The Hardness of LPN over Any Integer Ring and Field for PCG Applications

Hanlin Liu
Shanghai Jiao Tong University
hans1024@sjtu.edu.cn

Xiao Wang
Northwestern University
wangxiao@cs.northwestern.edu

Kang Yang
State Key Laboratory of Cryptology
yangk@sklc.org

Yu Yu
Shanghai Jiao Tong University
Shanghai Qizhi Institute
yuyu@yuyu.hk

Abstract

Learning parity with noise (LPN) has been widely studied and used in cryptography. It was recently brought to new prosperity since Boyle et al. (CCS'18), putting LPN to a central role in designing secure multi-party computation, zero-knowledge proofs, private set intersection, and many other protocols. In this paper, we thoroughly studied the concrete security of LPN problems in these settings. We found that many conclusions from classical LPN cryptanalysis do not apply to this new setting due to the low noise rates, extremely high dimensions, various types (in addition to $\mathbb{F}_2$) and noise distributions.

- For LPN over field $\mathbb{F}_q$, we give a parameterized reduction from an exact noise distribution to a regular one that not only generalizes the recent result by Feneuil, Joux and Rivain (Crypto'22), but also significantly reduces the security loss by paying only an additive price in dimension and number of samples.
- We analyze the security of LPN over a ring $\mathbb{Z}_{2^\lambda}$. Although existing protocols based on LPN over integer rings use parameters as if they are over fields, we found an attack that effectively reduces the weight of a noise by half compared to LPN over fields. Consequently, prior works that use LPN over $\mathbb{Z}_{2^\lambda}$ overestimate up to 40 bits of security.
- We provide a complete picture of the hardness of LPN over integer rings by showing: 1) the equivalence between its search and decisional versions; 2) an efficient reduction from LPN over $\mathbb{F}_2$ to LPN over $\mathbb{Z}_{2^\lambda}$; and 3) generalization of our results to any integer ring.
- For LPN over finite fields, we found that prior analysis ignored some important differences between classical LPN cryptanalysis and the new setting, leading to overly conservative parameters. We show that even after bringing all classical LPN cryptanalysis, including the latest SD 2.0 analysis (Asiacrypt'22), to the setting over finite fields, much less weight of noises is needed for the same level of security.

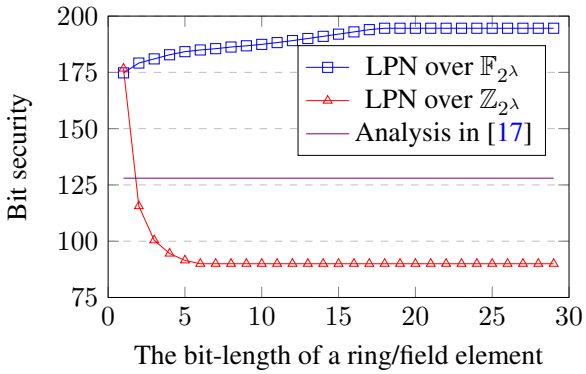
To improve the use of LPN assumptions for a wide range of cryptographic protocols, we provide an open-sourced script that estimates the concrete security of LPN over integer rings and finite fields.

1 Introduction

The learning parity with noise (LPN) assumption states that it is hard to distinguish LPN samples $(\mathbf{A}, \mathbf{A} \cdot \boldsymbol{s} + \boldsymbol{e})$ from random samples, where $\mathbf{A}$ is a public matrix, $\boldsymbol{s}$ is a random secret and $\boldsymbol{e}$ is a noise vector sampled from a sparse distribution. The LPN assumption has been applied to build various primitives, e.g., symmetric encryption and authentication (e.g., [52] and follow-up works), public key encryption [4], commitment

	Protocol	LPN type
[19, 79]	OT	$\mathbb{F}_2$
[70]	VOLE	$\mathbb{F}_{2^{61}-1}$ and $\mathbb{Z}_{2^{64}}$
[76]	ZK	$\mathbb{F}_2$ and $\mathbb{F}_{2^{61}-1}$
[45]	ZK	$\mathbb{F}_{2^{128}}$
[69, 23, 68]	PSI	$\mathbb{F}_{2^{128}}$
[12]	ZK	$\mathbb{F}_{2^{40}}$ and $\mathbb{F}_{2^{61}-1}$
[9]	ZK	$\mathbb{Z}_{2^{72}}$
[10]	ZK	$\mathbb{Z}_{2^{104}}$

(a) Prior works in the PCG framework and their required LPN variants over different fields and rings.



(b) **The bit-security from our analysis for LPN over $\mathbb{F}_{2^\lambda}$ and $\mathbb{Z}_{2^\lambda}$.** Parameters $N = 2^{10}, k = 652, t = 106$ are used.

Figure 1: LPN assumptions in prior work and our analysis on one set of parameters. For a set of parameters (N, k, t), N is the number of samples, k is the dimension and t is the Hamming weight of a noise vector.

scheme [55], garbled circuits [5], oblivious transfer [29] and collision-resistant hash functions [22, 81]. All these primitives adopt LPN over binary field $\mathbb{F}_2$ with moderate dimensions.

Recent works by Boyle et al. [17, 20] introduced the pseudorandom correlation generator (PCG) paradigm that can produce a large batch of correlated randomness, e.g., oblivious transfer (OT) and vector oblivious linear evaluation (VOLE), at a small cost. At the core of the PCG idea is to build a pseudorandom generator (PRG) with a simple internal structure from LPN-style assumptions and then privately evaluate such a PRG using function secret sharing [21]. The sparsity of a noise $\boldsymbol{e}$ translates to communication efficiency, while the efficiency of LPN encoding translates to computational efficiency. Later, the PCG paradigm was used to build a series of concretely efficient protocols with sublinear communication [70, 20, 79, 76, 26, 11] for generating correlated OT (COT) or VOLE correlations. These PCG-like protocols have gained a lot of interests in designing various concretely efficient protocols, including secure multi-party computation (MPC) (e.g., [28, 64, 75, 50, 27, 32]), zero-knowledge (ZK) proofs (e.g., [76, 12, 34, 78, 33, 11]), privacy-preserving machine learning [70, 77], private set intersection (PSI) [69, 23, 68], etc.

Although widely used in a lot of cryptographic protocols and real-world applications, many of them use LPN variants that have not attracted much attention in cryptanalysis, especially compared to the classical LPN assumption over $\mathbb{F}_2$ [4, 43, 48, 73]. Furthermore, prior analysis on the classical LPN assumption does not directly cover the LPN variants used in the PCG setting because of their unique features:

- Protocols often require an LPN assumption over a ring other than $\mathbb{F}_2$, including a finite field or even an integer ring [1] like $\mathbb{Z}_{2^\lambda}$.
- Most existing analyses focus on a Bernoulli noise distribution, but all PCG-like protocols adopt an exact noise distribution. Meanwhile, the difference in concrete security between the Bernoulli and exact noise is unclear.
- Most applications require an LPN assumption with very high dimension (e.g., millions) and low noise rate (e.g., $1/10^5$), which is out of the typically reported range of parameters considered for coding theoretic primitives.

At this point, all implementations of PCG use the LPN parameters from the original work by Boyle et al. [17], who analyzed the concrete security of LPN over $\mathbb{F}_{2^{128}}$. However, as we summarize in Table 1a,

[1] By integer ring we refer to $\mathbb{Z}_N$ for any composite number N, which is used to distinguish from polynomial rings.

LPN			This work					[17]
N	k	t	$\mathbb{F}_{2^{128}}$	$\mathbb{F}_{2^8}$	$\mathbb{Z}_{2^{128}}$	$\mathbb{Z}_4$	$\mathbb{F}_2$	Any field size
2^{10}	652	106	194	186	89	116	176	128
2^{12}	1589	172	155	146	76	95	131	128
2^{14}	3482	338	150	144	78	95	132	128
2^{16}	7391	667	151	148	82	99	135	128
2^{18}	15336	1312	153	153	87	104	139	128
2^{20}	32771	2467	155	157	92	108	143	128
2^{22}	67440	4788	156	160	97	113	147	128

Table 1: The comparison of our analysis and [17] about the bit-security of an LPN problem with dimension k, number of samples N and Hamming weight of noises t over different rings. Each noise vector is sampled uniformly at random such that it has a fixed Hamming weight t.

follow-up works used the same analysis to choose parameters for many different variants of LPN over $\mathbb{F}_2$, $\mathbb{F}_p$, and $\mathbb{Z}_{2^\lambda}$, many of which were not covered by the original analysis. It was not clear how large a gap in security when using LPN parameters over a field for LPN over another field or ring.

In addition, multiple PCG-like protocols [17, 20, 19, 79, 76, 26, 18] together with some MPC protocols such as [49] adopt a variant of exact noise distributions called regular noise distributions to achieve significantly better efficiency, where a regular noise distribution further divides a noise vector into t consecutive sub-vectors of size $\lfloor N/t \rfloor$ and each sub-vector has a single noisy coordinate, where N is the number of samples. The recent reduction by Feneuil et al. [42] from an exact noise distribution to a regular noise distribution incurs a large security loss, which makes the security gap between two noise distributions be still large.

1.1 Our Contribution

We conduct a systematic study on the LPN assumptions used by recent PCG-like protocols that can be in turn applied in MPC, ZK, PSI etc. For all relevant applications, each requiring a field or ring of different sizes, we provide concrete parameter recommendations by giving an open-sourced Python script [2]that estimates the bit-security of LPN problems over arbitrary integer rings and finite fields. The previous cryptanalysis provides a good coarse-level estimation but the exact bit-security can differ by 20 to 80 bits. In particular, the field size and the type of elements (finite field vs integer ring) hugely impact the security of the underlying LPN assumption. For example, as shown in Figure 1b [3], for a set of parameters that [17] estimates to have 128-bit security, our analysis shows that the bit-security of LPN over finite fields is about 175 to 194 bits and its security over integer rings is mostly below 90 bits when the ring size is larger than 2^6. These findings are a combination of correction/refinement from prior works as well as new attacks/reductions on LPN over rings, where we provide the upper bound and lower bound of their exact security. We also give a tighter reduction from an exact noise distribution to a regular noise distribution, and thus significantly shrink the security gap between two noise distributions.

The hardness of LPN under regular noise distributions. Recently, Feneuil et al. [42] observed that an exact noise vector (of Hamming weight t) is also regular with some probability (estimated to e^{-t} in Section 3), and thus (T,ϵ)-hard LPN under exact noise implies $(T,e^t\epsilon)$-hard LPN under regular noise. However, the

[2]This script can be found at https://gist.github.com/hansliu1024/21c87609e75f6cc52decdc69981e1d5b.

[3]In this figure and the following tables, the number of samples N and dimension k are adopted from [17], and the weight of a noise vector t in [17] is increased to make the parameters achieve 128-bit security according to the analysis [17].

dual-LPN			This work					[17]
n	N	t	$\mathbb{F}_{2^{128}}$	$\mathbb{F}_{2^8}$	$\mathbb{Z}_{2^{128}}$	$\mathbb{Z}_4$	$\mathbb{F}_2$	Any field size
2^{10}	2^{12}	88	208	198	99	122	177	128
2^{12}	2^{14}	83	200	190	101	126	175	128
2^{14}	2^{16}	78	195	184	104	128	175	128
2^{16}	2^{18}	73	190	180	104	126	173	128
2^{18}	2^{20}	68	186	176	106	127	169	128
2^{20}	2^{22}	63	182	172	107	127	165	128
2^{22}	2^{24}	58	177	168	109	126	162	128

Table 2: The comparison of our analysis and [17] about the bit-security of a dual-LPN problem with $n = N/4$ (corresponding to the number of COT/VOLE correlations), number of samples N, Hamming weight of noises t. Each noise vector is sampled uniformly at random such that it has a fixed weight t.

security loss is sometimes unaffordable as LPN may not have security beyond e^t in many practical settings. We introduce a tunable parameter $\alpha \geq 1$ and show that if the LPN over $\mathbb{F}_q$ with dimension k, sample number N, and exact weight-t noise is (T,ϵ)-hard, then the LPN over $\mathbb{F}_q$ with dimension $(k + \alpha t)$, sample number $(N + \alpha t)$, and regular weight-(αt) noise is $(T - \mathsf{poly}(k, N), 2^{\frac{t}{\alpha}+2} \cdot \epsilon)$-hard, where factor $1/\alpha$ allows to weaken the security loss at the exponent, while it adds only an additive term αt to the dimension, and the number of samples.

The hardness of LPN over integer rings. Although having been used in protocol design [70, 9, 10], the LPN assumption over an integer ring has only been studied in a handful of papers, without much concrete analysis. Akavia [2] studied the hardness of Learning Characters with Noise problem in the random samples access model, which generalizes LPN over any integer ring and noise of bounded Hamming weight.

All existing works select the parameters assuming that LPN over an integer ring is as secure as LPN over a finite field and then use the parameter recommendation from [17]. However, as our analysis shows in Figure 1b and in Tables 1 and 2, LPN over an integer ring is significantly more vulnerable to attacks than LPN over a finite field of similar size. *What's more, although LPN over a finite field becomes harder to attack as the field size increases, LPN over an integer ring becomes easier to attack as the ring size increases!* We briefly discuss our results below and summarize their relationships in Figure 2.

1. Focusing on the most commonly used ring $\mathbb{Z}_{2^\lambda}$, we show a concrete attack that can solve a t-noise LPN over $\mathbb{Z}_{2^\lambda}$ by solving a $\left(\frac{2^{(\lambda-1)}}{2^\lambda-1} \cdot t\right)$-noise (which approximates to $t/2$) LPN over $\mathbb{F}_2$. This means that LPN over an integer ring is concretely weaker than LPN over a finite field and we need to double the weight of noise vectors to cover this attack. The impact to existing cryptographic protocols is significant. It will lead to roughly $2\times$ more communication and computation.

2. On the positive side, we provide evidence that the LPN assumption over an integer ring is generally hard. In particular, we show a reduction between t-noise LPN over $\mathbb{F}_2$ and $(\lambda \cdot t)$-noise LPN over a ring $\mathbb{Z}_{2^\lambda}$, which means that LPN over an integer ring is asymptotically as hard as classical LPN. This "efficient" reduction requires a different noise distribution: instead of sampling t locations and putting a uniform non-zero entry from $\mathbb{Z}_{2^\lambda}$ in each location, we need to sample λ weight-t noise vectors $\boldsymbol{e}_0, \ldots, \boldsymbol{e}_{\lambda-1}$ over $\mathbb{F}_2$, and define the final noise vector as $\boldsymbol{e} = \sum_{i\in[\lambda]} 2^i \cdot \boldsymbol{e}_i$ with weight $\leq \lambda \cdot t$. This noise distribution may be interesting, as it can be used in the design of PCG-like protocols by adopting the upper bound $\lambda \cdot t$ to run these protocols. This change of distributions is crucial: without such change, the best reduction that we can find goes from t-noise LPN over $\mathbb{F}_2$ to $(2^\lambda \cdot t)$-noise LPN over $\mathbb{Z}_{2^\lambda}$, which is exponentially

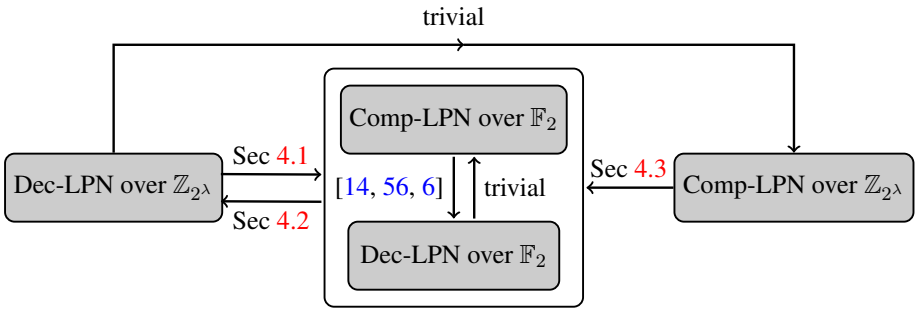


Figure 2: The reduction relations between computational and decisional versions of LPN over $\mathbb{F}_2$ and $\mathbb{Z}_{2^\lambda}$ in the presence of Bernoulli and exact noise distribution.

worse than the above. Another interesting fact is that the above reductions only require the code matrix $\mathbf{A}$ to be Boolean, which eliminates the need for integer multiplication during encoding. Prior work [26] observed that using a Boolean code matrix is not vulnerable to existing linear-test attacks for LPN over finite fields; here we show that for LPN over integer rings, using a Boolean matrix is provably secure assuming that classical LPN over $\mathbb{F}_2$ is hard.

3. While the above reductions focus on the decisional version of LPN, we also give a reduction from computational LPN over $\mathbb{Z}_{2^\lambda}$ to that over $\mathbb{F}_2$. Thus, we show the equivalence between computational and decisional versions of LPN over $\mathbb{Z}_{2^\lambda}$ as shown in Figure 2. We also generalize all the results to any integer ring. In particular, we show a concrete attack that can solve a t-noise LPN over ring $\mathbb{Z}_{p^{\lambda_1}q^{\lambda_2}}$ by solving either a $\left(\frac{p-1}{p}\cdot t\right)$-noise LPN over $\mathbb{F}_p$ or a $\left(\frac{q-1}{q}\cdot t\right)$-noise LPN over $\mathbb{F}_q$, where p, q are two primes. This attack works for both computational and decisional versions of LPN. We also give a reduction from t-noise LPN over $\mathbb{F}_p$ and t-noise LPN over $\mathbb{F}_q$ to $\big((\lambda_1+\lambda_2)\cdot t\big)$-noise LPN over $\mathbb{Z}_{p^{\lambda_1}q^{\lambda_2}}$. Given these reductions over $\mathbb{Z}_{p^{\lambda_1}q^{\lambda_2}}$, one can easily generalize them to any integer ring.

The hardness of LPN over finite fields. When $|\mathbb{F}| > 2$, [17] is the main work that provides comprehensive cryptanalysis in this setting. They analyzed the concrete security of LPN instances over a large field by considering Pooled Gauss (called Gaussian elimination in [17]), statistical decoding (SD, a.k.a., low-weight parity check) and information set decoding (ISD) attacks. Our new analysis shows that the parameters of LPN over finite fields given in [17] are overly conservative (see Table 1 and Table 2). Their analysis shows that the low-weight parity check is almost always the best attack and the attack cost does not change when changing the field size. We find that [17] ignored some important differences between classical LPN cryptanalysis and the new setting, and have the following:

1. Concurrent to our work, Carrier et al. [24] modified the framework of the SD attack and proposed a new variant, called the SD 2.0 algorithm. Under the Gilbert-Varshamov (GV) bound noise regime [4], the attack even outperforms all ISD algorithms for $k < 0.3N$, e.g., $(N, k, t) = (2^{14}, 3482, 3850)$ or $(2^{16}, 7391, 19900)$. However, we observe that the SD 2.0 algorithm [24] does not behave well in solving the low-noise LPN problems underlying the PCG-like protocols, because the collision technique used by the SD 2.0 algorithm takes exponential time $|\mathbb{F}|^{\theta(k)}$ that is much larger than the subexponential security $2^{O(k\mu)}$ of the low-noise LPN assumption, where dimension k and low noise rate $\mu = k^{-c}$ for constant $0 < c < 1$. Inspired by this observation, we propose an adapted algorithm to analyze the low-noise LPN assumption by replacing it with other collision techniques, e.g., [17].

[4]The GV bound decoding over $\mathbb{F}_2$ is to solve LPN instances that achieve the GV relative distance $t/N = \mathbf{H}^{-1}(1 - k/N)$, known as the hardest non-trivial instance of interest for classical LPN cryptanalyses, where $\mathbf{H}(\cdot)$ is the binary entropy function.

For all the LPN parameters given in [17], we show that SD attack, even taking into account the adapted SD 2.0 algorithm, has the largest cost (see Table 3 and Table 4 of Section 5). Furthermore, we prove that both an optimal SD attack and SD 2.0 attack (adapted to the low-noise setting) require more cost than the original Prange's ISD algorithm [67] for a large set of commonly used parameters: $|\mathbb{F}| = k^{\omega(1)}$, the Hamming weight of the noise $t = o(N)$ and the number of samples $(1+\beta)\cdot k \leq N = \mathsf{poly}(k)$ for constant $\beta > 0$. We note that our results do not invalidate SD 2.0 due to the different parameter settings considered in [24] and by us. This again shows the disparity between classical LPN cryptanalysis and ones used for MPC.

2. All the attacks listed in [17] do not take into account the sizes of fields. As a result, follow-up works also use the same analysis formula for fields of other sizes. We show that ISD (including Pooled Gauss as its special case) is the best attack to solve the low-noise LPN problems over finite fields, and the cost of ISD attack increases as the field size increases. We also find that the advantage of the advanced ISD algorithm over Pooled Gauss attack gradually disappears as the field size increases (see Figure 3 of Section 5). Particularly, when the field size is large enough, ISD has the same cost as Pooled Gauss.

Compared to the analysis [17], our new analysis shows that one can reduce the weight parameter t by $20\% - 40\%$ when keeping the same security level. This directly allows us to obtain the efficiency improvement of PCG-like COT and VOLE protocols, which in turn improves the MPC, ZK and PSI applications.

2 Preliminary

2.1 Notation

We will use κ and ρ to denote the computational and statistical security parameters, respectively. We denote by $\log$ the logarithm in base 2. For $a, b \in \mathbb{N}$ with $a \leq b$, we write $[a, b] = \{a, \ldots, b-1\}$ and use $[n]$ to denote $[0, n-1]$ for simplicity. We use $x \leftarrow S$ to denote sampling x uniformly at random from a set S and $x \leftarrow \mathcal{D}$ to denote sampling x according to a distribution $\mathcal{D}$. For a ring $\mathcal{R}$, we denote by $|\mathcal{R}|$ the size of $\mathcal{R}$. By slightly abusing the notation, for a vector $\boldsymbol{a}$, we use $|\boldsymbol{a}|$ to denote the Hamming weight of $\boldsymbol{a}$, and denote by $\boldsymbol{a}[i]$ the i-th component of $\boldsymbol{a}$. For a vector $\boldsymbol{a} \in (\mathbb{Z}_{2^\lambda})^k$, we use $\mathsf{BitDecomp}(\boldsymbol{a})$ to denote the bit-decomposition of $\boldsymbol{a}$, and the output of $\mathsf{BitDecomp}(\boldsymbol{a})$ is denoted by $(\boldsymbol{a}^0, \boldsymbol{a}^1, \cdots, \boldsymbol{a}^{\lambda-1})$ such that $\boldsymbol{a}^i \in \mathbb{F}_2^k$ for each $i \in [\lambda]$ and $(\boldsymbol{a}^0[j], \boldsymbol{a}^1[j], \ldots, \boldsymbol{a}^{\lambda-1}[j])$ is the bit-decomposition of ring element $\boldsymbol{a}[j] \in \mathbb{Z}_{2^\lambda}$ for each $j \in [k]$. Let $\mathsf{BitDecomp}^{-1}(\boldsymbol{a}^0, \boldsymbol{a}^1, \cdots, \boldsymbol{a}^{\lambda-1}) = \sum_{i=0}^{\lambda-1} 2^i \cdot \boldsymbol{a}^i \in (\mathbb{Z}_{2^\lambda})^k$ be the inverse of $\mathsf{BitDecomp}(\boldsymbol{a})$. We use $\mathsf{poly}(\cdot)$ to denote a polynomial function. For $0 < \mu < 1/2$, let $\mathbf{H}(\mu) \stackrel{\text{def}}{=} \mu \cdot \log(1/\mu) + (1-\mu)\cdot\log(1/(1-\mu))$ be the binary entropy function. We will use the following lemmas throughout the paper:

Lemma 1 (Asymptotics for binomial coefficients (see, e.g., [46]))**.** *For any $0 < \mu < 1/2$, we have $\binom{n}{n\mu} = 2^{n\mathbf{H}(\mu) - \frac{\log n}{2} + O(1)}$, where binary entropy $\mathbf{H}(\mu) \in \left(\mu\log(1/\mu), \mu(\log(1/\mu) + \frac{3}{2})\right)$.*

Lemma 2 (see, e.g., [80])**.** *For any $\mu \in (0,1)$, if each coordinate of a vector $\boldsymbol{v} \in \mathbb{F}_2^t$ is independently set to 1 with probability μ, then the probability that the Hamming weight of $\boldsymbol{v}$ is equal to $\lceil \mu t \rceil$ is at least $\Omega(1/\sqrt{t})$.*

2.2 Learning Parity with Noise

Recently, variants of the Learning Parity with Noise (LPN) assumption [14] are used to build PCG-like protocols with sublinear communication for generating (C)OT and (V)OLE correlations. The LPN variants are defined over a general finite ring $\mathcal{R}$. The known LPN-based protocols mainly consider three cases for the choices of ring $\mathcal{R}$:

- Case 1 that $\mathcal{R} = \mathbb{F}_2$ is used to design the COT protocols [20, 19, 79, 26], which is in turn able to be transformed into standard OT protocols.
- Case 2 that $\mathcal{R}$ is a large finite field $\mathbb{F}$ with $|\mathbb{F}| \geq 2^\rho$ is used to construct the VOLE protocols [17, 19, 70, 76, 26] and the OLE protocol [20].
- Case 3 that $\mathcal{R} = \mathbb{Z}_{2^\lambda}$ (e.g., $\lambda \in \{32, 64, 128\}$) is used to obtain the VOLE protocols [70, 9].

For the other case that $\mathcal{R}$ is a small finite field (e.g., $\mathcal{R} = \mathbb{F}_{2^8}$), it is still interesting to design some subfield VOLE protocols [70, 76]. When considering some more general rings such as $\mathcal{R} = \mathbb{Z}_{p^\lambda}$ for a prime $p > 2$ and $\mathcal{R} = \mathbb{Z}_{p^{\lambda_1} q^{\lambda_2}}$ for two primes p, q, the LPN problems over such rings may be useful for future protocols.

Following the previous work (e.g., [20, 19]), we define the (primal-)LPN and dual-LPN assumptions over a general ring $\mathcal{R}$ as follows:

Definition 1 (LPN). *Let $\mathcal{D}(\mathcal{R}) = \{\mathcal{D}_{t,N}(\mathcal{R})\}_{t,N\in\mathbb{N}}$ denote a family of distributions over a ring $\mathcal{R}$ such that for any $t, N \in \mathbb{N}$, $\mathsf{Im}(\mathcal{D}_{t,N}(\mathcal{R})) \subseteq \mathcal{R}^N$. Let $\mathbf{C}$ be a probabilistic code generation algorithm such that $\mathbf{C}(k, N, \mathcal{R})$ outputs a matrix $\mathbf{A} \in \mathcal{R}^{N\times k}$. For dimension $k = k(\kappa)$, number of samples $N = N(\kappa)$, Hamming weight of a noise vector $t = t(\kappa)$, and a ring $\mathcal{R}$, we say that the decisional $(\mathcal{D}, \mathbf{C}, \mathcal{R})$-$\mathsf{LPN}(k, N, t)$ problem is (T, ϵ)-hard if for every probabilistic distinguisher $\mathcal{B}$ running in time T, we have*

$$\left| \Pr_{\mathbf{A},\boldsymbol{s},\boldsymbol{e}} \left[\mathcal{B}(\mathbf{A}, \boldsymbol{b} \stackrel{def}{=} \mathbf{A}\cdot\boldsymbol{s} + \boldsymbol{e}) = 1\right] - \Pr_{\mathbf{A},\boldsymbol{u}} \left[\mathcal{B}(\mathbf{A}, \boldsymbol{u}) = 1\right] \right| \leq \epsilon,$$

where $\mathbf{A} \leftarrow \mathbf{C}(k, N, \mathcal{R})$, $\boldsymbol{s} \leftarrow \mathcal{R}^k$, $\boldsymbol{e} \leftarrow \mathcal{D}_{t,N}(\mathcal{R})$ and $\boldsymbol{u} \leftarrow \mathcal{R}^N$. We say that the computational $(\mathcal{D}, \mathbf{C}, \mathcal{R})$-$\mathsf{LPN}(k, N, t)$ problem is (T, ϵ)-hard if for every probabilistic algorithm $\mathcal{B}$ running in time T, we have

$$\Pr_{\mathbf{A},\boldsymbol{s},\boldsymbol{e}} \left[\mathcal{B}(\mathbf{A}, \boldsymbol{b} \stackrel{def}{=} \mathbf{A}\cdot\boldsymbol{s} + \boldsymbol{e}) = (\boldsymbol{s}, \boldsymbol{e})\right] \leq \epsilon,$$

where $\mathbf{A}, \boldsymbol{s}, \boldsymbol{e}$ are defined as above.

In the above definition, both T and ϵ are functions of computational security parameter κ. We mainly consider the following families of noise distributions:

- **Bernoulli.** Let $\mathsf{Ber}(\mathcal{R}) = \{\mathsf{Ber}_{\mu,N}(\mathcal{R})\}_{\mu,N}$ be the family of Bernoulli distributions. In particular, $\mathsf{Ber}_{\mu,N}(\mathcal{R})$ is a Bernoulli distribution with parameters μ, N over a ring $\mathcal{R}$, such that each component in a vector sampled from $\mathsf{Ber}_{\mu,N}(\mathcal{R})$ is a uniform element in $\mathcal{R}$ with probability μ and 0 otherwise. The expected Hamming weight of the sampled vector is $t = \mu N(|\mathcal{R}| - 1)/|\mathcal{R}|$. Note that the definition is equivalent to sampling a uniform *non-zero* element in $\mathcal{R}$ with probability $\mu(|\mathcal{R}| - 1)/|\mathcal{R}|$ for each component.
- **Exact.** Let $\mathsf{HW}(\mathcal{R}) = \{\mathsf{HW}_{t,N}(\mathcal{R})\}_{t,N}$ be the family of exact distributions. In particular, each component of a noise vector sampled from $\mathsf{HW}_{t,N}(\mathcal{R})$ is a uniform non-zero element in t random positions and zero elsewhere. To simplify the description, we often refer to such a distribution as an *exact* noise distribution.
- **Regular.** To achieve better efficiency, prior works [49, 20, 19, 79, 76] further consider the family of *regular* noise distributions, denoted by $\mathsf{RHW}(\mathcal{R}) = \{\mathsf{RHW}_{t,N}(\mathcal{R})\}_{t,N}$. In addition to fixed Hamming weight, the noise vector further divides into t consecutive sub-vectors of size $\lfloor N/t \rfloor$, where each sub-vector has a single noisy coordinate. We are *not* aware of any non-trivial attacks that can utilize the regular structure beyond reducing the dimension by t, which has also been noted by previous work, e.g., [49, 17, 20, 19]. Thus, our analysis of the hardness of (dual-)LPN for exact noise distributions (as shown in Section 5) has covered the case of regular noise distributions.

The existing LPN-based PCG-like protocols adopt the latter two noise distributions, and the standard LPN assumption adopts the Bernoulli distribution. While the standard LPN assumption uses random linear codes to instantiate $\mathbf{C}$ (i.e., sampling $\mathbf{A}$ uniformly at random), these LPN-based protocols adopt other kinds of linear codes to obtain faster computation. The other linear codes to instantiate $\mathbf{C}$ include: local linear codes [4], LDPC codes and its variant [54, 26], quasi-cyclic codes [1], MDPC codes [63], Druk-Ishai codes [35], etc. In this paper, we did not analyze the hardness of the LPN problems based on quasi-cyclic codes, which needs to take into account the effect of the DOOM attack [71] that allows providing $\sqrt{N}$ computational speedup. As far as we know, other kinds of linear codes listed above seem *not* to lead to significantly better attacks, compared to random linear codes. To simplify the notation, we often omit $\mathbf{C}$ from the $(\mathcal{D}, \mathbf{C}, \mathcal{R})$-$\mathsf{LPN}(k, N, t)$ problem, and only write $(\mathcal{D}, \mathcal{R})$-$\mathsf{LPN}(k, N, t)$.

Below, we define the dual-LPN assumption over a general finite ring $\mathcal{R}$ with a family $\mathcal{D}$ of noise distributions, where both the decisional version and search version are described. The dual-LPN is also known as syndrome decoding.

Definition 2 (Dual LPN)**.** *Let $\mathcal{D}(\mathcal{R})$ and $\mathbf{C}$ be as in Definition 1. For two integers N, n with $N > n$, we define*

$$\mathbf{C}^{\perp}(N, n, \mathcal{R}) = \left\{\mathbf{H} \in \mathcal{R}^{n \times N} : \mathbf{H} \cdot \mathbf{A} = 0,\ \mathbf{A} \in \mathbf{C}(N-n, N, \mathcal{R}), \mathsf{rank}(\mathbf{H}) = n\right\}.$$

For output length $n = n(\kappa)$, number of samples $N = N(\kappa)$, noise-vector Hamming weight $t = t(\kappa)$, we say that the decisional $(\mathcal{D}, \mathbf{C}^{\perp}, \mathcal{R})$-$\mathsf{dual\text{-}LPN}(N, n, t)$ problem is (T, ϵ)-hard if for every PPT distinguisher $\mathcal{B}$ running in time T:

$$\left|\Pr_{\mathbf{H}, \boldsymbol{e}}\left[\mathcal{B}(\mathbf{H}, \mathbf{H} \cdot \boldsymbol{e}) = 1\right] - \Pr_{\mathbf{H}, \boldsymbol{u}}\left[\mathcal{B}(\mathbf{H}, \boldsymbol{u}) = 1\right]\right| \leq \epsilon,$$

where $\mathbf{H} \leftarrow \mathbf{C}^{\perp}(N, n, \mathcal{R})$, $\boldsymbol{e} \leftarrow \mathcal{D}_{t,N}(\mathcal{R})$ and $\boldsymbol{u} \leftarrow \mathcal{R}^N$.
We say that the computational $(\mathcal{D}, \mathbf{C}^{\perp}, \mathcal{R})$-$\mathsf{dual\text{-}LPN}(N, n, t)$ problem is (T, ϵ)-hard if for every probabilistic algorithm $\mathcal{B}$ running in time T, we have

$$\Pr_{\mathbf{H}, \boldsymbol{e}}\left[\mathcal{B}(\mathbf{H}, \mathbf{H} \cdot \boldsymbol{e}) = \boldsymbol{e}\right] \leq \epsilon,$$

where $\mathbf{H}, \boldsymbol{e}$ are defined as above.

For any fixed code generation algorithm $\mathbf{C}$ and noise distribution $\mathcal{D}$, the dual-LPN problem defined as above is equivalent to the primal-LPN problem from Definition 1 with dimension $k = N - n$ and the number of samples N. The direction transforming an LPN instance into a dual-LPN instance directly follows the simple fact that $\mathbf{H} \cdot (\mathbf{A} \cdot \boldsymbol{s} + \boldsymbol{e}) = (\mathbf{H} \cdot \mathbf{A}) \cdot \boldsymbol{s} + \mathbf{H} \cdot \boldsymbol{e} = \mathbf{H} \cdot \boldsymbol{e}$, as $\mathbf{H}$ is the parity-check matrix of the code generated by $\mathbf{A}$. The reverse direction can be obtained in a way similar to [62, Lemma 4.9]. As such, we often write the $(\mathcal{D}, \mathcal{R})$-$\mathsf{dual\text{-}LPN}(N, n, t)$ problem for simplicity. We give an overview of some attacks on the LPN problem in Appendix C.

3 The Hardness of LPN with Regular Noise Distributions

Many MPC protocols [49, 20, 19, 79, 76] rely on the hardness of low-noise LPN under the regular noise distribution $(\mathsf{RHW}, \mathbb{F}_q)$-$\mathsf{LPN}(k, N, t)$, where the noise vector is divided into t blocks, each of length $\lfloor N/t \rfloor$ and Hamming weight 1. Intuitively, LPN with regular noise may look easier than that with exact noise. For example, the LPN over $\mathbb{F}_2$ with regular noise simply leaks t parity bits of its secret and thus reduces the dimension by t.

Recently, Feneuil et al. [42] introduced a reduction from the (dual)-LPN with a regular noise distribution to that with an exact noise distribution[5].

[5] In particular, [42] considers a d-split noise, which consists of d blocks of length N/d and each block has weight t/d. For $d = t$, it corresponds to the regular noise.

Theorem 1 (Theorem 1 of [42], adapted)**.** *If the* $(\mathsf{HW}, \mathbb{F}_q)$-problem *is* (T, ϵ)*-hard, then the* $(\mathsf{RHW}, \mathbb{F}_q)$-problem *is* $(T, \epsilon \cdot \binom{N}{t} / (\frac{N}{t})^t)$*-hard, where the statement holds for both versions of LPN, i.e.,* problem $\in \{\mathsf{dual\text{-}LPN}(N, n, t), \mathsf{LPN}(k, N, t)\}$.

The reduction suffers significant security loss, i.e., the penalty factor

$$p_t = \binom{N}{t} \Big/ \Big(\frac{N}{t}\Big)^t = (\frac{t^t}{t!}) \cdot \prod_{i=1}^{t-1}(1 - \frac{i}{N}) = e^{t - \Theta(\ln t) - \Theta(t^2/N)} = e^{t(1-o(1))}.$$

where the approximation is based on Stirling's $\ln(t!) = t \ln t - t + \Theta(\ln t)$, $t = o(N)$, and $4^{-x} \leq 1 - x \leq e^{-x}$ for $0 \leq x \leq 1/2$. Meanwhile, it is not hard to see that for many non-trivial parameter choices $\epsilon > e^{-t}$. Consider the dual LPN problem

$$[\mathbf{H}_1, \mathbf{H}_2] \cdot (\boldsymbol{e}_1 \| \boldsymbol{e}_2) = \mathbf{H}_1 \boldsymbol{e}_1 + \mathbf{H}_2 \boldsymbol{e}_2 \ = \boldsymbol{y}$$

where $\mathbf{H}_1 \in \mathbb{F}_q^{n \times n}$, $\mathbf{H}_2 \in \mathbb{F}_q^{n \times (N-n)}$, $\boldsymbol{e}_1 \in \mathbb{F}_q^n$ and $\boldsymbol{e}_2 \in \mathbb{F}_q^{N-n}$. A polynomial-time attack simply bets $\boldsymbol{e}_2 = \mathbf{0}$ and computes $\boldsymbol{e}_1 = \mathbf{H}_1^{-1} \boldsymbol{y}$ (assuming WLOG invertible $\mathbf{H}_1$), which succeeds with probability $\binom{n}{t} / \binom{N}{t} = \prod_{i=1}^{N-n} (1 - \frac{t}{n+i}) > e^{-\frac{t(N-n)}{n+1}}$. E.g., for $N \leq 2n$ the term $\epsilon > e^{-t}$ cannot afford multiplying with p_t, which renders the bound meaningless. This is just the ϵ for $T = \mathsf{poly}(n, N)$, and for larger T it only becomes worse due to the known optimizations and trade-offs between T and ϵ (see, e.g., [73]). The case for (primal-)LPN is likewise.

In retrospect, [42, Theorem 1] incurs significant loss because it simply uses $1/p_t$ to account for the probability that an exact noise vector is regular at the same time. We provide a new reduction below with a new parameter α such that [42, Theorem 1] becomes a corollary for $\alpha = 1$. More importantly, with large α we are able to reduce the security loss dramatically by dividing the exponent by α, while paying only an additive price αt in dimension and number of samples.

Theorem 2. *Let* $t, N \in \mathbb{N}$*, and* $\alpha \geq 1$ *such that* $\alpha t \in \mathbb{N}$ *and* $(\alpha t) | N$*. If the* $(\mathsf{HW}, \mathbb{F}_q)$-$\mathsf{LPN}(k, N, t)$ *is* (T, ϵ)*-hard, then the* $(\mathsf{RHW}, \mathbb{F}_q)$-$\mathsf{LPN}(k + \alpha t, N + \alpha t, \alpha t)$ *is* $(T - \mathsf{poly}(k, N), 2^{\frac{t}{\alpha}+2} \cdot \epsilon)$*-hard.*

Proof. Let $N = \alpha t m$ for $m \in \mathbb{N}$. On input of $(\mathsf{HW}, \mathbb{F}_q)$-$\mathsf{LPN}(k, N, t)$ samples parsed as

$$\mathbf{A} \stackrel{\text{def}}{=} \begin{bmatrix} \mathbf{A}_1 \\ \vdots \\ \mathbf{A}_{\alpha t} \end{bmatrix}, \boldsymbol{b} \stackrel{\text{def}}{=} \begin{bmatrix} \boldsymbol{b}_1 = \mathbf{A}_1 \boldsymbol{s} + \boldsymbol{e}_1 \\ \vdots \\ \boldsymbol{b}_{\alpha t} = \mathbf{A}_{\alpha t} \boldsymbol{s} + \boldsymbol{e}_{\alpha t} \end{bmatrix},$$

where each $\mathbf{A}_i \in \mathbb{F}_q^{m \times k}$ and $\boldsymbol{e}_i \in \mathbb{F}_q^m$. Our analysis is conditioned on the event $\mathcal{E}$ that $|\boldsymbol{e}_i| \leq 1$ for every $1 \leq i \leq \alpha t$, which occurs with probability

$$\Pr_{(\boldsymbol{e}_1, \ldots, \boldsymbol{e}_{\alpha t}) \leftarrow \mathsf{HW}_{t,N}(\mathbb{F}_q)}[\mathcal{E}] = \frac{\binom{\alpha t}{t} \cdot \left(\frac{N}{\alpha t}\right)^t}{\binom{N}{t}} = \prod_{i=1}^{t-1} \frac{(1 - \frac{i}{\alpha t})}{(1 - \frac{i}{N})} \geq 2^{1 - \frac{t}{\alpha}} \ .$$

Sample matrix $\mathbf{B} \in \mathbb{F}_q^{k \times (k+\alpha t)}$, row vectors $\boldsymbol{r}_1^\mathsf{T}, \ldots, \boldsymbol{r}_{\alpha t}^\mathsf{T} \in \mathbb{F}_q^{k+\alpha t}$ uniformly at random. For each i, define $\mathbf{C}_i \stackrel{\text{def}}{=} \begin{bmatrix} \mathbf{A}_i \mathbf{B} \\ \boldsymbol{r}_i^\mathsf{T} - \mathbf{1}^\mathsf{T}(\mathbf{A}_i \mathbf{B}) \end{bmatrix}$ and $\boldsymbol{b}'_i \stackrel{\text{def}}{=} \begin{bmatrix} \boldsymbol{b}_i \\ \boldsymbol{r}_i^\mathsf{T} \boldsymbol{x} - \mathbf{1}^\mathsf{T} \boldsymbol{b}_i + u_i \end{bmatrix}$, where $\boldsymbol{x}$ is uniform over $\mathbb{F}_q^{k+\alpha t}$ (subject to $\boldsymbol{s} = \mathbf{B}\boldsymbol{x}$), $\mathbf{1}^\mathsf{T}$ is the all-ones row vector (whose every coordinate is 1), u_i is the non-zero entry of $\boldsymbol{e}_i$ if $|\boldsymbol{e}_i| = 1$, and otherwise u_i is uniformly over $\mathbb{F}_q / \{0\}$. We can verify that conditioned on $\mathcal{E}$ the shuffled

samples of $(\mathbf{C_i}, \boldsymbol{b}'_i = \mathbf{C}_i\boldsymbol{x} + \begin{bmatrix} \boldsymbol{e}_i \\ u_i - \mathbf{1}^\mathsf{T}\boldsymbol{e}_i \end{bmatrix})$ are identically distributed to the i-th ($1 \le i \le \alpha t$) block of the $(\mathsf{RHW}, \mathbb{F}_q)$-$\mathsf{LPN}(k+t, N+t, t)$ instance, where $\begin{bmatrix} \boldsymbol{e}_i \\ u_i - \mathbf{1}^\mathsf{T}\boldsymbol{e}_i \end{bmatrix}$ is of Hamming weight 1 by the definition of u_i. Therefore, we just feed all the shuffled $(\mathbf{C}_i, \boldsymbol{b}'_i)$ samples to the $(\mathsf{RHW}, \mathbb{F}_q)$-LPN solver and then recovers $\boldsymbol{s} = \mathbf{B}\boldsymbol{x}$. Note that we can just use random elements (over $\mathbb{F}_q$) as values for $(\boldsymbol{r}_i^\mathsf{T}\boldsymbol{x} - \mathbf{1}^\mathsf{T}\boldsymbol{b}_i + u_i)$ since $(\mathbf{B}\boldsymbol{x}, \boldsymbol{r}_1^\mathsf{T}\boldsymbol{x}, \ldots, \boldsymbol{r}_{\alpha t}^\mathsf{T}\boldsymbol{x})$ is uniformly random over $\mathbb{F}_q^{k+\alpha t}$ as long as the square matrix $[\mathbf{B}^\mathsf{T}, \boldsymbol{r}_1, \ldots, \boldsymbol{r}_\alpha]$ has full rank, which is almost true. E.g., consider $q = 2$ as the worst case[6], where $[\mathbf{B}^\mathsf{T}, \boldsymbol{r}_1, \ldots, \boldsymbol{r}_{\alpha-\sigma}]$ has full rank with probability at least $1 - 2^{-\sigma+1}$ (see, e.g., [80, Fact A.2]), and the rest σ bits will be hit with probability $2^{-\sigma}$. This yields another factor $2^{-\sigma}(1 - 2^{-\sigma+1})$ that reaches its maximum 1/8 when $\sigma = 2$. Therefore, the overall security loss factor is $2^{1-\frac{t}{\alpha}}/8 = 2^{-\frac{t}{\alpha}-2}$. □

We get similar result for dual-LPN in Corollary 1 via a simple reduction from LPN to dual-LPN, i.e, $(\mathbf{A}, \boldsymbol{b} = \mathbf{A} \cdot \boldsymbol{s} + \boldsymbol{e})$ can be efficiently transformed into a dual-LPN instance $(\mathbf{H}, \boldsymbol{y} = \mathbf{H} \cdot \boldsymbol{b} = \mathbf{H} \cdot \boldsymbol{e})$ due to the duality between $\mathbf{H}$ and $\mathbf{A}$, regardless of the distribution of $\mathbf{e}$.

Corollary 1. *Let $t, N \in \mathbb{N}$ and $\alpha \ge 1$ such that $\alpha t \in \mathbb{N}$ and $(\alpha t)|N$. If the $(\mathsf{HW}, \mathbb{F}_q)$-$\mathsf{LPN}(k, N, t)$ is (T,ϵ)-hard, then the $(\mathsf{RHW}, \mathbb{F}_q)$-dual-$\mathsf{LPN}(N + \alpha t, N - k, \alpha t)$ is $(T - \mathsf{poly}(k, N), 2^{\frac{t}{\alpha}+2} \cdot \epsilon)$-hard.*

4 The Hardness of LPN over Integer Rings

LPN over an integer ring has been used in multiple prior works. For example, Schoppmann et al. [70] used it to construct VOLE over $\mathbb{Z}_{2^{64}}$, which can then be used for performing private machine learning tasks in the semi-honest setting. Recently, Baum et al. [9, 10] proposed VOLE protocols based on LPN over a ring $\mathbb{Z}_{2^\lambda}$, and used them to construct concretely efficient ZK protocols on arithmetic circuits over $\mathbb{Z}_{2^\lambda}$. These VOLE protocols could also benefit other works that need VOLE over integer rings like the MPC protocol SPD$\mathbb{Z}_{2^k}$ [27]. The current security estimate of LPN over $\mathbb{Z}_{2^\lambda}$ in prior work is directly adapted from that for LPN over a field $\mathbb{F}$ of size $|\mathbb{F}| \approx 2^\lambda$ [17]. As we will show in this section the hardness of LPN over $\mathbb{Z}_{2^\lambda}$ is more related to that over $\mathbb{F}_2$ (rather than that over the λ-bit field). As depicted in Figure 2, we provide the following reductions between the hardness of LPN over $\mathbb{Z}_{2^\lambda}$ and that over $\mathbb{F}_2$.

- **Decisional LPN over $\mathbb{Z}_{2^\lambda}$ → Decisional LPN over $\mathbb{F}_2$.** We show that distinguishing LPN samples over $\mathbb{Z}_{2^\lambda}$ with noise weight t from exact noise distribution is no harder than distinguishing LPN over $\mathbb{F}_2$ with noise weight $\frac{2^{(\lambda-1)}}{2^\lambda-1} \cdot t \approx t/2$. This reduction directly gives an attack that reduces the noise weight by half for an LPN instance over $\mathbb{Z}_{2^\lambda}$.

- **Decisional LPN over $\mathbb{F}_2$ → Decisional LPN over $\mathbb{Z}_{2^\lambda}$.** We show that distinguishing LPN over $\mathbb{F}_2$ with noise weight t is no harder than the distinguishing attack on LPN over $\mathbb{Z}_{2^\lambda}$ with 1) non-standard Bernoulli-like integer noise of weight at most $\lambda \cdot t$; and 2) standard Bernoulli noise of weight $\approx 2^\lambda \cdot t$.

- **Computational LPN over $\mathbb{Z}_{2^\lambda}$ → Computational LPN over $\mathbb{F}_2$.** We show that a secret recovery attack on LPN over $\mathbb{Z}_{2^\lambda}$ with noise weight t is no harder than that on LPN over $\mathbb{F}_2$ with noise weight roughly $t/2$. While a generic reduction requires $k^{\omega(\lambda)}$-hardness for LPN over $\mathbb{Z}_{2^\lambda}$, we also give more efficient reductions for their weakly one-wayness that is more relevant to practical attacks and security estimates. We also discuss how to optimize the secret recovery attack on LPN over $\mathbb{Z}_{2^\lambda}$ based on that over $\mathbb{F}_2$ in practice.

[6] For large q, we can get full-rank square matrices with at least constant probability.

We give similar reductions for the LPN over $\mathbb{Z}_{p^{\lambda_1}q^{\lambda_2}}$ (for distinct primes p, q) in Appendix B. When we give the reductions between different computational LPN variants, we assume that LPN over a field in consideration mostly has a unique solution in the average case (except for a negligible fraction), which will simplify the analysis. Note that this is true for most interesting parameter regimes of LPN, which give rise to cryptographic applications (e.g., PCG and public-key encryption).

Lemma 3 (Unique decoding of LPN over any finite field $\mathbb{F}$)**.** *For any $N > k + 4t \log N$, the following is bounded by $\frac{N^{2t}}{|\mathbb{F}|^{N-k-2t}}$:*

$$\Pr_{\mathbf{A} \leftarrow \mathbb{F}^{k \times N}} \left[\exists \boldsymbol{s}_1 \neq \boldsymbol{s}_2 \in \mathbb{F}^k, \boldsymbol{e}_1, \boldsymbol{e}_2 \in \mathbb{F}^N : |\boldsymbol{e}_1|, |\boldsymbol{e}_2| \leq t \wedge (\mathbf{A} \cdot \boldsymbol{s}_1 + \boldsymbol{e}_1 = \mathbf{A} \cdot \boldsymbol{s}_2 + \boldsymbol{e}_2) \right].$$

Proof. Let $\boldsymbol{s} \stackrel{\text{def}}{=} \boldsymbol{s}_1 - \boldsymbol{s}_2 \in \mathbb{F}^k$ and $\boldsymbol{e} \stackrel{\text{def}}{=} \boldsymbol{e}_2 - \boldsymbol{e}_1 \in \mathbb{F}^N$. For any $\boldsymbol{s} \neq \mathbf{0}$, $\mathbf{A} \cdot \boldsymbol{s}$ is uniform over $\mathbb{F}^N$. Together with $|\boldsymbol{e}| \leq 2t$, the probability that $\mathbf{A} \cdot \boldsymbol{s} = \boldsymbol{e}$ is at most $\sum_{i=0}^{2t} \binom{N}{i} / |\mathbb{F}|^{N-i} \leq \left(\sum_{i=0}^{2t} \binom{N}{i} \right) / |\mathbb{F}|^{N-2t} \leq N^{2t} / |\mathbb{F}|^{N-2t}$. By a union bound on all possible $\boldsymbol{s} \in \mathbb{F}^k$, we obtain the bound claimed in the lemma. □

4.1 Reduction from Decisional LPN over $\mathbb{Z}_{2^\lambda}$ to LPN over $\mathbb{F}_2$

We start with a simple observation that the distinguishing attack on LPN over $\mathbb{Z}_{2^\lambda}$ can be based on that over $\mathbb{F}_2$ with roughly halved noise weight. Specifically, we have the following theorem.

Theorem 3. *If decisional $(\mathcal{D}, \mathbb{Z}_{2^\lambda})$-$\mathsf{LPN}(k, N, w_1)$ is (T, ϵ)-hard, then decisional $(\mathcal{D}, \mathbb{F}_2)$-$\mathsf{LPN}(k, N, w_2)$ is $(T - \mathsf{poly}(k, N), O(d\epsilon))$-hard, where $(\mathcal{D}, w_1, w_2, d) \in \{(\mathsf{HW}, t, \frac{2^{(\lambda-1)}}{2^\lambda - 1}t, \sqrt{t}), (\mathsf{Ber}, \mu, \mu, 1)\}$.*

Proof. Given LPN samples over ring $\mathbb{Z}_{2^\lambda}$ $(\mathbf{A}, \boldsymbol{b} = \mathbf{A} \cdot \boldsymbol{s} + \boldsymbol{e} \mod 2^\lambda)$, we observe that least significant bits (LSBs) of these samples $(\mathbf{A}^0 := \mathbf{A} \mod 2, \boldsymbol{b}^0 := \boldsymbol{b} \mod 2)$ constitute exactly the LPN samples over $\mathbb{F}_2$ for noise $\boldsymbol{e}^0 = \boldsymbol{e} \mod 2$. In case that $\boldsymbol{e} \leftarrow \mathsf{HW}_{t,N}(\mathbb{Z}_{2^\lambda})$, the noise vector $\boldsymbol{e}^0$ follows a Bernoulli-like distribution over $\mathbb{F}_2^N$, which is sampled by first picking t out of N coordinates at random and then filling in these t coordinates with random non-zero elements over $\mathbb{Z}_{2^\lambda}$ (and the rest with zeros). Thus, overall $\boldsymbol{e}^0$ has expected weight $t' = \frac{2^{(\lambda-1)}}{2^\lambda - 1} \cdot t$, where $\frac{2^{(\lambda-1)}}{2^\lambda - 1}$ is the probability that a random non-zero element of $\mathbb{Z}_{2^\lambda}$ is odd. By Lemma 2, this implies that with probability $\Omega(1/\sqrt{t})$, the noise vector $\boldsymbol{e}^0$ follows the exact noise distribution $\mathsf{HW}_{t',N}(\mathbb{F}_2)$. On the other hand, the LSBs of $(\mathbf{A}, \boldsymbol{u})$ with an exact $\boldsymbol{u} \in \mathbb{Z}_{2^\lambda}$ are uniform as well. Therefore, one can use $(\mathsf{HW}, \mathbb{F}_2)$-$\mathsf{LPN}(k, N, t')$ solver to distinguish $(\mathbf{A}^0, \boldsymbol{b}^0)$ from uniform samples. The proof for the second statement is likewise, except when taking the LSBs of $\boldsymbol{e} \leftarrow \mathsf{Ber}_{\mu,N}(\mathbb{Z}_{2^\lambda})$ we immediately get $\boldsymbol{e}^0 \sim \mathsf{Ber}_{\mu,N}(\mathbb{F}_2)$ as desired. □

Despite the preserved noise probability μ in the case of Bernoulli distribution, we note that $\mathsf{Ber}_{\mu,N}(\mathbb{Z}_{2^\lambda})$ has expected weight $(1 - 2^{-\lambda})\mu N$, while $\mathsf{Ber}_{\mu,N}(\mathbb{F}_2)$ has expected weight $\mu N/2$ that is roughly $2\times$ smaller than $\mathsf{Ber}_{\mu,N}(\mathbb{Z}_{2^\lambda})$.

4.2 Reduction from LPN over $\mathbb{F}_2$ to Decisional LPN over $\mathbb{Z}_{2^\lambda}$

We first show that the LPN assumption over $\mathbb{F}_2$ implies that over $\mathbb{Z}_{2^\lambda}$ under the standard Bernoulli noise distribution. However, we achieve the goal by paying a price in the security loss due to the dependence among different noise vectors. As a result, we get the very conservative statement that decisional LPN over $\mathbb{F}_2$ with noise weight t is no harder than decisional LPN over $\mathbb{Z}_{2^\lambda}$ with noise weight roughly $2^\lambda t$. We then introduce more useful Bernoulli-like noise distributions to enable more efficient reductions. In particular, we can reduce to an LPN over $\mathbb{Z}_{2^\lambda}$ with noise weight λt (rather than $2^\lambda t$).

Theorem 4. *If decisional* $(\mathsf{Ber}, \mathbb{F}_2)$-$\mathsf{LPN}(k, N, \mu/2^\lambda)$ *is* (T, ϵ)*-hard, then decisional* $(\mathsf{Ber}, \mathbb{Z}_{2^\lambda})$-$\mathsf{LPN}(k, N, \mu)$ *is* $(T - \mathsf{poly}(k, N), \lambda\epsilon)$*-hard.*

Proof. Let $(\mathbf{A}, \boldsymbol{b} = \mathbf{A} \cdot \boldsymbol{s} + \boldsymbol{e} \mod 2^\lambda)$ be an LPN over $\mathbb{Z}_{2^\lambda}$. Decompose the matrix and vectors into λ ones over $\mathbb{F}_2$ as follows: $(\mathbf{A}^0, \mathbf{A}^1, \cdots, \mathbf{A}^{\lambda-1}) := \mathsf{BitDecomp}(\mathbf{A})$, $(\boldsymbol{s}^0, \boldsymbol{s}^1, \cdots, \boldsymbol{s}^{\lambda-1}) := \mathsf{BitDecomp}(\boldsymbol{s})$, $(\boldsymbol{e}^0, \boldsymbol{e}^1, \cdots, \boldsymbol{e}^{\lambda-1}) := \mathsf{BitDecomp}(\boldsymbol{e})$ and $(\boldsymbol{b}^0, \boldsymbol{b}^1, \cdots, \boldsymbol{b}^{\lambda-1}) := \mathsf{BitDecomp}(\boldsymbol{b})$. Therefore, for $i \in [\lambda]$ we can write

$$\boldsymbol{b}^i = \mathbf{A}^0 \cdot \boldsymbol{s}^i + \boldsymbol{e}^i + f_i(\mathbf{A}, \boldsymbol{s}^0, \cdots, \boldsymbol{s}^{i-1}, \boldsymbol{e}^0, \cdots, \boldsymbol{e}^{i-1}) \mod 2 \ ,$$

where f_i is the sum of all other terms involving the individual $\boldsymbol{s}^j$ and $\boldsymbol{e}^j$ with index $j \leq i-1$. Define the hybrid distributions $H_0, \cdots, H_\lambda$ as

$$\begin{aligned} H_0 = &\quad (\mathbf{A}, \boldsymbol{u}_0, \cdots, \boldsymbol{u}_{i-1}, \boldsymbol{u}_i \cdots, \boldsymbol{u}_{\lambda-1}) \\ &\vdots \\ H_i = &\quad (\mathbf{A}, \boldsymbol{b}^0, \cdots, \boldsymbol{b}^{i-1}, \boldsymbol{u}_i \cdots, \boldsymbol{u}_{\lambda-1}) \\ &\vdots \\ H_\lambda = &\quad (\mathbf{A}, \boldsymbol{b}^0, \cdots, \boldsymbol{b}^{i-1}, \boldsymbol{b}^i \cdots, \boldsymbol{b}^{\lambda-1}) \end{aligned}$$

where $\boldsymbol{u}_j \leftarrow \mathbb{F}_2^N$ for $j \in [\lambda]$ is sampled independently at random. Note that all the $\boldsymbol{s}^i$'s are independent and uniformly random. It suffices to consider the effective noise rate on $\boldsymbol{e}^i$ conditioned on $\boldsymbol{e}^0, \cdots, \boldsymbol{e}^{i-1}$ (as part of f_i). Therefore, if all the adjacent H_{i-1} and H_i are computationally indistinguishable, then H_0 and H_λ are computationally indistinguishable by a hybrid argument. It thus remains to estimate the effective noise rate. Consider a single noise sample $(\boldsymbol{e}^0[j], \boldsymbol{e}^1[j], \ldots, \boldsymbol{e}^{\lambda-1}[j]) \leftarrow \mathsf{Ber}_{\mu,N}(\mathbb{Z}_{2^\lambda})$, where $\boldsymbol{e}^i[j]$ is the j-th bit of $\boldsymbol{e}^i$. Conditioned on any non-zero $(\boldsymbol{e}^0[j], \ldots, \boldsymbol{e}^{i-1}[j])$, $\boldsymbol{e}^i[j]$ is uniformly random and thus unconditionally masks the corresponding $\boldsymbol{b}^i[j]$. Otherwise, we have that

$$\Pr\left[\boldsymbol{e}^i[j] = 1 \,\middle|\, (\boldsymbol{e}^0[j], \ldots, \boldsymbol{e}^{i-1}[j]) = 0^i\right] = \frac{\mu 2^{-(i+1)}}{1 - \mu + \mu 2^{-i}} \geq \mu 2^{-(i+1)}$$

is the noise rate needed to keep the computational indistinguishability between H_i and H_{i+1}, which reaches its minimum $\mu 2^{-\lambda}$ when $i = \lambda - 1$. □

Based on the above theorem, we easily obtain the following corollary whose proof is given in Appendix A.

Corollary 2. *If decisional* $(\mathsf{Ber}, \mathbb{F}_2)$-$\mathsf{LPN}(k, N, \mu/2^\lambda)$ *is hard, then computational* $(\mathsf{HW}, \mathbb{Z}_{2^\lambda})$-$\mathsf{LPN}$ $(k, N, t = (1 - 2^{-\lambda})\mu N)$ *is hard.*

In retrospect, the dependency among the noise vectors $\{\boldsymbol{e}^i\}$ incurs significant losses during the reduction. This motivates us to introduce the specific noise distributions, $\mathsf{IndBer}_{\mu,N}(\mathbb{Z}_{2^\lambda})$ and $\mathsf{IndHW}_{t,N}(\mathbb{Z}_{2^\lambda})$, where Ind refers that the bit decomposition of the noise, $\boldsymbol{e}^0$, …, $\boldsymbol{e}^{\lambda-1}$ are independent and identically distributed, and parameter μ (resp., t) is noise rate (resp., weight) of each $\boldsymbol{e}^i$.

- $\mathsf{IndBer}_{\mu,N}(\mathbb{Z}_{2^\lambda})$ is bit-wise independent. By $\boldsymbol{e} \leftarrow \mathsf{IndBer}_{\mu,N}(\mathbb{Z}_{2^\lambda})$, we mean that $\boldsymbol{e} := \sum_{i=0}^{\lambda-1} 2^i \cdot \boldsymbol{e}^i$ with $\boldsymbol{e}^i \leftarrow \mathsf{Ber}_{\mu,N}(\mathbb{F}_2)$ for $i \in [\lambda]$. The noise rate of $\mathsf{IndBer}_{\mu,N}(\mathbb{Z}_{2^\lambda})$ is the probability that each coordinate of $\boldsymbol{e}$ is non-zero, i.e., $1 - (1 - \mu/2)^\lambda \leq \lambda\mu/2$ by Bernoulli's inequality. Therefore, the expected Hamming weight of $\boldsymbol{e} \leftarrow \mathsf{IndBer}_{\mu,N}(\mathbb{Z}_{2^\lambda})$ is at most λt where $t = \mu N/2$.
- $\mathsf{IndHW}_{t,N}(\mathbb{Z}_{2^\lambda})$ decomposes into λ independent vectors from $\mathsf{HW}_{t,N}(\mathbb{F}_2)$. By $\boldsymbol{e} \leftarrow \mathsf{IndHW}_{t,N}(\mathbb{Z}_{2^\lambda})$, we mean that $\boldsymbol{e} := \sum_{i=0}^{\lambda-1} 2^i \cdot \boldsymbol{e}^i$ with $\boldsymbol{e}^i \leftarrow \mathsf{HW}_{t,N}(\mathbb{F}_2)$ for $i \in [\lambda]$. It is easy to see that the Hamming weight of $\boldsymbol{e}$ is at most λt.

Algorithm 1: $\mathcal{A}_{\mathrm{LPN}_{2^\lambda}}$, the secret recovery algorithm on LPN over $\mathbb{Z}_{2^\lambda}$ ($\lambda \geq 2$) with oracle access to $\mathcal{A}_{\mathrm{LPN}_2}$ (the solver for LPN over $\mathbb{F}_2$).

Input: $(\mathcal{D}, \mathbb{Z}_{2^\lambda})$-$\mathsf{LPN}(k, N, t)$ samples $(\mathbf{A}, \boldsymbol{b} = \mathbf{A} \cdot \boldsymbol{s} + \boldsymbol{e} \mod 2^\lambda)$

Output: $\boldsymbol{s} \in \mathbb{Z}_{2^\lambda}$

$(\mathbf{A}^0, \mathbf{A}^1, \cdots, \mathbf{A}^{\lambda-1}) := \mathsf{BitDecomp}(\mathbf{A})$;

$(\boldsymbol{b}^0, \boldsymbol{b}^1, \cdots, \boldsymbol{b}^{\lambda-1}) := \mathsf{BitDecomp}(\boldsymbol{b})$;

$(\boldsymbol{s}^0, \boldsymbol{e}^0) \leftarrow \mathcal{A}_{\mathrm{LPN}_2}(\mathbf{A}^0, \boldsymbol{b}^0)$;

$\boldsymbol{b}' := (\boldsymbol{b} - \mathbf{A} \cdot \boldsymbol{s}^0 - \boldsymbol{e}^0)/2$;

Return $\boldsymbol{s} = \boldsymbol{s}^0 + 2 \cdot \mathcal{A}_{\mathrm{LPN}_{2^{(\lambda-1)}}}\left(\mathbf{A}' := \sum_{i=0}^{\lambda-2} 2^i \cdot \mathbf{A}^i,\ \boldsymbol{b}'\right)$;

Distributions $\mathsf{IndBer}_{\mu,N}(\mathbb{Z}_{2^\lambda})$ and $\mathsf{IndHW}_{t,N}(\mathbb{Z}_{2^\lambda})$ have not been used in existing protocols. However, LPN with such distributions can be used to design PCG-like protocols by running these protocols with maximum weight λt. Below, we show that decisional LPN over $\mathbb{F}_2$ with noise weight t is *tightly* equivalent to decisional LPN over $\mathbb{Z}_{2^\lambda}$ with noise weight roughly λt under the new noise distributions.

Theorem 5. *Let* $(\mathcal{D}_1, \mathcal{D}_2, w) \in \{(\mathsf{Ber}, \mathsf{IndBer}, \mu), (\mathsf{HW}, \mathsf{IndHW}, t)\}$ *and we have:*

1. *If decisional* $(\mathcal{D}_1, \mathbb{F}_2)$-$\mathsf{LPN}(k, N, w)$ *is* (T, ϵ)*-hard, then decisional* $(\mathcal{D}_2, \mathbb{Z}_{2^\lambda})$- $\mathsf{LPN}(k, N, w)$ *is* $(T - \mathsf{poly}(k, N), \lambda\epsilon)$*-hard.*
2. *If decisional* $(\mathcal{D}_2, \mathbb{Z}_{2^\lambda})$-$\mathsf{LPN}(k, N, w)$ *is* (T, ϵ)*-hard, then decisional* $(\mathcal{D}_1, \mathbb{F}_2)$- $\mathsf{LPN}(k, N, w)$ *is* $(T - \mathsf{poly}(k, N), \epsilon)$*-hard.*

Proof. The proof of the first statement is similar to that of Theorem 4, except that now every $\boldsymbol{e}^i$ is independent of the previous $\boldsymbol{e}^0, \cdots, \boldsymbol{e}^{i-1}$, where $\boldsymbol{e}^i$ follows $\mathsf{Ber}_{\mu,N}(\mathbb{F}_2)$ by the definition of $\boldsymbol{e} \leftarrow \mathsf{IndBer}_{\mu,N}(\mathbb{Z}_{2^\lambda})$. The proof of the second statement can be trivially adapted from that of Theorem 3, i.e., $\mathbf{A}^0 = \mathbf{A} \mod 2$, $\mathbf{A}^0 \cdot \boldsymbol{s}^0 + \boldsymbol{e}^0 = \mathbf{A} \cdot \boldsymbol{s} + \boldsymbol{e} \mod 2$. □

On the choice of matrix $\mathbf{A}$ As we can see from the proofs of Theorem 4, Theorem 5 and Theorem 6 (shown in Section 4.3), all the reductions only rely on that $\mathbf{A}^0$ is uniformly distributed over $\mathbb{F}_2^{N \times k}$ while $\mathbf{A}^1, \cdots, \mathbf{A}^{\lambda-1}$ can be arbitrary (or even zero matrix), where $(\mathbf{A}^0, \mathbf{A}^1, \ldots, \mathbf{A}^{\lambda-1}) := \mathsf{BitDecomp}(\mathbf{A})$. In other words, it suffices to use a Boolean matrix $\mathbf{A} = \mathbf{A}^0$, and the choices of $\mathbf{A}^1, \ldots, \mathbf{A}^{\lambda-1}$ do not introduce any further hardness to the LPN problem over $\mathbb{Z}_{2^\lambda}$. Overall, we give a positive result that LPN over ring $\mathbb{Z}_{2^\lambda}$ with Boolean matrices is secure if the corresponding LPN over binary field $\mathbb{F}_2$ is secure.

4.3 Reduction from Computational LPN over $\mathbb{Z}_{2^\lambda}$ to LPN over $\mathbb{F}_2$

We show that an LPN instance over $\mathbb{Z}_{2^\lambda}$ can be efficiently translated to λ instances of LPN over $\mathbb{F}_2$, which are independent except that they share the same random matrix $\mathbf{A}^0$ over $\mathbb{F}_2$ and that the noise vectors of the λ instances are somehow correlated. We refer to the proof of Theorem 6 on how to address the correlation issue. Here we give a reduction from computational LPN over a ring $\mathbb{Z}_{2^\lambda}$ to that over $\mathbb{F}_2$ by extending the corresponding reduction between their decisional versions shown in Section 4.1. Algorithm 1 shows how computational LPN over $\mathbb{Z}_{2^\lambda}$ is reduced to that over $\mathbb{Z}_{2^{\lambda-1}}$ by recursion, whose correctness is analyzed in Lemma 4. Note that $\mathcal{A}_{\mathrm{LPN}_{2^\lambda}}$ degenerates to secret recovery algorithm for LPN over $\mathbb{F}_2$ when $\lambda = 1$. Without loss of generality, we assume that $\mathcal{A}_{\mathrm{LPN}_2}$ returns the noise vector in addition to the recovered secret.

Lemma 4. *Let* $(\mathbf{A}, \boldsymbol{b} = \mathbf{A} \cdot \boldsymbol{s} + \boldsymbol{e} \mod 2^\lambda)$ *be the LPN samples over* $\mathbb{Z}_{2^\lambda}$*, then* $(\mathbf{A}', \boldsymbol{b}')$ *as defined in Algorithm 1 constitute the LPN samples over* $\mathbb{Z}_{2^{(\lambda-1)}}$*, where* $\mathbf{A}' = \sum_{i=0}^{\lambda-2} 2^i \cdot \mathbf{A}^i$*,* $\boldsymbol{b}' = \mathbf{A}' \cdot \boldsymbol{s}' + \boldsymbol{e}'$*,* $\boldsymbol{s}' = \sum_{i=1}^{\lambda-1} 2^{i-1} \cdot \boldsymbol{s}^i$ *and* $\boldsymbol{e}' = \sum_{i=1}^{\lambda-1} 2^{i-1} \cdot \boldsymbol{e}^i$

Proof. Let $(\mathbf{A}^0, \mathbf{A}^1, \cdots, \mathbf{A}^{\lambda-1})$ and $(\boldsymbol{b}^0, \boldsymbol{b}^1, \cdots, \boldsymbol{b}^{\lambda-1})$ be the matrices and vectors defined in Algorithm 1. Let $(\boldsymbol{s}^0, \boldsymbol{s}^1, \cdots, \boldsymbol{s}^{\lambda-1}) := \mathsf{BitDecomp}(\boldsymbol{s})$ and $(\boldsymbol{e}^0, \boldsymbol{e}^1, \ldots, \boldsymbol{e}^{\lambda-1}) := \mathsf{BitDecomp}(\boldsymbol{e})$. Note that $\mathbf{A}'$ is obtained from $\mathbf{A}$ by truncating the most significant bits (MSBs), and thus follows the distribution in LPN over $\mathbb{Z}_{2^{(\lambda-1)}}$. It suffices to prove $\boldsymbol{b}' = \mathbf{A}' \cdot \boldsymbol{s}' + \boldsymbol{e}' \mod 2^{(\lambda-1)}$, where $\boldsymbol{s}' = \sum_{i=1}^{\lambda-1} 2^{i-1} \cdot \boldsymbol{s}^i$ and $\boldsymbol{e}' = \sum_{i=1}^{\lambda-1} 2^{i-1} \cdot \boldsymbol{e}^i$ are the secret and noise of LPN over $\mathbb{Z}_{2^{(\lambda-1)}}$ respectively. In particular, we have the following:

$$\begin{aligned} 2 \cdot \boldsymbol{b}' - 2 \cdot (\mathbf{A}' \cdot \boldsymbol{s}' + \boldsymbol{e}') &= \boldsymbol{b} - \mathbf{A} \cdot \boldsymbol{s}^0 - \boldsymbol{e}^0 - 2 \cdot (\mathbf{A}' \cdot \boldsymbol{s}' + \boldsymbol{e}') \\ &= \boldsymbol{b} - \mathbf{A} \cdot \boldsymbol{s}^0 - \boldsymbol{e}^0 - \mathbf{A} \cdot (\boldsymbol{s} - \boldsymbol{s}^0) - \boldsymbol{e} + \boldsymbol{e}^0 \\ &= \boldsymbol{b} - \mathbf{A} \cdot \boldsymbol{s} - \boldsymbol{e} = \mathbf{0} \mod 2^\lambda, \end{aligned}$$

where note that $2 \cdot \boldsymbol{e}' = \boldsymbol{e} - \boldsymbol{e}^0$. □

Below, we show that $(\epsilon^{\lambda+1})$-hard computational LPN over $\mathbb{Z}_{2^\lambda}$ implies $O(\epsilon)$-hard LPN over $\mathbb{F}_2$, where $\lambda = O(1)$ needs to be small in general (for polynomial hardness), and it can be up to $\lambda = k^{\Theta(1)}$ (for subexponential hardness). E.g., if the former is $2^{\sqrt{k}}$-hard with $\lambda = k^{0.25}$, then the latter is $2^{\Theta(k^{0.25})}$-hard.

Theorem 6. *If computational* $(D_1, \mathbb{Z}_{2^\lambda})$-$\mathsf{LPN}(k, N, w)$ *is* $(\lambda T + \mathsf{poly}(k, N), \epsilon^{\lambda+1})$*-hard, then computational* $(D_2, \mathbb{F}_2)$-$\mathsf{LPN}(k, N, w)$ *is* $(T, 2\epsilon)$*-hard, where* $(D_1, D_2, w) \in \{(\mathsf{Ber}, \mathsf{Ber}, \mu), (\mathsf{IndBer}, \mathsf{Ber}, \mu), (\mathsf{IndHW}, \mathsf{HW}, t)\}$.

Proof. For contradiction assume that there exists an algorithm $\mathcal{A}_{\mathrm{LPN}_2}$ that recovers the secret of LPN over $\mathbb{F}_2$ with probability more than 2ϵ within time T. It suffices to prove the case $(D_1, D_2, w) = (\mathsf{Ber}, \mathsf{Ber}, \mu)$. For proof convenience[7], consider distribution $\mathsf{Ber}_{\mu,N}(\mathbb{F}_2)$ being sampled in two steps: first pick each coordinate with probability μ independently (and let the rest with 0's), and second assign the picked coordinates with uniform random bits. By a simple argument there exist at least an ϵ fraction of good $(\mathbf{A}^0, \mathsf{coin}_1(\boldsymbol{e}^i))$ for which $\mathcal{A}_{\mathrm{LPN}_2}$ recovers $\boldsymbol{s}^i$ from $(\mathbf{A}^0, \mathbf{A}^0\boldsymbol{s}^i + \boldsymbol{e}^i)$ with probability at least ϵ, where $\mathsf{coin}_1(\boldsymbol{e}^i)$ denotes the step-1 random coin for sampling $\boldsymbol{e}^i$ and the probability is taken over the $\boldsymbol{s}^i$ and the step-2 coin of $\boldsymbol{e}^i$. Therefore, $\mathcal{A}_{\mathrm{LPN}_{2^\lambda}}$ (see Algorithm 1) invokes $\mathcal{A}$ on $(\mathbf{A}, \boldsymbol{b} = \mathbf{A} \cdot \boldsymbol{s} + \boldsymbol{e} \mod 2^\lambda)$ for λ times and recovers $\boldsymbol{s}$ with an overall probability of at least $\epsilon^{\lambda+1}$, a contradiction to the assumption. The proofs for the other two cases are slightly simpler because $\boldsymbol{e}^0, \ldots, \boldsymbol{e}^{\lambda-1}$ are independent and thus no two-step sampling is needed, i.e., $\mathsf{coin}_1(\boldsymbol{e}^i)$ is empty. □

In Appendix A, we further prove the following theorem.

Theorem 7. *If computational* $(\mathsf{HW}, \mathbb{Z}_{2^\lambda})$-$\mathsf{LPN}(k, N, t)$ *is* $(\lambda T + \mathsf{poly}(k, N), \epsilon^{\lambda+1})$*-hard, then computational* $(\mathsf{HW}, \mathbb{F}_2)$-$\mathsf{LPN}(k, N, t')$ *is* $(T, \frac{2\epsilon}{1-\exp(-\delta^2 t/6)})$*-hard, where* $t' = t\frac{2^{(\lambda-1)}}{2^\lambda-1}(1+\delta)$ *for any constant* $\delta > 0$.

The above reduction suffers a significant security loss by exponent factor $1/(\lambda+1)$ since computationally intractable problems typically require a small success probability for efficient adversaries. In the setting of practical key recovery attacks, however, we often expect the success probability to be $(1 - 1/\mathsf{poly}(k))$ or even overwhelming. In this case, we get more efficient reductions as below.

[7] The two-step sampling is defined to be in line with $\mathsf{Ber}_{\mu,N}(\mathbb{Z}_{2^\lambda})$, and therefore captures the correlations among $\boldsymbol{e}^0, \ldots, \boldsymbol{e}^{\lambda-1}$, which share the same step-1 randomness.

$(\mathsf{HW}, \mathbb{F})$-LPN Parameters			This work ($\log \lvert\mathbb{F}\rvert = 128$)				This work ($\log \lvert\mathbb{F}\rvert = 1$)				[17] (Any field size)		
N	k	t	Gauss	SD	SD 2.0	ISD	Gauss	SD	SD 2.0	ISD	Gauss	SD	ISD
2^{10}	652	57	111	184	184	111	111	194	116	94	80	93	115
2^{12}	1589	98	100	151	151	100	100	154	137	83	85	80	104
2^{14}	3482	198	101	149	149	101	101	150	144	86	94	80	108
2^{16}	7391	389	103	147	147	103	103	148	146	91	99	80	112
2^{18}	15336	760	105	146	146	105	105	146	146	95	103	80	117
2^{20}	32771	1419	107	145	145	107	107	145	145	99	106	80	121
2^{22}	67440	2735	108	144	144	108	108	144	144	104	108	80	126

Table 3: The comparison of our analysis and [17] about the costs of Pooled Gauss, SD attack, our adapted SD 2.0 (SD 2.0) and ISD attacks to solve an LPN problem with dimension k, number of samples N, Hamming weight of noises t.

Theorem 8. *If the computational $(D_1, \mathbb{F}_2)$-$\mathsf{LPN}(k, N, w)$ problem can be broken (say by $\mathcal{A}_{LPN_2}$) in time T with success probability at least $(1-\epsilon)$, then the computational $(D_2, \mathbb{Z}_{2^\lambda})$-$\mathsf{LPN}(k, N, w)$ problem can be broken by $\mathcal{A}_{LPN_{2^\lambda}}$ (see Algorithm 1) in time $\lambda T + \mathsf{poly}(k, N)$ with success probability at least $1-(\lambda+1)\sqrt{\epsilon}$, where $(D_1, D_2, w) \in \{(\mathsf{Ber}, \mathsf{Ber}, \mu), (\mathsf{Ber}, \mathsf{IndBer}, \mu), (\mathsf{HW}, \mathsf{IndHW}, t)\}$.*

Proof. Similar to the proof of Theorem 6, we have by a Markov inequality that for at least a $(1-\sqrt{\epsilon})$ fraction of $(\mathbf{A}^0, \mathsf{coin}_1(\boldsymbol{e}^i))$ $\mathcal{A}_{\mathrm{LPN}_2}$ recovers $\boldsymbol{s}^i$ from $(\mathbf{A}^0, \mathbf{A}^0\boldsymbol{s}^i + \boldsymbol{e}^i)$ with probability at least $1-\sqrt{\epsilon}$. Overall, $\mathcal{A}_{\mathrm{LPN}_{2^\lambda}}$ succeeds with probability $(1-\sqrt{\epsilon})(1-\lambda\sqrt{\epsilon}) \geq 1-(\lambda+1)\sqrt{\epsilon}$ by a union bound. □

In Appendix A, we also give a proof of the following theorem.

Theorem 9. *If the computational $(\mathsf{HW}, \mathbb{F}_2)$-$\mathsf{LPN}(k, N, t')$ problem can be broken (say by $\mathcal{A}_{LPN_2}$) in time T with success probability at least $(1-\epsilon/2)$, then the computational $(\mathsf{HW}, \mathbb{Z}_{2^\lambda})$-$\mathsf{LPN}(k, N, t)$ problem can be broken by $\mathcal{A}_{LPN_{2^\lambda}}$ (see Algorithm 1) in time $\lambda T + \mathsf{poly}(k, N)$ with success probability at least $1-(\lambda+1)\sqrt{\epsilon}$, where $t' = t\frac{2^{\lambda-1}}{2^\lambda-1}(1+\delta)$ for any δ and ϵ satisfying $\delta^2 t \geq 6\ln(2/\epsilon)$.*

Optimized attacks on $(\mathsf{Ber}/\mathsf{HW}, \mathbb{Z}_{2^\lambda})$-$\mathsf{LPN}$. In practice, we optimize the attacks on $(\mathsf{Ber}/\mathsf{HW}, \mathbb{Z}_{2^\lambda})$-$\mathsf{LPN}$ by exploiting the correlations among the error vectors of the λ instances, $\boldsymbol{e}^0, \ldots, \boldsymbol{e}^{\lambda-1}$. In particular, Algorithm 1 recovers the corresponding secrets $\boldsymbol{s}^0, \boldsymbol{s}^1, \cdots, \boldsymbol{s}^{\lambda-1}$ sequentially. That means when the attacker works on the $(i+1)$-th LPN instance he already sees $\boldsymbol{e}^0, \ldots, \boldsymbol{e}^{i-1}$ from the previous i broken instances. As analyzed in the proof of Theorem 4, for any single noise sample $(\boldsymbol{e}^0[j], \boldsymbol{e}^1[j], \ldots, \boldsymbol{e}^{\lambda-1}[j]) \leftarrow \mathsf{Ber}_{\mu,N}(\mathbb{Z}_{2^\lambda})$, $\boldsymbol{e}^i[j]$ is uniformly random conditioned on any non-zero $(\boldsymbol{e}^0[j], \ldots, \boldsymbol{e}^{i-1}[j])$, and thus sample $\boldsymbol{b}^i[j]$ is useless (encrypted by one-time pad) and should be discarded. In other words, the effective noise rate of the i-th LPN instance is roughly $\mu 2^{-(i+1)}$ given the attacker's knowledge about $\boldsymbol{e}^0, \ldots, \boldsymbol{e}^{i-1}$. Therefore, the success rate of solving the $(\mathsf{Ber}, \mathbb{Z}_{2^\lambda})$-$\mathsf{LPN}(k, N, \mu)$ is roughly the product of the λ instances of $(\mathsf{Ber}, \mathbb{F}_2)$-$\mathsf{LPN}$ with continuously halving noise rates μ, $\mu/2$, …, $\mu/2^{\lambda-1}$. For instance, if solving these instances can succeed with probability ϵ, $\epsilon^{2^{-1}}$, …, $\epsilon^{2^{-(\lambda-1)}}$ respectively, then it leads to a success probability of approximately ϵ^2 (instead of $\epsilon^{\lambda+1}$). The optimization for reducing $(\mathsf{HW}, \mathbb{Z}_{2^\lambda})$-$\mathsf{LPN}(k, N, t)$) to $(\mathsf{HW}, \mathbb{F}_2)$-$\mathsf{LPN}$ is likewise.

5 Concrete Analysis of Low-Noise LPN over Finite Fields

Recently, a series of works [17, 19, 70, 20, 79, 76, 26] use the (dual-)LPN problem with very low noise rate over finite fields to construct concretely efficient PCG-like protocols, which extend a small number of correlations (e.g., COT, ROT, VOLE and OLE) to a large number of correlations with sublinear communication.

$(\mathsf{HW}, \mathbb{F})$-dual-LPN Parameters			This work ($\log \lvert\mathbb{F}\rvert = 128$)				This work ($\log \lvert\mathbb{F}\rvert = 1$)				[17] (Any field size)		
n	N	t	Gauss	SD	SD 2.0	ISD	Gauss	SD	SD 2.0	ISD	Gauss	SD	ISD
2^{10}	2^{12}	44	117	189	189	117	117	191	170	97	80	100	117
2^{12}	2^{14}	39	111	170	169	111	111	170	166	95	80	92	112
2^{14}	2^{16}	34	107	151	151	107	107	151	151	93	80	84	107
2^{16}	2^{18}	32	108	145	145	108	108	145	145	95	84	82	109
2^{18}	2^{20}	31	112	143	143	112	112	143	143	99	88	82	112
2^{20}	2^{22}	30	116	141	141	116	116	141	141	103	93	82	116
2^{22}	2^{24}	29	119	139	139	119	119	139	139	107	97	82	120

Table 4: The comparison of our analysis and [17] about the costs of Pooled Gauss, SD attack, our adapted SD 2.0 (SD 2.0) and ISD attacks to solve a dual-LPN problem with $n = N/4$ (related to the number of COT/VOLE correlations), number of samples N, Hamming weight of noises t.

These protocols can be used as building blocks to design a variety of MPC and ZK protocols. Therefore, the hardness of (dual-)LPN problems is crucial to guarantee the security of all the protocols.

Almost all of the known PCG-like protocols based on (dual-)LPN over a field adopt the formulas by Boyle et al. [17] to select the concrete parameters for some specified security level. Boyle et al. [17] obtained the formulas by analyzing three attacks: Pooled Gauss attack, information set decoding (ISD) attack and SD attack. However, their analysis is oversimplified and the formulas are not accurate, which express in the following aspects:

- When analyzing the hardness of LPN with an exact noise distribution $\mathsf{HW}_{t,N}(\mathbb{F})$, the formula against Pooled Gauss attack is obtained by viewing $\mathsf{HW}_{t,N}(\mathbb{F})$ as a Bernoulli distribution, which makes the formula not accurate.
- When analyzing the hardness of LPN against ISD attack, the formula is just an upper bound of the complexity of the Prange's ISD algorithm [67] to solve LPN problems over a large field. This does not cover the advanced ISD variants [72, 36, 59, 13]. Furthermore, their analysis does not capture the impact of field sizes when calculating the ISD cost.
- When analyzing the hardness of LPN against SD attack, each parity-check vector is assumed to be independently in compliance with a Bernoulli distribution, which is inaccurate [30]. In addition, the time estimated by them just allows for achieving a small successful probability, which does not match with that by the above two attacks.

In the following subsections, we will give more detailed analysis for the hardness of low-noise (dual-)LPN problems as well as more accurate formulas. Since all known PCG-like protocols use an exact noise vector with fixed weight, our formulas just consider the case of exact noise distributions. According to our analysis, we show more accurate costs of the above three attacks in Table 3 and Table 4. In these tables, we also compare the attack costs with that given by Boyle et al. [17] and all the LPN parameters are adopted from [17]. Under the same LPN parameters, while Boyle et al. [17] showed that either Pooled Gauss attack or SD attack has the lowest cost, our more detailed analysis shows that ISD attack has the lowest cost. In other words, for PCG-like protocols, one can only consider the ISD attack to choose a set of LPN parameters in many cases. For large fields, our analysis shows that Pooled Gauss attack has the same cost as ISD attack, where recall that Pooled Gauss can be considered a special case of ISD. The formulas given by Boyle et al. [17] show that SD attack performs the best in most of LPN parameters. However, we point out that this

$(\mathsf{HW}, \mathbb{F})$-$\mathsf{LPN}(k, N, t)$				$(\mathsf{HW}, \mathbb{F})$-dual-$\mathsf{LPN}(N, n, t)$			
N	k	t ([17])	t (ours)	n	N	t ([17])	t (ours)
2^{10}	652	106	68	2^{10}	2^{12}	88	50
2^{12}	1589	172	136	2^{12}	2^{14}	83	48
2^{14}	3482	338	274	2^{14}	2^{16}	78	45
2^{16}	7391	667	531	2^{16}	2^{18}	73	42
2^{18}	15336	1312	1023	2^{18}	2^{20}	68	39
2^{20}	32771	2467	1876	2^{20}	2^{22}	63	36
2^{22}	67440	4788	3552	2^{22}	2^{24}	58	34

Table 5: **The comparison of noise weights between our analysis and [17] for 128-bit security level.** Among the LPN parameters, N is the number of samples, k is the dimension, $n = N/4$ and t is the Hamming weight of a noise vector. We assume the field size is $\log |\mathbb{F}| = 128$, and an exact noise distribution $\mathsf{HW}_{t,N}(\mathbb{F})$.

is generally not true. In particular, our analysis results in Tables 3 and 4 show that SD attack has the highest cost among three kinds of attacks. Furthermore, we show in Section 5.3 that the SD 2.0 attack adapted to the low-noise setting (which recovers the optimal SD attack) require more cost than the ISD attack against LPN problems over large fields using random linear codes. Our proof does *not* require that the Hamming weight of noise vectors achieves the Gilbert-Varshamov bound, and instead allow all weights for the LPN parameters used in PCG-like protocols. Tables 3 and 4 also show that ISD attack has lower cost for smaller field size, which justifies that it is not accurate to use the same formulas for all field sizes as in [17].

Our analysis shows that most of LPN parameter sets used in know PCG-like protocols actually achieve better security. In other words, given a security parameter, smaller weight for noise vectors can be chosen, which allows for achieving higher efficiency in these PCG-like protocols. In particular, for 128-bit security level, Table 5 reports the comparison of Hamming weights of noise vectors between our analysis and previous work [17], given several sets of dimensions and the numbers of samples used in [17]. From this table, we can see that the Hamming weight of a noise vector is reduced by $19\% - 36\%$ for LPN and $41\% - 43\%$ for dual-LPN, respectively. While the COT/VOLE protocols in the PCG framework need to communicate $O(t\kappa \log \frac{N}{t})$ bits and compute t GGM trees for each iteration, the reduction of parameter t will be directly transferred to the improvements of communication and computation costs in these protocols.

5.1 The Hardness of LPN against Pooled Gauss Attack

For solving the problems of LPN and dual-LPN with exact noise distributions, Boyle et al. [17] gave a formula to compute the cost of Pooled Gauss attack by simplifying $\mathsf{HW}_{t,N}(\mathbb{F})$ as $\mathsf{Ber}_{t/N,N}(\mathbb{F})$. Below, we give a more accurate formula to calculate the cost of Pooled Gauss attack, and discuss the difference between the simplified analysis [17] and our analysis.

We first extend Pooled Gauss attack [39] from solving a (dual-)LPN problem from a Bernoulli distribution to solving that with an exact noise distribution $\mathsf{HW}_{t,N}(\mathbb{F})$. Specifically, the Pooled Gauss attack performs as follows:

- Given a $(\mathsf{HW}, \mathbb{F})$-$\mathsf{LPN}(k, N, t)$ instance $\boldsymbol{b} = \mathbf{A} \cdot \boldsymbol{s} + \boldsymbol{e}$, for each iteration, guess k non-noisy coordinates of vector $\boldsymbol{b}$ by sampling them at random, and obtain a length-k vector $\boldsymbol{b}'$ and $k \times k$ matrix $\mathbf{A}'$. Then, compute $\boldsymbol{s}' := (\mathbf{A}')^{-1} \cdot \boldsymbol{b}'$ and verify whether $\boldsymbol{s}'$ is correct or not by running the test algorithm in [39].
- Given a $(\mathsf{HW}, \mathbb{F})$-dual-$\mathsf{LPN}(N, n, t)$ instance $\boldsymbol{b} = \mathbf{H} \cdot \boldsymbol{e}$, for each iteration, guess n coordinates of vector $\boldsymbol{e}$ by sampling them uniformly at random such that these coordinates contain t noisy coordinates of $\boldsymbol{e}$, choose

the corresponding $n \times n$ sub-matrix $\mathbf{H}'$ from $\mathbf{H}$ according to the n coordinates, compute $\boldsymbol{e}' := (\mathbf{H}')^{-1} \cdot \boldsymbol{b}$, and then checks if $|\boldsymbol{e}'| = t$.

The above attack uses the fixed N samples for both LPN and dual-LPN. For solving LPN, Pooled Gauss attack runs in time $\frac{\binom{N}{t}}{\binom{N-k}{t}} \cdot k^{2.8}$, where $\frac{\binom{N-k}{t}}{\binom{N}{t}}$ is the probability of guessing successfully in one iteration, and $k^{2.8}$ is the cost of inverting matrix $\mathbf{A}'$ via Strassen's algorithm. For solving dual-LPN, Pooled Gauss attack runs in time $\frac{\binom{N}{t}}{\binom{N-k}{t}} \cdot (N-k)^{2.8}$, where $k = N - n$ is the dimension and $(N-k)^{2.8}$ is the cost of inverting matrix $\mathbf{H}'$ via Strassen's algorithm. As LPN can be efficiently transformed into dual-LPN and vice versa, Pooled Gauss attack solves a (dual-)LPN problem with number of samples N, dimension k and weight of noises t in time $\frac{\binom{N}{t}}{\binom{N-k}{t}} \cdot \min(k^{2.8}, (N-k)^{2.8})$. Therefore, the bit-security of a (dual)-LPN instance with respect to Pooled Gauss attack is computed as

$$\log\left(\min(k^{2.8}, (N-k)^{2.8})\right) + \left(\log\binom{N}{t} - \log\binom{N-k}{t}\right).$$

For the LPN problem with an exact noise distribution, Boyle et al. [17] simplified the cost of Pooled Gauss attack as $(\frac{1}{1-t/N})^k \cdot k^{2.8}$. In the following, we show the difference between the attack cost in [17] and that by us is huge. In particular, from Lemma 1, we have

$$\frac{\frac{\binom{N}{t}}{\binom{N-k}{t}}}{(\frac{1}{1-t/N})^k} \approx \frac{2^{N\mathbf{H}(t/N)}}{2^{(N-k)\mathbf{H}(\frac{t}{N-k})} \cdot e^{tk/N}} \gtrapprox \frac{2^{t\log(N/t)}}{2^{t(\log(\frac{N-k}{t})+\frac{3}{2})} \cdot e^{tk/N}} = \frac{2^{t\log\left(\frac{N}{N-k}\right)}}{2^{3t/2} \cdot e^{tk/N}},$$

where $\approx$ denotes an approximate relation that omits a polynomial factor, and $\mathbf{H}(\cdot)$ denotes the binary entropy function. For the (dual-)LPN parameters N, k used in known PCG-like protocols, the cost of Pooled Gauss attack estimated by Boyle et al. [17] is about $O(2^t)$ times larger than that estimated by us.

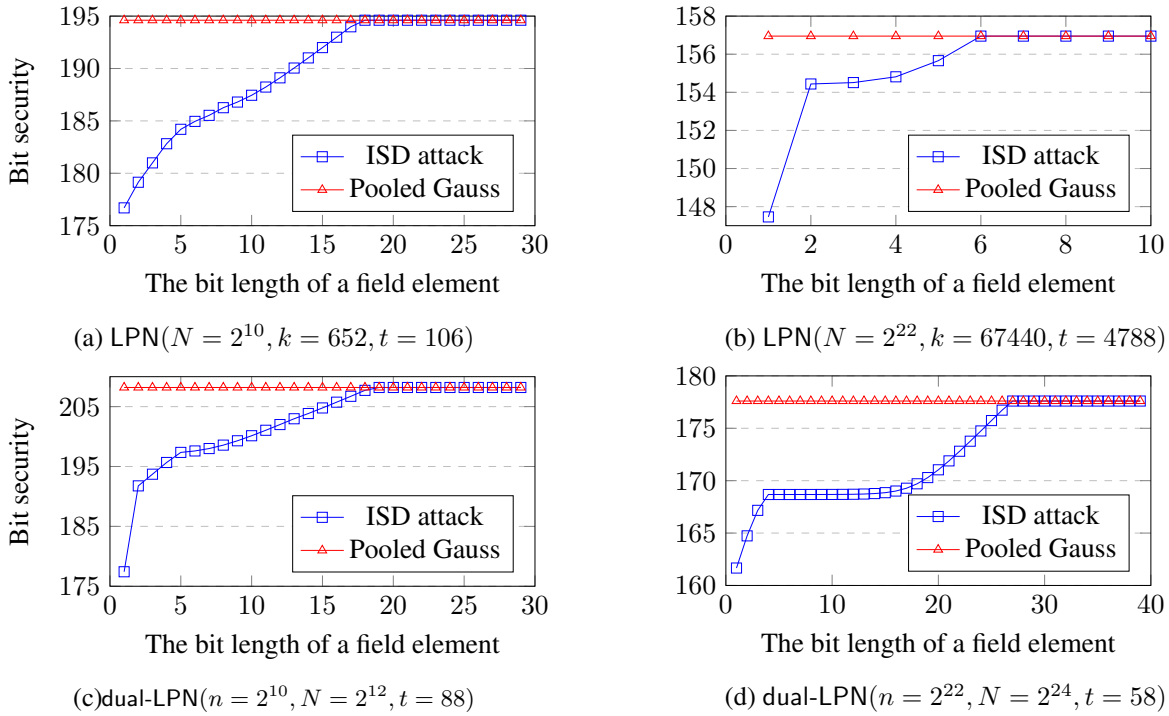
5.2 The Hardness of LPN against ISD Attack

Based on the concrete analysis of ISD attack in [49], Boyle et al. [17] used an upper bound of the cost of the Prange's ISD algorithm [67] to evaluate the hardness of LPN problems over any field against ISD attacks. As shown in Tables 3 and 4, the upper-bound formula cannot capture accurately the cost of more advanced ISD variants [72, 36, 59, 13]. In the following, we first summarize the known ISD variants, and then use the state-of-the-art ISD algorithm to evaluate the hardness of LPN problems over finite fields. We distinguish two cases: binary field $\mathbb{F}_2$ and other larger fields.

Binary field $\mathbb{F}_2$. For binary field $\mathbb{F}_2$, we adopt the state-of-the-art BJMM-ISD algorithm [13] to analyze the concrete hardness of low-noise (dual-)LPN problems. From the expected time of BJMM-ISD shown in Theorem 16 of Appendix D, we can see that it is hard to give a succinct formula to compute the cost of BJMM-ISD. Thus, we choose to provide a script [8], which allows to automatically compute the cost of the BJMM-ISD attack. Our analysis does not consider the known NN-ISD variants [60, 16], because these ISD variants introduce quite large polynomial overheads, and make them less efficient when analyzing the concrete costs of low-noise LPN problems. Besides, our analysis does not cover the ISD algorithms [7, 37, 41] with more efficient space consumption, and always assume that sufficient memory is available which makes the LPN parameters more conservative.

Larger fields. For the case of non-binary fields, we focus on the hardness of (dual-)LPN problems over

[8] This script can be found at https://gist.github.com/hansliu1024/21c87609e75f6cc52decdc69981e1d5b.



(a) LPN($N = 2^{10}, k = 652, t = 106$)

(b) LPN($N = 2^{22}, k = 67440, t = 4788$)

(c)dual-LPN($n = 2^{10}, N = 2^{12}, t = 88$)

(d) dual-LPN($n = 2^{22}, N = 2^{24}, t = 58$)

Figure 3: The costs of Pooled Gauss and SD-ISD attacks from our analysis to solve LPN and dual-LPN problems on different field sizes, where N is the number of samples, k is the dimension, $n = N/4$ and t is the weight of a noise vector.

any finite field $\mathbb{F}$ with $|\mathbb{F}| \geq 256$ (especially for large fields with $|\mathbb{F}| \geq 2^{\rho}$). For other field sizes (i.e., $|\mathbb{F}| \in \{4, 8, 16, 32, 64, 128\}$), low-noise (dual-)LPN seem to be less interesting for PCG, MPC and ZK applications. We adopt the generalized SD-ISD algorithm [66] to analyze the cost of solving low-noise (dual-)LPN problems over field $\mathbb{F}$. As such, we provide a Python script to automatically compute the cost of the SD-ISD attack.

Compared to the SD-ISD algorithm by Peters [66], the SD-ISD variant by Meurer [61] has the performance advantage when the field size $|\mathbb{F}| < 128$, and the advantage becomes vanishingly small when $|\mathbb{F}| \geq 128$. Thus, for the case that $|\mathbb{F}| \geq 256$, it is enough to adopt the Peters's SD-ISD algorithm to evaluate the hardness of low-noise (dual-)LPN problems. Note that it is unclear how to extend the generalization approaches [66, 61] to more efficient MMT-ISD and BJMM-ISD algorithms for solving low-noise LPN over a larger field $\mathbb{F}$ and make the resulting ISD algorithm be significantly more efficient than the SD-ISD variant [66]. We leave it as an interesting future work. Even if these generalization approaches can be efficiently applied in the MMT-ISD and BJMM-ISD algorithms, the performance advantage of MMT-ISD and BJMM-ISD (compared to SD-ISD) decreases when the field size increases, and will diminish for a sufficiently large field. Our analysis does not consider the ISD variants [51, 47] that generalize the May-Ozerov nearest-neighbor algorithm to solve dual-LPN problems any finite field, as these ISD variants introduce quite large polynomial overheads and thus are less efficient when analyzing the concrete costs of solving low-noise LPN.

For any finite field $\mathbb{F}$, the generalized SD-ISD variant [66] generates two sets $\mathcal{S}_0$ and $\mathcal{S}_1$ with size $\binom{(k+\ell)/2}{p/2} \cdot (|\mathbb{F}| - 1)^{p/2}$, where p and ℓ are two additional parameters for the SD-ISD variant. Note that the size of $|\mathcal{S}_0|$ and $|\mathcal{S}_1|$ increases exponentially with p. Following the work [66], p should be quite small to minimize the cost of going though sets $\mathcal{S}_0$ and $\mathcal{S}_1$, and ℓ is set as $\ell = \log_{|\mathbb{F}|} \binom{k/2}{p} + p \log_{|\mathbb{F}|}(|\mathbb{F}| - 1)$. If the

even integer $p \neq 0$, then the cost of the generalized SD-ISD algorithm is at least $O((k+\ell)|\mathbb{F}|)$, which is the cost to just find identical elements in two sets $\mathcal{S}_0$ and $\mathcal{S}_1$. When the field size is large enough, we will have to choose $p = 0$ and $\ell = 0$ to minimize the cost of the generalized SD-ISD attack, according to the above equation. In this case, the generalized SD-ISD attack actually becomes Pooled Gauss attack. In Figure 3, we give a comparison of costs between Pooled Gauss attack and the generalized SD-ISD attack for solving LPN and dual-LPN problems on different field sizes. From this figure, we can see that ISD attack has the same cost as Pooled Gauss attack when the field size is sufficiently large.

5.3 The Hardness of LPN against SD Attack

Boyle et al. [17] analyzed the cost of solving LPN problems against SD attack, and shown that this attack performs the best among three kinds of attacks for their parameters selection. In the following, we show two oversimplifications in their analysis [17], which lead to underestimate the cost of this attack. First, Boyle et al. [17] assumes that each parity-check vector $\boldsymbol{v} \in \mathcal{V}$ independently follows a Bernoulli distribution $\mathsf{Ber}_{w/N,N}(\mathbb{F})$ where $w \in \mathbb{N}$ is the Hamming weight of $\boldsymbol{v}$, which is inaccurate [9] [30] and leads to underestimated cost of SD attack. To obtain more accurate complexity of this attack, a weaker assumption was proposed by [30] where each parity-check vector $\boldsymbol{v} \in \mathcal{V}$ is assumed to be independently in compliance with an exact noise distribution $\mathsf{HW}_{w,N}(\mathbb{F})$. Second, Boyle et al. [17] underestimated the cost of this attack as $T = T_1/\epsilon$, where T_1 is the time of finding one parity check vector $\boldsymbol{v} \in \mathcal{V}$ and ϵ is the distinguishing advantage for one vote. The SD attack solves the decisional LPN problem with negligible advantage in time T, while other attacks solve the LPN problem with constant advantage. Following the previous work [3, 65, 44, 30], this attack takes time $T = T_1/\epsilon^2$ to distinguish LPN samples from random samples with constant advantage. [10]

The SD 2.0 algorithm [24] outperforms the ISD algorithm for the GV bound decoding over $\mathbb{F}_2$ with $k < 0.3N$. Given a $(\mathsf{HW}, \mathbb{F})$-$\mathsf{LPN}(k, N, t)$ instance $\boldsymbol{b} = \mathbf{A} \cdot \boldsymbol{s} + \boldsymbol{e}$, SD 2.0 solves the computational LPN problem by recovering the noise $\boldsymbol{e}$, which introduces a new parameter s and proceeds in two stages as follows:

1. Find a set of parity-check vectors

$$\mathcal{V} \subset \left\{ \boldsymbol{v} \stackrel{\text{def}}{=} \begin{bmatrix} \boldsymbol{v}_1 & \boldsymbol{v}_2 \end{bmatrix} \;\middle|\; \boldsymbol{v}_1 \in \mathbb{F}^s, \boldsymbol{v} \cdot \mathbf{A} = \mathbf{0} \text{ and } |\boldsymbol{v}_2| = w \right\},$$

   with a sufficiently small $w > 0$. Note that for each $\boldsymbol{v}$, $\langle \boldsymbol{v}, \boldsymbol{b} \rangle = \langle \boldsymbol{v}, \mathbf{A} \cdot \boldsymbol{s} \oplus \boldsymbol{e} \rangle = \langle \boldsymbol{v}, \boldsymbol{e} \rangle = \langle \boldsymbol{v}_1, \boldsymbol{e}_1 \rangle \oplus \langle \boldsymbol{v}_2, \boldsymbol{e}_2 \rangle$, where $\boldsymbol{e} \stackrel{\text{def}}{=} \begin{bmatrix} \boldsymbol{e}_1 & \boldsymbol{e}_2 \end{bmatrix}$ with $\boldsymbol{e}_2 \in \mathbb{F}^{N-s}$. The SD 2.0 algorithm [24] assumes that $\langle \boldsymbol{v}_2, \boldsymbol{e}_2 \rangle$ independently follows a Bernoulli distribution $\mathsf{Ber}_{\epsilon,1}(\mathbb{F})$ with some $\epsilon > 0$ for each $\boldsymbol{v} \in \mathcal{V}'$.

2. Recover $\boldsymbol{e}_1 \in \mathbb{F}^s$ from

$$\left\{ (\boldsymbol{v}_1, \langle \boldsymbol{v}_1, \boldsymbol{e}_1 \rangle \oplus \Delta) \;\middle|\; \begin{bmatrix} \boldsymbol{v}_1 & \boldsymbol{v}_2 \end{bmatrix} \in \mathcal{V}', \boldsymbol{v}_1 \in \mathbb{F}^s \text{ and } \Delta \leftarrow \mathsf{Ber}_{\epsilon,1}(\mathbb{F}) \right\}$$

   via a fast Fourier transform in time $O(s \log(|\mathbb{F}|)|\mathbb{F}|^s)$.

[9] The Bernoulli distribution admits a slackness event that the weight of a parity-check vector $\boldsymbol{v}$ goes much below the expected w leading to underestimated attack costs. However, for an optimal weight w, such low-weight vectors $\boldsymbol{v}$'s violating the Gilbert–Varshamov bound may not exist at all.

[10] We shall distinguish the differences between $1/\epsilon$ and $1/\epsilon^2$. If a single *key-recovery* attack succeeds with probability ϵ, then repeating roughly $1/\epsilon$ independent instances achieves constant (or even overwhelming) success probability. In contrast, if a single *distinguishing attack* gains advantage ϵ, then the number of independent votes needed to amplify the advantage to constant is about $1/\epsilon^2$ by a Chernoff bound.

where an appropriate value for parameter s is chosen so that the two stages take about the same cost. The procedure to recover other bits of $\boldsymbol{e}$ is likewise. The SD 2.0 algorithm [24] achieves larger advantage, in terms of $(\Pr[\langle \boldsymbol{v}_2, \boldsymbol{e}_2\rangle = 0] - 1/|\mathbb{F}|)$, than that achieved by the traditional SD attack, i.e., $(\Pr[\langle \boldsymbol{v}, \boldsymbol{e}\rangle = 0] - 1/|\mathbb{F}|)$.

In order to compete with the advanced ISD algorithms [72, 36, 59, 13] for the GV bound decoding over $\mathbb{F}_2$, SD 2.0 [24] uses the same collision technique invoked by these ISD algorithms to find set $\mathcal{V}$ with a much smaller w. However, SD 2.0 [24] does not perform well on low-noise LPN, as the collision technique takes exponential time $|\mathbb{F}|^{\theta(k)}$ that is already larger than the overall subexponential security $2^{o(k)}$ of the low-noise LPN problem.

We tweak the SD 2.0 algorithm [24] for analyzing the low-noise LPN by replacing the collision technique with other collision techniques, e.g., [17]. Note that our adapted SD 2.0 attack is considered to cover the traditional SD attack with the same collision technique (by setting $s = 0$). Therefore, our analysis of the adapted SD 2.0 below also implicitly recovers that of the traditional SD attack. We consider two cases: binary field $\mathbb{F}_2$ and larger fields.

Binary field $\mathbb{F}_2$. Given a $(\mathsf{HW}, \mathbb{F})$-$\mathsf{LPN}(k, N, t)$ instance, the adapted SD 2.0 finds parity-check vector-set with Hamming weight w. According to [58, Lemma 3], we have that

$$\epsilon_s = \Pr[\langle \boldsymbol{v}_2, \boldsymbol{e}_2\rangle = 0] - 1/|\mathbb{F}| = \frac{1}{2}\Big(\frac{N-2w-t+1}{N-t+1}\Big)^t ,$$

and the adapted algorithm solves the problem in time $T = \min_s \left(T_1 \cdot (1/\epsilon_s)^2 + s2^s\right)$, where $w = (k+1-s)/2$ and assume $T_1 = k+1$ following [17].

Larger fields. We analyze the cost of solving a $(\mathsf{HW}, \mathbb{F})$-$\mathsf{LPN}(k, N, t)$ problem using our adapted SD 2.0 algorithm and the traditional SD attack over field of size $|\mathbb{F}| \geq (2+c)t$ for some constant $c > 0$. In particular, we have the following theorem.

Theorem 10. *For $w = w(s) \in \mathbb{N}$ and a finite field $\mathbb{F}$ with size $|\mathbb{F}| \geq 4t$, the adapted SD 2.0 algorithm solves the $(\mathsf{HW}, \mathbb{F})$-$\mathsf{LPN}(k, N, t)$ problem in time*

$$T = \min_s \left(T_1 \cdot \left(\frac{\binom{N}{t}}{\binom{N-w}{t}} \cdot \frac{2|\mathbb{F}|}{|\mathbb{F}|-1} \right)^2 + s \log\left(|\mathbb{F}|\right) |\mathbb{F}|^s \right) ,$$

where T_1 is the time of finding one parity check vector.

The proof of Theorem 10 is postponed to Appendix E. We set $w = k - s + 1$ and assume $T_1 = k + 1$ following the work [17]. For the case of a field $\mathbb{F}$ with $2 < |\mathbb{F}| < 4t$, we can still use above formula to estimate the cost of our adapted SD 2.0 algorithm, which can be only smaller than the actual cost and makes our analysis more conservative.

Below, we show that for solving LPN problems over large fields, both the optimal SD attack and the SD 2.0 attack (adapted to the low-noise setting) take more cost than the Prange's ISD algorithm [67]. In the following theorem, we assume that LPN problems adopt random linear codes as in [30]. The proof of Theorem 11 is postponed to Appendix E.

Theorem 11. *For any $(\mathsf{HW}, \mathbb{F})$-$\mathsf{LPN}(k, N, t)$ problem with $|\mathbb{F}| = k^{\omega(1)}$, $(1+\beta)k \leq N = \mathsf{poly}(k)$ for a constant $\beta > 0$ and $t = o(N)$, both the traditional SD attack and our adapted SD 2.0 attack (that distinguishes with constant advantage) require more cost than the Prange's ISD algorithm (that recovers the secret with constant probability).*

Acknowledgements

Work of Xiao Wang is supported in part by DARPA under Contract No. HR001120C0087, NSF award #2016240, research awards from Facebook and Google, and a Gift from PlatON. The views, opinions, and/or findings expressed are those of the author(s) and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government. Work of Kang Yang is supported by the National Natural Science Foundation of China (Grant Nos. 62102037, 61932019, 62022018). Work of Yu Yu is supported by the National Key Research and Development Program of China (Grant Nos. 2020YFA0309705 and 2018YFA0704701) and the National Natural Science Foundation of China (Grant Nos. 62125204 and 61872236). Yu Yu also acknowledges the support from the XPLORER PRIZE.

References

[1] Carlos Aguilar-Melchor, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, and Gilles Zémor. Efficient encryption from random quasi-cyclic codes. *IEEE Transactions on Information Theory*, 64, 2018.

[2] Adi Akavia. *Learning Noisy Characters, Multiplication Codes, and Cryptographic Hardcore Predicates*. PhD thesis, Massachusetts Institute of Technology, 2008.

[3] A. Kh. Al Jabri. A statistical decoding algorithm for general linear block codes. In Bahram Honary, editor, *8th IMA International Conference on Cryptography and Coding*, volume 2260 of *LNCS*. Springer, December 17–19, 2001.

[4] Michael Alekhnovich. More on average case vs approximation complexity. In *44th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2003.

[5] Benny Applebaum. Garbling XOR gates "for free" in the standard model. *J. Cryptology*, 29(3), July 2016.

[6] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography with constant input locality. In *Advances in Cryptology—Crypto 2007*, volume 4622 of *LNCS*. Springer, 2007.

[7] Marco Baldi, Alessandro Barenghi, Franco Chiaraluce, Gerardo Pelosi, and Paolo Santini. A finite regime analysis of information set decoding algorithms. *Algorithms*, 12(10), 2019.

[8] Gregory Bard and Martin Albrecht. M4ri(e)- linear algebra over $\mathbf{F}_2$ (and $\mathbf{F}_{2^e}$). In *Free Open Source Software*, 2021.

[9] Carsten Baum, Lennart Braun, Alexander Munch-Hansen, Benoît Razet, and Peter Scholl. Appenzeller to brie: Efficient zero-knowledge proofs for mixed-mode arithmetic and Z2k. In *ACM Conf. on Computer and Communications Security (CCS) 2021*. ACM Press, 2021.

[10] Carsten Baum, Lennart Braun, Alexander Munch-Hansen, and Peter Scholl. Moz$\mathbb{Z}_{2^k}$arella: Efficient vector-ole and zero-knowledge proofs over $\mathbb{Z}_{2^k}$. In *CRYPTO 2022*, Lecture Notes in Computer Science. Springer, 2022.

[11] Carsten Baum, Lennart Braun, Alexander Munch-Hansen, and Peter Scholl. Moz$\mathbb{Z}_{2^k}$arella: Efficient vector-ole and zero-knowledge proofs over $\mathbb{Z}_{2^k}$, August 13–18 2022.

[12] Carsten Baum, Alex J. Malozemoff, Marc B. Rosen, and Peter Scholl. Mac'n'cheese: Zero-knowledge proofs for boolean and arithmetic circuits with nested disjunctions. In *Advances in Cryptology—Crypto 2021, Part IV*, LNCS. Springer, 2021.

[13] Anja Becker, Antoine Joux, Alexander May, and Alexander Meurer. Decoding random binary linear codes in $2^{n/20}$: How 1 + 1 = 0 improves information set decoding. In *Advances in Cryptology—Eurocrypt 2012*, LNCS. Springer, 2012.

[14] Avrim Blum, Merrick L. Furst, Michael J. Kearns, and Richard J. Lipton. Cryptographic primitives based on hard learning problems. In *Advances in Cryptology—Crypto 1993*, LNCS. Springer, 1994.

[15] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. In *32nd Annual ACM Symposium on Theory of Computing (STOC)*. ACM Press, 2000.

[16] Leif Both and Alexander May. Decoding linear codes with high error rate and its impact for LPN security. In *PQCrypto 2018*. Springer, April 9–11 2018.

[17] Elette Boyle, Geoffroy Couteau, Niv Gilboa, and Yuval Ishai. Compressing vector OLE. In *ACM Conf. on Computer and Communications Security (CCS) 2018*. ACM Press, 2018.

[18] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Nicolas Resch, and Peter Scholl. Correlated pseudorandomness from expand-accumulate codes, August 13–18 2022.

[19] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Peter Rindal, and Peter Scholl. Efficient two-round OT extension and silent non-interactive secure computation. In *ACM Conf. on Computer and Communications Security (CCS) 2019*. ACM Press, 2019.

[20] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators: Silent OT extension and more. In *Advances in Cryptology—Crypto 2019, Part III*, volume 11694 of *LNCS*. Springer, 2019.

[21] Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing. In *Advances in Cryptology—Eurocrypt 2015, Part II*, volume 9057 of *LNCS*. Springer, 2015.

[22] Zvika Brakerski, Vadim Lyubashevsky, Vinod Vaikuntanathan, and Daniel Wichs. Worst-case hardness for LPN and cryptographic hashing via code smoothing. In *Advances in Cryptology—Eurocrypt 2019, Part III*, volume 11478 of *LNCS*. Springer, 2019.

[23] Dung Bui and Geoffroy Couteau. Private set intersection from pseudorandom correlation generators. Cryptology ePrint Archive, Paper 2022/334, 2022. https://eprint.iacr.org/2022/334.

[24] Kevin Carrier, Thomas Debris-Alazard, Charles Meyer-Hilfiger, and Jean-Pierre Tillich. Statistical decoding 2.0: Reducing decoding to lpn. In *ASIACRYPT 2022*, Lecture Notes in Computer Science. Springer, 2022.

[25] John T. Coffey and Rodney M. Goodman. The complexity of information set decoding. *IEEE Transactions on Information Theory*, 36(5), 1990.

[26] Geoffroy Couteau, Peter Rindal, and Srinivasan Raghuraman. Silver: Silent VOLE and oblivious transfer from hardness of decoding structured LDPC codes. In *Advances in Cryptology—Crypto 2021, Part III*, LNCS. Springer, 2021.

[27] Ronald Cramer, Ivan Damgård, Daniel Escudero, Peter Scholl, and Chaoping Xing. SPD $\mathbb{Z}\_2^k$: Efficient MPC mod 2^k for dishonest majority. In *Advances in Cryptology—Crypto 2018, Part II*, volume 10992 of *LNCS*. Springer, 2018.

[28] Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In *Advances in Cryptology—Crypto 2012*, volume 7417 of *LNCS*. Springer, 2012.

[29] Bernardo David, Rafael Dowsley, and Anderson C. A. Nascimento. Universally composable oblivious transfer based on a variant of LPN. In *CANS 14 International Conference on Cryptology and Network Security*, LNCS. Springer, 2014.

[30] Thomas Debris-Alazard and Jean-Pierre Tillich. Statistical decoding. In *ISIT 2017*, 2017.

[31] Claire Delaplace, Andre Esser, and Alexander May. Improved low-memory subset sum and LPN algorithms via multiple collisions. In Martin Albrecht, editor, *17th IMA International Conference on Cryptography and Coding*, volume 11929 of *LNCS*, Oxford, UK, December 16–18, 2019. Springer.

[32] Samuel Dittmer, Yuval Ishai, Steve Lu, and Rafail Ostrovsky. Authenticated garbling from simple correlations. LNCS. Springer, August 13–18 2022.

[33] Samuel Dittmer, Yuval Ishai, Steve Lu, and Rafail Ostrovsky. Improving line-point zero knowledge: Two multiplications for the price of one. Cryptology ePrint Archive, Report 2022/552, 2022. https://ia.cr/2022/552.

[34] Samuel Dittmer, Yuval Ishai, and Rafail Ostrovsky. Line-point zero knowledge and its applications. In *ITC 2021*, volume 199, 2021.

[35] Erez Druk and Yuval Ishai. Linear-time encodable codes meeting the gilbert-varshamov bound and their cryptographic applications. In Moni Naor, editor, *ITCS 2014: 5th Conference on Innovations in Theoretical Computer Science*, Princeton, NJ, USA, January 12–14, 2014.

[36] Ilya Dumer. On minimum distance decoding of linear codes. In *Proc. 5th Joint Soviet-Swedish Int. Workshop Inform. Theory*, 1991.

[37] Andre Esser and Emanuele Bellini. Syndrome decoding estimator. In *PKC 2022*, volume 13177, 2022.

[38] Andre Esser, Felix Heuer, Robert Kübler, Alexander May, and Christian Sohler. Dissection-BKW. In *Advances in Cryptology—Crypto 2018, Part II*, volume 10992 of *LNCS*. Springer, 2018.

[39] Andre Esser, Robert Kübler, and Alexander May. LPN decoded. In *Advances in Cryptology—Crypto 2017, Part II*, volume 10402 of *LNCS*. Springer, 2017.

[40] Andre Esser, Robert Kübler, and Floyd Zweydinger. A faster algorithm for finding closest pairs in hamming metric. In *FSTTCS 2021*, volume 213, 2021.

[41] Andre Esser, Alexander May, and Floyd Zweydinger. Mceliece needs a break – solving mceliece-1284 and quasi-cyclic-2918 with modern isd. Cryptology ePrint Archive, Report 2021/1634, 2021. https://ia.cr/2021/1634.

[42] Thibauld Feneuil, Antoine Joux, and Matthieu Rivain. Syndrome decoding in the head: Shorter signatures from zero-knowledge proofs. In *CRYPTO 2022*, Lecture Notes in Computer Science. Springer, 2022.

[43] Matthieu Finiasz and Nicolas Sendrier. Security bounds for the design of code-based cryptosystems. In *Advances in Cryptology—Asiacrypt 2009*, volume 5912 of *LNCS*. Springer, 2009.

[44] Marc P. C. Fossorier, Kazukuni Kobara, and Hideki Imai. Modeling bit flipping decoding based on nonorthogonal check sums with application to iterative decoding attack of mceliece cryptosystem. *IEEE Trans. Inf. Theory*, 53(1), 2007.

[45] Nicholas Franzese, Jonathan Katz, Steve Lu, Rafail Ostrovsky, Xiao Wang, and Chenkai Weng. Constant-overhead zero-knowledge for RAM programs. In *ACM Conf. on Computer and Communications Security (CCS) 2021*. ACM Press, 2021.

[46] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. Concrete mathematics: a foundation for computer science. 1994. 2nd edition.

[47] Cheikh Thiécoumba Gueye, Jean Belo Klamti, and Shoichi Hirose. Generalization of BJMM-ISD using may-ozerov nearest neighbor algorithm over an arbitrary finite field $\mathbb{F}_q$. In *International Conference on Codes, Cryptology, and Information Security*, volume 10194, 2017.

[48] Yann Hamdaoui and Nicolas Sendrier. A non asymptotic analysis of information set decoding. Cryptology ePrint Archive, Report 2013/162, 2013. https://eprint.iacr.org/2013/162.

[49] Carmit Hazay, Emmanuela Orsini, Peter Scholl, and Eduardo Soria-Vazquez. TinyKeys: A new approach to efficient multi-party computation. In *Advances in Cryptology—Crypto 2018, Part III*, volume 10993 of *LNCS*. Springer, 2018.

[50] Carmit Hazay, Peter Scholl, and Eduardo Soria-Vazquez. Low cost constant round MPC combining BMR and oblivious transfer. *J. Cryptology*, 33(4), October 2020.

[51] Shoichi Hirose. May-ozerov algorithm for nearest-neighbor problem over $\mathbb{F}(q)$ and its application to information set decoding. In *SECITC 2016*, volume 10006, 2016.

[52] Nicholas J. Hopper and Manuel Blum. Secure human identification protocols. In *Advances in Cryptology—Asiacrypt 2001*, LNCS. Springer, 2001.

[53] Nick Howgrave-Graham and Antoine Joux. New generic algorithms for hard knapsacks. In *Advances in Cryptology—Eurocrypt 2010*, LNCS. Springer, 2010.

[54] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography with constant computational overhead. In *40th Annual ACM Symposium on Theory of Computing (STOC)*. ACM Press, 2008.

[55] Abhishek Jain, Stephan Krenn, Krzysztof Pietrzak, and Aris Tentes. Commitments and efficient zero-knowledge proofs from learning parity with noise. In *Advances in Cryptology—Asiacrypt 2012*, LNCS. Springer, 2012.

[56] Jonathan Katz, Ji Sun Shin, and Adam Smith. Parallel and concurrent security of the HB and HB+ protocols. *J. Cryptology*, 23(3), July 2010.

[57] Hanlin Liu and Yu Yu. A non-heuristic approach to time-space tradeoffs and optimizations for BKW. Cryptology ePrint Archive, Report 2021/1343, 2021. https://eprint.iacr.org/2021/1343.

[58] Vadim Lyubashevsky. The parity problem in the presence of noise, decoding random linear codes, and the subset sum problem. In *RANDOM 2005*, volume 3624, 2005.

[59] Alexander May, Alexander Meurer, and Enrico Thomae. Decoding random linear codes in $\tilde{\mathcal{O}}(2^{0.054n})$. In *Advances in Cryptology—Asiacrypt 2011*, LNCS. Springer, 2011.

[60] Alexander May and Ilya Ozerov. On computing nearest neighbors with applications to decoding of binary linear codes. In *Advances in Cryptology—Eurocrypt 2015, Part I*, volume 9056 of *LNCS*. Springer, 2015.

[61] Alexander Meurer. A coding-theoretic approach to cryptanalysis. Universität Bochum Ruhr, November 2012. Dissertation thesis.

[62] Daniele Micciancio and Petros Mol. Pseudorandom knapsacks and the sample complexity of LWE search-to-decision reductions. In *Advances in Cryptology—Crypto 2011*, volume 6841 of *LNCS*. Springer, 2011.

[63] Rafael Misoczki, Jean-Pierre Tillich, Nicolas Sendrier, and Paulo S. L. M. Barreto. Mdpc-mceliece: New mceliece variants from moderate density parity-check codes. In *Proceedings of the 2013 IEEE International Symposium on Information Theory, 2013*, pages 2069–2073. IEEE, 2013.

[64] Jesper Buus Nielsen, Peter Sebastian Nordholt, Claudio Orlandi, and Sai Sheshank Burra. A new approach to practical active-secure two-party computation. In *Advances in Cryptology—Crypto 2012*, volume 7417 of *LNCS*. Springer, 2012.

[65] Raphael Overbeck. Statistical decoding revisited. In *ACISP 06: 11th Australasian Conference on Information Security and Privacy*, LNCS. Springer, 2006.

[66] Christiane Peters. Information-set decoding for linear codes over F_q. In Nicolas Sendrier, editor, *The Third International Workshop on Post-Quantum Cryptography, PQCRYPTO 2010*, Darmstadt, Germany, May 25–28 2010. Springer.

[67] Eugene Prange. The use of information sets in decoding cyclic codes. *IRE Trans. Inf. Theory*, 8, 1962.

[68] Srinivasan Raghuraman and Peter Rindal. Blazing fast psi from improved okvs and subfield vole. Cryptology ePrint Archive, Paper 2022/320, 2022. https://eprint.iacr.org/2022/320.

[69] Peter Rindal and Phillipp Schoppmann. VOLE-PSI: Fast OPRF and circuit-PSI from vector-OLE. In *Advances in Cryptology—Eurocrypt 2021, Part II*, LNCS. Springer, 2021.

[70] Phillipp Schoppmann, Adrià Gascón, Leonie Reichert, and Mariana Raykova. Distributed vector-OLE: Improved constructions and implementation. In *ACM Conf. on Computer and Communications Security (CCS) 2019*. ACM Press, 2019.

[71] Nicolas Sendrier. Decoding one out of many. In Bo-Yin Yang, editor, *Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011*, Tapei, Taiwan, November 29 – December 2 2011. Springer.

[72] Jacques Stern. A method for finding codewords of small weight. In *Coding Theory and Applications*, volume 388, 1988.

[73] Rodolfo Canto Torres and Nicolas Sendrier. Analysis of information set decoding for a sub-linear error weight. In Tsuyoshi Takagi, editor, *Post-Quantum Cryptography - 7th International Workshop, PQCrypto 2016*, Fukuoka, Japan, February 24–26 2016. Springer.

[74] David Wagner. A generalized birthday problem. In *Advances in Cryptology—Crypto 2002*, volume 2442 of *LNCS*. Springer, 2002.

[75] Xiao Wang, Samuel Ranellucci, and Jonathan Katz. Authenticated garbling and efficient maliciously secure two-party computation. In *ACM Conf. on Computer and Communications Security (CCS) 2017*. ACM Press, 2017.

[76] Chenkai Weng, Kang Yang, Jonathan Katz, and Xiao Wang. Wolverine: Fast, scalable, and communication-efficient zero-knowledge proofs for boolean and arithmetic circuits. In *IEEE Symp. Security and Privacy 2021*. IEEE, 2021.

[77] Chenkai Weng, Kang Yang, Xiang Xie, Jonathan Katz, and Xiao Wang. Mystique: Efficient conversions for zero-knowledge proofs with applications to machine learning. In *USENIX Security Symposium 2021*. USENIX Association, 2021.

[78] Kang Yang, Pratik Sarkar, Chenkai Weng, and Xiao Wang. QuickSilver: Efficient and affordable zero-knowledge proofs for circuits and polynomials over any field. In *ACM Conf. on Computer and Communications Security (CCS) 2021*. ACM Press, 2021.

[79] Kang Yang, Chenkai Weng, Xiao Lan, Jiang Zhang, and Xiao Wang. Ferret: Fast extension for correlated OT with small communication. In *ACM Conf. on Computer and Communications Security (CCS) 2020*. ACM Press, 2020.

[80] Yu Yu and John P. Steinberger. Pseudorandom functions in almost constant depth from low-noise LPN. In *Advances in Cryptology—Eurocrypt 2016, Part II*, volume 9666 of *LNCS*. Springer, 2016.

[81] Yu Yu, Jiang Zhang, Jian Weng, Chun Guo, and Xiangxue Li. Collision resistant hashing from subexponential learning parity with noise. In *Advances in Cryptology—Asiacrypt 2019, Part II*, LNCS. Springer, 2019.

A Proofs of Theorems and Lemmas

Corollary 2. If decisional $(\mathsf{Ber}, \mathbb{F}_2)$-$\mathsf{LPN}(k, N, \mu/2^\lambda)$ is hard, then computational $(\mathsf{HW}, \mathbb{Z}_{2^\lambda})$-$\mathsf{LPN}\big(k, N, t = (1-2^{-\lambda})\mu N\big)$ is hard.

Proof. By Theorem 4 it suffices to show that decisional $(\mathsf{Ber}, \mathbb{Z}_{2^\lambda})$-LPN implies its computational analogue, which in turns implies computational $(\mathsf{HW}, \mathbb{Z}_{2^\lambda})$-LPN. The former is trivial and thus it left out to show the latter. The difference is that for $\boldsymbol{e} \leftarrow \mathsf{Ber}_{\mu,N}(\mathbb{Z}_{2^\lambda})$ $\boldsymbol{e}$ has expected (instead of exact) Hamming weight $\mu N \frac{2^\lambda - 1}{2^\lambda}$. The difference is not substantial for computational problems as conditioned on $|\boldsymbol{e}| = (1 - 2^{-\lambda})\mu N$, which has probability $\Omega(1/\sqrt{N})$, $\boldsymbol{e}$ follows the exact-weight distribution $\mathsf{HW}_{(1-2^{-\lambda})\mu N, N}(\mathbb{Z}_{2^\lambda})$, which completes the proof. □

Theorem 7. If computational $(\mathsf{HW}, \mathbb{Z}_{2^\lambda})$-$\mathsf{LPN}(k, N, t)$ is $(\lambda T + \mathsf{poly}(k, N), \epsilon^{\lambda+1})$-hard, then computational $(\mathsf{HW}, \mathbb{F}_2)$-$\mathsf{LPN}(k, N, t')$ is $(T, \frac{2\epsilon}{1-\exp(-\delta^2 t/6)})$-hard, where $t' = t\frac{2^{(\lambda-1)}}{2^\lambda - 1}(1+\delta)$ for any constant $\delta > 0$.

Proof. Similar to the proof of Theorem 6, we can show if there exists $\mathcal{A}_{\mathrm{LPN}_2}$ that breaks LPN over $\mathbb{F}_2$ and noise $\widetilde{\mathsf{HW}}_{t,N}(\mathbb{F}_2)$ with probability more than 2ϵ within time T, then it can be used to breaks LPN over $\mathbb{Z}_{2^\lambda}$ and noise $\mathsf{HW}_{t,N}(\mathbb{Z}_{2^\lambda})$ with probability $\epsilon^{\lambda+1}$, where $\widetilde{\mathsf{HW}}_{t,N}(\mathbb{F}_2)$ is the distribution of $\boldsymbol{e}^i$ when $\boldsymbol{e} \leftarrow \mathsf{HW}_{t,N}(\mathbb{Z}_{2^\lambda})$. The expected weight of $\boldsymbol{e}^i$ is $t\frac{2^{\lambda-1}}{2^\lambda-1}$, and thus by a Chernoff bound $\boldsymbol{e}^i$ is a convex combination of distributions $\mathsf{HW}_{1,N}(\mathbb{F}_2), \ldots, \mathsf{HW}_{t',N}(\mathbb{F}_2)$ with $t' = t\frac{2^{\lambda-1}}{2^\lambda-1}(1+\delta)$ and $\delta > 0$ except for an error probability bounded by $\exp(-\delta^2 t/6)$. Since $\mathcal{A}_{\mathrm{LPN}_2}$ works on LPN over $\mathbb{F}_2$ with noise $\mathsf{HW}_{t',N}(\mathbb{F}_2)$, it should work on that with noise $\mathsf{HW}_{i,N}(\mathbb{F}_2)$ of weight up to $i = t'$ (and any their convex combination) as well.[11] Therefore, $\mathcal{A}_{\mathrm{LPN}_2}$ that breaks $(\mathsf{HW}, \mathbb{F}_2)$-$\mathsf{LPN}(k, N, t')$ with probability $2\epsilon/(1-\exp(-\delta^2 t/6))$ is the hypothetical algorithm desired to complete the proof. □

Theorem 9. If the computational $(\mathsf{HW}, \mathbb{F}_2)$-$\mathsf{LPN}(k, N, t')$ problem can be broken (say by $\mathcal{A}_{\mathrm{LPN}_2}$) in time T with success probability at least $(1-\epsilon/2)$, then the computational $(\mathsf{HW}, \mathbb{Z}_{2^\lambda})$-$\mathsf{LPN}(k, N, t)$ problem can be broken by $\mathcal{A}_{\mathrm{LPN}_{2^\lambda}}$ (see Algorithm 1) in time $\lambda T + \mathsf{poly}(k, N)$ with success probability at least $1-(\lambda+1)\sqrt{\epsilon}$, where $t' = t\frac{2^{\lambda-1}}{2^\lambda-1}(1+\delta)$ for any δ and ϵ satisfying $\delta^2 t \geq 6\ln(2/\epsilon)$.

Proof. Similar to the proof of Theorem 8, as long as there is $\mathcal{A}_{\mathrm{LPN}_{2^\lambda}}$ that succeeds in breaking the LPN over $\mathbb{F}_2$ and noise $\boldsymbol{e}^i$ with probability at least $(1-\epsilon)$, then the rest follows from Markov inequality and a union bound. As analyzed in the proof of Theorem 7, $\boldsymbol{e}^i$ is $\exp(-\delta^2 t/6)$-close to a convex combination of distributions $\mathsf{HW}_{1,N}(\mathbb{F}_2), \ldots, \mathsf{HW}_{t',N}(\mathbb{F}_2)$ with $t' = t\frac{2^{\lambda-1}}{2^\lambda-1}(1+\delta)$ and $\delta > 0$. Therefore, we need $\mathcal{A}_{\mathrm{LPN}_{2^\lambda}}$ to be successful on over $\mathbb{F}_2$ and noise $\mathsf{HW}_{t',N}(\mathbb{F}_2)$ with probability at least

$$\frac{1-\epsilon}{1-\exp(-\delta^2 t/6)} \leq 1 - \Big(\epsilon - \exp(-\delta^2 t/6)\Big) \leq 1 - \frac{\epsilon}{2} \ .$$

□

Lemma 5. *Let* $(\mathbf{A}, \boldsymbol{b} = \mathbf{A}\cdot\boldsymbol{s} + \boldsymbol{e} \mod p^{\lambda_1} q^{\lambda_2})$ *be the LPN samples over* $\mathbb{Z}_{p^{\lambda_1} q^{\lambda_2}}$*, then* $(\mathbf{A}', \boldsymbol{b}')$ *as defined as in line* 6 *(resp., line* 9*) of Algorithm 2 constitute the LPN samples over* $\mathbb{Z}_{p^{\lambda_1 - 1} q^{\lambda_2}}$ *(resp., over* $\mathbb{Z}_{q^{\lambda_2 - 1}}$*).*

Proof. Let $(\mathbf{A}^0, \mathbf{A}^1, \cdots, \mathbf{A}^{\lambda-1})$, $(\boldsymbol{b}^0, \boldsymbol{b}^1, \cdots, \boldsymbol{b}^{\lambda-1})$ be as per Algorithm 1, and let $(\boldsymbol{s}^0, \boldsymbol{s}^1, \cdots, \boldsymbol{s}^{\lambda-1}) := \mathsf{DigitDecomp}(\boldsymbol{s})$, $(\boldsymbol{e}^0, \boldsymbol{e}^1, \cdots, \boldsymbol{e}^{\lambda-1}) := \mathsf{DigitDecomp}(\boldsymbol{e})$. The proof of the statement about $(\mathbf{A}', \boldsymbol{b}')$ defined as in line 6 and line 9 are similar with that of Lemma 4. For the statement about $(\mathbf{A}', \boldsymbol{b}')$ defined as in

[11] Strictly speaking, $\mathcal{A}_{\mathrm{LPN}_2}$ implies such an algorithm with roughly the same complexity and success probability, which can be seen by a simple reduction.

line 6, it suffices to prove $\boldsymbol{b}' = \mathbf{A}' \cdot \boldsymbol{s}' + \boldsymbol{e}' \mod p^{\lambda_1 - 1} q^{\lambda_2}$, where $\boldsymbol{s}' = (\boldsymbol{s} - \boldsymbol{s}^0)/p$ and $\boldsymbol{e}' = (\boldsymbol{e} - \boldsymbol{e}^0)/p$ are the secret and noise of the LPN over $\mathbb{Z}_{p^{\lambda_1 - 1} q^{\lambda_2}}$ respectively. That is,

$$\begin{aligned} p\boldsymbol{b}' - p(\mathbf{A}' \cdot \boldsymbol{s}' + \boldsymbol{e}') &= \boldsymbol{b} - \mathbf{A} \cdot \boldsymbol{s}^0 - \boldsymbol{e}^0 - p(\mathbf{A}' \cdot \boldsymbol{s}' + \boldsymbol{e}') \\ &= \boldsymbol{b} - \mathbf{A} \cdot \boldsymbol{s}^0 - \boldsymbol{e}^0 - \mathbf{A} \cdot (\boldsymbol{s} - \boldsymbol{s}^0) - \boldsymbol{e} + \boldsymbol{e}^0 \\ &= \boldsymbol{b} - \mathbf{A} \cdot \boldsymbol{s} - \boldsymbol{e} = \mathbf{0} \mod p^{\lambda_1} q^{\lambda_2} , \end{aligned}$$

where the first equality follows from $\boldsymbol{b}' = (\boldsymbol{b} - \mathbf{A} \cdot \boldsymbol{s}^0 - \boldsymbol{e}^0)/p$, the second is due to $p\mathbf{A}' \cdot \boldsymbol{s}' = p\mathbf{A} \cdot \boldsymbol{s}' \mod p^{\lambda_1} q^{\lambda_2}$ and $p\boldsymbol{e}' = \boldsymbol{e} - \boldsymbol{e}^0$, and the last is by the assumption of LPN over $\mathbb{Z}_{p^{\lambda_1 - 1} q^{\lambda_2}}$. For the statement about $(\mathbf{A}', \boldsymbol{b}')$ defined as in line 9, it suffices to prove $\boldsymbol{b}' = \mathbf{A}' \cdot \boldsymbol{s}' + \boldsymbol{e}' \mod q^{\lambda_2 - 1}$, where $\boldsymbol{s}' = (\boldsymbol{s} - \boldsymbol{s}^0)/q$ and $\boldsymbol{e}' = (\boldsymbol{e} - \boldsymbol{e}^0)/q$ are the secret and noise of the LPN over $\mathbb{Z}_{q^{\lambda_2 - 1}}$ respectively. That is,

$$\begin{aligned} q\boldsymbol{b}' - q(\mathbf{A}' \cdot \boldsymbol{s}' + \boldsymbol{e}') &= \boldsymbol{b} - \mathbf{A} \cdot \boldsymbol{s}^0 - \boldsymbol{e}^0 - q(\mathbf{A}' \cdot \boldsymbol{s}' + \boldsymbol{e}') \\ &= \boldsymbol{b} - \mathbf{A} \cdot \boldsymbol{s}^0 - \boldsymbol{e}^0 - \mathbf{A} \cdot (\boldsymbol{s} - \boldsymbol{s}^0) - \boldsymbol{e} + \boldsymbol{e}^0 \\ &= \boldsymbol{b} - \mathbf{A} \cdot \boldsymbol{s} - \boldsymbol{e} = \mathbf{0} \mod q^{\lambda_2} , \end{aligned}$$

where the first equality follows from $\boldsymbol{b}' = (\boldsymbol{b} - \mathbf{A} \cdot \boldsymbol{s}^0 - \boldsymbol{e}^0)/q$, the second is due to $q\mathbf{A}' \cdot \boldsymbol{s}' = q\mathbf{A} \cdot \boldsymbol{s}' \mod q^{\lambda_2}$ and $q\boldsymbol{e}' = \boldsymbol{e} - \boldsymbol{e}^0$, and the last is by the assumption of LPN over $\mathbb{Z}_{q^{\lambda_2 - 1}}$. □

B The Hardness of LPN over More General Rings

We generalize the reductions between LPN over $\mathbb{Z}_{p^{\lambda_1} q^{\lambda_2}}$ and LPN over $\mathbb{F}_p$/$\mathbb{F}_q$ for distinct primes p, q. Our techniques extend to an arbitrary integer ring with more than two prime factors as well.

We recall that every number $\boldsymbol{a} \in [p^{\lambda_1} q^{\lambda_2}]$ can be uniquely represented using the multi-base (1, p, …, p^{λ_1}, $p^{\lambda_1} q$, …, $p^{\lambda_1} q^{\lambda_2 - 1}$) for distinct primes p, q. We define function $\mathsf{DigitDecomp}$ that decomposes a number/vector/matrix $\boldsymbol{a} \in [p^{\lambda_1} q^{\lambda_2}]^{dim}$ (where $dim = 1, n, n_1 \times n_2$ for number, vector, or matrix respectively by applying the operation component-wise) into the above multi-base representation, i.e.,

$$\mathsf{DigitDecomp}(\boldsymbol{a}) = (\boldsymbol{a}^0, \boldsymbol{a}^1, \cdots, \boldsymbol{a}^{\lambda_1 + \lambda_2 - 1})$$

such that $\boldsymbol{a}^i \in [p]^{dim}$ and $\boldsymbol{a}^j \in [q]^{dim}$ for every $i \in [\lambda_1]$ and $j \in [\lambda_1, \lambda_1 + \lambda_2]$, and $\boldsymbol{a} = \sum_{i=0}^{\lambda_1 - 1} p^i \boldsymbol{a}^i + \sum_{j=\lambda_1}^{\lambda_1 + \lambda_2 - 1} p^{\lambda_1} q^{j - \lambda_1} \boldsymbol{a}^j$, and its inverse $\mathsf{DigitDecomp}^{-1}$ such that $\boldsymbol{a} = \mathsf{DigitDecomp}^{-1}(\boldsymbol{a}^0, \boldsymbol{a}^1, \cdots, \boldsymbol{a}^{\lambda_1 + \lambda_2 - 1})$.

We show how to reduce the hardness of both decisional LPN over $\mathbb{F}_p$ and decisional LPN over $\mathbb{F}_q$ to that of decisional LPN over $\mathbb{Z}_{p^{\lambda_1} q^{\lambda_2}}$ with the noise distributions, $\mathsf{IndBer}_{\mu,N}(\mathbb{Z}_{p^{\lambda_1} q^{\lambda_2}})$ and $\mathsf{IndHW}_{t,N}(\mathbb{Z}_{p^{\lambda_1} q^{\lambda_2}})$, extending the noise distributions $\mathsf{IndBer}_{\mu,N}$ and $\mathsf{IndHW}_{t,N}$ from over $\mathbb{Z}_{2^\lambda}$ to $\mathbb{Z}_{p^{\lambda_1} q^{\lambda_2}}$.

- $\boldsymbol{e} \leftarrow \mathsf{IndBer}_{\mu,N}(\mathbb{Z}_{p^{\lambda_1} q^{\lambda_2}})$ refers to $\boldsymbol{e} := \mathsf{DigitDecomp}^{-1}(\boldsymbol{e}^0, \boldsymbol{e}^1, \cdots, \boldsymbol{e}^{\lambda_1 + \lambda_2 - 1})$ with $\boldsymbol{e}^i \leftarrow \mathsf{Ber}_{\mu,N}(\mathbb{F}_p)$ and $\boldsymbol{e}^j \leftarrow \mathsf{Ber}_{\mu,N}(\mathbb{F}_q)$ for $i \in [\lambda_1]$ and $j \in [\lambda_1, \lambda_1 + \lambda_2]$.
- $\boldsymbol{e} \leftarrow \mathsf{IndHW}_{t,N}(\mathbb{Z}_{p^{\lambda_1} q^{\lambda_2}})$ means $\boldsymbol{e} := \mathsf{DigitDecomp}^{-1}(\boldsymbol{e}^0, \boldsymbol{e}^1, \cdots, \boldsymbol{e}^{\lambda_1 + \lambda_2 - 1})$ with $\boldsymbol{e}^i \leftarrow \mathsf{HW}_{t,N}(\mathbb{F}_p)$ and $\boldsymbol{e}^j \leftarrow \mathsf{HW}_{t,N}(\mathbb{F}_q)$ for $i \in [\lambda_1]$ and $j \in [\lambda_1, \lambda_1 + \lambda_2]$.

Theorem 12 (Equivalence of Decisional LPN over $\mathbb{Z}_{p^{\lambda_1} q^{\lambda_2}}$ and $\mathbb{F}_p/\mathbb{F}_q$)**.**

1. *Both decisional* $(\mathcal{D}_1, \mathbb{F}_p)$-$\mathsf{LPN}(k, N, w)$ *and decisional* $(\mathcal{D}_1, \mathbb{F}_q)$-$\mathsf{LPN}(k, N, w)$ *are hard iff decisional* $(\mathcal{D}_2, \mathbb{Z}_{p^{\lambda_1} q^{\lambda_2}})$-$\mathsf{LPN}(k, N, w)$ *is hard.*

Algorithm 2: $\mathcal{A}_{\mathsf{LPN}_{p^{\lambda_1}q^{\lambda_2}}}$, the secret recovery algorithm on LPN over $\mathbb{Z}_{p^{\lambda_1}q^{\lambda_2}}$ $(\lambda_1+\lambda_2 \geq 2)$ with oracle access to $\mathcal{A}_{\mathsf{LPN}_r}$ (the solver for LPN over $\mathbb{F}_r$) and $r \in \{p, q\}$.

Input: $(\mathcal{D}, \mathbb{Z}_{p^{\lambda_1}q^{\lambda_2}})$-$\mathsf{LPN}(k, N, t)$ samples $(\mathbf{A}, \boldsymbol{b} = \mathbf{A} \cdot \boldsymbol{s} + \boldsymbol{e} \mod p^{\lambda_1}q^{\lambda_2})$
Output: $\boldsymbol{s} \in \mathbb{Z}_{p^{\lambda_1}q^{\lambda_2}}$

$(\mathbf{A}^0, \mathbf{A}^1, \cdots, \mathbf{A}^{\lambda_1+\lambda_2-1}) := \mathsf{DigitDecomp}(\mathbf{A})$;
$(\boldsymbol{b}^0, \boldsymbol{b}^1, \cdots, \boldsymbol{b}^{\lambda_1+\lambda_2-1}) := \mathsf{DigitDecomp}(\boldsymbol{b})$;
if $\lambda_1 >= 1$ **then**
  $(\boldsymbol{s}^0, \boldsymbol{e}^0) \leftarrow \mathcal{A}_{\mathsf{LPN}_p}(\mathbf{A}^0, \boldsymbol{b}^0)$;
  $\boldsymbol{b}' := (\boldsymbol{b} - \mathbf{A} \cdot \boldsymbol{s}^0 - \boldsymbol{e}^0)/p$;
  Return $\boldsymbol{s} = \boldsymbol{s}^0 + p \cdot \mathcal{A}_{\mathsf{LPN}_{p^{\lambda_1-1}q^{\lambda_2}}}\left(\mathbf{A}' := \mathbf{A} \mod p^{\lambda_1-1}q^{\lambda_2},\ \boldsymbol{b}'\right)$;
$(\boldsymbol{s}^0, \boldsymbol{e}^0) \leftarrow \mathcal{A}_{\mathsf{LPN}_q}(\mathbf{A}^0, \boldsymbol{b}^0)$;
$\boldsymbol{b}' := (\boldsymbol{b} - \mathbf{A} \cdot \boldsymbol{s}^0 - \boldsymbol{e}^0)/q$;
Return $\boldsymbol{s} = \boldsymbol{s}^0 + q \cdot \mathcal{A}_{\mathsf{LPN}_{q^{\lambda_2-1}}}\left(\mathbf{A}' := \mathbf{A} \mod q^{\lambda_2-1},\ \boldsymbol{b}'\right)$;

2. *If decisional* $(\mathcal{D}, \mathbb{Z}_{p^{\lambda_1}q^{\lambda_2}})$-$\mathsf{LPN}(k, N, t)$ *is hard, then both of decisional* $(\mathcal{D}, \mathbb{F}_p)$- $\mathsf{LPN}(k, N, w_p)$ *and decisional* $(\mathcal{D}, \mathbb{F}_q)$-$\mathsf{LPN}(k, N, w_q)$ *are hard.*

where $(\mathcal{D}_1, \mathcal{D}_2, w) \in \{(\mathsf{Ber}, \mathsf{IndBer}, \mu), (\mathsf{HW}, \mathsf{IndHW}, t)\}$ *and* $(\mathcal{D}, w_1, w_p, w_q) \in \{(\mathsf{HW}, t, \frac{(p-1)p^{\lambda_1-1}q^{\lambda_2}}{p^{\lambda_1}q^{\lambda_2}-1} \cdot t, \frac{(q-1)p^{\lambda_1}q^{\lambda_2-1}}{p^{\lambda_1}q^{\lambda_2}-1} \cdot t), (\mathsf{Ber}, \mu, \mu, \mu)\}$.

Proof. The proof of the first statement is essentially similar to that of Theorem 4 and Theorem 5. Let $(\mathbf{A}, \boldsymbol{b} = \mathbf{A} \cdot \boldsymbol{s} + \boldsymbol{e} \mod p^{\lambda_1}q^{\lambda_2})$ be the LPN over $\mathbb{Z}_{p^{\lambda_1}q^{\lambda_2}}$. Decompose the matrices and vectors into the corresponding size-$(\lambda_1+\lambda_2)$ lists, $(\boldsymbol{s}^0, \boldsymbol{s}^1, \cdots, \boldsymbol{s}^{\lambda_1+\lambda_2-1}) := \mathsf{DigitDecomp}(\boldsymbol{s})$, $(\boldsymbol{e}^0, \boldsymbol{e}^1, \cdots, \boldsymbol{e}^{\lambda_1+\lambda_2-1}) := \mathsf{DigitDecomp}(\boldsymbol{e})$ and $(\boldsymbol{b}^0, \boldsymbol{b}^1, \cdots, \boldsymbol{b}^{\lambda_1+\lambda_2-1}) := \mathsf{DigitDecomp}(\boldsymbol{b})$. Therefore, for $i \in [\lambda_1]$ and $j \in [\lambda_1, \lambda_1+\lambda_2]$, we can write

$$\begin{aligned}
\boldsymbol{b}^i &= \mathbf{A}' \cdot \boldsymbol{s}^i + \boldsymbol{e}^i + f_i(\mathbf{A}, \boldsymbol{s}^0, \cdots, \boldsymbol{s}^{i-1}, \boldsymbol{e}^0, \cdots, \boldsymbol{e}^{i-1}) \mod p\ , \\
\boldsymbol{b}^j &= \mathbf{A}'' \cdot \boldsymbol{s}^j + \boldsymbol{e}^j + f_j(\mathbf{A}, \boldsymbol{s}^0, \cdots, \boldsymbol{s}^{j-1}, \boldsymbol{e}^0, \cdots, \boldsymbol{e}^{j-1}) \mod q\ ,
\end{aligned}$$

where $\mathbf{A}' = \mathbf{A} \mod p$, $\mathbf{A}'' = \mathbf{A} \mod q$ and f_i (resp., f_j) is the sum of all other terms involving the individual components of $\boldsymbol{s}$ and $\boldsymbol{e}$ with index up to $i-1$ (resp., $j-1$). Define the hybrid distributions $H_0, \cdots, H_{\lambda_1+\lambda_2}$ as

$$\begin{aligned}
H_0 &= (\mathbf{A}, \boldsymbol{u}_0, \cdots, \boldsymbol{u}_{k-1}, \boldsymbol{u}_k \cdots, \boldsymbol{u}_{\lambda_1+\lambda_2-1}) \\
&\vdots \\
H_k &= (\mathbf{A}, \boldsymbol{b}^0, \cdots, \boldsymbol{b}^{k-1}, \boldsymbol{u}_k \cdots, \boldsymbol{u}_{\lambda_1+\lambda_2-1}) \\
&\vdots \\
H_{\lambda_1+\lambda_2} &= (\mathbf{A}, \boldsymbol{b}^0, \cdots, \boldsymbol{b}^{k-1}, \boldsymbol{b}^k \cdots, \boldsymbol{b}^{\lambda_1+\lambda_2-1})
\end{aligned}$$

where every $\boldsymbol{u}_i \leftarrow \mathbb{F}_p^N$ and $\boldsymbol{u}_j \leftarrow \mathbb{F}_q^N$ is sampled independently for $i \in [\lambda_1]$ and $j \in [\lambda_1, \lambda_1+\lambda_2]$. Note that all the $\boldsymbol{s}^k$'s are independent, and by the definition of $\boldsymbol{e} \leftarrow \mathsf{IndBer}_{\mu,N}(\mathbb{Z}_{p^{\lambda_1}q^{\lambda_2}})$ (resp., $\boldsymbol{e} \leftarrow \mathsf{IndHW}_{t,N}(\mathbb{Z}_{p^{\lambda_1}q^{\lambda_2}})$) we have that $\boldsymbol{e}^i$ follows $\mathsf{Ber}_{\mu,N}(\mathbb{F}_p)$ (resp., $\mathsf{HW}_{t,N}(\mathbb{F}_p)$) and $\boldsymbol{e}^j$ follows $\mathsf{Ber}_{\mu,N}(\mathbb{F}_q)$ (resp., $\mathsf{HW}_{t,N}(\mathbb{F}_q)$) given its (independent) prefix $\boldsymbol{e}^0, \cdots, \boldsymbol{e}^{i-1}$ and prefix $\boldsymbol{e}^0, \cdots, \boldsymbol{e}^{j-1}$ for $i \in [\lambda_1]$ and

$j \in [\lambda_1, \lambda_1+\lambda_2]$. Therefore, all the adjacent H_{i-1} and H_i are computationally indistinguishable and so are H_0 and H_λ by a hybrid argument.

The proof of the second statement is essentially similar to that of Theorem 3. Given the LPN over ring $\mathbb{Z}_{p^{\lambda_1}q^{\lambda_2}}$ samples $(\mathbf{A}, \boldsymbol{b} = \mathbf{A}\cdot\boldsymbol{s}+\boldsymbol{e} \mod p^{\lambda_1}q^{\lambda_2})$, we also observe that the samples $(\mathbf{A}^0 := \mathbf{A} \mod r, \boldsymbol{b}^0 := \boldsymbol{b} \mod r)$ constitute exactly the LPN over $\mathbb{F}_r$ samples for noise $\boldsymbol{e}^0 = \boldsymbol{e} \mod r$, where $r \in \{p, q\}$. In case that $\boldsymbol{e} \leftarrow \mathsf{HW}_{t,N}(\mathbb{Z}_{p^{\lambda_1}q^{\lambda_2}})$, the noise vector $\boldsymbol{e} \mod p$ has expected weight $t_p = \frac{(p-1)p^{\lambda_1-1}q^{\lambda_2}}{p^{\lambda_1}q^{\lambda_2}-1}\cdot t$ and the noise vector $\boldsymbol{e} \mod q$ has expected weight $t_q = \frac{(q-1)p^{\lambda_1}q^{\lambda_2-1}}{p^{\lambda_1}q^{\lambda_2}-1}\cdot t$. This implies that with probability $\Omega(1/\sqrt{t})$ the noise vector $\boldsymbol{e} \mod p$ follows the exact-weight distribution $\mathsf{HW}_{t_p,N}(\mathbb{F}_p)$ and the noise vector $\boldsymbol{e} \mod q$ follows the exact-weight distribution $\mathsf{HW}_{t_q,N}(\mathbb{F}_q)$. The proof for the second statement is likewise, except when taking $\boldsymbol{e} \leftarrow \mathsf{Ber}_{\mu,N}(\mathbb{Z}_{p^{\lambda_1}q^{\lambda_2}})$ we get $\boldsymbol{e} \mod r \sim \mathsf{Ber}_{\mu,N}(\mathbb{F}_r)$, where $r \in \{p, q\}$. □

Similar with the analysis of Theorem 8, we give a reduction both computational LPN over $\mathbb{F}_q$ and $\mathbb{F}_p$ to that over a ring $\mathbb{Z}_{p^{\lambda_1}q^{\lambda_2}}$ (see Theorem 13), extending the corresponding reduction between computational LPN over $\mathbb{F}_2$ and $\mathbb{Z}_{2^\lambda}$. The proof of the below theorem can be found in Appendix A.

Theorem 13. *If computational* $(\mathsf{Ber}, \mathbb{F}_p)$-$\mathsf{LPN}(k, N, \mu)$ *and* $(\mathsf{Ber}, \mathbb{F}_q)$-$\mathsf{LPN}(k, N, \mu)$ *problems can be broken with probability at least* $(1-\epsilon)$ *in time* T_1 *and* T_2 *respectively, then the computational* $(\mathsf{Ber}, \mathbb{Z}_{p^{\lambda_1}q^{\lambda_2}})$-$\mathsf{LPN}(k, N, \mu)$ *problem can be broken in time* $\lambda_1 T_1 + \lambda_2 T_2 + \mathsf{poly}(k, N)$ *with success probability at least* $1-(\lambda_1+\lambda_2+2)\sqrt{\epsilon}$.

Proof. Algorithm 2 translates an $(\mathsf{Ber}, \mathbb{Z}_{p^{\lambda_1}q^{\lambda_2}})$-$\mathsf{LPN}(k, N, \mu)$ instance into λ_1 $(\mathsf{Ber}, \mathbb{F}_p)$-$\mathsf{LPN}(k, N, \mu)$ instances and λ_2 $(\mathsf{Ber}, \mathbb{F}_q)$-$\mathsf{LPN}(k, N, \mu)$ instances, which are independent except that all $(\mathsf{Ber}, \mathbb{F}_p)$-$\mathsf{LPN}(k, N, \mu)$ instances share the same random matrix $\mathbf{A} \mod p$, all $(\mathsf{Ber}, \mathbb{F}_q)$-$\mathsf{LPN}(k, N, \mu)$ instances share the same random matrix $\mathbf{A} \mod q$ and that the noise vectors of the $\lambda_1+\lambda_2$ instances are somehow correlated.

For contradiction assume that there exists an algorithm $\mathcal{A}_{\mathrm{LPN}_p}$ (resp., an algorithm $\mathcal{A}_{\mathrm{LPN}_q}$) that recovers the secret of LPN over $\mathbb{F}_p$ (resp., the secret of LPN over $\mathbb{F}_q$) with probability more than 2ϵ within time T. Similar to the proof of Theorem 6, we consider distribution $\mathsf{Ber}_{\mu,N}(\mathbb{F}_r)$, for $r \in \{p, q\}$, being sampled in two steps: first pick each coordinate with probability with probability μ independently (and let the rest with 0's), and second assign the picked coordinates with uniform random field element.

We have by a Markov inequality that for at least a $(1-\sqrt{\epsilon})$ fraction of $(\mathbf{A} \mod p, \mathsf{coin}_1(\boldsymbol{e}^i))$ $\mathcal{A}_{\mathrm{LPN}_p}$ recovers $\boldsymbol{s}^i$ (for $i \in [\lambda_1]$) from $(\mathbf{A} \mod p, (\mathbf{A} \mod p)\cdot\boldsymbol{s}^i + \boldsymbol{e}^i)$ with probability at least $1-\sqrt{\epsilon}$. Thus, the secret of all $(\mathsf{Ber}, \mathbb{F}_p)$-$\mathsf{LPN}(k, N, \mu)$ instances can be recovered with the following probability by a union bound

$$(1-\sqrt{\epsilon})(1-\lambda_1\sqrt{\epsilon}) \geq 1-(\lambda_1+1)\sqrt{\epsilon} \ .$$

The proof about the $(\mathsf{Ber}, \mathbb{F}_q)$-$\mathsf{LPN}(k, N, \mu)$ instances is likewise, and we have that the secrets of all $(\mathsf{Ber}, \mathbb{F}_q)$-$\mathsf{LPN}(k, N, \mu)$ instances can be recovered with the following probability by a union bound

$$(1-\sqrt{\epsilon})(1-\lambda_2\sqrt{\epsilon}) \geq 1-(\lambda_2+1)\sqrt{\epsilon} \ .$$

Therefore, overall $\mathcal{A}_{\mathrm{LPN}_{p^{\lambda_1}q^{\lambda_2}}}$ succeeds with the following probability by a union bound

$$1-(\lambda_1+\lambda_2+2)\sqrt{\epsilon} \ .$$

□

C Attacks on LPN Problems

In the following, we outline the known attacks on LPN problems and refer the reader to the related works listed below for more details.

Gauss. Gauss attack [39] is the most natural extension of Gaussian elimination to recover the secret vector from an LPN instance with a Bernoulli distribution. This attack guesses a fresh batch of k non-noisy LPN samples by picking them at random in each iteration, inverts the corresponding submatrix, computes a candidate secret $\boldsymbol{s}'$, and then verifies whether $\boldsymbol{s}'$ is correct or not. For LPN with noise rate r, this attack recovers the secret in time $\tilde{\Theta}(1/(1-r)^k)$ using $\tilde{\Theta}(1/(1-r)^k)$ samples, where $r = \mu(|\mathcal{R}|-1)/|\mathcal{R}|$ for a Bernoulli distribution $\mathsf{Ber}_{\mu,N}(\mathcal{R})$. When considering the concrete LPN parameters, the number of samples required is *not* achieved.

To reduce the number of samples in Gauss attack, Esser, Kübler and May [39] introduced *Pooled Gauss* attack, which guesses k non-noisy samples by picking them at random from a pool of the *fixed* $N = k^2 \log^2 k$ LPN samples in each iteration, and then inverts the corresponding subsystem to get a candidate vector $\boldsymbol{s}'$ and verifies if $\boldsymbol{s}'$ is correct. For LPN with noise rate r, this attack recovers the secret in time $\frac{k^3 \log^2 k}{(1-r)^k}$ using $k^2 \log^2 k$ samples. The Pooled Gauss attack [39] only considers the LPN problem with a Bernoulli distribution. In Section 5.1, we extend it to solve LPN and dual-LPN with an exact noise distribution.

As pointed out in [39], Pooled Gauss attack solves the LPN problem via finding a non-noisy index set, and such a non-noisy index set is called an information set in coding theory language. Therefore, we can view Pooled Gauss as a special case of the information set decoding (ISD) algorithm, particularly Pooled Gauss algorithm resembles the well-known algorithm of Prange [67].

Information set decoding (ISD). As shown in the previous subsection, solving LPN is equivalent to solving its dual variant, which is able to be interpreted as the task of decoding a linear code from its syndrome. The best-known algorithms for this task are improvements of the Prange's ISD algorithm [67], which aims to find a size-t subset of the rows of the parity-check matrix $\mathbf{H}$ that spans $\mathbf{H} \cdot \boldsymbol{e}$, where recall that t is the Hamming weight of noise vector $\boldsymbol{e}$.

For solving dual-LPN over binary field $\mathbb{F}_2$, the Prange's ISD algorithm [67] was gradually improved in recnt years. The improved ISD algorithm [72, 36] is called the Stern-Dumer variant (SD-ISD) in [48]. Later, the May-Meurer-Thomae variant (MMT-ISD) [59] and the Becker-Joux-May-Meurer variant (BJMM-ISD) [13] improved the SD-ISD by using the generalized birthday algorithm [74]. We give an overview of the three ISD variants in Appendix D. Recently, several works [7, 37, 41] reduced the significant space consumption of the MMT-ISD and BJMM-ISD algorithms.

Compared to the case of $\mathbb{F}_2$, the ISD algorithms to solve dual-LPN over larger fields are less studied. An initial study of ISD over a field $\mathbb{F}$ with $|\mathbb{F}| > 2$ was given by Coffey and Goodman [25], who provided an asymptotic analysis of the Prange's ISD algorithm over $\mathbb{F}$. Peters [66] generalized the more efficient SD-ISD algorithm from binary field $\mathbb{F}_2$ to any finite field $\mathbb{F}$. Later, Meurer [61] proposed a new generalization of the SD-ISD algorithm over any finite field.

These NN-ISD [60, 16, 40, 51, 47] applied "nearest neighbors" search to the SD-ISD or BJMM-ISD algorithms, and obtain better asymptotic complexities, where the works [60, 16, 40] focus on the case of binary field $\mathbb{F}_2$, and the works [51, 47] allow an arbitrary finite field $\mathbb{F}$. However, these NN-ISD introduce quite large polynomial overheads.

Statistical decoding. While Gauss and ISD attacks recover the secret vector $\boldsymbol{s}$, statistical decoding attack [3, 65, 44, 30] (a.k.a., low-weight parity-check attack in [17, 19, 20]) can directly distinguishes LPN samples $(\mathbf{A}, \boldsymbol{b} = \mathbf{A} \cdot \boldsymbol{s} + \boldsymbol{e})$ over a finite field $\mathbb{F}$ from random samples $(\mathbf{A}, \boldsymbol{u})$. The core of this attack is to find a set of parity-check vectors

$$\mathcal{V} \subset \{\boldsymbol{v} \,|\, \boldsymbol{v} \cdot \mathbf{A} = \mathbf{0} \text{ and } |\boldsymbol{v}| = w \text{ with a sufficiently small } w > 0\}\,.$$

While $\Pr[\langle \boldsymbol{v}, \boldsymbol{u} \rangle = 0] = 1/|\mathbb{F}|$ for random samples, $\Pr[\langle \boldsymbol{v}, \boldsymbol{b} \rangle = \langle \boldsymbol{v}, \boldsymbol{e} \rangle = 0] = 1/|\mathbb{F}| + \epsilon$ with some $\epsilon > 0$ for LPN samples, since both Hamming weights of vectors $\boldsymbol{e}$ and $\boldsymbol{v}$ are small. For each $\boldsymbol{v} \in \mathcal{V}$, this attack can compute a vote $\langle \boldsymbol{v}, \boldsymbol{y} \rangle$ with either $\boldsymbol{y} = \boldsymbol{b}$ or $\boldsymbol{y} = \boldsymbol{u}$. By repeating the process $|\mathcal{V}| = 1/\epsilon^2$ times, this attack outputs a majority of votes indicating whether $\boldsymbol{y} = \boldsymbol{b}$ or $\boldsymbol{y} = \boldsymbol{u}$.

In terms of asymptotic costs, Debris-Alazard and Tillich [30] show that an optimal statistical decoding algorithm requires more cost than the Prange's ISD algorithm [67] for the GV bound decoding over $\mathbb{F}_2$.

Recently, Carrier, Debris-Alazard, Meyer-Hilfiger and Tillich [24] tweaked the framework of statistical decoding and beat the above bound by introducing the fast Fourier transform. Furthermore, they [24] proposed an advanced attack, called the SD 2.0 algorithm, which improved statistical decoding considerably and even outperforms the ISD algorithm for the GV bound decoding over $\mathbb{F}_2$ and $k < 0.3N$. The SD 2.0 algorithm [24] invokes two techniques to reduce the time complexity, as follows:

1. generalize the majority voting to the fast Fourier transform, leading to a smaller ϵ.
2. use the same collision technique invoked by the ISD algorithms [72, 36, 59, 13] to find set $\mathcal{V}$ with a smaller w.

However, the running time of the second technology is $|\mathbb{F}|^{\theta(k)}$, which limits its advantage in analyzing the $(\mathsf{HW}, \mathbb{F})$-$\mathsf{LPN}(k, N, t)$ problem of security $|\mathbb{F}|^{\theta(k)}$. Therefore, the low-noise LPN assumption is beyond its strengths, because of the subexponential security $2^{O(k\mu)}$ of the low-noise LPN assumption, where dimension k and low noise rate $\mu = k^{-c}$ for constant $0 < c < 1$.

Our adapted SD 2.0 algorithm adjusts the SD 2.0 algorithm [24] for analyzing the low-noise LPN assumption by replacing the second technology with other collision techniques, e.g., [17]. However, this adapted SD algorithm still performs poorly on all the LPN parameters given in [17] (see Table 3 and Table 4 of Section 5). Moreover, in Section 5.3, we will prove that both an optimal statistical decoding attack and SD 2.0 attack (adapted to the low-noise setting) require more cost than the Prange's ISD algorithm [67] for analyzing the LPN assumptions over a large field, while allowing any weight $t = o(N)$ satisfied by the low-noise LPN assumption.

BKW. Blum, Kalai and Wassermann [15] introduced the first sub-exponential algorithm (referred to as the BKW algorithm) that solves LPN problems using a sub-exponential number of samples. In particular, the BKW algorithm recovers the secret of a $(\mathcal{D}, \mathbb{F}_2)$-$\mathsf{LPN}(k, N, t)$ instance with noise rate r in time $2^{O(k/\log(k/r))}$ using $2^{O(k/\log(k/r))}$ samples. Later, Lyubashevsky [58] reduced the sample complexity of the BKW algorithm to $k^{1+\epsilon}$ (for any constant $\epsilon > 0$) at the price of increasing the time complexity to $2^{O(k/\log\log(k/r))}$. The BKW algorithms have worse performance in time and the number of samples for solving low-noise LPN over larger fields. More recent works [38, 31, 57] focused on reducing the space complexity of the BKW algorithm. Esser, Kübler and May [39] also described the combinations and refinements of Pooled Gauss, ISD and BKW attacks, which are called Well-Pooled Gauss, Hybrid and Well-Pooled MMT attacks. All these attacks either require a sub-exponential number of samples that is *not* satisfied in the LPN-based protocols under the PCG framework, or take significantly more time than the previous three attacks for solving LPN with low noise rate. Thus, our analysis does not consider the BKW algorithms.

D Variants of Information Set Decoding

Following the analysis [43, 71, 48, 73], we summarize the ISD variants by Stern-Dumer [72, 36], May et al. [59] and Becker et al. [13]. We refer the reader to the corresponding papers and the surveys [43, 71, 48, 73] for a more detailed description.

D.1 Stern-Dumer Variant

The SD-ISD attack introduces two additional parameters p and ℓ, and adjusts both parameters to minimize the whole running time. Specifically, given an instance of the $(\mathsf{HW}, \mathbb{F}_2)$-dual-$\mathsf{LPN}(N, N-k, t)$ problem $(\mathbf{H} \in \mathbb{F}_2^{(N-k)\times N}, \boldsymbol{y} = \mathbf{H} \cdot \boldsymbol{e} \in \mathbb{F}_2^{(N-k)})$, the SD-ISD attack first transforms the instance to the following equation (1) via partial Gaussian elimination, where $\mathbf{U}$ is a non-singular $(N-k) \times (N-k)$ matrix and $\mathbf{P}$ is a random $N \times N$ permutation matrix.

$$\mathbf{U} \cdot \mathbf{H} \cdot \mathbf{P} = \begin{array}{c} \\ \\ \ell \end{array} \begin{array}{c} \overset{N-k-\ell}{\begin{array}{|c|}\hline \begin{array}{c} 1 \\ \ddots \\ 1 \end{array} \\ \hline \mathbf{0} \\ \hline \end{array}} \quad \overset{k+\ell}{\begin{array}{|c|}\hline \mathbf{R}_0 \\ \hline \mathbf{R}_1 \\ \hline \end{array}} \end{array}, \quad \mathbf{U} \cdot \boldsymbol{y} = \begin{array}{|c|}\hline \boldsymbol{y}_0 \\ \hline \boldsymbol{y}_1 \\ \hline \end{array} \tag{1}$$

Then, this attack finds $\boldsymbol{e}_1 \in \mathbb{F}_2^{(k+\ell)}$ such that $\mathbf{R}_1 \cdot \boldsymbol{e}_1 = \boldsymbol{y}_1$ and $|\boldsymbol{e}_1| \le p$ via the following meet-in-the-middle attack:

1. For each $\boldsymbol{e}_{1,0} \in \mathbb{F}_2^{(k+\ell)/2}$ with $|\boldsymbol{e}_{1,0}| \le p/2$, add the vector $\mathbf{R}_1 \cdot \left[\begin{array}{c|c} \boldsymbol{e}_{1,0} & \mathbf{0} \end{array}\right]$ into a sorted set $\mathcal{S}_0$.
2. For each $\boldsymbol{e}_{1,1} \in \mathbb{F}_2^{(k+\ell)/2}$ with $|\boldsymbol{e}_{1,1}| \le p/2$, add the vector $\boldsymbol{y}_1 - \mathbf{R}_1 \cdot \left[\begin{array}{c|c} \mathbf{0} & \boldsymbol{e}_{1,1} \end{array}\right]$ into a sorted set $\mathcal{S}_1$.
3. Search for identical elements in sets $\mathcal{S}_0$ and $\mathcal{S}_1$, and then add the corresponding vectors $(\boldsymbol{e}_{1,0} + \boldsymbol{e}_{1,1})$ into a set $\mathcal{S} \subseteq \{\boldsymbol{e}_1 \,|\, \mathbf{R}_1 \cdot \boldsymbol{e}_1 = \boldsymbol{y}_1\}$.

This attack repeats the above steps until $|\boldsymbol{e}_0| + |\boldsymbol{e}_1| \le t$ where $\boldsymbol{e}_0 = \mathbf{R}_0 \cdot \boldsymbol{e}_1 + \boldsymbol{y}_0$, and then outputs a noise vector $\boldsymbol{e} = \mathbf{P} \cdot (\boldsymbol{e}_0, \boldsymbol{e}_1)^\mathsf{T}$. The expected running time and more details of the SD-ISD variant are given in Appendix D.

Many variants [59, 13, 60, 16] improved the above step 3 of SD-ISD attack (finding candidate $\boldsymbol{e}_1$) via the generalized birthday algorithm [74], the representation technique [53] and the "Nearest Neighbours" search.

D.2 May-Meurer-Thomae Variant

The May-Meurer-Thomae variant (MMT-ISD) [59] replaced the birthday algorithm of Stern-Dumer variant [72, 36] with an order-2 generalized birthday algorithm [74]. This variant applied this and the representation technique [53] to improve ISD asymptotically and does as following,

1. For all $\boldsymbol{e}_{1,0} \in \mathbb{F}_2^{(k+\ell)/2}$ with $|\boldsymbol{e}_{1,0}| \le p/4$, store $\mathbf{R}_1 \cdot \left[\begin{array}{c|c} \mathbf{0} & \boldsymbol{e}_{1,0} \end{array}\right]$ in sorted $\mathcal{S}_1$ and $\mathcal{S}_3$ with $\mathcal{S}_1 = \mathcal{S}_3$.
2. For all $\boldsymbol{e}_{1,1} \in \mathbb{F}_2^{(k+\ell)/2}$ with $|\boldsymbol{e}_{1,1}| \le p/4$, store $\mathbf{R}_1 \cdot \left[\begin{array}{c|c} \boldsymbol{e}_{1,1} & \mathbf{0} \end{array}\right]$ in sorted $\mathcal{S}_2$ and store $\boldsymbol{y}_1 - \mathbf{R}_1 \cdot \left[\begin{array}{c|c} \boldsymbol{e}_{1,1} & \mathbf{0} \end{array}\right]$ in sorted $\mathcal{S}_4$.
3. Search from $\mathcal{S}_1$ to $\mathcal{S}_4$ and generate $\mathcal{S} \subseteq \{(\boldsymbol{e}_1 \,|\, \mathbf{R}_1 \cdot \boldsymbol{e}_1 = \boldsymbol{y}_1\}$ by an order-2 generalized birthday algorithm [74].

Note that the set $\mathcal{C}$ is a singleton in the original algorithm. By allowing several $\boldsymbol{c} \in \mathcal{C}$ we allow larger values of r_1 and give more flexibility in the search for optimal parameters. A larger r_1 also allows smaller memory requirements with the same algorithmic complexity.

D.3 Becker-Joux-May-Meurer Variant

The Becker-Joux-May-Meurer variant (BJMM-ISD) [13] further applied an order 3 generalized birthday algorithm [74] and does as following,

1. For all $\boldsymbol{e}_{1,0} \in \mathbb{F}_2^{(k+\ell)/2}$ with $|\boldsymbol{e}_{1,0}| \leq p_2/2$, store $\mathbf{R}_1 \cdot \left[\begin{array}{c|c} \mathbf{0} & \boldsymbol{e}_{1,0} \end{array}\right]$ in sorted $\mathcal{S}_1$, $\mathcal{S}_3$, $\mathcal{S}_5$ and $\mathcal{S}_7$ with $\mathcal{S}_1 = \mathcal{S}_3 = \mathcal{S}_5 = \mathcal{S}_7$.
2. For all $\boldsymbol{e}_{1,1} \in \mathbb{F}_2^{(k+\ell)/2}$ with $|\boldsymbol{e}_{1,1}| \leq p_2/2$, store $\mathbf{R}_1 \cdot \left[\begin{array}{c|c} \boldsymbol{e}_{1,1} & \mathbf{0} \end{array}\right]$ in sorted $\mathcal{S}_2$, $\mathcal{S}_4$ and $\mathcal{S}_6$ with $\mathcal{S}_2 = \mathcal{S}_4 = \mathcal{S}_6$ and store $\boldsymbol{y}_1 - \mathbf{R}_1 \cdot \left[\begin{array}{c|c} \boldsymbol{e}_{1,1} & \mathbf{0} \end{array}\right]$ in sorted $\mathcal{S}_8$.
3. Search from $\mathcal{S}_1$ to $\mathcal{S}_8$ and generate generate $\mathcal{S} \subseteq \{(\boldsymbol{e}_1 \,|\, \mathbf{R}_1 \cdot \boldsymbol{e}_1 = \mathbf{0}\}$ by an order-3 generalized birthday algorithm [74], where $p_2 = O(p)$ is positive additional parameters.

D.4 Cost of ISD variants

We essentially follow and simplify the analysis done in [43, 71, 48, 73] and count complexity by the number of field operations. The expected run-time of ISD attack consists of the below parts.

1. We denote T_{Gauss} as the cost of the partial Gaussian elimination. A naive implementation leads to $T_{Gauss} = (N-k-\ell)N^2$ field operation. Fast linear algebra [8] leads to $T_{Gauss} = (N-k-\ell)N^2/\log(N-k-\ell)$.
2. We estimate the success probability of one iteration. It is common in existing literature [71] that each individual $\boldsymbol{e}_1$ leads independently to success with the probability

$$\varepsilon(p,\ell)2^\ell \approx \binom{N-k-\ell}{t-p} 2^\ell \Big/ \binom{N}{t}.$$

   It follows that the probability of success of one iteration is equal to

$$\mathcal{P}(p,\ell) \approx \varepsilon(p,\ell)2^\ell |\mathcal{S}|$$

   The expected value of the set $\mathcal{S}$ will depend on various birthday_decoding.
3. Complexity of various birthday_decoding.
4. The final test cost $2|\mathcal{S}|N$ field operation.

Theorem 14 (SD-ISD [48, 72, 36])**.** *The* $(\mathsf{HW}, \mathbb{F}_2)$-$\mathsf{LPN}(k,N,t)$ *problem can be solved by the SD-ISD variant in expected time*

$$T_{SD}(N,k,t) = \min_{p,\ell} \frac{1}{\mathcal{P}(p,\ell)} \Big(T_{Gauss} + 2L_0 \cdot N + 2\mathbb{E}\big[|\mathcal{S}|\big] \cdot N \Big),$$

where $L_0 = |\mathcal{S}_1| = |\mathcal{S}_2| = \binom{(k+\ell)/2}{p/2}$ *and* $\mathbb{E}\big[|\mathcal{S}|\big] = \frac{L_0^2}{2^\ell}$.

Theorem 15 (MMT-ISD variant [48, 59])**.** *the* $(\mathsf{HW}, \mathbb{F}_2)$-$\mathsf{LPN}(k,N,t)$ *problem can be solved in expected time* $T_{MMT}(N,k,t)$ *by the MMT-ISD variant as below*

$$T_{MMT}(N,k,t) = \min_{\ell,r_1,p,|\mathcal{C}|} \frac{1}{\mathcal{P}(p,\ell)} \Big(T_{Gauss} + |\mathcal{C}| \cdot N \cdot \Big(4L_0 + \frac{2L_0^2}{2^{r_1}} + \frac{2L_0^4}{2^{\ell+r_1}} \Big) \Big),$$

where $L_0 = \binom{(k+\ell)/2}{p/4}$ *and* $|\mathcal{S}| = \frac{|\mathcal{C}|L_0^4}{2^{\ell+r_1}}$.

Theorem 16 (BJMM-ISD variant [48, 13]). *The* $(\mathsf{HW}, \mathbb{F}_2)$-$\mathsf{LPN}(k, N, t)$ *problem can be solved in expected time* $T_{BJMM}(N, k, t)$ *by the BJMM-ISD variant as below*

$$T_{BJMM}(N,k,t) = \min_{p,\ell,r_1,r_2,e_1,e_2} \frac{1}{\mathcal{P}(p,\ell)} \Big(T_{Gauss} + (8S_3 + 4C_3 + 2C_2 + 2C_1) \cdot N \Big) ,$$

where $S_3 = \binom{(k+\ell)/2}{p_2}$, $C_3 = \frac{S_3^2}{2^{r_2}}$, $C_2 = \frac{C_3^2}{2^{r_1}}$, $C_1 = \frac{S_1^2}{2^{\ell - r_1 - r_2}}$, $S_1 = \min\{\mu_2 C_2, \frac{\binom{k+\ell}{p_1}}{2^{r_1+r_2}}\}$, $|\mathcal{S}| = \min\{\mu_1 C_1, \frac{\binom{k+\ell}{p}}{2^\ell}\}$, $\mu_1 = \frac{\binom{p_1}{e_1}\binom{k+\ell-p_1}{p_1-e_1}}{\binom{k+\ell}{p_1}}$, $\mu_2 = \frac{\binom{p_2}{e_2}\binom{k+\ell-p_2}{p_2-e_2}}{\binom{k+\ell}{p_2}}$, $p_2 = p_1/2 + e_2$ *and* $p_1 = p/2 + e_1$.

E Proofs of Theorem 10 and Theorem 11

Theorem 17 (Theorem 10, restated). *For* $w = w(s) \in \mathbb{N}$ *and a finite field* $\mathbb{F}$ *with size* $|\mathbb{F}| \geq 4t$*, the adapted SD* 2.0 *algorithm solves the* $(\mathsf{HW}, \mathbb{F})$-$\mathsf{LPN}(k, N, t)$ *problem in time*

$$T = \min_s \left(T_1 \cdot \left(\frac{\binom{N}{t}}{\binom{N-w}{t}} \cdot \frac{2|\mathbb{F}|}{|\mathbb{F}|-1} \right)^2 + s \log\big(|\mathbb{F}|\big) |\mathbb{F}|^s \right) ,$$

where T_1 *is the time of finding one parity check vector.*

Proof. To simplify the analysis, we replace the exact noise distribution $\mathsf{HW}_{t,N}(\mathbb{F})$ with a slightly different $\mathsf{HW}'_{t,N}(\mathbb{F})$ as: 1) sample t out of N positions uniformly at random (and set the rest $N-t$ positions to 0); 2) for each selected position, use a random element in $\mathbb{F}$. Using the modified noise distribution $\mathsf{HW}'_{t,N}(\mathbb{F})$, we analyze the cost of statistical decoding attack to solve the decisional $(\mathsf{HW}', \mathbb{F})$-$\mathsf{LPN}(k, N, t)$ problem by extending the analysis approach for binary field $\mathbb{F}_2$ [30], where $\mathsf{HW}'(\mathbb{F}) = \{\mathsf{HW}'_{t,N}(\mathbb{F})\}_{t,N}$ is the corresponding family of distributions. This attack can obtain a distinguishing advantage at least $\frac{1}{2}$ in time T. Then, we show that the advantage is reduced to $\frac{1}{4}$ in the same time, when the noise distribution is converted from $\mathsf{HW}'_{t,N}(\mathbb{F})$ to $\mathsf{HW}_{t,N}(\mathbb{F})$.

Consider $\boldsymbol{b} = \mathbf{A} \cdot \boldsymbol{s} + \boldsymbol{e}'$ with $\boldsymbol{e}' \leftarrow \mathsf{HW}'_{t,N}(\mathbb{F})$, we denote by E the event that there is no intersection between the non-zero positions of vector $\boldsymbol{v}' = \begin{bmatrix} \mathbf{0} & \boldsymbol{v}_2 \end{bmatrix}$ and the noisy positions of $\boldsymbol{e}'$. Note that $\langle \boldsymbol{v}', \boldsymbol{e}' \rangle = \langle \boldsymbol{v}_2, \boldsymbol{e}'_2 \rangle$, where $\boldsymbol{e}' \stackrel{\text{def}}{=} \begin{bmatrix} \boldsymbol{e}'_1 & \boldsymbol{e}'_2 \end{bmatrix}$ with $\boldsymbol{e}'_2 \in \mathbb{F}^{N-s}$. Then, we have the following:

$$\begin{aligned} \Pr[\langle \boldsymbol{v}', \boldsymbol{b} \rangle = \langle \boldsymbol{v}', \boldsymbol{e}' \rangle = 0] &= \Pr[\langle \boldsymbol{v}', \boldsymbol{e}' \rangle = 0 \,|\, E] \cdot \Pr[E] + \Pr[\langle \boldsymbol{v}', \boldsymbol{e}' \rangle = 0 \,|\, \neg E] \cdot \Pr[\neg E] \\ &\leq \Pr[E] + \frac{1}{|\mathbb{F}|} \cdot \Pr[\neg E] = \frac{\binom{N-w}{t}}{\binom{N}{t}} + \frac{1}{|\mathbb{F}|} \cdot \left(1 - \frac{\binom{N-w}{t}}{\binom{N}{t}} \right). \end{aligned}$$

Therefore, our adapted SD 2.0 algorithm solves the $(\mathsf{HW}', \mathbb{F})$-$\mathsf{LPN}(k, N, t)$ in time

$$T = \min_s \; T_1 \cdot \left(\frac{\binom{N}{t}}{\binom{N-w}{t}} \cdot \frac{2|\mathbb{F}|}{|\mathbb{F}|-1} \right)^2 + s \log\big(|\mathbb{F}|\big) |\mathbb{F}|^s ,$$

We consider the statistical distance between $\boldsymbol{e} \leftarrow \mathsf{HW}_{t,N}(\mathbb{F})$ and $\boldsymbol{e}' \leftarrow \mathsf{HW}'_{t,N}(\mathbb{F})$

$$\mathsf{SD}(\boldsymbol{e}, \boldsymbol{e}') \leq \mathsf{SD}(\mathcal{U}_{(\mathbb{F}\setminus\{0\})^t}, \mathcal{U}_{\mathbb{F}^t}) = 1 - (1 - \frac{1}{|\mathbb{F}|})^t \leq \frac{t}{|\mathbb{F}|} \leq \frac{1}{4} ,$$

where the first inequality is because $\mathsf{HW}'_{t,N}(\mathbb{F})$ shares the same first-step sampling procedure as $\mathsf{HW}_{t,N}(\mathbb{F})$ ($\mathcal{U}_{\mathcal{R}}$ denotes a uniform distribution over $\mathcal{R}$), and the second equality is due to that for $\boldsymbol{b}' \leftarrow \mathcal{U}_{\mathbb{F}^t}$, conditioned on $\boldsymbol{b}' \in (\mathbb{F}\setminus\{0\})^t$, which has probability $(1-1/|\mathbb{F}|)^t$, $\boldsymbol{b}'$ follows distribution $\mathcal{U}_{(\mathbb{F}\setminus\{0\})^t}$. Therefore, taken into account the statistical difference between $\boldsymbol{e}$ and $\boldsymbol{e}'$, both our adapted SD 2.0 algorithm and the traditional SD attack solve the $(\mathsf{HW}, \mathbb{F})$-$\mathsf{LPN}(k, N, t)$ with constant probability. □

Theorem 18 (Theorem 11, restated)**.** *For any* $(\mathsf{HW}, \mathbb{F})$-$\mathsf{LPN}(k, N, t)$ *problem with* $|\mathbb{F}| = k^{\omega(1)}$, $(1+\beta)k \leq N = \mathsf{poly}(k)$ *for a constant* $\beta > 0$ *and* $t = o(N)$*, both the traditional SD attack and our adapted SD* 2.0 *attack (that distinguishes with constant advantage) require more cost than the Prange's ISD algorithm (that recovers the secret with constant probability).*

Proof. As we analyze above, the traditional SD attack takes no less cost than the adapted SD 2.0. Then, It's enough to prove that SD 2.0 attack (adapted to the low-noise setting) cannot be more effective than the Prange's ISD algorithm. Note that Prange's ISD algorithm recovers the secret of the $(\mathsf{HW}, \mathbb{F})$-$\mathsf{LPN}(k, N, t)$ problem in time $2^{o(k)}$ and constant probability. We only consider $s = o(k)$ in the SD 2.0 framework. If $s = \Omega(k)$, then the statement is true, because the SD 2.0 algorithm takes at least $|\mathbb{F}|^s$ to solve the $(\mathsf{HW}, \mathbb{F})$-$\mathsf{LPN}(k, N, t)$ problem.

The SD 2.0 framework needs to find parity-check vectors from

$$\mathcal{V} \subset \left\{ \boldsymbol{v} \stackrel{\text{def}}{=} \begin{bmatrix} \boldsymbol{v}_1 & \boldsymbol{v}_2 \end{bmatrix} \mid \boldsymbol{v}_1 \in \mathbb{F}^s, \boldsymbol{v} \cdot \mathbf{A} = \mathbf{0} \text{ and } |\boldsymbol{v}_2| = w \right\},$$

with a sufficiently small $w > 0$. The expected number of parameter-w parity-check vectors is $\mathbb{E}[|\mathcal{V}|] \leq \binom{N-s}{w}|\mathbb{F}|^{w-k+s}$. We need $\mathbb{E}[|\mathcal{V}|] \geq 1$ to guarantee existence. In particular, we have the following:

$$2^{w \cdot \log N + (w-k+s)\cdot \log |\mathbb{F}|} = N^w \cdot |\mathbb{F}|^{w-k+s} \geq \binom{N-s}{w} \cdot |\mathbb{F}|^{w-k+s} \geq 1.$$

From the above relation, $|\mathbb{F}| = k^{\omega(1)}$ and $N = \mathsf{poly}(k)$, we have that $w \geq (1 - o(1))k$. Since the cost of SD 2.0 increases as w increases, we set $w = (1 - o(1))k$ for an optimal attack in the SD 2.0 framework and we conservatively assume that the time of finding a parity-check vector with minimal weight w is at least the time using the Gaussian elimination method (denoted by T_{Gauss}). [12] According to Theorem 10, we have that the SD 2.0 solves the LPN problem with constant probability in time

$$T = \min_{s=o(k)} T_{Gauss} \cdot \left(\frac{\binom{N}{t}}{\binom{N-w}{t}} \cdot \frac{2|\mathbb{F}|}{|\mathbb{F}| - 1} \right)^2 + s \log\left(|\mathbb{F}|\right)|\mathbb{F}|^s \,.$$

The cost of the Prange's ISD algorithm is upper-bounded by $T_{Gauss} \cdot \frac{\binom{N}{t}}{\binom{N-k}{t}}$ [17, 49]. From $t = o(N)$, we have the following:

$$\frac{\left(\frac{\binom{N}{t}}{\binom{N-w}{t}}\right)^2}{\frac{\binom{N}{t}}{\binom{N-k}{t}}} = \frac{\binom{N}{t} \cdot \binom{N-k}{t}}{\binom{N-w}{t}^2} = \left(\frac{(N - o(N)) \cdot (N - k - o(N))}{(N - w - o(N))^2}\right)^t$$
$$= \left(\frac{(N - o(N))^2 - k \cdot (N - o(N))}{(N - o(N))^2 - 2w \cdot (N - o(N)) + w^2}\right)^t.$$

From $N \geq (1 + \beta)k$, we have that $2w \cdot (N - o(N)) - k \cdot (N - o(N)) = (2w - k) \cdot (N - o(N)) \geq (1 + \beta - o(1)) \cdot k^2 \geq k^2 \geq w^2$. Then, we have that the value in the above equation is at least 1. Therefore, the cost-ratio between the optimal attack in the SD 2.0 framework and the Prange's ISD algorithm is at least 4. □

[12] If there exist more efficient algorithms to find such a parity-check vector, then they can also be used to improve the ISD algorithm.