From Perfect to Approximate Hints: Efficient LWE Secret Recovery Leveraging Low Hamming Weight

Minki Hhan*, Ga Hee Hong†, Jiseung Kim‡, Changmin Lee† and JeongHwan Lee†
*KAIST, Daejun, Republic of Korea,
Email: minkihhan@kaist.ac.kr
†Korea University, Seoul, Republic of Korea,
Email: {hongh, changminlee, hwani0814}@korea.ac.kr
‡Jeonbuk National University, Jeonju, Republic of Korea,
Email:jiseungkim@jbnu.ac.kr

***Abstract*—The Learning With Errors (LWE) problem is a cornerstone of lattice-based cryptography and underpins the security of numerous cryptographic schemes. To enhance efficiency, practitioners often employ sparse secrets in LWE, where the secret vector $\mathbf{s}$ has a significantly lower Hamming weight than its dimension n. While this approach improves performance, it raises security concerns, particularly against side-channel attacks that can leak partial information—or "hints"—about the secret key.**

In this paper, we revisit the LWE with side information framework on sparse ternary secrets, focusing on approximate/perfect hints of the form $(\mathbf{v}, l)$ satisfying $l = \langle \mathbf{v}, \mathbf{s} \rangle + e$, where e is a small error term, or $l = \langle \mathbf{v}, \mathbf{s} \rangle$. While previous results needed about $n/2$ perfect or modular hints to break LWE in polynomial time, we show empirically, supported by a conservative lower-bound analysis under the Gaussian Approximation Assumption (GAA), that the task can be accomplished with only $O(h \log_2 h)$ hints, where h denotes the Hamming weight of s.

We demonstrate the effectiveness of our algorithm on practical parameter sets used in Fully Homomorphic Encryption (FHE) schemes. For instance, for a sparse-secret FHE bootstrapping regime with $(n, h) = (2^{15}, 32)$, our method requires only 320 approximate/perfect hints to recover the secret key, compared to the 2^{14} perfect/modular hints required by previous methods. For the OpenFHE library with $(n, h) = (2^{15}, 192)$, we heuristically confirm secret-key recovery via $O(h \log_2 h)$ perfect hints; approximate hints have not yet been validated in this setting. After collecting the necessary hints, our algorithm recovers the secret key in polynomial time in dimension n.

Keywords: Sparse Secret LWE, LWE with hints

1. Introduction

The Learning With Errors (LWE) problem [1] asks to find the secret key $\mathbf{s}$ when given a pair $(\mathbf{A}, \mathbf{b}) \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$ satisfying $\mathbf{b} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e} \bmod q$, where $\mathbf{s} \leftarrow \mathbb{Z}_q^n$ and $\mathbf{e}$ is a small noisy integer vector drawn from a discrete Gaussian distribution.

The LWE problem and its various forms have served as the foundation of lattice-based cryptography, finding widespread applications in the field [2], [3], [4], [5], [6]. While the hardness of the LWE problem has been considered through reductions from the worst-case lattice problems on Euclidean lattices such as Gap Shortest Vector Problem (GapSVP) and Shortest Independent Vectors Problem (SIVP) [1], the concrete parameter regime for ensuring better efficiency is set to be robust to known analysis algorithms. For instance, for more efficiency, cryptographers have considered a lower entropy setting, known as the LWE with sparse binary/ternary secret or sparse binary/ternary secret LWE. In this setting, $\mathbf{s}$ is either binary or ternary, and its Hamming weight–the number of its nonzero entries– is significantly smaller than the secret dimension n. The specific regime is used to several applications such as fully homomorphic encryption schemes [7], [8], [9], [10], [11], [12]. This sparse secret key setup is not considered in the theoretical hardness of LWE, but it is robust against all known analysis algorithms largely classified into primal (lattice) algorithms [13], [14] and dual (lattice) algorithms [15], [16].

As FHE moves closer to real-world deployment, security against not only cryptanalytic ideals but also implementation-level threats becomes critical. Beyond purely theoretical analyses, recent works [17], [18], [19], [20] demonstrate practical side-channel attacks on concrete FHE libraries, recovering sensitive material.

Moreover, Dachman-Soled, Ducas, Gong and Rossi (DDGR) initiated a systematic study to leverage side-channel information in the form of hints such as linear relationships or modular constraints on secret keys. They categorized these hints into four types: perfect hints, modular hints, approximate hints, and short vector hints [21]. They integrate and utilize these hints progressively, showing how side-channel leakage could lead to a concrete reduction in the security of lattice-based systems [21]. While their methodology successfully integrates hints into stan-

Author list in alphabetical order; see https://www.ams.org/profession/leaders/culture/JointResearchandItsPublicationfinal.pdf.
Corresponding author: Changmin Lee

dard lattice reduction attacks, their results emphasize that even minimal amounts of side information can significantly impact security, highlighting the delicate balance between leakage and security loss. In particular, [18] demonstrates that hints can be obtained from side-channel attacks on SEAL FHE library (v3.2) [22][1], and that the SEAL achieves only 4.4-bit security by utilizing hints; without hints, the same parameters are expected to achieve 128-bit security.

Existing attacks typically assume access to a large number of (perfect or approximate) hints, but in practice it is costly and often unrealistic to obtain so many reliable leakages. Recent studies [23], [24] have investigated more efficient methods to combine hints with fewer hints to break lattice schemes. May and Nowakowski [23], in particular, solely focused on two types of hints from the four introduced by DDGR: perfect hints and modular hints. They proposed a *too many hints regime* which determines how many hints are required to solve LWE and break cryptosystems in practice. Their work identifies the number of hints required to effectively compromise various LWE-based schemes. Their results reveal that as few as $n/2$, where n is the secret dimension of the lattice, perfect hints can break the LWE problem in polynomial time using LLL reduction algorithm. In the case of FHE schemes, n is typically set to 2^{15} or 2^{16}, which is considerably larger than that used in other lattice-based cryptosystems that employ at most $n = 2^{10}$. Consequently, a significantly larger number of hints is required to break FHE schemes.

Motivated by this, we develop an attack that depends on the Hamming weight h of the secret, which is very small compared to n. Understanding how to leverage such minimal hints in cryptanalysis not only exposes potential vulnerabilities in existing schemes but also informs the design of more robust cryptographic systems.

1.1. This work

We propose a key-recovery attack on sparse secrets in a low-sample regime, using *two types* of hints of the form $(\mathbf{v}, l)$, where $l = \langle \mathbf{v}, \mathbf{s} \rangle + e$ or $l = \langle \mathbf{v}, \mathbf{s} \rangle$, referred to as *approximate hints* and *perfect hints*, respectively. In other words, we solve the integer-LWE problem [25] with sparse secret key even if $m \ll n$, since collecting m approximate/perfect hints can be viewed as the Integer-LWE problem that asks to find the secret $\mathbf{s}$, given $(\mathbf{A}, \mathbf{b}) \in \mathbb{Z}^{m \times (n+1)}$ such that $\mathbf{b} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$.

Contributions. To break the LWE problem, our algorithm heuristically requires only $O(h \log_2 h)$ approximate/perfect hints, where h is the Hamming weight of the secret vector $\mathbf{s}$. By exploiting the sparse ternary structure, our method significantly reduces the number of required hints compared with the $O(n)$ threshold of prior work, supported by a conservative lower-bound analysis under the GAA.

1. The authors did not explicitly address which types of hints are used or how many hints are required in their attack. Indeed, the authors [23] explicitly noted that the framework in [21] fails to explore too many hints regime.

Furthermore, we implemented our algorithm for various parameters. In our experiments, our approach achieves higher empirical success rates than [26], [27] in several representative fixed low-hint sparse-secret settings (e.g., Table 3 and Figures 6, 7). Even with $\sigma = q/8$, the big standard deviation of e in $\langle \mathbf{v}, \mathbf{s} \rangle + e$, our algorithm still finds $\mathbf{s}$ well. For more details, we refer to Section 4.1.

TABLE 1: Comparison of required number of hints: The column labeled ' [23]' indicates the number of perfect or modular hints needed to successfully attack sparse ternary secret LWE with Hamming weight h. The column labeled 'Ours' shows the number of approximate/perfect hints required to recover the secret key.

Schemes	Parameters			[23]	Ours
	n	$\log_2 q$	h		
[7], [28]	2^{15}	161	32	2^{14}	320
	2^{15}	161	64		768
	2^{16}	161	32	2^{15}	320
	2^{16}	161	64		768
[29]	2^{16}	91	32	2^{15}	320
[30]	2^{16}	795	64	2^{15}	768
[12]	2^{14}	66	32	2^{13}	320
	2^{16}	438	32	2^{15}	320
[31], [32], [33]	2^{15}	699	128	2^{14}	1792
	2^{15}	768	192	2^{14}	2913
[8]	2^{16}	1450	64	2^{15}	768
[34]	2^{14}	40	128	2^{13}	1792
	2^{15}	40	192	2^{14}	2913

Comparison with other algorithms using hints. In several representative fixed low-hint sparse-secret settings (e.g., Table 3 and Figures 6, 7), our attack achieves higher empirical success rates than prior LWE solving algorithms using hints. Compared with [23], [25] that quantify the required number of hints, we solve LWE with fewer hints under matched parameter sets and noise levels. These hints can be organized into a matrix $(\mathbf{A}, \mathbf{b}) \in \mathbb{Z}^{m \times (n+1)}$ such that $\mathbf{b} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$ as defined in the integer LWE problem [25]. Bootle et al. demonstrated that this integer LWE problem could be addressed using the least squares method, provided the variance of $\mathbf{e}$ is not superpolynomially higher than that of $\mathbf{A}$. They also established nearly optimal requirements for the sample size necessary to estimate $\mathbf{s}$, which exceeds n.[2] And, [23] formalizes a *too many hints* regime for LWE in which an adversary collects k linearly independent hints about the secret $\mathbf{s} \in \mathbb{Z}^n$. With k linearly independent hints, the LWE instance effectively reduces from dimension n to $n-k$. In this reduced space, standard lattice reduction (e.g., LLL/BKZ) recovers $\mathbf{s}$ once k exceeds a concrete threshold;

2. They further discussed an adapted algorithm of compressed sensing for cases with a sparse secret but did not specify the required sample size, offering only results for the standard scenario.

in particular, they prove that roughly $n/2$ perfect or modular hints are sufficient to break the LWE problem.

Indeed, for various sparse secret parameter regimes in FHE [7], [12], [28], [29], [34], our algorithm needs significantly smaller hints rather than the previous work [23], [25]. A detailed comparison of the required number of hints is provided in Table 1. For the representative parameter regimes we tested, we experimentally confirm that our algorithm can recover the secret using perfect hints.

Moreover, given the same number of hints, our method achieves higher empirical success rates than [26], [27] in many fixed low-hint sparse-secret settings, as shown in Table 3. Nolte et al. [26] propose a statistical heuristic, *Greedy recovery of cool bits*, which attempts to recover $\mathbf{s}$ by comparing the standard deviation of $\mathbf{b}$ with that of $\mathbf{b}-\mathbf{A}_i$ for each column $\mathbf{A}_i$ of $\mathbf{A}$. If $\mathbf{A}_i$ contributes to $\mathbf{b}$, removing it tends to decrease the variance; otherwise, it tends to increase. They report strong experimental performance but provide no analytic success-probability bound. Cheon et al. [27] instead work in the KR^D setting and consider hints of the form $b_i - \sum_{j=1}^{n} a_{ij}s_j > Kq$ for known (a_{ij}, b_i, K). Their method exploits the observation that coordinates with $s_j = 1$ yield systematically larger contributions on average than those with $s_j = 0$. As in [26], their results are purely empirical. In contrast, we add an explicit conservative lower bound under the Gaussian Approximation Assumption (GAA), which remains heuristic but provides the first analytic threshold guide for this class of attacks.

Side Channel Attack Simulation. As an application of our algorithm, we simulate the side-channel attack on the OpenFHE library [34]. We point out that efficient polynomial multiplication algorithms such as Number Theoretic Transform (NTT)-variants can leak information in the form $\langle \mathbf{v}_0, \mathbf{s} \rangle$ for some $\mathbf{v}_0$ even though the distribution of $\mathbf{v}_0$ is not a uniform distribution. More specifically, in OpenFHE [34], modular reductions and coefficient-wise products performed in the NTT domain introduce hints about the product of secret key and $2n$-th primitive roots of unity. We heuristically confirm that the non-uniform perfect hints can be used to recover the secret $\mathbf{s}$ via $O(h \log_2 h)$ perfect hints; approximate hints have not yet been validated in this setting. For more details, refer to Section 4.2.1.

Related Works. The formal integration of side-channel hints into lattice attacks has since been actively applied to the NIST PQC standards, ML-KEM [35] and ML-DSA [36]. ElGhamrawy et al. [37] study a differential fault attack on ML-DSA in which a single instruction-skip fault induces a deterministic difference between correct and faulty signatures. This difference collapses the original MLWE instance to a lower-dimensional RLWE instance and can be modeled as a small set of perfect hints on the secret. Follow-up fault attacks on ML-DSA similarly interpret relations obtained from faulty signatures or weakened rejection sampling as linear constraints (inequalities) on the secret key that are then solved by lattice reduction [38].

For profiled side-channel attacks, Fouque et al. [39] (in the context of ML-KEM and ML-DSA) use machine-learning classifiers to produce, for each secret coefficient, a probability distribution conditioned on the observed traces. From these distributions they extract moments (mean and variance) that can be modeled as approximate hints. The same work proposes a hybrid strategy that treats very high-confidence ML predictions as perfect hints, while arguing that representing the full soft information only via modular hints is unnecessarily lossy, and thus motivates working directly with real-valued probabilistic hints.

In parallel, Hermelink et al. [40] show that belief propagation (BP) can be combined with lattice reduction to enable error-tolerant key recovery from decryption errors. There, each faulty decryption produces an inequality that behaves like a noisy constraint on the secret, and BP is used to iteratively refine these constraints before the final lattice step—an interpretation that aligns cleanly with the hint-based view. A subsequent work [41] generalizes this idea by introducing a broader notion of *distribution hints* for side-channel attacks on LWE-based schemes. Rather than collapsing SCA-derived probabilities into a single mean–variance pair, they retain a full discrete distribution for each constraint and apply BP (or a greedy surrogate) as a *pre-processing* stage to denoise and sharpen these distributions. Only after this refinement are the distributions summarized into moment pairs and passed to a standard hint-based lattice solver, thereby unifying Bayesian distributional inference with hint processing and providing a common abstraction that captures many earlier ML-KEM attacks.

Beyond the direct use of hints in lattice reductions, recent works have also explored *structurally modifying* the underlying LWE-type problems to better capture scenarios where only partial or highly constrained information on the secret is available. In this direction, Damm et al. [42] propose a variant of ILWE, termed *concealed ILWE*, in which the pair $(\mathbf{v}, \ell)$ is either zero or information-theoretically hides the secret; in other words, most samples carry no usable hint at all, and only rare non-concealed samples behave like classical DDGR hints. Their model is explicitly motivated by recent hardware attacks on Fiat–Shamir-with-aborts lattice signatures [43], [44], [45], where masking and other countermeasures aim to suppress or randomize side-channel information so that only sporadic, highly structured leakages remain. Concealed ILWE can thus be viewed as a complementary abstraction to DDGR: instead of optimizing the exploitation of abundant hints, it formalizes settings in which hints are deliberately *hidden* or sparsified by design.

Scope: sparse-secret deployments. Our results target sparse-secret LWE deployments ($h \ll n$) and do not improve upon [23] for dense-secret PQC parameters. For instance, ML-KEM [35] uses a centered-binomial secret distribution with expected Hamming weight $h \approx \frac{11}{16}n$ (ML-KEM512) or $h \approx \frac{5}{8}n$ (ML-KEM768/1024). In this regime, our method requires about $2h \log h$ hints—exceeding the $n/2$ hints that [23] showed to be sufficient. For dense secrets, [23] is preferable.

1.2. Technical overview

This section introduces a technical idea of our algorithm given a pair $(\mathbf{A}, \mathbf{c})$ such that $\mathbf{A} \in ([-q/2, q/2] \cap \mathbb{Z})^{m \times n}$ and

$$\mathbf{c} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e} \text{ over } \mathbb{R}. \tag{1}$$

Here, we consider $\mathbf{s} \in \mathcal{T}^n(h)$ and $h \ll n$, where $\mathcal{T}^n(h)$ is a set of ternary vectors of dimension n whose Hamming weight is h.

We call this pair $(\mathbf{A}, \mathbf{c})$ an Integer-LWE instance. An algorithm for solving a general Integer-LWE instance was proposed by [25]. The key idea in [25] is to apply the least squares method to $(\mathbf{A}, \mathbf{c})$ in order to recover $\mathbf{s}$. To guarantee the correct recovery of $\mathbf{s}$, this algorithm requires the number of samples m to be at least larger than n. However, we are interested in the case where $m < n$, which necessitates a different strategy to recover $\mathbf{s}$ from $(\mathbf{A}, \mathbf{c}) \in \mathbb{Z}^{m \times n} \times \mathbb{Z}^m$.

Instead, we consider an approach to leverage linear algebra over $\mathbb{R}$ and the statistical moments, which are also defined over $\mathbb{R}$. More precisely, the Equation (1) can be rearranged as

$$\mathbf{c} = \sum_{i \in J} \mathbf{A}_i \cdot s_i + \mathbf{e}, \tag{2}$$

where J is the set of indices such that $s_i \neq 0$ and $\mathbf{A}_i$ is the i-th column vector of $\mathbf{A}$. We now consider a size $X_j := \mathbf{A}_j^T \cdot \mathbf{c}$. If $j \in J$, this value is identical to

$$X_j = \mathbf{A}_j^T \cdot \mathbf{A}_j \cdot s_j + \mathbf{A}_j^T \cdot \left(\sum_{i \in J \setminus \{j\}} \mathbf{A}_i \cdot s_i + \mathbf{e} \right).$$

We note that $\mathbf{A}_j$ is sampled from a uniform distribution on $([-q/2, q/2] \cap \mathbb{Z})^m$. It implies that the size of this value is expected to converge to $\frac{q^2}{12} \cdot m \cdot s_j$ as $m \to \infty$ due to the term $\mathbf{A}_j^T \cdot \mathbf{A}_j$. If not, the value is expected to be close to zero. Hence, by checking the size of X_j, one can deduce the index J and s_j.

Suppose that we find s_{j_0} for some index j_0 by determining the size of X_{j_0}. As naive methods, one can iterate the algorithm for every $j \neq j_0$. However, to amplify the success probability, we employ a modified algorithm of checking X_j for another index $j \neq j_0$. Once s_{j_0} is found, we update $\mathbf{c}$ as $\mathbf{c} - \mathbf{A}_{j_0} \cdot s_{j_0}$ in Equation (2). More precisely, after updating $\mathbf{c}$, it holds that

$$\mathbf{c} = \sum_{i \in J'} \mathbf{A}_i \cdot s_i + \mathbf{e},$$

where $J' = J \setminus \{j_0\}$, which reduces the size of an index set $|J'| = |J| - 1$. Subsequently, for some $j \neq j_0$, s_j can be determined by checking the size of $\mathbf{A}_j^T \cdot \mathbf{c}$. This increases the success probability of the algorithm compared to an identical version that does not update $\mathbf{c}$.

In the detailed analysis in Section 3, under the GAA, this algorithm successfully finds $\mathbf{s}$ when $m = C \cdot h \log_2 h$ for a constant $C \geq 2$ with non-negligible probability. According to experimental results in Section 4, taking $C = 2$ suffices to find $\mathbf{s}$ with high probability.

2. Preliminaries

Notations. We use bold letters to denote vectors or matrices. Let $\mathbb{R}$ be the set of all real numbers and $\mathbb{Z}$ the set of all integers, respectively. For every positive integer N, $[N]$ denotes the set $\{1, 2, \cdots, N\}$. We denote the set of integers in $[-\frac{q-1}{2}, \frac{q}{2}) \cap \mathbb{Z}$ by $\mathbb{Z}_q$.

Given two vectors $\mathbf{w}$ and $\mathbf{u}$, we denote the dot product of two vectors by $\mathbf{w} \cdot \mathbf{u}$ or $\langle \mathbf{w}, \mathbf{u} \rangle$. For any vector $\mathbf{v}$, $\|\mathbf{v}\|$ is denoted by the Euclidean norm of $\mathbf{v}$. That is, $\|\mathbf{v}\| = \sqrt{\sum_i v_i^2}$. The infinity norm of $\mathbf{v}$ is denoted by $\|\mathbf{v}\|_\infty$. Also, 1-norm is denoted by $\|\mathbf{v}\|_1$.

Let $\mathcal{T}(h)$ be a subset of ternary vectors $\{-1, 0, 1\}^*$ such that the Hamming weight of vectors is h. In case that Hamming weight is equal or less than h, we denote it as $\mathcal{T}(\leq h)$.

The notation $x \leftarrow D$ denotes that x is sampled from a distribution D. For any finite set S, $x \leftarrow S$ denotes sampling x from a uniform distribution on S.

Lattice backgrounds. A lattice Λ of rank $d \leq n$ is a discrete subgroup of $\mathbb{R}^n$. If the lattice Λ is generated by d linearly independent vectors $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_d\} \subset \mathbb{R}^n$, then $\mathbf{B}$ is called a basis matrix. When $d = n$, we call Λ the full rank lattice. Given a full rank lattice Λ, the volume of Λ $\det(\Lambda)$ is defined by $|\det(\mathbf{B})|$, where $\mathbf{B}$ is a basis of Λ.

Continuous Gaussian distribution. The continuous Gaussian distribution $\mathcal{G}_{\mu,\sigma}$ over $\mathbb{R}$ with mean μ and standard deviation σ has the probability density function proportional to $\exp(-(x-\mu)^2/(2\sigma^2))$. A standard tail bound gives $\Pr_{X \leftarrow \mathcal{G}_{\mu,\sigma}}[X \geq \mu + a\sigma] \leq \exp(-a^2/2)$. When $\mu = 0$, we denote the continuous Gaussian distribution as $\mathcal{G}_\sigma$.

Gaussian distribution on lattices. Let $\sigma > 0$. The Discrete Gaussian over a lattice Λ with zero mean and standard deviation σ is defined as follows:

$$\mathcal{D}_{\Lambda,\sigma}(\mathbf{x}) = \frac{\rho_\sigma(\mathbf{x})}{\rho_\sigma(\Lambda)},$$

where $\mathbf{x} \in \Lambda$ and $\rho_\sigma(\mathbf{x}) = \exp(-\pi\|\mathbf{x}\|^2/\sigma^2)$ and $\rho_\sigma(\Lambda) = \sum_{\mathbf{x} \in \Lambda} \rho_\sigma(\mathbf{x})$. If there is no confusion of Λ, we omit the subscript Λ, so $\mathcal{D}_{\Lambda,\sigma}$ will be denoted by $\mathcal{D}_\sigma$ for simplicity. Then, it holds that for any d-dimensional lattice Λ and $\sigma > 0$, if $\mathbf{x} \leftarrow \mathcal{D}_{\Lambda,\sigma}$, then $\Pr[\|\mathbf{x}\| < \sigma\sqrt{d}] \geq 1 - 2^{-d}$.

Learning with Errors (LWE) and its variants. Let $\chi_\mathbf{s}$ be a distribution over $\mathbb{Z}_q^n$. For a vector $\mathbf{s} \leftarrow \chi_\mathbf{s}$ and a distribution χ_e over $\mathbb{Z}$, the LWE distribution $\mathcal{A}_{\chi_e,n,q}(\mathbf{s})$ is the probability distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$ defined as follows:

$$(\mathbf{a}_i, b_i = \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i \bmod q)$$

where $\mathbf{a}_i \leftarrow \mathbb{Z}_q^n$ and $e_i \leftarrow \chi_e$ for each i. The search LWE problem is to find $\mathbf{s}$ given $(\mathbf{a}_i, b_i) \leftarrow \mathcal{A}_{\chi_e,n,q}(\mathbf{s})$. The decision LWE problem is to distinguish between the distribution of LWE samples $\{(\mathbf{a}_i, b_i) \leftarrow \mathcal{A}_{\chi_e,n,q}(\mathbf{s})\}$ and the uniformly random distribution $\{(\mathbf{a}_i, r_i) \leftarrow \mathbb{Z}_q^n \times \mathbb{Z}_q\}$. To simplify our explanation, we assume that the noise elements e_i follow a discrete Gaussian distribution over $\mathbb{Z}$ with a

standard deviation of σ, denoted as $\mathcal{D}_\sigma$. Moreover, for a fixed m, we consider a matrix form of LWE problem represented by

$$(\mathbf{A}, \mathbf{b} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e} \bmod q)$$

where $\mathbf{A}$ is a concatenation of vectors $\mathbf{a}_i$, $\mathbf{b} = (b_i)$, and $\mathbf{e} = (e_i) \leftarrow \mathcal{D}_\sigma^m$.

We say that $(n, q, \mathcal{T}(h), \sigma)$-LWE if $\chi_{\mathbf{s}}$ is a uniform distribution over $\mathcal{T}^n(h) := \mathcal{T}(h) \cap \mathbb{Z}^n$ and polynomially many LWE instances sampled from a distribution $\mathcal{A}_{\mathcal{D}_\sigma, n, q}(\mathbf{s})$ are given. We sometimes denote the problem by $(m, n, q, \mathcal{T}^n(h), \sigma)$-LWE to specify the number of instances m. In particular, if $h \ll n$, we call the problem the *sparse secret* LWE problem. Moreover, if there is no modulus q, we call the problem Integer-LWE problem.

Hint framework. We remind the definition of hint introduced by Dachman-Soled, Ducas, Gong and Rossi [21]. Even though they provide four types of hints: Perfect hint, Modular hint, Approximate hint and Short vector hint, this paper mainly considers only perfect hints/approximate hints. This is because they can be regarded as instances of Integer-LWE problem.

Let $\mathbf{s} \in \mathcal{T}^n(h)$ be a secret vector of sparse secret LWE problem. Hint regime assumes that the attackers know the side information $\mathbf{v}, l$ and σ such that

1) Perfect hint: $\langle \mathbf{v}, \mathbf{s} \rangle = l$
2) Approximate hint: $\langle \mathbf{v}, \mathbf{s} \rangle = l + e_\sigma$, where e_σ is a small element.

It is obvious that when attackers obtain m approximate/perfect hints, then it can be regarded as a pair $(\mathbf{A}, \mathbf{c})$ such that

$$\mathbf{c} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$$

over real numbers.

2.1. Algorithms in [26], [27]

This section reviews algorithms in [26], [27] that can exploit approximate hints. The algorithm in [25] requires a much larger number of approximation hints than n. We do not consider this algorithm in this paper because it falls outside the *too many hints regime* of determining the minimum number of hints that breaks the LWE in practice.

In Nolte *et al.* [26], an algorithm was proposed to recover a partial secret vector, called cool bits, after successfully finding complementary of cool bits, called cruel bits. An identity for cool bits, $\mathbf{s} \in \{0, 1\}^n$ is given of the form: an Integer LWE form: $(\mathbf{A}, \mathbf{A} \cdot \mathbf{s} + \mathbf{e})$ with $\mathbf{e}$ of large standard deviation. It seems that in case a modulus reduction may occur for some rows, the algorithm still works, but there is no proper success probability analysis. In this paper, we only consider a case there is no modulus reduction. In the paper, they present an algorithm to find the cool bits $\mathbf{s}$. We define std$(\cdot)$ as a function for computing a standard deviation given an input. The specific algorithm for recovering $\mathbf{s}$ is given in Algorithm 1:

Algorithm 1: Greedy recovery of cool bits in [26]

Input: $(m, n, q, \mathcal{T}^n(h), \sigma_e)$-Mod-free LWE instances $(\mathbf{A}, \mathbf{b})$

```
Set s ∈ Z^n as a zero vector
for i = 1 upto n do
    σ_0 ← std(b)
    σ_1 ← std(b − A_i)
    if σ_0 > σ_1 then
        Update s_i ← 1
    end
end
return s
```

Alternatively, Cheon et al. present an algorithm that recovers the secret $\mathbf{s}$ when an approximate hint LWE is provided in the form $(\mathbf{a}_i, b_i)$, where $b_i - \mathbf{a}_i \cdot \mathbf{s} > K \cdot q$ under KR^D security games exploiting a bootstrapping failure. The constant K depends on the specific bootstrapping parameter setup. For the index $\mathcal{I} = \{j \in [n] \mid s_j \neq 0\}$, the identity can be written as $b_i - \sum_{j \in \mathcal{I}} a_{ij} \cdot s_j$. It indicates that to ensure the evaluation value exceeds $K \cdot q$, a_{ij} should be concentrated around large values, while a_{ij} for $j \notin \mathcal{I}$ should have an average of zero. Let $\mathbb{E}_i(a_{ij})$ represent the average value of a_{ij} over i. For more details, please refer to Lemma 4.2 in the paper [27]. Based on these observations, they provide an algorithm to recover $\mathbf{s}$: As K increases, the success probability of the algorithm improves. However, this comes with a trade-off, as the decryption failure rate decreases. In the experimental section, we set $K = \sqrt{h/12}$, which corresponds to the standard deviation of $\sum_{j \in \mathcal{I}} a_{ij} \cdot s_j$.

Algorithm 2: KR^D attack in [27]

Input: $(m, n, q, \mathcal{T}^n(h), \sigma_e)$-LWE instances $(\mathbf{a}_i, b_i)$ such that $b_i - \mathbf{a}_i \cdot \mathbf{s} > K \cdot q$.

```
Set S = [n], and s ∈ Z^n as a zero vector
while |S| > n − h do
    s_ℓ ← sign(E_i(a_ij)) for ℓ = argmax_{j∈S}(|E_i(a_ij)|)
    S ← S \ {ℓ}
end
return s
```

3. Integer-LWE with Sparse Ternary Keys

This section focuses on heuristic algorithms for solving the variant of the LWE problem that does not have modulus reductions, which we call *Integer-LWE* with sparse ternary secret. Obviously, solving integer-LWE problem can be regarded as solving LWE problem using approximate hints. Throughout this section, we mainly focus on an algorithm using approximate hints, since the perfect hints can be viewed as approximate hints with a zero error $\mathbf{e} = 0$

Suppose that Integer-LWE instance $(\mathbf{A}, \mathbf{c})$ is given as follows:

$$(\mathbf{A}, \mathbf{c} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e}) \in \mathbb{Z}^{m \times n} \times \mathbb{Z}^m$$

where $\mathbf{s} \in \mathcal{T}^n(h)$ and $h \ll n$. We assume that the other settings are almost the same as the ternary LWE problem; each entry of $\mathbf{A}$ is uniformly sampled from $[-q/2, q/2] \cap \mathbb{Z}$, the secret and error vectors $\mathbf{s}, \mathbf{e}$ are sampled from $\mathcal{T}^n(h)$ and $\mathcal{G}_\sigma^m$, respectively. While $\mathbf{e}$ is drawn from a discrete Gaussian distribution, for the purpose of our analysis, we treat it as if it were drawn from a continuous Gaussian distribution; this is a standard approximation for $\sigma \geq 3.19$, widely adopted in practice [22], [28], [34], [46].

Here, the number of samples m could be small since too many hint regime tries to determine the minimal number of hints to break LWE problem. In our algorithm, suppose that the integer LWE problem will be used in the following setting:

- Each entry of $\mathbf{A}$ is uniformly random over $\mathbb{Z}_q$, $h \ll n$, $m \approx O(h \log_2 h)$[3] and $\sigma \approx dq$ for some $0 < d < 1$.

Our approach is to find a most-nonzero-like entry one by one based on the inner products. Algorithm 3 formally describes our method. This algorithm terminates after hmn operations.

Algorithm 3: NONZEROENTRYGUESS

Input: $(\mathbf{A}, \mathbf{c}, h) \in \mathbb{Z}^{m \times n} \times \mathbb{Z}^m \times \mathbb{Z}$ such that $h < n$
Output: A vector $\mathbf{s} \in \mathbb{Z}^n$ of h nonzero entries

Set $K = \emptyset$ and $(s_1, ..., s_n) = (0, ..., 0)$
while $|K| < h$ **do**
  $K \leftarrow K \cup \{j\}$ for $j = \mathsf{argmax}_i \left(|\langle \mathbf{A}_i, \mathbf{c} \rangle|\right)$
  $s_j \leftarrow \mathsf{sign} \langle \mathbf{A}_j, \mathbf{c} \rangle$
  $\mathbf{c} \leftarrow \mathbf{c} - s_j \mathbf{A}_j$
  Remove j-th column of $\mathbf{A}$ // The column indices are not modified
end
return $\mathbf{s} = (s_1, ..., s_n) \in \mathcal{T}^n(h)$

Before proceeding to the formal analysis, we give an intuitive summary of why the algorithm works. For $i \in J$ (a nonzero coordinate of $\mathbf{s}$), the score $X_i = \langle \mathbf{A}_i, \mathbf{c} \rangle$ has a large expected absolute value $|\mu| = mq^2/12$ due to the self-correlation term $s_i \|\mathbf{A}_i\|^2$; for $i \notin J$, the score X_i has mean zero. The algorithm picks $\mathsf{argmax}_i |X_i|$ and peels it off by updating $\mathbf{c}$. The analysis bounds the probability that this ranking is correct across h iterations.

We provide the success probability analysis under the following assumption.

Assumption (GAA: Gaussian Approximation Assumption). Every score variable $X_i := \langle \mathbf{A}_i, \mathbf{c} \rangle$ is modeled as an independent Gaussian with the mean and variance computed exactly from the uniform distribution on $\mathbb{Z}_q$ and the error distribution.

All subsequent bounds on success probability are *conditional on the GAA*; GAA-dependent steps are explicitly identified in the surrounding text. The moments themselves (mean, variance) are exact computations from the uniform distribution on $\mathbb{Z}_q$—the heuristic enters only when we model the score distributions as Gaussian and treat them as independent. While the GAA yields only a conservative lower bound on the success probability, the experiments show that the actual algorithm behaves even better than the analysis given in this section. The gap (e.g., bound ≈ 0.0368 vs. experiment $\approx 6\times$ higher) arises because the GAA ignores favorable higher-order structure in the score distributions.

In Section 3.1.1, we also provide a conservative analytic lower bound on the success probability under the GAA. For example, when $n = 1024, h = 8$ and $\sigma = q/8$, our analysis gives the success probability lower bound of 0.0368 for $m = 48$, which is ≈ 6 times lower than the experimental success probabilities. This gap is expected: the GAA ignores favorable higher-order structure in the score distributions. In comparison, [26] and [27] are purely empirical and provide no analytic bound at all; our Lemma 3.1 gives the first explicit conservative lower bound under the GAA for this class of attacks.

The remainder of this section presents the detailed analysis of the algorithms and the detailed estimations of the success probability.

3.1. Detailed analysis

Statistics of the distributions and assumptions. Recall the input instance $(\mathbf{A}, \mathbf{c}) \in \mathbb{Z}^{m \times n} \times \mathbb{Z}^m$ such that $\mathbf{c} = \mathbf{A}\mathbf{s} + \mathbf{e}$ for some $\mathbf{s} = (s_1, ..., s_n) \in \mathcal{T}^n(h)$. Let us consider $\mathbf{A}$ and $\mathbf{e}$ as random variables such that $\mathbf{A}_i \leftarrow \mathbb{Z}_q^m$ and $\mathbf{e} \leftarrow \mathcal{G}_\sigma$. The set J denotes the indices of nonzero entries of $\mathbf{s}$. We note that the columns of $\mathbf{A} = (\mathbf{A}_1, ..., \mathbf{A}_n)$ and the vectors $\mathbf{e}$, $\mathbf{s}$ (and their entries) are independent. Define the random variables

$$X_i := \langle \mathbf{A}_i, \mathbf{c} \rangle = \left(\sum_{j \in J} s_j \cdot \langle \mathbf{A}_i, \mathbf{A}_j \rangle \right) + \langle \mathbf{A}_i, \mathbf{e} \rangle$$

for $i \in [n]$. When $i \notin J$, the expectation $\mathbb{E}[X_i]$ is 0. For $i \in J$, we have

$$X_i = s_i \cdot \|\mathbf{A}_i\|^2 + \left(\sum_{j \in J \setminus \{i\}} s_j \cdot \langle \mathbf{A}_i, \mathbf{A}_j \rangle \right) + \langle \mathbf{A}_i, \mathbf{e} \rangle .$$

The following calculation is straightforward:

$$\mathbb{E}[X_i] = \begin{cases} 0 & \text{if } i \notin J, \\ s_i \cdot \mathbb{E}\left[\|\mathbf{A}_i\|^2\right] = s_i \cdot \frac{m(q^2-1)}{12} \approx s_i \cdot \frac{mq^2}{12} & \text{if } i \in J. \end{cases}$$

We define $\mu := \frac{mq^2}{12}$, so that $|\mathbb{E}[X_i]| = \mu$ for every $i \in J$ (since $|s_i| = 1$). Ignoring the low-order terms, the variances can be approximated by:

$$\mathbb{V}[X_i] \approx \begin{cases} \frac{mq^2(hq^2+12\sigma^2)}{144} =: \sigma_0^2 & \text{if } i \notin J, \\ \frac{mq^2((h-0.2)q^2+12\sigma^2)}{144} =: \sigma_1^2 & \text{if } i \in J. \end{cases} \quad (3)$$

3. Note a very small m, e.g., $Ch \log(2en/h)/\log(q/\sigma)$ for $C \gg 0$, ensures the unique solution $\mathbf{s}$. The other parameters are chosen so that our m is larger than that.

We aim at distinguishing the distributions of $|X_i|$ and $|X_j|$ for $i \in J$ and $j \notin J$. To this end, we set $m = C \cdot h \cdot \log_2 h$ so that $\mu = |\mathbb{E}[X_i]| > \max\{\sqrt{\mathbb{V}[X_i]}, \sqrt{\mathbb{V}[X_j]}\}$ for $i \in J, j \notin J$.

Computations. Let a, b be uniformly and independently sampled from $\mathbb{Z}_q$, and let e be uniformly sampled from $\mathcal{G}_\sigma$. It holds that $\mathbb{E}[a] = \mathbb{E}[a^3] = 0$, $\mathbb{E}[a^2] = \frac{q^2-1}{12}$, and $\mathbb{E}[a^4] = \frac{(q^2-1)(3q^2-7)}{240}$. Ignoring the low-order terms we have $\mathbb{E}[a^2] \approx \frac{q^2}{12}$ and $\mathbb{E}[a^4] \approx \frac{q^4}{80}$.[4] The following computation verifies Equation (3) for $i \notin J$:

$$\begin{aligned}\mathbb{E}[X_i^2] &= \mathbb{E}\left[\left(\sum_{j\in J}\langle \mathbf{A}_i, \mathbf{A}_j\rangle^2\right) + \langle \mathbf{A}_i, \mathbf{e}\rangle^2\right] \\ &\approx hm \cdot \mathbb{E}[a^2b^2] + m \cdot \mathbb{E}[a^2e^2] \\ &= \frac{hmq^4}{144} + \frac{mq^2\sigma^2}{12} = \frac{mq^2(hq^2+12\sigma^2)}{144}\end{aligned}$$

where the odd-degree terms are removed in the first equation. The cardinality $h = |J|$ and dimension m are used in the approximation, and we use the independence of the variables. Similarly, for $i \in J$, we have:

$$\begin{aligned}&\mathbb{V}[X_i] = \mathbb{E}[X_i^2] - \mathbb{E}[X_i]^2 \\ &= \mathbb{E}\left[\|\mathbf{A}_i\|^4 + \left(\sum_{j\in J\setminus\{i\}}\langle \mathbf{A}_i, \mathbf{A}_j\rangle^2\right) + \langle \mathbf{A}_i, \mathbf{e}\rangle^2\right] - \mu^2 \\ &\approx \frac{mq^4}{80} + \frac{m(m-1)q^4}{144} + \frac{(h-1)mq^4}{144} \\ &\quad + \frac{mq^2\sigma^2}{12} - \frac{m^2q^4}{144} \\ &= \frac{mq^2(hq^2 - 0.2q^2 + 12\sigma^2)}{144}\end{aligned}$$

where we use $\mathbb{E}[\|\mathbf{A}_i\|^4] = m\mathbb{E}[a^4] + \binom{m}{2} \cdot \mathbb{E}[2a^2b^2]$.

3.1.1. Analysis. To analyze the success probability, we invoke the GAA, which means that the random variable X_i is modeled as an independent Gaussian: $X_i \sim \mathcal{G}_{0,\sigma_0}$ if $i \notin J$, and $X_i \sim \mathcal{G}_{s_i\mu,\sigma_1}$ if $i \in J$. Since $s_i \in \{-1, +1\}$, the distribution of $|X_i|$ is the same as that of $|Y|$ for $Y \sim \mathcal{G}_{\mu,\sigma_1}$; hence all tail probabilities for $|X_i|$ below are stated in terms of $\mu > 0$ without loss of generality. Recall the basic Gaussian tail bound $\Pr[X \geq \mu + a\sigma] \leq \exp(-a^2/2)$ for $X \leftarrow \mathcal{G}_{\mu,\sigma}$. The problem is that m is too small, and the usual concentration inequalities fail to provide a fine-grained analysis. To remedy this, we explicitly compute tail probabilities using the erf function. The following lemma summarizes the analysis.

Lemma 3.1. *Algorithm 3 finds the correct secret key* $\mathbf{s}$ *with probability at least*

$$p = \sup_{T_1,T_2,\ldots,T_h>0} p_1(T_1) \cdot \ldots \cdot p_h(T_h)$$

where p_1 *is given in Equation (4) and* p_k *for* $2 \leq k \leq h$ *is given in Equation (5), under GAA.*

The product bound $p_1(T_1) \cdots p_h(T_h)$ additionally assumes independence across successive update steps; we treat this as part of the overall GAA-style heuristic.

Intuition. Let $T_1 > 0$ be a real value to be specified later. The occurrence of the following event $\mathbf{F}(T_1)$ ensures the first iteration of Algorithm 3 finds a correct index:

$$\text{Event } \mathbf{F}(T_1) = \left(\bigvee_{i\in J}[|X_i| > T_1]\right) \wedge \left(\bigwedge_{i\notin J}[|X_i| \leq T_1]\right).$$

Here we consider $T_1 \geq \mu$. If T_1 is slightly larger than μ, the probability that $|X_i| > T_1$ for $i \in J$ is not-so-small, say $1/3$, and we can expect that the former event is true with a high probability. On the other hand, for $i \notin J$, the event $|X_i| > T_1$ is unlikely because $\mathbb{E}[X_i] = 0$.

Below, we heuristically estimate the probability that $\mathbf{F}(T_1)$ occurs and optimize T_1. This will derive the success probability in Lemma 3.1, and also provide a rudimentary estimation of the actual success probability for a set of fixed parameters.

First iteration. Let $T_1 > 0$ be the threshold for the first iteration. The probability that each component of the event $\mathbf{F}(T_1)$ can be computed using the error function as follows: If $i \notin J$, we have

$$\Pr[|X_i| \leq T_1] = \operatorname{erf}\left(\frac{T_1}{\sqrt{2}\sigma_0}\right)$$

and if $i \in J$, by the symmetry noted above,

$$\Pr[|X_i| > T_1] \geq \Pr[Y > T_1] = \frac{1}{2} - \frac{1}{2}\operatorname{erf}\left(\frac{T_1-\mu}{\sqrt{2}\sigma_1}\right)$$

where $Y \sim \mathcal{G}_{\mu,\sigma_1}$ (the omitted left-tail $\Pr[Y < -T_1]$ is negligible for $T_1 \geq \mu$). The probability $\Pr[\mathbf{F}(T_1) = \text{True}] =: p_1(T_1)$ is bounded below by

$$\left(1 - \left(\frac{1}{2} + \frac{1}{2}\operatorname{erf}\left(\frac{T_1-\mu}{\sqrt{2}\sigma_1}\right)\right)^h\right) \cdot \left(\operatorname{erf}\left(\frac{T_1}{\sqrt{2}\sigma_0}\right)\right)^{n-h}. \tag{4}$$

The next iterations. Let us consider the k-th iteration for $2 \leq k \leq h$. Suppose all previous iterations are correct, and define J_k and $X_{i,k}$ as follows: Let $J_k := J \setminus \{i_1, \ldots, i_{k-1}\}$. To deal with the k-th iteration, we need to consider the random variables $X_{i,k}$ which are defined by

$$X_{i,k} := \left(\sum_{j\in J_k} s_j \cdot \langle \mathbf{A}_i, \mathbf{A}_j\rangle\right) + \langle \mathbf{A}_i, \mathbf{e}\rangle \quad \text{for } i \in [n] \setminus \{i_1, \ldots, i_{k-1}\}.$$

The correctness probability of the k-th iteration can be bounded by the probability that the following event occurs for some $T_k > 0$:

$$\text{Event } \mathbf{F}_k(T_k)$$

4. At our smallest modulus $q = \texttt{next_prime}(2^{40})$, the relative errors of both approximations are below 10^{-23}; at $q = \texttt{next_prime}(2^{100})$ they fall below 10^{-59}. These truncations therefore cannot affect coordinate rankings or threshold locations at any modulus used in our experiments.

$$= \left(\bigvee_{i \in J_k} [|X_{i,k}| > T_k] \right) \wedge \left(\bigwedge_{i \notin J} [|X_{i,k}| \leq T_k] \right)$$

By the same GAA modeling as before, $X_{i,k} \sim \mathcal{G}_{0,\sigma_{0,k}}$ for $i \notin J$ and $X_{i,k} \sim \mathcal{G}_{s_i\mu,\sigma_{1,k}}$ for $i \in J_k$, where

$$\sigma_{0,k}^2 := \frac{mq^2\big((h-k+1)q^2 + 12\sigma^2\big)}{144},$$
$$\sigma_{1,k}^2 := \frac{mq^2\big((h-k+0.8)q^2 + 12\sigma^2\big)}{144}.$$

Assuming additionally that the current iteration is independent of the previous, and applying the same symmetry argument, we have the following lower bound of $p_k(T_k)$:

$$\left(1 - \left(\frac{1}{2} + \frac{1}{2}\operatorname{erf}\left(\frac{T_k - \mu}{\sqrt{2}\sigma_{1,k}}\right)\right)^{h-k+1}\right) \cdot \left(\operatorname{erf}\left(\frac{T_k}{\sqrt{2}\sigma_{0,k}}\right)\right)^{n-h} \tag{5}$$

By optimizing the choices of $T_1, ..., T_h$, we obtain Lemma 3.1 under the GAA.

Concrete lower bound estimations. As an example, we consider $n = 1024$, $h = 8$, $m = 2h\log_2 h = 48$ and $\sigma = q/8$[5] to compare with the experiments in Section 4.1. This gives $\mu = 4 \cdot q^2$ and from the formulas in Equations (4) and (5), we have

$$p_1(6.00q) \approx 0.456, \quad p_2(5.67q) \approx 0.499,$$
$$p_3(5.31q) \approx 0.552, \quad p_4(4.93q) \approx 0.617,$$
$$p_5(4.51q) \approx 0.695, \quad p_6(4.04q) \approx 0.788,$$
$$p_7(3.49q) \approx 0.889, \quad p_8(2.78q) \approx 0.973.$$

This gives the success probability about 0.0368 in Lemma 3.1.

Analysis without update. When applying Algorithm 3, we update $\mathbf{A}_j$ and $\mathbf{c}$ by eliminating the previous $\mathbf{A}_j$-term. Alternatively, it is possible to find a candidate for $\mathbf{s}$ without performing the update. Specifically, we can consider the following algorithm:

Algorithm 4: SIMPLENONZEROENTRYGUESS

Input: $(\mathbf{A}, \mathbf{c}, h) \in \mathbb{Z}^{m\times n} \times \mathbb{Z}^m \times \mathbb{Z}$ such that $h < n$
Output: A vector $\mathbf{s} \in \mathbb{Z}^n$ of h nonzero entries

Set $\mathcal{I} = [n]$ and $(s_1, ..., s_n) = (0, ..., 0)$
while $|\mathcal{I}| > n - h$ **do**
  $s_j \leftarrow \mathsf{sign}\,\langle \mathbf{A}_j, \mathbf{c}\rangle$ for $j = \mathsf{argmax}_{i\in\mathcal{I}}\,(|\langle \mathbf{A}_i, \mathbf{c}\rangle|)$
  $\mathcal{I} \leftarrow \mathcal{I} \setminus \{j\}$
end
return $\mathbf{s} = (s_1, ..., s_n) \in \mathcal{T}^n(h)$

However, for this algorithm to correctly recover $\mathbf{s}$, the following condition is sufficient for a certain $T_1 > 0$. Since Algorithm 4 also recovers signs via $s_j \leftarrow \mathrm{sign}\langle \mathbf{A}_j, \mathbf{c}\rangle$, we write the event in terms of $s_i X_i$ (which satisfies $s_i X_i \sim \mathcal{G}_{\mu,\sigma_1}$ under the GAA):

$$\text{Event } \mathbf{H}(T_1) = \left(\bigwedge_{i \in J} [s_i X_i > T_1] \right) \wedge \left(\bigwedge_{i \notin J} [|X_i| \leq T_1] \right).$$

Under the GAA and coordinate independence, the probability of this event is:

$$\left(\frac{1}{2} - \frac{1}{2}\operatorname{erf}\left(\frac{T_1 - \mu}{\sqrt{2}\sigma_1}\right)\right)^h \cdot \left(\operatorname{erf}\left(\frac{T_1}{\sqrt{2}\sigma_0}\right)\right)^{n-h}. \tag{6}$$

Consequently, the success probability decreases exponentially as h increases. For the aforementioned parameter setup, the success probability at $T_1 = \mu$ is approximately 0.00391, which is significantly lower than that of the updated version. We thus consider the Algorithm 3.

4. Experimental Results

This section validates our algorithm and reports results under concrete parameter sets. In Section 4.1, we devise experiments to support the validity of Algorithm 3. In Section 4.2, we present side-channel scenarios that yield approximate/perfect hints in OpenFHE library [34], and compare empirical success rates on Integer-LWE instances, observing that Algorithm 3 performs better than [26], [27] in the low-hint regimes studied here. All experiments were conducted on Linux with Intel(R) Xeon(R) Silver 4208 CPU @ 2.10GHz with 30 GB RAM.

4.1. Numerical Results for Integer-LWE with sparse ternary keys

We implemented Algorithm 1 ([26]), Algorithm 2 ([27]) and Algorithm 3 on various parameters n, h, m, q, or σ using SageMath [47]. Note that all experiments were performed using a discrete Gaussian distribution $\mathcal{D}_\sigma$ even though our analysis employs a continuous Gaussian distribution $\mathcal{G}_\sigma$. We take a threshold value $K = \sqrt{h/12}$ in Algorithm 2. In every figure, the shaded region around each curve indicates the 95% Wilson score confidence interval (CI), providing a rigorous measure of statistical uncertainty given the finite number of trials.

Table 2 summarizes the parameter settings for each figure. In several figures, black/blue/red squares indicate the success probability of Algorithm 1, Algorithm 2 and Algorithm 3, respectively.

In particular, to ascertain the influence of parameters on the algorithm, experiments were conducted in which one variable was increased while the remaining variables were held constant. We mainly fixed $\sigma = 3.19 = 8/\sqrt{2\pi}$ (the default standard deviation in OpenFHE, SEAL, and Lattigo; cf. HES v1.1 §2.1.5) except for experiments that vary σ. Similarly, we usually employ $n = 512, h = 8$ and $m = \lfloor 2h \cdot \log_2 h \rceil$ for quick experiments. We compute the success probability of algorithms by recovering secret vector $\mathbf{s}$. We

5. As long as $\sigma \ll q$, the specific choice does not much matter as shown in both the calculations and the experiment (Figure 3).

TABLE 2: Per-figure parameter summary. A dash indicates the swept variable.

Figure	n	h	m	$\log_2 q$	σ	Trials
1	—	8	48	100	3.19	100
2	—	8	48	100	3.19, $q/8$	100
3	512	8	48	100	—	100
4	512	8	48	—	3.19	100
5	512	8	—	100	3.19	100
6	512	—	48	100	3.19	1100
7	1024	—	$\lfloor 2h \log_2 h \rceil$	100	3.19	100

do not claim a calibrated predictor of the success probability; Lemma 3.1 gives an explicit conservative lower bound under the GAA. The gap between this bound and empirical success rates is expected, as the GAA ignores favorable higher-order structure in the score distributions.

According to several experiments, we observe that Algorithm 1 fails to recover the secret key $\mathbf{s}$ when the number of samples m is small (e.g., $m = 48$). However, as shown in Figure 5, when m is sufficiently large (e.g., $m \geq 9h \log_2 h$), Algorithm 1 also begins to recover the secret, reaching a success rate of ≈ 0.25 at $m = 288$. This is expected, as the algorithm's sample complexity is inherently higher than ours. Experimental results of Algorithm 2 and Algorithm 3 show similar performance for various parameters except for Figure 7, which provides the success probability of algorithms depending on h with $m = \lfloor 2h \log_2 h \rceil$. In Figure 7, we observe that the success probability of our algorithm increases whenever h also increases, but that of Algorithm 2 decreases. This difference is significant for practical parameter regimes where h is moderately large.

According to Table 3, we observe that given approximate/perfect hints $(\mathbf{A}, \mathbf{c}) \in \mathbb{Z}^{m \times n} \times \mathbb{Z}^m$, we can efficiently recover the secret $\mathbf{s}$.

Varying n. We conducted experiments to measure the success probability of algorithms while varying the secret dimension n. Other parameters were held constant: $h = 8$, q is a 100-bit prime number, $\sigma = 3.19$, and $m = \lfloor 2h \log_2 h \rceil$. As illustrated in Figure 1, as n increases, there is a trend of decreasing success probability. This outcome is somewhat expected since, for an index i to be accurately determined, $\mathbf{A}_i \cdot \mathbf{c}$ must exceed $\mathbf{A}_j \cdot \mathbf{c}$ for all j in the set $[n] \backslash \{i\}$. The probability of this condition being met increases as n decreases. Nonetheless, when $n \leq 200$, the success probability exceeds 0.5. The probability remains non-negligible (at least ≈ 0.09) even at $n = 2000$. We also observe that Algorithm 1 fails to recover the secret $\mathbf{s}$ for $m = 48$ samples. This is because Algorithm 1 inherently requires a larger number of samples to function; with sufficient samples (see Figure 5), it can partially recover the secret.

As a side experiment, we directly compare Algorithm 3 under $\sigma = 3.19$ versus $\sigma = q/8$, as shown in Figure 2. We thus speculate that σ does not have a significant impact on the success probability. In fact, according to Figure 3, we

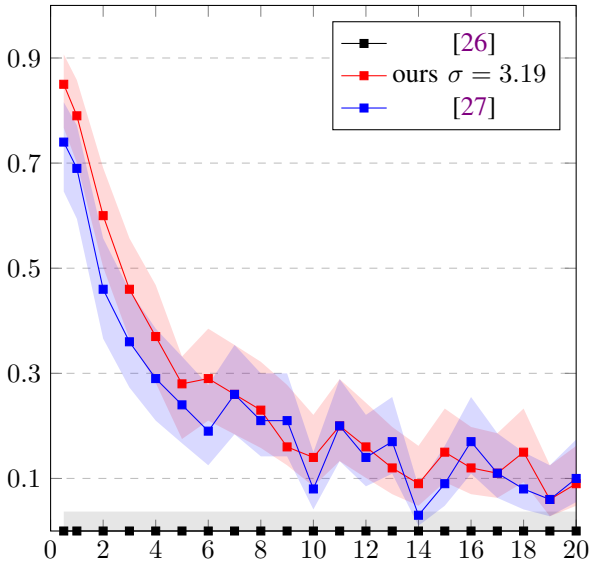


Figure 1: Success probabilities of the algorithms for varying n. The x-axis and y-axis indicate $n/100$ and the success probabilities of Algorithm 1, Algorithm 2 and Algorithm 3. We fixed $h = 8, q = \mathsf{next_prime}(2^{100}), \sigma = 3.19$ and $m = \lfloor 2h \log_2 h \rceil = 48$. For these parameters, we experiment 100 times. Shaded regions indicate 95% Wilson score CIs. Algorithm 1 achieves $\approx 0\%$ success across all n at this sample count.

can verify if $\sigma < q/8$ that the change in success probability with σ is not significant.

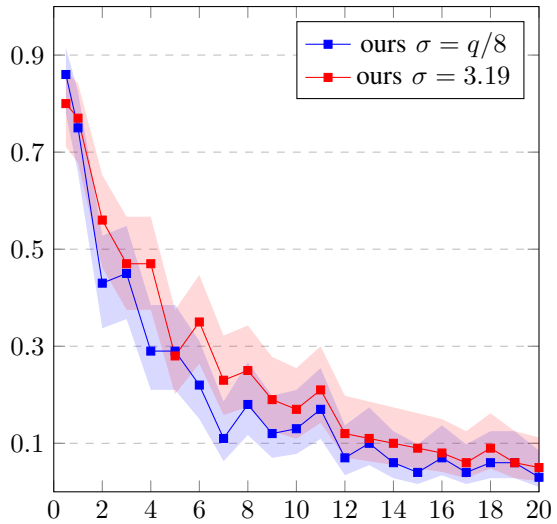


Figure 2: Comparison of Algorithm 3 with two noise levels: $\sigma = 3.19$ vs. $\sigma = q/8$. The x-axis and y-axis indicate $n/100$ and the success probability. We fixed $h = 8, q = \mathsf{next_prime}(2^{100})$ and $m = \lfloor 2h \log_2 h \rceil = 48$. Shaded regions indicate 95% Wilson score CIs.

Varying σ. Figure 3 shows the success probability as a

function of the noise standard deviation $\sigma = q/2^t$. When $\sigma < q/8$ (i.e., $t \geq 3$), the success probability of Algorithm 3 remains largely stable, and its performance is comparable to that of Algorithm 2. For very large σ (small t), performance degrades for all algorithms. In summary, σ is not a dominant factor as long as it stays below $q/8$.

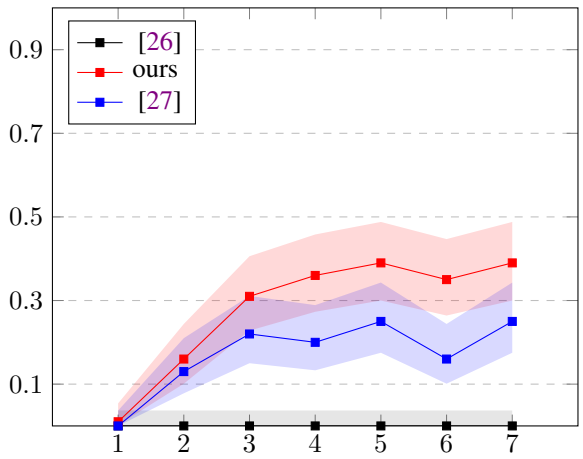


Figure 3: Success probabilities of the algorithms for varying σ. The x-axis and y-axis indicate t, where $\sigma = \frac{q}{2^t}$, and the success probabilities, respectively. We fix $n = 512, h = 8, m = \lfloor 2h \log_2 h \rceil = 48$. For these parameters, we experiment 100 times. Shaded regions indicate 95% Wilson score CIs.

Varying q. Despite varying the modulus q, our experiments demonstrate that changes in q do not significantly affect the success probability, as illustrated in Figure 4. The success rate of Algorithm 3 remains in the range 0.27–0.44 across all tested q values. This indicates that q is not a critical variable for Integer-LWE in our regime, lending confidence that results obtained with one choice of $\log_2 q$ transfer to other choices.

Varying m. We also performed experiments to measure the success probability of algorithms for varying the number of hints m. We fixed parameters $n = 512, \log_2 q = 100, h = 8$ and $\sigma = 3.19$, and conducted Algorithm 1, Algorithm 2 and Algorithm 3 100 times with various m. As shown in Figure 5, the success probability increases as m grows. In particular, Algorithm 2 reaches a success rate of 1.0 at $m = 4h \log_2 h = 96$, while Algorithm 3 reaches 1.0 at $m = 5h \log_2 h = 120$ (with 0.96 already at $m = 96$). Both algorithms substantially outperform Algorithm 1, which reaches only ≈ 0.25 at $m = 12h \log_2 h = 288$, confirming that the latter requires substantially more samples.

Varying h. We fixed $n = 512, \log_2 q = 100$ and $\sigma = 3.19$, and explored how the success probability of Algorithm 3 changes as the parameter h is increased. To determine the impact of h, we implemented two kinds of experiments. The first experiment (Figure 6) is to increase h with fixed number of samples $m = 48 = 2 \cdot 8 \cdot \log_2 8$. In this case, when h is less than or equal to 7, Algorithm 3 recovers the secret $\mathbf{s}$ with at least 0.5 probability. However, when h is larger than 7, then the success probability degrades rapidly since the fixed $m = 48$ becomes insufficient. The second experiment (Figure 7) is to increase h with various $m = \lfloor 2h \cdot \log_2 h \rceil$. In this case, as h increases, the success probability also increases. On the other hand, the performance of Algorithm 2 gets remarkably worse in this situation.

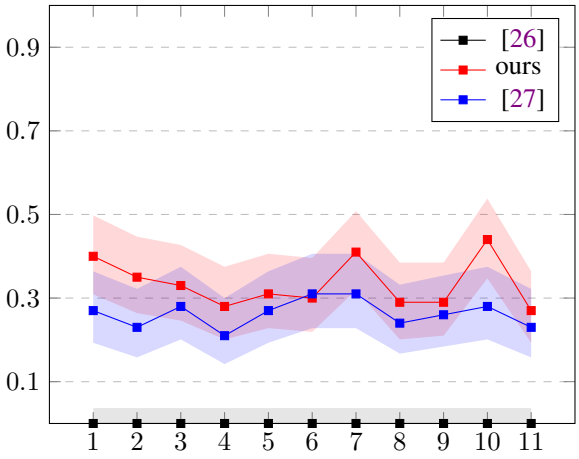


Figure 4: Success probabilities of the algorithms for varying q. The x-axis and y-axis indicate t, where $q = \mathsf{next_prime}(2^{t \cdot 100})$, and the success probabilities, respectively. We fix $n = 512, h = 8, m = 2 \cdot h \log_2 h$ and $\sigma = 3.19$. For these parameters, we experiment 100 times. Shaded regions indicate 95% Wilson score CIs.

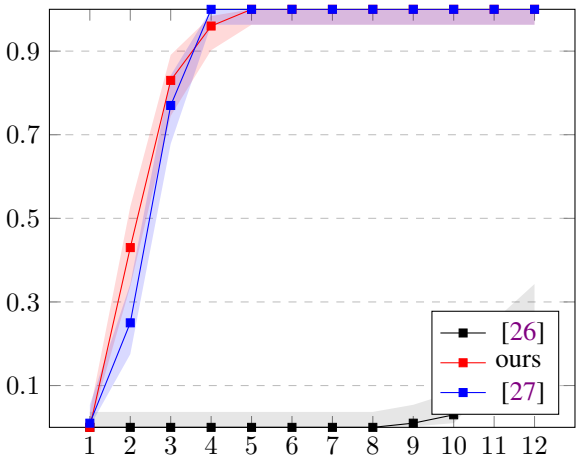


Figure 5: Success probability of varying m. The x-axis indicates t, where $m = \lfloor t \cdot h \log_2 h \rceil$ and $h = 8$. The y-axis means the success probability of algorithms. We fix $n = 512, q = \mathsf{next_prime}(2^{100}), \sigma = 3.19$. For these parameters, we experiment 100 times. Shaded regions indicate 95% Wilson score CIs.

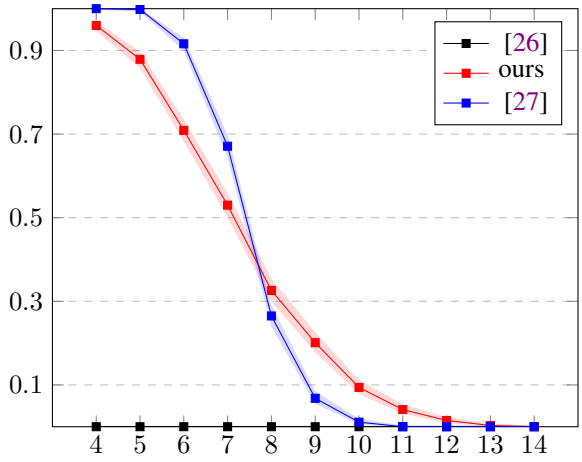


Figure 6: Success probabilities of the algorithms for varying h with *fixed* $m = 48$. The x-axis and y-axis indicate h and the success probabilities, respectively. We fix $n = 512, m = 48, q = \mathsf{next_prime}(2^{100})$ and $\sigma = 3.19$. For each parameter, we experiment 1100 times. Shaded regions indicate 95% Wilson score CIs. When $(h, m) = (14, 48)$, the probability drops to ≈ 0. Since m is held fixed while the problem becomes harder with growing h, performance degrades as expected.

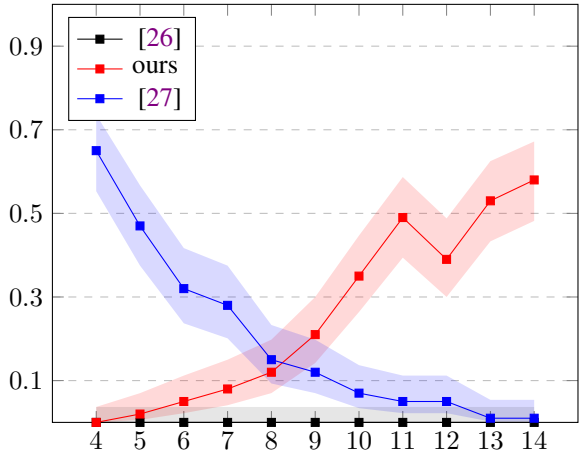


Figure 7: Success probabilities of the algorithms for varying h with *adaptive* $m = \lfloor 2h \log_2 h \rceil$. The x-axis and y-axis indicate h and the success probabilities, respectively. We fix $n = 1024, q = \mathsf{next_prime}(2^{100}), \sigma = 3.19$. Unlike Figure 6 where m is fixed, here m scales with h as $m = \lfloor 2h \log_2 h \rceil$. For each parameter, we experiment 100 times. Shaded regions indicate 95% Wilson score CIs. The increasing success rate of Algorithm 3 with h demonstrates that the $O(h \log_2 h)$ scaling provides sufficient samples even as h grows.

4.2. Side-Channel–Derived Hints in OpenFHE: An Illustrative Instantiation

This subsection demonstrates how our hint abstraction connects to realistic side-channel leakage. The scaling results of Section 4.1 are library-agnostic; the OpenFHE case study serves as an illustrative instantiation showing that the required hints can arise from concrete implementations.

4.2.1. Leakage from Side-channel Analysis. We present an example of side-channel leakage that provides a perfect/approximate hint in [34], an open-source library implementing state-of-the-art post-quantum FHE schemes.

Algorithm 5: Forward NTT

```
   Input: Vector a[0 .. n − 1], root table ω[·], modulus q
1  n ← length(a)
2  for ℓ = 1 upto n − 1 step ℓ do
3  |   s = 2ℓ
4  |   for i = 0 upto n − 1 step s do
5  |   |   for j = 0 upto ℓ − 1 do
6  |   |   |   u ← a[i + j]
7  |   |   |   v ← a[i + j + ℓ] · ω[ℓ + j]
8  |   |   |   sum ← u + v
9  |   |   |   if sum ≥ q then
10 |   |   |   |   sum ← sum − q
11 |   |   |   end
12 |   |   |   a[i + j] ← sum
13 |   |   |   diff ← u − v
14 |   |   |   if diff < 0 then
15 |   |   |   |   diff ← diff + q
16 |   |   |   end
17 |   |   |   a[i + j + ℓ] ← diff
18 |   |   end
19 |   end
20 end
21 return a
```

It performs its arithmetic over rings and employs the NTT to achieve efficiency. As a result, the NTT is invoked even during key generation, where it operates directly on the secret key.

Algorithm 5 illustrates the core logic of the `ForwardTransformToBitReverse` function, the NTT conversion routine used in OpenFHE's reference C++ implementation. In this function, a length-n vector `a` (which, in our attack, contains the secret key) is transformed using a precomputed table `rootOfUnityTable`. Each entry of that table is called a *twiddle factor*—a power of a primitive $2n$-th root of unity over $\mathbb{Z}_q$—used to perform the butterfly multiplications of the NTT.

Using a power consumption model, we introduce the following two types of side-channel leakage. **Side-channel leakage from conditional reductions**. Both conditional operations in the `ForwardNTT` routine introduce distinct sources of side-channel leakage. During the addition step of Algorithm 5, the code conditionally subtracts the modulus q after computing $u + v$ in line 9: if $u + v < q$ no correction occurs, whereas if $u+v \geq q$ an extra subtraction is executed.

This control-flow divergence produces a clear difference in power consumption. After simple thresholding or pattern classification it yields a secret-dependent leakage revealing whether the addition overflowed (i.e., whether $u + v \geq q$), which we will use later. An analogous effect arises in the subtraction step, where the code conditionally adds q to correct an underflow in $u - v$ in line 14.

Side-channel leakage during the multiplication. In each inner-loop iteration, the implementation multiplies the current intermediate value `a[i + j + l]` by the corresponding twiddle factor ω`[l + j]` in line 7. This multiplication is carried out over the integers (at the level of machine-word arithmetic), and only later is the value reduced modulo q. Consequently, a profiling side-channel attack targeting the multiplier can be used to recover the value of the product in $\mathbb{Z}$. Since the root of unity is public and known to the adversary, learning this product directly yields a precise hint about the corresponding intermediate coefficient of the secret key that was multiplied by this root of unity.

Combining both leakages: construction of a perfect hint. Fix an iteration of the innermost loop of `ForwardNTT`, and let q be the NTT modulus. Focus on the coefficient stored in `a[i + j + len]` in this iteration. Let $x \in \mathbb{Z}$ denote the value of this coefficient if all intermediate computations were carried out over the integers, without any reduction modulo q. The implementation, however, only stores its residue

$$x' \coloneqq x \bmod q \in \{0, \ldots, q-1\}$$

in `a[i + j + len]` just before the multiplication by the corresponding twiddle factor. Let ω denote this (public) twiddle factor, i.e., the entry `rootOfUnityTable[len + j]` used in that iteration. The side-channel leakages that we observe will allow us to recover more information about x than just its residue x'.

From the multiplication leakage described above, the adversary learns the integer product

$$M = \omega \cdot x' \in \mathbb{Z}.$$

Since ω is public and invertible modulo q, this determines the residue $x' = x \bmod q$.

From the leakage on conditional reductions, the adversary also learns how many times q has been added to or subtracted from this coefficient along its computation path. Equivalently, the branch trace reveals an integer $k \in \mathbb{Z}$ such that the unreduced value satisfies

$$x = x' + kq.$$

Combining both pieces of information, the side-channel transcript for this iteration can be summarized as the pair (M, k). For each such pair there is a unique integer x consistent with it, and hence a unique secret-dependent intermediate value. In the terminology of our leakage model, the combined side channel thus yields a *perfect hint* about the corresponding secret-dependent coefficient.

In summary. At the level of a single coefficient, the multiplication leakage reveals the reduced value $x' = x \bmod q$, while the conditional-reduction leakage reveals the corresponding quotient $k \in \mathbb{Z}$, so that the unreduced integer is reconstructed exactly as $x = x' + kq$. Thus, under this idealized leakage model, for each visited coefficient the side channel determines its true (unreduced) value without ambiguity. Viewing the NTT as a linear map, the final NTT output can be written as $\mathbf{x} = W\mathbf{s}$, where $\mathbf{s}$ is the secret-key vector and W is the public NTT matrix whose rows $\mathbf{v}_t$ consist of powers of the public twiddle factor ω. Each recovered intermediate or output coefficient x_t is therefore of the form $x_t = \langle \mathbf{v}_t, \mathbf{s} \rangle$. Consequently, every side-channel reconstruction of a coefficient x_t can be interpreted as a *perfect hint* $(\mathbf{v}_t, x_t)$ in the sense of our hint framework. If the leakage were noisy so that we only observe $\tilde{x}_t = x_t + e_t$, the same mechanism would instead yield an *approximate hint* of the form $\langle \mathbf{v}_t, \mathbf{s} \rangle + e_t$.

TABLE 3: Experimental results of algorithms on target parameters. Given parameter $(n, \log_2 q, h)$, we deploy $\sigma = 3.19$ and $m = \lfloor 2h \log_2 h \rfloor$. The Hints column specifies whether hints are approximate or perfect. Three values in the 'Algorithms' column show the success probability of Algorithm 1, Algorithm 2 and Algorithm 3, respectively.

Schemes	Hints	Parameters			Algorithms
		n	$\log_2 q$	h	(Algorithm 1, Algorithm 2, Algorithm 3)
[7], [28]	approximate	2^{15}	161	32	(0, 0.05, 0.43)
		2^{15}	161	64	(0, 0, 0.85)
[12]	approximate	2^{14}	66	32	(0, 0.07, 0.7)
		2^{16}	438	32	(0, 0.01, 0.15)
[34]	perfect	2^{14}	40	128	(0, 0, 1)
		2^{15}	40	192	(0, 0, 1)

4.2.2. Experimental Results on concrete Parameters. Table 3 compares the success rates of Algorithm 1, Algorithm 2, and Algorithm 3 on concrete parameter sets, including the OpenFHE parameters from Section 4.2.1. To estimate the success probability, we ran experiments 100 times. For the settings from [7], [28] and [12], we evaluated using *approximate* hints. For OpenFHE, **only the perfect-hint setting yielded successful recovery** in our preliminary tests; extending to approximate hints remains open and would require a more refined leakage model. We therefore run the OpenFHE experiments under perfect hints only. Our algorithm solves all instances within a one-hour timeout, whereas Algorithm 2 and Algorithm 1 did not finish within the same time limit in our implementation.

We note that prior side-channel work on FHE libraries has focused on SEAL [18], [19], [20]. Our contribution here is not an "OpenFHE-first" claim, but rather the quantitative scaling insight that sparse secrets reduce the required leakage from $n/2$ to $O(h \log_2 h)$ hints.

Code availability. All source code for the experiments in this paper is publicly available at:

https://github.com/ChangminLee-kr/sparseLWE.git

5. Ethical considerations

This work analyzes potential side-channel leakage in the open-source library OpenFHE and argues that secret-key recovery could be feasible under realistic lab conditions. All experiments were performed on hardware and software under our control in a closed lab environment; we did not target third-party systems or interact with real users. We collected only synthetic keys and measurement traces and did not handle any personally identifiable information (PII).

The leakage class (secret-dependent control flow during NTT-style modular adjustments) and the standard mitigation (constant-time/branchless implementation) are well established in the side-channel literature. In our evaluation we identified a concrete instantiation in OpenFHE's NTT path; we have shared reproducible details and suggested mitigations with the maintainers. To avoid gratuitous harm, the submission and our planned artifact release omit exploit-enabling parameters and limit materials to what is necessary for scientific reproducibility.

6. LLM usage considerations

LLMs were used for editorial purposes in this manuscript, and all outputs were inspected by the authors to ensure accuracy and originality. We used an LLM-based assistant (ChatGPT) only for grammar and stylistic editing of author-written text. All technical content, data, and claims originate from the authors, and the authors verified the final manuscript for accuracy and integrity. No sensitive or proprietary data were provided to the tool.

Acknowledgments

Jiseung Kim is supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (No. RS-2023-00273635). Changmin Lee is supported by Korea University grants and by the Creation of the Quantum Information Science R&D Ecosystem (Grant No. RS-2023-NR068116) through the National Research Foundation of Korea (NRF) funded by the Korean government (Ministry of Science and ICT).

References

[1] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *Journal of the ACM (JACM)*, vol. 56, no. 6, pp. 1–40, 2009.

[2] C. Gentry, C. Peikert, and V. Vaikuntanathan, "Trapdoors for hard lattices and new cryptographic constructions," in *Proceedings of the fortieth annual ACM symposium on Theory of computing*, 2008, pp. 197–206.

[3] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(leveled) fully homomorphic encryption without bootstrapping," in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, 2012, pp. 309–325.

[4] V. Lyubashevsky, "Lattice signatures without trapdoors," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2012, pp. 738–755.

[5] S. Gorbunov, V. Vaikuntanathan, and D. Wichs, "Leveled fully homomorphic signatures from standard lattices," in *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, 2015, pp. 469–477.

[6] S. Agrawal, B. Libert, and D. Stehlé, "Fully secure functional encryption for inner products, from standard assumptions," in *Annual International Cryptology Conference*. Springer, 2016, pp. 333–362.

[7] J.-P. Bossuat, J. Troncoso-Pastoriza, and J.-P. Hubaux, "Bootstrapping for approximate homomorphic encryption with negligible failure-probability by using sparse-secret encapsulation," in *International Conference on Applied Cryptography and Network Security*. Springer, 2022, pp. 521–541.

[8] K. Han and D. Ki, "Better bootstrapping for approximate homomorphic encryption," in *Cryptographers' Track at the RSA Conference*. Springer, 2020, pp. 364–390.

[9] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2017, pp. 409–437.

[10] J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song, "Bootstrapping for approximate homomorphic encryption," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2018, pp. 360–384.

[11] Y. Lee, D. Micciancio, A. Kim, R. Choi, M. Deryabin, J. Eom, and D. Yoo, "Efficient fhew bootstrapping with small evaluation keys, and applications to threshold homomorphic encryption," in *Advances in Cryptology–EUROCRYPT 2023: 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part III*. Springer, 2023, pp. 227–256.

[12] Y. Bae, J. H. Cheon, J. Kim, and D. Stehlé, "Bootstrapping bits with ckks," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2024, pp. 94–123.

[13] N. Howgrave-Graham, "A hybrid lattice-reduction and meet-in-the-middle attack against NTRU," in *Annual International Cryptology Conference*. Springer, 2007, pp. 150–169.

[14] M. Hhan, J. Kim, C. Lee, and Y. Son, "How to meet ternary lwe keys on babai's nearest plane," Cryptology ePrint Archive, Paper 2022/1473, 2022, https://eprint.iacr.org/2022/1473. [Online]. Available: https://eprint.iacr.org/2022/1473

[15] M. R. Albrecht, "On dual lattice attacks against small-secret LWE and parameter choices in HElib and SEAL," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2017, pp. 103–129.

[16] J. H. Cheon, M. Hhan, S. Hong, and Y. Son, "A hybrid of dual and meet-in-the-middle attack on sparse and ternary secret LWE," *IEEE Access*, vol. 7, pp. 89 497–89 506, 2019.

[17] R. Primas, P. Pessl, and S. Mangard, "Single-trace side-channel attacks on masked lattice-based encryption," in *Cryptographic Hardware and Embedded Systems–CHES 2017: 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*. Springer, 2017, pp. 513–533.

[18] F. Aydin, E. Karabulut, S. Potluri, E. Alkim, and A. Aysu, "Reveal: Single-trace side-channel leakage of the seal homomorphic encryption library," in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2022, pp. 1527–1532.

[19] N. Drucker and T. Pelleg, "Timing leakage analysis of non-constant-time ntt implementations with harvey butterflies," in *International Symposium on Cyber Security, Cryptology, and Machine Learning*. Springer, 2022, pp. 99–117.

[20] F. Aydin and A. Aysu, "Exposing side-channel leakage of seal homomorphic encryption library," in *Proceedings of the 2022 Workshop on Attacks and Solutions in Hardware Security*, 2022, pp. 95–100.

[21] D. Dachman-Soled, L. Ducas, H. Gong, and M. Rossi, "LWE with side information: attacks and concrete security estimation," in *Annual International Cryptology Conference*. Springer, 2020, pp. 329–358.

[22] "Microsoft SEAL (release 4.1)," https://github.com/Microsoft/SEAL, Jan. 2023, microsoft Research, Redmond, WA.

[23] A. May and J. Nowakowski, "Too many hints–when lll breaks lwe," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2023, pp. 106–137.

[24] D. Dachman-Soled, H. Gong, T. Hanson, and H. Kippen, "Revisiting security estimation for lwe with hints from a geometric perspective," in *Annual International Cryptology Conference*. Springer, 2023, pp. 748–781.

[25] J. Bootle, C. Delaplace, T. Espitau, P.-A. Fouque, and M. Tibouchi, "Lwe without modular reduction and improved side-channel attacks against bliss," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2018, pp. 494–524.

[26] N. Nolte, M. Malhou, E. Wenger, S. Stevens, C. Li, F. Charton, and K. Lauter, "The cool and the cruel: separating hard parts of lwe secrets," in *International Conference on Cryptology in Africa*. Springer, 2024, pp. 428–453.

[27] J. H. Cheon, H. Choe, A. Passelègue, D. Stehlé, and E. Suvanto, "Attacks against the ind-cpa^{D} security of exact fhe schemes," in *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, 2024, pp. 2505–2519.

[28] "Lattigo," https://github.com/tuneinsight/lattigo, accessed: 2022-10-07.

[29] Y. Bae, J. H. Cheon, J. Kim, J. H. Park, and D. Stehlé, "Hermes: Efficient ring packing using mlwe ciphertexts and application to transciphering," in *Annual International Cryptology Conference*. Springer, 2023, pp. 37–69.

[30] W.-j. Lu, Z. Huang, C. Hong, Y. Ma, and H. Qu, "Pegasus: bridging polynomial and non-polynomial evaluations in homomorphic encryption," in *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021, pp. 1057–1073.

[31] J.-P. Bossuat, C. Mouchet, J. Troncoso-Pastoriza, and J.-P. Hubaux, "Efficient bootstrapping for approximate homomorphic encryption with non-sparse keys," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2021, pp. 587–617.

[32] J.-W. Lee, E. Lee, Y. Lee, Y.-S. Kim, and J.-S. No, "High-precision bootstrapping of RNS-CKKS homomorphic encryption using optimal minimax polynomial approximation and inverse sine function," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2021, pp. 618–647.

[33] Y. Lee, J.-W. Lee, Y.-S. Kim, Y. Kim, J.-S. No, and H. Kang, "High-precision bootstrapping for approximate homomorphic encryption by error variance minimization," in *Advances in Cryptology–EUROCRYPT 2022: 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Trondheim, Norway, May 30–June 3, 2022, Proceedings, Part I*. Springer, 2022, pp. 551–580.

[34] A. A. Badawi, A. Alexandru, J. Bates, F. Bergamaschi, D. B. Cousins, S. Erabelli, N. Genise, S. Halevi, H. Hunt, A. Kim, Y. Lee, Z. Liu, D. Micciancio, C. Pascoe, Y. Polyakov, I. Quah, S. R.V., K. Rohloff, J. Saylor, D. Suponitsky, M. Triplett, V. Vaikuntanathan, and V. Zucca, "OpenFHE: Open-source fully homomorphic encryption library," Cryptology ePrint Archive, Paper 2022/915, 2022, https://eprint.iacr.org/2022/915. [Online]. Available: https://eprint.iacr.org/2022/915

[35] NIST, "Federal information processing standards publication 203: Module-lattice-based key-encapsulation mechanism (ml-kem)," https://csrc.nist.gov/pubs/fips/203/final, 2024.

[36] ——, "Federal information processing standards publication 204: Module-lattice-based digital signature algorithm (ml-dsa)," https://csrc.nist.gov/pubs/fips/204/final, 2024.

[37] M. ElGhamrawy, M. Azouaoui, O. Bronchain, J. Renes, T. Schneider, M. Schönauer, O. Seker, and C. van Vredendaal, "From mlwe to rlwe: A differential fault attack on randomized & deterministic dilithium," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2023, no. 4, pp. 262–286, 2023.

[38] P. Azevedo-Oliveira, A. Calle Viera, B. Cogliati, and L. Goubin, "Finding a polytope: A practical fault attack against dilithium," in *IACR International Conference on Public-Key Cryptography*. Springer, 2025, pp. 259–283.

[39] P.-A. Fouque, D. Marion, Q. Nguyen, and A. Wallet, "Avengers assemble! supervised learning meets lattice reduction: A single power trace attack against crystals-kyber key generation," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2025, no. 4, pp. 409–436, 2025.

[40] J. Hermelink, E. Mårtensson, S. Samardjiska, P. Pessl, and G. D. Rodosek, "Belief propagation meets lattice reduction: Security estimates for error-tolerant key recovery from decryption errors," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2023, no. 4, pp. 287–317, 2023.

[41] J. Hermelink, S. Streit, E. Mårtensson, and R. Petri, "A generic framework for side-channel attacks against lwe-based cryptosystems," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2025, pp. 3–32.

[42] S. Damm, A. Fischer, A. May, S. Marzougui, L. Schwarz, H. Seidler, J.-P. Seifert, J. Thietke, and V. Q. Ulitzsch, "Solving concealed ilwe and its application for breaking masked dilithium," *Cryptology ePrint Archive*, 2025.

[43] E. Krahmer, P. Pessl, G. Land, and T. Güneysu, "Correction fault attacks on randomized crystals-dilithium," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2024, no. 3, pp. 174–199, 2024.

[44] V. Q. Ulitzsch, S. Marzougui, A. Bagia, M. Tibouchi, and J.-P. Seifert, "Loop aborts strike back: Defeating fault countermeasures in lattice signatures with ilp," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2023, no. 4, pp. 367–392, 2023.

[45] A. S. B.-F. L. Breaks, "Profiling side-channel attacks on dilithium," in *Selected Areas in Cryptography: 29th International Conference, SAC 2022, Windsor, ON, Canada, August 24–26, 2022, Revised Selected Papers*, vol. 13742. Springer Nature, 2024, p. 3.

[46] B. Applebaum, D. Cash, C. Peikert, and A. Sahai, "Fast cryptographic primitives and circular-secure encryption based on hard learning problems," in *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*, ser. Lecture Notes in Computer Science, vol. 5677. Springer, 2009, pp. 595–618.

[47] W. Stein *et al.*, *Sage Mathematics Software (Version 9.0)*, The Sage Development Team, 2020, `http://www.sagemath.org`.

Appendix A.
Meta-Review

The following meta-review was prepared by the program committee for the 2026 IEEE Symposium on Security and Privacy (S&P) as part of the review process as detailed in the call for papers.

A.1. Summary

This paper studies the security of LWE with sparse ternary secrets under side-channel leakage modeled as (perfect or approximate) linear hints. It gives a new Integer-LWE formulation and a low-hint recovery approach, and evaluates the resulting risk on practically motivated parameter sets.

A.2. Scientific Contributions

- Identifies an Impactful Vulnerability.
- Provides a Valuable Step Forward in an Established Field.

A.3. Reasons for Acceptance

1) Identifies an impactful vulnerability: With sparse ternary secrets, a small number of linear hints can be enough for practical key recovery, showing a previously unexplored vulnerability of sparsification under leakage.
2) Provides a valuable step forward in an established field: Reviewers acknowledge the technical soundness and novelty of the low-hint scaling insight, and it is supported by experiments on realistic parameter ranges.

A.4. Noteworthy Concerns

1) The analytical guarantees remain largely heuristic and the stated success bounds are loose; this can be strengthened by making the heuristic nature explicit and/or tightening the bounds where possible.