Reduce and Prange: Revisiting Prange's ISD for Solving LPN/RSD over Large Fields

Jiseung Kim[1] and Changmin Lee[2]

[1] Jeonbuk National University, jiseungkim@jbnu.ac.kr
[2] Korea University, changminlee@korea.ac.kr

Abstract. The Learning Parity with Noise (LPN) problem and its regular noise variant are fundamental to many cryptographic primitives. While recent proposals extend these problems to larger fields to enhance cryptographic applications, the security implications of LPN over large fields remain understudied. This gap has facilitated more effective attacks, potentially compromising the security of LPN-based primitives over large fields.

In this paper, we present an improved algorithm for solving LPN over large fields. Our method enhances the classical Gaussian elimination attack, specifically Prange's information set decoding algorithm, by iteratively applying Gaussian elimination to reduced-size matrices rather than the full-size matrix. We call this the "Reduce and Prange (RP)" algorithm and demonstrate its effectiveness for both LPN and its variant with regular noise over large finite fields (e.g., $\mathbb{F}_{2^{128}}$). Building on the RP algorithm, we develop a modified algebraic algorithm for LPN with regular noise over $\mathbb{F}_q$ with $q \geq 2$, extending the work of Briaud and Øygard (Eurocrypt'23).

Furthermore, we demonstrate that Scalable Multiparty Garbling (CCS'23) fails to meet their claimed security levels.

For LPN over large fields (e.g., 128-bit field size), our algorithm reduces bit-security by up to 9 bits relative to Liu et al. (Eurocrypt'24). In the case of LPN with regular noise, our RP algorithm surpasses both Liu et al. and Briaud and Øygard (Eurocrypt'23) under specific parameter settings, achieving up to 16-bit reduction in security.

Keywords: LPN (over large fields), LPN with regular noise, Concrete security

1 Introduction

As the central problem of learning theory and coding theory, the learning parity with noise (LPN) problem has affected numerous cryptographic primitives such as secure arithmetic computations [7, 9, 17, 19, 20, 28, 36, 41, 43, 55, 61, 69], zero knowledge proofs [10, 32, 39, 65] and more [4–6, 14, 25, 29, 35, 44, 45, 47, 50, 70, 71]. The LPN problem is defined as follows.

Problem 1 (Learning Parity with Noise (LPN)). Let m, n, t be positive integers and $\mathcal{R}$ be a ring. Let $\mathcal{C}$ be a probabilistic code generation algorithm such that

$\mathcal{C}(m, n, \mathcal{R})$ returns a matrix $\mathbf{A} \in \mathcal{R}^{m \times n}$. Let $\chi(\mathcal{R}) = \{\chi_{m,t}\}_{m,t \in \mathbb{N}}(\mathcal{R})$ be a family of distributions over $\mathcal{R}^m$ that returns a vector in $\mathcal{R}^m$, where the number of nonzero coefficients in the vector is t.

The computational Learning Parity with Noise (LPN) problem with respect to parameters m, n and t involves obtaining a secret vector $\mathbf{s}$ given instances

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

where $\mathbf{A} \leftarrow \mathcal{C}(m, n, \mathcal{R}), \mathbf{s} \leftarrow \mathcal{R}^n$ and $\mathbf{e} \leftarrow \chi_{m,t}(\mathcal{R})$. We simply say the problem (m, n, t)-LPN problem over $\mathcal{R}$.[1]

To the best of our knowledge, $\mathbb{F}_2, \mathbb{F}_q$ and ring $\mathbb{Z}_{2^\lambda}$ for some $\lambda > 1$ have been primarily considered. Recently, [49] proved that (m, n, t)-LPN over $\mathbb{Z}_{2^\lambda}$ can be reduced to $(m, n, t/2)$-LPN over $\mathbb{F}_2$, which suffices to analyze the security of LPN over both fields $\mathbb{F}_2$ and $\mathbb{F}_q$, respectively. If $\mathcal{R} = \mathbb{F}_2$, then it is called the standard LPN problem or simply LPN. If $\mathcal{R} = \mathbb{F}_q$ with a prime power $q \gg 2$, then we call it 'LPN over large fields', where it is a natural variant of the (standard) LPN. On the other hand, for post-quantum signatures, LPN over $\mathbb{F}_{2^8}$ is usually considered [1–3, 39].

PCG-based primitives often assume a regular error distribution, significantly improving the efficiency of cryptographic applications [9, 10, 18, 20, 22, 28, 30, 31, 33, 34, 42, 46, 56, 59–61, 63–69]. This variant, called LPN with regular noise (regular-LPN), is defined as follows. Even more, these protocols additionally employ a low-noise setting that $t/m = 1/n^\varepsilon$ for some constant $\varepsilon > 0$ with bounded samples.

Problem 2 (LPN with regular noise, regular-LPN). Let m, n, t, β be positive integers with $m = t \cdot \beta$ and $\mathcal{R}$ be a ring. Let $\mathcal{C}$ be a probabilistic code generation algorithm such that $\mathcal{C}(m, n, \mathcal{R})$ returns a matrix $\mathbf{A} \in \mathcal{R}^{m \times n}$. Let $\tau(\mathcal{R}) = \{\tau_{m,t}\}_{m,t \in \mathbb{N}}(\mathcal{R})$ be a family of distributions over $\mathcal{R}^m$. Given m, t, a distribution $\tau_{m,t}$ returns a vector $\mathbf{e} = (\mathbf{e}_1 \| \dots \| \mathbf{e}_t) \in \mathcal{R}^m$ such that $\mathbf{e}_i \in \mathcal{R}^\beta$ is of Hamming weight $|\mathbf{e}_i| = 1$ for $1 \le i \le t$.

The computational LPN with regular noise (regular-LPN) involves obtaining a secret vector $\mathbf{s}$ given instances

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

where $\mathbf{A} \leftarrow \mathcal{C}(m, n, \mathcal{R}), \mathbf{s} \leftarrow \mathcal{R}^n$ and $\mathbf{e} \leftarrow \tau_{m,t}(\mathcal{R})$. We simply say the problem (m, n, t)-regular-LPN problem over $\mathcal{R}$.

[1] Following the previous work [19, 20, 49], we adapt the definition for our purpose. We note that the definition of LPN originally states that the goal is to find $\mathbf{s}$ using oracle access to $\mathcal{O}$, which returns a LPN instance. To be specific, we only consider an LPN where the number of oracle queries are limited to m. (The problem can be considered as a variant of decoding linear code.) This constraint is reasonable because most cryptographic primitives built on LPN use the parameter regime where the number of oracle queries are bounded.

Note that LPN instances of the form $(\mathbf{A}, \mathbf{b}) = (\mathbf{A}, \mathbf{A} \cdot \mathbf{s} + \mathbf{e} \bmod \mathcal{R})$ can be easily transformed into a problem of finding $\mathbf{e}$ when there exists a matrix $\mathbf{H}$ such that $\mathbf{H} \cdot \mathbf{e} = \mathbf{H} \cdot \mathbf{b}$, and vice versa, as demonstrated in [54, Lemma 4.9]. Therefore, we can restrict our attention to $(\mathbf{A}, \mathbf{b}) = (\mathbf{A}, \mathbf{A} \cdot \mathbf{s} + \mathbf{e} \bmod \mathcal{R})$. In particular, this paper mainly focuses on $\mathcal{R} = \mathbb{F}_q$ with $q \gg 2$.

Concrete Analysis for LPN over Large Fields. Despite the development of various applications utilizing LPN over large fields, cryptanalysis has primarily focused on standard LPN. Accurately estimating the security of cryptographic primitives is crucial for evaluating their robustness

The Blum-Kalai-Wasserman (BKW) algorithm [15] is highly effective for standard LPN but becomes less efficient for LPN over large fields when $q \gg 2$. This inefficiency occurs because BKW is designed for small-field settings, relying on a subexponential number of samples to achieve optimal performance. As q increases, the required number of samples grows exponentially, making the algorithm impractical.

Boyle et al. [17] initiates a study of a concrete security estimation of LPN over $\mathbb{F}_q$ in order to instantiate PCG-based protocols. However, Liu et al. [49] demonstrated that this estimation was oversimplified, leading to an overestimation of actual attack costs. Moreover, they demonstrate that to solve LPN over $\mathbb{F}_2$ and $\mathbb{Z}_{2^\lambda}$, advanced algorithms such as [12, 16, 51, 52] yield the most effective results, as expected. While these techniques perform well for LPN over $\mathbb{F}_2$, their effectiveness significantly declines as q increases in LPN over large fields.

Concrete Analysis for regular-LPN (over large fields). In the case of regular-LPN, the work proposed by [21] describes a novel algebraic attack that exploits regular noise distributions on $\mathbb{F}_q$ for any $q \geq 2$. Specifically, the regular noise distribution is used to generate multivariate polynomials, which can reduce the concrete security of regular-LPN. Subsequently, Esser and Santini [38] modified the well-known algorithms for solving LPN into specified algorithms for solving regular-LPN on $\mathbb{F}_2$. They mainly exploit the specific structure derived from the regular distribution of the error vector. These results demonstrate the importance of addressing the regular noise distribution in LPN. Unfortunately, similar to a LPN case, if $\mathcal{R} = \mathbb{F}_q$ with large q, algorithms in [38] are not suitable for solving regular-LPN, except for the permutation-based algorithm.[2]

These two results have in common that, despite powerful applications in the case of LPN over large fields, more advanced analysis algorithms are still lacking. According to [21,24,38,49,53,57], when the field size is sufficiently large, Prange's Information Set Decoding (ISD) algorithm has a comparable cost to other algorithms.

1.1 Our Contribution

In this paper, we revisit Prange's ISD algorithm, and enhance its performance by reducing the computational cost of the Gaussian elimination. We refer to our

[2] This permutation-based algorithm is identical to a Prange's ISD algorithm except for the guessing probability induced by regular noise assumption.

modified version as the Reduce and Prange (RP) algorithm. We then extend this approach to develop a variant specifically for solving regular-LPN, which we call regular-RP. Furthermore, building on the RP algorithm, we propose an algebraic algorithm for regular-LPN over $\mathbb{F}_q$ with $q \geq 2$, extending the work of Briaud and Øygarden [21], sometimes called an algebraic-RP.

Furthermore, our RP analysis shows that the Scalable Multiparty Garbling scheme from [11] does not achieve its claimed security levels. For instance, parameter sets $(m, n, t) = (555, 127, 139)$ and $(m, n, t) = (785, 214, 197)$, which are claimed to provide 80-bit and 128-bit security respectively, actually offer only about 76-bit and 123-bit security according to our analysis.

We implement our attacks using SageMath [62] to determine near-optimal parameter settings. The impact of our algorithms is two-fold:

Implicit Impacts on PCG-like protocols. Recently, Liu et al. [49] provides a bit-security estimation for solving LPN over $\mathbb{F}_q$, especially focusing on PCG protocols. They re-calculate all parameters in [17] and reproduce more accurate security estimation to tightly choose parameters. Furthermore, based on their estimation, they also provide recommendation parameters for PCG applications to achieve 128-bit security.

By applying RP to these parameter settings, we show that RP achieves lower computational complexity than existing methods for solving the LPN problem over large fields in PCG settings. The estimated results for various parameters are given by Table 1. Indeed, the bit-security of LPN over $\mathbb{F}_{2^{128}}$ is reduced by at most 9 bits when $\log q = 128$ compared to [49].

For regular-LPN over $\mathbb{F}_q$ with $\log q = 128$, the results demonstrate that our algorithm outperforms the recent work proposed in [21, 49] for almost parameter settings.[3] For detailed numerical results on solving LPN with regular noise, please refer to Section 5.3.

Furthermore, for the recommended parameters to achieve 128-bit security in [49], we provide numerical results for our algorithms. For LPN defined over $\mathbb{F}_q$ with $\log q = 128$, our RP algorithm reduces the bit-security by 5-11 bits compared to previous works. For all recommended parameters, our estimation algorithm reduces the bit-security of regular-LPN by 8-16 bits. For additional details, see Table 2 and Table 4.

Moreover, when $q = 2$, there exist parameter regimes where our algorithm outperforms other known algorithms. Detailed results can be found in Table 5 and Table 6.

1.2 Technical Idea

We first revisit Prange's Information Set Decoding (ISD) algorithm to describe our approach. To this end, we introduce some notations: For any vector $\mathbf{x} \in \mathbb{F}_q^m$, we denote $|\mathbf{x}|$ as its Hamming weight. Let $(\mathbf{A}, \mathbf{b}) \in \mathbb{F}_q^{m \times n} \times \mathbb{F}_q^m$ be LPN instances and $\mathcal{I} \subset [m]$ an index set of size n. Then, we can obtain $\mathbf{A}_\mathcal{I}$ (Resp. $\mathbf{b}_\mathcal{I}$) by a

[3] Note that [38] primarily focuses on efficiently solving regular-LPN over the binary field. Their algorithms are not well-suited for regular-LPN over large fields.

submatrix of $\mathbf{A}$ (Resp. subvector of $\mathbf{b}$) by collecting i-th row of $\mathbf{A}$ (Resp. i-th coordinate of $\mathbf{b}$) for $i \in \mathcal{I}$.

Using this notation, we present Prange's Information Set Decoding algorithm in Algorithm 1.

Algorithm 1 Prange's ISD

Input: m, n, t: Parameters.
(m, n, t)-LPN over $\mathbb{F}_q$ instances $(\mathbf{A}, \mathbf{b}) \in \mathbb{F}_q^{m \times n} \times \mathbb{F}_q^m$.
Output: The secret $\mathbf{s}$ such that $|\mathbf{b} - \mathbf{A} \cdot \mathbf{s} \bmod \mathbb{F}_q| = t$.

1: **repeat**
2: Choose random index subsets $\mathcal{I} \subset [m]$ of size n
3: Generate a submatrix $(\mathbf{A}_\mathcal{I}, \mathbf{b}_\mathcal{I})$ collecting i-th row of $\mathbf{A}$ (Resp. $\mathbf{b}$) for $i \in \mathcal{I}$
4: **until** $\mathbf{A}_\mathcal{I}$ is invertible and $|\mathbf{b} - \mathbf{A} \cdot \mathbf{s} \bmod \mathbb{F}_q| = t$ where $\mathbf{s} = \mathbf{A}_\mathcal{I}^{-1} \cdot \mathbf{b}_\mathcal{I} \bmod \mathbb{F}_q$
5: **return** $(\mathbf{s}, \mathcal{I})$

Prange's algorithm terminates in expected time $\mathcal{G}/\mathcal{P}$, where $\mathcal{P}$ is the probability of collecting n error-free instances (i.e., $b_i = \langle \mathbf{a}_i, \mathbf{s} \rangle$) and $\mathcal{G}$ represents the time complexity of the Gaussian elimination step. Although the Gaussian elimination operation has a time complexity of only $O(n^\omega)$ for the linear algebra constant ω, where it is typically disregarded in the context of asymptotic analysis, its concrete time complexity is of significant consequence in estimating the bit-security. For instance, we consider the $(1024, 652, 57)$-LPN over $\mathbb{F}_q$. According to [49], the bit-size of the total cost associated with this problem is reported to be 111, with the Gaussian elimination accounting for a bit-size of 23 when we take $O(n^\omega) = n^{2.8}$. This highlights that the Gaussian elimination contributes to approximately one-fifth of the total complexity exponent involved in solving the problem. Consequently, to improve Prange's ISD, it is essential to reduce the cost of the Gaussian algorithm $\mathcal{G}$ since the probability of collecting n error-free samples cannot be changed.

Canteaut and Chabaud (CC) [23] and Bernstein, Lange and Peters (BLP) [13] proposed algorithms for solving LPN through reducing the cost of Gaussian elimination via so-called *reusing existing pivots* and *forcing more existing pivots*. In this work, we extend these algorithms by introducing the Reduce and Prange (RP) method, which consists of two key steps:

1. Guess N zero positions in $\mathbf{e}$ and transform from (m, n, t)-LPN instances to $(m - N, n - N, t)$-LPN instances.
2. Solve the $(m - N, n - N, t)$-LPN problem using Prange's ISD algorithm.

Since the secret dimension is reduced to $n - N$, the Gaussian elimination now operates on a smaller matrix of size $n - N$, which reduces its computational cost. This approach is performed iteratively, with the number of iterations being referred to as a *level*. As a result, the Reduce and Prange algorithms, which are defined by their level, are naturally defined. For example, when $k = 2$, then we first guess N_1 positions, to generate $(m - N_1, n - N_1, t)$-LPN instances. Then, we

additionally guess N_2 positions to get $(m-N_1-N_2, n-N_1-N_2, t)$-LPN instances. As the final step, we solve the LPN problem using Prange's ISD algorithm.

Advantages of Reduce and Prange's ISD. The RP algorithm lowers the per-iteration cost compared to Prange's original ISD method. While the original algorithm requires n^ω cost for each iteration due to performing Gaussian elimination on an n-dimensional matrix, the Reduce and Prange's algorithm strategically divides zero indices into sets, $\mathcal{I}_1$ and $\mathcal{I}_2$. If an error occurs in $\mathcal{I}_2$, only this subset is replaced rather than all n indices, reducing the iteration cost to $(n-N_1)^\omega$. Similarly, when $\mathcal{I}_1$ is incorrectly guessed, the Reduce and Prange's algorithm only repeats the Step 1 with a complexity $\mathcal{C}_{m,N_1}$. This approach significantly lowers the (concrete) iteration cost, while maintaining the overall success probability, as the algorithm still collects n zero indices.

Comparison to previous works [13,23]. We provide a conceptual comparison between previous works and our algorithm in terms of $\mathcal{G}$. The key idea behind the algorithms in [13,23] involves reusing the pivots from the previous iteration.

In a nutshell, suppose that $(\mathbf{A}, \mathbf{b})$ with $\mathbf{b} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e} \bmod \mathbb{F}_q$ is given. The BLP/CC algorithms [13, 23] transform this into a pair $(\mathbf{A}', \mathbf{b}') \in \mathbb{F}_q^{m-n\times(n+1)}$ where $\mathbf{b}' = \mathbf{A}' \cdot \mathbf{e}' + \mathbf{e}'' \bmod \mathbb{F}_q$ by reducing $\mathbf{s}$. If the partial noise vector $\mathbf{e}' \in \mathbb{Z}_q^n$ is zero, then $\mathbf{b}'$ has a Hamming weight of t, leading to the algorithm's termination. If not, some components of $\mathbf{e}'$ will be zero, allowing the reuse of the $n-c$ zero coordinates in $\mathbf{e}'$ and collection of c new samples to obtain n error-free instances.

When guessing that $n-c$ coordinates of $\mathbf{e}'$ are zero, the corresponding columns of $\mathbf{A}'$ are not needed. As a result, the pair $(\mathbf{A}', \mathbf{b}')$ becomes $(\mathbf{A}'', \mathbf{b}') \in \mathbb{Z}_q^{m-n\times(c+1)}$. Since Gaussian elimination is required over a smaller dimension $c < n$, the overall complexity is reduced compared to Prange's algorithm. This optimization can be seen as collecting error-free instances from two separate lists, one of size $n-c$ and the other of size c.

Our RP algorithm starts from an extension of the BLP/CC observation: In the perspective, we aim at collecting n error-free instances using 'multi' separated lists $(N_1, N_2, \ldots, N_k)$ such that $\sum_{i=1}^k N_i = n$, where k is called a level. Given the multi-size set, the RP collects N_1 error-free instances and converts (m, n, t)-LPN instances into (m_1, n_1, t)-LPN instances with $m_1 = m-N_1, n_1 = n-N_1$ by reducing the secret vector from the N_1 error-free instances. This technique can be iteratively applied for multiple rounds with the given set. The time complexity of each iteration is then less than that of $\mathcal{G}$. By selecting $\{N_i\}_{i=1}^k$ properly, it improves the overall performance of the algorithm. For the detailed algorithmic description, we refer Section 2.

2 Reduce and Prange Technique

This section introduces a new approach for solving LPN over large fields, simply called the Reduce and Prange. For the algorithm description, we suppose that LPN instances of the form $(\mathbf{A}, \mathbf{b} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e} \bmod \mathbb{F}_q)$ with $\mathbf{A} \in \mathbb{F}_q^{m\times n}, \mathbf{s} \in \mathbb{F}_q^n$ and $\mathbf{e} \leftarrow \chi_{t,m}$ are given.

The probability of obtaining n error-free polynomials is given by $P = \frac{\binom{m-n}{t}}{\binom{m}{t}}$. Thus, the expected number of iterations required for Prange's ISD (Algorithm 1) to recover the secret vector is $\frac{1}{P} = \frac{\binom{m}{t}}{\binom{m-n}{t}}$. Hence, the time complexity of Prange's ISD algorithm is given by

$$O\left(\frac{\binom{m}{t}}{\binom{m-n}{t}} \cdot n^{\omega}\right)$$

where n^{ω} represents the Gaussian elimination cost, with the linear algebra constant $2 \leq \omega \leq 3$.

2.1 Reduce and Prange's ISD

The core idea of our approach is to first guess N_1 error-free samples to reduce the dimension of the LPN problem. We then apply Prange's algorithm to the LPN of $n - N_1$ dimensions.

Suppose that LPN instances of the form $(\mathbf{A}, \mathbf{b} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e} \bmod \mathbb{F}_q)$ with $\mathbf{A} \in \mathbb{F}_q^{m \times n}, \mathbf{s} \in \mathbb{F}_q^n$ and $\mathbf{e} \leftarrow \chi_{t,m}$. We let $\mathbf{a}_i$ (Resp. b_i and e_i) denote the i-th vector of $\mathbf{A}$ (Resp. $\mathbf{b}$ and $\mathbf{e}$).

Step 1. Choose an index set $\mathcal{I}_1 \subset [m]$ of the size $N_1 (< n)$, which corresponds to the zero positions in $\mathbf{e}$. For simplicity, assume that the last N_1 positions in $\mathbf{e}$ are zero. In general, the zero positions can be shifted to the end through permutation. Let $\mathbf{A}_{\mathsf{top}}$ and $\mathbf{A}_{\mathsf{bot}}$ be a submatrix of $\mathbf{A}$ which consists of the first $m - N_1$ rows of $\mathbf{A}$ and the last N_1 rows, respectively. Analogously to the notation, we define $\mathbf{b}_{\mathsf{top}}$, $\mathbf{b}_{\mathsf{bot}}$, $\mathbf{e}_{\mathsf{top}}$, and $\mathbf{e}_{\mathsf{bot}}$. The above assumption then says that $\mathbf{e}_{\mathsf{bot}} = \mathbf{0}$. Using the notation, we express the system as:

$$\begin{pmatrix} \mathbf{b}_{\mathsf{top}} \\ \mathbf{b}_{\mathsf{bot}} \end{pmatrix} = \begin{pmatrix} \mathbf{A}_{\mathsf{top}} \\ \mathbf{A}_{\mathsf{bot}} \end{pmatrix} \cdot \mathbf{s} + \begin{pmatrix} \mathbf{e}_{\mathsf{top}} \\ \mathbf{0} \end{pmatrix} \bmod \mathbb{F}_q$$

We then apply the Gaussian elimination algorithm to obtain LPN of less dimension. For this purpose, We parse $\mathbf{s}$ to $(\mathbf{s}^1 \| \mathbf{s}^2)$ such that $\mathbf{s}^1 \in \mathbb{F}_q^{n-N_1}$ and $\mathbf{s}^2 \in \mathbb{F}_q^{N_1}$. Similarly to the $\mathbf{s}$, we split $\mathbf{A}_{\mathsf{top}}$ (Resp. $\mathbf{A}_{\mathsf{bot}}$) into $(\mathbf{A}_{\mathsf{top}}^{\mathsf{left}} \| \mathbf{A}_{\mathsf{top}}^{\mathsf{right}})$ (Resp. $(\mathbf{A}_{\mathsf{bot}}^{\mathsf{left}} \| \mathbf{A}_{\mathsf{bot}}^{\mathsf{right}})$) as well. Assuming that the matrix $\mathbf{A}_{\mathsf{bot}}^{\mathsf{right}}$ is invertible over $\mathbb{F}_q$, (If not, we change the order of columns until $\mathbf{A}_{\mathsf{bot}}^{\mathsf{right}}$ is invertible), one can get the following LPN samples:

$$\begin{aligned}
\mathbf{b}' :&= \left(\mathbf{I}_{m-N_1} \| -\mathbf{A}_{\mathsf{top}}^{\mathsf{right}} \cdot (\mathbf{A}_{\mathsf{bot}}^{\mathsf{right}})^{-1}\right) \cdot \begin{pmatrix} \mathbf{b}_{\mathsf{top}} \\ \mathbf{b}_{\mathsf{bot}} \end{pmatrix} \\
&= \left(\mathbf{I}_{m-N_1} \| -\mathbf{A}_{\mathsf{top}}^{\mathsf{right}} \cdot (\mathbf{A}_{\mathsf{bot}}^{\mathsf{right}})^{-1}\right) \cdot \left(\begin{pmatrix} \mathbf{A}_{\mathsf{top}} \\ \mathbf{A}_{\mathsf{bot}} \end{pmatrix} \cdot \mathbf{s} + \begin{pmatrix} \mathbf{e}_{\mathsf{top}} \\ \mathbf{0} \end{pmatrix}\right) \bmod \mathbb{F}_q \\
&= \underbrace{(\mathbf{A}_{\mathsf{top}}^{\mathsf{left}} - \mathbf{A}_{\mathsf{top}}^{\mathsf{right}} \cdot (\mathbf{A}_{\mathsf{bot}}^{\mathsf{right}})^{-1} \cdot \mathbf{A}_{\mathsf{bot}}^{\mathsf{left}})}_{=:\mathbf{A}'} \cdot \mathbf{s}^1 + \mathbf{e}_{\mathsf{top}} \bmod \mathbb{F}_q.
\end{aligned}$$

We denote it as $(\mathbf{A}', \mathbf{b}') \leftarrow \mathsf{Conv}((\mathbf{A}, \mathbf{b}), \mathcal{I}_1)$ since this conversion technique is employed several times.

Complexity. In each guessing, this algorithm computes the matrix $\mathbf{A}'$, which consists of one matrix inversion and two matrix multiplications so it takes in time $\mathcal{C}_{m,N_1} := O(N_1^\omega + N_1^2 \cdot (m - N_1) + N_1 \cdot (m - N_1) \cdot (n - N_1))$, where ω is the linear algebra constant. The probability that attackers can correctly guess N_1 positions is $\mathcal{P}_{m,t,N_1} := \frac{\binom{m-t}{N_1}}{\binom{m}{N_1}}$, which directly implies that the time complexity to get LPN samples of dimension $n - N_1$ is $\frac{1}{\mathcal{P}_{m,t,N_1}} \cdot \mathcal{C}_{m,N_1}$.

Step 2. For ease of representation, we assume that $(m - N_1, n - N_1, t)$-LPN instances $(\mathbf{A}', \mathbf{b}')$ such that $\mathbf{b}' = \mathbf{A}' \cdot \mathbf{s}^1 + \mathbf{e}_{\mathsf{top}} \bmod \mathbb{F}_q$ are given, where $(\mathbf{A}', \mathbf{b}') \leftarrow \mathsf{Conv}(\mathbf{A}, \mathbf{b}), \mathcal{I}_1), \mathbf{s}^1 \in \mathbb{F}_q^{n-N_1}$ and $\mathbf{e}_{\mathsf{top}} \in \chi_{t,m-N_1}$. In addition, the index of $\mathbf{e}_{\mathsf{top}}$ is given as $[m] \setminus \mathcal{I}_1$.

The second step just runs Prange's algorithm. That is, one first collects an index set $\mathcal{I}_2 \subset [m] \setminus \mathcal{I}_1$ of size $n - N_1$ from given $m - N_1$ LPN instances. Then, one can recover the secret $\mathbf{s}^1$ via the Gaussian elimination algorithm. After recovering $\mathbf{s}^1$, one can recover the vector $\mathbf{s}^2$ from the identity $\mathbf{b}_{\mathsf{bot}} = \mathbf{A}_{\mathsf{bot}} \cdot (\mathbf{s}^1 \| \mathbf{s}^2)$.

Algorithm 2 Simple Reduce and Prange's ISD

Input: (m, n, t)-LPN over $\mathbb{F}_q$ instances $(\mathbf{A}, \mathbf{b}) \in \mathbb{F}_q^{m \times (n+1)}$, and an index size $\{N_1, N_2 := n - N_1\}$.
Output: The secret vector $\mathbf{s}$ such that $|\mathbf{b} - \mathbf{A} \cdot \mathbf{s} \bmod \mathbb{F}_q| = t$.

Compute $P_1 = \frac{\binom{m-t}{N_1}}{\binom{m}{N_1}}$ and $P_2 = \frac{\binom{m-N_1-t}{N_2}}{\binom{m-N_1}{N_2}}$
for $i = 1$ up to $\frac{1}{P_1}$ **do**
  Choose a random index set $\mathcal{I}_1$ of size N_1
  Compute $(\mathbf{A}^1, \mathbf{b}^1) \leftarrow \mathsf{Conv}((\mathbf{A}, \mathbf{b}), \mathcal{I}_1)$
  for $j = 1$ up to $\frac{1}{P_2}$ **do**
    Choose a random index set $\mathcal{I}_2 \subset [m] \setminus \mathcal{I}_1$ of size N_2
    $(\mathbf{A}^1_{\mathcal{I}_2}, \mathbf{b}^1_{\mathcal{I}_2}) \leftarrow [\mathbf{A}^1, \mathbf{b}^1]_{\mathcal{I}_2}$, $\mathbf{s}^1 = (\mathbf{A}^1_{\mathcal{I}_2})^{-1} \cdot \mathbf{b}^1_{\mathcal{I}_2}$
    if $|\mathbf{b}^1 - \mathbf{A}^1 \cdot \mathbf{s}^1 \bmod F_q| = t$ **then**
      break
    end if
  end for
end for
$\mathcal{I} \leftarrow \mathcal{I}_1 \cup \mathcal{I}_2$,$(\mathbf{A}_{\mathcal{I}}, \mathbf{b}_{\mathcal{I}}) \leftarrow [\mathbf{A}, \mathbf{b}]_{\mathcal{I}}$, and $\mathbf{s} \leftarrow \mathbf{A}_{\mathcal{I}}^{-1} \cdot \mathbf{b}_{\mathcal{I}}$
return $(\mathbf{s}, \mathcal{I})$

Correctness and Verification. Suppose that $\mathcal{I}_1$ and $\mathcal{I}_2$ are correctly guessed and $\mathbf{s}_1$ and $\mathbf{s}_2$ be an output vector. Then the vector $(\mathbf{s}_1 \| \mathbf{s}_2)$ should be $\mathbf{s}$ and satisfy $|\mathbf{b} - \mathbf{A} \cdot \mathbf{s}| = t$.

To ensure the correctness, we consider the case where either $\mathcal{I}_1$ or $\mathcal{I}_2$ is incorrectly guessed. In the RP algorithm, $\mathbf{s}^1$ is determined using *Algorithm* 1$((\mathbf{A}, \mathbf{b}), t)$,

which outputs a solution only if one exists. We show that no solution is produced when either $\mathcal{I}_1$ or $\mathcal{I}_2$ is incorrectly guessed.

If $\mathcal{I}_1$ is incorrect, the following equation holds:

$$\mathbf{b}' = \mathbf{A}' \cdot \mathbf{s}^1 + \mathbf{e}_{\mathsf{top}} + \mathbf{A}^\dagger \cdot \mathbf{s}^2,$$

where $\mathbf{A}^\dagger$ is properly defined to satisfy the equation. The presence of the term $\mathbf{A}^\dagger$ obscures the recovery of $\mathbf{s}^1$. In other words, for any $\mathbf{s}^\dagger \neq \mathbf{s}^1$ it is impossible to satisfy $|\mathbf{b}' - \mathbf{A}' \cdot \mathbf{s}^\dagger| = t$. Thus, Prange's algorithm, a subroutine of RP, does not return any vector, as explained in Algorithm 1.

Similarly, in case $\mathcal{I}_1$ is correct but $\mathcal{I}_2$ is incorrect, $\mathbf{s}^1$ cannot be recovered from the $(m - N_1, n - N_1, t)$-LPN instance $(\mathbf{A}', \mathbf{b}')$.

Hence, the RP algorithm successfully outputs $\mathbf{s}_1$ only when both $(\mathcal{I}_1, \mathcal{I}_2)$ are correctly guessed. Consequently, one can automatically verify the solution $\mathbf{s}$ by checking the RP algorithm produces solution, which implies that no additional cost is required to confirm the correctness of the solution.

Time Complexity. The complexity of Prange's ISD for solving LPN with parameters $(m - N_1, n - N_1, t)$ can be computed as follows.

$$\begin{aligned}\mathcal{S}_{m-N_1,n-N_1,t} &\stackrel{def}{=} \frac{\binom{m-N_1}{n-N_1}}{\binom{m-N_1-t}{n-N_1}} \cdot (n-N_1)^\omega \\ &= \frac{1}{\mathcal{P}_{m-N_1,t,n-N_1}} \cdot \left((n-N_1)^\omega + (m-N_1)\cdot(n-N_1)\right),\end{aligned}$$

where the linear algebra constant ω. The last term, $(m - N_1) \cdot (n - N_1)$, represents the verification cost of verifying whether $|\mathbf{b}^1 - \mathbf{A}^1 \cdot \mathbf{s}^1| = t$. Although this verification cost is typically negligible in Prange's original ISD, we include it here for an accurate computation. We emphasize that this verification cost can be large, depending on the size of N_1.

We note that given the probability p, performing the p repetition gives a success probability $(1-1/e)$. In the RP case, the factor $(1-1/e)^2$ appears, since the algorithm correctly guesses $\mathcal{I}_1$ and $\mathcal{I}_2$ both.

Putting them together, the LPN problem is solved within

$$\frac{1}{\mathcal{P}_{m,t,N_1}} \cdot (\mathcal{C}_{m,N_1} + \mathcal{S}_{m-N_1,n-N_1,t})$$

with a success probability $(1 - 1/e)^2$. Hence, its expected time complexity is represented by

$$\frac{1}{\mathcal{P}_{m,t,N_1} \cdot (1-1/e)^2} \cdot (\mathcal{C}_{m,N_1} + \mathcal{S}_{m-N_1,n-N_1,t}).$$

Impact of the Reduce and Prange's ISD. We present the advantages of the Reduce and Prange's ISD algorithm. The key improvement lies in reducing the

computational cost per iteration. In Prange's original ISD algorithm, each iteration requires Gaussian elimination on an n-dimensional square matrix, incurring a cost of n^ω.

The RP algorithm significantly reduces this per-iteration cost. The algorithm partitions the zero indices into two sets, $\mathcal{I}_1$ and $\mathcal{I}_2$, such that $\#\mathcal{I}_1+\#\mathcal{I}_2 = n$ with $\#\mathcal{I}_1 = N_1$. When an incorrect zero index is found in $\mathcal{I}_2$, the RP algorithm only needs to replace $\mathcal{I}_2$ (Step 2), rather than all n zero indices. This optimization reduces the per-iteration cost to $(n-N_1)^\omega + (m-N_1)\cdot(n-N_1)$. Similarly, if $\mathcal{I}_1$ is incorrectly guessed, the algorithm repeats Step 1 with complexity $\mathcal{C}_{m,N_1}$.

While the RP algorithm maintains the same overall success probability as Prange's ISD (since it still requires finding n correct zero indices), it achieves significant efficiency gains through reduced per-iteration costs.

Indeed, the time complexity of RP is dominated by

$$\frac{\mathcal{S}_{m-N_1,n-N_1,t}}{\mathcal{P}_{m,t,N_1}\cdot(1-1/e)^2}\cdot = \frac{\binom{m}{n}}{\binom{m-t}{n}}\cdot\frac{(n-N_1)^\omega+(m-N_1)\cdot(n-N_1)}{(1-1/e)^2}.$$

Compared to the original Prange's ISD algorithm, which has a complexity $\frac{\binom{m-t}{n}}{\binom{m}{n}}\cdot n^\omega$, our Reduce and Prange (RP) algorithm shares the fractional part but features a smaller multiplicative factor $\frac{(n-N_1)^\omega+(m-N_1)\cdot(n-N_1)}{(1-1/e)^2}$. As a consequence, we expect this reduction in the multiplier to lower the overall complexity of the algorithm.

2.2 Iterative Reduce and Prange Technique

'Step 1' in Section 2 can be viewed as a self-reduction technique for LPN. Naturally, this approach can be iterated to progressively reduce the dimension of LPN samples. Thereafter, one can solve the reduced LPN problems by Step 2. With respect to the number of reductions, we call this algorithm the level-k Reduce and Prange algorithm.

For instance, consider $k = 2$. The level-2 Reduce and Prange algorithm proceeds as follows: First, select three index sets $\{\mathcal{I}_1,\mathcal{I}_2,\mathcal{I}_3\}\subset[m]$ of size $\#\mathcal{I}_i = N_i$ and $\sum_{i=1}^3 N_i = n$. Then, one can get LPN samples $(\mathbf{A}^{(2)},\mathbf{b}^{(2)})$ of dimension N_3 from $(\mathbf{A}^{(2)},\mathbf{b}^{(2)}) \leftarrow \mathsf{Conv}((\mathbf{A}^{(1)},\mathbf{b}^{(1)}),\mathcal{I}_2)$ and $(\mathbf{A}^{(1)},\mathbf{b}^{(1)}) \leftarrow \mathsf{Conv}((\mathbf{A},\mathbf{b}),\mathcal{I}_1)$. Then it can be solved via the Prange algorithm on the less dimension N_3. The interesting fact is that since this auxiliary step proceeds after the first step occurs, the complexity of the level-2 Reduce and Prange algorithm is given as

$$\frac{1}{\mathcal{P}_{m,t,N_1}}\cdot\left(\mathcal{C}_{m,N_1}+\frac{1}{\mathcal{P}_{m-N_1,t,N_2}}\left(\mathcal{C}_{m-N_1,N_2}+\mathcal{S}_{m-N_1-N_2,N_3,t}\right)\right).$$

However, as previously mentioned, we must also account for the probability of successfully guessing the three sets $\mathcal{I}_1,\mathcal{I}_2$, and $\mathcal{I}_3$. Consequently, the overall complexity of the level-2 Reduce and Prange algorithm is more accurately

represented as

$$\frac{1}{\mathcal{P}_{m,t,N_1}\cdot(1-1/e)^3}\cdot\left(\mathcal{C}_{m,N_1}+\frac{1}{\mathcal{P}_{m-N_1,t,N_2}}\left(\mathcal{C}_{m-N_1,N_2}+\mathcal{S}_{m-N_1-N_2,N_3,t}\right)\right).$$

In general, the time complexity for level-k Reduce and Prange algorithm can be computed as follows. Given a set $\{N_i\}_{i=1}^{k+1}$ and sample an index set $\{\mathcal{I}_i\}_{i=1}^{k+1}\subset[m]$ such that $\#\mathcal{I}_i=N_i$ and $\sum_{i=1}^{k+1}N_i=n$. To simplify notation, we introduce the following terms $\mathcal{P}(j),\mathcal{C}(j),\mathcal{S}$ as follows:

$$\begin{aligned}\mathcal{P}(j)&=\mathcal{P}_{m-\mu_{j-1},t,N_j}\\ \mathcal{C}(j)&=\mathcal{C}_{m-\mu_{j-1},N_j}\\ \mathcal{S}&=\frac{1}{\mathcal{P}_{m-\mu_k,t,n-\mu_k}}\cdot\left((n-\mu_k)^{\omega}+(n-\mu_k)\cdot(m-\mu_k)\right),\end{aligned}$$

where $\mu_j=\sum_{i=1}^{j}N_i$ for every $j\geq 1$. Then, using this notation, the total cost of our algorithm is

$$\mathcal{T}=\frac{1}{\mathcal{P}(1)}\left(\mathcal{C}(1)+\frac{1}{\mathcal{P}(2)}\left(\mathcal{C}(2)+\cdots+\frac{1}{\mathcal{P}(k)}\left(\mathcal{C}(k)+\mathcal{S}\right)\right)\right)$$

To simplify it, we also define $\mathcal{Q}(j)$ as $\mathcal{Q}(j)=\prod_{i=1}^{j}\mathcal{P}(i)$. We then demonstrate that level-k RP algorithm terminates in

$$\sum_{j=1}^{k}\frac{\mathcal{C}(j)}{\mathcal{Q}(j)}+\frac{\mathcal{S}}{\mathcal{Q}(k)}.$$

Its success probability relies on the assumption that the index sequence $\{\mathcal{I}_i\}_{i=1}^{k}$ is correctly guessed. Since each $\mathcal{I}_i$ is correct with probability $\mathcal{P}(i)$, the overall success probability is $(1-1/e)^{k+1}$. Accordingly, the total time complexity of the k-level RP algorithm is given by

$$\mathcal{T}=\sum_{j=1}^{k}\frac{\mathcal{C}(j)}{\mathcal{Q}_{\mathsf{correct}}(j)}+\frac{\mathcal{S}}{\mathcal{Q}_{\mathsf{correct}}(k)},$$

where $\mathcal{Q}_{\mathsf{correct}}(j)=\left(\prod_{i=1}^{j}\mathcal{P}(i)\right)\cdot(1-1/e)^{k+1}$.

Given a sequence $\{N_i\}_{i=1}^{k+1}$ as input, the RP algorithm proceeds as Algorithm 3. An optimized construction of $\{N_i\}_{i=1}^{k+1}$ to minimize the algorithm's complexity will be discussed later.

Correctness and Verification. Following the arguments in Section 2.1, the RP algorithm successfully outputs a solution only when $(\mathcal{I}_i)_{i=1}^{k+1}$ are correctly guessed. Hence, the output RP algorithm is identical to $\mathbf{s}$. Furthermore, RP does not require additional verification steps and costs.

Algorithm 3 Reduce and Prange's ISD

Input: (m, n, t)-LPN over $\mathbb{F}_q$ instances $(\mathbf{A}, \mathbf{b}) \in \mathbb{F}_q^{m\times(n+1)}$,
an index set $\{N_i\}_{i=1}^k$ such that $\sum_{i=1}^k N_i = n$,
current level i,
an index set $\mathcal{I} = \cup_{t=0}^{i-1}\mathcal{I}_i$.
Output: The secret vector $\mathbf{s}$ such that $|\mathbf{b} - \mathbf{A}\cdot\mathbf{s} \bmod \mathbb{F}_q| = t$.

Compute $P_1 = \frac{\binom{m-t}{N_1}}{\binom{m}{N_1}}$ and $P_i = \frac{\binom{m-\sum_{j=1}^{i-1} N_j - t}{N_i}}{\binom{m-\sum_{j=1}^{i-1} N_j}{N_i}}$ for $2 \le i \le k$
if $i < k$ **then**
  for $j = 1$ up to $\frac{1}{P_i}$ **do**
    Choose a random index set $\mathcal{I}_i \subset [m] \setminus \mathcal{I}$ of size N_i
    Compute $(\mathbf{A}, \mathbf{b}) \leftarrow \mathsf{Conv}((\mathbf{A}, \mathbf{b}), \mathcal{I}_i)$
    Update $\mathcal{I} \leftarrow \mathcal{I} \cup \mathcal{I}_i$
    $i = i + 1$
    Call Algorithm 1$(m, n, t, (\mathbf{A}, \mathbf{b}), \{N_i\}_{i=1}^k, i, \mathcal{I})$
  end for
end if
if $i = k$ **then**
  Choose a random index set $\mathcal{I}_k \subset [m] \setminus \mathcal{I}$ of size N_k
  $(\mathbf{A}_{\mathcal{I}_k}, \mathbf{b}_{\mathcal{I}_k}) \leftarrow [\mathbf{A}, \mathbf{b}]_{\mathcal{I}_k}$, $\mathbf{s}_{\mathcal{I}_k} = (\mathbf{A}_{\mathcal{I}_k})^{-1} \cdot \mathbf{b}_{\mathcal{I}_k}$
  if $|\mathbf{b}_{\mathcal{I}_k} - \mathbf{A}_{\mathcal{I}_k} \cdot \mathbf{s}_{\mathcal{I}_k} \bmod F_q| = t$ **then**
    break
  end if
  $\mathcal{I} \leftarrow \mathcal{I} \cup \mathcal{I}_k$,$(\mathbf{A}_{\mathcal{I}}, \mathbf{b}_{\mathcal{I}}) \leftarrow [\mathbf{A}, \mathbf{b}]_{\mathcal{I}}$, and $\mathbf{s} \leftarrow \mathbf{A}_{\mathcal{I}}^{-1} \cdot \mathbf{b}_{\mathcal{I}}$
  return $(\mathbf{s}, \mathcal{I})$
end if

Optimal index size. To (approximately) optimize the cost, we observe $\mathcal{T}$ as two perspectives. First of all, for every j, we observe

$$\mathcal{Q}_{\mathsf{correct}}(j) = \frac{\binom{m-t}{\mu_j}}{\binom{m}{\mu_j}} \cdot \frac{1}{\left(\prod_{i=1}^{j} \mathcal{P}(i)\right) \cdot (1-1/e)^{k+1}}$$

where $\mu_j = \sum_{i=1}^{j} N_i$. From the observation, $\frac{\mathcal{C}(1)}{\mathcal{Q}(1)}$ is only determined by the N_1 when $j = 1$. Inductively, when the $\sum_{i=1}^{j-1} N_i$ is fixed once, the ratio $\frac{\mathcal{C}(j)}{\mathcal{Q}(j)}$ is only determined by the N_j.

Next, we also have

$$\frac{\mathcal{S}}{\mathcal{Q}_{\mathsf{correct}}(k)} = \frac{\binom{m}{\mu_j}}{\binom{m-t}{\mu_j}} \cdot \mathcal{S} = \frac{\binom{m}{\mu_j}}{\binom{m-t}{\mu_j}} \cdot \frac{(n-\mu_k)^\omega}{\mathcal{P}_{m-\mu_k,t,n-\mu_k}} \cdot \left(\prod_{i=1}^{k} \mathcal{P}(i)\right) \cdot (1-1/e)^{k+1}$$

where $\mu_j = \sum_{i=1}^{j} N_j$ for each j. Therefore, it satisfies that

$$\frac{\mathcal{S}}{\mathcal{Q}_{\mathsf{correct}}(k)} = (n-\mu_k)^\omega \cdot \frac{\binom{m}{n}}{\binom{m-t}{n}} \cdot \left(\prod_{i=1}^{k} \mathcal{P}(i)\right) \cdot (1-1/e)^{k+1}$$

Since the overall cost of a positive addition will be optimized when all values are similar, we let the threshold be $\frac{\binom{m-t}{n}}{\binom{m}{n}} \cdot \Delta$ for some integer $\Delta \in \mathbb{Z}$ and choose the largest integer N_i such that each $\frac{\mathcal{C}(i)}{\mathcal{Q}_{\mathsf{correct}}(i)}$ term is less than the threshold, starting at $i = 1$. If there is no integer N_j for an index j, we let $N_j = n - \mu_{j-1}$ and operate the Prange's algorithm of LPN of dimension N_j. Then the total cost is given as $\sum_{i=1}^{j-1} \frac{\mathcal{C}(j)}{\mathcal{Q}_{\mathsf{correct}}(j)} + \frac{\binom{m}{n}}{\binom{m-t}{n}}(N_j^{\omega} + (m - \mu_{j-1}) \cdot N_j)$.

We evaluate all Δ values up to $n^{2.8}$ to determine the optimal cost. The full algorithm for estimating the Reduce and Prange will be given by Algorithm 4.

Algorithm 4 RP Estimation

Input: LPN parameters (m, n, t)
Output: The total cost of the Reduce and Prange algorithm.
Set $\mathsf{T} = \frac{\binom{m}{n}}{\binom{m-t}{n}}$ // T is a threshold.
Set $Cost = \mathsf{T} \cdot n^{2.8}$
for $\Delta = 1$ up to $n^{2.8}$ **do**
  Set $k = 1$
  while $\exists N_k$ such that $\frac{\mathcal{C}(k)}{\mathcal{Q}_{\mathsf{correct}}(k)} < \Delta \cdot \mathsf{T}$ **do**
    Set largest N_k such that $\frac{\mathcal{C}(k)}{\mathcal{Q}_{\mathsf{correct}}(k)} < \Delta \cdot \mathsf{T}$
    Set $k = k + 1$
  end while
  Set $N_{k+1} = n - \sum_{i=1}^{k} N_i$
  $Cost = \min\left\{Cost, \sum_{i=1}^{k} \frac{\mathcal{C}(i)}{\mathcal{Q}_{\mathsf{correct}}(i)} + \mathsf{T} \cdot (N_{k+1}^{2.8} + (m - \mu_k) \cdot N_{k+1})\right\}$
end for
Return $Cost$ and $\{N_i\}_{i=1}^{k+1}$

2.3 Graphical Results and Its Discussion

This section presents graphical results illustrating the performance improvements of the RP algorithm across different parameter sizes. To analyze the asymptotic behavior, we define $C = \frac{\log T_{\mathsf{RP}}}{\log T_{\mathsf{Prange}}}$, where T_{RP} and T_{Prange} are the time complexities of the algorithms. It is obvious that $C = 1$ when we exploit the original Prange's ISD. If C is strictly less than 1, then one concludes that RP outperforms Prange's ISD.

We analyze the parameter sets $(m, n, t) = (2^{10}, 652, 106)$ and $(m, n, t) = (2^{12}, 1589, 172)$, plotting the changes in C while varying a single parameter at a time. While we observe a general trend where C approaches 1 as each parameter increases (as shown across all figures), intriguingly, focusing on parameter m reveals a specific range where C noticeably decreases, reaching a local minimum (Figure 1, Figure 2). However, we have not yet identified a closed-form expression or formula that precisely characterizes the optimal value of m that minimizes

C. Based on our experiments (Section 5.2), we observe that C falls within the range $[0.94, 0.98]$.

Approximating C closely to the observed values aids in estimating RP's asymptotic complexity. We unfortunately fail to find the closeness of C. Nevertheless, we find that graphical visualization effectively highlights RP's performance improvements.

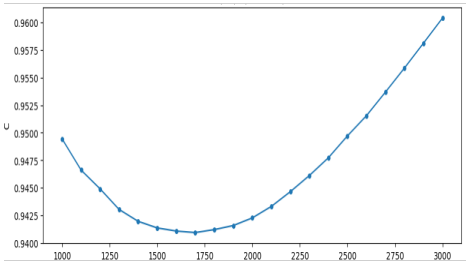

Fig. 1. Effect of increasing m on C with a fixed $(n, t) = (652, 106)$

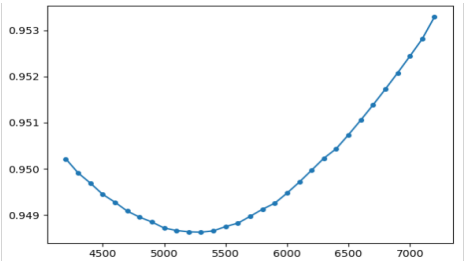

Fig. 2. Effect of increasing m on C with a fixed $(n, t) = (1589, 172)$

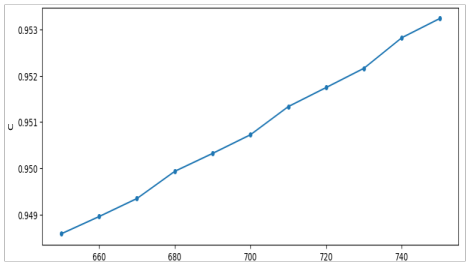

Fig. 3. Effect of increasing n on C with a fixed $(m, t) = (2^{10}, 106)$

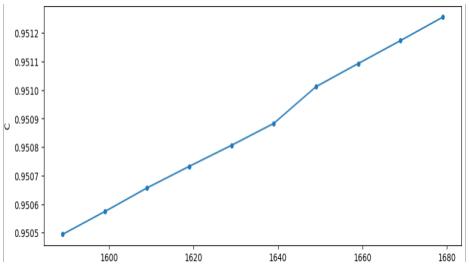

Fig. 4. Effect of increasing n on C with a fixed $(m, t) = (2^{12}, 172)$

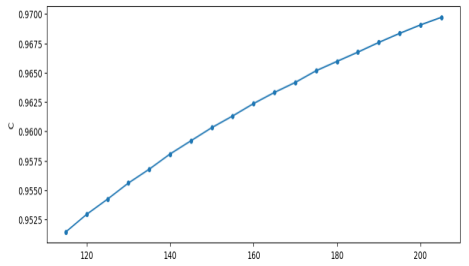

Fig. 5. Effect of increasing t on C with a fixed $(m, n) = (2^{10}, 652)$

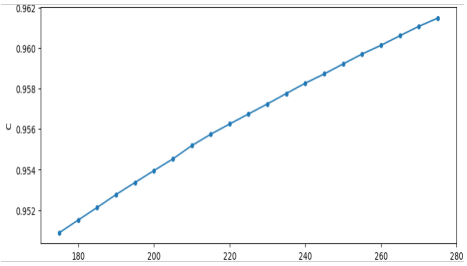

Fig. 6. Effect of increasing t on C with a fixed $(m, n) = (2^{12}, 1589)$

3 Regular-RP: Application to regular-LPN

This section presents the application of RP to LPN with regular noise, which is equivalent to the regular syndrome decoding (RSD) problem. Throughout this section, we set $\mathcal{R} = \mathbb{F}_q$ with $q \geq 2$.

We first introduce a definition of RSD, originated [8]. The problem has a lot of applications in PGC-like protocols [9, 18, 20, 22, 28, 31, 40, 59–61, 65, 69].

Problem 3 (Regular Syndrome Decoding over $\mathbb{F}_q$). Let m, n, t, β be positive integers with $m = t \cdot \beta$ and $\mathbb{F}_q$ be a finite field. Sample a full rank matrix $\mathbf{H} \leftarrow \mathbb{F}_q^{(m-n)\times m}$ and a column vector $\mathbf{e} := (\mathbf{e}_1 \| \mathbf{e}_2 \| \dots \| \mathbf{e}_t) \leftarrow \mathbb{F}_q^m$ such that $\mathbf{e}_i \in \mathbb{F}_q^\beta$ is of Hamming weight $|\mathbf{e}_i| = 1$ for $1 \leq i \leq t$. Given $(\mathbf{H}, \mathbf{y} = \mathbf{H} \cdot \mathbf{e})$, recover the error $\mathbf{e}$.

3.1 Algorithm for Regular-LPN over Large Fields

Esser and Santini [38] proposed algorithms for solving regular-LPN by leveraging the structure of regular noise to identify error-free instances. We integrate this approach into our algorithm to refine security estimations for regular-LPN over large fields.

This approach optimizes the guessing probability in the RP algorithm to solve regular-LPN more efficiently in large fields, which we refer to as *regular-RP*. Except for the probability, the strategy of RP is used.

Regular-RP Algorithm. As the standard RP, suppose that $\{N_i\}_{i=1}^{k+1}$ is given and select index sets $\{\mathcal{I}_i\}_{i=1}^{k+1}$ such that $\#\mathcal{I}_i = N_i$ and $\sum_{i=1}^{k+1} N_i = n$. In the first step, we select $\mathcal{I}_1$ and compute $(\mathbf{A}', \mathbf{b}') \leftarrow \mathsf{Conv}((\mathbf{A}, \mathbf{b}), \mathcal{I}_1)$. We now focus on how to sample $\mathcal{I}_1$ to leverage the regular constraint. Here, we set N_i as a multiple of t for a simple description.

Since each $\mathbf{e}_j$ has exactly one nonzero element, the probability of guessing a zero coordinate of $\mathbf{e}_1$ equals to $\frac{\beta-1}{\beta}$. To guess the second zero position, one considers two cases:

- E_1: Find the zero coordinate in $\mathbf{e}_1$.
- E_2: Find the zero coordinate in $\mathbf{e}_j$ for $j \neq 1$.

The probability of event E_1 is $\frac{\beta-2}{\beta-1}$, whereas the probability of event E_2 is $\frac{\beta-1}{\beta}$. Thus, the optimal strategy for selecting $\mathcal{I}_1$ is to choose N_1/t-zero coordinates from each $\mathbf{e}_j$. The probability of correctly selecting all N_1 zero positions is given by:

$$\mathcal{P}^R_{m,t,N_1} = \left(1 - \frac{N_1/t}{\beta}\right)^t = \left(1 - \frac{N_1}{m}\right)^t.$$

Similarly, when sampling $\mathcal{I}_i$, guessing N_i/t zero coordinates in each $\mathbf{e}_j$ yields an optimal probability. To describe it formally, we denote a coordinate set of $\mathbf{e}_j$ as $[\mathbf{e}_j]$ and parse $\mathcal{I}_i = \cup_{j=1}^t \mathcal{I}_{ij}$ such that $\mathcal{I}_{ij} \subset [\mathbf{e}_j] \backslash \cup_{\ell=1}^{i-1} \mathcal{I}_{\ell j}$ of size $\#\mathcal{I}_{ij} = N_i/t$. We

define the RP algorithm with this modified sampling technique as the regular-RP algorithm, detailed in Algorithm 5.

Algorithm 5 Regular-RP

Input: (m, n, t)-LPN over $\mathbb{F}_q$ instances $(\mathbf{A}, \mathbf{b}) \in \mathbb{F}_q^{m\times(n+1)}$,
an index set $\{N_i\}_{i=1}^k$ such that $\sum_{i=1}^k N_i = n$,
current level i,
an index set $\mathcal{I} = \cup_{t=0}^{i-1}\mathcal{I}_i$.
Output: The secret vector $\mathbf{s}$ such that $|\mathbf{b} - \mathbf{A}\cdot\mathbf{s} \bmod \mathbb{F}_q| = t$.

Compute $P_1^R = \left(1 - \frac{N_1}{m}\right)^t$ and $P_i^R = \left(\frac{m-\sum_{j=1}^{i} N_j}{m-\sum_{j=1}^{i-1} N_j}\right)^t$ for $2 \le i \le k$
if $i < k$ **then**
  for $j = 1$ up to $\frac{1}{P_i^R}$ **do**
    Choose a random index set $\mathcal{I}_{i\ell} \subset [m] \setminus \mathcal{I}$ of size N_i/t for $1 \le \ell \le t$
    Update $\mathcal{I}_i \leftarrow \cup_\ell \mathcal{I}_{i\ell}, \mathcal{I} \leftarrow \mathcal{I} \cup \mathcal{I}_i$
    Compute $(\mathbf{A}, \mathbf{b}) \leftarrow \mathsf{Conv}((\mathbf{A}, \mathbf{b}), \mathcal{I}_i)$
    $i = i + 1$
    Call Regular Reduce and Prange's ISD$(m, n, t, (\mathbf{A}, \mathbf{b}), \{N_i\}_{i=1}^k, i, \mathcal{I})$
  end for
end if
if $i = k$ **then**
  Choose a random index set $\mathcal{I}_{k\ell} \subset [m] \setminus \mathcal{I}$ of size N_k/t for $1 \le \ell \le t$ and $\mathcal{I}_k = \cup_\ell \mathcal{I}_{k\ell}$
  $(\mathbf{A}_{\mathcal{I}_k}, \mathbf{b}_{\mathcal{I}_k}) \leftarrow [\mathbf{A}, \mathbf{b}]_{\mathcal{I}_k}$, $\mathbf{s}_{\mathcal{I}_k} = (\mathbf{A}_{\mathcal{I}_k})^{-1} \cdot \mathbf{b}_{\mathcal{I}_k}$
  if $|\mathbf{b}_{\mathcal{I}_k} - \mathbf{A}_{\mathcal{I}_k} \cdot \mathbf{s}_{\mathcal{I}_k} \bmod F_q| = t$ **then**
    break
  end if
  $\mathcal{I} \leftarrow \mathcal{I} \cup \mathcal{I}_k$,$(\mathbf{A}_\mathcal{I}, \mathbf{b}_\mathcal{I}) \leftarrow [\mathbf{A}, \mathbf{b}]_\mathcal{I}$, and $\mathbf{s} \leftarrow \mathbf{A}_\mathcal{I}^{-1} \cdot \mathbf{b}_\mathcal{I}$
  return $(\mathbf{s}, \mathcal{I})$
end if

Complexity. Using the adjusted probability, we now compute the time complexity of the regular-RP algorithm. To this end, we define $\mathcal{P}^R(j), \mathcal{C}(j)$ and $\mathcal{S}^R$ as follows:

$$\begin{aligned}
\mathcal{P}^R(j) &= \mathcal{P}^R_{m-\mu_{j-1},t,N_j} \\
\mathcal{C}(j) &= \mathcal{C}_{m-\mu_{j-1},N_j} \\
\mathcal{S}^R &= \frac{1}{\mathcal{P}^R_{m-\mu_k,t,n-\mu_k}} \cdot ((n-\mu_k)^\omega + (n-\mu_k)\cdot(m-\mu_k))
\end{aligned}$$

where $\mu_j = \sum_{i=1}^{j} N_i$ for every j. Then, using the notation, the total cost of level-k regular-RP algorithm, denoted by $\mathcal{T}^R$, can be expressed as

$$\frac{1}{\mathcal{P}^R(1)}\left(\mathcal{C}(1) + \frac{1}{\mathcal{P}^R(2)}\left(\mathcal{C}(2) + \cdots + \frac{1}{\mathcal{P}^R(k)}\left(\mathcal{C}(k) + \mathcal{S}^R\right)\right)\right)$$

To simplify it, we also define $\mathcal{Q}^R(j)$ as $\mathcal{Q}^R(j) = \prod_{i=1}^{j} \mathcal{P}^R(i)$. We then denote the time complexity as

$$\sum_{j=1}^{k} \frac{\mathcal{C}(j)}{\mathcal{Q}^R(j)} + \frac{\mathcal{S}^R}{\mathcal{Q}^R(k)}.$$

As in the RP algorithm, the success probability is given by $(1-1/e)^{k+1}$. Combining all components, the total cost is expressed as

$$\mathcal{T}^R = \sum_{j=1}^{k} \frac{\mathcal{C}(j)}{\mathcal{Q}^R_{\mathsf{correct}}(j)} + \frac{\mathcal{S}^R}{\mathcal{Q}^R_{\mathsf{correct}}(k)},$$

where $\mathcal{Q}^R_{\mathsf{correct}}(j) = \mathcal{Q}^R(j) \cdot (1-1/e)^{k+1}$.

4 Revisiting Algebraic Approach for Regular-LPN using RP Observation

A key insight from the RP algorithm is that Gaussian elimination contributes significantly to the overall computational complexity—more than initially anticipated. In this section, we leverage this observation to enhance the recent algebraic algorithm for solving the regular-LPN problem, as proposed by Briaud and Øygarden [21].

In essence, the algebraic approach exploits the fact that an (m, n, t)-regular LPN instance $(\mathbf{A}, \mathbf{b})$, where $\mathbf{b} = \mathbf{A}\cdot\mathbf{s}+\mathbf{e} \bmod \mathbb{F}_q$, can be converted into a system of multivariate polynomials whose common root corresponds to $\mathbf{s}$.

Formally, define $\mathbf{a}_{h,i}$ (Resp. $b_{h,i}$) as the $(h-1)\cdot\beta + i$-th row of $\mathbf{A}$ (Resp. $\mathbf{b}$) Likewise, we denote the i-th entry of $\mathbf{e}_h$ in a noise vector of regular LPN as $e_{h,i}$. Under the notations, we define a multivariate polynomial $f_{h_1,i_1,h_2 i_2}$:

$$\begin{aligned} f_{h_1,i_1,h_2,i_2}(\mathbf{x}) &= (b_{h_1,i_1} - \langle \mathbf{a}_{h_1,i_1}, \mathbf{x}\rangle) \cdot (b_{h_2,i_2} - \langle \mathbf{a}_{h_2,i_2}, \mathbf{x}\rangle) \\ &= \langle (b_{h_1,i_1}, -\mathbf{a}_{h_1,i_1}) \otimes (b_{h_2,i_2}, -\mathbf{a}_{h_2,i_2}), (1,\mathbf{x}) \otimes (1,\mathbf{x})\rangle, \end{aligned}$$

where $\otimes$ is a standard tensor product. On $\mathbf{x} = \mathbf{s}$, this polynomial equals $e_{h_1,i_1} \cdot e_{h_2,i_2}$. This implies that if $e_{h_1,i_1} \cdot e_{h_2,i_2} = 0$, the secret vector $\mathbf{s}$ is a root of the polynomial. We say the polynomial f_{h_1,i_1,h_2,i_2} as an error-free polynomial when $\mathbf{s}$ is a root.[4]

A standard approach to recovering $\mathbf{s}$ from the error-free polynomials is relinearization, which involves disregarding the ring structure. To be precise, via the relinearization, each polynomial $f_{h_1,i_1,h_2,i_2}(\mathbf{x})$ can be expressed as a linear polynomial $g_{h_1,i_1,h_2,i_2}(\mathbf{y})$:

$$g_{h_1,i_1,h_2,i_2}(\mathbf{y}) = \langle \mathbf{c}_{h_1,i_1,h_2,i_2}, \mathbf{y}\rangle,$$

[4] More generally, an error free polynomial can be extended into more higher-degree polynomials. In this paper, we however focus on only quadratic polynomials.

where $\mathbf{y}$ is a variable vector of $\mathbb{F}_q^{(1+n)^2}$ and $\mathbf{c}_{h_1,i_1,h_2,i_2} = (b_{h_1,i_1}, -\mathbf{a}_{h_1,i_1}) \otimes (b_{h_2,i_2}, -\mathbf{a}_{h_2,i_2})$. It is obvious that this polynomial has a solution $(1,\mathbf{s}) \otimes (1,\mathbf{s})$. In addition, an error-free instance $(b_{h,i}, \mathbf{a}_{h,i})$ such that $b_{h,i} - \mathbf{a}_{h,i} \cdot \mathbf{s} = 0$ gives an error-free polynomial

$$g_{h,i,h,i}(\mathbf{y}) = \langle (b_{h,i}, -\mathbf{a}_{h,i}) \otimes (b_{h,i}, \mathbf{a}_{h,i}), \mathbf{y} \rangle.$$

This polynomial has a solution $(1,\mathbf{s}) \otimes (1,\mathbf{s})$. By collecting $(1+n)^2$ error-free polynomials, we construct a Macaulay matrix that allows recovery of $(1,\mathbf{s}) \otimes (1,\mathbf{s})$ and ultimately $\mathbf{s}$ via Gaussian elimination. The concrete cost of the Gauss elimination algorithm is $(1+n)^{2\cdot\omega} \approx n^{2w}$.

4.1 Algebraic-RP: Combining the Algebraic Algorithm RP

We introduce Algebraic-RP, a new method for solving regular-LPN over $\mathbb{F}_q$ by integrating the algebraic approach with RP. As the observation in the RP algorithm, it is useful to reduce the concrete complexity of the algorithm.

Step 1. Let $\mathbf{A}_{\mathsf{top}}$ and $\mathbf{A}_{\mathsf{bot}}$ be submatrices of $\mathbf{A}$, consisting of the first $m-n$ rows and the last n rows of $\mathbf{A}$, respectively. Similarly, we define $\mathbf{b}_{\mathsf{top}}$, $\mathbf{b}_{\mathsf{bot}}$, $\mathbf{e}_{\mathsf{top}}$, and $\mathbf{e}_{\mathsf{bot}}$ to correspond to these partitions.

Assuming that the matrix $\mathbf{A}_{\mathsf{bot}}$ is invertible over $\mathbb{F}_q$ (if not, we reorder the columns until $\mathbf{A}_{\mathsf{bot}}$ becomes invertible), we can derive the following regular-LPN samples:

$$\begin{aligned}
\mathbf{b}' &= \left(\mathbf{I}_{m-n} \| -\mathbf{A}_{\mathsf{top}} \cdot (\mathbf{A}_{\mathsf{bot}})^{-1}\right) \cdot \begin{pmatrix} \mathbf{b}_{\mathsf{top}} \\ \mathbf{b}_{\mathsf{bot}} \end{pmatrix} \\
&= \left(\mathbf{I}_{m-n} \| -\mathbf{A}_{\mathsf{top}} \cdot (\mathbf{A}_{\mathsf{bot}})^{-1}\right) \cdot \left(\begin{pmatrix} \mathbf{A}_{\mathsf{top}} \\ \mathbf{A}_{\mathsf{bot}} \end{pmatrix} \cdot \mathbf{s} + \begin{pmatrix} \mathbf{e}_{\mathsf{top}} \\ \mathbf{e}_{\mathsf{bot}} \end{pmatrix} \right) \bmod \mathbb{F}_q \\
&= \underbrace{\left(\mathbf{A}_{\mathsf{top}} - \mathbf{A}_{\mathsf{top}} \cdot (\mathbf{A}_{\mathsf{bot}})^{-1} \cdot \mathbf{A}_{\mathsf{bot}}\right)}_{=:\mathbf{A}'} \cdot \mathbf{e}_{\mathsf{bot}} + \mathbf{e}_{\mathsf{top}} \bmod \mathbb{F}_q.
\end{aligned}$$

We denote this modified instance as $(\mathbf{A}', \mathbf{b}') \leftarrow \mathsf{Conv}_0(\mathbf{A}, \mathbf{b})$. Notably, this transformation does not require any guessing of the entries of $\mathbf{e}$, and thus, we employ a distinct notation.

Step 2. Due to the regularity of the error vector, each β-dimensional subvector of $\mathbf{e}$ contains a single nonzero entry. Thus we can expect $\mathbf{e}_{\mathsf{bot}}$ has $\frac{t\cdot n}{m}$[5] non-zero entries. Let $t_0 = \frac{t\cdot n}{m}$ and $t_1 = t - t_0$ be a number of non-zero entries in $\mathbf{e}_{\mathsf{bot}}$ and $\mathbf{e}_{\mathsf{top}}$, respectively. Our strategy is to guess $t_1 \cdot n_1$-zero entries in each $\mathbf{e}_{\mathsf{top}}$ and n_2-zero entries in $\mathbf{e}_{\mathsf{bot}}$, separately.

Suppose that the n_2 zero positions are correctly guessed in $\mathbf{e}_{\mathsf{bot}} \in \mathbb{F}_q^n$, and let $\mathbf{e}_{\mathsf{bot}}^{(2)}$ denote the remaining nonzero part of $\mathbf{e}_{\mathsf{bot}}$. Using this notation, the instance can be rewritten as

$$\mathbf{b}' = \mathbf{A}^{(2)} \cdot \mathbf{e}_{\mathsf{bot}}^{(2)} + \mathbf{e}_{\mathsf{top}} \bmod \mathbb{F}_q,$$

[5] For simplicity, we omit the round $\lfloor \cdot \rceil$ notations.

where $\mathbf{A}^{(2)}$ is obtained by removing the columns of $\mathbf{A}'$ corresponding to the guessed zero positions in $\mathbf{e}_{\mathsf{bot}}$.

Next, suppose that the $t_1 \cdot n_1$ zero positions in $\mathbf{e}_{\mathsf{top}}$, indexed by $\mathcal{I}$, are also correctly guessed. We then update the instance $(\mathbf{A}^{(3)}, \mathbf{b}^{(3)})$ by applying

$$(\mathbf{A}^{(3)}, \mathbf{b}^{(3)}) \leftarrow \mathsf{Conv}((\mathbf{A}^{(2)}, \mathbf{b}'), \mathcal{I}).$$

This results in the following representation:

$$\mathbf{b}^{(3)} = \mathbf{A}^{(3)} \cdot \mathbf{s}^{(3)} + \mathbf{e}^{(3)} \bmod \mathbb{F}_q,$$

where $\mathbf{s}^{(3)} \in \mathbb{F}_q^{n-n_1 \cdot t_1 - n_2}$ and $\mathbf{e}^{(3)} \in \mathbb{F}_q^{t_1 \cdot (n-n_1)}$.

Step 3. We now construct error-free polynomials from the instance $(\mathbf{A}^{(3)}, \mathbf{b}^{(3)})$. For simplicity, we denote $\mathbf{e}^{(3)} = (\mathbf{e}_1^{(3)}, \ldots, \mathbf{e}_{t_1}^{(3)})$, where each $\mathbf{e}_i^{(3)} \in \mathbb{F}_q^{\beta - n_1}$ contains a single nonzero entry.

Following the notation above, let $\mathbf{a}_{i,j}^{(3)}$ (resp. $b_{i,j}^{(3)}$) denote the $(i-1)\cdot(t-t_1)+j$-th row of $\mathbf{A}^{(3)}$ (resp. $\mathbf{b}^{(3)}$). We then define the polynomial $g_{i,j,k}(\mathbf{y})$ as $g_{i,j,k}(\mathbf{y}) := \langle (b_{i,j}^{(3)}, -\mathbf{a}_{i,j}^{(3)}) \otimes (b_{i,k}^{(3)}, -\mathbf{a}_{i,k}^{(3)}), \mathbf{y} \rangle$. When $\mathbf{y} = (1, \mathbf{s}^{(3)}) \otimes (1, \mathbf{s}^{(3)})$, this polynomial evaluates to $e_{i,j}^{(3)} \cdot e_{i,k}^{(3)}$, which equals zero entry, whenever $j \neq k$, due to the property that each $\mathbf{e}_i^{(3)}$ contains only a single nonzero. By leveraging all $\mathbf{e}_{i,j}^{(3)}$, we can construct $\left(\binom{\beta - n_1}{2}\right) \cdot t_1$ error-free polynomials.

Aggregating these polynomials, we define a matrix $\mathbf{A}^{(4)}$, where each row vector corresponds to the coefficient vector of $g_{i,j,k}(\mathbf{y})$. Thus, the number of rows in $\mathbf{A}^{(4)}$ is $\left(\binom{\beta - n_1}{2}\right) \cdot t_1$. Since the dimension of $\mathbf{s}^{(3)}$ is $n_3 := n - n_1 \cdot t_1 - n_2$, the number of columns in $\mathbf{A}^{(4)}$ is $(n_3 + 1)^2$.

Step 4. The column dimension $(n_3 + 1)^2$ can be further reduced. First, we redefine the target vector $(1, \mathbf{s}^{(3)}) \otimes (1, \mathbf{s}^{(3)})$ as $\mathbf{s}^{(5)} = (1, \mathbf{s}^{(3)}, (\mathbf{s}^{(3)})^2)$, where $(\mathbf{s}^{(3)})^2$ is a vector consisting of terms $(s_i^{(3)} \cdot s_j^{(3)})$. Thus, $\mathbf{A}^{(4)}$ can be transformed into a lower-dimensional matrix $\mathbf{A}^{(5)}$ such that $\mathbf{A}^{(4)} \cdot \left((1, \mathbf{s}^{(3)}) \otimes (1, \mathbf{s}^{(3)})\right) = \mathbf{A}^{(5)} \cdot \mathbf{s}^{(5)}$.

Moreover, since $\mathbf{s}^{(5)}$ is derived from $\mathbf{e}_{\mathsf{bot}}$ and each β-dimensional subvector has a single nonzero entry, we observe that $s_i^{(3)} \cdot s_j^{(3)} = 0$ whenever $s_i^{(3)}$ and $s_j^{(3)}$ originate from the same component $\mathbf{e}_i$ of the initial noise vector in the regular LPN instance $(\mathbf{A}, \mathbf{b})$. Thus, when removing the $n_1 \cdot t_1 + n_2$ coordinates of $\mathbf{e}_{\mathsf{bot}}$, we eliminate $(n_1 \cdot t_1 + n_2)/t_0$ coordinates from each β-dimensional subvector, reducing the dimension of $s_i^{(3)} \cdot s_j^{(3)}$ to $\binom{\beta - (n_1 \cdot t_1 + n_2)/t_0}{2} \cdot t_0$. Incorporating this information, the number of columns in $\mathbf{A}^{(5)}$ becomes $n_4 := \binom{n_3+1}{2} - \binom{\beta - (n_1 \cdot t_1 + n_2)/t_0}{2} \cdot t_0$. We denote this transformation as $\mathbf{A}^{(5)} \leftarrow \mathsf{AlgCons}(\mathbf{A}^{(3)}, \mathbf{b}^{(3)})$.

Step 5. As in standard algebraic attacks, assuming that all column vectors of $\mathbf{A}^{(5)}$ are linearly independent, one can recover $\mathbf{s}^{(5)}$ by computing the right kernel

of $\mathbf{A}^{(5)}$ whenever

$$\left(\binom{\beta - n_1}{2}\right) \cdot t_1 \geq \binom{n_3 + 1}{2} - \binom{\beta - \frac{n_1 \cdot t_1 + n_2}{t_0}}{2} \cdot t_0. \tag{1}$$

If the guess is incorrect, each row of $\mathbf{A}^{(5)}$ does not satisfy the property that $\mathbf{s}^{(5)}$ is orthogonal, meaning the algorithm fails to return a valid solution in such cases.

Additionally, since $\mathbf{s}^{(5)}$ contains $\mathbf{s}^{(3)}$, which is recovered after identifying the $n_1 \cdot t_1 + n_2$ zero coordinates, this guessing and computation process is equivalent to recovering all n entries of $\mathbf{e}$. Consequently, the true secret vector $\mathbf{s}$ can be reconstructed once $\mathbf{e}$ is recovered. We denote this algorithm as $\mathbf{s} \leftarrow \mathsf{SolveLPN}(\mathbf{A}^{(5)})$.

The exact algorithm is then described as follows:

Algorithm 6 Algebraic Regular Reduce and Prange's ISD

Input: (m, n, t)-regular LPN over $\mathbb{F}_q$ instances $(\mathbf{A}, \mathbf{b}) \in \mathbb{F}_q^{m \times (n+1)}$ and an integral pair (n_1, n_2).
Output: The secret vector $\mathbf{s}$ such that $|\mathbf{b} - \mathbf{A} \cdot \mathbf{s} \bmod \mathbb{F}_q| = t$.

Compute $t_0 = \lfloor \frac{t \cdot n}{m} \rceil$, $t_1 = t - t_0$, and $n_3 = n - n_1 \cdot t_1 - n_2$
if $\left(\binom{\beta - n_1}{2}\right) \cdot t_1 < \binom{n_3+1}{2} - \binom{\beta - \frac{n_1 \cdot t_1 + n_2}{t_0}}{2} \cdot t_0$ **then**
  return $\perp$
end if
Compute $(\mathbf{A}', \mathbf{b}') \leftarrow \mathsf{Conv}_0(\mathbf{A}, \mathbf{b})$
Choose a random index set $\mathcal{J}$ of size n_2 considering the regular constraint
Update $(\mathbf{A}^{(2)}, \mathbf{b}')$ removing the columns following the index set $\mathcal{J}$
Choose a random index set $\mathcal{I}_j$ of size n_1 for $1 \leq j \leq t_1$ and define $\mathcal{I} = \cup_j \mathcal{I}_j$
Update $(\mathbf{A}^{(3)}, \mathbf{b}^{(3)}) \leftarrow \mathsf{Conv}((\mathbf{A}^{(2)}, \mathbf{b}'), \mathcal{I})$
Compute $\mathbf{A}^{(5)} \leftarrow \mathsf{AlgCons}(\mathbf{A}^{(3)}, \mathbf{b}^{(3)})$
return $\mathbf{s} \leftarrow \mathsf{SolveLPN}(\mathbf{A}^{(5)})$.

Complexity. Given (n_1, n_2)-pair satisfying Equation (1), the secret dimension is reduced to n_3 following the **S**tep 2. To successfully reduce it, it is required to correctly guess the $t_1 \cdot n_1$ zero positions in $\mathbf{e}_{\mathsf{top}}$ and n_2 zero positions in $\mathbf{e}_{\mathsf{bot}}$. After the guessing, the **Step 4** terminates in n_4^ω. Consequently, the regular-LPN problem can be solved using the Algorithm 6 in time:

$$\mathcal{T}^R = \frac{\mathcal{C}_{m-n, t_1 \cdot n_1} + n_4^\omega}{\mathcal{P}^R_{t_1 \cdot \beta, t_1, t_1 \cdot n_1} \cdot \mathcal{P}^R_{t_0 \cdot \beta, t_0, n_2}} + \mathcal{C}_{m,n}, \tag{2}$$

where $\mathcal{C}_{m,n}$ represents the Conv_0 complexity. Its success probability relies on guessing $\mathcal{J}$ and $\mathcal{I}$. Thus, its success probability equals to $(1 - 1/e)^2$. It indicates that the total complexity of the algebraic RP becoems $\mathcal{T}^R/(1-1/e)^2$. Compared to the naive approach, the iteration cost is reduced from $n^{2\omega}$ to n_4^ω.

Remark 1 (Comparison to Briaud and Øygarden's algebraic attack [21]). The attack proposed by Briaud and Øygarden involves constructing a high-degree

multivariate polynomial system that has a unique common root. By employing the relinearization technique, also known as the Macaulay matrix, as described above, one can recover the secret $\mathbf{s}$ via the Gaussian elimination algorithm.

We have observed that the process entails guessing error-free polynomials, constructing a Macaulay matrix, and then using the Gaussian elimination algorithm to find the solution. This implies that each iteration requires the application of Gaussian elimination.

Therefore, this algorithm can be accelerated using the same principles as RP, and the main distinction between our algorithm and the [21] algorithm lies in this aspect.

5 Concrete Estimation of LPN via Reduce and Prange algorithm

In this section, we provide the estimated results of our algorithms for solving LPN over large fields. In particular, we focus on security estimations for LPN over large fields in PCG applications. The estimated results are summarized in the following tables:

- Table 1: Comparison of RP with previous works for solving LPN over $\mathbb{F}_{2^{128}}$.
- Table 2: Numerical results of RP on recommendation parameters for 128-bit security in [49]
- Table 3: Comparison of regular-RP with previous works solving regular-LPN
- Table 4: Numerical results of regular-RP on recommendation parameters for 128-bit security in [49]

Throughout this section, the time complexity refers to the number of arithmetic operations in each field.

Our numerical estimations are derived using the previously defined complexity metrics $\mathcal{Q}_{\mathsf{correct}}$ and $\mathcal{Q}^R_{\mathsf{correct}}$, rather than their non-tilde counterparts. We further compare the results with recent attacks [21, 38, 49] depending on problems in each tables. For fair comparisons, the time complexity of the Gaussian elimination algorithm on a $k \times k$ matrix is evaluated using the same cost model. Specifically, we use a cost model of $k^{2.8}$, as in previous works [17, 49].[6]

As shown in Table 2, our algorithm consistently outperforms previous works for LPN over large fields. For regular-LPN over large fields, which achieves 128-bit security, our algorithm tends to be better when m, n is small, but the algorithm quickly becomes worse for large parameters. For this purpose, we first recall the parameters for LPN over large fields.

- q: the size of fields $= 2^{128}$.
- m: the number of LPN instances.
- n: the dimension of LPN secret $\mathbf{s}$.

[6] The exponent is inspired by the Strassen matrix multiplication algorithm.

– t: the number of nonzero entries in $\mathbf{e}$.

Our focus is mainly on the parameter regime $(m, q, t) = (poly(n), n^{w(1)}, o(m))$, which is widely used in cryptographic applications [9,10,17,19,20,28,32,61,65,69].

For a more detailed analysis, the complete source code for our computational experiments is publicly available The repository includes implementations of the RP estimation and its faster variant, quick-RP, applied to both LPN and regular-LPN problems. The code can be accessed via the following link: RP and Quick-RP Estimations.

– RP: The basic cost model proposed by Section 2.2.
– regular-RP: RP for regular-LPN Section 3.
– Quick-RP: As an alternative, we propose a quicker estimation method using the RP algorithm with $k = 3$. We configure it by setting $N_1 = \frac{13}{15} \cdot n, N_2 = \frac{1}{15} \cdot n, N_3 = \frac{1}{15} \cdot n$. We say the RP on the index size quick-RP.

To evaluate our algorithm, we compare its performance against ISD and its variants. We begin by recalling the results from [49]:

– For the parameter regime $\mathcal{R} = \mathbb{F}_q$ with prime power $q = n^{w(1)}$, $t = o(m)$ and $(1 + \beta) \cdot n \leq m = poly(n)$ for some constant β, the statistical decoding including the 2.0 variant [26] needs more cost than Prange's algorithm [58].
– For the above parameter setup, information set decoding and its variants outperform other algorithms.

Therefore, ISD and its variants serve as the most relevant benchmarks for evaluating our results. In the following, we then describe the complexity model of the algorithms.

5.1 Cost models of the Previous algorithms

This section introduces cost models for three major algorithms used to solve LPN over large fields: Gauss ISD [58], ISD frameworks, and Statistical Decoding [26, 49].

Gauss Algorithm. The Gauss algorithm is identical to Prange's algorithm (see Algorithm 1), but it covers an additional case: LPN instances $(\mathbf{A}, \mathbf{b}) = (\mathbf{A}, \mathbf{A} \cdot \mathbf{s} + \mathbf{e})$ can be transformed into the problem of finding $\mathbf{e}$ when $\mathbf{H} \cdot \mathbf{e} = \mathbf{H} \cdot \mathbf{b}$ for a parity check matrix $\mathbf{H}$, and vice versa, as shown in [54, Lemma 4.9].

The computational cost in this case is: $(m-n)^{2.8} \cdot \left(\frac{\binom{m}{t}}{\binom{m-n}{t}} \right)$. Thus, the Gauss algorithm follows the cost model:

$$\texttt{Gauss_cost_model} = \log \left(\min\{n^{2.8}, (m-n)^{2.8}\} \cdot \left(\frac{\binom{m}{t}}{\binom{m-n}{t}} \right) \right)$$

Information Set Decoding. For LPN over large fields and regular-LPN problems, Liu et. al. [49] showed that SD-ISD is the most efficient algorithm. As

a result, they estimate the hardness of LPN over large fields and regular-LPN based on the cost of SD-ISD. Thus, we introduce only a cost of SD-ISD attack as a representative of ISD frameworks.

Lemma 5.1 (Cost of SD-ISD attack [49]) *The (m, n, t)-LPN problem over $\mathbb{F}_q$ can be solved by the SD-ISD variant in expected time*

$$T_{SD}(m,n,t) = \min_{p,\ell} \frac{1}{\mathcal{P}(p,\ell)} \left(T_{\mathsf{Gauss}} + 2L \cdot m \left(1 + \frac{L}{q^\ell} \right) \right)$$

where $T_{\mathsf{Gauss}} = \frac{(m-n-\ell)\cdot m^2}{\log(m-n-\ell)}$, $L = \binom{\frac{n+\ell}{2}}{p/2} \cdot (q-1)^{p/2}$ *and* $\mathcal{P}(p,\ell) = \frac{\binom{m-n-\ell}{t-p}}{\binom{m}{t}} \cdot \frac{L^2}{(q-1)^p}$.[7]

Statistical Decoding framework. While ISD algorithms aim to recover the secret element $\mathbf{s}$ directly, the Low Weight Parity Check (LPC) algorithm focuses on distinguishing between random instances and LPN instances. In other words, it determines whether $\mathbf{b}$ is equal to $\mathbf{A} \cdot \mathbf{s} + \mathbf{e} \bmod \mathbb{F}_q$ or just a random element. Although this algorithm is not well-suited for solving LPN over large fields, we still provide its attack complexity as a comparison for decision problems. For $q \geq 4t$, the adapted SD 2.0 algorithm is proposed in [49].

Lemma 5.2 ([26, 49]) *The adapted SD 2.0 algorithm can solve the decisional (m, n, t)-LPN over $\mathbb{F}_q$ in time T with constant advantage, where $w = w(s) \in \mathbb{N}$ and $q \geq 4t$. The formula for T is given by*

$$T = \min_s \left(T_1 \cdot \left(\frac{\binom{m}{t}}{\binom{m-w}{t}} \cdot \frac{q}{q-1} \right)^2 + s \cdot \log q \cdot q^s \right).$$

Here, T_1 represents the time it takes to find one parity check vector. Following [17, 49], we also set $w = n - s + 1$ and $T_1 = n + 1$. As a result, we have

$$T = \min_s \left((n+1) \cdot \left(\frac{\binom{m}{t}}{\binom{m-n+s-1}{t}} \cdot \frac{q}{q-1} \right)^2 + s \cdot \log q \cdot q^s \right).$$

5.2 Numerical Results of LPN over Large Fields

This section presents the numerical results obtained from our attack estimation for different parameter settings. We developed a SageMath [62] script to search for near-optimal parameterization for our attack. The original algorithm would have examined the attack complexity that is minimized for all Δ values, but we examined the values via the binary search with depth ≤ 12 to find an approximate minimum quickly with a constraint $\Delta \leq n^{2.8}$. The cost we found also provides a sufficiently small value compared to the original cost. We also provide

[7] As the paper [49] only presents an estimation of LPN over $\mathbb{F}_2$, we rely on the formulation provided by their Python script to describe our approach.

comparison results between [49] and ours for a few parameters. The results are given by Table 1.

Our results indicate that for the parameters used in [49], the bit-security of LPN over large fields is reduced by at most 9 bits when $\log q = 128$. Consequently, the parameters described in [49] do not achieve 128-bit security. To ensure 128-bit security, we propose new parameter settings, as shown in Table 2.

Table 1. Comparison results with previous results [49, Tab. 2] and ours. All values are logarithmic scale of arithmetic operations over $\mathbb{F}_q$ with the field size $\log q = 128$. The column SD indicates the result of algebraic attacks In our works, 'RP' column presents the estimated result of Reduce and Prange algorithm. † indicates estimation results obtained by quick-RP.

Parameters			[49][8]			**This work**
m	n	t	Gauss	ISD	SD	
2^{10}	652	57	111	111	184	**105**
2^{12}	1589	98	100	100	151	**94**
2^{14}	3482	198	101	101	149	**97**
2^{16}	7391	389	103	103	147	**99**
2^{18}	15536	760	**105**	105	146	108
2^{20}	32771	1419	107	145	**102**	104
2^{22}	67440	2735	108	144	**104**	107
2^{12}	$3/4 \cdot 2^{12}$	44	117	117	189	**111**
2^{14}	$3/4 \cdot 2^{14}$	39	111	111	170	**107**
2^{16}	$3/4 \cdot 2^{16}$	34	107	107	151	**104**
2^{18}	$3/4 \cdot 2^{18}$	32	108	108	145	**106**
2^{20}	$3/4 \cdot 2^{20}$	31	112	112	143	**110**†
2^{22}	$3/4 \cdot 2^{22}$	30	116	116	141	**114**†
2^{24}	$3/4 \cdot 2^{24}$	29	119	119	139	**118**†

Remark 2. Some readers might be curious about the configuration of the sub-dimension sets $\{N_i\}_{i=1}^{k}$. Given our observations, with k typically ranging from over 20 to as high as 2000, accurately defining the set can be challenging. Concurrently, it has been noted that the size of N_1 often exceeds $N/3$. Similar occurrences are also observed in the context of the regular-LPN.

[8] [49] also presents numerical results for the statistical decoding 2.0 attack [27]. However, we did not take these results into account because they showed lower performance compared to than Gauss and ISD results in these parameters in the table.

Table 2. Comparison results with previous results [49, Tab. 3] and ours. All values are logarithmic scale of arithmetic operations over $\mathbb{F}_q$ with the field size $\log q = 128$. In our works, 'RP' column presents the estimated result of Reduce and Prange algorithm. RP for recommended parameters takes at least 2^{128} arithmetic operations.

Parameters			[49]	RP	Recommend Parameters					
					Increase n			Increase t		
m	n	t			m	n	t	m	n	t
2^{12}	1321	172	128	**121**	2^{12}	1400	172	2^{12}	1321	188
2^{14}	2895	338		**122**	2^{14}	2895	338	2^{14}	2895	367
2^{16}	6005	667		**123**	2^{16}	6450	667	2^{16}	6005	720
2^{18}	12160	1312		**124**	2^{18}	12900	1312	2^{18}	12160	1400
2^{20}	25346	2467		**124**	2^{20}	26900	2467	2^{20}	25346	2617
2^{22}	50854	4788		**125**	2^{22}	53490	4788	2^{22}	50854	5050

5.3 Numerical Results of Regular-LPN over Large Fields

This section provides numerical results for solving regular-LPN for various parameters using the regular-RP and algebraic-RP algorithms, and compares these results with [21, 49]. As in the previous section, we primarily focus on regular-LPN over large fields. The estimations of regular-LPN over the binary field are deferred to the next subsection.

We first note that [21, 49] provide numerical results on the bit-security of regular-LPN over $\mathbb{F}_q$ with $\log q = 128$, as both algorithms have minimal dependency on the field size. Conversely, [38] exclusively considers the bit-security of regular-LPN over $\mathbb{F}_2$. Therefore, we revisit the algorithms in [38] and re-estimate the bit-security of regular-LPN over $\mathbb{F}_{2^{128}}$ in this section.

[38] introduces three types of algorithms for solving regular-LPN: permutation-based regular-ISD (P-ISD), enumeration-based regular-ISD (E-ISD), and representation-based regular-ISD (R-ISD). These algorithms are adaptations of well-known techniques [12, 51, 58] tailored to the structure of regular-LPN. Among these, we specifically focus on the P-ISD algorithm. This choice is motivated by the fact that the other algorithms require collecting a large number of collision vectors over $\mathbb{Z}_q$, which significantly increases the concrete complexity.

Indeed, numerical results from [49, Tab. 3, Tab. 4] indicate that the Gaussian elimination costs are consistently comparable to the ISD costs for regular-LPN over $\mathbb{F}_{2^{128}}$. However, for regular-LPN over $\mathbb{F}_2$, the ISD algorithms consistently outperform Gaussian elimination in terms of cost.

The numerical results are presented in Table 3 and Table 4, demonstrating that our attack outperforms existing methods across almost all parameter settings. As in the previous section, all numerical values are logarithmic scale of the number of the number of arithmetic operations on $\mathbb{F}_{2^{128}}$.

Revisiting P-ISD for regular-LPN over $\mathbb{F}_q$ with $q \gg 2$. As discussed in Section 3.1 and [37, Sec. 4.1], we can just compute the time cost as follows

$$\left(1 - \frac{\binom{n}{t}}{\binom{m}{t}}\right)^{-t} \cdot \min\{n^{2.8}, (m-n)^{2.8}\} = \left(1 - \frac{n}{m}\right)^{-t} \cdot \min\{n^{2.8}, (m-n)^{2.8}\}$$

for solving (m, n, t)-regular-LPN. We further note that in [37,38], they computed the time complexity

$$\left(1 - \frac{n-t}{m}\right)^{-t} \cdot \min\{n^{2.8}, (m-n)^{2.8}\}$$

because one can directly obtain extra t error-free polynomials as in Appendix A. However, in case of regular-LPN over $\mathbb{F}_q$, it does not occur. Hence, we re-estimate the bit-security of regular LPN over $\mathbb{F}_{2^{128}}$ as follows:

$$\left(1 - \frac{n}{m}\right)^{-t} \cdot \min\{n^{2.8}, (m-n)^{2.8}\}.$$

5.4 Numerical Results of regular-LPN over $\mathbb{F}_2$

This section presents numerical results for solving regular-LPN over $\mathbb{F}_2$ across various parameters. All results in this section are obtained using the algebraic-RP algorithm detailed in Section 4. We chose this approach because the regular-RP algorithm presented in Section 3 shows significant deviation from the best estimations in [38] when $q = 2$. We also note the technique in Appendix A additionally provides error free polynomials. Our complexity estimations are derived from Equation (2), with results presented in Table 5 and Table 6.

References

1. C. Aguilar-Melchor et al. The syndrome decoding in the head (sd-in-the-head) signature scheme. submission to the nist call for additional post-quantum signatures (2023). https://www.sdith.org/docs/sdith-v1.1.pdf.
2. C. Aguilar-Melchor, N. Gama, J. Howe, A. Hülsing, D. Joseph, and D. Yue. The return of the sdith. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 564–596. Springer, 2023.
3. C. Aguilar-Melchor, A. Hülsing, D. Joseph, C. Majenz, E. Ronen, and D. Yue. Sdith in the qrom. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 317–350. Springer, 2023.
4. M. Alekhnovich. More on average case vs approximation complexity. In *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.*, pages 298–307. IEEE, 2003.

[9] This is a re-estimated result of permutation-based algorithm that only suits for regular-LPN over large fields.

Table 3. Numerical results for regular-LPN over $\mathbb{F}_{2^{128}}$. The value in parentheses indicates the results of algebraic RP. † indicates estimation results obtained by quick regular-RP

Parameters			[48, 49]	[21]	[38][9]	**This work**
m	n	t				
2^{10}	652	106	194	179	178	**161** (245)
2^{12}	1589	172	155	150	151	**141** (195)
2^{14}	3482	338	150	150	149	**144** (173)
2^{16}	7391	667	151	150	151	**142** (152)
2^{18}	15336	1312	153	**133**	154	145 **(132)**
2^{20}	32771	2467	155	131	155	148 **(120)**
2^{22}	67440	4788	152	110	156	150 **(99)**
2^{10}	652	57	111	111	107	**105** (129)
2^{12}	1589	98	100	107	99	**91** (112)
2^{14}	3482	198	101	110	101	**94** (109)
2^{16}	7391	389	103	111	103	**97** (103)
2^{18}	15336	760	105	107	105	100 **(98)**
2^{20}	32771	1419	107	102	107	102 **(91)**
2^{22}	67440	2735	108	104	108	105 **(89)**
2^{12}	$3/4 \cdot 2^{12}$	44	117	127	116	**107** (137)
2^{14}	$3/4 \cdot 2^{14}$	39	111	127	111	**105** (135)
2^{16}	$3/4 \cdot 2^{16}$	34	107	128	107	**101** (132)
2^{18}	$3/4 \cdot 2^{18}$	32	108	132	108	**104** (140)
2^{20}	$3/4 \cdot 2^{20}$	31	112	139	112	**110** (150)
2^{22}	$3/4 \cdot 2^{22}$	30	116	145	116	$120^{\dagger}$ (157)
2^{24}	$3/4 \cdot 2^{24}$	29	119	152	119	$124^{\dagger}$ (155)

Table 4. Comparison results with previous results [49, Tab. 3] and ours for numerical results for regular-LPN. The recommended parameters are set to require at least 2^{128} arithmetic operations for regular-RP and algebraic RP algorithms.

Parameters			[49]	RP	Recommend Parameters					
					Increase n			Increase t		
m	n	t			m	n	t	m	n	t
2^{12}	1377	172	128	**120**	2^{12}	1460	172	2^{12}	1377	187
2^{14}	2909	338		**118**	2^{14}	3165	338	2^{14}	2909	373
2^{16}	6091	667		**118**	2^{16}	6500	667	2^{16}	6091	713
2^{18}	14796	1312		128	-	-	-	-	-	-
2^{20}	30978	2467		**113**	2^{20}	34980	2467	2^{20}	30978	2969
2^{22}	75396	4788		**112**	2^{22}	84590	4788	2^{22}	75396	5776

Table 5. Numerical results of algebraic-RP over $\mathbb{F}_2$. The dash (-) indicates the paper did not provide estimation results for such parameters.

Parameters			[48, 49]	[21]	[38]	**This work**
m	n	t				
2^{10}	652	106	145	145	**113**	188
2^{12}	1589	172	121	131	**109**	166
2^{14}	3482	338	122	138	**118**	152
2^{16}	7391	667	**125**	139	126	136
2^{18}	15336	1312	**122**	**122**	138	**122**
2^{20}	32771	2467	125	125	155	**111**
2^{22}	67440	4788	103	103	156	**91**
2^{10}	652	57	94	94	**76**	113
2^{12}	1589	98	83	83	**78**	104
2^{14}	3482	198	86	86	**84**	103
2^{16}	7391	389	91	91	**90**	99
2^{18}	15336	760	95	95	**93**	95
2^{20}	32771	1419	99	99	98	**88**
2^{22}	67440	2735	104	104	103	**89**
2^{12}	$3/4 \cdot 2^{12}$	44	**96**	125	-	132
2^{14}	$3/4 \cdot 2^{14}$	39	**95**	127	-	134
2^{16}	$3/4 \cdot 2^{16}$	34	**93**	128	-	135
2^{18}	$3/4 \cdot 2^{18}$	32	**95**	132	-	140
2^{20}	$3/4 \cdot 2^{20}$	31	**99**	139	-	151
2^{22}	$3/4 \cdot 2^{22}$	30	**103**	145	-	157
2^{24}	$3/4 \cdot 2^{24}$	29	**107**	152	-	155

Table 6. Comparison results with previous results [49, Tab. 3] and ours for numerical results for regular-LPN over $\mathbb{F}_2$. The algebraic-RP for recommend parameters takes at least 2^{128} arithmetic operations.

Parameters			[49]	RP	Recommend Parameters					
					Increase n			Increase t		
m	n	t			m	n	t	m	n	t
2^{12}	1657	172	128	179						
2^{14}	3655	338		163						
2^{16}	7560	667		140						
2^{18}	15996	1312		127						
2^{20}	33354	2467		**113**	2^{20}	37360	2467	2^{20}	33354	3055
2^{22}	80074	4788		**112**	2^{22}	89074	4788	2^{22}	80074	5907

5. B. Applebaum, B. Barak, and A. Wigderson. Public-key cryptography from different assumptions. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 171–180, 2010.
6. B. Applebaum, D. Cash, C. Peikert, and A. Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*, volume 5677 of *Lecture Notes in Computer Science*, pages 595–618. Springer, 2009.
7. B. Applebaum, I. Damgård, Y. Ishai, M. Nielsen, and L. Zichron. Secure arithmetic computation with constant computational overhead. In *Annual International Cryptology Conference*, pages 223–254. Springer, 2017.
8. D. Augot, M. Finiasz, and N. Sendrier. A family of fast syndrome based cryptographic hash functions. In *Mycrypt*, volume 3715, pages 64–83. Springer, 2005.
9. C. Baum, L. Braun, A. Munch-Hansen, and P. Scholl. Moz$\mathbb{Z}_2^k$arella: efficient vector-ole and zero-knowledge proofs over z 2 k. In *Advances in Cryptology–CRYPTO 2022: 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15–18, 2022, Proceedings, Part IV*, pages 329–358. Springer, 2022.
10. C. Baum, A. J. Malozemoff, M. B. Rosen, and P. Scholl. Mac'n'cheese mac′ n′ cheese: Zero-knowledge proofs for boolean and arithmetic circuits with nested disjunctions. In *Advances in Cryptology–CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part IV 41*, pages 92–122. Springer, 2021.
11. G. Beck, A. Goel, A. Hegde, A. Jain, Z. Jin, and G. Kaptchuk. Scalable multiparty garbling. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 2158–2172, 2023.
12. A. Becker, A. Joux, A. May, and A. Meurer. Decoding random binary linear codes in 2 n/20: How 1+ 1= 0 improves information set decoding. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 520–536. Springer, 2012.
13. D. J. Bernstein, T. Lange, and C. Peters. Attacking and defending the mceliece cryptosystem. In J. Buchmann and J. Ding, editors, *Post-Quantum Cryptography*, pages 31–46, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.
14. A. Blum, M. L. Furst, M. J. Kearns, and R. J. Lipton. Cryptographic primitives based on hard learning problems. In *Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology Conference, Santa Barbara, California, USA, August 22-26, 1993, Proceedings*, volume 773 of *Lecture Notes in Computer Science*, pages 278–291. Springer, 1993.
15. A. Blum, A. Kalai, and H. Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *Journal of the ACM (JACM)*, 50(4):506–519, 2003.
16. L. Both and A. May. Decoding linear codes with high error rate and its impact for lpn security. In *International Conference on Post-Quantum Cryptography*, pages 25–46. Springer, 2018.
17. E. Boyle, G. Couteau, N. Gilboa, and Y. Ishai. Compressing vector OLE. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 896–912. ACM, 2018.
18. E. Boyle, G. Couteau, N. Gilboa, Y. Ishai, L. Kohl, N. Resch, and P. Scholl. Correlated pseudorandomness from expand-accumulate codes. In *Advances in Cryptology–CRYPTO 2022: 42nd Annual International Cryptology Conference,*

CRYPTO 2022, Santa Barbara, CA, USA, August 15–18, 2022, Proceedings, Part II, pages 603–633. Springer, 2022.

19. E. Boyle, G. Couteau, N. Gilboa, Y. Ishai, L. Kohl, P. Rindal, and P. Scholl. Efficient two-round ot extension and silent non-interactive secure computation. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 291–308, 2019.
20. E. Boyle, G. Couteau, N. Gilboa, Y. Ishai, L. Kohl, and P. Scholl. Efficient pseudorandom correlation generators: Silent ot extension and more. In *Advances in Cryptology–CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2019, Proceedings, Part III 39*, pages 489–518. Springer, 2019.
21. P. Briaud and M. Øygarden. A new algebraic approach to the regular syndrome decoding problem and implications for pcg constructions. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 391–422. Springer, 2023.
22. D. Bui and G. Couteau. Private set intersection from pseudorandom correlation generators. *IACR Cryptol. ePrint Arch.*, 2022:334, 2022.
23. A. Canteaut and F. Chabaud. A new algorithm for finding minimum-weight words in a linear code: Application to mceliece's cryptosystem and to narrow-sense bch codes of length 511. *IEEE Transactions on Information Theory*, 44(1):367–378, 1998.
24. R. Canto Torres and N. Sendrier. Analysis of information set decoding for a sublinear error weight. In *International Workshop on Post-Quantum Cryptography*, pages 144–161. Springer, 2016.
25. E. Carozza, G. Couteau, and A. Joux. Short signatures from regular syndrome decoding in the head. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 532–563. Springer, 2023.
26. K. Carrier, T. Debris-Alazard, C. Meyer-Hilfiger, and J.-P. Tillich. Statistical decoding 2.0: Reducing decoding to lpn. In *Advances in Cryptology–ASIACRYPT 2022: 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5–9, 2022, Proceedings, Part IV*, pages 477–507. Springer, 2022.
27. K. Carrier, T. Debris-Alazard, C. Meyer-Hilfiger, and J.-P. Tillich. Statistical decoding 2.0: Reducing decoding to lpn. *Asiacrpyt 2022*, 2022.
28. G. Couteau, P. Rindal, and S. Raghuraman. Silver: silent vole and oblivious transfer from hardness of decoding structured ldpc codes. In *Advances in Cryptology–CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part III*, pages 502–534. Springer, 2021.
29. I. Damgård and S. Park. Is public-key encryption based on LPN practical? *IACR Cryptol. ePrint Arch.*, 2012:699, 2012.
30. I. Damgård, V. Pastro, N. Smart, and S. Zakarias. Multiparty computation from somewhat homomorphic encryption. In *Advances in Cryptology–CRYPTO 2012: 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, pages 643–662. Springer, 2012.
31. S. Dittmer, Y. Ishai, S. Lu, and R. Ostrovsky. Authenticated garbling from simple correlations. In *Advances in Cryptology–CRYPTO 2022: 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15–18, 2022, Proceedings, Part IV*, pages 57–87. Springer, 2022.

32. S. Dittmer, Y. Ishai, S. Lu, and R. Ostrovsky. Improving line-point zero knowledge: Two multiplications for the price of one. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 829–841, 2022.
33. S. Dittmer, Y. Ishai, S. Lu, and R. Ostrovsky. Improving line-point zero knowledge: Two multiplications for the price of one. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 829–841, 2022.
34. S. Dittmer, Y. Ishai, and R. Ostrovsky. Line-point zero knowledge and its applications. *Cryptology ePrint Archive*, 2020.
35. Y. Dodis, E. Kiltz, K. Pietrzak, and D. Wichs. Message authentication, revisited. In D. Pointcheval and T. Johansson, editors, *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, volume 7237 of *Lecture Notes in Computer Science*, pages 355–374. Springer, 2012.
36. N. Döttling, S. Ghosh, J. B. Nielsen, T. Nilges, and R. Trifiletti. Tinyole: Efficient actively secure two-party computation from oblivious linear function evaluation. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS*, page 2263–2276, New York, NY, USA, 2017. Association for Computing Machinery.
37. A. Esser and P. Santini. Not just regular decoding: Asymptotics and improvements of regular syndrome decoding attacks. Cryptology ePrint Archive, Paper 2023/1568, 2023. https://eprint.iacr.org/2023/1568.
38. A. Esser and P. Santini. Not just regular decoding: asymptotics and improvements of regular syndrome decoding attacks. In *Annual International Cryptology Conference*, pages 183–217. Springer, 2024.
39. T. Feneuil, A. Joux, and M. Rivain. Syndrome decoding in the head: Shorter signatures from zero-knowledge proofs. In *Annual International Cryptology Conference*, pages 541–572. Springer, 2022.
40. N. Franzese, J. Katz, S. Lu, R. Ostrovsky, X. Wang, and C. Weng. Constant-overhead zero-knowledge for ram programs. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 178–191, 2021.
41. S. Ghosh, J. B. Nielsen, and T. Nilges. Maliciously secure oblivious linear function evaluation with constant overhead. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 629–659. Springer, 2017.
42. C. Hazay, P. Scholl, and E. Soria-Vazquez. Low cost constant round mpc combining bmr and oblivious transfer. *Journal of Cryptology*, 33(4):1732–1786, 2020.
43. Y. Ishai, M. Prabhakaran, and A. Sahai. Secure arithmetic computation with no honest majority. In *Theory of Cryptography Conference*, pages 294–314. Springer, 2009.
44. A. Jain, S. Krenn, K. Pietrzak, and A. Tentes. Commitments and efficient zero-knowledge proofs from learning parity with noise. In *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, volume 7658 of *Lecture Notes in Computer Science*, pages 663–680. Springer, 2012.
45. A. Jain, H. Lin, and A. Sahai. Indistinguishability obfuscation from well-founded assumptions. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 60–73, 2021.

46. M. Keller, E. Orsini, and P. Scholl. Mascot: faster malicious arithmetic secure computation with oblivious transfer. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 830–842, 2016.
47. E. Kiltz, K. Pietrzak, D. Venturi, D. Cash, and A. Jain. Efficient authentication from hard learning problems. *J. Cryptol.*, 30(4):1238–1275, 2017.
48. H. Liu, X. Wang, K. Yang, and Y. Yu. The hardness of lpn over any integer ring and field for pcg applications. Cryptology ePrint Archive, Paper 2022/712, 2022. https://eprint.iacr.org/2022/712.
49. H. Liu, X. Wang, K. Yang, and Y. Yu. The hardness of lpn over any integer ring and field for pcg applications. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 149–179. Springer, 2024.
50. V. Lyubashevsky and D. Masny. Man-in-the-middle secure authentication schemes from LPN and weak prfs. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 308–325. Springer, 2013.
51. A. May, A. Meurer, and E. Thomae. Decoding random linear codes in $\tilde{O}(2^{0.054n})$. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 107–124. Springer, 2011.
52. A. May and I. Ozerov. On computing nearest neighbors with applications to decoding of binary linear codes. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 203–228. Springer, 2015.
53. A. Meurer. *A coding-theoretic approach to cryptanalysis*. PhD thesis, Verlag nicht ermittelbar, 2013.
54. D. Micciancio and P. Mol. Pseudorandom knapsacks and the sample complexity of lwe search-to-decision reductions. In *Advances in Cryptology–CRYPTO 2011: 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings 31*, pages 465–484. Springer, 2011.
55. M. Naor and B. Pinkas. Oblivious polynomial evaluation. *SIAM Journal on Computing*, 35(5):1254–1281, 2006.
56. J. B. Nielsen, P. S. Nordholt, C. Orlandi, and S. S. Burra. A new approach to practical active-secure two-party computation. In *Advances in Cryptology–CRYPTO 2012: 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, pages 681–700. Springer, 2012.
57. C. Peters. Information-set decoding for linear codes over $\mathfrak{U}_q$. In *International Workshop on Post-Quantum Cryptography*, pages 81–94. Springer, 2010.
58. E. Prange. The use of information sets in decoding cyclic codes. *IRE Transactions on Information Theory*, 8(5):5–9, 1962.
59. S. Raghuraman and P. Rindal. Blazing fast psi from improved okvs and subfield vole. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 2505–2517, 2022.
60. P. Rindal and P. Schoppmann. Vole-psi: fast oprf and circuit-psi from vector-ole. In *Advances in Cryptology–EUROCRYPT 2021: 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17–21, 2021, Proceedings, Part II*, pages 901–930. Springer, 2021.
61. P. Schoppmann, A. Gascón, L. Reichert, and M. Raykova. Distributed vector-ole: improved constructions and implementation. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 1055–1072, 2019.
62. The Sage Developers. *SageMath, the Sage Mathematics Software System (Version x.y.z)*, YYYY. `https://www.sagemath.org`.

63. X. Wang, S. Ranellucci, and J. Katz. Authenticated garbling and efficient maliciously secure two-party computation. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pages 21–37, 2017.
64. X. Wang, S. Ranellucci, and J. Katz. Global-scale secure multiparty computation. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 39–56, 2017.
65. C. Weng, K. Yang, J. Katz, and X. Wang. Wolverine: fast, scalable, and communication-efficient zero-knowledge proofs for boolean and arithmetic circuits. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1074–1091. IEEE, 2021.
66. C. Weng, K. Yang, Z. Yang, X. Xie, and X. Wang. Antman: Interactive zero-knowledge proofs with sublinear communication. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 2901–2914, 2022.
67. K. Yang, P. Sarkar, C. Weng, and X. Wang. Quicksilver: Efficient and affordable zero-knowledge proofs for circuits and polynomials over any field. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 2986–3001, 2021.
68. K. Yang, X. Wang, and J. Zhang. More efficient mpc from improved triple generation and authenticated garbling. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1627–1646, 2020.
69. K. Yang, C. Weng, X. Lan, J. Zhang, and X. Wang. Ferret: Fast extension for correlated ot with small communication. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1607–1626, 2020.
70. Y. Yu and J. P. Steinberger. Pseudorandom functions in almost constant depth from low-noise LPN. In *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 154–183. Springer, 2016.
71. Y. Yu and J. Zhang. Cryptography with auxiliary input and trapdoor from constant-noise LPN. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 214–243. Springer, 2016.

A Algorithm for Regular-LPN over Binary Field

According to [21], the regularity of an error distribution τ provides an extra relations: For any $1 \leq i \leq t$, we observe that

$$1 - \sum_{j=1}^{\beta} e_{i,j} = 0,$$

where $e_{i,j}$ is the j-th entry of a vector $\mathbf{e}_i$. Consequently, the attack can easily get t linear equations from the regularity of $\tau_{m,t}$.

We can parse $\mathbf{b} = (\mathbf{b}_1 \| \cdots \| \mathbf{b}_t)$ where $\mathbf{b}_i \in \mathbb{F}_2^\beta$, and our attack exploits this fact as follows: For every $1 \leq i \leq t$,

$$\begin{aligned}\sum_{j=1}^{\beta} b_{i,j} &= (\sum_{j=1}^{\beta} \mathbf{a}_j^T) \cdot \mathbf{s} + \sum_{j=1}^{\beta} e_{i,j} \bmod 2 \\ &= (\sum_{j=1}^{\beta} \mathbf{a}_j^T) \cdot \mathbf{s} + 1 \bmod 2\end{aligned}$$

Here, $b_{i,j}$ is the j-th entry of $\mathbf{b}_i$, and $\mathbf{a}_j^T$ is the j-th row vector of $\mathbf{A}$. Thus, we automatically obtain an error-free polynomial g_i of the form

$$g_i(\mathbf{x}) = \sum_{j=1}^{\beta} b_{i,j} - 1 - (\sum_{j=1}^{\beta} \mathbf{a}_j^T) \cdot \mathbf{x}.$$

This exploitation can significantly enhance our attack. Specifically, the purpose of RP is to collect error-free polynomials, and the t error-free polynomials obtained from the regularity of $\tau_{m,t}$ reduce the number of error-free polynomials that we need to obtain. Consequently, we need only $n - t$ error-free polynomials to solve regular-LPN.

The remaining part is to obtain $n - t$ error-free polynomial through RP in Section 3.1.