $z_i < 0$ (resp. $z_i > 0$).

As we can see on Figure 7a and Figure 7b, the relative position of those two groups is inverted depending on the value of $S_{i,j}$: if $S_{i,j} = -1$, the power traces corresponding to $z_i < 0$ will be positioned above the power traces corresponding to $z_i > 0$, and conversely for the $S_{i,j} = 1$.

Note that the special case where $z_i = 0$ is a blind spot of the classification detailed in Section 3. Indeed, the Hamming weight of the result will be set to 0 whatever the value of $S_{i,j}$. Thus, such z_i will not help us distinguish between ± 1 and may even induce some errors. However, this happens only when $z_{\text{CDT}} = 0$ and $c \leq 8$. Since we already decided to discard the values $z_{\text{CDT}} \geq 0 \wedge c \leq 8$, there are no $z_i = 0$ in our dataset.

5.4 Results

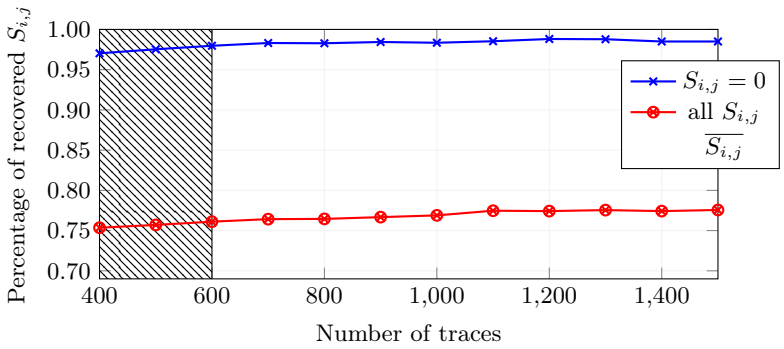


Fig. 8: Percentage of recovered $S_{i,j}$ equal to zero in blue and all the $S_{i,j}$ in red. Dashed area show where false positives $\overline{S_{i,j}}$ are found.

Figure 8 shows the percentage of recovered coefficients depending on the number of traces. The results concerning the zero coefficients have already been discussed in Section 3. If we consider all coefficients, it is possible to recover a high amount of coefficients with a low number of traces, but this may lead to false positives. As our attacks based on lattice reduction do not tolerate any false positives, we increase the number of traces up to 1500. Results for lower number of traces are only displayed to encourage future works regarding false positives tolerance. Besides, recall that to distinguish between the coefficients $S_{i,j} \in \{-1, 1\}$ we need additional information on the sign of z_i as detailed in Section 5.3, and we can only get this information for 63% of the z_i as explained in Section 5.2. The results shown on Figure 8 take this into account.

Overall, with 1500 power traces, we retrieve without any false positive:

- more than 95% of the $S_{i,j} = 0$ with no additional information (Section 3),
- around 75% of all the $S_{i,j}$ with prior information on the sign of z_i (here).

6 Forging with Partial Knowledge of S

We combine the results from Section 5 with the attack from Section 4.1, and assume that d columns of $\mathbf{S}$ out of m were recovered this way. We exhibit in this section a strategy to forge a signature for a diminished cost using this knowledge.

Assuming that the first d columns $\mathbf{S}_1$ of $\mathbf{S}$ are known, up to reordering, let:

$$\mathbf{A} = (\mathbf{Id}_m|\hat{\mathbf{A}}|\mathbf{B}) = \begin{pmatrix} \mathbf{Id}_d & \mathbf{0}^{d\times m-d} & \hat{\mathbf{A}}^{high} & \mathbf{B}_1^{high} & \mathbf{B}_2^{high} \\ \mathbf{0}^{m-d\times d} & \mathbf{Id}_{m-d} & \hat{\mathbf{A}}^{low} & \mathbf{B}_1^{low} & \mathbf{B}_2^{low} \end{pmatrix},$$

where $\mathbf{B}_1^{high} \in \mathbb{Z}_Q^{d\times d}$, $\mathbf{B}_1^{low} \in \mathbb{Z}_Q^{m-d\times d}$, $\mathbf{B}_2^{high} \in \mathbb{Z}_Q^{d\times m-d}$ and $\mathbf{B}_2^{low} \in \mathbb{Z}_Q^{m-d\times m-d}$. We also let $\mathbf{B}_i^\top = ((\mathbf{B}_i^{high})^\top|(\mathbf{B}_i^{low})^\top)$ for $i \in \{1,2\}$.

6.1 Partial Forgery using $\mathbf{S_1}$

We first forge in the case where the $m-d$ last coefficients of the target are all 0s.

Lemma 6.1. *Let p, q, m, n, d be integers and $Q = pq$. Let $\mathbf{u} \in \mathbb{Z}_Q^d \times \{\mathbf{0}^{m-d}\}$. Let $\mathbf{v} = \lceil \mathbf{u}/p \rfloor \in \mathbb{Z}^d \times \{\mathbf{0}^{m-d}\}$. Then $(\mathbf{x}_1^*, \mathbf{x}_2^*) = ((\mathbf{S}_1|\mathbf{0}^{n\times m-d})\mathbf{v}, \mathbf{v})$ is a valid HuFu signature for any μ such that $H(\mu) = \mathbf{u}$.*

Proof. Verification computes $\mathbf{x}_0^* = \mathbf{u} - \hat{\mathbf{A}}\mathbf{x}_1^* - \mathbf{B}\mathbf{x}_2^* = (\mathbf{u} \bmod p) + (\mathbf{E}_1|\mathbf{0}^{m\times m-d})\mathbf{v}$. Moreover $\|\mathbf{x}_0^*, \mathbf{x}_1^*, \mathbf{x}_2^*\| \leq B$ as we essentially set $\mathbf{p} = \mathbf{0}^{2m+n}$ and chose a short vector in $q\mathbb{Z}^m + \mathbf{v}'$, using notations from Section 2.2. □

6.2 Complete Forgery via Lattice Reduction

In practice, the probability of finding a $\mathbf{u} \in \mathbb{Z}_Q^m$ satisfying the above constraint is $1/Q^{m-d}$. Instead, we take any target $\mathbf{u} \in \mathbb{Z}_Q^m$, ignore its first d coordinates and find a suitable preimage of it for the last $m-d$ rows of $\mathbf{A}$.

Forgery. Let $H(\mu) = \mathbf{u} = (\mathbf{u}_1^\top|\mathbf{u}_2^\top)^\top$, with $\mathbf{u}_1 \in \mathbb{Z}_Q^d$ and $\|\mathbf{y}\| \leq B_\mathbf{y}$ with:

$$(\mathbf{Id}_{m-d}|\hat{\mathbf{A}}^{low}|\mathbf{B}_1^{low}|\mathbf{B}_2^{low}) \cdot \mathbf{y} = \mathbf{y}_0 + \hat{\mathbf{A}}^{low}\mathbf{y}_1 + \mathbf{B}_1^{low}\mathbf{y}_2 + \mathbf{B}_2^{low}\mathbf{y}_3 = \mathbf{u}_2 \bmod Q,$$

for some $B_\mathbf{y} > 0$ set later. We let $\mathbf{u}_1' = \mathbf{u}_1 - (\hat{\mathbf{A}}^{high}|\mathbf{B}_1^{high}|\mathbf{B}_2^{high})\mathbf{y}$. At this point, our target is $\mathbf{u}'^\top = (\mathbf{u}_1'^\top|\mathbf{0}^{1\times(m-d)})$. Let us now consider the forgery vector $(\mathbf{x}_1', \mathbf{x}_2')$ obtained with Lemma 6.1. The final forgery is:

$$\mathbf{x}_1 = \begin{pmatrix}\mathbf{x}_1' + \mathbf{y}_1\end{pmatrix} \quad \text{and} \quad \mathbf{x}_2 = \begin{pmatrix} \mathbf{x}_2' + \mathbf{y}_2 \\ \mathbf{y}_3 \end{pmatrix}.$$

Verification. The verification algorithm recovers:

$$\begin{aligned} \mathbf{x}_0 &= \mathbf{u} - \hat{\mathbf{A}}\mathbf{x}_1 - \mathbf{B}\mathbf{x}_2 \\ &= \begin{pmatrix} (\mathbf{u}_1' \bmod p) + \mathbf{E}_1\lceil \mathbf{u}_1'/p \rfloor \\ \mathbf{y}_0 \end{pmatrix} = \begin{pmatrix} \mathbf{x}_0' \\ \mathbf{y}_0 \end{pmatrix}. \end{aligned}$$

It then checks the norm of $(\mathbf{x}_0^\top, \mathbf{x}_1^\top, \mathbf{x}_2^\top)$, which is at most $\|(\mathbf{x}_0', \mathbf{x}_1', \mathbf{x}_2')\| + \|\mathbf{y}\|$ and must be $\leq B$. This last constraint drives the choice of $B_\mathbf{y}$. In practice, we empirically estimated $\|(\mathbf{x}_0', \mathbf{x}_1', \mathbf{x}_2')\|$ to derive $B_\mathbf{y}$.

6.3 Cost of Forging as a Function of d

The cost of lattice reduction is estimated following the methodology outlined in the original HuFu submission. Specifically, we rely on the nearest colattice algorithm to determine the required BKZ block size and then estimate the associated computational cost using the Core-SVP model. Figure 9 illustrates the estimated BKZ block size for HuFu-1. The bound $B_{\mathbf{y}}$ is derived by performing a hundred forgeries for each value of d from Section 6.1 and computing the average norm of the resulting vector $(\mathbf{x}'_0, \mathbf{x}'_1, \mathbf{x}'_2)$ as a function of d as shown in Figure 10.

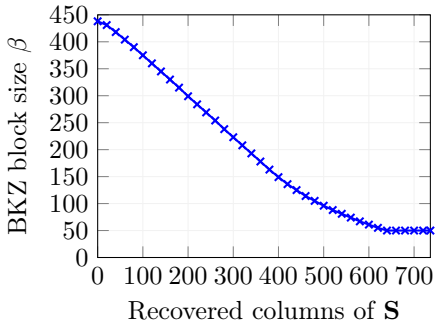


Fig. 9: Evolution of the block size to forge a signature.

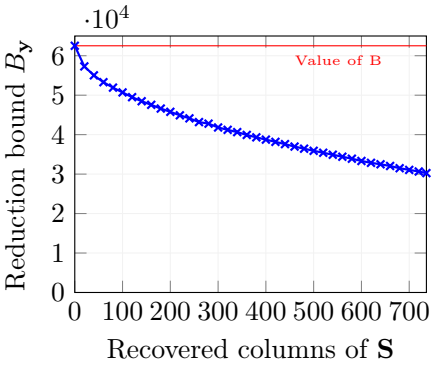


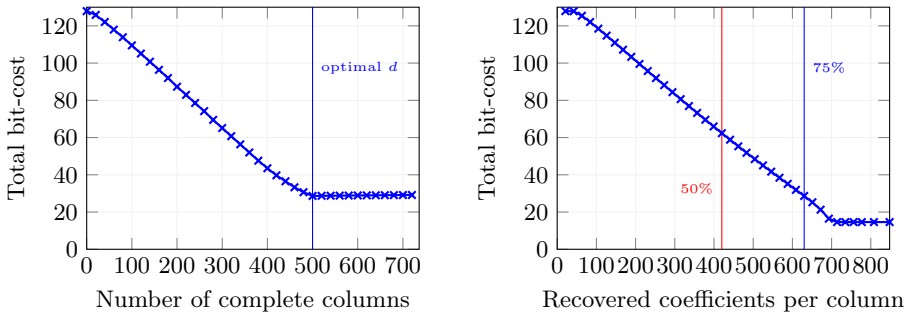
Fig. 10: $B_{\mathbf{y}}$ as a function of the number d of recovered columns of $\mathbf{S}$.

6.4 Combining Key Recovery and Forgery

We try to estimate the total cost of running both the key recovery and the forgery attack. The combination of the two is parameterized by the amount d of columns recovered by the key recovery and used by the forgery attack. We estimate the cost of the combination as $\max(\log_2(d) + 0.292\beta_{\text{kr}}, 0.292\beta_{\text{forge}}(d))$, where β_{kr} is the block size necessary to recover a column given 75% of its coordinates as estimated via Section 4.1 and $\beta_{\text{forge}}(d)$ is the block size necessary to forge given d columns of $\mathbf{S}$ as estimated in Figure 9. We present in Figure 11a the evolution of the estimation of the total cost as a function of d when the amount of recovered coefficients is about 75%.

Then, we optimize over d and show in Figure 11b the cost of the attack as a function of recovered coefficients per column. Of particular interests are the 50% and 75% threshold, as they correspond to the 0-only and all coefficients SCA attacks on $\mathbf{S}$. In the former case, the attack costs at least 62 bits, while in the latter, the attack cost is down to 29 bits.

We challenge our hypothesis that leaked coefficients are evenly distributed among all columns. In the case of the 75% recovered coefficients, we only need



(a) ... as a function of the number of recovered columns when 75% of the coefficients of **S** are known.

(b) ... as a function of the amount of recovered coefficients per column during the side-channel step.

Fig. 11: Total estimated bit-cost of key-recovery and forgery...

that there exist 500 columns whose at least 75% of their coefficients are recovered to run the attack with the claimed bit-cost. Running estimations with `proba_estimates.py`, it turned out to be the case every time.

7 Discussion

Combined Attack on Unstructured Lattices. Although our results target an unprotected implementation, they highlight an important insight: the absence of structure does not mitigate combined attacks (also known as partial key exposure attacks). Furthermore, full key recovery during the side-channel step is not necessary for a successful attack.

Further Countermeasures. The side-channel attack described in Section 3 and Section 5 relies heavily on the ternary nature of the matrices **S** and **E**. For this reason, increasing the standard deviation of the distribution used for the key generation could offer protection against such attacks. Alternatively, arithmetic masking, proven in the ISW [ISW03] model, is a reliable countermeasure, widely used in lattice-based cryptography [BBE$^+$18,BGR$^+$21,dPKPR24]. Regarding the vector **z**, masking Gaussian samplers remains a complex and performance-intensive task. In our attack, exploiting **z** is only effective when combined with information about **S**, so protecting **S** alone is sufficient to thwart our approach.

Future Works. Due to similarities with Falcon, HuFu may be vulnerable to Hidden Parallelepiped Attacks that rely solely on **z**. However, such attacks have only been performed on structured lattices. Extending them to unstructured lattices may present additional challenges.

References

ABB10. Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient Lattice (H)IBE in the Standard Model. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 553–572. Springer, Berlin, Heidelberg, May / June 2010. doi:10.1007/978-3-642-13190-5_28.

ACPS09. Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast Cryptographic Primitives and Circular-Secure Encryption Based on Hard Learning Problems. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 595–618. Springer, Berlin, Heidelberg, August 2009. doi:10.1007/978-3-642-03356-8_35.

ADH+19. Martin R. Albrecht, Léo Ducas, Gottfried Herold, Elena Kirshanova, Eamonn W. Postlethwaite, and Marc Stevens. The General Sieve Kernel and New Records in Lattice Reduction. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 717–746. Springer, Cham, May 2019. doi:10.1007/978-3-030-17656-3_25.

ADPS16. Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum Key Exchange - A New Hope. In Thorsten Holz and Stefan Savage, editors, *USENIX Security 2016*, pages 327–343. USENIX Association, August 2016.

Ajt96. Miklós Ajtai. Generating Hard Instances of Lattice Problems (Extended Abstract). In *28th ACM STOC*, pages 99–108. ACM Press, May 1996. doi:10.1145/237814.237838.

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. Springer, Cham, April / May 2018. doi:10.1007/978-3-319-78375-8_12.

BDGL16. Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. New directions in nearest neighbor searching with applications to lattice sieving. In Robert Krauthgamer, editor, *27th SODA*, pages 10–24. ACM-SIAM, January 2016. doi:10.1137/1.9781611974331.ch2.

BFM+18. Joppe W. Bos, Simon Friedberger, Marco Martinoli, Elisabeth Oswald, and Martijn Stam. Assessing the Feasibility of Single Trace Power Analysis of Frodo. In *SAC*, volume 11349 of *Lecture Notes in Computer Science*, pages 216–234. Springer, 2018.

BGR+21. Joppe W. Bos, Marc Gourjon, Joost Renes, Tobias Schneider, and Christine van Vredendaal. Masking Kyber: First- and Higher-Order Implementations. *IACR TCHES*, 2021(4):173–214, 2021. URL: https://tches.iacr.org/index.php/TCHES/article/view/9064, doi:10.46586/tches.v2021.i4.173-214.

BLNR22. Olivier Bernard, Andrea Lesavourey, Tuong-Huy Nguyen, and Adeline Roux-Langlois. Log-S-unit Lattices Using Explicit Stickelberger Generators to Solve Approx Ideal-SVP. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part III*, volume 13793 of *LNCS*, pages 677–708. Springer, Cham, December 2022. doi:10.1007/978-3-031-22969-5_23.

BR20. Olivier Bernard and Adeline Roux-Langlois. Twisted-PHS: Using the Product Formula to Solve Approx-SVP in Ideal Lattices. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part II*, volume

12492 of *LNCS*, pages 349–380. Springer, Cham, December 2020. doi:10.1007/978-3-030-64834-3_12.

BVC+23. Alexandre Berzati, Andersson Calle Viera, Maya Chartouny, Steven Madec, Damien Vergnaud, and David Vigilant. Exploiting Intermediate Value Leakage in Dilithium: A Template-Based Approach. *IACR TCHES*, 2023(4):188–210, 2023. doi:10.46586/tches.v2023.i4.188-210.

BVCV24. Alexandre Berzati, Andersson Calle Viera, Maya Chartouny, and David Vigilant. Simple Power Analysis assisted Chosen Cipher-Text Attack on ML-KEM. Cryptology ePrint Archive, Paper 2024/2051, 2024. URL: https://eprint.iacr.org/2024/2051.

CDW17. Ronald Cramer, Léo Ducas, and Benjamin Wesolowski. Short Stickelberger Class Relations and Application to Ideal-SVP. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 324–348. Springer, Cham, April / May 2017. doi:10.1007/978-3-319-56620-7_12.

CGM19. Yilei Chen, Nicholas Genise, and Pratyay Mukherjee. Approximate Trapdoors for Lattices and Smaller Hash-and-Sign Signatures. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part III*, volume 11923 of *LNCS*, pages 3–32. Springer, Cham, December 2019. doi:10.1007/978-3-030-34618-8_1.

DDGR20. Dana Dachman-Soled, Léo Ducas, Huijing Gong, and Mélissa Rossi. LWE with Side Information: Attacks and Concrete Security Estimation. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 329–358. Springer, Cham, August 2020. doi:10.1007/978-3-030-56880-1_12.

DGHK23. Dana Dachman-Soled, Huijing Gong, Tom Hanson, and Hunter Kippen. Revisiting Security Estimation for LWE with Hints from a Geometric Perspective. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part V*, volume 14085 of *LNCS*, pages 748–781. Springer, Cham, August 2023. doi:10.1007/978-3-031-38554-4_24.

DM14. Léo Ducas and Daniele Micciancio. Improved Short Lattice Signatures in the Standard Model. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 335–352. Springer, Berlin, Heidelberg, August 2014. doi:10.1007/978-3-662-44371-2_19.

dPKPR24. Rafaël del Pino, Shuichi Katsumata, Thomas Prest, and Mélissa Rossi. Raccoon: A Masking-Friendly Signature Proven in the Probing Model. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part I*, volume 14920 of *Lecture Notes in Computer Science*, pages 409–444. Springer, 2024. doi:10.1007/978-3-031-68376-3_13.

dPLS18. Rafaël del Pino, Vadim Lyubashevsky, and Gregor Seiler. Lattice-Based Group Signatures and Zero-Knowledge Proofs of Automorphism Stability. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 574–591. ACM Press, October 2018. doi:10.1145/3243734.3243852.

Duc18. Léo Ducas. Shortest Vector from Lattice Sieving: A Few Dimensions for Free. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 125–145. Springer, Cham, April / May 2018. doi:10.1007/978-3-319-78381-9_5.

EK20. Thomas Espitau and Paul Kirchner. The nearest-colattice algorithm: Time-approximation tradeoff for approx-cvp. *Open Book Series*, 4(1):251–266, 2020.

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. doi:10.46586/tches.v2022.i3.141-164.

GPV08. Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008. doi:10.1145/1374376.1374407.

GR24. Morgane Guerreau and Mélissa Rossi. A Not So Discrete Sampler: Power Analysis Attacks on HAWK signature scheme. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2024(4):156–178, 2024. URL: https://doi.org/10.46586/tches.v2024.i4.156-178, doi:10.46586/TCHES.V2024.I4.156-178.

ISW03. Yuval Ishai, Amit Sahai, and David Wagner. Private Circuits: Securing Hardware against Probing Attacks. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 463–481. Springer, Berlin, Heidelberg, August 2003. doi:10.1007/978-3-540-45146-4_27.

JRS23. Corentin Jeudy, Adeline Roux-Langlois, and Olivier Sanders. Lattice Signature with Efficient Protocols, Application to Anonymous Credentials. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part II*, volume 14082 of *LNCS*, pages 351–383. Springer, Cham, August 2023. doi:10.1007/978-3-031-38545-2_12.

JS24. Corentin Jeudy and Olivier Sanders. Improved Lattice Blind Signatures from Recycled Entropy. Cryptology ePrint Archive, Report 2024/1289, 2024. URL: https://eprint.iacr.org/2024/1289.

KAA21. Emre Karabulut, Erdem Alkim, and Aydin Aysu. Single-Trace Side-Channel Attacks on ω-Small Polynomial Sampling: With Applications to NTRU, NTRU Prime, and CRYSTALS-DILITHIUM. In *HOST*, pages 35–45. IEEE, 2021.

Kan87. Ravi Kannan. Minkowski's convex body theorem and integer programming. *Mathematics of operations research*, 12(3):415–440, 1987.

KH18. Suhri Kim and Seokhie Hong. Single Trace Analysis on Constant Time CDT Sampler and Its Countermeasure. *Applied Sciences*, 8(10), 2018. URL: https://www.mdpi.com/2076-3417/8/10/1809, doi:10.3390/app8101809.

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. Cryptology ePrint Archive, Paper 2025/351, 2025. URL: https://eprint.iacr.org/2025/351.

MN23. Alexander May and Julian Nowakowski. Too Many Hints - When LLL Breaks LWE. In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT 2023, Part IV*, volume 14441 of *LNCS*, pages 106–137. Springer, Singapore, December 2023. doi:10.1007/978-981-99-8730-6_4.

MP12. Daniele Micciancio and Chris Peikert. Trapdoors for Lattices: Simpler, Tighter, Faster, Smaller. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 700–718. Springer, Berlin, Heidelberg, April 2012. doi:10.1007/978-3-642-29011-4_41.

NIS24a. NIST. Module-Lattice-Based Digital Signature Standard. Federal Information Processing Standards Publication, NIST FIPS 204, 2024. https://doi.org/10.6028/NIST.FIPS.204.

NIS24b. NIST. Module-Lattice-Based Key-Encapsulation Mechanism Standard. Federal Information Processing Standards Publication, NIST FIPS 203, 2024. https://doi.org/10.6028/NIST.FIPS.203.

Pei10. Chris Peikert. An Efficient and Parallel Gaussian Sampler for Lattices. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 80–97. Springer, Berlin, Heidelberg, August 2010. doi:10.1007/978-3-642-14623-7_5.

PHS19. Alice Pellet-Mary, Guillaume Hanrot, and Damien Stehlé. Approx-SVP in Ideal Lattices with Pre-processing. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 685–716. Springer, Cham, May 2019. doi:10.1007/978-3-030-17656-3_24.

Reg05. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005. doi:10.1145/1060590.1060603.

SE94. Claus Peter Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Mathematical Programming*, 1994.

YJL+23. Yang Yu, Huiwen Jia, Leibo Li, Delong Ran, Zhiyuan Qiu, Shiduo Zhang, Xiuhan Lin, and Xiaoyun Wang. HuFu: Hash-and-Sign Signatures From Powerful Gadgets, 2023. URL: https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/round-1/spec-files/HuFu-spec-web.pdf.

YJW23. Yang Yu, Huiwen Jia, and Xiaoyun Wang. Compact Lattice Gadget and Its Applications to Hash-and-Sign Signatures. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part V*, volume 14085 of *LNCS*, pages 390–420. Springer, Cham, August 2023. doi:10.1007/978-3-031-38554-4_13.

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. Springer, Cham, April 2023. doi:10.1007/978-3-031-30634-1_19.

A Proof of Lemma 2.1

Proof (of Lemma 2.1). Since Q is a power of 2, invertibility modulo Q is equivalent to invertibility modulo 2. Therefore, wlog we work modulo 2.

Consider $k_1 < n - k_2$ and let $\{\mathbf{a}_1, \ldots, \mathbf{a}_{k_1+1}\}$ be random vectors in $\mathbb{Z}_2^{n-k_2}$. For $\{\mathbf{a}_1, \ldots, \mathbf{a}_{k_1+1}\}$ to be linearly independent, two conditions must hold:

1. $\{\mathbf{a}_1, \ldots, \mathbf{a}_{k_1}\}$ are linearly independent (event $\mathcal{A}_{k_1}$).
2. $\mathbf{a}_{k_1+1} \notin \mathrm{span}(\mathbf{a}_1, \ldots, \mathbf{a}_{k_1})$ (event $\mathcal{B}_{k_1}$).

Let p_{k_1} denote the probability that $\{\mathbf{a}_1, \ldots, \mathbf{a}_{k_1}\}$ are linearly independent. Thus:

$$p_{k_1+1} = p_{k_1} \cdot \Pr[\mathcal{B}_{k_1} \mid \mathcal{A}_{k_1}] = p_{k_1} \cdot \left(1 - 2^{k_1-(n-k_2)}\right).$$

Starting with $p_1 = 1 - 2^{-(n-k_2)}$, the result follows by induction. □