# Halfspace Learning for Lattice Signature Key Recovery from Signs

Marcus Brinkmann , Nicolai Kraus , and Alexander May

Ruhr-University Bochum
{marcus.brinkmann,nicolai.kraus,alex.may}@rub.de

**Abstract.** Any signature scheme has to protect its secret key via some properly chosen, secret randomness. We show that, for the lattice signatures HAWK, Falcon and ML-DSA, even minimal leakage of this randomness suffices for secret key recovery.
In particular, leaking either the Hamming weight or a single bit of any randomness coordinate allows an attacker to infer the sign of that coordinate. This corresponds to learning $\text{sign}(\langle \mathbf{b}, \mathbf{w} \rangle)$, where $\mathbf{b}$ is the secret key and $\mathbf{w}$ is public. We model key recovery from such sign information as an instance of *Learning a Halfspace*. This well-studied problem from learning theory provides a rich solution machinery, which we adapt for the cryptanalysis of lattice-based signatures.
As a first main result, we resolve the open problem of recovering the secret key in HAWK from sign leakage. At the 128-bit security level and in the noise-free setting, we recover the secret key from only 30 signatures in 10 minutes.
As a second main result, we recover the secret key in Falcon via sign leakage from only 100 signatures in under a minute. In comparison to existing attacks, this reduces the number of required signatures by a factor of 250.
As a third result, we show the first ML-DSA secret key recovery from sign leakage, which requires 190,000 signatures and completes within seconds. In comparison to existing ML-DSA attacks, we require a comparable amount of signatures, but utilize a less restrictive leakage model.
In addition, our attack is alarmingly noise-tolerant, succeeding with up to 35% noise for HAWK, 30% for Falcon, and 35% for ML-DSA, albeit requiring significantly more signatures in the noisy case.

## 1 Introduction

*Sign leakage.* It is well agreed that the main source of power leakage stems from Hamming weight differences in the observed operands, see e.g. the survey of Mangard, Oswald and Popp [MOP07].

Lattice-based signature schemes heavily rely on proper randomness $\mathbf{x} \in \mathbb{Z}^n$ to protect their secret key $\mathbf{b}$. Usually $\mathbf{x} = (x_1, \ldots, x_n)$ is sampled coordinate-wise from a discrete distribution that is close to a Gaussian centered around 0, implying that the entries $x_i$ are usually rather small in absolute value, but vary in a certain range. The standard way to represent these entries $x_i$ is in binary

two's complement, where the bit size is significantly larger than the entropy of $x_i$.

As a result, positive entries $x_i$ tend to have a small Hamming weight in binary two's complement representation, whereas negative $x_i$ tend to have comparatively large Hamming weight. Such a Hamming weight gap can be exploited via Simple Power Analysis (SPA), or Differential Power Analysis (DPA). Guerreau and Rossi [GR24] demonstrate that for HAWK's randomness $\mathbf{x} = (x_1, \ldots, x_n)$, SPA reliably determines whether $x_i \geq 0$, or $x_i < 0$. In other words, SPA leaks $\mathrm{sign}(x_i)$, where

$$\mathrm{sign}(x_i) = \begin{cases} +1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases}.$$

However, it was left as an open problem in [GR24] whether such a sign leakage is sufficient for secret key recovery. Guerreau and Rossi suggested to use the hints framework of [DDGR20], but realized that this framework is not applicable to the HAWK setting without significant modifications.

We solve this problem by introducing methods from learning theory for *Learning a Halfspace* that recover not only HAWK's secret key efficiently but also apply to Falcon and ML-DSA.

*Learning a Halfspace, noisefree.* Let $\mathbf{b} \in \mathbb{R}^n$ be a secret vector, in our case a secret lattice vector. In the *Learning a Halfspace (LH) problem* [BFKV96] we have to learn $\mathbf{b}$ from *LH-samples*

$$(\mathbf{w}_i, y_i) := (\mathbf{w}_i, \mathrm{sign}(\langle \mathbf{b}, \mathbf{w}_i \rangle)) \in \mathbb{R}^n \times \{-1, 1\}$$

for $i = 1, \ldots, m$, where the $\mathbf{w}_i$, called *examples*, are randomly chosen from some probability distribution. The $y_i \in \{-1, 1\}$ are called *labels*.

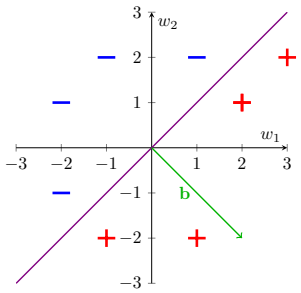


Fig. 1: Learning a Halfspace.

A two-dimensional illustration of *Learning a Halfspace* is given in Figure 1. We obtain examples $\mathbf{w} = (w_1, w_2)$ with either positive labels $y_i = \mathrm{sign}(\langle \mathbf{b}, \mathbf{w} \rangle) \geq 0$ (red plus), or negative labels $y_i < 0$ (blue minus). Our goal is to determine the line separating positive from negative labels, thereby recovering the direction of $\mathbf{b}$, perpendicular to the separation line. Note that by the limited sign information in *Learning a Halfspace* it is impossible to recover the magnitude of $\mathbf{b}$. This is also seen by observing that $\mathrm{sign}(\langle \mathbf{b}, \mathbf{w} \rangle) = \mathrm{sign}(\langle c\mathbf{b}, \mathbf{w} \rangle)$ for any real constant $c \geq 0$. Thus, we can only hope to recover $c\mathbf{b}$ for some unknown constant $c$, and then *rescale* back to our desired $\mathbf{b}$. We show that *rescaling* can be done in cryptographic settings, where we know the distribution of the secret key $\mathbf{b}$.

*Related work.* The problem of sign leakage has been frequently addressed in the cryptographic literature, e.g. in the context of Falcon side-channel analysis [KA21, GMRR22, ZLYW23, LZY+25, LTYZ25].

Guerreau, Martinelli, Ricosset and Rossi [GMRR22] transform their side-channel leakage into a Hidden Parallelepiped problem and use the algorithm of Nguyen and Regev [NR06] for key recovery. Zhang, Lin, Yu and Wang [ZLYW23] already model sign leakage as a *Learning a Halfspace* problem. However, they transform this into a more restrictive *Learning a Slice* problem, which they solve with a method based on the spectral decomposition of the covariance of filtered signatures. The rich literature for solving *Learning a Halfspace* (LH) [BFKV96, KKMS05, FGKP06, GR06] has been widely ignored.

Side-channel attacks on ML-DSA (Dilithium) [LZS+20, BVC+23, UMTS24, KPLG24, WNGD24, DKM+25, HNP25, DFM+25] so far require a significantly stronger leakage model. The authors of [LZS+20, DKM+25] use (error-free) leakage of a single bit of $\mathbf{x}$ for millions of signatures. Such an explicit bit leakage allows to reduce secret key recovery to an Integer LWE problem that in turn is solved via Ordinary Least Squares regression [BDE+18]. However, no ML-DSA attack for a more realistic sign leakage attack on $\mathbf{x}$ is known so far.

*Masking Protection.* Due to the aforementioned bulk of work on side-channel analysis of lattice based signature schemes, there is already a high level of awareness of the necessity to properly mask the involved randomness. However, despite significant efforts by the cryptographic community [KA24, CC24, LZY+25, KLS+25] proper masking protection remains a challenge, especially for sampling non-uniform randomness. In fact, the specifications of Falcon [FHK+18] and HAWK [BBD+23] explicitly state that no masking is known for their Gaussian samplers.

**Our results.** We show that for lattice-based signature schemes – such as HAWK, Falcon, and ML-DSA – sign leakage of their randomness $\mathbf{x}$ directly leads to instances of the *Learning a Halfspace* (LH) problem. Thus, solving LH is the central problem to recover the secret key for all schemes. To make similarities between the considered schemes visible, in the following we stick to the notation of HAWK and adapt its notation (for signature variables) to Falcon and ML-DSA accordingly.

*HAWK sign leak as LH.* A HAWK signature has the form $\mathbf{s} = \frac{1}{2}(\mathbf{h} - \mathbf{w})$, where $\mathbf{h} = \text{hash}(\mathbf{m})$ and $\mathbf{Bw} = \mathbf{x}$ for the secret key $\mathbf{B} \in \mathbb{Z}^{2n \times 2n}$ and some randomness $\mathbf{x} \in \mathbb{Z}^{2n}$. This implies $\mathbf{B}(\mathbf{h} - 2\mathbf{s}) = \mathbf{x}$. As a consequence, each of $\mathbf{B}$'s rows $\mathbf{b}$ implies an identity

$$\langle \mathbf{b}, \mathbf{h} - 2\mathbf{s} \rangle = x,$$

for some coordinate $x$ of $\mathbf{x} \in \mathbb{Z}^n$. Thus, setting the example $\mathbf{w} = \mathbf{h} - 2\mathbf{s}$ and leaking the label $y = \text{sign}(x) = \text{sign}(\langle \mathbf{b}, \mathbf{w} \rangle)$ (via side-channel) gives us an LH-sample $(\mathbf{w}, y)$ for $\mathbf{b}$. Since by the ring structure of HAWK a single row $\mathbf{b}$ suffices to recover all of $\mathbf{B}$, a solution of the LH problem recovers the HAWK secret key.

*Falcon sign leak as (noisy) LH.* A Falcon signature has the form $\mathbf{s} = (\mathbf{h}, 0^n) - \mathbf{x} \cdot \mathbf{B} \in \mathbb{Z}^{2n}$, where $\mathbf{h} = \text{hash}(\mathbf{m})$ is the hashed message, $\mathbf{x}$ some randomness

and $\mathbf{B} \in \mathbb{R}^{2n \times 2n}$ the secret key. It follows that $((\mathbf{h}, 0^n) - \mathbf{s}) \cdot \mathbf{B}^{-1} = \mathbf{x}$. As a consequence, every column $\mathbf{b}$ of $\mathbf{B}^{-1}$ implies an identity

$$\langle \mathbf{b}, (\mathbf{h}, 0^n) - \mathbf{s} \rangle = x,$$

for some coordinate $x$ of $\mathbf{x} \in \mathbb{Z}^{2n}$. Thus, leaking label $\mathrm{sign}(x)$ yields an LH-sample $(\mathbf{w}, y) = ((\mathbf{h}, 0^n) - \mathbf{s}, \mathrm{sign}(x))$ for $\mathbf{b}$. Analogous to HAWK, Falcon's ring structure allows to recover the full secret key $\mathbf{B}$ from a single column $\mathbf{b}$.

Let us further refine $\mathbf{b} = (\mathbf{b}^{(1)}, \mathbf{b}^{(2)}) \in \mathbb{R}^n \times \mathbb{R}^n$ and $\mathbf{s} = (\mathbf{s}^{(1)}, \mathbf{s}^{(2)}) \in \mathbb{Z}^n \times \mathbb{Z}^n$. Then

$$\langle \mathbf{b}^{(1)}, \mathbf{h} - \mathbf{s}^{(1)} \rangle = x + \langle \mathbf{b}^{(2)}, \mathbf{s}^{(2)} \rangle.$$

For Falcon parameters, the coefficients of $\mathbf{h}$ are significantly larger than the coefficients of $\mathbf{s}^{(1)}, \mathbf{s}^{(2)}$. This allows us to define a *noisy* LH-sample

$$(\mathbf{w}, y) = (\mathbf{h} - \mathbf{s}^{(1)}, \mathrm{sign}(x)) \text{ for } \mathbf{b}^{(1)},$$

where we treat $\langle \mathbf{b}^{(2)}, \mathbf{s}^{(2)} \rangle$ as a noise term.

Solving this noisy LH instance yields $\mathbf{b}^{(1)}$. Interestingly, the Falcon public key allows us to directly recover $\mathbf{b}^{(2)}$ from $\mathbf{b}^{(1)}$.

*ML-DSA (Dilithium) sign leak as LH.* An ML-DSA signature $\mathbf{s} \in \mathbb{Z}^{\ell n}$ has coordinates satisfying $s = \langle \mathbf{b}, \mathbf{h} \rangle + x \in \mathbb{Z}$, where $\mathbf{h} = \mathrm{hash}(\mathbf{m}) \in \mathbb{Z}^n$ is the hashed message, $\mathbf{b} \in \mathbb{Z}^n$ an LWE secret (sub-)key[1], and $x \in \mathbb{Z}$ some randomness. Therefore,

$$\langle (1, \mathbf{b}), (s, -\mathbf{h}) \rangle = x.$$

Setting $(\mathbf{w}, y) = ((s, -\mathbf{h}), \mathrm{sign}(x)) \in \mathbb{Z}^{n+1} \times \{-1, 1\}$ defines an LH instance for the secret $(1, \mathbf{b})$.

*Solving LH, noise-free and noisy.* Not only is LH an extremely central problem to which many signature leakages can be easily transformed, it also offers efficient and robust solvers.

In the noise-free setting, our LH-samples $(\mathbf{w}, y)$ for $\mathbf{b}$ directly define linear inequalities

$$\begin{aligned} \langle \mathbf{b}, \mathbf{w} \rangle &\geq 0 \text{ if } y = 1, \\ \langle \mathbf{b}, \mathbf{w} \rangle &< 0 \text{ if } y = (-1), \end{aligned}$$

that can be solved via Linear Programming (LP). However, LP cannot handle *noisy* LH-instances.

Notice that noise in LH-samples may originate from different sources. As a primary source, side-channel analysis relies on noisy measurements that lead to flipped labels with a certain noise rate $\rho$. Therefore, we model the noisy LH case via LH-samples of the form $(\mathbf{w}, (-1)^c \cdot \mathrm{sign}(\langle \mathbf{b}, \mathbf{w} \rangle))$ for a Bernoulli distributed bit $c \in \mathrm{Ber}(\rho)$. The better the masking protection of our cryptographic scheme, the

[1] ML-DSA's module-LWE structure splits the secret key in $\ell$ length-$n$ sub-keys.

larger $\rho$ usually is. As a secondary source, as seen by the aforementioned Falcon LH attack, it can be (highly) beneficial to directly model secret key recovery as a noisy rather than a noise-free LH instance.

Thus, it is of great importance to identify LH solvers that are robust to noise. We propose to use Logistic Regression (LR). First, LR is extremely efficient, handling a few thousand LH-samples in a matter of seconds, see also Table 1. Second, in some cases (e.g. our ML-DSA leak attack) LR requires even in the noise-free case significantly less signatures than LP. Third and most important, LR is surprisingly robust to noise, thereby allowing for our novel Falcon attack via noisy LH modeling, and tolerating large noise rates up to $\rho = 35\%$ that might appear in real-world measurements.

*Rescaling and success rate.* As discussed before, LH solvers such as LP and LR naturally only determine the direction of $\mathbf{b}$, but cannot recover its magnitude. However, knowledge of the secret key distribution in cryptographic settings allows us to propose an efficient rescaling of LH solutions. After rescaling we round coordinate-wise, thereby recovering the secret key in case sufficiently many LH-samples were provided.

For noise rate $\rho = 0$, we always succeed to recover the complete secret key of HAWK, Falcon and ML-DSA, simply by rescaling and rounding. For $\rho > 0$, we still recover for HAWK and ML-DSA all secret keys, while for Falcon the success rate decreases with an increasing noise level.

We do not apply other techniques such as enumeration or lattice reduction. However, one could use our attack in a hybrid approach as in [HMS+23] to increase the success rate for Falcon or to reduce the required number of samples.

*HAWK key recovery from sign leak, including a first end-to-end attack.* We ran our HAWK secret key recovery attack on error-free LH-samples ($\rho = 0$). In this case, we recover the secret key with only 30 signatures, which is an extremely small amount compared to other successful key recovery attacks on lattice signature schemes in the literature, see also Table 1.

We also ran our attack using the (unprotected) HAWK reference implementation, for which we recorded power traces with a ChipWhisperer Husky, and obtained the first automatic end-to-end key recovery. Crucially, our attack does not only identify the sign leakage for HAWK's randomness (with experimental noise level $\rho < 3.5\%$), but also a specific operation called *symbreak*. The *symbreak* operation flips the sign in order to prevent weak forgery attacks. Therefore, it is crucial to identify *symbreak*, otherwise we receive random labels. We show that in the reference implementation *symbreak* can be reliably detected via a side-channel. Our practical end-to-end key recovery attack on HAWK recovers the secret key from an embedded platform using only 1,600 HAWK signatures.

*Falcon key recovery from sign leak.* For Falcon, the prior work of Zhang, Lin, Yu, Wang [ZLYW23] and Zhang, Yu, Wang, You, Xu, Wang [LZY+25] requires error-free sign leakage of 170,000, respectively 25,000 signatures to recover the secret key with a 25% success rate. In contrast, our LR solver has a 100% success rate

for error-free sign leakage and requires only 100 Falcon signatures on average. Thus, we improve upon the attacks of [ZLYW23, LZY+25] in the number of signatures by large multiplicative factors of 1700 and 250, respectively.

Our drastic signature count improvement can be explained by the general applicability of our LH approach. The attack of [ZLYW23] can only exploit sign leakage in a single, specific coordinate of the randomness per signature. This is improved by the attack of [LZY+25] to four coordinates per signature. In contrast, our attack easily exploits sign leakage in all $n$ coordinates, using Falcon's ring structure. Thus, every Falcon signature on the 128-bit security levels provides us 512 LH-samples, see also Table 1.

In a real-world attack on some randomness $\mathbf{x} = (x_1, \ldots, x_n)$, it is natural to assume that an attacker tries to extract the full $n$ possible sign informations of all $x_i$. However, our attack would also work if for some reason only a single $\mathrm{sign}(x_i)$ is extractable, at the cost of an $n$ times increase in signatures. In such a single-sample extraction regime, our attack still requires 2 times, respectively 3.3 times fewer signatures than [ZLYW23, LZY+25].

*ML-DSA key recovery from sign leak.* We introduce the first sign leakage attack on ML-DSA, for which we recover the secret key from 190,000 signatures. Other key recovery attacks [LZS+20, DKM+25] succeed with a comparable order of signatures, but require a significantly more restrictive leakage model, where a single (explicit) bit of ML-DSA's randomness is leaked. The large amount of required signatures for our LH attack (in comparison to HAWK and Falcon) is due to ML-DSA signatures yielding a non-trivial sign leakage only with small probability.

Table 1: Key recovery from sign leakage for lattice signatures (128-bit security, noise-free setting). There exists no prior work for HAWK and ML-DSA.

| Scheme | Authors | Signatures | Samples | Time | Solver | Success Rate |
|---|---|---|---|---|---|---|
| HAWK | this work | 30 | 15k | 10 min | LP | 100 % |
| Falcon | [ZLYW23] | 170k | 170k | 30 min | – | 25 % |
| | [LZY+25] | 25k | 100k | – | – | 25 % |
| | this work | 100 | 51k | 10 sec | LR | 100 % |
| ML-DSA | this work | 190k | 29k | 5 sec | LR | 100 % |

*Robustness of LH attacks to noise.* Our LR solver handles even large noise levels $\rho$ surprisingly well. In the 128-bit security level, we recovered the HAWK secret key with a success rate of 100% for $\rho = 35\%$ in 120 minutes using 110k signatures. For Falcon, we achieved a 20% success rate for $\rho = 15\%$ in 2 minutes using 5.1k signatures, and a 5% success rate for $\rho = 25\%$ in 4 minutes using 10k signatures.

For ML-DSA, we recovered the key with a success rate of 100% for $\rho = 35\%$ in 3 minutes using the sign information from 90 million signatures.

*Artifacts.* An implementation of the attack, a script for acquiring side-channel traces from the ChipWhisperer STM32L562, and a set of precollected traces is available at https://github.com/nicolkraus/halfspace.

*Outline.* After introducing some notation in Section 2, we formulate in Section 3 the Learning a Halfspace (LH) problem and its cryptanalytic use. Section 4 presents our first LH application, a novel key recovery attack on the HAWK signature scheme by exploiting sign leakage. In Section 5 we adapt the LH attack to Falcon, which greatly improves over previous key recovery attacks. In Section 6, we present the first sign leakage ML-DSA attack. Whereas we initially evaluate our LH attacks for HAWK, Falcon and ML-DSA on simulated leak data, in Section 7 we provide a practical end-to-end key recovery attack on the HAWK reference implementation.

## 2 Preliminaries

We denote vectors by bold lowercase letters, e.g., $\mathbf{x}$, and matrices by bold uppercase letters, e.g., $\mathbf{B}$. The transpose of a vector or matrix is written as $\mathbf{x}^T$ or $\mathbf{B}^T$. Inner products are written as $\langle \mathbf{a}, \mathbf{b} \rangle$ and the Euclidean norm is denoted as $\|\mathbf{x}\| = \|\mathbf{x}\|_2$. The sign function $\mathrm{sign}(\cdot)$ returns $+1$ for nonnegative inputs and $-1$ for negative inputs.

The standard representation of integers in software is the binary *two's complement* form. A $w$-bit word then represents values in the range $[-2^{w-1}, 2^{w-1})$ by interpreting the bit string $(x_{w-1}, \ldots, x_0)$ as the integer $\sum_{j=0}^{w-1} x_j 2^j$, where values $\geq 2^{w-1}$ are reinterpreted as negative numbers by subtracting $2^w$. For $0 \leq j < w$, we call $x_j$ the *$j$-th bit* of the representation of $x$.

Let $n$ be a power of two and set $K_n = \mathbb{Q}[X]/(X^n + 1)$. For $f \in K_n$, we write $f = f_0 + f_1 X + \cdots + f_{n-1} X^{n-1}$ and define the coefficient embedding $\mathrm{vec} : K_n \to \mathbb{Q}^n, \quad f \mapsto (f_0, \ldots, f_{n-1})$. We freely identify elements of $K_n$ with their coefficient vectors.

We denote the Hermitian adjoint of an element $f \in K_n$ by $f^\star = f_0 - \sum_{i=1}^{n-1} f_{n-i} X_i$. We extend the notations to elements $\mathbf{B} \in K_n^{r \times r}$ : $\mathbf{B}^\star$ denotes the entry-wise adjoint of $\mathbf{B}^\top$. For any element $\mathbf{B} \in K_n^{r \times r}$, we call the Gram matrix of $\mathbf{B}$ the matrix $\mathbf{Q} = \mathbf{B}^\star \mathbf{B}$.

For a matrix $\mathbf{B} \in K_n^{r \times r}$, set $\mathbf{Q} = \mathbf{B}^\star \mathbf{B}$. We define the $\mathbf{Q}$-inner product on $K_n^r$ by $\langle \mathbf{f}, \mathbf{g} \rangle_{\mathbf{Q}} = \frac{1}{n} \mathrm{Tr}(\mathbf{f}^\star \mathbf{Q} \mathbf{g})$, where Tr is the algebraic trace over the field extension $K_n/\mathbb{Q}$. The corresponding norm, for all $\mathbf{f} \in K_n^r$, is $\|\mathbf{f}\|_{\mathbf{Q}} = \sqrt{\langle \mathbf{f}, \mathbf{f} \rangle_{\mathbf{Q}}}$. In particular, we have $\|\mathbf{B}\mathbf{f}\| = \|\mathbf{f}\|_{\mathbf{Q}}$.

## 3 Learning a Halfspace

In this section, we introduce *Learning a Halfspace* as a central problem for secret key recovery in cryptanalysis of lattice-based signature schemes. The problem

is also known as *binary linear classification* or *perceptron*. When precision is required, we adopt standard statistical notation; but we aim to keep the presentation accessible.

A linear classifier computes a linear function of some input $\mathbf{w}$, and assigns a label $y \in \{-1, 1\}$, describing whether or not the resulting value is positive or negative. Input vectors $\mathbf{w} \in \mathbb{R}^n$ are called *examples*. The secret $\mathbf{b} \in \mathbb{Z}^n$ determines a *decision boundary*

$$h := \{\mathbf{w} \in \mathbb{R}^n : \langle \mathbf{b}, \mathbf{w} \rangle = 0\}, \tag{1}$$

which is the hyperplane that separates the example space into two regions corresponding to labels $+1$ and $-1$. These two regions, $\{\mathbf{w} \in \mathbb{R}^n : \langle \mathbf{b}, \mathbf{w} \rangle > 0\}$ with label $y = 1$ and $\{\mathbf{w} \in \mathbb{R}^n : \langle \mathbf{b}, \mathbf{w} \rangle < 0\}$ with label $y = (-1)$, are called *halfspaces*. An example $\mathbf{w}$ lies in the same halfspace as $\mathbf{b}$ iff its label is $y = 1$, see also Figure 2.

To recover $\mathbf{b}$, we are given a collection of pairs $(\mathbf{w}_i, y_i)$, $i > 0$, called *LH-samples*. Importantly, the labels are not guaranteed to be correct; each label is corrupted by some random classification noise $\rho \in [0, 0.5)$, meaning that an incorrect label is assigned with probability $\rho$. More precisely, we wish to solve the following problem.

**Definition 1 (Learning a Halfspace (LH)).** *Let $\mathbf{b} \in \mathbb{Z}^n$ be a secret key, sampled coordinate-wise from some known distribution. We obtain an oracle for* LH-samples *$(\mathbf{w}_i, y_i) \in \mathbb{R}^n \times \{-1, 1\}$. Let $\rho \in [0, 0.5)$ be the* noise rate *of some* random classification noise. *The labels are defined*

$$y_i = \begin{cases} \operatorname{sign}(\langle \mathbf{b}, \mathbf{w}_i \rangle) & \text{with probability } 1 - \rho \\ -\operatorname{sign}(\langle \mathbf{b}, \mathbf{w}_i \rangle) & \text{with probability } \rho \end{cases}.$$

*We call the case $\rho = 0$* noise-free *LH, whereas $\rho > 0$ is called the* noisy *LH case. Our goal is to recover the secret vector $\mathbf{b}$.*

*Remark 2 (LH-sample generation).* In subsequent sections we show how signatures in HAWK, Falcon and ML-DSA provide us an oracle for LH-samples. Whereas every HAWK or Falcon signature provides $n$ LH-samples, ML-DSA signatures only provide LH-samples with small probability.

*Scale invariance.* Note that in *Learning a Halfspace*, the magnitude of the secret $\mathbf{b}$ is not identifiable from the LH-samples alone. Indeed, for any scalar $c > 0$ we have

$$\operatorname{sign}(\langle \mathbf{b}, \mathbf{w} \rangle) = \operatorname{sign}(c \cdot \langle \mathbf{b}, \mathbf{w} \rangle) = \operatorname{sign}(\langle c\mathbf{b}, \mathbf{w} \rangle).$$

Thus, the sign information depends only on the direction of $\mathbf{b}$, not on its magnitude.

This *scale invariance* is also evident from Figure 2: the decision boundary $h$ defined by $\mathbf{b}$ via Eq. (1) is identical for all positive multiples of $\mathbf{b}$. Consequently,

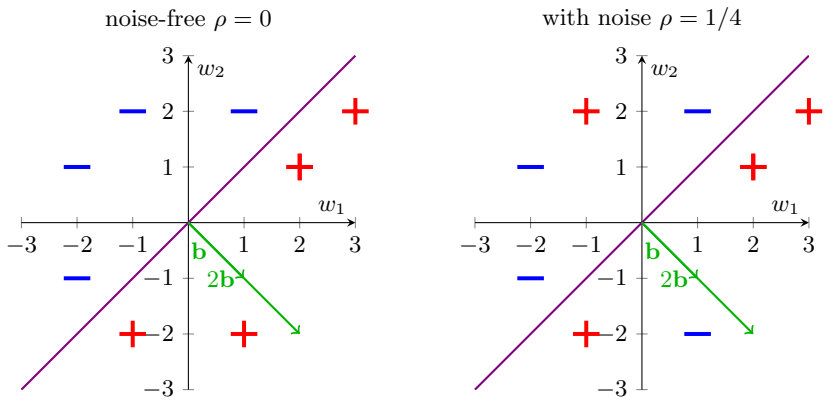



Fig. 2: A visualization of the *Learning a Halfspace* problem. The secret is $\mathbf{b} = (1, -1)$, which defines the decision boundary $h$ shown in violet. The plus symbols correspond to examples $\mathbf{w} = (w_1, w_2)$ with positive label, while minus symbols correspond to examples with negative label. The left panel shows the noise-free case, and the right panel illustrates a noisy setting where $\rho = 1/4$ of the labels are incorrect.

if only the signs are observed *and no additional information about the secret* $\mathbf{b}$ *is present*, one cannot distinguish between these multiples.

This is the reason that we included in Definition 1 the *known distribution* of $\mathbf{b}$, which is the typical case in cryptographic settings. We show in Section 3.2 how to infer the magnitude of $\mathbf{b}$ from its direction and distribution.

### 3.1 Secret Key Recovery – Direction

*Solving noise-free LH.* In the absence of noise ($\rho = 0$), the problem is solved by finding a vector $\mathbf{u}$ that satisfies all sign constraints simultaneously:

$$\text{find } \mathbf{u} \in \mathbb{R}^n \text{ such that } \begin{cases} \langle \mathbf{u}, \mathbf{w}_i \rangle \geq 0 & \text{if } y_i = +1, \\ \langle \mathbf{u}, \mathbf{w}_i \rangle < 0 & \text{if } y_i = -1, \end{cases} \quad \text{for all } i. \tag{2}$$

This is a feasibility problem over a system of linear inequalities. Standard linear-programming (LP) methods can efficiently find a solution. However, a single erroneous label renders the system infeasible, see also the noisy case in Figure 2 which has no separating hyperplane for all plus and minus symbols.

*Solving LH in general.* When the observed signs are corrupted with random noise $\rho > 0$, the constraints can no longer be satisfied simultaneously. Instead, the objective becomes to find a hyperplane that minimizes the number of misclassified LH-samples.

To this end, we need to introduce the concept of *classifiers* and *loss functions*. A *linear classifier* is a function $f_{\mathbf{u}} : \mathbb{R}^n \to \{-1, +1\}$ that assigns to an example $\mathbf{w}$ a classification $f_{\mathbf{u}}(\mathbf{w})$. A loss function now measures how accurately the classification matches the labels.

More precisely, a loss function is a map $\ell : (\mathbb{R}^n)^2 \times \{-1, +1\} \to \mathbb{R}_{\geq 0}$, where $\ell(\mathbf{u}, \mathbf{w}, y)$ quantifies the penalty incurred when classifying as $f_{\mathbf{u}}(\mathbf{w})$ while the LH label is $y = (-1)^c \text{sign}(\langle \mathbf{b}, \mathbf{w} \rangle)$ for $c \in \text{Ber}(\rho)$, called a *misclassification*. Minimizing the amount of misclassified LH-samples corresponds to minimizing the *0–1 loss*, which for a single LH-sample $(\mathbf{w}, y)$ is defined as

$$\ell_{0-1}(\mathbf{u}, \mathbf{w}, y) = \begin{cases} 1, & f_{\mathbf{u}}(\mathbf{w}) \neq y_i \\ 0, & f_{\mathbf{u}}(\mathbf{w}) = y_i \end{cases}.$$

In other words, the 0–1 loss takes value 0 for correct classifications and 1 for misclassifications. Assume that we are given a distribution on the space of examples $\mathbf{w}$. Then we want to determine a classifier $f_{\mathbf{u}}$ that minimizes the *expected 0-1 loss* $\mathbb{E}_w[\ell_{0-1}(\mathbf{u}, \mathbf{w}_i, y_i)]$. It is a fundamental observation in learning theory that even in the noisy case $0 < \rho < \frac{1}{2}$ the linear classifier $f_{\mathbf{b}} = \text{sign}(\langle \mathbf{b}, \mathbf{w} \rangle)$ minimizes the expected loss, and is the unique linear minimizer up to scaling of $\mathbf{b}$. Theorem 3 formalizes this observation. Geometrically speaking, the decision boundary $h$ that minimizes the number of misclassifications is the same in both the noise-free and noisy settings shown in Figure 2.

**Theorem 3 (Direction of b).** *We are given an LH problem with secret* $\mathbf{b}$*, where the oracle chooses* $\mathbf{w}$ *according to some probability distribution over* $\mathbb{R}^n$ *with labels* $y = (-1)^c \text{sign}(\langle \mathbf{b}, \mathbf{w} \rangle)$*, where* $c \in \text{Ber}(\rho)$ *and* $\rho < 1/2$*. Every linear classifier* $f_{\mathbf{u}^\star}$ *that minimizes the expected 0–1 loss is of the form*

$$f_{\mathbf{u}^\star}(\mathbf{w}) = \text{sign}(\langle \mathbf{u}^\star, \mathbf{w} \rangle),$$

*where* $\mathbf{u}^\star = c\mathbf{b}$ *for some scalar* $c > 0$*.*

*Proof.* Let $Y_{\mathbf{w}}$ denote a random variable for label $\mathbf{w}$. In order to minimize the expected 0–1 loss, an *optimal linear classifier* $f_\star : \mathbb{R}^n \to \{-1, 1\}$ is a decision rule that classifies $\mathbf{w}$ with the most likely label, i.e.,

$$\begin{aligned} f_\star(\mathbf{w}) &\coloneqq \begin{cases} +1, & \text{if } \Pr(Y = 1|\mathbf{w}) \geq \Pr(Y = -1|\mathbf{w}) \\ -1, & \text{if } \Pr(Y = 1|\mathbf{w}) < \Pr(Y = -1|\mathbf{w}) \end{cases} \\ &= \text{sign}\left(\Pr(Y = 1|\mathbf{w}) - \frac{1}{2}\right). \end{aligned} \tag{3}$$

For any fixed $\mathbf{w}$, we have

$$\Pr(Y = 1|\mathbf{w}) = \begin{cases} 1 - \rho, & \langle \mathbf{b}, \mathbf{w} \rangle \geq 0, \\ \rho, & \langle \mathbf{b}, \mathbf{w} \rangle < 0. \end{cases}$$

Since $\rho < \frac{1}{2}$, we have $\Pr(Y = 1|\mathbf{w}) > \frac{1}{2}$ exactly when $\langle \mathbf{b}, \mathbf{w} \rangle \geq 0$, and $\Pr(Y = 1|\mathbf{w}) < \frac{1}{2}$ otherwise. Thus, an optimal classifier is

$$f_\star(\mathbf{w}) = \operatorname{sign}\big(\Pr(Y = 1|\mathbf{w}) - \tfrac{1}{2}\big) = \operatorname{sign}(\langle \mathbf{b}, \mathbf{w} \rangle).$$

It remains to show uniqueness up to positive scaling. Suppose there exists another optimal classifier of the form $f_\star(\mathbf{w}) = \operatorname{sign}(\langle \mathbf{u}^\star, \mathbf{w} \rangle)$ with $\mathbf{u}^\star$ not a positive multiple of $\mathbf{b}$. Then the two halfspaces

$$\{\mathbf{w} : \langle \mathbf{u}^\star, \mathbf{w} \rangle > 0\} \quad \text{and} \quad \{\mathbf{w} : \langle \mathbf{b}, \mathbf{w} \rangle > 0\}$$

define different decision regions. Their separating hyperplanes are distinct, so their symmetric difference contains an open cone. Because the distribution of $\mathbf{w}$ has full-dimensional support, this cone must contain points where $\mathbf{u}^\star$ and $\mathbf{b}$ disagree on the label. This contradicts optimality of the optimal classifier.

Hence $\mathbf{u}^\star = c\mathbf{b}$ for some $c > 0$, completing the proof. □

*Relaxation 1: Replace 0–1 loss by a convex surrogate loss.* By Theorem 3, minimization of the 0–1 loss reveals the direction of $\mathbf{b}$. However, the 0–1 loss is non-convex, discontinuous, and its exact minimization is known to be NP-hard [ABSS97]. Consequently, minimizing it directly becomes intractable in high-dimensional settings. A standard approach in learning theory to resolve this issue is to replace the 0–1 loss with a *convex surrogate loss*, which approximates the classification objective while allowing efficient optimization with gradient-based methods.

*Surrogate loss functions.* The most established surrogate losses are

$$\begin{aligned} &\text{hinge loss } \ell_{hinge}(\mathbf{u}, \mathbf{w}, y) = \max\{1 - \langle \mathbf{u}, \mathbf{w} \rangle y, 0\}, \\ &\text{squared loss } \ell_{squared}(\mathbf{u}, \mathbf{w}, y) = (\langle \mathbf{u}, \mathbf{w} \rangle - y)^2, \text{ and} \\ &\text{logistic loss } \ell_{log}(\mathbf{u}, \mathbf{w}, y) = \ln 2^{-1} \ln(1 + \exp(-\langle \mathbf{u}, \mathbf{w} \rangle y)). \end{aligned} \tag{4}$$

Notice that we may write

$$\ell_{0\text{–}1}(\mathbf{u}, \mathbf{w}, y) = \frac{1}{2}\left(1 - \operatorname{sign}(\langle \mathbf{u}, \mathbf{w} \rangle) y\right).$$

Thus, the hinge loss can be seen as a straight-forward continuous adaptation of the 0–1 loss, by removing the non-continuous sign function.

See Figure 3 for a comparison of the loss functions. We observe that all surrogate loss functions upper bound the 0–1 loss. As a consequence, minimizing one of these surrogate losses also minimizes the 0–1 loss. Moreover, the hinge and the logistic loss approximate the 0–1 loss quite well, whereas the squared loss puts an undesired penalty on large margins. This is the reason that Ordinary Least Squares regression methods that were successfully applied in other cryptanalytic settings [BDE+18, LZS+20, DKM+25] do not work well for LH.

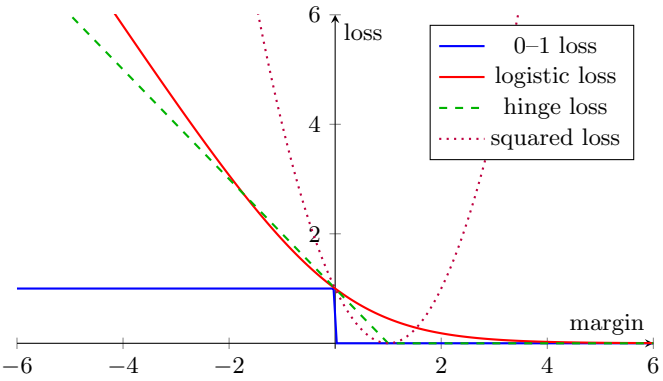



Fig. 3: Losses as a function of the *margin*. For an LH-sample $(\mathbf{w}, y)$, the margin is defined as the distance $y\langle \mathbf{b}, \mathbf{w} \rangle$ to the decision boundary, where $\mathbf{b}$ is the secret key. A positive margin indicates a correctly classified point, and a negative margin indicates a misclassification.

*Relaxation 2: Computing with a limited number of LH-samples.* By Theorem 3 a minimizer of the expected 0–1 loss $\mathbb{E}_w[\ell_{0-1}(\mathbf{u}, \mathbf{w}_i, y_i)]$ yields the direction of $\mathbf{b}$. However, this expectation is taken over all possible example vectors $\mathbf{w}$, which makes its computation infeasible.

In practice, we obtain a limited number $m$ of LH-samples, and may minimize the loss with respect to these samples.

**Definition 4 (Empirical minimizer).** *Let $\ell$ be a loss function, and let $(\mathbf{w}_i, y_i)$, $i = 1, \ldots, m$ be LH-samples. The empirical minimizer is defined as*

$$\hat{\mathbf{u}} := \arg\min_{\mathbf{u} \in \mathbb{R}^n} \sum_{i=1}^{m} \ell(\mathbf{u}, \mathbf{w}_i, y_i)\,.$$

A major result in learning theory is that the empirical minimizer of any of the aforementioned surrogate loss functions yields a classifier that is optimal with respect to the 0–1 loss [BJM06, Zha04, RDVC+04]. Intuitively, this means that as the number of LH-samples grows, all such losses converge to the optimal classifier. However, they may differ in the number of samples required.

*Logistic regression.* In experiments, we compared the hinge loss, square loss, and logistic loss. Among these, as expected by Figure 3 the hinge loss and the logistic loss (by far) outperformed the square loss in the number $m$ of required LH-samples. The hinge loss usually required a number of LH-samples comparable to the logistic loss, but its computation was significantly more expensive and thus not applicable in high (cryptographic) dimensions. For this reason we decided to focus on the logistic loss $\ell_{log}$, as defined in Eq. (4), throughout the paper.

We call any LH solver that minimizes the logistic loss a *Logistic Regression* (LR). Our LH solver is given in Algorithm 1, where for the noise-free LH case one can use LP as an alternative to LR.

**Algorithm 1:** SolveLH($\widetilde{L}, solver$)

**Input :** List $\widetilde{L}$ of LH-samples $(\mathbf{w}, y)$ with
example $\mathbf{w} \in \mathbb{R}^n$,
label $y \in \{-1, 1\}$,
LH $solver \in \{LR, LP\}$.

**Output:** Multiple of the secret key $\mathbf{u}^\star = c\mathbf{b} \in \mathbb{R}^n$, where $c \in \mathbb{R}_{\geq 0}$.

$(\mathbf{w}_i, y_i) \leftarrow \widetilde{L}$.
**if** $solver = LR$ **then** // Logistic Regression
  $\mathbf{u}^\star \leftarrow \arg\min_{\mathbf{u}\in\mathbb{R}^n} \sum_{i=1}^m \ell_{log}(\mathbf{u}, \mathbf{w}_i, y_i)$
**else if** $solver = LP$ **then** // Linear Programming
  find $\mathbf{u}^\star \in \mathbb{R}^n$ such that $\begin{cases} \langle \mathbf{u}^\star, \mathbf{w}_i \rangle \geq 0 & \text{if } y_i = +1, \\ \langle \mathbf{u}^\star, \mathbf{w}_i \rangle < 0 & \text{if } y_i = -1, \end{cases}$
**return** $\mathbf{u}^\star$

### 3.2 Secret Key Recovery – Rescaling

Algorithm 1 gives us a vector $\mathbf{u}^\star = c\mathbf{b}$ for some unknown $c \in \mathbb{R}$. By Definition 1 we know the distribution $D$ from which $\mathbf{b}$ is sampled coordinate-wise. Let $B \sim D$ be a random variable for a secret key coordinate from a distribution $D$ with mean zero. Then $cB$ has

$$\mathbb{E}[cB] = c \cdot \mathbb{E}[B] = 0 \text{ and } \mathrm{Var}(cB) = c^2 \cdot \mathrm{Var}(B). \tag{5}$$

*Estimating variance.* Equation (5) motivates that we estimate the variance from our observed $\mathbf{u}^\star = (u_1, \ldots, u_n)$ in order to estimate the unknown scaling factor $c$. Let $cD$ denote the scaled distribution. We assume that $\mathbf{u}^\star \sim (cD)^n$. Let $U_i \sim cD$, $i = 1, \ldots, n$ be iid samples for the $i^{th}$ entry of a random variable $U \sim cD$ with $\mathbb{E}[U] = 0$. It follows that

$$\widehat{\mathrm{Var}}(U_1, \ldots, U_n) = \frac{1}{n} \sum_{i=1}^{n} U_i^2$$

is a *consistent estimator* for $\mathrm{Var}(U)$ (see e.g. [HB23, Chapter 15.4]), i.e.,

$$\lim_{n\to\infty} \Pr[|\widehat{\mathrm{Var}}(U_1, \ldots, U_n) - \mathrm{Var}(U)| \geq \epsilon] = 0 \text{ for any } \epsilon > 0.$$

**Lemma 5 (Rescaling).** *Let* $\mathbf{b}$ *be a secret key sampled coordinate-wise with random* $B \sim D$ *from some distribution* $D$ *with mean* $\mathbb{E}[B] = 0$ *and variance*

*Var*$(B) = \sigma^2$. *Let* $\mathbf{u}^\star = c\mathbf{b} \in \mathbb{R}^n$ *for some unknown* $c \in \mathbb{R}$. *Then*

$$\hat{c} = \frac{\|\mathbf{u}^\star\|}{\sqrt{n} \cdot \sigma}$$

*is an asymptotically optimal estimator of* $c$.

*Proof.* Evaluating our consistent variance estimator at the point $\mathbf{u}^\star = (u_1, \ldots, u_n)$ yields

$$\widehat{\mathrm{Var}}(U_1 = u_1, \ldots, U_n = u_n) = \frac{\|\mathbf{u}^\star\|^2}{n}.$$

Since the estimator is consistent, it converges to $\mathrm{Var}(U) = c^2\mathrm{Var}(B) = c^2\sigma^2$. Solving for $c$ yields our (asymptotic) formula for $\hat{c}$. □

*Remark 6 (Alternative scaling factor).* In cryptographic settings, the coordinates of the secret key $\mathbf{b}$ are usually sampled from a discrete distribution $D$. In this case, the rescaled $c^{-1}\mathbf{u}^\star$ is expected to be close to an integer vector. This motivates the following heuristic estimator for the scaling factor:

$$\hat{c} \coloneqq \arg\min_{c \in C} \left\| c^{-1}\mathbf{u}^\star - \left\lfloor c^{-1}\mathbf{u}^\star \right\rceil \right\|, \tag{6}$$

where the set $C$ of candidate scaling factors simply enumerates coordinate-wise from $D$ as $C = \{u_i^\star/d \mid d \in D \setminus \{0\},\ i = 1, \ldots, n\}$.

## 4 Key Recovery from Sign Leakage in HAWK

We outline the key generation and signing process of HAWK; full details are available in the specification [BBD+23].

*Key generation.* Let $n \in \{512, 1024\}$. The secret key consists of four polynomials $f, g, F, G$ with small coefficients, satisfying the NTRU equation

$$fG - gF \equiv 1 \mod X^n + 1. \tag{7}$$

The coefficients $f_i, g_i$ of the polynomials $f = \sum_{i=0}^{n-1} f_i X^i$ and $g = \sum_{i=0}^{n-1} g_i X^i$ are sampled from a binomial distribution $\mathrm{Bin}(\frac{n}{64}, \frac{1}{2})$, centered around 0. The polynomials $F$, $G$ are derived via Eq. (7). The secret basis is a $2n \times 2n$ block-wise negacyclic matrix:

$$\mathbf{B} \coloneqq \begin{bmatrix} C(f)\ C(F) \\ C(g)\ C(G) \end{bmatrix} \in \mathbb{Z}^{2n \times 2n}, \text{ where } C(a) = \begin{bmatrix} a_0 & -a_{n-1} & \ldots & -a_1 \\ a_1 & a_0 & \ldots & -a_2 \\ \vdots & & \ddots & \vdots \\ a_{n-1} & a_{n-2} & \ldots & a_0 \end{bmatrix} \in \mathbb{Z}^{n \times n}. \tag{8}$$

The public key $\mathbf{Q}$ is the corresponding Gram matrix $\mathbf{Q} \coloneqq \mathbf{B}^\star\mathbf{B}$ of the secret basis $\mathbf{B}$, see also Section 2.

*Signing.* In the signing algorithm we omit compression for clarity. Moreover, we mark the operations that we attack via our end-to-end key recovery in Section 7.

- Hash the message to a vector $\mathbf{h} \in \mathbb{Z}^{2n}$.
- Compute a binary target $\mathbf{t} := \mathbf{B}\mathbf{h} \bmod 2$ using the private key $\mathbf{B}$.
- [**Sign Leak**] Sample a vector $\mathbf{x}$ from a distribution approximating a discrete Gaussian over $2\mathbb{Z}_n^2 + \mathbf{t}$ with standard deviation $2\sigma_{\text{sign}}$, ensuring that $\|\mathbf{x}\|$ remains small.
- Compute $\mathbf{w} := \mathbf{B}^{-1}\mathbf{x}$.
- [**Symbreak Leak**] Set $\mathbf{w} := -\mathbf{w}$ if the *symmetry-break* condition is not fulfilled. The *symmetry-break* (or *symbreak*) condition is true only if $\mathbf{w}$ is non-zero and its first non-zero coefficient is positive.
- Compute $\mathbf{s} := \frac{1}{2}(\mathbf{h} - \mathbf{w})$ and output $(\mathbf{s}, \mathbf{h})$.

*Verification.* To verify a signature, one derives $\mathbf{w} = \mathbf{h} - 2\mathbf{s}$ from $\mathbf{s}$ and $\mathbf{h}$, and checks that the *symbreak* condition is satisfied, and that $\|\mathbf{w}\|_{\mathbf{Q}} = \|\mathbf{x}\|$ is small.

*Weak forgery.* The *symbreak* condition enforces that the first non-zero coefficient of $\mathbf{w}$ is positive. This prevents the following weak forgery attack. Assume $(\mathbf{h}, \mathbf{s})$ is a valid signature. Then $(\mathbf{s}^\dagger, \mathbf{h})$ with $\mathbf{s}^\dagger := -\mathbf{s} + 2\mathbf{h}$ would also be a valid signature for $\mathbf{h}$ if the *symbreak* condition was not enforced, since

$$\mathbf{w}^\dagger = \mathbf{h} - 2\mathbf{s}^\dagger = -(\mathbf{h} - 2\mathbf{s}) = -\mathbf{w},$$

and $\|-\mathbf{w}\|_{\mathbf{Q}} = \|-\mathbf{x}\| = \|\mathbf{x}\| = \|\mathbf{w}\|_{\mathbf{Q}}$ has small norm.

### 4.1 Sign and Symbreak Leakage

The sampler outputs a randomness vector $\mathbf{x}$ drawn from a discrete Gaussian distribution. In our model, the attacker learns the sign of at least one of its coordinates $x_i$. We now argue that this is a realistic scenario, as *a single higher-order bit or the Hamming weight* of $x_i$ already yield its sign.

Each HAWK randomness vector $\mathbf{x}$ consists of $2n$ samples, independently drawn from $x \sim \mathcal{D}_{2\sigma_{\text{sign}}}$, where $\sigma_{\text{sign}} \in \{1.278, 1.299\}$ depending on the parameter set. Due to the small standard deviation of the sampler, each coordinate $x_i$ has a small magnitude. Specifically, [GR24] shows that in practice $|x| \leq 9$.

Five bits are required to encode such $x_i$: one for the sign and four for the magnitude. The HAWK reference implementation encodes each coordinate in an 8-bit word using two's complement representation. More precisely, let $x_i = (x_{i,7}, \ldots, x_{i,0})$ denote the 8-bit two's complement representation of $x_i$, with $x_{i,7}$ as the most significant bit. For any $j \geq 4$, the bit $x_{i,j}$ coincides with the sign bit: 0-bit for positive sign $+1$, 1-bit for negative sign $-1$.

The sign of $x_i$ can also be inferred from its *Hamming weight*. For $0 \leq x_i \leq 9$, at most 3 bits are set to 1, whereas for negative values $-9 \leq x_i < 0$, at least 5 bits are set to 1. This asymmetry induces a Hamming weight gap between positive and negative values, from which $\text{sign}(x_i)$ can be inferred.

*Importance of symbreak leak.* Notice that, for randomness $\mathbf{x} = (x_1, \ldots, x_{2n})$, we leak all $\text{sign}(x_i)$ *before* the symbreak operation. If a symbreak occurs, the signing algorithm sets $\mathbf{w} := (-\mathbf{w})$. It follows that $-\mathbf{w} = -\mathbf{B}^{-1}\mathbf{x} = \mathbf{B}^{-1}(-\mathbf{x})$.

Thus, explicitly negating $\mathbf{w} := (-\mathbf{w})$ implicitly negates the randomness $\mathbf{x} := (-\mathbf{x})$, too. As a consequence, if a symbreak occurs we have to either negate $\mathbf{w}$ back, or to flip all $sign(x_i)$, $i = 1, \ldots, n$ of our sign leakage.

Since a symbreak occurs with probability $\frac{1}{2}$, without a symbreak detection our sign leak does not provide us any sign information about the (implicitly) used randomness which can be either $\mathbf{x}$ or $-\mathbf{x}$. While symbreak was originally designed to prevent the aforementioned weak forgery attack, it may therefore also provide some protection against sign leakage attacks and, more generally, randomness leakage attacks on HAWK. However, our attack in Section 7 demonstrates that the occurance of a symbreak can reliably be detected.

### 4.2 LH-sample Extraction from Sign Leakage

A HAWK signature satisfies

$$\mathbf{s} = \frac{1}{2}(\mathbf{h} - \mathbf{w}),$$

where $\mathbf{h}$ is the hash of the message and $\mathbf{w}$ is a vector satisfying

$$\mathbf{B}\mathbf{w} = \mathbf{x}.$$

Here, $\mathbf{B}$ is the secret key matrix, and $\mathbf{x}$ is a Gaussian vector, i.e., the randomness in the signing process. Thus, each coordinate of $\mathbf{x}$ can be written as

$$x_i = \langle \mathbf{b}_i, \mathbf{w} \rangle \text{ for } 1 \leq i \leq 2n, \tag{9}$$

where $\mathbf{b}_i$ is the $i^{th}$ row of $\mathbf{B}$. It follows that leaking $\text{sign}(x_i) = \text{sign}(\langle \mathbf{b}_i, \mathbf{w} \rangle)$ yields an LH-sample $(\mathbf{w}, \text{sign}(\langle \mathbf{b}_i, \mathbf{w} \rangle))$ that can be used to recover the secret $\mathbf{b}_i$ by solving an instance of Learning a Halfspace. However, also notice that all $\mathbf{b}_i$ are different.

**Extracting $n$ LH-samples per signature.** Using $i = 1, \ldots, n$ in Eq. (9) provides linear equations in the secret coefficients of $(f, F)$, using $i = n+1, \ldots, 2n$ provides equations for $(g, G)$. We only use the first $n$ equations to recover $(f, F)$. We then compute $(g, G)$ via Eq. (7), see also Section 4.3.

In order to extract $n$ LH-samples for the *same secret* $\mathbf{b}$ we have to revert the negacyclic rotations for $i > 1$. Consider the upper half of the secret key in matrix form:

$$\mathbf{B}_{\text{upper}} = \big[C(f)\ C(F)\big] = \begin{bmatrix} f_0 & -f_{n-1} & \ldots & -f_1 & F_0 & -F_{n-1} & \ldots & -F_1 \\ f_1 & f_0 & \ldots & -f_2 & F_1 & F_0 & \ldots & -F_2 \\ \vdots & & \ddots & \vdots & \vdots & & \ddots & \vdots \\ f_{n-1} & f_{n-2} & \ldots & f_0 & F_{n-1} & F_{n-2} & \ldots & F_0 \end{bmatrix}.$$

Let $\mathbf{b}$ denote the first row of $\mathbf{B}_{\text{upper}}$, and let $\mathbf{w} = (w_1, \ldots, w_{2n})$. Notice that Eq. (9) yields the following $n$ LH-samples with the same secret $\mathbf{b}$

$$x_i = \left\langle \mathbf{b}, \mathsf{rot}^{(i-1)}(\mathbf{w}) \right\rangle \text{ for } i = 1, \ldots, n, \tag{10}$$

where $\mathsf{rot}^{(i)}$ is $i$ times the rotation

$$\mathsf{rot}(\mathbf{w}) \coloneqq (w_2, w_3, \ldots, w_n, -w_1, w_{n+2}, w_{n+3}, \ldots, w_{2n}, -w_{n+1}). \tag{11}$$

### 4.3 Secret Key Recovery

By applying Algorithm 1 on the LH-samples derived via Eq. (10), we recover a positive multiple $\mathbf{u}^\star = c\mathbf{b} := (\tilde{f}, \tilde{F})$, $c \in \mathbb{R}$ of the secret $\mathbf{b} = (f, F)$.

**Deriving the scaling factor $c$.** For the coefficients of $f$ we have $f_i \sim \mathrm{Bin}(\frac{n}{64}, \frac{1}{2})$ with $\mathrm{Var}(f_i) = \frac{n}{256}$. Using Lemma 5, we have to rescale $\mathbf{u}^\star$ by $\frac{n}{16 \cdot \|\tilde{f}\|}$. We then round the resulting vector $\hat{c}^{-1}\mathbf{u}^\star = (\hat{f}, \hat{F})$ coordinate-wise. Whenever our number $m$ of LH-samples was sufficiently large, rounding recovers the secret $(f, F)$.

**Completing the secret key.** It remains to show that one can recover $(g, G)$ in Eq. (7) from $(f, F)$. Notice that NTRU key generation already provides an algorithm $\mathsf{NTRUSolve}$ [HHGP+03, PP19] that on input $(f, g)$ computes $(F, G)$ satisying Eq. (7). We simply run $\mathsf{NTRUSolve}$ on input $(f, F)$ to recover $(g, G)$.

We observe that $(g, G)$ is not uniquely determined, since $(g', G')$ with

$$g' = g - \alpha f X^k, \;\; G' = G - \alpha F X^k \text{ for some } \alpha \in \mathbb{Z}, k \in \{0, 1, \ldots, n-1\} \tag{12}$$

also satisfies $fG' - g'F = 1 \pmod{X^n + 1}$. To recover the correct pair, we iteratively length reduce some $(g', G')$ output from $\mathsf{NTRUSolve}$ using the identity from Eq. (12), see Algorithm 2.

**Putting things together.** Algorithm 3 describes our complete $\mathsf{HAWK}$ secret key recovery from sign and symbreak leakage. In Section 7 we apply Algorithm 3 to perform an end-to-end attack on real world leak data, whereas in the following we test Algorithm 3 on experimental data with varying LH noise rate $\rho \in [0, 0.5)$.

### 4.4 Experimental Results

*Setup.* We implemented the attack in Python 3.8 and ran experiments on a server equipped with an AMD EPYC 7742 CPU and 1 TB of RAM. For logistic regression, we used scikit-learn's `LogisticRegression` (v1.3) [PVG+11] with the Newton–Cholesky solver and no regularization. Linear programs were solved with Gurobi (v11.0) [Gur24] under an academic license. Table 2 shows the results, which are averaged over multiple experiments. In each experiment, we fixed a secret key and gradually increased the number of processed signatures. We applied the attack until the key was successfully recovered. The reported runtime denotes the real execution time of the attack once sufficient signatures were available.

**Algorithm 2:** ReduceKey$(f, F, g', G')$

**Input :** Secret key candidates $f, F, g', G'$
**Output:** Secret key $(f, F, g, G)$

**repeat**
  **for** $k \leftarrow 0$ **to** $n-1$ **do**
    $\mathbf{u} \leftarrow (g', G')$; $\mathbf{v} \leftarrow (fX^k, FX^k)$; $\alpha \leftarrow \left\lfloor \frac{\langle \mathbf{u}, \mathbf{v} \rangle}{\langle \mathbf{v}, \mathbf{v} \rangle} \right\rceil$
    **if** $\alpha \neq 0$ **then**
      $G' \leftarrow G' - \alpha F X^k$
      $g' \leftarrow g' - \alpha f X^k$
**until** *no update occurs*
$(g, G) \leftarrow (g', G')$
**return** $(f, F, g, G)$

**Algorithm 3:** HAWK Key Recovery

**Input :** List $L$ of signatures and leaks
$(\mathbf{s}, \mathbf{h}, \mathrm{sign}(x_1), \ldots, \mathrm{sign}(x_n), \zeta) \in (\mathbb{Z}^{2n})^2 \times \{-1, +1\}^n \times \{0, 1\}$,
LH *solver* $\in \{LR, LP\}$.
**Output:** HAWK secret key $(f, F, g, G)$.

Initialize empty list $\widetilde{L}$.
**for** $(\mathbf{s}, \mathbf{h}, \mathit{sign}(x_1), \ldots, \mathit{sign}(x_n), \zeta) \in L$ **do**
  $\mathbf{w} \coloneqq \mathbf{h} - 2\mathbf{s}$
  **if** $\zeta = 0$ **then** // Symbreak occured.
    $\mathbf{w} \coloneqq -\mathbf{w}$
  **for** $i = 1, \ldots, n$ **do** // Extract $n$ LH-samples per signature.
    Append $(\mathsf{rot}^{(i-1)}(\mathbf{w}), \mathrm{sign}(x_i))$ to $\widetilde{L}$. // See Eq. (12).
$(\widetilde{f}, \widetilde{F}) \leftarrow \mathsf{SolveLH}(\widetilde{L}, \mathit{solver})$ // Recover direction with Algorithm 1.
$(\hat{f}, \hat{F}) \leftarrow$ rescaling of $(\widetilde{f}, \widetilde{F})$ // Rescale via Lemma 5 or Eq. (6).
$(f, F) \leftarrow$ coordinate-wise rounding of $(\hat{f}, \hat{F}) \in \mathbb{R}^{2n}$
$(g', G') \leftarrow \mathsf{NTRUSolve}(f, F)$
$(g, G) \leftarrow \mathsf{ReduceKey}(f, F, g', G')$ // See Algorithm 2.
**return** $(f, F, g, G)$

*Results.* Depending on the number of processed signatures, the attack completes in between 1 and 200 minutes. Linear programming (LP), applicable only to the noise-free case $\rho = 0$, requires significantly fewer LH-samples (and thus signatures) than Logistic Regression (LR) for both parameter sets. Specifically, LP needs 30 signatures for HAWK-512 (compared to 350 for LR), and again 30 signatures for HAWK-1024 (compared to 2,400).

Regarding the noisy setting, we notice that $\rho = 5\%$ requires roughly 10 times as many signatures as $\rho = 0$. Any further increase of $\rho$ by 5% only roughly doubles the required signatures. For HAWK-512, we were able to recover the key up to noise levels of 35%, and for HAWK-1024 up to noise levels of 10%.

Table 2: Number of signatures required for key recovery in HAWK as a function of the noise level $\rho$.

| | Noise $\rho$ | Time | LH-Samples | Signatures | Algorithm |
|---|---|---|---|---|---|
| HAWK-512 | 0 | 10m | 15k | 30 | LP |
| | 0 | 1m | 180k | 350 | LR |
| | 5 % | 3m | 2.7M | 5.2k | LR |
| | 10 % | 7m | 5.8M | 12k | LR |
| | 15 % | 8m | 10M | 20k | LR |
| | 20 % | 16m | 15M | 29k | LR |
| | 25 % | 17m | 25M | 49k | LR |
| | 30 % | 70m | 40M | 78k | LR |
| | 35 % | 120m | 55M | 110k | LR |
| HAWK-1024 | 0 | 30m | 31k | 30 | LP |
| | 0 | 5m | 2.5M | 2.4k | LR |
| | 5 % | 90m | 20M | 20k | LR |
| | 10 % | 200m | 35M | 34k | LR |

## 5 Key Recovery from Sign Leakage in Falcon

In this section, we apply our Learning Halfspaces framework to the signature standardization candidate Falcon. We outline the scheme in the following and refer to the specification [FHK+18] for more details.

*Key generation.* Let $n \in \{512, 1024\}$, $q = 12289$ and $R = \mathbb{Z}[X]/(X^n + 1)$. The secret key consists of four polynomials $f, g, F, G \in R$ with short coefficients, satisfying the NTRU equation $fG - gF \equiv q \mod X^n + 1$. The polynomials $f$ and $g$ are sampled from a discrete Gaussian distribution with standard deviation $\sigma_{f,g} = 1.17\sqrt{q/2n}$, the polynomials $F$, $G$ are derived via the NTRU equation. The secret basis is the following $2n \times 2n$ block-wise negacyclic matrix:

$$\mathbf{B} := \begin{bmatrix} C_{row}(g) & C_{row}(-f) \\ C_{row}(G) & C_{row}(-F) \end{bmatrix} \in \mathbb{Z}_q^{2n \times 2n} \text{ with } C_{row}(a) = \begin{bmatrix} a_0 & a_1 & \dots & a_{n-1} \\ -a_{n-1} & a_0 & \dots & a_{n-2} \\ \vdots & & \ddots & \vdots \\ -a_1 & -a_2 & \dots & a_0 \end{bmatrix}.$$

The public key is the polynomial $h_{\text{pub}} = gf^{-1} \mod (X^n + 1, q)$. The NTRU equation implies

$$\mathbf{B}^{-1} = \frac{1}{q} \begin{bmatrix} C_{row}(-F) & C_{row}(f) \\ C_{row}(-G) & C_{row}(g) \end{bmatrix}. \tag{13}$$

*Signing.* The signing algorithm, simplified for clarity, and with notation adjusted to HAWK for consistency, proceeds as follows:

– Hash the message to a vector $\mathbf{h} \in \mathbb{Z}_q^n$.

– Compute a target $\mathbf{t} := (\mathbf{h}, \mathbf{0}^n) \cdot \mathbf{B}^{-1}$ using the inverse of the private key $\mathbf{B}$.
– **[Sign Leak]** Sample a vector $\mathbf{x}$ from a distribution approximating a discrete Gaussian over $L(\mathbf{B})$.
– Compute $\mathbf{s} := (\mathbf{t} - \mathbf{x}) \cdot \mathbf{B}$. If $\|\mathbf{s}\|$ exceeds some bound, reject and restart with a fresh $\mathbf{x}$.
– Output $(\mathbf{s}, \mathbf{h}) \in \mathbb{Z}^{2n} \times \mathbb{Z}_q^n$.

### 5.1 Sign Leakage

Let $\mathbf{x} = (x_1, \ldots, x_{2n})$ be the randomness. The authors of [LZY+25] already demonstrated that the $\text{sign}(x_i)$ can be inferred from the Hamming weight of their binary representation.

We emphasize that the Fast Fourier Transform (FFT) (which we omitted in the description of the signing procedure) does not affect the sign leakage, as the $x_i$ are sampled as integers and only cast to floating-point for FFT computation.

### 5.2 LH-sample Extraction from Sign Leakage

**Extracting noise-free LH-samples.** We rewrite the signature identity $\mathbf{s} = (\mathbf{t} - \mathbf{x})\,\mathbf{B} = (\mathbf{h},\ \mathbf{0}^n) - \mathbf{x}\,\mathbf{B}$ as

$$\mathbf{x} = \big((\mathbf{h}, \mathbf{0}^n) - \mathbf{s}\big)\,\mathbf{B}^{-1}.$$

Let $\mathbf{b}_i$ be the $i^{th}$ column of $\mathbf{B}^{-1}$. Then we have

$$x_i = \big\langle \mathbf{b}_i, (\mathbf{h}, \mathbf{0}^n) - \mathbf{s} \big\rangle \text{ for } i = 1, \ldots, 2n.$$

This yields noise-free LH-samples

$$(\mathbf{w}, y) = \big((\mathbf{h}, \mathbf{0}^n) - \mathbf{s},\ \text{sign}(x_i)\big) \in \mathbb{Z}^{2n} \times \{-1, 1\} \text{ for } \mathbf{b}_i. \tag{14}$$

Since the entries in $\mathbf{h} \in \mathbb{Z}_q^n$ are significantly larger than in $\mathbf{s}$, our example $\mathbf{w}$ has larger entries in the first $n$ coordinates. We will use this fact in the following to halve our dimension from $2n$ to $n$ at the mild cost of introducing a small error to our LH-sample. Halving the dimension drastically reduces the number of required LH-samples and thus in turn also the number of required signatures, which by far outweighs the drawback from a small error term.

**Extracting noisy, dimension-halved LH-samples.** Write $\mathbf{b}_i = (\mathbf{b}^{(1)}, \mathbf{b}^{(2)}) \in \mathbb{R}^n \times \mathbb{R}^n$ and $\mathbf{s} = (\mathbf{s}^{(1)}, \mathbf{s}^{(2)}) \in \mathbb{Z}^n \times \mathbb{Z}^n$. Then

$$\big\langle \mathbf{b}^{(1)}, \mathbf{h} - \mathbf{s}^{(1)} \big\rangle = x_i + \big\langle \mathbf{b}^{(2)}, \mathbf{s}^{(2)} \big\rangle. \tag{15}$$

This allows us to define a *noisy* LH-sample

$$(\mathbf{w}, y) = (\mathbf{h} - \mathbf{s}^{(1)}, \text{sign}(x_i)) \in \mathbb{Z}^n \times \{-1, 1\} \text{ for } \mathbf{b}^{(1)}, \tag{16}$$

where we treat $e_i := \langle \mathbf{b}^{(2)}, \mathbf{s}^{(2)} \rangle$ as a error term. In the following we estimate that for Falcon parameters the error term $e_i$ flips the sign of $x_i$ with probability $p \approx 1\%$. Therefore, in the remaining section, we always consider the noisy LH-samples from Eq. (16).

**Secret key completion using the public key.** Let $\mathbf{b}_{n+1} = (\mathbf{b}^{(1)}, \mathbf{b}^{(2)})$ be the $n+1$-th column of $\mathbf{B}^{-1}$ in Eq. (13). Notice that we can infer from $\mathbf{b}^{(1)}$ the secret polynomial $f$. From the Falcon public key $h_{\text{pub}}$ we obtain the polynomial $g$ as

$$g = h_{\text{pub}} \cdot f \bmod (X^n + 1, q).$$

The knowledge of $f, g$ directly yields the polynomials $F, G$ by an application of the NTRUSolve algorithm [HHGP+03, PP19].

**Extracting $n$ LH-samples per signature.** Let $\mathbf{w} = (w_1, \ldots, w_n) = \mathbf{h} - \mathbf{s}^{(1)}$ as defined in Eq. (16). Analogous to Section 4.2, we can extract $n$ noisy LH-samples for the same secret $\mathbf{b}^{(1)}$ from each signature as

$$\left\langle \mathbf{b}^{(1)}, \mathsf{rot}^{(i-1)}(\mathbf{w}) \right\rangle = x_{n+i} + e_{n+i} \text{ for } i = 1, \ldots, n, \tag{17}$$

where $\mathsf{rot}^{(i)}$ is $i$ times the rotation $\mathsf{rot}(\mathbf{w}) \coloneqq (w_2, w_3, \ldots, w_n, -w_1)$, and we have error terms $e_{n+i} = \langle \mathbf{b}^{(2)}, \mathsf{rot}^{(i-1)}(\mathbf{s}^{(2)}) \rangle$.

**Estimating the probability $p$ of a sign flip by the error term.** Let us consider Eq. (15). Since the coordinates of $\mathbf{h} \in \mathbb{Z}_q^n$ are much larger in absolute value than the coordinates of $\mathbf{s}^{(1)}, \mathbf{s}^{(2)}$, the noise term $e_i = \langle \mathbf{b}^{(2)}, \mathbf{s}^{(2)} \rangle$ only rarely infers a sign flip in the LH-sample from Eq. (16). To estimate the probability $p$ of a sign flip, we treat the discrete Gaussian as continuous and heuristically assume independence of signature $\mathbf{s}$ and hash $\mathbf{h}$.

**Estimate 1 (Sign flips)** *Let $\mathbf{b} = (\mathbf{b}^{(1)}, \mathbf{b}^{(2)}) \in \mathbb{R}^{2n}$ and $\mathbf{s} = (\mathbf{s}^{(1)}, \mathbf{s}^{(2)}) \in \mathbb{R}^{2n}$ be independently sampled from centered Gaussian distributions with parameters $\sigma_b$ and $\sigma_s$, respectively. Let $\mathbf{h}$ be sampled uniformly from $\mathbb{Z}_q^n$. Then, for sufficiently large $n$,*

$$\begin{aligned} p \coloneqq \Pr[\text{Flip}] &\coloneqq \Pr\left[\text{sign}(\langle \mathbf{b}^{(1)}, \mathbf{h} - \mathbf{s}^{(1)} \rangle) \neq \text{sign}(\langle \mathbf{b}, (\mathbf{h}, \mathbf{0}^n) - \mathbf{s} \rangle)\right] \\ &\approx \frac{1}{\pi} \arctan\left( \frac{\sigma_s}{\sqrt{\frac{q^2-1}{12} + \sigma_s^2}} \right). \end{aligned}$$

*Justification.* Write $X = \langle \mathbf{b}^{(2)}, -\mathbf{s}^{(2)} \rangle$ and $Y = \langle \mathbf{b}^{(1)}, \mathbf{h} - \mathbf{s}^{(1)} \rangle$, so that

$$\Pr[\text{Flip}] = \Pr[\text{sign}(Y) \neq \text{sign}(Y + X)].$$

We consider the distribution of $X$. Each product $b_i^{(2)} s_i^{(2)}$ has mean 0 and, by independence of $b_i^{(2)}$ and $s_i^{(2)}$, variance $\text{Var}(b_i^{(2)} s_i^{(2)}) = \sigma_b^2 \sigma_s^2$. Since the summands are independent,

$$\text{Var}(X) = n \sigma_b^2 \sigma_s^2.$$

By a central limit approximation, for large $n$,

$$X \approx \mathcal{N}(0, \sigma_X^2), \qquad \sigma_X = \sqrt{n}\, \sigma_b \sigma_s.$$

Now we consider the distribution of $Y$. Let $w_i = h_i - s_i^{(1)}$. Since $h_i$ and $s_i^{(1)}$ are independent by assumption,

$$\mathrm{Var}(w_i) = \mathrm{Var}(h_i) + \mathrm{Var}(s_i^{(1)}) = \frac{q^2-1}{12} + \sigma_s^2.$$

Thus

$$\mathrm{Var}(b_i^{(1)} w_i) = \sigma_b^2 \left( \frac{q^2-1}{12} + \sigma_s^2 \right),$$

and summing $n$ independent terms yields

$$\mathrm{Var}(Y) = n\sigma_b^2 \left( \frac{q^2-1}{12} + \sigma_s^2 \right).$$

Again by a central limit approximation,

$$Y \approx \mathcal{N}(0, \sigma_Y^2), \qquad \sigma_Y = \sqrt{n}\,\sigma_b \sqrt{\frac{q^2-1}{12} + \sigma_s^2}.$$

Finally, we compute the probability $p$ of event Flip. Since $X$ and $Y$ depend on disjoint sets of independent random variables, they are independent. Moreover, both are symmetric around 0, hence

$$\Pr[\mathrm{sign}(Y) \neq \mathrm{sign}(Y+X)] = 2\,\Pr[Y > 0,\, X < -Y].$$

Let $U = X/\sigma_X$ and $V = Y/\sigma_Y$. Then $U, V$ are independent standard normal random variables, and

$$\{Y > 0,\, X < -Y\}$$

corresponds to

$$\{V > 0,\, U < -(\sigma_Y/\sigma_X)V\}.$$

This is a sector at the origin with opening angle $\arctan(\sigma_X/\sigma_Y)$. Since the joint distribution of the standard normal variables $(U, V)$ is rotationally invariant, the probability of this sector depends only on its opening angle, and equals this angle divided by $2\pi$, so

$$\Pr[Y > 0,\, X < -Y] = \frac{1}{2\pi} \arctan\left( \frac{\sigma_X}{\sigma_Y} \right).$$

Therefore,

$$\Pr[\mathrm{Flip}] \approx \frac{1}{\pi} \arctan\left( \frac{\sigma_X}{\sigma_Y} \right) = \frac{1}{\pi} \arctan\left( \frac{\sigma_s}{\sqrt{\frac{q^2-1}{12} + \sigma_s^2}} \right). \qquad \square$$

Falcon-512 has $\sigma_s \approx 165.74$ and thus by Estimate 1, a fraction of $p = 1.48\%$ sign flips in the LH-samples. For Falcon-1024, $\sigma_s \approx 168.39$ and hence $p = 1.5\%$ signs flip. When attacking the actual Falcon implementation, we measure slightly lower flip rates of roughly $p \approx 1\%$, see also Table 3.

### 5.3 Secret Key Recovery

Applying Algorithm 1 on the LH-samples from Eq. (16) recovers $c\mathbf{b}^{(1)} = cf = \tilde{f}$. Since all coefficients of $f$ follow a discrete Gaussian distribution with $\sigma_{f,g} = 1.17\sqrt{q/2n}$, we rescale by Lemma 5 with $\hat{c}^{-1} = \frac{\sqrt{n}\cdot\sigma_{f,g}}{\|\tilde{f}\|} = \frac{1.17\sqrt{q/2}}{\|\tilde{f}\|}$.

This gives us an approximation $\hat{f} \in \mathbb{R}^n$ of $f$. For a sufficiently large $m$ of LH-samples coordinate-wise rounding of $\hat{f}$ recovers $f$, from which we derive the entire secret key. The complete key recovery algorithm is provided in Algorithm 4.

**Algorithm 4:** Falcon Key Recovery

**Input :** List $L$ of signatures and sign leaks $(\mathbf{s}, \mathbf{h}, \text{sign}(x_{n+1}), \ldots, \text{sign}(x_{2n})) \in \mathbb{Z}^{2n} \times \mathbb{Z}_q^n \times \{-1, +1\}^n$, modulus $q$, public key $h$, LH *solver* $\in \{LR, LP\}$.

**Output:** Falcon secret key $(f, F, g, G)$.

Initialize empty list $\widetilde{L}$.
**for** $(\mathbf{s}, \mathbf{h}, \text{sign}(x_1), \ldots, \text{sign}(x_n)) \in L$ **do**
  $(\mathbf{s}^{(1)}, \mathbf{s}^{(2)}) \leftarrow \mathbf{s}$
  $\mathbf{w} \coloneqq \mathbf{h} - \mathbf{s}^{(1)}$
  **for** $i = 1, \ldots, n$ **do** // Extract $n$ LH-samples per signature.
    Append $(\mathsf{rot}^{(i-1)}(\mathbf{w}), \text{sign}(x_{n+i}))$ to $\widetilde{L}$. // See Eq. (17).
$\tilde{f} \leftarrow \mathsf{SolveLH}(\widetilde{L}, \mathit{solver})$ // Recover direction with Algorithm 1.
$\hat{f} \leftarrow$ rescaling of $\tilde{f}$ // Rescale via Lemma 5 or Eq. (6).
$f \leftarrow$ coordinate-wise rounding of $\hat{f} \in \mathbb{R}^n$
$g = hf \bmod (X^n + 1)$ // Use public key.
$(F, G) \leftarrow \mathsf{NTRUSolve}(f, g)$ // Complete key.
**return** $(f, F, g, G)$

### 5.4 Experimental Results

We applied Algorithm 4 to the Falcon setting, considering both parameter sets. The experimental setup was the same as in Section 4.4. Whenever the number of LH-samples exceeded a threshold of 15 million, the attempted key recovery was counted as failure.

The number of signatures required for successful key recovery is summarized in Table 3. The attack succeeds only with the logistic regression (LR) approach, since LP requires noise-free LH-samples and hence cannot handle the error term $e_{n+i}$ from Eq. (17). The number of LH-samples required for Falcon-1024 is similar to Falcon-512, since the smaller secret coefficients in Falcon-1024 partly offset the larger $n$ (recall that $\sigma_b = \sigma_{f,g}$ decreases with $n$).

In contrast to the results for HAWK from Table 2, the attack succeeds only for noise level $\rho = 0$ with 100% probability. For higher noise levels, the success rate decreases down to 5%.

Table 3: Falcon secret key recovery for various noise levels $\rho$. The success rate denotes the fraction of successful key recoveries among all attempted recoveries.

| | Noise $\rho$ | Pr[Flip] | Time | LH-Samp. | Sigs. | Alg. | Succ. Rate |
|---|---|---|---|---|---|---|---|
| Falcon-512 | 0 | 1.0 % | 10s | 51k | 100 | LR | 100 % |
| | 5% | 0.9 % | 1m | 1.2M | 2.4k | LR | 40 % |
| | 10% | 0.9 % | 1m | 1.9M | 3.8k | LR | 25 % |
| | 15% | 0.8 % | 2m | 2.6M | 5.1k | LR | 20 % |
| | 20% | 0.7 % | 3m | 3.1M | 6.0k | LR | 5 % |
| | 25% | 0.7 % | 4m | 5.1M | 10k | LR | 5 % |
| Falcon-1024 | 0 | 1.0 % | 30s | 110k | 110 | LR | 100 % |
| | 5% | 0.8 % | 3m | 1.8M | 1.8k | LR | 55 % |
| | 10% | 0.8 % | 3m | 2.6M | 2.5k | LR | 30 % |
| | 15% | 0.7 % | 4m | 3.6M | 3.5k | LR | 15 % |
| | 20% | 0.6 % | 5m | 5.1M | 5.0k | LR | 15 % |
| | 25% | 0.6 % | 7m | 6.1M | 6.0k | LR | 15 % |
| | 30% | 0.5 % | 6m | 6.1M | 6.0k | LR | 5 % |

*The importance of halving the dimension.* We also implemented the attack without halving the dimension, i.e. with the LH-samples from Equation (14). For $\rho = 0$ and Falcon-512, the success rate then decreases from 100% to 30% and the successful cases require 7.7 million samples and 10 minutes on average. This stands in stark contrast to the 51 thousand LH-samples and 10 seconds required for the attack when halving the dimension. Thus, halving not only increases the success rate but also reduces the required number of samples and signatures by roughly a factor of 151. Furthermore, without halving, the attack never succeeds in the noisy setting with $\rho > 0$ in recovering the entire secret key.

## 6 Key Recovery from Sign Leakage in ML-DSA

Let us show how to apply sign leakage attacks to the signature standard ML-DSA. We outline ML-DSA signatures below, with notation adjusted to HAWK for consistency. For full details, we refer to [NIS24].

Table 4: ML-DSA params.

| | 44 | 65 | 87 |
|---|---|---|---|
| $k$ | 4 | 6 | 8 |
| $\ell$ | 4 | 5 | 7 |
| $\eta$ | 2 | 4 | 2 |
| $\gamma$ | $2^{17}$ | $2^{19}$ | $2^{19}$ |
| $\tau$ | 39 | 49 | 60 |

*Key generation.* Let $R = \mathbb{Z}[X]/(X^n + 1)$ with $n = 256$, and let $q = 8380417$. The secret key is a pair of vectors $\mathbf{b} = (\mathbf{b}_1, \ldots, \mathbf{b}_\ell) \in R_q^\ell, \mathbf{e} = (\mathbf{e}_1, \ldots, \mathbf{e}_k) \in R_q^k$ with the coefficients of each $\mathbf{b}_i, \mathbf{e}_i \in R_q$ drawn uniformly from a small integer range $\{-\eta, \ldots, \eta\}$. The public key $(\mathbf{A}, \mathbf{t})$ satisfies $\mathbf{Ab} + \mathbf{e} \equiv \mathbf{t} \pmod q$.

*Signing.* The signing algorithm proceeds as follows:

– Hash the message to a polynomial $h \in R_q$ with $\tau$ non-zero entries uniformly taken from $\{-1, +1\}$.

– **[Sign Leak]** Sample $\mathbf{x} \in \{-\gamma, \ldots, \gamma\}^{\ell n}$ uniformly at random and compute $\mathbf{s} = h\mathbf{b} + \mathbf{x}$. If $\|\mathbf{s}\|_\infty > \gamma - \eta\tau$, reject and restart with a fresh $\mathbf{x}$.
– Output $(\mathbf{s}, h) \in \mathbb{Z}^{\ell n} \times R_q$.

*Parameter sets.* ML-DSA is defined with three parameter sets ML-DSA-44, ML-DSA-65, and ML-DSA-87 (see Table 4), corresponding to security levels of 128, 192, and 256 bits.

### 6.1 Sign Leakage

ML-DSA's randomness $\mathbf{x} = (x_1, \ldots, x_{\ell n})$ has entries $x_i$ sampled uniformly from $\{-\gamma + 1, \ldots, \gamma - 1\}$, where $\gamma$ depends on the parameter set. Since the $x_i$ are integers, they are stored in binary two's complement representation. The authors of [HNP25] demonstrated that the Hamming weight of $x_i$ leaks via side-channels, and observed that $\text{sign}(x_i)$ can be deduced.

### 6.2 Sample Extraction from Sign Leakage

Since $\mathbf{s} = h\mathbf{b} + \mathbf{x} \in R_q$, the coordinates of $\mathbf{s}$ satisfy

$$s_i = \langle \mathbf{b}_{\lceil i/n \rceil}, \mathbf{h}_i \rangle + x_i \text{ for } i = 1, \ldots, \ell n, \tag{18}$$

where $\mathbf{b}_{\lceil i/n \rceil}$ is the $\lceil i/n \rceil$-th subkey of $\mathbf{b}$ and $\mathbf{h}_i$ is the $i^{th}$ row of

$$\begin{pmatrix} C(h) \\ \vdots \\ C(h) \end{pmatrix} \in \mathbb{Z}^{\ell n \times n} \text{ (see Eq.(8) for a definition of } C(h)).$$

**Non-trivial sign leakage.** Notice that the signature coordinates have (even after the rejection step) large absolute values $|s_i| \leq \gamma - \eta\tau$ in comparison to the inner products $|\langle \mathbf{b}_{\lceil i/n \rceil}, -\mathbf{h}_i \rangle| \leq \eta\tau$, see also Table 4.

Thus, whenever $|s_i| > \eta\tau$, the sign of $x_i = s_i + \langle \mathbf{b}_{\lceil i/n \rceil}, -\mathbf{h}_i \rangle$ is fully determined by the sign of the public $s_i$. In this case, an attacker can simply compute $\text{sign}(x_i) = \text{sign}(s_i)$, providing no useful leak information for the randomness $x_i$. This motivates the following definition.

**Definition 7 (Non-trivial sign leakage).** *Let $s_i = \langle \mathbf{b}_{\lceil i/n \rceil}, \mathbf{h}_i \rangle + x_i$ for $i = 1, \ldots, \ell n$ be ML-DSA signature coordinates. Then we call leakage of $\text{sign}(x_i)$ trivial* if $s_i > \eta\tau$, and *non-trivial* otherwise.

We thus filter and use only non-trivial sign leakages, where $|s_i| \leq \eta\tau$, i.e., for very small signature coefficients, that rarely occur. Since $s_i$ is distributed uniformly [Lyu09] in $\{-\gamma + \eta\tau, \ldots, \gamma - \eta\tau\}$, the fraction of non-trivial sign leakages is given by

$$\Pr[|s_i| \leq \eta \cdot \tau] = \frac{2\eta\tau + 1}{2(\gamma - \eta\tau) + 1}, \tag{19}$$

For ML-DSA-44, this yields roughly 0.06% non-trivial sign leakages. For ML-DSA-65 and ML-DSA-87 we obtain 0.04% and 0.02%, respectively.

**Extracting $\ell n$ LH-samples.** We rewrite Equation (18) in the form

$$\langle (1, \mathbf{b}_{\lceil i/n \rceil}), (s_i, -\mathbf{h}_i) \rangle = x_i \text{ for } i = 1, \ldots, \ell n. \tag{20}$$

This produces $n$ LH-sample $(\mathbf{w}, y) = ((s_i, -\mathbf{h}_i), \text{sign}(x_i))$ for each of the $\ell$ (extended) ML-DSA secret subkeys $(1, \mathbf{b}_j) \in \mathbb{Z}^{n+1}$, $j = 1, \ldots, \ell$. Extending $\mathbf{b}_j \in \mathbb{Z}^n$ by an extra dimension helps us to integrate $s_i$ into the examples $\mathbf{w}$, and also simplifies the rescaling step after application of an LH solver.

### 6.3 Secret Key Recovery

Applying Algorithm 1 on the LH-samples from Eq. (20) with non-trivial sign leakage (Definition 7) recovers multiples of all (extended) subkeys $c_j(1, \mathbf{b}_j) = (c_j, \widetilde{\mathbf{b}}_j)$, $j = 1 \ldots, \ell$. Thus, we simply read off the rescale factors $c_j^{-1}$ from the first coordinate.

This gives us an approximation $\hat{\mathbf{b}}_j \in \mathbb{R}^n$ of $\mathbf{b}_j$. Whenever the number of LH-samples is sufficiently large, coordinate-wise rounding of $\hat{\mathbf{b}}_j$ recovers $\mathbf{b}_j$.

This gives the first component $\mathbf{b} = (\mathbf{b}_1, \ldots, \mathbf{b}_\ell)$ of the secret key. The second component $\mathbf{e}$ is computed from the public key as $\mathbf{e} = (\mathbf{t} - \mathbf{A}\mathbf{b}) \bmod q$.

**Algorithm 5:** ML-DSA Key Recovery

**Input :** List $L$ of signatures and leaks $(\mathbf{s}, h, \text{sign}(x_1), \ldots, \text{sign}(x_{\ell n})) \in \mathbb{Z}^{\ell n} \times R_q \times \{-1, +1\}^{\ell n}$, public key $(\mathbf{A}, \mathbf{t})$, $\tau$, $\eta$, LH *solver* $\in \{LR, LP\}$.
**Output:** ML-DSA secret key $(\mathbf{b}, \mathbf{e})$.

Initialize empty lists $\widetilde{L}_j$ for $j \in \{1, \ldots, \ell\}$.
**for** $(\mathbf{s}, h, \text{sign}(x_1), \ldots, \text{sign}(x_{\ell n})) \in L$ **do** // $\ell$ dimension-$n$ subkeys.
  **for** $i \leftarrow 1$ **to** $\ell n$ **do**
    **if** $|s_i| \leq \eta \cdot \tau$ **then** // Non-trivial LH-sample (Definition 7).
      Append $((s_i, -\mathbf{h}_i), \text{sign}(x_i))$ to $\widetilde{L}_{\lceil i/n \rceil}$.
**for** $j \leftarrow 1$ **to** $\ell$ **do** // Recover $\ell$ subkeys.
  $(c_j, \widetilde{\mathbf{b}}_j) \leftarrow \mathsf{SolveLH}(\widetilde{L}_j, \mathit{solver})$ // Recover direction (Algorithm 1).
  $\hat{\mathbf{b}}_j \leftarrow c_j^{-1} \widetilde{\mathbf{b}}_j$ // Rescale.
  $\mathbf{b}_j \leftarrow$ coordinate-wise rounding of $\hat{\mathbf{b}}_j \in \mathbb{R}^n$
$\mathbf{b} = (\mathbf{b}_1, \ldots, \mathbf{b}_\ell)$
$\mathbf{e} = \mathbf{t} - \mathbf{A}\mathbf{b} \bmod q$
**return** $(\mathbf{b}, \mathbf{e})$

### 6.4 Experimental Results

We applied Algorithm 5 to the ML-DSA setting, across all three parameter sets and for noise levels $\rho \in [0, 0.5)$. The experimental setup was the same as in Section 4.4.

The number of signatures required for key recovery is summarized in Table 5. Interestingly, in the noise-free case Logistic Regression (LR) is not only significantly faster than Linear Programming (LP), but also requires less signatures (and thus LH-samples) for key recovery across all three parameter sets.

Notice that ML-DSA-87 (256-bit security) requires less LH-samples and signatures than ML-DSA-65 (192-bit security), since it has by Table 4 a smaller secret key range $\mathbf{b} \in \{-2, \ldots, 2\}^{\ell n}$ rather than $\mathbf{b} \in \{-4, \ldots, 4\}^{\ell n}$, which is beneficial for Algorithm 5.

Table 5: Number of signatures required for key recovery in ML-DSA.

| | Noise $\rho$ | Time | LH-Samples | Signatures | Algorithm |
|---|---|---|---|---|---|
| ML-DSA-44 | 0 | 4m | 42k | 280k | LP |
| | 0 | 5s | 29k | 190k | LR |
| | 5 % | 15s | 620k | 3.8M | LR |
| | 10 % | 18s | 1M | 6.4M | LR |
| | 15 % | 40s | 1.6M | 10M | LR |
| | 20 % | 50s | 2.9M | 18M | LR |
| | 25 % | 1m | 4.6M | 28M | LR |
| | 30 % | 2m | 8.8M | 53M | LR |
| | 35 % | 3m | 15M | 90M | LR |
| ML-DSA-65 | 0 | 10m | 120k | 1.3M | LP |
| | 0 | 4s | 95k | 1M | LR |
| | 5 % | 50s | 2.6M | 26M | LR |
| | 10 % | 1m | 5.1M | 51M | LR |
| | 15 % | 2m | 10M | 102M | LR |
| | 20 % | 5m | 13M | 130M | LR |
| ML-DSA-87 | 0 | 10m | 58k | 1M | LP |
| | 0 | 4s | 29k | 500k | LR |
| | 5 % | 20s | 770k | 13M | LR |
| | 10 % | 30s | 1.5M | 26M | LR |
| | 15 % | 40s | 3.1M | 51M | LR |
| | 20 % | 1m | 4.2M | 70M | LR |
| | 25 % | 1m | 7.7M | 130M | LR |

## 7 End-to-End Attack Against HAWK-512 on Hardware

We now present an automatic end-to-end key recovery attack against HAWK using power traces from the reference implementation.

Guerreau and Rossi [GR24] already show how to determine the sign of the first coordinate of the randomness $\mathbf{x}$ in a power analysis attack. In our setting, however, we face two additional challenges. First, to efficiently use each signature we recover the signs of $n$ coordinates within a single trace. Second, recovering

the secret key requires not only the signs of the coordinates of $\mathbf{x}$, but also the value of symbreak (see Section 4.1).

**Attacker model.** Our attack is split into three phases: First, the profiling phase, where the attacker generates power traces of signing operations with known secret keys and known random seeds. Second, an attack phase, where the attacker obtains power traces for a sufficient number of signatures, generated with the victim's unknown secret key and unknown random seeds, resulting in noisy, measured randomness signs and symbreak flags. Third, a key recovery phase, where the attacker uses Algorithm 3 to recover the secret key.

**Experimental setup.** We record the power traces with a ChipWhisperer Husky and CW308 UFO board hosting an ARM STM32L562 target CPU. We capture power consumption around the invocation of `hawk_sign_finish_alt`. Details on the compiler and test harness for the end-to-end attack can be found in the artifacts.

**Profiling.** Our goal is to identify the randomness signs and the symbreak flag by choosing, for each leaked bit $i$, a single Point-of-Interest (PoI) $k_i^*$ in a power trace $p$, and comparing the power trace value $p[k_i^*]$ at that position with a threshold value $\theta_i^*$. The threshold value is determined by a difference-of-means classifier $\hat{L}_i(p)$ that is able to separate the mean values for the two populations (positive vs. negative, or symbreak vs. no symbreak).

To obtain threshold values, we generate 500 secret keys and capture power traces $p_1, p_2, \ldots, p_{500}$ for one signature for each key. For each trace $p_j$, the true signs of the randomness coordinates $y_{j,1}^*, y_{j,2}^*, \ldots y_{j,n}^*$, and the true symbreak flag $sym_j^*$ are obtained offline, re-running the same signature generation with the same seed, just with extra debug instructions to leak this ground truth data. For simplicity of exposition, we harmonize the two different types of leaks by defining true leak values $L_{j,1}^*, L_{j,2}^*, \ldots, L_{j,n+1}^*$ as follows:

$$L_{j,i}^* := \begin{cases} (y_{j,i}^* + 1)/2 & \text{if } 1 \le i \le n, \\ sym_j^* & \text{if } i = n+1. \end{cases}$$

In other words, $L_j^*$ makes all true leak bits for key $j$ available in a streamlined interface, enumerated from 1 to $n+1$ and mapped to $\{0, 1\}$.

Each power trace $p_j = (p_j[1], p_j[2], \ldots, p_j[S])$ contains $S$ samples. For each leak bit $i$, we define two sets of power traces $P_i^0 := \{p_j | L_{j,i}^* = 0\}$ and $P_i^1 := \{p_j | L_{j,i}^* = 1\}$, corresponding to the two possible true leak bit values. We can now define the means $\mu_i^b[k]$ and sample variances $\sigma_i^b[k]^2$ of the two classes $b \in \{0, 1\}$, as well as the *t-test score* $t_i[k]$ for each leak bit $i$ and sample position $k$,

$$n_i^b := |P_i^b|, \quad \mu_i^b[k] := \frac{1}{n_i^b} \sum_{p \in P_i^b} p[k], \quad \sigma_i^b[k]^2 := \frac{1}{n_i^b - 1} \sum_{p \in P_i^b} \big(p[k] - \mu_i^b[k]\big)^2,$$

$$t_i[k] := \frac{\mu_i^0[k] - \mu_i^1[k]}{\sqrt{\frac{\sigma_i^0[k]^2}{n_i^0} + \frac{\sigma_i^1[k]^2}{n_i^1}}}.$$

We pick the optimal sample position $k_i^*$ and threshold value $\theta_i^*$ for leak bit $i$ by taking the position with the maximum t-score and the midpoint between the means of the two classes. Together, these values define $n+1$ classifier $\hat{L}_i(p)$ by

$$k_i^* := \arg\max_k |t_i[k]|, \quad \theta_i^* := \tfrac{1}{2}\big(\mu_i^0[k_i^*] + \mu_i^1[k_i^*]\big),$$

$$\hat{L}_i(p) := \begin{cases} 0 & \text{if } \mu_i^0[k_i^*] < \mu_i^1[k_i^*] \text{ and } p[k_i^*] < \theta_i^*, \\ 1 & \text{if } \mu_i^0[k_i^*] < \mu_i^1[k_i^*] \text{ and } p[k_i^*] \geq \theta_i^*, \\ 0 & \text{if } \mu_i^0[k_i^*] > \mu_i^1[k_i^*] \text{ and } p[k_i^*] \geq \theta_i^*, \\ 1 & \text{if } \mu_i^0[k_i^*] > \mu_i^1[k_i^*] \text{ and } p[k_i^*] < \theta_i^*. \end{cases}$$

When measuring a power trace $p$ in the attack scenario, we take the samples at positions $k_i^*$ and compare them to the threshold values $\theta_i^*$, mapping the result to the leak bits used in the attack, i.e., $y_i(p) := 2 * \hat{L}_i(p) - 1$ for $1 \leq i \leq n$, and $sym(p) := \hat{L}_{n+1}(p)$.

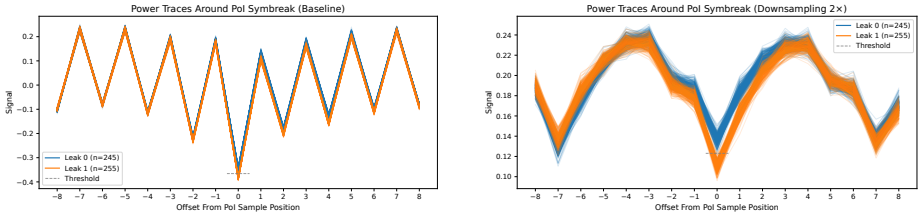
**Baseline and stress test configurations.** To validate our findings at different noise levels, we use downsampling using a low-pass filter in the analog-to-digital converter of the ChipWhisperer hardware.

**Results.** We verified the practicality of our attack by creating three keys and attacking each at a different noise level, based on the capture of 500 random key-signature pairs. As an example, we show in Figure 4 the traces $P_{513}^0$ and $P_{513}^1$ (i.e., those corresponding to the symbreak bit) in an interval around the optimal sample position $k_{513}^*$ for two noise levels. The dashed line shows the threshold value $\theta_{513}^*$. The overall quality of the classifiers can be seen in Figure 5, where we show, for each noise level and PoI $i$, the z-score at the decision boundary

$$z_i = \frac{|\mu_i^0[k_i^*] - \mu_i^1[k_i^*]|}{2\sqrt{\frac{1}{2}(\sigma_i^0[k_i^*]^2 + \sigma_i^1[k_i^*]^2)}}.$$

Our experimental results show good separation ($z_i > 1.5$) for the baseline and the two stress test configurations, given that we allow a maximum error rate of 35% for HAWK-512, corresponding to a minimum z-score of $\approx 0.39$.

We captured the traces of 2000, 4000, and 7000 signatures respectively at the corresponding noise levels.




(a) Baseline (b) Downsampling 2×

Fig. 4: Power trace collections $P^0_{513}$ (blue) and $P^1_{513}$ (orange) for the classification of the symbreak flag at different noise levels. The obtained threshold value $\theta^*_{513}$ is also shown.

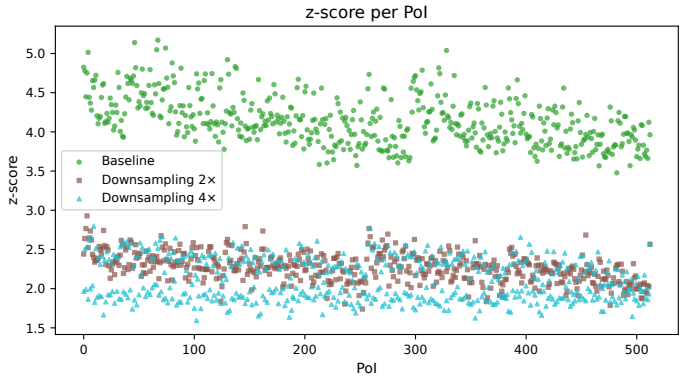



Fig. 5: The z-score for each leak position and classifier in our experimental setup, showing good class separation for each configuration.

Table 6: Experimental results for our end-to-end key recovery attack against HAWK-512 on ARM STM32L5 at different noise levels.

| Downsampling | Samples per Trace | Noise Level $\rho$ | Signatures | Key Recovery |
|---|---|---|---|---|
| Baseline | 11.7M | 0.0035% | 1600 | Successful |
| 2× | 5.8M | 1.6600% | 2900 | Successful |
| 4× | 2.9M | 3.4777% | 5100 | Successful |

Finally, we ran Algorithm 3 on increasingly large subsets of these traces, increasing the number of traces by 100 at each step. The results are shown in Table 6. For all three noise levels, we successfully recovered the secret key, requiring 1600 signatures for $\rho = 0.0035\%$, 2900 signatures for $\rho = 1.66\%$, and 5100 signatures for $\rho = 3.48\%$. In two cases, NTRUSolve recovered the key directly, in the third case it only recovered a key multiple that had to be reduced with ReduceKey.

**Acknowledgments.** Nicolai Kraus is funded by the German Federal Ministry of Education and Research (BMBF) project PQ-CCA. Alexander May is funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2092 CASA - 390781972.

## References

ABSS97. Sanjeev Arora, László Babai, Jacques Stern, and Z Sweedyk. The hardness of approximate optima in lattices, codes, and systems of linear equations. *Journal of Computer and System Sciences*, 54(2):317–331, 1997.

BBD+23. Joppe W. Bos, Olivier Bronchain, Léo Ducas, Serge Fehr, Yu-Hsuan Huang, Thomas Pornin, Eamonn W. Postlethwaite, Thomas Prest, Ludo N. Pulles, and Wessel van Woerden. HAWK. In *National Institute of Standards and Technology (NIST) PQC Project: Round-2 Additional Signatures*, 2023. Available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-2-additional-signatures.

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. Springer, Cham, December 2018.

BFKV96. Avrim Blum, Alan M. Frieze, Ravi Kannan, and Santosh Vempala. A polynomial-time algorithm for learning noisy linear threshold functions. In *37th FOCS*, pages 330–338. IEEE Computer Society Press, October 1996.

BJM06. Peter L Bartlett, Michael I Jordan, and Jon D McAuliffe. Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156, 2006.

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.

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.

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.

DFM+25. Simon Damm, Asja Fischer, Alexander May, Soundes Marzougui, Leander Schwarz, Henning Seidler, Jean-Pierre Seifert, Jonas Thietke, and Vincent Quentin Ulitzsch. Solving concealed ILWE and its application for breaking masked dilithium. Cryptology ePrint Archive, Paper 2025/1629, 2025.

DKM+25. Simon Damm, Nicolai Kraus, Alexander May, Julian Nowakowski, and Jonas Thietke. One bit to rule them all - imperfect randomness harms lattice signatures. In Tibor Jager and Jiaxin Pan, editors, *PKC 2025, Part I*, volume 15674 of *LNCS*, pages 284–316. Springer, Cham, May 2025.

FGKP06. Vitaly Feldman, Parikshit Gopalan, Subhash Khot, and Ashok Kumar Ponnuswami. New results for learning noisy parities and halfspaces. In *47th FOCS*, pages 563–574. IEEE Computer Society Press, October 2006.

FHK+18. Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Prest, Thomas Ricosset, Gregor Seiler, William Whyte, Zhenfei Zhang, et al. Falcon: Fast-fourier lattice-based compact signatures over ntru. *Submission to the NIST's post-quantum cryptography standardization process*, 36(5):1–75, 2018.

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.

GR06. Venkatesan Guruswami and Prasad Raghavendra. Hardness of learning halfspaces with noise. In *47th FOCS*, pages 543–552. IEEE Computer Society Press, October 2006.

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

Gur24. Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2024.

HB23. Mor Harchol-Balter. *Introduction to probability for computing*. Cambridge University Press, 2023.

HHGP+03. Jeffrey Hoffstein, Nick Howgrave-Graham, Jill Pipher, Joseph H Silverman, and William Whyte. Ntrusign: Digital signatures using the ntru lattice. In *Cryptographers' track at the RSA conference*, pages 122–140. Springer, 2003.

HMS+23. Julius Hermelink, Erik Mårtensson, Simona Samardjiska, Peter Pessl, and Gabi Dreo Rodosek. Belief propagation meets lattice reduction: Security estimates for error-tolerant key recovery from decryption errors. *IACR TCHES*, 2023(4):287–317, 2023.

HNP25. Julius Hermelink, Kai-Chun Ning, and Richard Petri. Finding and protecting the weakest link - on side-channel attacks on y in masked ML-DSA. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part V*, volume 16004 of *LNCS*, pages 3–37. Springer, Cham, August 2025.

KA21. Emre Karabulut and Aydin Aysu. Falcon down: Breaking falcon post-quantum signature scheme through side-channel attacks. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pages 691–696. IEEE, 2021.

KA24. Emre Karabulut and Aydin Aysu. Masking FALCON's floating-point multiplication in hardware. *IACR TCHES*, 2024(4):483–508, 2024.

KKMS05. Adam Tauman Kalai, Adam R. Klivans, Yishay Mansour, and Rocco A. Servedio. Agnostically learning halfspaces. In *46th FOCS*, pages 11–20. IEEE Computer Society Press, October 2005.

KLS+25. Markus Krausz, Georg Land, Florian Stolz, Jan Richter-Brockmann, and Tim Güneysu. To extend or not to extend: Agile masking instructions for pqc. In *International Conference on Cryptology and Network Security*, pages 70–92. Springer, 2025.

KPLG24. Elisabeth Krahmer, Peter Pessl, Georg Land, and Tim Güneysu. Correction fault attacks on randomized CRYSTALS-Dilithium. *IACR TCHES*, 2024(3):174–199, 2024.

LTYZ25. Xiuhan Lin, Mehdi Tibouchi, Yang Yu, and Shiduo Zhang. Do not disturb a sleeping falcon - floating-point error sensitivity of the falcon sampler and

its consequences. In Serge Fehr and Pierre-Alain Fouque, editors, *EUROCRYPT 2025, Part II*, volume 15602 of *LNCS*, pages 213–244. Springer, Cham, May 2025.

Lyu09. Vadim Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 598–616. Springer, Berlin, Heidelberg, December 2009.

LZS+20. Yuejun Liu, Yongbin Zhou, Shuo Sun, Tianyu Wang, Rui Zhang, and Jingdian Ming. On the security of lattice-based fiat-shamir signatures in the presence of randomness leakage. *IEEE Transactions on Information Forensics and Security*, 16:1868–1879, 2020.

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. In Tibor Jager and Jiaxin Pan, editors, *PKC 2025, Part I*, volume 15674 of *LNCS*, pages 229–258. Springer, Cham, May 2025.

MOP07. Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power analysis attacks: Revealing the secrets of smart cards*. Springer, 2007.

NIS24. NIST. Module-Lattice-Based Digital Signature Standard. 2024. Available at https://doi.org/10.6028/NIST.FIPS.204.

NR06. Phong Q. Nguyen and Oded Regev. Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 271–288. Springer, Berlin, Heidelberg, May / June 2006.

PP19. Thomas Pornin and Thomas Prest. More efficient algorithms for the NTRU key generation using the field norm. In Dongdai Lin and Kazue Sako, editors, *PKC 2019, Part II*, volume 11443 of *LNCS*, pages 504–533. Springer, Cham, April 2019.

PVG+11. Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

RDVC+04. Lorenzo Rosasco, Ernesto De Vito, Andrea Caponnetto, Michele Piana, and Alessandro Verri. Are loss functions all the same? *Neural computation*, 16(5):1063–1076, 2004.

UMTS24. Vincent Quentin Ulitzsch, Soundes Marzougui, Mehdi Tibouchi, and Jean-Pierre Seifert. Profiling side-channel attacks on Dilithium - A small bit-fiddling leak breaks it all. In Benjamin Smith and Huapeng Wu, editors, *SAC 2022*, volume 13742 of *LNCS*, pages 3–32. Springer, Cham, August 2024.

WNGD24. Ruize Wang, Kalle Ngo, Joel Gärtner, and Elena Dubrova. Unpacking needs protection: A single-trace secret key recovery attack on Dilithium. *CiC*, 1(3):12, 2024.

Zha04. Tong Zhang. Statistical behavior and consistency of classification methods based on convex risk minimization. *The Annals of Statistics*, 32(1):56–85, 2004.

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.