EMAI

Erasmus Mundus joint
Master in Artificial Intelligence

Universitat Pompeu Fabra Barcelona · Sapienza Università di Roma · Radboud Universiteit · University of Ljubljana · Co-funded by the European Union

ML-based Attack on Module-LWE

Assessing the security of module-lattice-based schemes

Thesis MSc in Artificial Intelligence

Author:
Cristian Bassotto

Supervisor:
dr. Stjepan Picek

Second reader:
prof. Lejla Batina

August 2025

Contents

Abstract

The dawn of quantum computing compromises the security foundations of classical public-key cryptography, motivating NIST's recent standardization of post-quantum cryptographic schemes. Two of the four selected algorithms are based on the hardness of the Module Learning With Errors (Module-LWE) problem. As Module-LWE becomes the mathematical primitive of next-generation cryptographic standards, its robustness needs to be evaluated not just against classical and quantum algorithms, but also against novel AI-powered attacks.

In this work, we introduce NoMod ML-Attack, a hybrid white-box cryptanalytic attack that avoids the difficulty of directly modelling modular reduction in Module-LWE. Instead, NoMod views modular wraparounds as a type of statistical corruption and reformulates secret recovery as a robust linear estimation problem. Our method begins with advanced lattice preprocessing, improved using several optimizations. In particular, we propose (i) a reduced-vector saving strategy that accumulates and reuses short vectors during tours, and (ii) an algebraic amplification technique that exploits Ring-LWE automorphisms to expand the pool of usable samples. After this preprocessing step, we train robust linear estimators based on Tukey's Biweight loss, prioritizing direct secret recovery and sample-efficient methods over transformer-based architectures.

Our experimental tests demonstrate that NoMod broadens the range of ML-based cryptanalysis. We achieve complete recovery of binary secrets for dimension $n = 350$, recovery of sparse binomial secrets at $n = 256$, and successful recovery of sparse secrets in CRYSTALS-Kyber settings with parameters $(n, k) = (128, 3)$ and $(256, 2)$. Throughout these regimes, NoMod outperforms classical lattice-only techniques and, in some instances, produces results competitive with transformer-based frameworks, such as SALSA [1] and SALSA PICANTE [2]. Finally, to enable future research, we release our open-source implementation of NoMod ML-Attack to support continued study and benchmarking.

Keywords: Post-Quantum Cryptography; Module-LWE; Machine Learning Attacks; Lattice Cryptanalysis

Chapter 1

Introduction

The dawn of quantum computing presents a significant and growing threat to current cryptographic systems, many of which will be vulnerable to decryption through quantum-based attacks. At the heart of this risk is *Shor's algorithm*, a quantum-based algorithm developed in 1994 by Peter Shor that could efficiently factor large integers and compute discrete logarithms. These two mathematical problems are computationally challenging for classical computers when the input size is large. In particular, while classical algorithms to factor integers, such as the General Number Field Sieve (GNFS), run in *sub-exponential time* $L_n[1/3, (64/9)^{1/3}]$, Shor's algorithm could run in *polynomial time* $O((\log N)^3)$, when implemented on a sufficiently robust quantum computer [3, 4].

This development raises a serious threat to the security assumptions behind widely used public-key cryptographic schemes such as RSA, Elliptic Curve Cryptography (ECC), and Diffie-Hellman key exchange. These algorithms are central to the *Public Key Infrastructure (PKI)* that secures virtually all modern digital communications. RSA, for example, remains widely used for digital signatures in TLS certificates, code signing, and legacy PGP encryption. However, its use in key exchange has been replaced in favour of forward-secure alternatives. On the other hand, ECC is widely used in modern systems due to its efficiency, allowing secure key exchange (ECDHE) in TLS, authentication in mobile/IoT devices (FIDO2), and blockchain transactions (e.g., Bitcoin's secp256k1). Diffie-Hellman (DH) and its elliptic-curve variant (ECDH) support ephemeral session establishment in TLS, SSH, and IPsec, guaranteeing forward secrecy. Thus, the compromise of these algorithms endangers not only internet traffic, but also critical infrastructure, financial systems, medical records, and defence communications.

The security of RSA relies on the difficulty of factoring large semi-prime numbers. In contrast, Elliptic Curve Cryptography (ECC) and the Diffie-Hellman key exchange are based on the challenges posed by the discrete logarithm problem, either in finite fields or on elliptic curves. However, Shor's algorithm can efficiently solve both of these problems. This means that once sufficiently powerful quantum computers become available, all three cryptographic systems could be compromised, making encrypted data vulnerable to quantum-enabled attacks.

1.1 The Quantum Era

Although no practical quantum computers currently exist with the required number of qubits and sufficiently low error rates to implement Shor's algorithm at the scale needed to compromise modern public-key schemes, ongoing improvements in quantum hardware, fault-tolerant architectures, and algorithmic optimization indicate that this capability may arise within the following decades [5]. Early analyses suggested that factoring RSA-2048 would require nearly a billion physical qubits due to the costly overhead of quantum error correction [6]. However, subsequent research has particularly refined these projections. In 2021, Gidney and Ekerå showed that, with depth-optimized circuit designs and improved fault tolerance, RSA-2048 could be factored using around 20 million physical qubits in about eight hours [7]. A 2024 study by Chevignard, Fouque, and Schrottenloher further reduced this to 1,730 logical qubits under efficient syndrome extraction [8], while Google's 2025 analysis projected that approximately 1,399 logical qubits, backed by nearly one million physical qubits, could obtain similar results within a week using realistic error thresholds and gate counts [9]. Although current systems, such as IBM's 1,121-qubit Condor processor, remain far from these benchmarks, leading developers, including IBM, Quantinuum, and PsiQuantum, are pursuing roadmaps targeting thousands of logical qubits by the start of the next decade [10]. If these predictions hold, cryptographically relevant quantum computers capable of breaking RSA-2048 may reasonably appear between 2035 and 2040, emphasising the urgent need to migrate to quantum-safe cryptographic standards well in advance [11].

Another key aspect in the context of quantum computing is the concept known as "Harvest now, decrypt later" (HNDL). This thread model describes a scenario where encrypted information transmitted today may be intercepted and stored by adversaries with the intention of decrypting it in the future once quantum computing matures sufficiently [12]. Consequently, data that are currently considered secure may become vulnerable when advanced quantum computers are developed. This scenario presents a serious risk of future data breaches concerning sensitive communications that were assumed to be secure at the time of transmission. The issue is especially critical for information that requires long-term confidentiality, such as classified government documents, personal medical records, and proprietary scientific or technical research. The potential for broad disruption of secure communications, identity systems, and digital signatures highlights the urgent need to transition to cryptographic standards that can resist quantum threats.

1.2 Post-Quantum Cryptography

In response to this emerging challenge, researchers, cryptographers, and standardisation bodies have mobilized to design, evaluate, and implement cryptographic algorithms that can resist quantum attacks. This new generation of cryptography, known as post-quantum cryptography (PQC), embraces cryptographic primitives that can be securely installed on classical hardware while being resilient against quantum adversaries.

1.2.1 NIST Post-Quantum Cryptography Standardization

The National Institute of Standards and Technology (NIST) launched the Post-Quantum Cryptography Standardization Process to identify public-key algorithms resistant to adversaries equipped with large-scale quantum computers. This multi-year effort combines broad public participation, sustained cryptanalysis, and NIST's own evaluation of security, efficiency, and practical implementability. Beginning with a 2016 call for proposals, candidates progressed through successive public rounds of scrutiny, with weaker or less practical schemes eliminated at each stage.

In 2022, NIST announced its first algorithm selections: the Module-Learning With Errors (Module-LWE or MLWE) Key Encapsulation Mechanism (KEM) CRYSTALS-Kyber, the Module-LWE signature scheme CRYSTALS-Dilithium, and the hash-based signature scheme SPHINCS+. These were published later as Federal Information Processing Standards (FIPS) on August 13, 2024:

- **FIPS 203** – Module-Lattice-Based Key-Encapsulation Mechanism (CRYSTALS-Kyber)
- **FIPS 204** – Module-Lattice-Based Digital Signature (CRYSTALS-Dilithium)
- **FIPS 205** – Stateless Hash-Based Digital Signature (SPHINCS+)

In parallel, NIST carried on a fourth round focused on diversifying beyond lattice-based primitives. Code-based (BIKE, Classic McEliece, HQC) and isogeny-based (SIKE) KEMs were analysed in depth for decryption failure behaviour, side-channel robustness, and cryptanalytic maturity. The 2022 break of SIKE underscored the importance of ongoing public analysis, while HQC demonstrated a stable and well-understood security profile. On March 11, 2025, NIST selected HQC for standardization as an additional KEM, adding a code-based primitive to the PQC suite. Classic McEliece remained cryptanalytically sound, but was suspended due to its impractically large public keys.

By the end of 2024, the NIST PQC standards portfolio comprised FIPS 203, 204, and 205, with HQC joining in 2025. This mix reflects a deliberate balance: Module-LWE schemes provide the backbone for efficient and deployable quantum-resistant encryption and signatures, while hash- and code-based schemes serve as critical hedges against unexpected weaknesses.

1.3 Motivation and Research Question

Given that two of the four standardized post-quantum algorithms are built on the Module-LWE problem, understanding and continually reevaluating the security of this family is critical. The hardness of MLWE has become a foundation of post-quantum security, with classical and quantum attacks significantly studied and security margins broadly established under current models. However, the rapid progress of artificial intelligence has introduced a new dimension: AI-powered approaches to cryptanalysis, still in their infancy, have shown an extraordinary ability to uncover patterns that resist traditional analysis, which raises a compelling question: could machine learning methods reduce the hardness of MLWE, either by directly learning modular correlations or by exploiting alternative representations that uncover hidden structures? The consequences of such developments would be profound: they could either reveal unforeseen vulnerabilities in standardized PQC algorithms or strengthen confidence in their resilience by showing resistance even against AI-assisted strategies. In either case, carefully testing MLWE against AI-based attacks is essential now, before these systems are implemented on a global scale.

A central technical barrier in applying machine learning to the MLWE problem lies in the modular arithmetic inherent to its structure. In contrast to classical regression or signal recovery tasks, where adversaries could directly exploit linear relationships between variables, the reduction modulo q disrupts linearity and introduces non-linear wrap-around effects that are difficult for neural models to capture. This raises several questions about how to represent and process MLWE data within an AI framework. One avenue is the use of the *Number Theoretic Transform* (NTT), which, as the finite-field analogue of the Fourier Transform, might expose alternative representations in which hidden structure becomes affordable. A more radical approach would be to bypass modular arithmetic entirely. Exploring these directions is crucial for determining whether AI can realistically jeopardize MLWE-based cryptography or whether the modular barrier constitutes a fundamental limitation to machine learning attacks.

This leads us to two guiding research questions of this work:

(1) Can machine learning, aided by algebraic embeddings, extract useful structure directly from modular MLWE samples and thereby recover information about the secret?

(2) Or is it necessary to bypass modular data altogether and reframe secret recovery as a learning problem in the noisy but linear domain?

By systematically pursuing both directions, we aim to determine whether AI-powered cryptanalysis can realistically manipulate modular MLWE data or whether meaningful attacks require stepping outside the modular domain.

1.4 Main Contributions

This thesis advances the study of Module-LWE security under machine learning-based attacks by combining lattice reduction techniques with robust statistical learning. The main contributions are as follows:

1. *A non-modular attack strategy.* We introduce a “NoMod” approach that deliberately avoids modular arithmetic by treating wrap-around effects as statistical outliers. By re-framing the problem into a noisy, yet linear domain, this strategy enables efficient secret recovery through lattice reduction combined with regression, offering a light alternative to black-box transformer-based attacks.

2. *Optimized preprocessing for learnability.* We perform a systematic study of the preprocessing pipeline, identifying key trade-offs between reduction quality, sample size, and computational cost. Several optimizations are proposed, including progressive BKZ strategies with moderate block sizes, analytical determination of the optimal sample count per reduced matrix, and accumulation of short vectors across multiple reduction tours.

3. *Algebraic amplification for Ring-LWE.* We exploit the automorphism structure of polynomial rings to amplify a small set of reduced samples into a full orbit, dramatically lowering the number of lattice reductions required. This technique highlights how algebraic properties of the problem can be directly harnessed to strengthen machine–learning–based attacks.

4. *Evaluation of robust regression models.* Our analysis demonstrates that robust regression can be tuned to reliably extract secrets from noisy reductions, even in the presence of modular wrap-around effects. It establishes robust linear methods as a foundation for AI-powered cryptanalysis, contrasting with black-box and computationally expensive transformer-based models.

5. *Experimental validation against state-of-the-art.* We validate our methods on various Kyber parameter sets and on novel AI-based attacks such as SALSA, PICANTE, and its variants, showing that our regression-based pipeline recovers secrets faster and with orders of magnitude fewer samples than some transformer-based attacks. These results establish that careful preprocessing, combined with robust regression, could outperform complex neural architectures in both efficiency and practicality.

Chapter 2

Background

This chapter provides the mathematical and conceptual foundations required for the remainder of this thesis. We begin with an overview of the main PQC algorithms that have been invented and used in practice. Then we continue with a detailed presentation of the LWE problem, its formal definitions, and its connection to lattice theory, as well as lattice reduction algorithms. We also introduce two critical variants, *Ring-LWE* and *Module-LWE*, and explain the relationship between Module-LWE and the standard LWE formulation.

Next, we focus on the CRYSTALS-Kyber key encapsulation mechanism, discussing, in particular, its key generation process with its specific NTT, used in efficient polynomial multiplication.

The chapter then surveys known attacks on LWE, starting with classical cryptanalytic methods and more recent AI-based approaches. Finally, we introduce the machine learning algorithm and losses that will be employed later in this work.

2.1 Post-Quantum Cryptography Algorithms

Rather than depending on a single primitive, PQC encompasses several distinct families of cryptographic schemes. Each is built on a different complex mathematical problem, offers different performance trade-offs, and is suitable for different cryptographic tasks, such as KEMs, encryption, or digital signatures. The following sections provide an overview of the most prominent PQC families, their intended use cases, underlying hard problems, and representative protocols.

Code-Based Cryptography

Code-based schemes are among the oldest proposals for quantum-resistant public-key encryption. Their security rests on the *decoding random linear codes* problem: given a noisy codeword from a random linear code, recover the original message. This problem is NP-hard, and despite more than four decades of cryptanalytic effort, no polynomial-time algorithms, nor classic or quantum, are known.

These schemes are used for *encryption* and *Key Encapsulation Mechanisms (KEMs)*. The McEliece cryptosystem, proposed in 1978, remains a standard: its large public keys (hundreds of kilobytes) are offset by rapid encryption and decryption and a long track record of resisting attacks. Modern variants include Classic McEliece, BIKE, and HQC, which refine parameter sets and code structures for efficiency while maintaining security.

Multivariate Quadratic (MQ) Cryptography

MQ cryptography bases its security on the *Multivariate Quadratic problem*, also known as *Polynomial System Solving (PoSSo)*: given a system of multivariate quadratic equations over a finite field, find a solution. This problem is NP-hard and has resisted all known efficient solvers, even in the presence of quantum computing.

MQ schemes are originally implemented for *digital signatures*. Examples include the Rainbow signature scheme, UOV (Unbalanced Oil and Vinegar), and MQDSS, which targets lightweight embedded applications. MQ signatures are generally fast and compact, but historically fragile: careful parameter selection is essential to avoid vulnerabilities.

Hash-Based Signatures

Hash-based signature schemes represent the most conservative branch of PQC. Their security reduces to the preimage and collision resistance of a cryptographic hash function, one of the most extensively studied primitives in the field. Because they do not rely on an algebraic structure, they offer strong confidence against both classical and quantum adversaries, assuming the underlying hash remains secure.

These schemes are used exclusively for *digital signatures*. Modern stateless designs such as SPHINCS+ overcome the state-management complexity of earlier stateful systems, like XMSS (RFC 8391) and LMS (RFC 8554).

Lattice-Based Cryptography

Lattice-based cryptography is definitely the most versatile and active area of PQC research, supporting *signatures*, *encryption*, and *KEMs*. Its security relies on well-studied problems, such as the *Shortest Vector Problem (SVP)* and the *Learning With Errors (LWE)* problem, including efficient structured variants, like *Ring-LWE* and *Module-LWE*. Importantly, multiple lattice problems admit worst-case to average-case reductions, providing firm theoretical foundations.

Notable protocols include the already presented CRYSTALS-Kyber and CRYSTALS-Dilithium, as well as the long-standing NTRU encryption scheme. Lattice-based primitives typically offer an attractive balance of key sizes, ciphertexts, and computational efficiency.

Isogeny-Based Cryptography

Isogeny-based cryptography is the youngest of the prominent PQC families and remains an active area of research. Its security is based on the hardness of finding an *isogeny* (a special type of morphism) between two super-singular elliptic curves. This problem appears resistant to known classical and quantum algorithms, with no sub-exponential attacks found in the super-singular setting.

Isogeny-based protocols are commonly applied to *KEMs*, *non-interactive key exchange (NIKE)*, and occasionally *signatures*. CSIDH offers a commutative isogeny framework for key exchange with relatively small public keys, which is appealing for scenarios with restricted bandwidth.

2.2 Learning With Errors

The LWE problem, firstly introduced by Regev in 2005 [13, 14], is a central hardness assumption in lattice-based cryptography and serves as the foundation for many post-quantum secure cryptographic primitives. Informally, LWE can be seen as the problem of solving a noisy system of linear equations over a finite field. Without noise, such systems are easily solvable in polynomial time using standard linear algebra. The introduction of slight random noise makes the problem computationally difficult under appropriate parameter choices, even for quantum adversaries.

Definition 2.2.1 (Search-LWE)**.** Let $\mathbb{Z}_q = \{0, 1, \ldots, q-1\}$ denote the ring of integers modulo q. Fix positive integers n and m, an integer modulus $q \geq 2$, and an error bound $B \ll q/2$. Let:

- $s \xleftarrow{\$} \mathbb{Z}_q^n$ be a *secret vector* chosen uniformly at random,
- $A \xleftarrow{\$} \mathbb{Z}_q^{m \times n}$ be a uniformly random matrix,
- $e \xleftarrow{\$} [-B, B]^m$ be an *error vector* whose entries are small integers.

We compute:

$$b = As + e \pmod q.$$

The problem *search-LWE*, denoted $\mathrm{LWE}(m, n, q, B)$, is: given (A, b), recover the secret vector s.

The parameters must be chosen carefully. If $B = 0$ (i.e., $e = 0$), then solving LWE reduces to solving a system of modular linear equations. Conversely, if $B \geq (q-1)/2$, the error term completely masks the signal, making recovery of the s information theoretically impossible. In cryptographic applications, B is chosen so that $B \ll q/2$ and $m \gg n$ to ensure uniqueness of the solution with overwhelming probability.

Definition 2.2.2 (Decisional LWE (DLWE))**.** Given parameters (m, n, q, B), let (A, b) be generated as in the LWE problem. The *Decisional LWE* problem is to distinguish with non-negligible advantage, between:

1. (A, b) where $b = As + e \pmod q$ with $s \xleftarrow{\$} \mathbb{Z}_q^n$ and $e \xleftarrow{\$} [-B, B]^m$,
2. (A, r) where $r \xleftarrow{\$} \mathbb{Z}_q^m$ is chosen uniformly at random.

It has been proven that search-LWE and decisional-LWE are computationally equivalent up to polynomial-time reductions for appropriate parameter regimes [15]. Therefore, DLWE is a common hardness assumption in cryptographic security proofs.

Definition 2.2.3 (Short-Secret LWE (ss-LWE))**.** In the *short-secret* LWE problem, the secret vector s is not chosen uniformly in $\mathbb{Z}_q^n$, but from a distribution of *small-norm* elements. Let $\mathcal{S}$ be a distribution over $\mathbb{Z}^n$ with support contained in $\{-B_s, \dots, B_s\}^n$ where $B_s \ll q/2$. Then:

- $s \xleftarrow{\$} \mathcal{S}$,
- $A \xleftarrow{\$} \mathbb{Z}_q^{m \times n}$,
- $e \xleftarrow{\$} [-B, B]^m$,

and $b = As + e \pmod q$ as before. The *search-ss-LWE* problem is: given (A, b), recover s. It has also been proven that ss-LWE is computationally equivalent to the standard LWE problem [16].

Other restricted-secret variants

Short secrets may be drawn from other constrained distributions in practice to improve performance and reduce key sizes:

- Binary-secret LWE: $s \in \{0, 1\}^n$
- Ternary-secret LWE: $s \in \{-1, 0, 1\}^n$
- CBD-secret LWE: s is sampled from a *centered binomial distribution* (CBD), producing small, approximately Gaussian-like coefficients.

These variants often preserve LWE's assumed hardness for appropriate parameters and are widely used in practical lattice-based cryptosystems.

2.3 Lattices

Lattices are discrete subgroups of $\mathbb{R}^n$ with rich algebraic and geometric structure, and they form the mathematical foundation underlying the hardness of LWE.

Definition 2.3.1 (Lattice)**.** Let $B = \{v_1, v_2, \dots, v_m\} \subset \mathbb{R}^n$ be a set of $m \leq n$ linearly independent vectors. The *lattice* generated by B is:

$$L(B) = \left\{ \sum_{i=1}^{m} x_i v_i \ : \ x_i \in \mathbb{Z} \right\}.$$

The set B is called a *basis* of $L(B)$, the rank of $L(B)$ is m, and if $m = n$ the lattice is called *full-rank.*

The *volume* of a lattice $L(B)$, also called its *determinant*, is defined as

$$\mathrm{vol}(L) = \sqrt{\det(B^{\mathsf{T}} B)}.$$

If L is *full-rank*, this simplifies to $\mathrm{vol}(L) = |\det(B)|$. The volume is an invariant of the lattice: it does not depend on the choice of basis. Intuitively, it measures the "density" of the lattice points in $\mathbb{R}^n$: a larger volume corresponds to a sparser lattice, while a smaller volume indicates that the lattice points are more densely packed.

2.3.1 Lattice Problems

Beyond their geometric interest, many computational problems on lattices are central to the study of their algorithmic complexity and play a role in the foundations of lattice-based cryptography:

Definition 2.3.2 (Shortest Vector Problem (SVP))**.** Let $L \subset \mathbb{R}^n$ be a lattice of rank n given by a basis B. The *SVP* asks to find a nonzero vector $v \in L$ such that

$$\|v\| = \lambda_1(L) := \min\{\|x\| \ : \ x \in L \setminus \{\mathbf{0}\}\},$$

i.e., the nonzero lattice vector of minimal Euclidean length. Similarly, the γ-approximate SVP ($\gamma \geq 1$) asks to find a nonzero vector $v \in L$ satisfying $\|v\| \leq \gamma \cdot \lambda_1(L)$.

A natural promise version of SVP is the *Unique Shortest Vector Problem* (uSVP), where the shortest vector is guaranteed to be well separated from all others.

Definition 2.3.3 (Unique Shortest Vector Problem (uSVP))**.** Let $L \subset \mathbb{R}^n$ be a lattice of rank n given by a basis B. For a parameter $\gamma > 1$, the γ-*uSVP* asks, given B, to find a shortest nonzero vector $v \in L$ under the promise that

$$\lambda_2(L) \geq \gamma \cdot \lambda_1(L),$$

where $\lambda_1(L)$ and $\lambda_2(L)$ are the first and second successive minima of L, respectively. That is, the shortest nonzero vector is unique up to sign and is at least a factor γ shorter than all other linearly independent vectors in L.

While SVP and uSVP focus on finding short vectors, the *Closest Vector Problem* (CVP) deals with proximity to an arbitrary target.

Definition 2.3.4 (Closest Vector Problem (CVP))**.** Let $L \subset \mathbb{R}^n$ be a lattice given by a basis B, and let $t \in \mathbb{R}^n$ be an arbitrary target vector. The *CVP* asks to find a lattice vector $v \in L$ such that

$$\|t - v\| = \mathrm{dist}(t, L) := \min\{\|t - x\| \ : \ x \in L\}.$$

The γ-approximate CVP ($\gamma \geq 1$) asks to find a lattice vector $v \in L$ satisfying $\|t - v\| \leq \gamma \cdot \mathrm{dist}(t, L)$.

A special case of CVP with a uniqueness guarantee is the *Bounded Distance Decoding* (BDD) problem, which is particularly relevant to LWE-based constructions.

Definition 2.3.5 (Bounded Distance Decoding (BDD))**.** Let $L \subset \mathbb{R}^n$ be a lattice and let $\alpha \in (0, \frac{1}{2})$ be a decoding radius parameter. Given a target vector $t \in \mathbb{R}^n$ such that there exists a *unique* lattice vector $v \in L$ satisfying

$$\|t - v\| < \alpha \cdot \lambda_1(L),$$

the BDD_α *problem* asks to find this unique lattice vector v.

Overall, these problems are believed to be computationally demanding even to approximate within small factors, both in the classical and quantum settings.

2.3.2 Lattice Reduction

A lattice reduction algorithm transforms a basis $B = \{b_1, \dots, b_n\}$ of a lattice $\mathcal{L} \subset \mathbb{R}^n$ into a basis $B' = \{b'_1, \dots, b'_n\}$ of relatively short and nearly orthogonal vectors. The goal is not necessarily to find the shortest vector, which is computationally hard, but to transform the basis into a form that is easier to work with.

Lenstra–Lenstra–Lovász (LLL) Reduction

The LLL algorithm, introduced by Lenstra, Lenstra, and Lovász [17], is a polynomial-time lattice reduction algorithm that iteratively applies a *size reduction* step and the *Lovász condition*. The size reduction ensures that each vector b_i is small in the direction of previous vectors:

$$b_i \leftarrow b_i - \sum_{j<i} \mu_{i,j} b_j, \quad \mu_{i,j} = \frac{\langle b_i, b_j^* \rangle}{\|b_j^*\|^2}.$$

Where b_j^* is the Gram-Schmidt orthogonalization of the basis. Instead, the Lovász condition

$$\delta \|b_{i-1}^*\|^2 \leq \|b_i^*\|^2 + \mu_{i,i-1} \|b_{i-1}^*\|^2, \quad 0.25 < \delta \leq 1,$$

controls the success of the LLL-reduction.

The output is a reduced basis with the guarantee that the first vector cannot be much larger than the shortest non-zero vector:

$$\|b'_1\| \leq (2/(\sqrt{4\delta - 1}))^{n-1} \cdot \lambda_1(\mathcal{L}).$$

While LLL does not solve SVP exactly, it provides an efficient approximation and forms the foundation for more advanced reduction algorithms, such as BKZ.

Block Korkine–Zolotarev (BKZ)

The BKZ algorithm [18] extends LLL by applying stronger reduction on *blocks* of consecutive basis vectors. Given a block size parameter β, BKZ repeatedly selects a β-dimensional sublattice, projects it onto the orthogonal complement of the preceding vectors, and applies a near-exact SVP solver to this block. The reduced block is then reinserted into the global basis, and LLL is used as a preprocessing and postprocessing step to maintain global size reduction. Increasing β improves the quality of the reduced basis, but incurs an exponential increase in runtime.

Heuristically, the quality of a BKZ-reduced basis is described by the *Root Hermite factor* δ_0, which for practical block sizes in the range $50 \leq \beta \leq 1000$ satisfies [19]:

$$\delta_0 \;\approx\; \left(\frac{\beta}{2\pi e}\,(\pi\beta)^{\frac{1}{\beta}}\right)^{\frac{1}{2(\beta-1)}} \tag{2.1}$$

This formula captures the trade-off between increased reduction quality (smaller δ_0) and exponential growth in running time.

Moreover, given a lattice of dimension d and volume $\mathrm{Vol}(\mathcal{L})$, the shortest vector length is expected to follow the Gaussian heuristic:

$$\|v_{\min}\| \approx \delta_0^d \cdot \mathrm{Vol}(\mathcal{L})^{1/d}. \tag{2.2}$$

BKZ 2.0

BKZ 2.0 [20] improves the original BKZ with various improvements:

- Early-abort: limit the number of tours to control runtime.
- Pruned enumeration: skip unpromising branches to speed up SVP searches.
- Improved local preprocessing: better reduction of local blocks before enumeration.
- Optimized initial radius: choose a smaller starting search radius to reduce enumeration effort.

These optimizations significantly improve the efficiency of high-quality lattice reduction without degrading the achieved root-Hermite factor. State-of-the-art implementations follow a running-time model of the form

$$T_{\mathrm{BKZ}}(\beta) \;\sim\; c\,d \cdot t_\beta, \tag{2.3}$$

with $c = 16$ as empirically calibrated in [21]. For the SVP oracle cost t_β, the best known estimates are

$$t_\beta^{\text{classical}} \approx 2^{0.292\,\beta+16.4}, \qquad t_\beta^{\text{quantum}} \approx 2^{0.265\,\beta+16.4}.$$

This exponential scaling places a premium on carefully tuning β: too small a block size leads to insufficient reduction, while too large a β results in impractical runtimes. Therefore, it is essential in any practical attack to select a block size schedule that balances this trade-off. In progressive BKZ strategies, initially proposed in [19] and further studied in later works [22, 23, 24, 25], the block size is increased gradually during the reduction process. This allows for early partial reductions using small β values, which in turn accelerate and stabilize subsequent higher-β phases.

Fast Lattice Reduction

The FLATTER algorithm [26] offers a high-performance alternative to traditional lattice reduction methods, like LLL and BKZ2.0. It achieves this by using an iterative compression technique that reduces the precision of the lattice basis during each recursive step, thereby accelerating the reduction process without compromising the quality of the reduced basis. This approach allows FLATTER to handle lattices of significantly higher dimensions and bit-lengths than previous algorithms, making it particularly effective for cryptanalytic applications involving large-scale lattices. The algorithm maintains approximation guarantees analogous to LLL, ensuring that the reduced basis remains within a constant factor of the shortest vector.

Empirical evaluations demonstrate that FLATTER outperforms existing implementations in terms of speed, especially for lattices with dimensions exceeding 1000 and entries with millions of bits.

2.4 Hardness of LWE

The hardness of the LWE problem is deeply correlated to the geometry of lattices. In particular, an LWE instance can be formulated as a special case of the *BDD* problem on a certain *LWE lattice*, which in turn can be reduced to the *SVP* via embedding techniques. This chain of reductions is fundamental for both security proofs and cryptanalytic strategies, as it places LWE within the established hardness framework of classical lattice problems.

From LWE to BDD

Define the *LWE lattice* as

$$L_A = \{y \in \mathbb{Z}^m : \exists z \in \mathbb{Z}^n \text{ s.t. } y \equiv Az \pmod q\}.$$

By construction, L_A is a full-rank q-ary lattice in $\mathbb{R}^m$. The noiseless vector $y = As \bmod q$ lies exactly in L_A, while the received vector b is a perturbation of y by the small error vector e. Thus, given (A, b), recovering s is equivalent to finding the unique lattice point $y \in L_A$ within Euclidean distance $\|e\|_2 \leq \sqrt{m}B$ from b. This is precisely an instance of the BDD_α problem, where

$$\alpha = \frac{\|e\|_2}{\lambda_1(L_A)} < \frac{1}{2}$$

ensures the uniqueness of the solution.

From BDD to SVP Via Embedding

The standard approach to solving BDD instances is to embed them into an SVP instance in one higher dimension, following Kannan's embedding technique. Given a lattice $L = L(B) \subset \mathbb{R}^m$, a target vector $b \in \mathbb{R}^m$ and a decoding radius $\alpha < \lambda_1(L)/2$, define the

$(m+1)$-dimensional lattice

$$L' = L(B') \quad \text{with} \quad B' = \begin{bmatrix} B & -b \\ 0 & \alpha \end{bmatrix}.$$

The vector:

$$\tilde{v} = \begin{bmatrix} y - b \\ \alpha \end{bmatrix} \in L'$$

(where y is the unique closest lattice point to b) has Euclidean norm at most $\sqrt{2}\,\alpha$. Moreover, it is the shortest non-zero vector in L' up to sign. Therefore, solving SVP exactly in L' recovers $\tilde{v}$, from which y and hence s can be extracted. This reduction is exact for $\alpha < \lambda_1(L)/2$ and underlies *primal lattice attacks* on LWE.

Implications for Hardness

This reduction sequence,

$$\text{LWE} \quad \longrightarrow \quad \text{BDD}_\alpha \quad \longrightarrow \quad \text{SVP},$$

is essential for two complementary reasons: from a cryptographic perspective, it situates LWE in the complexity hierarchy of lattice problems, solving LWE is at least as hard as solving certain instances of SVP, for which no polynomial-time algorithms are known; from an algorithmic perspective, it enables concrete attacks by constructing L_A, applying Kannan's embedding, and invoking advanced SVP solvers, one can estimate the concrete security of LWE-based schemes.

2.5 LWE Structural Variants

2.5.1 Ring-LWE

The *Ring-Learning With Errors* (Ring-LWE) problem generalizes the classical LWE problem from vectors over $\mathbb{Z}_q$ to elements in a polynomial ring modulo q. This implies more compact key sizes and faster computations due to the algebraic structure of the ring.

Definition 2.5.1 (Ring-LWE (RLWE) [27])**.** Let n be a power of two and let $R_q = \mathbb{Z}_q[x]/(x^n + 1)$ be the n-th cyclotomic integer ring. Fix an error bound $B \ll q/2$ and a probability distribution χ supported on *small* polynomials in R_q. Let:

- $s \xleftarrow{\$} R_q$ be a *secret polynomial* chosen uniformly at random,
- $a \xleftarrow{\$} R_q$ be a uniformly random *public polynomial*,
- $e \xleftarrow{\$} \chi$ be an *error polynomial* with small coefficients drawn from χ,

we define $b = a \cdot s + e \pmod{q}$. The *search Ring-LWE* problem is: given samples $(a, b) \in R_q \times R_q$, recover s.

From Ring-LWE to LWE

Writing each polynomial in R_q in coefficient form as a vector in $\mathbb{Z}_q^n$, multiplication is *negacyclic*: the reduction relation $x^n \equiv -1 \pmod{x^n+1}$ causes the coefficients to wrap around with a sign inversion. Let

$$a(x) = a_0 + a_1 x + \cdots + a_{n-1}x^{n-1} \quad \longleftrightarrow \quad a = (a_0, a_1, \ldots, a_{n-1})^\mathsf{T} \in \mathbb{Z}_q^n,$$

then multiplication by a in R_q corresponds to multiplication by the *negacyclic* (anti-circulant) matrix

$$\overline{\mathrm{circ}}(a) = \begin{bmatrix} a_0 & -a_{n-1} & -a_{n-2} & \cdots & -a_1 \\ a_1 & a_0 & -a_{n-1} & \cdots & -a_2 \\ a_2 & a_1 & a_0 & \cdots & -a_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n-1} & a_{n-2} & a_{n-3} & \cdots & a_0 \end{bmatrix} \in \mathbb{Z}_q^{n\times n},$$

whose rows are cyclic shifts of a with the wrapped entries negated. Under the coefficient embedding, a Ring-LWE sample $b(x) = a(x)s(x) + e(x) \bmod q$ is thus equivalent to

$$b = \overline{\mathrm{circ}}(a) \cdot s + e \pmod{q}, \quad b, s, e \in \mathbb{Z}_q^n,$$

which is an LWE instance in $\mathbb{Z}_q^n$ with a highly structured public matrix.

2.5.2 Module-LWE

The *Module-Learning With Errors* (Module-LWE) problem generalizes both LWE and Ring-LWE by working over modules of rank ℓ over the polynomial ring R_q. It can be seen as replacing the polynomials in Ring-LWE with *vectors* of polynomials in R_q, reducing the algebraic structure compared to Ring-LWE, allowing flexible trade-offs between efficiency and security.

Definition 2.5.2 (Module-LWE (MLWE) [28])**.** Let n be a power of two, $R_q = \mathbb{Z}_q[x]/(x^n+1)$, and $\ell, k \in \mathbb{N}$ with $k \geq \ell$. Fix an error bound $B \ll q/2$. Let:

- $s \xleftarrow{\$} R_q^\ell$ be a *secret vector* of ℓ polynomials chosen uniformly at random,
- $a_1, a_2, \ldots, a_k \xleftarrow{\$} R_q^\ell$ be k uniformly random *public vectors* of ℓ polynomials,
- $e \xleftarrow{\$} S_B^k$ be an *error vector* of k polynomials whose coefficients lie in $[-B, B] \subset \mathbb{Z}$.

For each $i \in \{1, \ldots, k\}$, compute

$$b_i = a_i^T s + e_i \in R_q.$$

The *search-Module-LWE problem* is, given $(a_1, \ldots, a_k, b_1, \ldots, b_k)$, to recover s.

When $k = \ell = 1$, the Module-LWE problem reduces exactly to a single instance of the Ring-LWE problem.

From Module-LWE to LWE

Writing each polynomial a_{ij} in coefficient form as an n-vector over $\mathbb{Z}_q$, multiplication by a_{ij} corresponds to multiplication by $\overline{\mathrm{circ}}(a_{ij}) \in \mathbb{Z}_q^{n\times n}$. Stacking these blocks yields a structured block-circulant matrix

$$A = \begin{bmatrix} \overline{\mathrm{circ}}(a_{11}) & \overline{\mathrm{circ}}(a_{12}) & \cdots & \overline{\mathrm{circ}}(a_{1\ell}) \\ \overline{\mathrm{circ}}(a_{21}) & \overline{\mathrm{circ}}(a_{22}) & \cdots & \overline{\mathrm{circ}}(a_{2\ell}) \\ \vdots & \vdots & \ddots & \vdots \\ \overline{\mathrm{circ}}(a_{k1}) & \overline{\mathrm{circ}}(a_{k2}) & \cdots & \overline{\mathrm{circ}}(a_{k\ell}) \end{bmatrix} \in \mathbb{Z}_q^{kn\times \ell n}.$$

Let $s \in \mathbb{Z}_q^{\ell n}$ and $e, b \in \mathbb{Z}_q^{kn}$ be the coefficient representations of the secrets, errors, and outputs. Then the MLWE equations become:

$$As + e \equiv b \pmod q.$$

Thus, MLWE is an LWE problem with a highly structured block-circulant matrix.

2.6 CRYSTALS-Kyber

CRYSTALS-Kyber is a quantum-safe Key Encapsulation Mechanism (KEM), standardized by NIST in FIPS 203 under the name ML-KEM (Module-Lattice-based KEM), because it is based on the hardness of the MLWE problem. The Kyber KEM is derived from the Kyber Public Key Encryption (Kyber-PKE) scheme through the Fujisaki–Okamoto transform, achieving chosen-ciphertext security from the underlying chosen-plaintext secure encryption.

As described in Table 1, Kyber comes in three standardized parameter sets, corresponding to different security categories (NIST Levels 1, 3, and 5). Each set specifies the dimension parameter k, the noise sampling parameters η_1 and η_2 for the secret and error polynomials during key generation and key encapsulation and the compression parameters d_u and d_v used for the two components of the ciphertext. These parameters balance security, bandwidth, and performance, with larger k values yielding higher security levels at the cost of increased computational and memory requirements.

Table 1: Default CRYSTALS-Kyber parameter sets and their associated NIST security.

Parameter Set	**k**	$\boldsymbol{\eta_1}$	$\boldsymbol{\eta_2}$	$\mathbf{d_u}$	$\mathbf{d_v}$	**Security**
ML-512	2	3	2	10	4	Level 1
ML-768	3	2	2	10	4	Level 3
ML-1024	4	2	2	11	5	Level 5

In what follows, we focus on the *key generation* procedure of Kyber, which constitutes the target for our later cryptanalytic analysis.

2.6.1 CRYSTALS-Kyber Key Generation

Kyber operates over the cyclotomic polynomial ring

$$R_{3329} = \mathbb{Z}_{3329}[x]/(x^{256} + 1).$$

The key generation algorithm for CRYSTALS-Kyber produces an encapsulation key ek (public) and a decapsulation key dk (private) as follows:

1. **Random seed sampling.** Sample two independent 256-bit uniform random strings:

   $$d \xleftarrow{\$} \{0,1\}^{256}, \quad z \xleftarrow{\$} \{0,1\}^{256}.$$

   The value z will be embedded into the private key for use in the Fujisaki-Okamoto transform.

2. **Seed derivation and matrix expansion.** Derive two 256-bit seeds from d with domain separation by the module rank k:

   $$(\rho, \sigma) \leftarrow G\big(d \,\|\, \text{byte}(k)\big),$$

   where $G = \mathsf{SHA3\text{-}512}$ and the 512-bit output is split into two halves. Use ρ to expand a public matrix deterministically

   $$A = (a_{ij}) \in R_q^{k \times k}, \qquad R_q = \mathbb{Z}_q[x]/(x^n + 1),$$

   by invoking an XOF based on SHAKE-128 on inputs (ρ, i, j) to pseudorandomly sample the coefficients of each a_{ij}. Implementations precompute this directly in the *Number Theoretic Transform* (NTT) domain, yielding $\hat{A}$.

3. **Secret and error sampling.** Using σ as the PRF key, sample

   $$\mathbf{s} = (s_1, \ldots, s_k)^T \in R_q^k, \qquad \mathbf{e} = (e_1, \ldots, e_k)^T \in R_q^k$$

   from the centered binomial distribution CBD_{η_1}. Concretely, a counter N is incremented per draw, and the PRF SHAKE-256 $(\sigma \,\|\, \text{byte}(N))$ produces bytes that are mapped by CBD to small coefficients in a bounded interval depending on η_1. Transform to the NTT domain:

   $$\hat{\mathbf{s}} \leftarrow \text{NTT}(\mathbf{s}), \qquad \hat{\mathbf{e}} \leftarrow \text{NTT}(\mathbf{e}).$$

4. **Public vector computation.** Compute in the NTT domain:

   $$\hat{\mathbf{b}} = \hat{A} \cdot \hat{\mathbf{s}} + \hat{\mathbf{e}} \pmod{q}.$$

   This corresponds to an MLWE instance $(A, \mathbf{b} = A\mathbf{s} + \mathbf{e})$ in coefficient form.

5. **Key encoding and output.** Encode $\hat{\mathbf{b}}$ using 12-bit coefficient packing and concatenate with ρ to form the public key:

$$\mathrm{ek} = \mathrm{Encode}_{12}(\hat{\mathbf{b}}) \,\|\, \rho.$$

Encode $\hat{\mathbf{s}}$ similarly to obtain the PKE secret key $\mathrm{sk}_{\mathrm{pke}} = \mathrm{Encode}_{12}(\hat{\mathbf{s}})$. The final KEM keys are:

$$\mathrm{ek} \quad \text{(public)}, \quad \mathrm{dk} = \mathrm{sk}_{\mathrm{pke}} \,\|\, \mathrm{ek} \,\|\, H(\mathrm{ek}) \,\|\, z \quad \text{(private)},$$

where H is SHA3-256.

Security Basis

Given $(A, \mathbf{b})$ where $\mathbf{b} = A\mathbf{s} + \mathbf{e}$ and both $\mathbf{s}$ and $\mathbf{e}$ are short, recovering $\mathbf{s}$ is exactly an instance of the *search* MLWE problem. This reduction is the foundation of Kyber's security, ensuring that unless MLWE can be solved efficiently, the secret key remains hidden even in the presence of the public key.

In our attack, we will target this key generation process, particularly the recovery of $\mathbf{s}$ from $(A, \mathbf{b})$, since this directly undermines the decapsulation procedure of Kyber-KEM.

2.6.2 Number Theoretic Transform

Polynomial multiplication in $R_q = \mathbb{Z}_q[x]/(x^n + 1)$ is one of the main computational bottlenecks in lattice-based schemes. A naive polynomial multiplication requires $\mathcal{O}(n^2)$ coefficient operations. The NTT is the finite field version of the Fast Fourier Transform (FFT) and reduces the complexity of polynomial multiplication to $\mathcal{O}(n \log n)$.

Kyber-NTT Basics

Kyber operates in R_{3329} with modulus $q = 3329$, where $q \equiv 1 \pmod{n}$ but $q \not\equiv 1 \pmod{2n}$. An n-th root of unity $\zeta \in \mathbb{Z}_q$ is chosen such that

$$\zeta^{n/2} \equiv -1 \pmod{q}, \quad \zeta^n \equiv 1,$$

which is $\zeta = 17$ for the Kyber parameter set.

The forward NTT of $a(x) = \sum_{i=0}^{n-1} a_i x^i$ is defined by

$$\hat{a}_j = \sum_{i=0}^{n-1} a_i \,\zeta^{ij} \pmod{q}, \quad 0 \le j < n,$$

which corresponds to evaluating $a(x)$ at $\zeta^0, \zeta^1, \ldots, \zeta^{n-1}$. The transform is implemented via radix-2 butterflies using precomputed powers of ζ, yielding $\mathcal{O}(n \log n)$ complexity.

Multiplication in the Kyber-NTT

Given $\hat{a}, \hat{b}$ in NTT form, coefficients are processed in pairs (a_0, a_1) and (b_0, b_1) via

$$(r_0, r_1) = (a_0 b_0 + \zeta\, a_1 b_1, \quad a_1 b_0 + a_0 b_1) \pmod q.$$

Quartets (a_0, a_1, a_2, a_3) and (b_0, b_1, b_2, b_3) are multiplied as

$$(r_0, r_1) = \mathrm{BaseMult}(a_0, a_1, b_0, b_1,\ \zeta), \quad (r_2, r_3) = \mathrm{BaseMult}(a_2, a_3, b_2, b_3,\ -\zeta),$$

where the sign alternation reflects the negacyclic structure.

Relation to Secret Recovery

In Kyber's multiplication, each output pair satisfies

$$(\hat{b}_0, \hat{b}_1) = (\hat{a}_0, \hat{a}_1) \circ (\hat{s}_0, \hat{s}_1) + (\hat{e}_0, \hat{e}_1) \pmod q,$$

where $\hat{a}_i$ are public, $\hat{s}_i$ are secret, and $\hat{e}_i$ are error terms. Hypothetically, any method capable of recovering $(\hat{s}_0, \hat{s}_1)$ from $((\hat{a}_0, \hat{a}_1), (b_0, b_1))$ could recover the entire secret vector for any Kyber parameter set.

2.7 Attacks on LWE

2.7.1 Primal (uSVP) Attack

As explained in Section 2.4, the LWE problem with $(A, b) \in \mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^m$ samples can be first reduced to a BDD problem. Consider the q-ary lattice generated by

$$B_0 = \begin{pmatrix} I_n & 0 \\ A & qI_m \end{pmatrix},$$

From the initial samples, we can then remove the modulo and write $\mathbf{b} = A\mathbf{s} + \mathbf{e} + q\mathbf{c}$ with $\mathbf{c} \in \mathbb{Z}^m$, so that the lattice contains the vector: $B_0 \begin{pmatrix} \mathbf{s} \\ \mathbf{c} \end{pmatrix} = \begin{pmatrix} \mathbf{s} \\ \mathbf{b} - \mathbf{e} \end{pmatrix}$. Solving the BDD problem in this lattice with target $\mathbf{t} = \begin{pmatrix} 0 \\ \mathbf{b} \end{pmatrix}$ recovers $(\mathbf{s}, \mathbf{c})$, hence the secret $\mathbf{s}$. Kannan's embedding transforms the BDD instance into a uSVP instance by adding one extra row and column:

$$B_1 = \begin{pmatrix} B_0 & \mathbf{t} \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} I_n & 0 & 0 \\ A & qI_m & \mathbf{b} \\ 0 & 0 & 1 \end{pmatrix}.$$

The lattice generated by B_1 contains the unique shortest vector:

$$\mathbf{v}_{\text{short}} = B_1 \begin{pmatrix} \mathbf{s} \\ \mathbf{c} \\ 1 \end{pmatrix} = \begin{pmatrix} \mathbf{s} \\ \mathbf{e} \\ 1 \end{pmatrix}.$$

If the gap between the shortest vector $\mathbf{v}_{\text{short}}$ and the second shortest vector is sufficiently large, lattice reduction algorithms such as BKZ 2.0 can recover $\mathbf{v}_{\text{short}}$, yielding the secret $\mathbf{s}$. The conditions for recovery are: (i) the secret $\mathbf{s}$ must be small relative to other lattice vectors, (ii) the error $\mathbf{e}$ must satisfy $\|\mathbf{e}\| < \frac{1}{2}\lambda_1(\mathcal{L}(B_0))$ and (iii) the shortest vector in the embedded lattice B_1 must be unique. Under these conditions, the uSVP instance derived from LWE via Kannan's embedding guarantees recovery of $\mathbf{s}$.

2.7.2 Dual-Hybrid Meet-in-the-Middle (MitM) Attack

The Dual-Hybrid MitM attack [29] targets Decision-LWE with sparse secrets by splitting the public matrix $A = [A_1 \mid A_2]$ and correspondingly $\mathbf{s} = (\mathbf{s}_1, \mathbf{s}_2)$. A scaled dual lattice on A_1 is constructed, and lattice reduction yields short vectors that essentially eliminate the contribution of $\mathbf{s}_1$ to the LWE samples, producing reduced instances depending only on $\mathbf{s}_2$. Repeating this process τ times generates reduced samples used to build a locality-sensitive hash table of candidate $\mathbf{s}_2$ values. The MitM step finds collisions between guessed and stored candidates, identifying possible partial secrets, which are then verified. The error bound B controls hash sensitivity, and parameters τ, ζ, and B are tuned to balance time, memory, and reduction quality. This attack does not directly recover the whole secret, but efficiently narrows the search space for sparse-secret LWE.

2.7.3 SALSA Attacks

SALSA

SALSA (*Secret-recovery Attacks on LWE via Seq2Seq with Attention*) [1] is the first end-to-end machine learning attack that targets LWE with small, sparse secrets by training a transformer to operate directly over LWE samples and then converting the trained model into a secret-recovery procedure. Given many samples $(\mathbf{a}_i, b_i = \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i \bmod q)$ that share the same secret $\mathbf{s}$, SALSA trains a seq2seq transformer to predict b from $\mathbf{a}$, thereby forcing the model to internalize modular linear structure in the presence of noise; the paper first demonstrates that transformers can learn modular arithmetic reliably, then uses this capability for cryptanalysis. After training, SALSA offers two recovery modes: (i) *direct recovery*, which queries the model on carefully chosen inputs so that its outputs reveal coordinates of $\mathbf{s}$; (ii) a *distinguisher-based recovery* mode, in which the trained model serves as an oracle for distinguishing LWE samples from uniformly random ones and this oracle is leveraged to recover the secret via the standard reduction from the search variant of LWE to its decision form; both modes include a verification step that tests candidate secrets by checking that residuals $b_i - \langle \mathbf{a}_i, \hat{\mathbf{s}} \rangle \bmod q$ have slight variance consistent with the noise. Practically, SALSA recovers *sparse binary* secrets in small-to-mid dimensions, but it is sample-hungry (millions of samples in the original experiments) and its effectiveness decreases as dimension or Hamming weight grows.

SALSA PICANTE

SALSA PICANTE [2] improves on the original SALSA by using lattice-based preprocessing for large-scale data amplification, enabling transformer-based attacks on higher-dimensional LWE instances. The core innovation is the preprocessing via an *error-penalized lattice embedding* reminiscent of the dual embedding: for an LWE matrix $A \in \mathbb{Z}_q^{m\times n}$, the algorithm constructs

$$\Lambda = \begin{bmatrix} \omega \cdot I_m & A \\ 0 & q \cdot I_n \end{bmatrix} \tag{2.4}$$

where $\omega \in \mathbb{Z}$ weights the contribution of error coordinates. Applying BKZ 2.0 to this block matrix yields a unimodular transformation $[R\ \ C] \in \mathbb{Z}^{(m+n)\times(m+n)}$, producing

$$[R\ \ C] \cdot \begin{bmatrix} \omega \cdot I_m & A \\ 0 & q \cdot I_n \end{bmatrix} = \begin{bmatrix} \omega \cdot R & RA + q \cdot C \end{bmatrix}.$$

This transformation defines a reduced LWE instance (RA, Rb) with transformed errors Re. The parameter ω explicitly mediates the trade-off between minimizing the LWE matrix coordinates and controlling error amplification, enabling a more substantial reduction in A without excessive noise growth.

In practice, PICANTE uses the linearity of LWE to amplify a small number of samples into a synthetic dataset. Starting with only $m = 4n$ real LWE pairs, the algorithm applies a resampling procedure to generate an exponentially large family of $n \times n$ matrices by drawing random subsets of rows. Subsampled matrices initially preserve the original noise distribution and BKZ reduction transforms them via $[R\ \ C]$, amplifying the error according to $\|R\|$. This process produces millions of reduced LWE samples, which are deduplicated and encoded as token sequences for transformer training.

The secret-recovery phase improves upon SALSA by introducing *cross-attention extraction*, which directly leverages the transformer's attention maps to read secret bits. After training on the multi-million example corpus, the model's attention layers highlight correlations between input tokens and the underlying secret. By systematically interpreting these attention weights, cross-attention extraction can infer individual secret bits with high confidence. This method complements SALSA's direct recovery and distinguisher approaches; combining all three mechanisms yields more accurate secret reconstruction than any single method alone.

As a result, PICANTE recovers sparse binary secrets in dimensions up to $n = 350$ with Hamming weights $h \approx n/10$, surpassing SALSA's previous practical limits of $n \leq 128$ and $h \leq 4$.

SALSA VERDE

SALSA VERDE [30] refines the PICANTE attack by arranging the lattice embedding, optimizing the BKZ preprocessing, and adapting the machine learning pipeline for broader secret distributions. The first change is the embedding:

$$\Lambda_i' = \begin{bmatrix} 0 & q \cdot I_n \\ \omega \cdot I_m & A \end{bmatrix}. \tag{2.5}$$

This embedding, while producing a reduced basis of the same form $[\omega \cdot R \;\; RA + q \cdot C]$, positions A in the lower-right block. By placing A in this block, the rearrangement changes the geometry of the embedding so that the rows containing A are less affected by the error-penalization scaling ω, allowing BKZ to achieve stronger size reduction on them, based on their results. At the same time, the upper block $q \cdot I_n$ is extremely sparse, which reduces the number of non-zero entries processed during size-reduction steps, improving floating-point efficiency.

In addition to this structural change, VERDE lowers the penalty parameter from $\omega = 15$ to $\omega = 10$, and incorporates several BKZ engineering optimizations: interleaved reduction, adaptive block size selection, and early stopping. All aimed at cutting down the preprocessing cost. The sample count per embedded matrix is also slightly reduced from n to $0.875n$ without noticeably affecting the attack's success rate.

On the machine learning side, VERDE avoids using a sequence-to-sequence architecture in favour of an encoder-only transformer equipped with rotary position embeddings, trying to capture the cyclic structure of modular arithmetic. In the recovery stage, VERDE drops both the cross-attention and direct recovery modes, relying solely on an improved distinguisher, now extended to a two-bit variant to handle also ternary and Gaussian distributions.

Finally, VERDE attributes many recovery failures for small q to excessive modular wrap-around in the LWE samples. Although the percentage of non-modular samples cannot be measured in practice without knowing s, experiments show that successful recovery is strongly correlated with an empirical threshold of about 67%.

SALSA FRESCA

SALSA FRESCA [31] refines the preprocessing pipeline of VERDE by combining the recent FLATTER lattice reduction algorithm with BKZ 2.0 in an interleaved approach, inserting a polishing [32] step after each iteration to improve basis quality at minimal additional cost. On the machine-learning side, FRESCA retains the encoder-only transformer architecture from VERDE, but replaces rotary embeddings with angular embeddings, which represent modular coordinates as points on the unit circle to capture the inherent periodicity of LWE samples better. Furthermore, the attack leverages pre-training on generic LWE-like instances before fine-tuning on the target distribution, substantially reducing the number of task-specific training steps required. These combined optimizations enable efficient recovery of sparse binary secrets in dimensions up to $n = 1024$, extending the reach of the SALSA family to larger parameter regimes.

2.7.4 The Cool and the Cruel Attack

Cool and the Cruel [33] is a statistical attack where the authors observed that after applying lattice reduction to subsampled LWE matrices, the columns of the reduced matrix RA exhibit sharply varying standard deviations: the first n_u columns (after called the *cruel* bits), retain near-uniform variance $\sigma_u \approx q/\sqrt{12}$. In contrast, the remaining *cool* columns have much smaller variance $\sigma_r \ll \sigma_u$. The attack splits recovery into two stages. In the first stage, the small set of cruel bits is brute-forced: for each candidate assignment, one computes residuals $x = \mathbf{a} \cdot s^* - b \pmod{q}$ over the reduced samples. If the cruel assignment is correct, the variance of x remains low; otherwise, it appears nearly uniform, allowing a clear statistical distinction.

Once the cruel bits are fixed, the cool bits are recovered greedily. For each cool coordinate k, two hypotheses are tested (bit $= 0$ or 1) by comparing the variance of residuals under each guess. The correct bit yields a lower variance, enabling linear-time reconstruction of all cool bits. This divide-and-conquer approach dramatically reduces the search complexity and memory requirement compared to full brute force, enabling efficient recovery for dimensions up to $n = 1024$ on moderate hardware [34].

2.8 Machine Learning

In this work, we focus on linear and robust linear models for secret recovery rather than transformer-based architectures employed in previous machine learning-based attacks [1, 2, 30, 31]. While transformer models can approximate modular arithmetic operations, they act as black-boxes: the learned weights do not directly correspond to secret components, and secret extraction requires both large amounts of training data and additional postprocessing. In contrast, linear models are inherently *white-box*. Once trained, their learned coefficients directly encode the secret vector $\mathbf{s}$, allowing immediate recovery without auxiliary algorithms. This transparency significantly reduces both computational and memory requirements and enables fast interleaved recovery during preprocessing. Moreover, linear regression is constructed to capture the underlying linear relationship

$$\mathbf{b} = A\mathbf{s} + \text{noise},$$

and robust variants allow us to treat modular wrap-around as statistical outliers.

We consider five linear models selected for their interpretability, robustness, and computational efficiency, each providing distinct losses to mitigate the influence of outliers:

2.8.1 Linear Regression

Standard linear regression estimates the parameter vector $\boldsymbol{\beta}$ by minimizing the mean squared error (MSE) over all training samples:

$$L(\mathbf{y}, \hat{\mathbf{y}}) = \sum_i (y_i - \hat{y}_i)^2,$$

where $\hat{y}_i = \mathbf{x}_i^\top \boldsymbol{\beta}$. This model assumes that errors are Gaussian, making it optimal for clean datasets. However, it is susceptible to outliers because the squared loss penalizes large deviations quadratically. In the context of Module-LWE, even a small fraction of wraparound-affected labels can dominate the loss function and bias the estimated secret, motivating the use of more robust alternatives.

2.8.2 Huber Regressor

The Huber Regressor [35] addresses the sensitivity of standard linear regression by combining quadratic and linear loss functions:

$$L(y, \hat{y}) = \begin{cases} \frac{1}{2}(y-\hat{y})^2 & \text{if } |y-\hat{y}| \leq \epsilon, \\ \epsilon|y-\hat{y}| - \frac{1}{2}\epsilon^2 & \text{otherwise.} \end{cases}$$

Residuals smaller than ϵ follow the squared loss, while larger deviations are linearly penalized, preventing extreme outliers from disproportionately affecting the model. The transition parameter ϵ is critical: too small, a value risks discarding valid samples as outliers, while too large, a value reduces robustness.

2.8.3 RANSAC Regressor

RANSAC (RANdom SAmple Consensus) [36] is an iterative method that estimates a model from random subsets of data, seeking the one that best fits the most extensive set of inliers. At each iteration, a small random subset is used to fit a provisional model, which is then evaluated against the entire dataset to identify samples whose residuals lie within a fixed tolerance. The model producing the largest consensus set is retained, and its parameters are optionally refined using all inliers. This strategy makes RANSAC exceptionally robust even when outliers corrupt a significant fraction of the training data. The trade-off lies in its higher computational cost compared to direct fitting, due to repeated random sampling and model refitting, particularly in high dimensions.

2.8.4 Tukey's Biweight Regressor

Tukey's Biweight regression [37] implements a bounded influence loss function:

$$L(y, \hat{y}) = \begin{cases} \frac{c^2}{6}\left[1 - \left(\frac{y-\hat{y}}{c}\right)^2\right]^3 & \text{for } |y-\hat{y}| \leq c, \\ \frac{c^2}{6} & \text{otherwise,} \end{cases}$$

where c is a threshold that controls the transition from quadratic to constant loss. While Huber down-weights large residuals linearly, Tukey's Biweight effectively ignores them entirely once they exceed c, granting extreme robustness in the presence of higher-magnitude outliers. This robustness makes it particularly effective for datasets where outliers can severely distort parameter estimation, such as in our case.

2.8.5 Theil–Sen Regressor

The Theil–Sen estimator [38] is a non-parametric approach that computes the slope of the line for all pairwise combinations of points and estimates the model as the median slope.

$$\hat{\beta} = \text{median}\left\{\frac{y_j - y_i}{x_j - x_i} \mid i < j\right\}.$$

The intercept is then estimated as the median of $y_i - \hat{\beta}x_i$. Using the median rather than the mean, Theil–Sen is highly robust to outliers in the response variable and resistant to heavy-tailed noise. It is particularly advantageous when errors may be significant, but rare, and it guarantees a 50% breakdown point. Its primary limitation is the computational cost, which scales quadratically with the number of points.

Chapter 3

Methods

Properties of the NTT inspired our initial approach to attacking the scheme. Since the NTT is a finite-field version of the *Discrete Fourier Transform* (DFT) and FFT, we hypothesized that certain transformations in the frequency domain could reveal structural weaknesses exploitable for key recovery. We therefore began by analysing the NTT directly and subsequently moved to the complex domain, treating integer coefficients as complex numbers to leverage geometric interpretations.

This reflection led us to design a Fourier attack (described in Section 3.1, in which the secret was embedded into the angular domain defined by the Fourier transformation. We paired this with an angular loss function intended to approximate the modulo wrapping effect inherent to the cryptosystem. In theory, this loss function would guide optimisation toward the true secret by aligning predicted and observed phases. However, in practice, the optimisation process became trapped in local minima, preventing secret recovery.

As defined in Section 3.2, the limitations of the Fourier attack suggested a different path: rather than working within the modular domain, we could attempt to recover the non-modular samples and extract the secret directly from them (a straightforward task for machine learning models, even under noisy conditions). Unfortunately, reconstructing pre-modular values from observed modular samples proved to be a fundamentally infeasible problem without further knowledge.

To address this, we developed a preprocessing step based on lattice reduction techniques, aiming to produce better approximations of the original, pre-modular samples. We then combined this with post-processing outlier detection algorithms (Section 3.3) to refine the estimates when the recovered secret was incorrect.

3.1 Fourier Attack

The design of our attack strategy is motivated by the observation that the key generation and encryption procedures rely on well-structured algebraic operations defined over the finite ring $\mathbb{Z}_q^n$ or its polynomial analogue $\mathbb{Z}_q[X]/(X^n \pm 1)$. In particular, the scheme repeatedly applies modular additions, scalar multiplications, and convolutions. If one could faithfully translate these operations into a continuous and differentiable space, then it would be possible to construct an analytic model of the cryptosystem suitable for optimisation via gradient descent.

The main advantage of such a translation is that the machine learning model no longer needs to learn the structure of modular arithmetic from data. Still, it can instead operate on a representation where the modulo operation is preserved analytically. This translation could reduce the number of parameters required, the complexity of the model, and the time needed for training. To achieve this, we tried to map integers modulo q into the complex unit circle via a Fourier-inspired embedding, thereby transforming modular operations into geometric operations on phases.

3.1.1 Fourier Embedding

We define the *Fourier embedding* of the ring $\mathbb{Z}_q$ into the complex unit circle as

$$\varphi_q : \mathbb{Z}_q \longrightarrow \mathbb{C}, \qquad \varphi_q(a) = e^{\frac{2\pi i}{q} a}.$$

Each residue class $a \pmod{q}$ is mapped to a q-th root of unity, i.e., a point on the unit circle in the complex plane. Figure 1 illustrates this embedding for the case $q = 17$, where the 17 possible integers are evenly distributed along the unit circle.

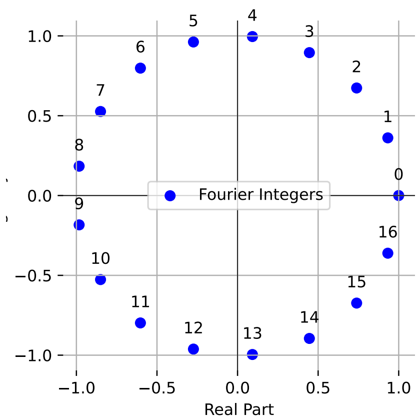


Figure 1: Embedding of integers modulo 17 into the complex unit circle.

Transformation of Addition

For two integers $a, b \in \mathbb{Z}_q$, we have

$$\varphi_q(a+b) = e^{\frac{2\pi i}{q}(a+b)} = \varphi_q(a)\,\varphi_q(b).$$

Thus, addition in the modular domain is transformed into multiplication in the complex domain. Geometrically, this corresponds to a rotation along the circle: adding b to a modulo q rotates the point $\varphi_q(a)$ by an angle of $\frac{2\pi}{q} b$ (see Figure 2).

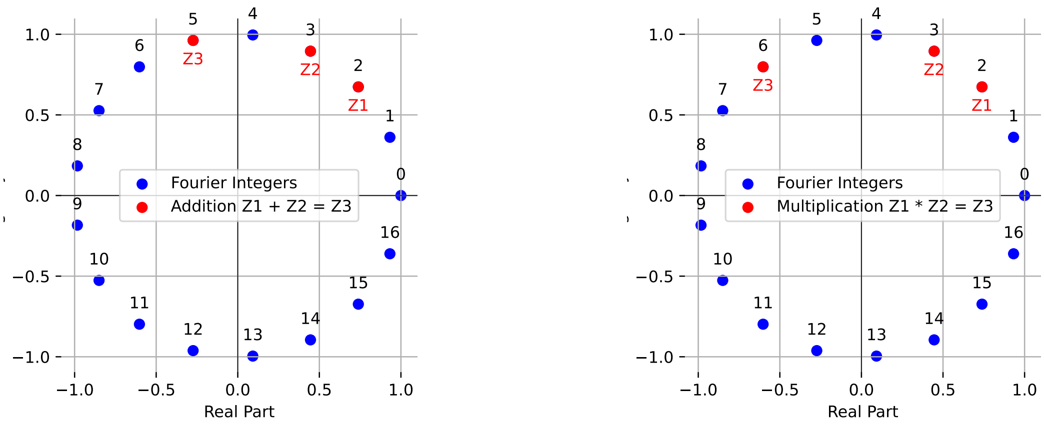


Figure 2: Operations in the complex fields modulo 17.

Transformation of Scalar Multiplication

For a scalar $c \in \mathbb{Z}$ and an element $a \in \mathbb{Z}_q$, we obtain

$$\varphi_q(ca) = e^{\frac{2\pi i}{q}(ca)} = \left(e^{\frac{2\pi i}{q}a}\right)^c = \varphi_q(a)^c.$$

Hence, multiplication by a scalar in the modular domain becomes exponentiation in the complex domain. The effect is a rotation of $\varphi_q(a)$ around the unit circle by an angle scaled by c (see Figure 2).

Vector Embedding and Product

For vectors $a = (a_0, \dots, a_{n-1}) \in \mathbb{Z}_q^n$, we embed component-wise:

$$\varphi_q(a) = \big(\varphi_q(a_0), \dots, \varphi_q(a_{n-1})\big) \in \mathbb{C}^n.$$

Given another vector $s = (s_0, \dots, s_{n-1}) \in \mathbb{Z}_q^n$, the inner product

$$\langle a, s\rangle = \sum_{j=0}^{n-1} a_j s_j \pmod q$$

in the integer domain transforms under the embedding into

$$\varphi_q(\langle a, s\rangle) = \exp\left(\frac{2\pi i}{q}\sum_{j=0}^{n-1} a_j s_j\right) = \prod_{j=0}^{n-1} \varphi_q(s_j)^{a_j}.$$

Under this embedding, the bilinear structure of the modular inner product corresponds to a single rotation on the complex unit circle, with each coefficient a_j contributing a rotation proportional to the associated component s_j. In other words, the modular inner product becomes a weighted sum of angles, producing a global phase in the complex plane.

Limitations

It is important to note that while modular addition and scalar multiplication are faithfully represented, linear combinations in the integer domain do not correspond to linear combinations in the complex domain. For instance, $\varphi_q(a) + \varphi_q(b)$ generally lies inside the convex hull of the roots of unity, not on the unit circle. Thus, naive addition of embedded values breaks the group structure. Figure 3 illustrates how direct complex addition leads outside the circle.

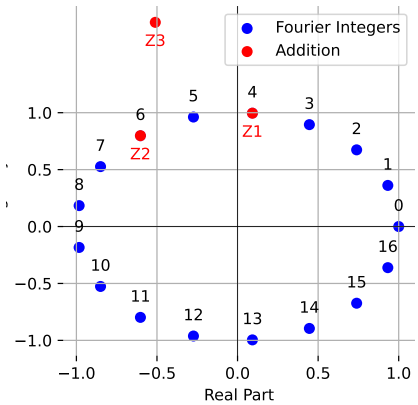


Figure 3: Direct addition of embedded values for modulo 17

This limitation underscores the necessity of carefully choosing which operations to emulate in the Fourier embedding. By restricting ourselves to modular addition and scalar multiplication that preserve the group structure, we ensure that the analytic model reflects the actual algebraic structure of the cryptosystem and can thus serve as the foundation for gradient-based key recovery.

3.1.2 Limitations of NTT and Failure of Fourier Attack

While the Fourier embedding provides a differentiable representation of modular arithmetic, our initial attempts to apply it directly in combination with the NTT quickly revealed a fundamental limitation. It has been observed in the literature [39] that the NTT domain has the effect of uniformising both the secret vector $\hat{s}$ and the error vector $\hat{e}$. In other words, once transformed by the NTT, these vectors no longer retain any exploitable structure but instead resemble uniformly distributed random variables over the field. As a consequence, one cannot hope to recover the secret pairwise from the NTT representation, since the transformed error obscures all useful statistical information.

This observation led us to abandon the idea of working directly in the NTT space and instead to rely solely on the differentiable complex embedding. The strategy was to construct a model whose parameters were the secret coefficients themselves. At each forward step, the model applied the Fourier embedding and the corresponding multiplications in the frequency domain between the parameter vector s and the public matrix A to predict b. Notably, the angular embedding allowed us to respect the topology of the modular field, where 0 and $q-1$ are close to each other on the circle. To train this model, we designed a custom *angular loss* function, measuring the geodesic distance between the predicted point and the actual point on the unit circle, together with a regularisation term enforcing the known distribution of the secret. Nevertheless, in practice, the model consistently failed to recover the correct secret. Instead, optimization became trapped in local minima.

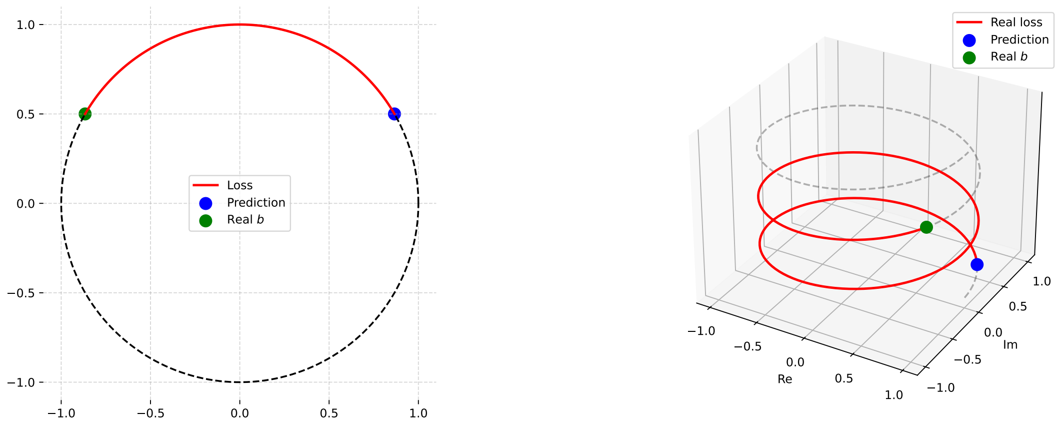


Figure 4: Visualization of losses in the complex domain. The left panel shows the angular loss in the unit circle, while the right panel depicts the true loss landscape in the unwrapped (spiral) domain.

The source of this failure can be understood as a discrepancy between the angular loss available to the learner and the actual loss determined by the underlying linear system before modular reduction. The angular loss measures distances along the arc of the unit circle (Figure 4), which effectively collapses all possible modular representatives into the same point on the circle. By contrast, the actual loss would have to be computed in the unwrapped domain, where each value of b corresponds not to a single point but to an entire coset of the form

$$b = As + e + kq, \qquad k \in \mathbb{Z}.$$

A useful geometric analogy is to view the real loss landscape as a three-dimensional spiral (Figure 4), where each non-modular b occupies a distinct height corresponding to the integer multiple kq. In contrast, the angular loss can only see the two-dimensional projection of this spiral onto the unit circle, thus losing the crucial vertical dimension that disambiguates between different modular representatives.

This insight made it clear that attempting to learn the secret within the modular operation was infeasible, even for machine learning methods, due to the irrecoverable information loss introduced by modular reduction. Although the angular embedding was helpful for correctly identifying the adjacency of 0 and $q-1$, it did not provide sufficient information to reconstruct the secret. At this point, we shifted our approach: instead of trying to learn from modularly reduced samples directly, we sought to infer the modular wrapping itself using statistical properties of A, s, and e. Once such an approximation of the unwrapped samples could be obtained, recovery of the secret became straightforward using simple linear regression, as described in the following section.

3.2 NoMod ML-Attack

The key idea behind our attack pipeline is to bypass the intrinsic difficulty of learning directly within the modular arithmetic domain. Rather than attempting to train complex models capable of implicitly approximating the modular reduction, we instead focus on recovering or reconstructing non-modular samples, i.e., pairs of the form

$$(A, b = A \cdot s + e),$$

where the modular wrapping term kq vanishes (i.e. $k = 0$). In this setting, the task of secret recovery becomes much simpler: since the error distribution e is Gaussian, estimating s reduces to a classical regression problem, solvable with lightweight machine learning models such as linear regression.

The main challenge lies in the fact that, in practice, many observed samples contain wraparounds where $k \neq 0$. These correspond to outliers from the perspective of regression, and if left untreated, they significantly degrade learning performance. Our approach, therefore, combines statistical modelling, lattice-based preprocessing, and efficient outlier handling to transform the observed modular samples into an approximation of the clean, non-modular distribution.

Concretely, in the NoMod ML-Attack, we proceed as follows. First, we apply preprocessing steps to reduce the effective contribution of modular wraps by exploiting the lattice structure of the data. This step serves to normalise the residual distribution, bringing it closer to the ideal case (one candidate). Next, we apply machine learning models to the preprocessed data, deliberately keeping these models simple to ensure both low training time and interpretability. Finally, we use outlier detection algorithms as a post-processing step to filter or correct remaining corrupted samples if the recovery fails in the training step.

By shifting the complexity of the attack away from the machine learning model and toward the data preprocessing stage, this framework allows us to recover the secret efficiently while avoiding the overhead associated with training deep or highly specialised architectures and the following costly secret recovery. In the following subsections, we detail the core components of the attack: the non-modular approximation, preprocessing strategies, and the final regression-based secret recovery.

3.2.1 Non-Modular Approximation

To characterise the distribution of a pre-modular LWE sample

$$\tilde{b} \;=\; As + e$$

it is natural and convenient to split the problem into two independent parts. We first approximate the linear contribution As and then we compute moments for the additive noise e. Under the standard assumption that the secret s and the error e are independent, the mean and variance of $\tilde{b}$ are the sum of the corresponding contributions:

$$\mu_{\tilde{b},i} = \mathbb{E}[\tilde{b}_i] = \mathbb{E}[(As)_i] + \mathbb{E}[e_i], \qquad \sigma^2_{\tilde{b},i} = \mathrm{Var}(\tilde{b}_i) = \mathrm{Var}((As)_i) + \mathrm{Var}(e_i). \tag{3.1}$$

Thus, the central task is to provide explicit expressions for $\mathbb{E}[(As)_i]$ and $\mathrm{Var}((As)_i)$ based on (i) the known row a_i of A and (ii) the distributional law of the coordinates of s. We denote by

$$a_i = (A_{i1}, \dots, A_{in}), \qquad S_1(i) = \sum_{j=1}^{n} A_{ij}, \qquad S_2(i) = \sum_{j=1}^{n} A_{ij}^2$$

the row statistics that will appear repeatedly below.

Distribution of As

We begin with the general identities:

$$\mathbb{E}[\tilde{b}_i] = \sum_{j=1}^{n} A_{ij}\, \mathbb{E}[s_j], \tag{3.2}$$

$$\mathrm{Var}(\tilde{b}_i) = \sum_{j=1}^{n} A_{ij}^2\, \mathrm{Var}(s_j) \; + \; 2 \sum_{1 \le j < \ell \le n} A_{ij} A_{i\ell}\, \mathrm{Cov}(s_j, s_\ell). \tag{3.3}$$

Equations (3.2)–(3.3) reduce the problem to two ingredients: the per-coordinate moments $\mathbb{E}[s_j]$ and $\mathrm{Var}(s_j)$, and any non-zero covariances $\mathrm{Cov}(s_j, s_\ell)$ which appear when coordinates are coupled (e.g., by fixing the Hamming weight). Below, we compute these quantities for the secret families of interest: Binary, Ternary, and CBD distributions.

1. Binary Secret: $s_j \in \{0, 1\}$. We distinguish two common sampling models:

a) Bernoulli (unconstrained): assume $s_j \overset{\text{i.i.d.}}{\sim} \mathrm{Bernoulli}(p)$. Then

$$\mathbb{E}[s_j] = p, \qquad \mathrm{Var}(s_j) = p \cdot (1 - p), \qquad \mathrm{Cov}(s_j, s_\ell) = 0 \;(j \neq \ell).$$

Substituting into (3.2)–(3.3) gives the closed form

$$\mathbb{E}[(As)_i] = p \cdot S_1(i), \qquad \mathrm{Var}((As)_i) = p \cdot (1 - p) \cdot S_2(i). \tag{3.4}$$

When no Hamming weight constraint is present, this baseline is used (the usual choice is $p = \frac{1}{2}$).

b) Exact Hamming weight h: suppose s is sampled uniformly from the set of binary vectors of length n with exactly h ones. In this model, coordinates are exchangeable but no longer independent. Elementary hypergeometric calculations yield

$$\mathbb{E}[s_j] = \frac{h}{n}, \qquad \mathrm{Var}(s_j) = \frac{h}{n}\Big(1 - \frac{h}{n}\Big), \tag{3.5}$$

$$\mathrm{Cov}(s_j, s_\ell) = \Pr[s_j = 1, s_\ell = 1] - \Big(\frac{h}{n}\Big)^2 = \frac{h(h-1)}{n(n-1)} - \Big(\frac{h}{n}\Big)^2 = -\frac{h(n-h)}{n^2(n-1)} \qquad (j \neq \ell). \tag{3.6}$$

Inserting (3.5)–(3.6) into (3.3) and using $\sum_{1 \le j < \ell \le n} A_{ij} A_{i\ell} = \frac{1}{2}\big(S_1(i)^2 - S_2(i)\big)$ produces the compact, exact variance formula

$$\mathbb{E}[(As)_i] \;=\; \frac{h}{n} \cdot S_1(i), \qquad \mathrm{Var}((As)_i) \;=\; \frac{h(n-h)}{n(n-1)} \cdot \Big(S_2(i) - \frac{S_1(i)^2}{n}\Big). \tag{3.7}$$

Equation (3.7) is exact for uniform sampling at fixed weight and is numerically stable: compute $S_1(i)$ and $S_2(i)$ per row and evaluate the prefactor $h(n-h)/(n(n-1))$. The model introduces negative pairwise covariance between coordinates, which reduces the variance of $\langle a_i, s\rangle$. The amount of reduction depends on the concentration of A (through $S_1(i)$ and $S_2(i)$).

2. Ternary Secret: $s_j \in \{-1, 0, 1\}$. As before, we define two common models:

a) Balanced ternary (symmetric): if $P(s_j = -1) = P(s_j = 0) = P(s_j = 1) = 1/3$ then

$$\mathbb{E}[s_j] = 0, \qquad \mathrm{Var}(s_j) = \mathbb{E}[s_j^2] = \frac{2}{3}, \qquad \mathrm{Cov}(s_j, s_\ell) = 0.$$

Hence

$$\mathbb{E}[(As)_i] = 0, \qquad \mathrm{Var}((As)_i) = \frac{2}{3}\, S_2(i). \tag{3.8}$$

b) Exact Hamming weight h: when exactly h coordinates are active and each active coordinate is assigned sign ± 1 independently and symmetrically, the per-coordinate mean remains zero and the per-coordinate variance equals the activity probability $p = h/n$. Cross-terms have zero expectation because the sign choices are independent and mean zero, so the covariance contribution vanishes. Thus:

$$\mathbb{E}[(As)_i] = 0, \qquad \mathrm{Var}((As)_i) = \frac{h}{n}\, S_2(i). \tag{3.9}$$

3. Centered Binomial Secret (CBD$_\eta$). A centered binomial with parameter η is generated by summing η independent $(+1, 0, -1)$ contributions:

$$s_j = \sum_{t=1}^{\eta} (u_t - v_t), \qquad u_t, v_t \overset{\text{i.i.d.}}{\sim} \text{Bernoulli}(1/2).$$

We again define the two common models:

a) CBD (unconstrained): this yields a symmetric law with

$$\mathbb{E}[s_j] = 0, \qquad \mathrm{Var}(s_j) = \frac{\eta}{2},$$

so that for i.i.d. CBD coordinates

$$\mathbb{E}[(As)_i] = 0, \qquad \mathrm{Var}((As)_i) = \frac{\eta}{2}\, S_2(i). \tag{3.10}$$

b) Exact Hamming weight h: if one enforces that the final secret has exactly h non-zero coordinates, the per-coordinate variance is reduced by the expected retention probability α of a coordinate. The derivation of α is straightforward:

(1) Let $P_0 = \Pr[s_j = 0]$ for the raw CBD_η. Then the raw non-zero probability is $q = 1 - P_0$.

(2) Let M denote the number of non-zero coordinates among the other $n-1$ positions. Then $M \sim \mathrm{Binomial}(n-1, q)$ and

$$\Pr[M = m] = \binom{n-1}{m} q^m (1-q)^{\,n-1-m}.$$

(3) Conditioned on $M = m$, when truncating to exactly h non-zeros the current coordinate remains non-zero with probability

$$\begin{cases} 1, & m < h, \\ \dfrac{h}{m+1}, & m \geq h, \end{cases}$$

because ties are resolved uniformly among the currently non-zero positions.

(4) Averaging over M gives the retention probability

$$\alpha \;=\; \sum_{m=0}^{h-1} \binom{n-1}{m} q^m (1-q)^{n-1-m} \;+\; \sum_{m=h}^{n-1} \binom{n-1}{m} q^m (1-q)^{n-1-m} \cdot \frac{h}{m+1}. \tag{3.11}$$

Modelling the surviving coordinates as independent with retention probability α, the effective per-coordinate variance becomes $\alpha \cdot (\eta/2)$. Hence

$$\mathbb{E}[(As)_i] = 0, \qquad \mathrm{Var}((As)_i) = \left(\frac{\eta}{2}\,\alpha\right) S_2(i). \tag{3.12}$$

Error Distributions

We now summarise the standard models for the additive error e, both of which are used in real-world scenarios and in our experiments. We treat only the moment calculations because the same combination rule (3.1) applies.

1. Discrete Gaussian Noise. If e_i is drawn i.i.d. from a (discrete) Gaussian with mean zero and standard deviation σ_e, then

$$\mathbb{E}[e_i] = 0, \qquad \mathrm{Var}(e_i) = \sigma_e^2.$$

Thus, the contribution of the error to the total moments is simply additive: for row i,

$$\mu_{\tilde{b},i} \leftarrow \mu_{(As),i} + 0, \qquad \sigma^2_{\tilde{b},i} \leftarrow \sigma^2_{(As),i} + \sigma_e^2.$$

2. CBD Noise. If each e_i is drawn i.i.d. from CBD_η, then (as for the secret CBD)

$$\mathbb{E}[e_i] = 0, \qquad \mathrm{Var}(e_i) = \frac{\eta}{2}.$$

Again, the error contribution is additive and homogeneous across rows:

$$\sigma^2_{\tilde{b},i} \;=\; \sigma^2_{(As),i} + \frac{\eta}{2}.$$

The expressions above provide closed formulas for the first two moments of $\tilde{b}_i$ for the secret and error laws used in LWE, including exact treatment of binary fixed-weight covariance and the actual retention factor α for CBD truncation. These moments are the only quantities required to implement the likelihood-ranking used in the selection of candidate unwrapped values.

3.2.2 Candidate Generation and Likelihood Ranking

Given a public (reduced) observation $b_i \in \mathbb{Z}_q$, the set of possible integer pre-images is

$$\mathcal{C}(b_i) \;=\; \{\, b_i + kq \;:\; k \in \mathbb{Z} \,\}.$$

Using the Gaussian approximation $\tilde{b}_i \overset{\text{approx}}{\sim} \mathcal{N}(\mu_{\tilde{b},i}, \sigma^2_{\tilde{b},i})$, the log-likelihood of each candidate $\tilde{b}_k = b_i + kq$ is proportional to

$$\ell_i(k) \;=\; -\frac{(\tilde{b}_k - \mu_{\tilde{b},i})^2}{2\sigma^2_{\tilde{b},i}}.$$

In practice, we restrict attention to the finite set of k satisfying

$$|\tilde{b}_k - \mu_{\tilde{b},i}| \le t\,\sigma_{\tilde{b},i},$$

for a chosen t (a standard choice is $t = 3$ to capture $\approx 99.7\%$ of Gaussian mass). The candidate set size for row i is therefore upper-bounded by

$$N_{\mathrm{cand},i}(t) \;\approx\; \left\lceil \frac{2t\,\sigma_{\tilde{b},i}}{q} \right\rceil, \tag{3.13}$$

because the integer shifts are spaced by q. Equation (3.13) is intuitive and effective for budgeting: narrow distributions (small $\sigma_{\tilde{b},i}$) yield one or a few candidates, wide distributions produce many.

We then obtain the candidate probabilities by normalizing the Gaussian densities over the finite candidate set:

$$p_i(k) \;=\; \frac{\exp\!\big(-\frac{(\tilde{b}_k-\mu_{\tilde{b},i})^2}{2\sigma_{\tilde{b},i}^2}\big)}{\sum_{k'\in\mathcal{C}_t(b_i)}\exp\!\big(-\frac{(\tilde{b}_{k'}-\mu_{\tilde{b},i})^2}{2\sigma_{\tilde{b},i}^2}\big)}, \tag{3.14}$$

where $\mathcal{C}_t(b_i)$ denotes the candidates inside the t-sigma window. Algorithmically, we generate $\mathcal{C}_t(b_i)$ for each row, compute $p_i(k)$, and either (i) retain the maximum likelihood candidate $\tilde{b}_i^* = \arg\max_k p_i(k)$, or (ii) propagate the entire candidate set (with probabilities) into subsequent stages (e.g. outlier detection, EM-style refinement or an ML model that handles soft labels).

3.2.3 Probability of Correct Zero-Shift (Inlier Rate)

A natural single-number characterization of the quality of unreduced LWE samples is the probability that the correct pre-modular representative coincides with the zero-shift case ($k = 0$), i.e., that no wraparound occurs. Writing each unreduced coordinate as

$$\tilde{b}_i = \mu_{\tilde{b},i} + \varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, \sigma_{\tilde{b},i}^2),$$

the event of correct recovery corresponds exactly to $|\varepsilon| < q/2$. Hence, we write the inlier probability in integral form as

$$P_{\text{inlier},i} = \Pr\!\big(|\varepsilon| < \tfrac{q}{2}\big) = \int_{-q/2}^{q/2} \frac{1}{\sqrt{2\pi}\,\sigma_{\tilde{b},i}} \exp\!\Big(-\frac{\varepsilon^2}{2\sigma_{\tilde{b},i}^2}\Big)\, d\varepsilon.$$

Introducing the substitution $t = \frac{\varepsilon}{\sqrt{2}\,\sigma_{\tilde{b},i}}$, so that $d\varepsilon = \sqrt{2}\,\sigma_{\tilde{b},i}\,dt$ and the integration limits become $\pm\,\frac{q}{2\sqrt{2}\,\sigma_{\tilde{b},i}}$, we obtain

$$P_{\text{inlier},i} = \frac{2}{\sqrt{\pi}} \int_0^{\frac{q}{2\sqrt{2}\,\sigma_{\tilde{b},i}}} e^{-t^2}\,dt = \operatorname{erf}\!\Big(\frac{q}{2\sqrt{2}\,\sigma_{\tilde{b},i}}\Big), \tag{3.15}$$

where $\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2} dt$ denotes the Gaussian error function.

This closed form provides an operational estimate of the per-coordinate inlier rate, i.e., the probability that a given row is immediately recoverable by simple likelihood maximization. For a dataset of M independent rows, the expected number of inliers follows as

$$\mathbb{E}[\#\text{inliers}] \;=\; \sum_{i=1}^{M} P_{\text{inlier},i},$$

which, under the simplifying assumption of approximately identical variances, reduces to

$\mathbb{E}[\#\text{inliers}] \approx M \cdot P_{\text{inlier}}$. In practice, one computes the parameters $\mu_{\tilde{b},i}$ and $\sigma^2_{\tilde{b},i}$ via Equation (3.1) enumerates candidate pre-modular representatives within a t-sigma window, ranks them by Gaussian likelihood, and invokes the closed form above to predict the fraction of samples that are unwrapped correctly without further refinement.

3.2.4 Preprocessing

Before attempting sample recovery, we apply a preprocessing stage based on lattice reduction. The goal is to transform the LWE instance into one with smaller coefficient magnitudes and thereby reduce the effective variance of the unreduced right-hand side b. Since the success of the unwrapping step depends critically on the distribution of the transformed A matrix, this preprocessing is essential to bring the samples into a regime where likelihood-based recovery becomes feasible.

Concretely, we embed the LWE matrix $A \in \mathbb{Z}_q^{n\times n}$ into a higher-dimensional lattice basis, apply lattice reduction, and obtain a unimodular transformation matrix $[R \quad C]$. This matrix yields a new instance (RA, Rb) with transformed error Re, whose variance depends directly on the quality of the reduction. Different embeddings of A govern how effectively reduction can shrink the norms of the rows of $RA + qC$, and therefore directly affect the trade-off between error amplification and variance reduction.

To further enhance the effectiveness of this preprocessing, we construct the matrices by sampling m rows out of a pool of $4n$ available samples. This randomized subsampling both amplifies the number of usable LWE instances and introduces variability into the embeddings, which can improve the chances of obtaining favorable reductions in subsequent steps.

Lattice Embeddings

We investigate three embeddings for preprocessing, each followed by lattice reduction. The first two correspond to the SALSA PICANTE and SALSA VERDE constructions introduced before in Section 2.7.3 and first experimented in [2, 30]. Both embeddings use a penalty parameter ω to balance two objectives: suppressing the growth of the error vector under the unimodular transformation, while still exposing the matrix A to strong reduction. By tuning ω, one can trade off error amplification against the strength of reduction on A, thereby controlling the variance of the transformed vector Rb.

In addition to these two established embeddings, we introduce a new custom embedding designed to exploit the empirical observation that reduction does not act uniformly across all columns of the lattice basis: earlier columns typically benefit from stronger norm reduction. To take advantage of this variance asymmetry, we place A directly in the upper-left block of the basis:

$$\Lambda_{\text{custom}} = \begin{bmatrix} A & \omega \cdot I_n \\ q \cdot I_n & 0 \end{bmatrix}. \tag{3.16}$$

Applying lattice reduction yields a reduced basis of the form

$$[R\ \ C] \cdot \begin{bmatrix} A & \omega \cdot I_n \\ q \cdot I_n & 0 \end{bmatrix} = \begin{bmatrix} RA + q \cdot C & \omega \cdot R \end{bmatrix}.$$

The motivation for this embedding stems from the empirical findings of [33], which show that after BKZ2.0 reduction, the variance of the columns in the transformed matrix is not uniform because specific columns exhibit significantly lower variance than others. By placing A in the upper-left block, this embedding ensures that the resulting transformed LWE matrix $RA + qC$ aligns with the early columns. This structure differs from the SALSA PICANTE and VERDE embeddings, where the transformed matrix $RA + qC$ typically appears in later columns of the basis and may thus benefit less from BKZ2.0's reduction pattern. Overall, this custom embedding explores an orthogonal direction to VERDE, leveraging variance asymmetry across columns.

From a geometric standpoint, in all three embeddings, the parameter ω also influences the norm of the shortest vector found. Once γ_0 is fixed, we expect the length of the shortest non-zero vector $v_{\min}$ in a lattice of dimension d and volume $\mathrm{Vol}(\mathcal{L})$ to satisfy the Gaussian-heuristic approximation described by Equation (2.2). Hence, our working estimate for the shortest vector norm becomes

$$\|v_{\min}\| \approx \gamma_0^{n+m} \left(q^m\, w^n\right)^{\frac{1}{n+m}} = \gamma_0^{n+m}\, q^{\frac{m}{n+m}}\, w^{\frac{n}{n+m}} \tag{3.17}$$

Minimizing this expression with respect to ω gives insight into the optimal choice of the penalty parameter. Taking the derivative of $\log \|v\|$ with respect to ω, we find:

$$\frac{\partial}{\partial \omega} \log \|v_{\min}\| = \frac{m}{n+m} \cdot \frac{1}{\omega},$$

implying that the shorter the target vector we want to find, the smaller ω should be. However, this comes at the cost of greater error amplification through R. Therefore, the choice of ω must carefully balance reduction quality against noise growth, and the optimal value depends on the specific attack strategy and target problem parameters. Empirically, a penalty of $\omega = 3$ can suffice in certain instances, but in most cases, a slightly higher value such as $\omega = 4$ tends to offer better overall performance for CBD errors. For Gaussian errors with $\sigma = 3$, instead, a much higher penalty $\omega = 10$ is required.

Effect of Reduction on the Distribution of $\tilde{b}$

Applying reduction replaces $\mathbf{b} = A\mathbf{s} + \mathbf{e}$ with

$$R\tilde{\mathbf{b}} \;=\; RA\mathbf{s} \;+\; R\mathbf{e},$$

where R is unimodular. In particular, we treat the deterministic part $RA\mathbf{s}$ analogously to $A\mathbf{s}$. Still, it has a key advantage: by construction, RA has shorter row norms than A, since lattice reduction algorithms are designed to produce such a basis. So, it turns out that the impact of R really focuses on the error term. If the error vector $\mathbf{e}$ has mean zero and i.i.d. entries of variance σ_e^2, then after reduction, each coordinate of $R\mathbf{e}$ is a linear combination of multiple independent errors. For the i-th row r_i of R we obtain

$$\mathrm{Var}((R\mathbf{e})_i) \;=\; \sigma_e^2 \,\|r_i\|_2^2.$$

Thus, the variance of each new error component grows proportionally to the squared Euclidean norm of the corresponding row of R: short and balanced rows keep the distribution tight. In contrast, long or unbalanced rows amplify the noise and increase the spread of $\tilde{\mathbf{b}}$.

Overall, the distribution of $\tilde{\mathbf{b}}$ reflects a trade-off: the deterministic term $RA\mathbf{s}$ benefits from the reduced norm of RA. In contrast, the error term $R\mathbf{e}$ is larger than the original $\mathbf{e}$ but still expected to remain short if R itself has short rows. In this sense, reduction decreases the contribution of the structured part $A\mathbf{s}$ while simultaneously inflating the noise, and the success of subsequent analysis depends on the balance between these two opposing effects.

In our embedding context, each short vector v decomposes as

$$\|v\| = \|wR \quad RA + qC\| \quad\Longrightarrow\quad \|v\|^2 = w^2 \cdot \|R\|^2 + \|RA + qC\|^2$$

Under the assumption that $RA+qC$ is centered mod-q (as it corresponds to the shortest representative in the equivalence class), we can also model the variance of the resulting non-modular transformed sample $R\tilde{b} = RAs + Re$ as:

$$\sigma^2_{R\tilde{b}_i} = \|(RA + qC)_i\|^2 \cdot \sigma_s^2 + \|R_i\|^2 \cdot \sigma_e^2 \qquad (3.18)$$

This expression highlights how both components contribute to the final deviation, which our learning model uses for inference. However, due to the non-uniform structure of reduced lattice bases, we cannot infer a precise analytic relationship between $\|R\|$ and $\|RA + qC\|$, since the Gram-Schmidt norms are irregular and unevenly distributed [33]. Consequently, we do not attempt to match the theoretical prediction of the average achievable standard deviation directly from the block size β.

Algorithms Used

In practice, we carried out the reduction of the embedding basis in two consecutive phases, which we found essential for both efficiency and stability. First, we applied a warm-up stage consisting of about ten iterations of the FLATTER algorithm [26], using a low compression parameter $\alpha = 0.001$. We observed that a small α preserves structural information while gradually improving orthogonality; if the algorithm stalls, the routine is occasionally stopped early, as further iterations cease to produce noticeable progress. After this initial stage, we reduce the basis using BKZ 2.0 [20] with $\delta = 0.99$ and a fixed progressive block size schedule. This transition is critical: preliminary experiments confirm that FLATTER alone cannot reach the quality required for decoding, while BKZ becomes far more effective after the warm-up phase. Between consecutive BKZ tours, we additionally call the polish routine, which consistently improves the basis without reversing the gains of previous tours.

To improve efficiency in practice, we further adapt the BKZ block size during the reduction process. We rely on the priority queue described below and update the block size whenever, for three consecutive tours, fewer than 10% of the reduced samples are saved in the queue. In such cases, we increment the block size to maintain progress. Moreover, once we reach a block size of 40, if the reduction stalls (defined as fewer than 2% of the samples being updated for three consecutive tours), we automatically increase the block size by 2. We deliberately use small increments at this stage: for block sizes ≥ 40, and especially beyond 50, the runtime per BKZ tour grows substantially. By adjusting the block size conservatively, we strike a balance between ensuring steady updates to the reduced samples and avoiding prohibitive runtimes per tour.

Finally, we stress that we avoid interleaving the two algorithms, as was done in SALSA VERDE and SALSA FRESCA: alternating back to FLATTER after BKZ caused a degradation of the reduced basis in our preliminary tests, undoing the progress achieved by BKZ.

Leveraging the Algebraic Structure of Ring-LWE

In the specific case of Ring-LWE (or Module-LWE with $k = 1$), we can exploit the algebraic structure of the underlying polynomial ring to enhance the efficiency of our lattice-based machine learning attack significantly. Traditionally, as in the SALSA VERDE [30] attack framework, samples are preprocessed by constructing composite matrices that mix multiple LWE samples before applying lattice reduction techniques. However, in the Ring-LWE setting, we observed that such mixing is not only unnecessary but also suboptimal when leveraging the ring structure. This observation is motivated by a similar idea applied in the context of the dual attack on MLWE [40], although in our case, we consider a different attack strategy. While it is unclear whether the same approach yields benefits for MLWE, it is well-suited to RLWE due to its algebraic structure.

Our optimised approach treats each Ring-LWE sample matrix independently, avoiding any form of replacement or combination. For each sample, we reduce its associated lattice using reduction algorithms, resulting in a matrix R whose rows correspond to short vectors such that the matrix $[w \cdot R \quad R \cdot A + q \cdot C]$ is also short. We conduct this reduction without exploiting cross-sample correlations, yet it allows us to use the functional structure due to the properties of the ring.

The key insight is that each short vector obtained from the reduction can be seen as an element of the ring $\mathbb{Z}_q[x]/(x^n + 1)$. By applying the ring's automorphism group (specifically, through cyclic rotation via the negative circulant structure), we can derive $\mathbf{n}$ distinct vectors per reduced vector that retain bounded norm and thus preserve their utility in constructing short vectors.

After preprocessing a single Ring-LWE sample, we obtain a matrix $R \in \mathbb{Z}^{t \times n}$ consisting of short vectors. Each row of R corresponds to a reduced vector in $\mathbb{Z}^n$, which we interpret as the coefficient embedding of a polynomial in the ring $R_q = \mathbb{Z}_q[x]/(x^n + 1)$:

$$R = \begin{bmatrix} r_0^{(0)} & r_1^{(0)} & r_2^{(0)} & \cdots & r_{n-1}^{(0)} \\ r_0^{(1)} & r_1^{(1)} & r_2^{(1)} & \cdots & r_{n-1}^{(1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_0^{(t-1)} & r_1^{(t-1)} & r_2^{(t-1)} & \cdots & r_{n-1}^{(t-1)} \end{bmatrix} \in \mathbb{Z}^{t\times n}$$

Let $\mathbf{r} = \mathbf{r}^{(0)} = [r_0, r_1, \ldots, r_{n-1}]$ be the first row of R, corresponding to a short polynomial $r(x) \in R_q$. We can construct its *transposed negative circulant* matrix $\overline{\mathrm{circ}}(\mathbf{r})^T \in \mathbb{Z}^{n\times n}$ as:

$$\overline{\mathrm{circ}}(\mathbf{r})^T = \begin{bmatrix} r_0 & r_1 & r_2 & \cdots & r_{n-1} \\ -r_{n-1} & r_0 & r_1 & \cdots & r_{n-2} \\ -r_{n-2} & -r_{n-1} & r_0 & \cdots & r_{n-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -r_1 & -r_2 & -r_3 & \cdots & r_0 \end{bmatrix}$$

Each of these $\overline{\mathrm{circ}}(\mathbf{r}^{(i)})$ can then be transformed into multiple LWE samples via multiplication with the original public matrices (A, b), effectively yielding n additional pseudo-samples per short vector $\mathbf{r}^{(i)}$. As a result, from g original Ring-LWE samples, we derive $g \cdot k \cdot n$ LWE samples for machine learning training while only performing lattice reduction on g matrices, instead of $g \cdot n$, as would be required in the naive approach.

This sample amplification technique drastically reduces the computational burden of preprocessing while increasing the diversity of training data.

Optimal Sample Size

In our attack framework, we employ BKZ2.0 to reduce the lattices arising from different embeddings. A central question is how the choice of the number of samples m interacts to determine the length of the shortest vector found.

To optimize the choice of m, we treat it as a real variable and differentiate the logarithm of our $\|v_{min}\|$ found in Equation (3.17). Setting

$$\log\|v\|(m) = (n+m)\log\gamma_0 + \frac{n\log q + m\log w}{n+m},$$

we compute

$$\frac{d\log\|v\|}{dm} = \log\gamma_0 + \frac{(n+m)\log w - (n\log q + m\log w)}{(n+m)^2} = \log\gamma_0 + \frac{n(\log w - \log q)}{(n+m)^2}.$$

Equating this derivative to zero yields

$$(n+m)^2 = \frac{n(\log q - \log w)}{\log\gamma_0} \quad\Longrightarrow\quad m^* = \sqrt{\frac{n(\log q - \log w)}{\log\gamma_0}} - n. \tag{3.19}$$

This closed-form guide indicates that a larger gap between $\log q$ and $\log w$ permits more samples, while poorer reduction quality (larger γ_0) forces m^* downwards. In practice, we round m^* to the nearest integer.

Number of Matrices

In our attack pipeline, we rely on randomly resampling subsets of the original LWE matrix to generate multiple reduced instances. However, due to the stochastic nature of this process, a critical question arises: how many resampled matrices are needed to ensure that all the original samples are likely to be utilised at least once? This problem is a generalization of the classical *coupon collector problem*, extended to the case where each matrix selects a subset of m elements (without replacement) from a pool of n total samples. To model this, we calculate the probability that each of the n elements is included in at least one of the k resampling processes. The inclusion-exclusion formula we use is:

$$\mathbb{P}(\text{all samples seen}) = \frac{1}{\binom{n}{m}^k} \sum_{j=0}^{n} (-1)^j \binom{n}{j} \binom{n-j}{m}^k,$$

which counts the total number of k-tuples of subsets of size m such that every element appears at least once. The alternating sign arises from inclusion-exclusion, where we iteratively subtract the number of configurations omitting j specific elements. This exact formula allows us to estimate the number of resampled matrices k required to achieve a desired coverage probability (e.g., at least 99% chance of having seen every sample) by using binary search over k. This exact approach ensures that the downstream attack pipeline efficiently leverages all available information. In practice, this formula only defines a minimum number of matrices to reduce to achieve good coverage, but a higher value would also be beneficial.

Vector Saving Strategy

To further reduce the average norm of the output vectors beyond what a standard BKZ 2.0 schedule can offer, we implement a modified reduction pipeline that retains and accumulates short vectors across multiple tours. Unlike standard lattice reduction on LWE, which overwrites the working basis at each tour and discards previously discovered vectors, our strategy selectively retains the shortest unique vectors seen throughout the entire reduction process. This approach shifts the focus from producing a fully reduced basis to generating a high-quality set of short vectors, suitable for statistical inference in our machine learning pipeline.

Concretely, after each BKZ 2.0 tour, we extract candidate short vectors from the current basis and evaluate them based on the resulting approximate $\sigma_{\tilde{b}}$ (that we will take as priority value). A bounded priority queue maintains the best vectors seen so far, with a fixed capacity to limit memory usage and computational overhead. In particular, the queue saves only the vectors with a priority lower than the current maximum in the queue, while it also checks and discards duplicates. Over multiple tours, this strategy produces a collection of diverse short vectors with significantly reduced average norm compared to any single tour of standard BKZ2.0. This process brings the distribution of the retained vectors closer to the theoretical shortest vector length, approximated by Equation 3.17.

While this strategy sacrifices the output being a reduced basis of the original embedded lattice (since many of the saved vectors are not mutually reduced or necessarily orthogonal), it aligns well with our application goal. Our attack does not require a basis, but only a collection of vectors with small norm and correct structure, allowing us to extract approximate LWE samples in the clear (i.e., without modular reduction). In this context, the breakdown of basis structure is a worthwhile trade-off for obtaining a tighter distribution on $R\tilde{b}$, which in turn boosts the effectiveness of our downstream machine learning attack.

3.2.5 Secret Recovery

After preprocessing, each reduced system of equations can be expressed in the form $(RA,\ R\mathbf{b} = RA \cdot \mathbf{s} + R\mathbf{e})$. From the initial $4n$ samples, we generate k different reduced matrices via subsampling, and from each matrix we extract up to t short vectors using lattice reduction (the maximum size of the priority queue in our reduction routine determines the bound t). This strategy produces a pool of $t \cdot k$ candidate samples after preprocessing.

Rather than using all of these samples indiscriminately, we introduce a ranking strategy based on the final estimated standard deviation $\sigma_{R\mathbf{b}}$. Intuitively, a smaller $\sigma_{R\mathbf{b}}$ corresponds to a dataset that is cleaner (fewer modular wraparounds and less variance induced by $R\mathbf{e}$), even if it is smaller in size. Since we are not able to identify outliers directly, we calculated in Section 3.2.3 that minimizing $\sigma_{R\mathbf{b}}$ increases the proportion of inliers and thus raises the probability of recovering the correct secret. For this reason, we first train the regression model on a suitably chosen subset, large enough to capture the linear pattern $R\mathbf{b} \approx RA \cdot \mathbf{s}$, but restricted enough to avoid the accumulation of outliers.

For the learning phase, we deliberately restrict ourselves to robust linear regression models. Linear models are a natural choice given the underlying algebraic structure of LWE, and their interpretability and efficiency enable us to perform recovery already during preprocessing. We tested multiple robust regressors (Huber, RANSAC, and Tukey's biweight regression) designed to mitigate the effect of corrupted samples by down-weighting them during training.

Once the regression model converges, the recovery procedure is straightforward: we extract the learned coefficient vector $\hat{\mathbf{s}}$, normalize it with respect to the expected secret distribution, and then round and clip it to enforce the known support of $\mathbf{s}$. We then verify the candidate secret by checking the residual distribution

$$\mathbf{r} = \mathbf{b} - A\hat{\mathbf{s}}.$$

If $\mathbf{r}$ is consistent with the known distribution of the original error vector $\mathbf{e}$ (i.e., bounded variance and support), then $\hat{\mathbf{s}}$ is accepted as the correct secret. Otherwise, the candidate is rejected, and training continues on additional subsets of samples until complete recovery.

This approach illustrates a fundamental trade-off: by prioritizing smaller-$\sigma_{R\mathbf{b}}$ datasets, we reduce the influence of modular artifacts and effectively increase the inlier ratio, which directly boosts the likelihood of successful recovery under robust regression.

3.3 Outlier Detection and Correction

Even after preprocessing and robust regression, the recovery step remains vulnerable to the presence of a large fraction of corrupted samples. To address this, we investigate whether we could exploit the learned weights to iteratively *re-estimate* the correct non-modular samples, thereby improving the effective inlier rate and enabling successful secret recovery in a subsequent retraining step. The core idea is to treat the regression model not only as a predictor of the secret, but also as a tool to evaluate the consistency of candidate samples through their gradients.

3.3.1 Gradient-Based Outlier Detection

We implemented a gradient-based scoring algorithm, similar to the one described in [41], where they use the sensitivity of a model's loss function to data perturbations to rank or filter observations. Intuitively, suppose a candidate sample aligns with the global trend learned by the model. In that case, the gradient of the loss with respect to the model parameters should be similar to the *average gradient direction.* Conversely, outliers induce gradients to deviate from this global direction. This deviation provides a natural scoring criterion for outlier detection: samples are ranked by the cosine similarity between their individual gradients and the full average gradient. High similarity indicates alignment with the underlying model structure (inlier), while low similarity signals inconsistency (outlier). In our setting, each preprocessed equation comes with a set of candidates $\{b_{i,1}, \dots, b_{i,m}\}$, together with associated probabilities $p_{i,j}$ reflecting their plausibility. During training, the model is initialized with the current best guesses $\{b_i^*\}$, and trained until convergence. After training, we re-evaluate the candidates using the following gradient-based scoring strategy:

1. Compute the *full gradient* ∇L of the loss function with respect to the guessed secret, using the current best candidates b_i^* weighted by their probabilities.
2. For each sample i and each candidate $b_{i,j}$, compute its *individual gradient* $\nabla L_{i,j}$ with respect to the model parameters.
3. Score each candidate as

   $$\text{score}(b_{i,j}) = p_{i,j} \cdot \cos\big(\nabla L_{i,j}, \nabla L\big),$$

   where cosine similarity measures alignment with the global gradient direction.
4. Update b_i^* by selecting the candidates $b_{i,j}$ with the highest score.

This procedure replaces inconsistent candidates with alternatives that better align with the global regression model, thereby iteratively refining the effective dataset $\{(A, b^*)\}$.

3.3.2 Unsupervised Gradient-Based Outlier Detection

While the gradient-cosine scoring method provides a mechanism for identifying and correcting corrupted samples, its effectiveness is lower in higher-dimensional settings (e.g., $n = 256$ or 512), due to the curse of dimensionality impacting cosine similarity. To overcome this limitation, we developed an *unsupervised learning framework* that leverages gradient information to detect clusters of samples corresponding to different modular shifts, and thereby tries to identify the actual non-modular samples directly.

Gradient Collection and Dimensionality Reduction

As in the previous approach, we compute the gradient of the regression loss with respect to the guessed secret for each sample, using the current best candidate labels. Formally, for each preprocessed equation (A_i, b_i^*), we compute

$$g_i = \nabla_{\mathbf{s}} L\big(f_\theta(A_i), b_i^*\big),$$

where f_θ is the trained model. These gradients $\{g_i\}_{i=1}^N$ encode how strongly each sample influences the current secret estimate. To reduce the dimensionality and extract the most relevant features for clustering, we apply principal component analysis (PCA), retaining a subset of principal components that explain the majority of variance in the gradient space. This step mitigates the high-dimensional noise that compromises the cosine-similarity approach.

Clustering and Candidate Reassignment

After dimensionality reduction, we perform unsupervised clustering on the gradients using K-means. The number of clusters is determined empirically, optionally guided by silhouette scores to select a configuration that best separates distinct gradient behaviours. Each cluster should then match a distinct modular shift k in the relationship $\tilde{b}_i = b_i + k_i \cdot q$. To resolve the cluster-to-k assignment, we construct a co-occurrence matrix comparing known c-values (modular offsets) to the cluster labels and apply the Hungarian algorithm to maximise correspondence. This algorithm yields an optimal one-to-one mapping between clusters and modular shifts, allowing the direct identification of candidates corresponding to $k = 0$. In practice, this is not possible, but since clusters are limited, we could retrain on all.

Once we manage to identify the $k = 0$ cluster, its corresponding samples should form a cleaner subset with a higher inlier rate. We retrain a robust linear regression model on this filtered dataset to refine the secret estimate. This step leverages the improved quality of the samples, effectively reducing the influence of wraparound-induced outliers.

3.3.3 Expectation-Maximization for Inlier Refinement

To further improve the identification of non-modular (inlier) samples within the post-processed dataset, we explored an Expectation-Maximization (EM) approach. The key idea behind this method is to iteratively refine a latent assignment of each sample to a modular shift k, effectively estimating the hidden variable that represents whether a given sample corresponds to the true linear relation or has been wrapped by some multiple of the modulus q.

During the *Expectation* step, we compute the most likely shift k_i for each sample b_i based on the current model's predictions, effectively aligning each observation closer to its unwrapped counterpart. This step produces an updated estimate of the actual underlying samples.

In the *Maximization* step, the regression model is retrained using these corrected samples, treating them as inliers. By alternating between estimating shifts and retraining the model, the EM framework should incrementally improve the likelihood of correctly identifying samples that follow the genuine linear relationship $\mathbf{b} = A\mathbf{s} + \mathbf{e}$, while reducing the influence of wraparound outliers. Theoretically, iterating this process over multiple epochs allows convergence toward a cleaner dataset, improving the effectiveness of subsequent secret recovery using robust regression models.

Chapter 4

Results and Discussion

We evaluate our lattice-reduction and machine learning attack on Module-LWE, including optimizations for Ring-LWE. Experiments are conducted on both synthetic and real-world parameters, with CRYSTALS-Kyber as the reference implementation. Our goal is to study how reduction, secret distributions, and learning algorithms affect recovery.

We start in Section 4.1 by analysing the effect of BKZ2.0 lattice reduction on the output distribution of $R\mathbf{b} = RA \cdot \mathbf{s} + R\mathbf{e} \bmod q$. Reduction significantly narrows the candidate set for pre-modular samples, improving the feasibility of subsequent recovery.

Next, in Section 4.2, we present results on Kyber-based MLWE instances, exploring how increasing dimension n and the number of modules k affect recovery. Moreover, Sections 4.3 and 4.4 compare our approach with other ML-based attacks: SALSA, SALSA PICANTE, and SALSA VERDE. We then benchmark against other state-of-the-art methods reported in the literature (Section 4.5): uSVP, SALSA FRESCA, the Cool and the Cruel (CC), and the MiTM approach. Across these comparisons, our method often achieves faster recovery and scales well to high-dimensional, dense secrets; however, in some complicated settings, it obtains slightly lower maximum recoverable Hamming weights, likely due to insufficient preprocessing time.

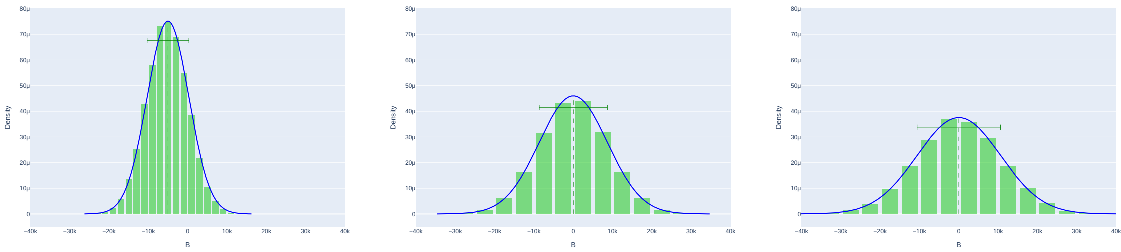
To quantify the contribution of individual designs, in Section 4.6, we present a systematic study of the settings. We examine preprocessing strategies, lattice embeddings, sample size optimisation, block-size schedules, and the choice of machine-learning algorithm, showing the trade-offs between runtime and recovery success.

Finally, in Section 4.7 we report on outlier-detection strategies, including gradient-based and EM-based methods. While these approaches showed some promise at low corruption rates, they gave no significant advantage over the baseline Tukey's Biweight regression in high-outlier regimes.

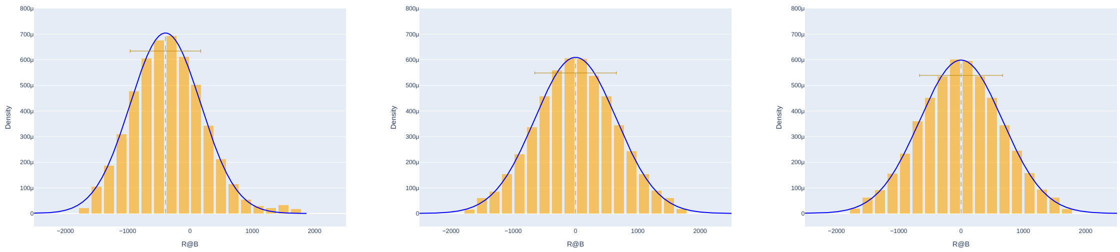
4.1 Distribution After Reduction

The output distribution of $\mathbf{b} = A \cdot \mathbf{s} + \mathbf{e} \bmod q$ plays a crucial role in determining how effectively we can recover pre-modular samples for secret recovery. The extent to which the reduction modulo q distorts the underlying distribution is influenced mainly by the modulus q, the dimension n, and, most importantly, the statistical properties of both the secret and error distributions. Conversely, specific parameter settings (e.g., a small modulus q) can cause severe aliasing effects due to modular reduction, which masks the sample structure and renders recovery infeasible.

In the case of structured variants, such as Module-LWE and Ring-LWE, these problems are typically treated analogously to standard LWE, with the modular reduction applied entry-wise and without regard for the underlying algebraic structure.



(a) Binary (b) Ternary (c) CBD



(d) Binary (Reduced) (e) Ternary (Reduced) (f) CBD (Reduced)

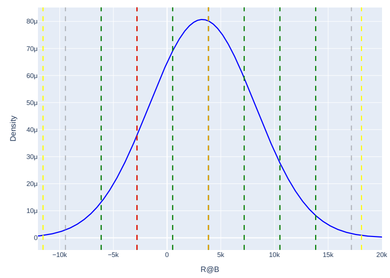
Figure 5: Comparison of unreduced and reduced distributions B across multiple secret distributions. In green, the unreduced samples (starting point) and in orange, the reduced samples. In blue, the approximation of the output distribution given A and the knowledge of the distribution of the secret and the error.

As we can see in Figure 5, the empirical distributions of the pre-reduction and post-reduction $\mathbf{b}$ samples vary significantly depending on the choice of secret distribution: binary, ternary, and centered binomial.

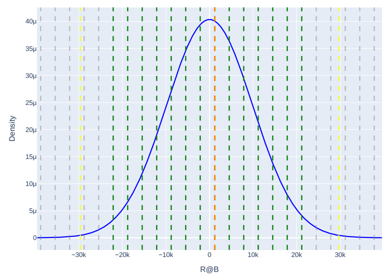
From a *statistical* perspective, the unreduced binary secret yields the narrowest distribution, consistent with its minimal variance. However, due to the inherent asymmetry of binary sampling (e.g., from $\{0, 1\}$), the distribution is noticeably biased away from zero. In contrast, both ternary and CBD secrets produce distributions that are symmetrically centered around zero. The CBD, derived from a discrete convolution of uniform variables, exhibits a higher standard deviation and heavier tails compared to the ternary case. These properties are reflected clearly in the broader and flatter shape of its curve.

Reduction has a profound effect on these distributions. Post-reduction, the range of $\tilde{\mathbf{b}}$ values is drastically compressed across all secret types, as seen in the second row of Figure 5. The narrowing of support directly improves the probability of accurately recovering the original (non-modular) $\tilde{\mathbf{b}}$ values, i.e., recovering $A{\cdot}\mathbf{s}{+}\mathbf{e} \in \mathbb{Z}^n$ before the modulo q operation. The tighter the reduced distribution, the higher the likelihood of identifying the correct pre-modulo image.

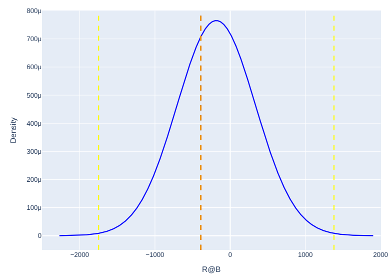
In the first row of Figure 6, we display two examples of unreduced samples for the binary and CBD secrets. Given the public value $b = \tilde{b} \bmod q$, we know that the pre-modular value satisfies $\tilde{b} = b + k \cdot q$, and we can compute the likelihood and respective probabilities using Equation 3.14. We can also approximate the number of final candidates using Equation 3.13; the more the candidates, the lower the probability of guessing the right one.



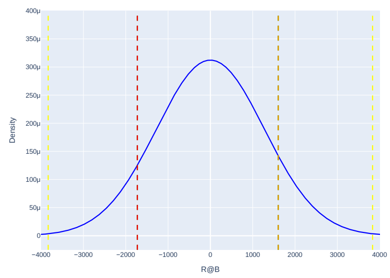
(a) Binary



(b) CBD



(c) Binary (Reduced)



(d) CDB (Reduced)

Figure 6: Comparison between possible non-modular samples for binary and centered binomial distributed secret before and after reduction.

In the unreduced high-variance regime, the Gaussian is broad, so numerous different k give similar likelihoods: for binary secrets, we already notice several candidates that seem plausible, and for CBD secrets, the flatness gives rise to even more. Following lattice reduction (we can see two examples in the second row of Figure 6), the effective standard deviation $\sigma_{R\tilde{b}}$ is considerably smaller. As a result, only one, or at most a few, of the shifts fall within the narrow $3\sigma_{R\tilde{b}}$ window, and the above maximization easily singles out the correct k. In the best case of perfect reduction, the number of high-likelihood candidates shrinks to one, leaving little work for a future machine learning model to separate inliers from outliers. In practice, reduction quality is variable and often leaves several candidates; the learning algorithm must then be able to discriminate between inlier and outlier samples or be highly robust to them.

To systematically assess the quality of the reduction, we first compute the ratio $\rho_A = \frac{\sigma_{RA}}{\sigma_A}$ as introduced in SALSA PICANTE [2], which quantifies the change in standard deviation before and after reduction of the uniform matrix. Nevertheless, since our goal is the reduction of $\tilde{\mathbf{b}}$, we introduce a more relevant estimate $\rho_b = \frac{\sigma_{R\tilde{b}}}{\sigma_{\tilde{b}}}$. This quantity is directly linked to the success probability of the secret recovery, as $\sigma_{R\tilde{b}}$ governs the inlier distribution. Empirically, we observe that the machine learning algorithm consistently succeeds when the inlier rate reaches approximately 65%. From the inlier rate expression given in Equation (3.15), and recalling that $\sigma_{R\tilde{b}} = \rho_b \cdot \sigma_{\tilde{b}}$, we can derive the threshold reduction factor required to guarantee recovery. Specifically, the critical value ρ_b' is obtained as

$$\rho_b' = \frac{q}{2\sqrt{2}\,\sigma_{\tilde{b}}\ \mathrm{erf}^{-1}\left(P^*_{\text{inlier}}\right)}$$

where q denotes the modulus and P^*_{inlier} is the target inlier probability ensuring recovery.

4.2 Results on CRYSTALS-Kyber Settings

This section presents our experimental results on secret recovery under CRYSTALS-Kyber parameter sets for centered binomial secrets. In particular, we targeted Kyber instances under dimensions ranging from 32 to 256 for the RLWE case ($k = 1$), and up to 128 for $k = 2, 3$. We have strictly followed the key generation procedure, as described in the literature, with $\eta_1 = 2$ for both the secret and error vectors, and modulus $q = 3329$, as in the latest published version.

Experiments on RLWE (MLWE with k = 1) were executed in parallel on 12 Intel(R) Xeon(R) CPU E5-2670 0 @ 2.60 GHz processors (from which only four where actually needed for parallel reduction), while MLWE (k > 1) experiments where performed in parallel on 48 AMD EPYC 7702P CPUs @ 1.50–2.18 GHz processors.

For the reduction strategy, we employed the FLATTER algorithm as an initial warm-up phase, followed by a fixed progressive BKZ schedule with block sizes ranging from 20 to 40 in increments of 10, with automatic updates thereafter, as described in Section 3.2.4. During the experiments, we observed that a penalty value of 4 gave the best trade-off: it sufficiently reduced the standard deviation of R to allow reliable secret recovery, while still enabling a significant reduction of RA and hence lowering the overall $\sigma_{R\tilde{b}}$. Notably, in lower-dimensional instances, the final BKZ block size was often not reached, as one or a few iterations of Flatter were already sufficient to achieve full secret recovery.

Tables 2 and 3 summarize the results of our experiments. For Hamming weights (hw) marked with an asterisk (*), we recovered the secret in its entirety without any specific maximum. For other cases, the algorithm succeeded in recovering only the sparse secret following the indicated Hamming weight.

In all these experiments, we can notice how increasing the dimension n has two main effects: it linearly increases the initial variance of $\tilde{b}$ samples, and it makes the lattice reduction problem inherently harder. From the vector norm estimation in Equation 3.17, we see that the length of the shortest recoverable vector depends exponentially on n and m, and polynomially on q and w. Likewise, the complexity of each BKZ tour increases significantly with dimension, as illustrated by Equation 2.3, highlighting that, even if

		NoMod ML-Attack			
n	k	max **hw**	$\mathbf{log_2}$ rop[1]	$\mathbf{log_2}$ samples	time
32	1	21*	33.1	11.26	3 s
40	1	25*	35.8	10.32	5 s
50	1	32*	39.2	10.97	12 s
64	1	44*	44.0	11.58	30 s
70	1	47*	38.9	11.94	34 s
80	1	52*	49.2	12.32	85 s
90	1	58*	52.8	12.66	2 m
100	1	75*	56.1	14.64	8 m
110	1	72*	40.2	14.78	30 m
120	1	77*	40.6	15.81	7 h
128	1	56	40.9	16.00	8.5 h
200	1	9	52.0	16.29	11.5 h
256	1	6	60.7	16.64	10.6 h

Table 2: Attack times in CRYSTAL-Kyber settings with centered binomial secrets over different dimensions n in the case of $k = 1$ (RLWE) for $q = 3329$.

		NoMod ML-Attack			
n	k	max **hw**	$\mathbf{log_2}$ rop[1]	$\mathbf{log_2}$ samples	time
16	2	21*	33.1	11.26	7 s
32	2	42*	44.0	11.30	23 s
40	2	53*	49.2	11.17	2 m
50	2	60*	56.1	12.46	2.3 h
64	2	22	61.9	13.20	5 h
128	2	4	59.1	13.78	5 h
256	2	4	101.9	14.94	44 h
16	3	32*	38.7	11.31	15 s
32	3	61*	54.7	12.30	18 m
40	3	33	60.8	13.20	5 h
50	3	9	41.2	13.20	5 h
64	3	6	49.2	13.20	5 h
128	3	3	78.9	14.20	5 h

Table 3: Attack complexities in CRYSTALS-Kyber settings with binary and CBD secrets over different dimensions n in the case of $k > 1$ (MLWE) for $q = 3329$.

the lattice dimension n contributes linearly to the runtime, high-dimensional reductions become increasingly expensive even for moderate block sizes.

Taken together, these theoretical considerations explain the observed exponential increase in runtime with dimension n and k. Nonetheless, our method achieves successful recovery up to $n = 128$ for centered binomial secrets within feasible timeframes, without requiring a pre-trained model or excessive computational resources.

4.3 Results on SALSA and SALSA PICANTE Settings

In this section, we evaluated our NoMod ML-attack against the two machine learning-based LWE attacks, SALSA [1] and SALSA PICANTE [2], which were the first to demonstrate secret recovery via ML. SALSA achieved proof-of-concept success on small dimensions ($n \leq 128$) and very low Hamming weights ($h \leq 4$) using transformer models trained on millions of LWE samples. However, it struggled at higher Hamming weights, a limitation inherent to its black-box architecture and the absence of preprocessing. PICANTE improved upon SALSA by introducing a preprocessing phase that reduced the data requirement to $4n$ samples and extended secret recovery up to $n = 350$ and $h = 60$, while also implementing a cross-attention secret recovery. Nevertheless, PICANTE's preprocessing becomes prohibitively slow as dimension and modulus grow, often making secret recovery impractical.

Our experiments were performed using dual embedding and executed in parallel on 64 AMD EPYC 7702P CPUs running at 2.00–2.18 GHz. Each attack uses a fixed penalty of 10 to account for the higher error magnitude drawn from a Gaussian distribution with standard deviation $\sigma = 3$, larger than the centered binomial error ($\eta = 2$) standard deviation used in the Kyber setting. Apart from the binary secret and this different noise distribution, we kept all other training parameters consistent: a warm-up phase with FLATTER followed by progressive BKZ2.0 reduction (block sizes from 20 to 40 in increments of 10 with automatic updates).

Runtime Comparison

n	hw	**SALSA**	**NoMod ML-attack**
30	3	$4.3 \cdot 10^3$	1.4
30	4	$46.4 \cdot 10^3$	1.4
32	3	$4.3 \cdot 10^3$	1.6
50	3	$16.9 \cdot 10^3$	3.5
50	4	$179.6 \cdot 10^3$	3.5
64	3	$28.8 \cdot 10^3$	4.4
70	3	$42.8 \cdot 10^3$	11.1
90	3	$156.2 \cdot 10^3$	6.9
110	3	$247.7 \cdot 10^3$	35.0
128	3	$165.6 \cdot 10^3$	45.4

Table 4: Runtime comparison in seconds between SALSA and NoMod ML-attack on dataset with $q = 251$ and variable dimension n with binary secret and gaussian error with $\sigma = 3$.

Tables 4 and 5 compare runtimes for SALSA, PICANTE, and our NoMod ML-attack. Across all tested instances, our attack is orders of magnitude faster. For example, at $n = 30$, $h = 4$, SALSA requires ≈ 1.2 hours compared to only 1.4 seconds for our method. At $n = 128$, $h = 3$, SALSA needs 46 hours, while we succeed in under one minute (45.4 seconds). This reduction factor of 10^3–$10^4\times$ is consistent across dimensions, reflecting both the efficiency of our preprocessing pipeline and the simplicity of regression-based recovery

n	q	hw	**SALSA picante**	**NoMod ML-attack**
80	113	9	42	1.2
150	6421	13	57	6.9
200	130769	22	87	19.1
256	6139999	31	139	20.1
300	94056013	33	205	1.9
350	3831165139	60	307	3.8

Table 5: Runtime comparison in hours between SALSA PICANTE and NoMod ML-Attack on dataset with variable dimension n and modulo q.

compared to transformer training.

Against PICANTE, the difference is similar. While PICANTE takes up to 307 hours at $n = 350$, our attack completes within 3.8 hours. Even at intermediate sizes ($n = 200$), our method terminates in under 20 hours, compared to nearly 90 hours for PICANTE. Thus, our approach remains practical at dimensions where lattice reduction and transformers' training dominate PICANTE's runtime.

Maximum Recoverable Hamming Weight

	SALSA				**NoMod ML-Attack**			
n	max **hw**	$\mathbf{log_2}$ rop[1]	$\mathbf{log_2}$ samples	time **(hours)**	max **hw**	$\mathbf{log_2}$ rop[1]	$\mathbf{log_2}$ samples	time
30	4	33.1	23.84	12.9	15*	33.1	12.06	11 s
32	3	33.1	20.93	1.2	17*	33.1	12.11	15 s
50	4	36.7	25.67	49.9	25*	39.5	11.92	42 s
64	3	38.0	22.39	8	32*	44.5	13.21	227 s
70	3	38.4	22.74	11.9	35*	46.5	13.21	51.1 m
90	3	39.5	23.93	43.4	19	47.2	13.62	24.1 h
110	3	44.1	24.07	68.8	10	50.3	13.76	24.1 h
128	3	48.0	22.25	46.0	8	53.6	13.94	24.1 h

Table 6: Recoverable success between SALSA[1] and NoMod ML-attack on dataset with $q = 251$ and variable dimension n with binary secret and gaussian error with $\sigma = 3$.

The more striking difference lies in the maximum Hamming weight recovered, shown in Tables 6 and 7. SALSA is limited to $h \leq 4$, whereas our method consistently recovers 3–4$\times$ higher Hamming weights under the same conditions. For example, at $n = 50$, SALSA recovers $h = 4$, while our method recovers $h = 25$. Even at higher dimensions, the advantage persists: at $n = 128$, our attack recovers $h = 8$, more than double SALSA's capability.

Against PICANTE, our method again shows significant gains. At $n = 200$, PICANTE succeeds up to $h = 22$, while we achieve recovery up to $h = 61$. Most notably, at $n = 350$, we not only surpass PICANTE's $h = 60$ limit, but also recover the whole secret with

[1]Real operations (rop) needed while performing a primal uSVP attack. Estimated using lattice-estimator `https://github.com/malb/lattice-estimator`

n	q	SALSA PICANTE				NoMod ML-Attack			
		max **hw**	**log$_2$** rop[1]	**log$_2$** samples	time **(hours)**	max **hw**	**log$_2$** rop[1]	**log$_2$** samples	time (**hours**)
80	113	9	46.8	22.41	42	13	48.9	13.62	12.5
150	6421	13	43.0	22.06	57	31	45.3	14.21	40
200	130769	22	41.7	22.04	87	61	43.3	14.94	40.1
256	6139999	31	41.6	22.02	139	104	41.8	15.21	40.3
300	94056013	33	41.8	22.02	205	87	41.9	15.43	40.8
350	3831165139	60	42.0	22.00	307	175*	42.1	15.53	17.5

Table 7: Recoverable success between SALSA PICANTE [2] and NoMod ML-attack on dataset with variable dimensions n and q, on binary secrets and gaussian error with $\sigma = 3$.

$h = 175$, demonstrating the scalability of our approach to high dimensions with dense secrets where transformer-based models falter.

Sample Complexity and Primal uSVP Hardness

Another critical difference is sample complexity. SALSA requires millions of samples, and PICANTE relies on $4n$ preprocessed samples to construct millions of samples for training. In contrast, our method succeeds with only a few thousand constructed LWE samples from the initial $4n$ samples (sampled from RLWE). This low sample complexity, combined with fast CPU-based training and recovery, makes our method practical even on modest computational resources such as small clusters or high-end desktop machines. In contrast to previous approaches that rely on thousands of CPUs and GPUs and massive datasets, our approach offers a lightweight yet effective alternative for learning-based cryptanalysis of LWE.

Finally, Tables 6 and 7 also report the estimated hardness of solving the same instances with a primal uSVP attack (using the LWE estimator). In many cases, the $\log_2$(rop) values exceed 50, indicating exponential hardness for traditional lattice algorithms. For example, at $n = 128$ in the SALSA comparison, the primal uSVP estimate is $\approx 2^{53.6}$ rop, yet our ML attack succeeds in under one hour. This contrast highlights that our approach not only reduces runtime compared to prior ML-based methods, but also recovers secrets in regimes that are considered hard even for state-of-the-art lattice attacks.

Qualitatively, the performance differences concentrate in two areas. First, our preprocessing pipeline optimizes the heavy lattice-reduction steps that PICANTE also significantly relies on. Second, our robust regression-based secret recovery is orders of magnitude faster, terminating within seconds instead of hours or days as in transformer training. Moreover, SALSA struggles with higher hamming weights, particularly when secrets were not sparse, due to its black-box transformer architecture and absence of preprocessing; our white-box ML approach effectively avoids this failure mode by down-weighting wraparounds samples rather than attempting to learn them.

4.4 Results on SALSA VERDE Settings

We also evaluate our implementation against SALSA VERDE, focusing specifically on binary and ternary secrets, while leaving Gaussian secrets aside. This choice stems from our interest in testing very short secrets derived from the Kyber key generation algorithm. Consequently, we did not implement a Gaussian approximation for secrets with a specified Hamming weight. Nevertheless, we sought to benchmark our improvements against VERDE to determine whether our method could approach, or even surpass, the performance of the third improvement of the SALSA attack.

For preprocessing, we reduce the matrix A using four AMD EPYC 7702P CPUs running at 2.00–2.18 GHz. We performed the preprocessing step following the approximations of binary secrets; training, however, was conducted on both binary and ternary secrets to minimize overall preprocessing time. We restricted computation to four processors by exploiting the algebraic structure of RLWE, enabling us to process each of the four matrices without resampling. This optimization significantly reduced the resources needed, lowering the requirement from 64 matrices to only four while keeping the effective number of samples unchanged. Specifically, we extracted 256 short vectors for the $n = 256$ case and 100 short vectors for the $n = 350$ case. Leveraging RLWE's structure, we expanded these vectors into $4 \times 256 \times 256$ and $4 \times 100 \times 350$ samples, respectively, which we then used for training.

Attack	s	n	256	256	350	350
		q	3329	842779	1489513	94056013
SALSA VERDE	Binary	Best h	8	33	12	36
		$\log_2$ rop[1]	65.5	45.7	55.5	45.3
		Recover hrs (1 GPU)	1.5	3	1.6	1.6
		Total hrs	3	10.5	17.6	218
	Ternary	Best h	9	24	13	36
		$\log_2$ rop[1]	66.3	45.4	55.5	45.3
		Recover hrs (1 GPU)	3	7.5	25.6	17.6
		Total hrs	4.5	15	41.6	234
		samples	4M	4M	4M	4M
		ρ_A	0.77	0.43	0.61	0.38
		Preproc. hrs · CPUs	$1.5 \cdot 7812$	$7.5 \cdot 7812$	$16 \cdot 5000$	$216 \cdot 5000$
NoMod ML-Attack	Bin.	Best h	7	21	9	24
		$\log_2$ rop[1]	64.9	45.0	54.9	44.7
	Ter.	Best h	6	18	10	24
		$\log_2$ rop[1]	64.3	44.8	55.2	44.7
		samples	$262k$	$262k$	$280k$	$280k$
		ρ_A	0.77	0.42	0.65	0.41
		Total hrs · CPUs	$10.6 \cdot 4$	$10.7 \cdot 4$	$11.1 \cdot 4$	$10.7 \cdot 4$

Table 8: Comparison against SALSA VERDE [30] settings with Gaussian errors ($\sigma = 3$).

As shown in Table 8, our method successfully recovered sparse secrets in both the binary and ternary settings. However, it did not achieve Hamming weight recovery comparable to

VERDE, underscoring the performance gap relative to highly engineered transformer-based approaches. In the case of $n = 256$, our preprocessing reached a reduction quality similar to VERDE's, although at a higher computational cost. However, our algorithm required significantly more time per tour for $n = 256, q = 3329$, a difference likely attributable to hardware disparities and training architectures. Indeed, in our experiments, we observed large runtime fluctuations across different clusters: identical implementations occasionally required up to four times longer to converge on different clusters. Because VERDE does not disclose its hardware setup, direct runtime comparisons are not meaningful, and we instead rely on ρ_A to compare preprocessing quality.

The authors of VERDE report successful recovery when at least 67% of samples remain linear (i.e., not wrapped by modular reduction). Interestingly, in our experiments, recovery was sometimes possible even below this threshold. For instance, our robust Regressor successfully recovered a ternary secret at $n = 350$, $q = 1489513$, and Hamming weight 10, with only $\sim 61\%$ inliers. Similar recoveries occurred in the 62–66% range, with multiple successful extractions at 62–64% for $n = 350$, $q = 94056013$, and Hamming weights between 19 and 24.

Conversely, some attempts failed even at higher inlier rates of 70–73%, typically when the number of samples was insufficient or when training subsets were selected using additional criteria beyond the estimated output variance. In particular, we experimented with discarding samples not only based on their reduction factor but also on their prior probability of being inliers under the approximated distribution. This approach often degraded performance. The reason is inherent to the learning process: the most informative samples are those further from the mean, as they shape the model's parameters most strongly. Standard least squares loss naturally emphasizes such samples by assigning them higher penalties. Therefore, focusing disproportionately on samples near the mean, despite their higher probability of being inliers, proved detrimental in this context.

In summary, although our method underperformed relative to VERDE, these experiments provide valuable insights into the uncertain inlier-rate threshold required for successful recovery. While sufficient sample sizes generally stabilized performance, we observed a broader and more variable inlier threshold at smaller sample counts. This observation suggests that scaling up the number of samples could yield more consistent recovery behaviour.

4.5 Results Against State-Of-The-Art Attacks

Table 9 summarises our evaluations of various attacks on real-world CRYSTALS-Kyber MLWE instances. The benchmarked results for uSVP, SALSA FRESCA, The Cool and The Cruel (CC), and MiTM (which solves Decision-LWE) are taken from [34], while our experiments were executed in parallel on 64 AMD EPYC 7702P CPUs running at 2.00–2.18 GHz.

We restricted our evaluation to the Kyber parameter sets, leaving out RLWE instances due to time constraints. In particular, preprocessing became infeasible for $n = 1024$ and large q on our hardware. Moreover, our entire framework is designed explicitly around Kyber, implementing its key generation mechanism. For these reasons, we focused exclusively on Kyber settings in this study. That said, homomorphic encryption (HE) parameter sets could also be tested with faster hardware and larger memory resources available.

Attack		Kyber MLWE Setting (n, k, q)		
		(256, 2, 3329) binomial	(256, 2, 179067461) binomial	(256, 3, 34088624597) binomial
uSVP	Best h	-	-	-
	Recover hrs (1 CPU)	> 1100	> 1100	> 1100
SALSA FRESCA	Best h	9	18	16
	$\log_2$ rop[1]	108.3	58.1	67.3
	$\log_2$ samples	20.93	20.93	20.93
	ρ_A	0.88	0.67	0.69
	Preproc. hrs · CPUs	28 · 3216	11 · 3010	23 · 1843
	Recover hrs · GPUs	8 · 256	16 · 256	6 · 256
	Total hrs	36	27	39
CC	Best h	11	25	19
	$\log_2$ rop[1]	110.0	58.7	67.5
	$\log_2$ samples	16.61	16.61	16.61
	ρ_A	0.88	0.67	0.69
	Preproc. hrs · CPUs	28 · 161	11 · 151	23 · 92
	Recover hrs · GPUs	0.1 · 256	42 · 256	0.9 · 256
	Total hrs	28.1	53	34
NoMod ML-Attack	Best h	4	8	6
	$\log_2$ rop[1]	101.9	57.0	66.1
	$\log_2$ samples	14.94	14.94	16.21
	ρ_A	0.83	0.64	0.78
	Prep. + rec. hrs · CPUs	44 · 64	44 · 64	44 · 64
MiTM (Decision-LWE)	Best h	4	12	14
	$\log_2$ rop[1]	101.9	57.6	67.2
	Memory requirements	10MB	> 3.3TB	> 42TB
	Preproc. hrs · CPUs	0.5 · 50	1.6 · 50	4.4 · 50
	Decide hrs (1 CPU)	0.2	0.01	25
	Total hrs	0.7	1.61	29.4

Table 9: Benchmark against existing machine learning-based attacks on real-world settings of CRYSTALS-Kyber. Details about other attacks taken from [34].

Compared to the uSVP attack, our method successfully recovered the correct secret for sparse secret instances, whereas uSVP remained computationally infeasible (requiring more than 1100 hours on a single CPU). Against SALSA FRESCA and The Cool and The Cruel, our approach required more preprocessing time, yet did not reach comparable recovery performance. In particular, for $k = 2$ our implementation achieved a deeper reduction than both SALSA FRESCA and CC, owing to the use of a priority queue and automatic updates after stalling. However, for instances with very large q, our method only managed to complete a single tour of BKZ 2.0 within the first 44 hours, resulting in a much weaker reduction factor. We therefore hypothesize that longer preprocessing would yield comparable or even deeper reduction, though hardware limitations prevented us from verifying this claim.

Another consistent difference lies in the key generation process. As explained earlier, we rely on the Kyber key generation mechanism, with modifications for binary and ternary secrets as well as for cases with specified Hamming weights. Since our algorithm generates the underlying polynomials independently, any Hamming weight not divisible by k introduces dependencies in the generation process. To avoid this, we restricted our tests to Hamming weights that are multiples of k, ensuring that each polynomial corresponding to a secret share maintains the same sparse structure and remains independent. By contrast, the other benchmarked attacks used the NumPy library for key generation, which may produce polynomials with different sparse structures.

In summary, while our NoMod ML-Attack managed to retrieve sparse secrets under Kyber settings, its performance remained limited by hardware constraints and its relatively simple architecture, falling short of state-of-the-art designs. Specifically, although in previous experiments we were able to recover secrets with inlier rates as low as 60–67%, here the effective threshold was much higher than that of FRESCA (which recovers secrets for 59–63% inliers) and even worse than VERDE's reported threshold of 67%, since in some cases we failed to recover secrets despite inlier rates reaching 69%. We attribute this stricter requirement to the high dimensionality of the instances and the use of centered binomial secrets. For higher-dimensional cases with $k = 3$, our preprocessing algorithm did not converge, completing only a few BKZ 2.0 tours with block size 20 before stalling, and thus leaving the reduction factor significantly higher than that achieved by competing implementations.

4.6 Ablation Study

To understand the individual contribution of each optimisation introduced in our attack pipeline, we conduct a systematic ablation study. Our framework combines several algorithmic and statistical improvements, ranging from structural enhancements in the problem representation to refined lattice reduction strategies and robust machine learning models. While the whole system demonstrates strong performance, it remains unclear how much each component contributes to overall efficiency and success.

By evaluating each design choice independently, we aim to (i) quantify their impact on attack efficiency and success probability, (ii) identify trade-offs between runtime and reduction quality, and (iii) provide practical guidance for future ML-assisted lattice attacks. This section reports the empirical results and insights obtained from this analysis.

4.6.1 Leveraging the Algebraic Structure of Ring-LWE

This sample amplification technique drastically reduces the computational burden of preprocessing while increasing the diversity of training data. More notably, it appears to reduce the dependency on high-quality (inlier) samples for successful secret recovery. For example, in the binary secret settings with $n = 128$ and noise parameter $\eta = 3$, the original method required the reduction of 64 matrices in parallel and achieved successful recovery in approximately 7 hours using Huber Regressor, given that at least 82% of the samples were inliers. In contrast, the ring-amplified method succeeded in just 36 minutes using only four independently reduced matrices and required merely 74% inliers for full key recovery.

These results mark not only a substantial computational acceleration but also a practical reduction in the necessary hardware to construct the training dataset.

These results demonstrate that exploiting the structure of Ring-LWE samples can yield improvements in attack efficiency and success rate, even on constrained hardware platforms. This optimisation suggests a new line of research into structural vulnerabilities in ring-based schemes under machine learning-assisted lattice attacks.

4.6.2 Lattice Embeddings

The choice of lattice embedding has a direct impact on the quality and efficiency of the reduction stage. Different embeddings place the LWE matrix and its associated vectors differently within the augmented lattice, which in turn affects both the speed of BKZ tours and the quality of the reduced basis. Prior work has shown that embeddings such as the classical dual embedding or SALSA VERDE can each offer advantages depending on the dimension and secret distribution. In this subsection, we compare these approaches against our custom embedding, with the goal of quantifying trade-offs between per-tour runtime, reduction quality, and overall attack time.

The experimental data presented in Figures 7 and 8 show that, across tested dimensions, none of the three embeddings exhibits a decisive long-term advantage over the others. In the standard deviation plots for $n = 128$ and $n = 150$ (Figure 7), the dual embedding yields only a marginally faster decrease in σ_b compared to both SALSA VERDE and the custom scheme. For $n = 128$, the reduction under the dual embedding is slightly smaller than VERDE and the custom embedding over the complete BKZ process. Similarly, at $n = 150$, the custom embedding sits between the dual and VERDE curves. Yet, all three trajectories converge to the same asymptotic σ_b, indicating that the steady-state quality of the reduced basis is effectively equivalent regardless of embedding choice.

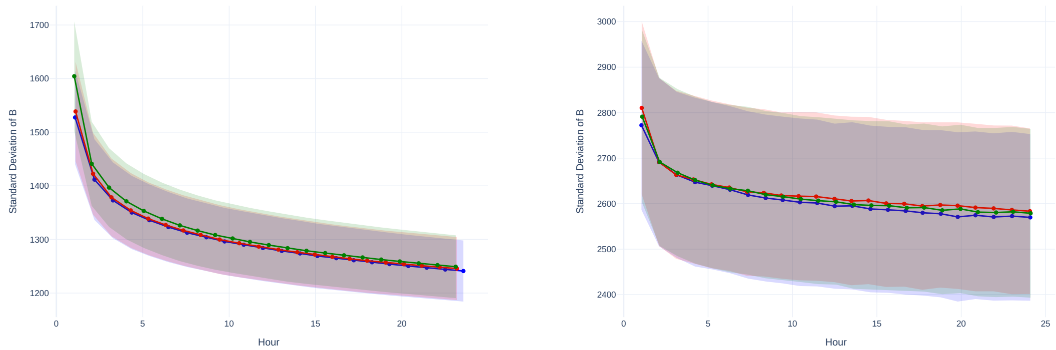


Figure 7: Comparison between standard deviation of B while using SALSA PICANTE or dual embedding (in blue), SALSA VERDE embedding (in red), or the custom embedding (in green) for different dimensions n ($n = 128$ on the left and $n = 150$ on the right).

On the other hand, the mean attack-time curves in Figure 8 reveal that while the two reductions are asymptotically equivalent in practice, notable differences emerge at smaller dimensions. In low dimensions ($n \leq 90$ for binary secrets, $n \leq 70$ for CBD), the dual embedding is slower because its per-tour reduction cost is higher, and a single tour often suffices to recover the secret. As n increases into the mid-range (roughly $90 < n \leq 120$ in the binary case and $70 < n \leq 100$ in CBD), the stronger per-tour reduction quality of the dual embedding pays off: it recovers the secret in fewer tours (1-3 tours), whereas SALSA and the custom embedding require 2-3$\times$ more tours to reach the same success probability, thus widening the attack-time gap in favour of the dual embedding. Beyond the mid-range dimensions, all three attack-time curves converge, indicating that asymptotically, the dominant factor becomes the reduction strength achieved per unit time.

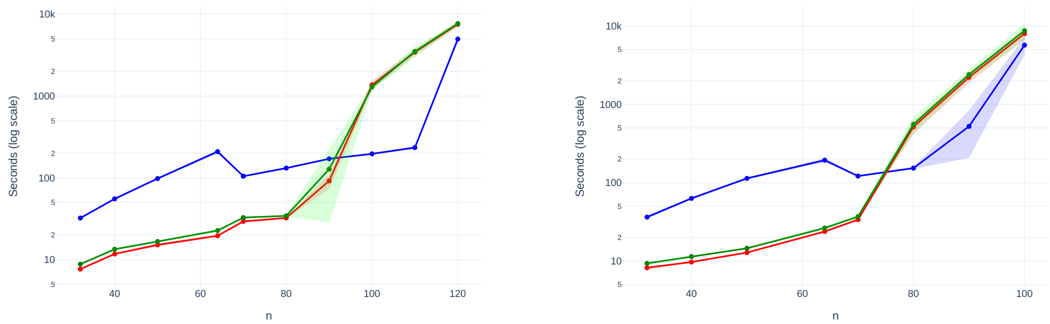


Figure 8: Comparison between total attack time using different embeddings over different dimensions n. SALSA VERDE embedding (in red), dual embedding (in blue), or custom embedding (in green) for binary secret distribution on the left and centered binomial distribution on the right.

To better understand this behaviour, we decompose the total attack time into two components: the *time per tour* and the *reduction quality per tour*. The previous results reveal a key trade-off: both the SALSA VERDE and custom embeddings yield significantly faster tours (up to 30–40% shorter than those for the dual embedding), but at the expense of slightly weaker per-tour reduction quality. As a result, when only a few BKZ2.0 tours are required (e.g., in low-dimensional spaces), the more substantial reduction per tour of the dual embedding can outweigh its slower runtime. Conversely, in high-dimensional settings where many tours are necessary, the faster tour times of SALSA VERDE and the custom embedding amortize better, finally closing the performance gap.

Configuration	Time	Reduction factor
Dual	219 s	0.37
SALSA-VERDE	45 s	0.57
Custom	47 s	0.70

Table 10: Comparison between different lattice embeddings after a first tour of reduction (with FLATTER) on $n = 128$ and $q = 3329$ for the binary secret case. All three configurations use the same sample size over the same matrix.

Table 10 complements our empirical findings by comparing the initial reduction quality and timing for each embedding strategy. The SALSA VERDE embedding clearly offers the fastest per-tour runtime, completing the first BKZ2.0 reduction tour in only 45 seconds. However, this speed comes with a cost: its reduction factor is significantly worse than that of the dual embedding (0.57 versus 0.37), indicating a weaker basis after one tour. While our proposed custom embedding matches VERDE in speed, it does not yield any gain in reduction quality (its reduction factor (0.70) is actually the worst among the three). These results suggest that although custom placement of A was theoretically motivated by variance asymmetry, it does not translate into practical advantages in this setting. Overall, both SALSA VERDE and the dual embedding exhibit distinct strengths: VERDE accelerates the per-tour reduction phase, making it appealing for time-sensitive pipelines, whereas the dual embedding achieves stronger basis quality per tour, which can lead to faster convergence and fewer total reduction passes.

While all embeddings behave asymptotically similarly at higher dimensions, practical issues arise with the dual embedding when operating over large moduli. In particular, for instances with $\log_2 q > 32$, we observed that the dual embedding frequently overflows in the `C++` implementation of BKZ2.0 and demands prohibitively high memory consumption. This behaviour is not present in the SALSA VERDE embedding: because its per-tour reduction is weaker, the lattice basis does not expand as aggressively, thereby avoiding numerical instability. A concrete example of this limitation occurs in the Kyber parameter regime ($n = 256$, $k = 3$, $\log_2 q = 35$), where we were only able to run our implementation using the VERDE embedding. We also note that VERDE generally requires lower floating-point precision than the dual embedding to achieve stable reductions. Nevertheless, these issues have little impact in practice for smaller moduli ($\log_2 q < 30$), where we could use both embeddings interchangeably.

4.6.3 Optimised Sample Size

Table 11 compares the impact of different sample sizes m on the reduction quality and runtime for the case $n = 128$, $q = 3329$, and binary secrets, all using the dual embedding. We evaluate three configurations: the full-rank choice $m = n = 128$, the reduced heuristic $m = 0.875 \cdot n = 112$ proposed in SALSA VERDE [30], and our analytically derived optimum $m^* = 134$ from Equation 3.19.

	First Loop			**After 16 h**	
m	Time	ρ_A	$\sigma_{R\tilde{b}}$	ρ_A	$\sigma_{R\tilde{b}}$
n (full-rank) $= 128$	143 s	0.38	4165	0.218	2443
$0.875 \cdot n = 112$	295 s	0.40	4449	0.226	2505
$m^* = 134$	219 s	0.37	4095	0.213	2381

Table 11: Comparison between different numbers of samples after a first tour of reduction (with FLATTER) and after 16 hours of preprocessing on $n = 128$ and $q = 3329$ for the centered binomial secret case. All three experiments used the dual lattice embedding.

The results highlight a clear trade-off. The heuristic setting $m = 112$ performs noticeably worse, both in terms of runtime (almost twice as slow at 295 seconds) and reduction factor (0.40) compared to the other two. In contrast, our optimised choice $m^* = 134$ almost matches the full-rank case in reduction strength (0.37 versus 0.38) while also recovering additional samples that we could exploit in subsequent stages of the attack. Although the full-rank configuration is slightly faster (143 s versus 219 s for m^*), it does not provide the same benefit in terms of usable samples.

From a broader perspective, the optimised m^* balances algebraic and algorithmic considerations: it maintains the reduction quality of the best performing setting while simultaneously increasing the pool of exploitable samples, leading to higher downstream success rates in both the LWE and MLWE attack pipelines. As more reduction tours are applied, we expect m^* to consistently achieve lower effective reduction factors compared to both the heuristic and the full-rank configurations, reinforcing its advantage as a principled choice.

4.6.4 BKZ Block Size

In our experiments, we observed that while increasing the final block size does improve the average reduction quality, as reflected in lower Gram-Schmidt vector norms and tighter bounds on the output $\tilde{b}$, the marginal gains beyond $\beta = 40$ are outweighed by the computational overhead. In particular, moving from a final block size of 40 to 50 significantly increased the run time without a proportionate improvement in the standard deviation of $\tilde{b}$, which is a critical feature in our subsequent machine learning phase. These preliminary results led us to favour more moderate block sizes (e.g. $\beta = 40$).

	First Loop			**After 16 h**	
β	Time	ρ_A	$\sigma_{R\tilde{b}}$	ρ_A	$\sigma_{R\tilde{b}}$
40	5543 s	0.236	2619	0.205	2287
4:40:2	270 s	0.363	3950	0.227	2523
20:40:10	1731 s	0.244	2716	0.213	2381

Table 12: Comparison between different progressive strategies after a first tour of reduction (with BKZ) and after 16 hours of preprocessing on $n = 128$ and $q = 3329$ for the centered binomial secret case. All three experiments used the dual lattice embedding.

Additionally, as can be visualised in Table 12, the granularity of the block size update steps plays a non-trivial role for smaller block sizes ($\beta < 40$). We found that an update interval of 10 was effective, offering a good balance between performance and reduction depth. However, as the block size increases, maintaining such a wide update interval becomes prohibitively expensive; more frequent updates (e.g., every five or even two steps) yield more consistent gains without excessive cost escalation. We also note that for higher-dimensional lattices (e.g. $n \geq 200$), a higher final block size becomes necessary to achieve meaningful reduction, which exponentially increases the overall preprocessing time.

Finally, since our goal is not to recover a specific short vector (as in traditional dual or primal attacks), but rather to reach a desired average norm distribution among the output samples $\tilde{b}$, we can afford to choose a final block size lower than is typically required in the dual-lattice setting. Instead of relying solely on deep reduction, we shift our strategy toward increasing the number of tours to extract more short vectors probabilistically.

4.6.5 Choice of Machine Learning Algorithm

A key aspect of this work is the selection of a machine learning model capable of learning from partially corrupted training data, which naturally arises in the context of recovering secrets from Module-LWE samples. Through empirical evaluation, we found that the Tukey Regressor offers the best trade-off between robustness to outliers and sensitivity to structure in the data. In particular, successful recovery was achievable when at least 63% of the training set consisted of clean (non-modular) samples of the form $(RA, R\mathbf{b} = RA \cdot \mathbf{s} + R\mathbf{e})$.

We experimented with a variety of models deliberately chosen from white-box models (over the black-box neural networks used in prior literature) to recover dense secrets (and not just the sparse ones typically targeted in ML-based LWE attacks [1, 2, 30, 31]). In our initial setup, we had access to only a small fraction of clean samples $\tilde{b}$, and a high proportion of outliers. Robustness was therefore critical. We tested three robust linear regression models: *Huber Regressor*, *RANSAC* (RANdom SAmple Consensus) Regressor, and *Tukey's Biweight Regressor*. We excluded the *Theil–Sen Regressor*, as it is prohibitively slow for large datasets in higher dimensions and thus infeasible in our case.

Interestingly, we observed that the minimum percentage of non-modular samples required for reliable recovery is not fixed, but rather depends on the distribution of the secret and error. In the binary and ternary cases, where the secret components are sparse and take values from a small set, recovery remained feasible even when up to $35\% - 40\%$ of the training labels were corrupted. However, for centered binomial distributions, which produce denser and more variable outputs, the model became significantly more sensitive to noise. In these settings, we found that recovery usually required at least $65\% - 69\%$ of the training set to be composed of clean samples. This increased sensitivity is likely due to the greater overlap between error vectors and small modular shifts, which blur the distinction between correct and incorrect labels and complicate the learning process. As a result, the regression model needs a higher proportion of accurate data points to identify the correct underlying structure of the secret.

To compare the models, we preprocessed and trained on a dataset with parameters $n = 128$, $q = 3329$, and a binary secret for 24 hours, testing recovery hourly with all three algorithms. Figure 9 shows the comparative performance between the models.

- *RANSAC Regressor* was the slowest and impractical, taking up to one hour per run for different data subsets and producing poor results. We also tested RANSAC using Huber as the inlier model to increase robustness, but without significant gains.

- *Huber Regressor* offered a good balance: training each configuration took approximately one second for five different data splits (based on the lowest standard deviations of $R\mathbf{b} = RA\mathbf{s} + R\mathbf{e}$). Huber successfully recovered the secret in around 15 hours.

- *Tukey's biweight Regressor* was both the fastest and most accurate, recovering the correct secret within the first or second hour of training.

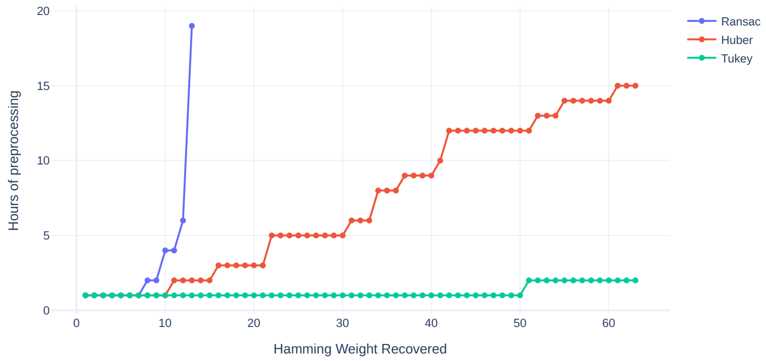


Figure 9: Comparison of secret recovery across robust regression models.

We also conducted a hyperparameter study on the Huber Regressor, focusing on the parameter ϵ. As shown in Figure 10, we tested values from $\epsilon = 1$ to 3 and found that $\epsilon \approx 1.1$ gives the best trade-off between robustness and recovery accuracy. Values of $\epsilon \geq 1.35$ consistently failed to recover the secret, exhibiting performance comparable to RANSAC's. These findings suggest that overly lenient settings of ϵ reduce the robustness of the estimator, causing it to behave too similarly to ordinary least squares and become excessively sensitive to wraparound-induced outliers.

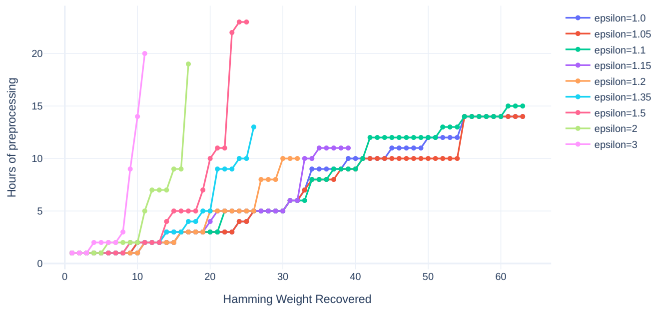


Figure 10: Performance of Huber regressor for different values of ϵ.

For *Tukey's Biweight Regressor*, we fix the outlier cutoff at $c = 1$. This threshold corresponds to the first standard deviation of the normalised residuals, since the inlier–selection routine implemented in *statsmodels* standardises residuals to unit scale (i.e., we work with standardised residuals r_i whose inlier spread is ≈ 1). With this normalisation, c becomes dimension-less and invariant across parameter sets and BKZ tours, so $c = 1$ has a straightforward interpretation: points with $|r_i| \leq 1$ (within one standard deviation) receive high weight, while $|r_i| > 1$ are rapidly down-weighted.

Empirically, $c = 1$ recovered the secret fastest, especially when a significant fraction of labels are corrupted ($\sim 65\%$). The reason is twofold: (i) wraparound-affected labels produce standardised residuals far beyond the unit range (due to the $\pm q$ shift), so $c = 1$ sharply suppresses their influence early in training; (ii) genuine inliers, after normalisation, concentrate within $|r_i| \lesssim 1$, so they retain sufficient weight to identify the linear structure.

Larger cutoffs ($c > 1$) admit too many corrupted samples with non-negligible weight, slowing or corrupting convergence; smaller cutoffs ($c < 1$) discard an excessive number of valid samples, increasing estimator variance.

Interestingly, we observed a similar success rate threshold phenomenon to the one reported in transformer-based approaches. Specifically, the Tukey Regressor systematically recovers the centered binomial secrets when the proportion of inliers reached around 67%. This estimate aligns with the empirical threshold seen in SALSA VERDE [30] and SALSA FRESCA [31]. Crucially, we achieved these results without relying on computationally intensive black-box secret extraction procedures or the millions of training samples typically required.

4.7 Outlier Detection

Finally, we looked into whether explicit outlier detection could enhance secret recovery even more, in addition to robust regression techniques. The idea was to recover a higher Hamming weight of the secret or even the entire secret vector by filtering or re-weighting corrupted samples before feeding them into the regression model.

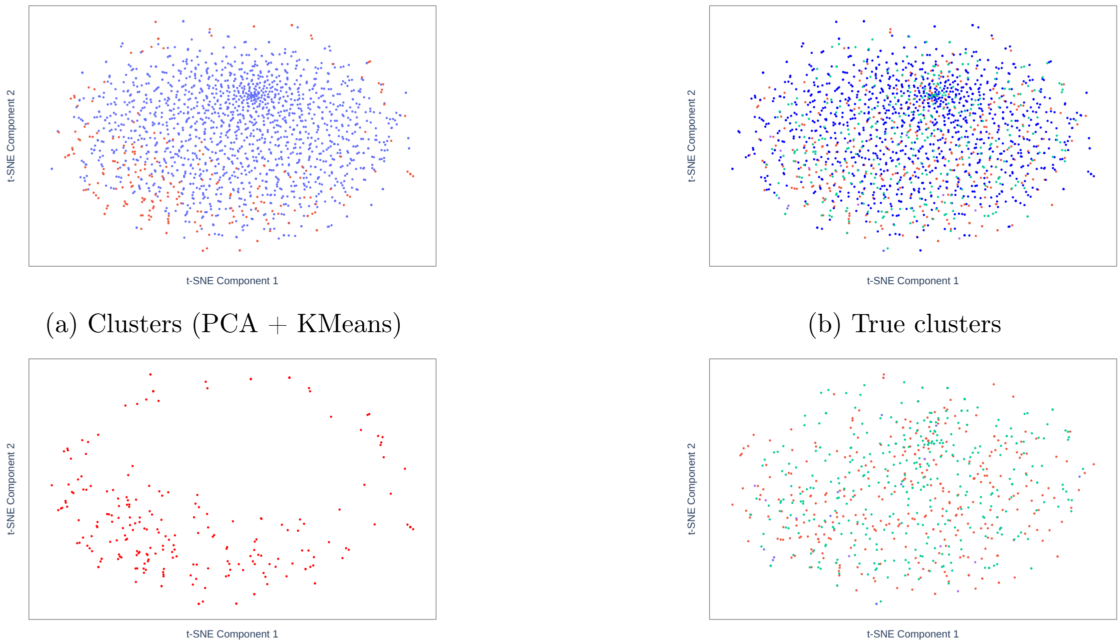
Overall, at low levels of corruption, where the number of samples impacted by wraparound remained small, these methods generally showed some promise. All methods, however, found it challenging to discern inliers from outliers once the percentage of corrupted samples exceeded about 65%. Specifically, we discovered that their clusterings and rankings quickly deteriorated, resembling the behaviour of a random classifier. In certain instances, like EM with less than 60% inliers, the detection mechanism even worsened the dataset quality by systematically misclassifying genuine inliers as outliers. These results highlight the strong dependence of outlier detection on the underlying model assumptions, and show that at high corruption rates, such methods not only fail to provide an advantage but can actively hinder recovery.

Gradient-Based Detection

We first tested the gradient-based scoring approach, where we ranked potential candidates according to their influence on the regression objective. At low outlier rates, this produced a meaningful ranking, with inliers consistently assigned higher priority. However, as the fraction of outliers increased towards 65%, the positional ranking became nearly indistinguishable from a random ordering. This indistinguishability indicates that the corrupted labels dominate the gradient signals once they form the majority, effectively erasing the separation between inliers and outliers.

Unsupervised Gradient-Based Detection

We also evaluated unsupervised gradient-based outlier detection to determine whether it could enhance secret recovery. Figure 11 shows a visualization of the results. The first subplot depicts the clusters obtained after PCA and K-Means on the gradient components. The second subplot shows the true clusters based on the known offset vector c. The third subplot highlights the samples identified as outliers by the unsupervised method, while the fourth subplot shows the true outliers according to c.



(a) Clusters (PCA + KMeans)

(b) True clusters

(c) Outliers detected

(d) True outliers

Figure 11: Unsupervised gradient-based detection: comparison of detected versus true clusters and outliers for $n = 128$, $q = 3329$, and inlier rate = 53.5%.

From these visualisations, we observe that although the gradients nominally cluster near specific points, the unsupervised method fails to separate inliers from outliers correctly. Outliers are scattered throughout the space and do not form well-defined clusters, which explains why our technique could not improve recovery. Essentially, the gradient information is insufficiently distinct: the inliers and outliers are nearly indistinguishable, especially when the proportion of corrupted samples is high. These results confirm that unsupervised gradient-based detection is unreliable for datasets with a significant fraction of outliers and cannot be used to enhance secret recovery in these scenarios.

Expectation–Maximization

Lastly, we tried the EM-based outlier detection process, with inliers and outliers as latent classes. Although the algorithm was sometimes practical when the dataset had more than 70% inliers, it invariably failed once the inlier fraction dropped below this level. In these regimes, the EM iterations tend to mark many previously selected inlier candidates as outliers, thus shrinking the effective inlier set and degrading the quality of regression. As an example, in one experiment with $n = 128$, $q = 3329$ and a binary secret, the non-modular candidates initially covered 61% of the dataset. Still, successive EM updates gradually reduced this fraction down to the range of 52–53%.

We also experimented with a different strategy in which, rather than updating the candidate inliers directly, we only updated the inlier probabilities based on learned parameters. Here, the technique did not revert to poorer inlier rates, but it plateaued at more or less the same level as before: initiating from 61%, the proportion of actual best candidates in the updated inlier set varies between 59% and 62% in consecutive iterations. These further results demonstrate that although probability updates prevent catastrophic breakdown, they have no real benefit and are effectively akin to a random choice in high-outlier regimes.

Chapter 5

Conclusion

Our research investigated the intersection of machine learning and lattice cryptanalysis within the context of the Module-LWE problem in Post-Quantum cryptographic schemes. We aimed to develop and assess a pipeline that converts noisy lattice instances to data suitable for strong statistical learning, thus facilitating efficient secret recovery.

Extending ideas from previous work, we described a set of improvements deepening the reduction for machine learning attacks. In particular, we proposed a progressive BKZ block size scheduling technique to stabilize the quality of the reduction, a new approach for saving and recycling short vectors across several reduction tours, and an optimization exploiting the algebraic structure of Ring-LWE to increase the effectiveness of reduced samples. Moreover, we also refined the pipeline by analytically optimizing the number of samples per matrix and lowering the number of matrices needed for reduction.

Our empirical findings illustrated that these strategies yield tangible advancements over prior ML-attacks like SALSA and PICANTE. Despite limited hardware and time budgets, our method outperformed these first transformer-based approaches. Notably, we illustrated that lightweight regression-based models, while demanding deeper preprocessing than neural network architectures, significantly trimmed the number of training samples and computational effort overall, reducing hardware demands. Such a scenario manifests in a trade-off between preprocessing time and overall resource usage, rendering these techniques more viable for practical cryptanalysis.

Our evaluation against SALSA VERDE highlighted both the potential and the limitations of our approach. While our method successfully recovered sparse binary and ternary secrets under certain conditions, it did not match the maximum recoverable Hamming weights reported by VERDE. At the same time, our experiments revealed that successful recovery was possible even below VERDE's reported 67% inlier threshold, with multiple cases recovered at 61–64%. This discrepancy suggests that our regression-based method may exploit different statistical properties than transformer-based designs, offering new perspectives on the inlier-rate requirements for recovery.

When compared to other state-of-the-art implementations, such as SALSA FRESCA, The Cool and the Cruel, MitM, and uSVP attacks, our NoMod ML-attack demonstrated mixed outcomes. For sparse secrets and moderate parameter sizes, our approach outperformed uSVP by achieving recovery where uSVP was computationally infeasible. Against FRESCA and Cool and the Cruel, our preprocessing pipeline achieved deeper reductions in lower moduli, albeit at higher runtimes. However, while we were able to recover certain sparse secrets, our method consistently underperformed against these modern ML-based attacks, yielding successful recovery only at significantly lower Hamming weights. This gap suggests that further methodological advances are necessary before such regression-driven approaches can realistically challenge state-of-the-art cryptanalytic techniques.

Finally, across all tested families of experiments, we observed a notable divergence in recoverable inlier thresholds. For enhanced RLWE and binary or ternary secrets, our method succeeded even at 60% inliers. In contrast, for MLWE instances with higher dimensions, centered binomial secrets, and fewer available samples, recovery failed despite inlier rates reaching 69%. This discrepancy highlights the combined effect of dimension, secret distribution, and sample availability on the success of machine–learning–based cryptanalysis.

Our work also reopened several assumptions in the literature that had been previously considered settled. Specifically, we reevaluated different lattice embeddings and other preprocessing techniques, demonstrating that previous assertions regarding their behaviour were not always correct.

Not every exploration was similarly fruitful. Our effort at incorporating new outlier detection methods did not produce quantifiable gains in secret recovery. Nevertheless, these experiments brought techniques not formerly contemplated within this framework and form a basis for continued investigation into a more proficient outlier detector.

To conclude, this work shows how the combination of machine learning with lattice reduction can lead to powerful and resource-saving attacks on LWE. By leveraging improvements in the reduction procedure, the use of algebraic structures, and the adoption of simple, lightweight models, we have shown that it is possible to improve over previous results and, in some cases, even beat classical state-of-the-art approaches. These findings strengthen the case for continued evaluation of Module-LWE security in light of the development of AI-powered techniques.

5.1 Future Work

While this research has pushed forward the state of knowledge on machine learning–based attacks against Module-LWE, a few promising avenues are left open for improvements and future work. One of the main priorities should be a deeper theoretical analysis of the preprocessing step to determine, in particular, the relationship between BKZ block size and the resulting distribution of reduced output samples. Having established this relationship, it would be possible to estimate attack success probabilities from the quality of preprocessing, thus providing a way to predict outcomes independently of purely empirical experiments. In addition, such an analysis would allow for a tighter comparison of the performance of our pipeline with that of classical attacks, which may eventually lead to better security assessments and a re-evaluation of specific parameter settings.

One other promising direction is the application of our methods to more general structured variants of LWE. While we have already suggested improvements for Ring-LWE, the same algebraic enlargements can be applied to Module-LWE, building on concepts from recent papers like [40]. Similarly, investigating the incorporation of modulo-switching methods, utilized in traditional attacks, might achieve even better efficiency in our pipeline.

Algorithmically, the progressive BKZ algorithm remains central to our approach. A more principled strategy for scheduling block size increase, guided by theoretical models such as that presented in [25], can further improve efficiency. Similarly, incorporation of more advanced lattice reduction techniques, such as Jump-and-Pump BKZ or its variant alternatives, represents a natural direction for improvement.

From an implementation standpoint, our experiments were limited by the use of Python for prototyping. Rebuilding the preprocessing pipeline in a lower-level language like C++ would give a significant speedup and enable exploration of higher-dimensional parameter spaces in reasonable timescales.

Lastly, future research ought to explore other methods of reconstructing or approximating non-modular samples more efficiently. Constructing algorithms that can confidently extrapolate pre-modular representations would directly tackle one of the intrinsic challenges in using machine learning on LWE. It could significantly improve the success rate and efficiency of our attack pipeline.

In summary, additional theoretical analysis, algorithmic improvements, and implementation enhancements can significantly reinforce our attack framework. When put together, these directions can offer tighter insight into the actual hardness of Module-LWE and further illuminate the security margins of lattice-based post-quantum cryptography even in the face of AI-based attacks.

Bibliography

[1] Wenger, E., Chen, M., Charton, F. & Lauter, K. E. Salsa: Attacking lattice cryptography with transformers. In Koyejo, S. *et al.* (eds.) *Advances in Neural Information Processing Systems*, vol. 35, 34981–34994 (Curran Associates, Inc., 2022).

[2] Li, C. Y. *et al.* Salsapicante: a machine learning attack on lwe with binary secrets. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, 2606–2620 (2023).

[3] Shor, P. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, 124–134 (1994).

[4] Shor, P. W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing* **26**, 1484–1509 (1997).

[5] Matt Campagna, B. L. & Ott, D. Post quantum cryptography: Readiness challenges and the approaching storm. Tech. Rep. White Paper 14, Computing Community Consortium, Washington, D.C. (2020).

[6] Fowler, A. G., Mariantoni, M., Martinis, J. M. & Cleland, A. N. Surface codes: Towards practical large-scale quantum computation. *Phys. Rev. A* **86**, 032324 (2012).

[7] Gidney, C. & Ekerå, M. How to factor 2048 bit rsa integers in 8 hours using 20 million noisy qubits. *Quantum* **5**, 433 (2021).

[8] Chevignard, C., Fouque, P.-A. & Schrottenloher, A. Reducing the number of qubits in quantum factoring. Cryptology ePrint Archive, Paper 2024/222 (2024).

[9] Gidney, C. How to factor 2048 bit rsa integers with less than a million noisy qubits (2025). `2505.15917`.

[10] AbuGhanem, M. Ibm quantum computers: evolution, performance, and future directions. *The Journal of Supercomputing* **81** (2025).

[11] Chen, L. *et al.* *Report on post-quantum cryptography*, vol. 12 (US Department of Commerce, National Institute of Standards and Technology . . . , 2016).

[12] Dam, D.-T., Tran, T.-H., Hoang, V.-P., Pham, C.-K. & Hoang, T.-T. A survey of post-quantum cryptography: Start of a new race. *Cryptogr.* **7**, 40 (2023).

[13] Regev, O. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)* **56**, 1–40 (2009).

[14] Regev, O. The learning with errors problem (invited survey). In *2010 IEEE 25th Annual Conference on Computational Complexity*, 191–204 (2010).

[15] Balbás, D. The hardness of LWE and ring-LWE: A survey. Cryptology ePrint Archive, Paper 2021/1358 (2021).

[16] Bai, S. *et al.* Improved security proofs in lattice-based cryptography: Using the rényi divergence rather than the statistical distance. *Journal of Cryptology* **31**, 610–640 (2018).

[17] Lenstra, A., Lenstra, H. & Lovász, L. Factoring polynomials with rational coefficients. *Math. ann* **261**, 515–534 (1982).

[18] Schnorr, C.-P. & Euchner, M. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Mathematical programming* **66**, 181–199 (1994).

[19] Chen, Y. *Réduction de réseau et sécurité concrete du chiffrement completement homomorphe*. Ph.D. thesis, Paris 7 (2013).

[20] Chen, Y. & Nguyen, P. Q. Bkz 2.0: Better lattice security estimates. In Lee, D. H. & Wang, X. (eds.) *Advances in Cryptology – ASIACRYPT 2011*, 1–20 (Springer Berlin Heidelberg, Berlin, Heidelberg, 2011).

[21] Albrecht, M. R. On dual lattice attacks against small-secret LWE and parameter choices in HElib and SEAL. Cryptology ePrint Archive, Paper 2017/047 (2017).

[22] Gama, N. & Nguyen, P. Q. Predicting lattice reduction. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 31–51 (Springer, 2008).

[23] Schnorr, C. P. & Shevchenko, T. Solving subset sum problems of densioty close to 1 by" randomized" bkz-reduction. *Cryptology ePrint Archive* (2012).

[24] Haque, M. M., Rahman, M. O. & Pieprzyk, J. Analysing progressive-bkz lattice reduction algorithm (2013).

[25] Aono, Y., Wang, Y., Hayashi, T. & Takagi, T. Improved progressive BKZ algorithms and their precise cost estimation by sharp simulator. Cryptology ePrint Archive, Paper 2016/146 (2016).

[26] Ryan, K. & Heninger, N. Fast practical lattice reduction through iterated compression. In *Annual International Cryptology Conference*, 3–36 (Springer, 2023).

[27] Lyubashevsky, V., Peikert, C. & Regev, O. On ideal lattices and learning with errors over rings. In *Annual international conference on the theory and applications of cryptographic techniques*, 1–23 (Springer, 2010).

[28] Brakerski, Z., Gentry, C. & Vaikuntanathan, V. Fully homomorphic encryption without bootstrapping. Cryptology ePrint Archive, Paper 2011/277 (2011).

[29] Howgrave-Graham, N. A hybrid lattice-reduction and meet-in-the-middle attack against ntru. In *Annual International Cryptology Conference*, 150–169 (Springer, 2007).

[30] Li, C., Wenger, E., Allen-Zhu, Z., Charton, F. & Lauter, K. E. Salsa verde: a machine learning attack on lwe with sparse small secrets. In Oh, A. *et al.* (eds.) *Advances in Neural Information Processing Systems*, vol. 36, 53343–53361 (Curran Associates, Inc., 2023).

[31] Stevens, S. *et al.* Salsa fresca: Angular embeddings and pre-training for ml attacks on learning with errors (2024). `2402.01082`.

[32] Charton, F., Lauter, K., Li, C. & Tygert, M. An efficient algorithm for integer lattice reduction. *SIAM Journal on Matrix Analysis and Applications* **45**, 353–367 (2024).

[33] Nolte, N. *et al.* The cool and the cruel: separating hard parts of lwe secrets. In *International Conference on Cryptology in Africa*, 428–453 (Springer, 2024).

[34] Wenger, E., Saxena, E., Malhou, M., Thieu, E. & Lauter, K. Benchmarking attacks on learning with errors. In *2025 IEEE Symposium on Security and Privacy (SP)*, 279–297 (IEEE, 2025).

[35] Huber, P. J. Robust estimation of a location parameter. In *Breakthroughs in statistics: Methodology and distribution*, 492–518 (Springer, 1992).

[36] Fischler, M. A. & Bolles, R. C. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM* **24**, 381–395 (1981).

[37] Chang, L., Roberts, S. & Welsh, A. Robust lasso regression using tukey's biweight criterion. *Technometrics* **60**, 36–47 (2018).

[38] Dang, X., Peng, H., Wang, X. & Zhang, H. Theil-sen estimators in a multiple linear regression model. *Olemiss Edu* **2**, 1–30 (2008).

[39] Satriawan, A. *et al.* Conceptual review on number theoretic transform and comprehensive review on its implementations. *IEEE Access* **11**, 70288–70316 (2023).

[40] Wu, H. & Xu, G. Enhancing the dual attack against MLWE: Constructing more short vectors using its algebraic structure. Cryptology ePrint Archive, Paper 2022/1661 (2022).

[41] Sun, D., Lee, V. C. S. & Lu, Y. A gradient-based algorithm for trend and outlier prediction in dynamic data streams. In *2017 12th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, 1978–1983 (2017).

Appendix A

Open Source Implementation

The full implementation of the proposed attack framework, including preprocessing, lattice reduction, and machine learning components, is publicly available at:

`https://github.com/cristianbass01/ML-attack-on-MLWE`

The repository includes notebooks for development testing, example cluster configurations, container definition, and scripts to reproduce the experiments presented in this work.