Reassessing the Security of LPN-C and its HHE-Oriented Variants

Orr Dunkelman[1,2], Semira Einsele[3], Hans Heum[4,5], Morten Øygarden[5,6], and Gerhard Wunder[3]

[1] University of Haifa, Israel.
[2] Technische Universität Berlin, Germany.
orrd@cs.haifa.ac.il
[3] Freie Universität Berlin, Germany.
{semira.einsele,g.wunder}@fu-berlin.de
[4] Norwegian University of Science and Technology, Trondheim, Norway.
hansh@simula.no
[5] Simula UiB, Bergen, Norway.
morten.oygarden@simula.no
[6] University of Bergen, Norway

Abstract. The idea of Hybrid Homomorphic Encryption (HHE) is to reduce the computational cost of Fully Homomorphic Encryption (FHE) by encrypting bulk data symmetrically while only encrypting the short symmetric key homomorphically. Its efficiency depends on the multiplicative depth of the symmetric cipher's decryption circuit, motivating FHE-friendly designs.

The Learning Parity with Noise (LPN) problem is a natural candidate for such designs, as it gives rise to simple encryption and decryption circuits over binary fields. In this context, Fouque, Hadjibeyli, and Kirchner proposed LPN-based symmetric encryption schemes based on the LPN-C cryptosystem of Gilbert et al. LPN-C is attractive for HHE while allowing parameter choices that bound decryption failures. However, the concrete security of LPN-C and its HHE-oriented variants remains poorly understood.

We quantify how enforcing bounded noise via rejection sampling reduces the observed noise rate, an effect not captured in prior analyses. This yields immediate speedups for all attacks based on LPN instance solving.

We then extend the Arora-Ge-style algebraic attacks to the bounded-noise setting and derive new bounds on the dimension of the induced linear spaces, refining and partially correcting earlier analyses. We show that some parameter regimes are more robust than previously estimated, while new algebraic strategies yield the best known attacks in others.

Overall, our results improve our understanding of the concrete security of LPN-based symmetric encryption schemes, informing parameter selection for FHE-friendly variants.

Keywords: Cryptanalysis, Algebraic Attacks, LPN-C, FHE-friendly

1 Introduction

The Learning Parity with Noise (LPN) problem is a central hardness assumption in cryptography. It sits at the heart of many constructions, from symmetric and asymmetric encryption schemes, to digital signatures and authentication protocols, and is even considered a promising post-quantum candidate problem. At its core, the LPN problem is the task of solving a noisy set of linear equations. This makes implementations of such schemes quite light-weight, as they only rely on addition and multiplication over a binary field.

A recent line of research has focused on *FHE-friendly symmetric primitives.* Fully Homomorphic Encryption (FHE) [27,51] enables arbitrary computation on encrypted data without decryption. However, practical FHE constructions remain elusive, as most schemes suffer from large ciphertext expansion, noise growth, and expensive homomorphic multiplications. Hybrid homomorphic encryption (HHE) [47] has been proposed as a partial solution to these issues: bulk data is encrypted with a symmetric cipher (with small ciphertext expansion) and sent to a server along with a homomorphically encrypted symmetric key. The server can now apply this key to homomorphically decrypt the symmetric ciphertexts, thus turning them into homomorphic ciphertexts, and perform computations on this data. This reduces bandwidth requirements at the expense of increased computational costs on the server side.

The main bottleneck in such schemes is the multiplicative depth of the symmetric cipher's decryption circuit, motivating the design of so-called "FHE-friendly" ciphers exhibiting low multiplicative depth. Notable examples include the block cipher LowMC [3], the stream cipher Kreyvium [12], FLIP [46] and FiLIP [45], Elisabeth [14], as well as the Rasta cipher and its descendants [17,18,33]. All of these constructions have already seen significant cryptanalytic effort.

It was in this context that Fouque, Hadjibeyli, and Kirchner [23], hereafter FHK, introduced variants of the LPN-Cryptosystem (LPN-C), as first introduced by Gilbert et al. [29]. LPN-C concatenates parallel LPN instances, so that the secret vector becomes a secret matrix, each column of which corresponds to a single instance. In this scheme, the message is encoded using an error-correcting code, and provided the sampled error (now an error *vector*) is small enough, the plaintext can be recovered by a user knowing the secret key by decoding the decrypted codeword.

Of course, with the errors following a standard Bernoulli distribution (biased coin flip), it is possible for the error to be large enough that the codeword no longer corresponds to a unique plaintext, leading to a decryption error. One way to mitigate this that was suggested by the authors of LPN-C is to reject sampled error vectors whenever their Hamming weight exceeds a certain threshold, taking a small hit in efficiency for potentially much lower chance of decryption errors. FHK observed that this modification could make the scheme susceptible to the Arora-Ge attack [4]. As a remedy, they introduced variants of LPN-C that instead employ *delinearization steps*, which are essentially degree-raising transformations that keep the error small, while resisting the attack vectors of Arora-Ge.

Table 1: Time complexities (in $\log_2$) for recovering the first secret column when replacing the noise rate τ by the effective noise rate τ_{eff}. ISD denotes best Information Set Decoding estimate and the rightmost column gives the best available classical (quantum) estimates from Tran and Vaudenay [54, Table 1].

Parameter Set	ISD ($\tau \to \tau_{\text{eff}}$)	Best classical (quantum) ($\tau \to \tau_{\text{eff}}$)
1	$123.07 \to 102.37$	$76.5 \to 71.4$ (66.5)
2	$123.07 \to 107.67$	$76.5 \to$ N/A
3	$83.37 \to 66.34$	74.6 (57.7) $\to$ N/A
4	$83.37 \to 70.36$	"
5	$83.37 \to 80.75$	"

The relevance of this line of work extends beyond FHE-friendly designs. In contrast to number-theoretic assumptions such as factoring or discrete logarithms, there are no known efficient quantum algorithms for solving the general LPN problem. Thus, LPN-based symmetric primitives are natural candidates for lightweight post-quantum constructions. At the same time, recent results show that certain LPN-based protocols such as HB+ and HB# can be broken in the quantum query (Q2) model [13], making a careful concrete security analysis particularly important.

Despite years of interest from the cryptographic community, the *concrete* security of LPN-C and its variants remains insufficiently understood.

In this work, we provide a systematic cryptanalysis of the LPN-C encryption scheme and its FHE-friendly variants. Our contributions are twofold.

First, we observe that rejection sampling can be seen as significantly shifting the noise distribution, effectively reducing the noise rate observed by an attacker. We quantify this phenomenon in section 3 as the effective noise rate τ_{eff}, which will be strictly smaller than the design noise rate parameter τ for any non-trivial rejection threshold t. Using this effective noise rate in place of the original noise rate then immediately yields improved attack complexities for standard LPN solvers, which naturally perform better for lower noise parameters. For all three LPN-C parameter sets suggested by Gilbert et al. [29], this already yields new-best attacks, falling for parameter set 3 significantly below the original target of 80 bits of security (see Table 1). We then give an iterated attack, exploiting a cascading effect in the reduced noise rate when solving for subsequent columns, with the result that the attack complexity for recovering the full secret matrix is completely dominated by that of recovering its first column, as illustrated in Figure 1.

Second, we investigate algebraic attacks against LPN-C and its extensions: In section 4, we refine the analysis underlying the Arora-Ge-type attack and obtain tighter monomial-counting bounds than those implicit in earlier works. For the delinearized LPN-C variants, our results are twofold. Firstly, our improved analysis shows that suggested algebraic attacks perform worse than previous

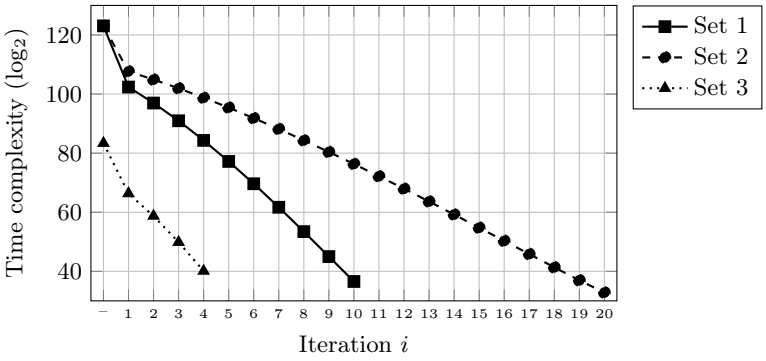

Fig. 1: Per-iteration ISD time complexity for iterative key recovery for the first three parameter sets. (See Table 4 for the exact values.)

estimates suggest, and that this construction enjoys stronger resistance to this attack vector. Secondly, we introduce a novel *dual* algebraic attack which exploits a different aspect of the secret key than what has previously been considered. This can lead to broken instances of the cipher unless properly accounted for.

Together, these results provide a more accurate and up-to-date security picture for the LPN-C family of symmetric ciphers, offering guidance towards updated LPN-C parameter sets, and informing possible future FHE-friendly constructions based on the LPN problem.

Existing cryptanalysis of LPN-C has followed two main directions so far: First, algebraic approaches, such as the Arora–Ge technique [4], which exploits the bounded Hamming weight of errors to construct polynomial equations in the secret. FHK [23] observed that the resampling strategy in LPN-C makes such attacks particularly relevant. Second, one can apply generic LPN algorithms by viewing LPN-C as a collection of column-wise LPN instances. BKW [9] were the first to give an algorithm that solves LPN in subexponential time, which Levieil and Fouque [42] subsequently refined. Guo et al. [32] employed covering codes to solve the $(512, 1/8)$ instance of LPN-C with complexity below 2^{80}, and Zhang et al. [56] further reduced complexities using cascaded perfect codes. These results suggest that the commonly used $(512, 1/8)$ parameters, which are currently used in HB authentication protocols and LPN-C alike, no longer provide adequate security.

2 Preliminaries

2.1 Notation

We take the natural numbers $\mathbb{N}$ to be the set $\{1, 2, \ldots\}$, and $\mathbb{F}_q$ denotes the finite field with q elements. For two vectors $\boldsymbol{a}, \boldsymbol{b}$, their inner product is denoted $\langle \boldsymbol{a}, \boldsymbol{b} \rangle$. List and vector indices start on 1, and sets of the form $\{x_i\}_{i=1}^n$ are taken to be multisets. Operations like the xor operation on vectors are defined element-wise:

$$\boldsymbol{a} \oplus \boldsymbol{b} = (a_1 \oplus b_1, \ldots, a_n \oplus b_n) \,.$$

Assignment is denoted $\leftarrow$, e.g. $a \leftarrow b$. We use $\xleftarrow{\$}$ to denote sampling from a distribution, e.g. $x \xleftarrow{\$} \mathsf{Dist}$. For $n \in \mathbb{N}$, we denote by $\boldsymbol{v} \xleftarrow{\$} \mathsf{Dist}^n$ the sampling of a vector $\boldsymbol{v}$ according to the product distribution Dist^n, i.e. the n vector elements $\boldsymbol{v}[i]$ are drawn independently and identically according to the distribution Dist. When sampling from a set $\mathcal{X}$, $x \xleftarrow{\$} \mathcal{X}$ denotes uniform sampling, and assigning the output of an algorithm Alg to a variable x is denoted $x \leftarrow \mathsf{Alg}$ resp. $x \xleftarrow{\$} \mathsf{Alg}$ for deterministic resp. randomized algorithms. Whenever an (oracle) algorithm Alg has access to oracle $\mathcal{O}$, this is written $\mathsf{Alg}^{\mathcal{O}}$.

We use $\Pr[\mathsf{Code} : \mathsf{Event} \mid \mathsf{Condition}]$ to denote the probability that running code Code leads to event Event, conditioned on $\mathsf{Condition}$. We exclude Code when it is clear from context and $\mathsf{Condition}$ when not relevant.

We will in particular be interested in the Bernoulli distribution, denoted Ber_τ, where the Bernoulli parameter $\tau \in [0, 1/2)$ is such that for bit $b \xleftarrow{\$} \mathrm{Ber}_\tau$, $\Pr[b = 1] = \tau$ and $\Pr[b = 0] = 1 - \tau$.

2.2 The LPN Problem

The *Learning Parity with Noise* (LPN) problem is a fundamental hard learning problem with widespread applications in cryptography [48]. In its search variant, the task is to recover an unknown secret $\boldsymbol{s}$ from Q noisy linear equations in $\boldsymbol{s}$.

Definition 1 (Search-LPN). *Let $k \in \mathbb{N}$ and $\tau \in [0, 1/2)$ be parameters, and let $\boldsymbol{s} \xleftarrow{\$} \mathbb{F}_2^k$. Then the $\mathrm{LPN}_{k,\tau}$ problem is: Given oracle access to the LPN sampling oracle $\mathcal{L}_{\boldsymbol{s}}$, which samples vector $\boldsymbol{a} \xleftarrow{\$} \mathbb{F}_2^k$ and error bit $\nu \xleftarrow{\$} \mathrm{Ber}_\tau$ and returns*

$$(\boldsymbol{a}, c) = (\boldsymbol{a}, \langle \boldsymbol{a}, \boldsymbol{s} \rangle \oplus \nu) \in \mathbb{F}_2^k \times \mathbb{F}_2 \,,$$

find $\boldsymbol{s}$.

Definition 2 (LPN Solver). *Let $\mathsf{Q}, \mathsf{T}, \mathsf{M} \in \mathbb{N}$ and $\varepsilon \in [0, 1]$, and let $\mathcal{L}_{\boldsymbol{s}}$ be an $\mathrm{LPN}_{k,\tau}$ oracle with secret $\boldsymbol{s} \in \mathbb{F}_2^k$. We say that an algorithm $\mathcal{A}$ $(\mathsf{Q}, \mathsf{T}, \mathsf{M}, \varepsilon)$-solves the $\mathrm{LPN}_{k,\tau}$ problem if it makes at most Q queries to $\mathcal{L}_{\boldsymbol{s}}$, uses time at most T and memory at most M, and outputs $\boldsymbol{s}$ with probability at least ε:*

$$\Pr[\boldsymbol{s}' \xleftarrow{\$} \mathcal{A}^{\mathcal{L}_{\boldsymbol{s}}} : \boldsymbol{s}' = \boldsymbol{s}] \geq \varepsilon \,.$$

Definition 3 (The Search-LPN Assumption). *The* $\mathrm{LPN}_{k,\tau}$ *problem is said to be* $(\mathsf{Q},\mathsf{T},\varepsilon)$*-hard if, for every* $\varepsilon' \in [0,1]$ *such that there exists an LPN solver* $\mathcal{A}$ *that* $(\mathsf{Q},\mathsf{T},\mathsf{M},\varepsilon')$*-solves* $\mathrm{LPN}_{k,\tau}$ *for some* $\mathsf{M} \leq \mathsf{T}$*, we have that* $\varepsilon' \leq \varepsilon$*.*

The search LPN problem can be viewed as an average-case decoding problem for random linear codes: given Q LPN samples, construct $A \in \mathbb{F}_2^{\mathsf{Q}\times k}$, and interpret $A^T \in \mathbb{F}_2^{k\times\mathsf{Q}}$ as the code generator matrix, the secret $\boldsymbol{s}$ as the message, and $\boldsymbol{c} = A\boldsymbol{s} \oplus \boldsymbol{\nu} \in \mathbb{F}_2^{\mathsf{Q}}$ as a noisy codeword.

2.3 LPN Solvers

Most state-of-the-art attacks on $\mathrm{LPN}_{k,\tau}$ follow the *reduction-plus-solver* paradigm: starting from samples $(A,\boldsymbol{b})$ with secret $\boldsymbol{s} \in \mathbb{F}_2^k$, a *reduction chain* combines and preprocesses equations to obtain a new LPN instance with a shorter effective secret $\boldsymbol{s}' \in \mathbb{F}_2^{k'}$. A *solving algorithm* then recovers $\boldsymbol{s}'$ (or sufficient information to reconstruct it), the result is substituted back, and the procedure is iterated until $\boldsymbol{s}$ is recovered. In the current state of the art, both reductions and solvers are predominantly *classical*, although hybrid/quantum variants exist [55].

The main reduction primitives are *drop-reduce* and *xor-reduce*: drop-reduce(b) keeps only samples with b trailing zeros (shrinking the dimension but losing many samples), whereas xor-reduce(b) buckets by the last b bits and XORs within buckets to cancel them, at the cost of squaring the bias (piling-up). More advanced reductions include *sparse-secret* and *code-reduce* (covering-code based bias boosting), and *c-sum/dissection*, which combines $c > 2$ samples for improved time–memory trade-offs.

Typical solver choices for the final phase include: (1) *linear-system based* methods (Gauss), which solve a linear system and validate them on extra samples; (2) *ISD-style* methods [50], which search for favorable information sets and small error patterns before decoding; (3) *correlation/Fourier* solvers, most notably the Walsh–Hadamard transform (WHT) [42], *significant Fourier transform* (SFT) techniques that aim to locate heavy Fourier coefficients without computing the full spectrum [1,16,26]; and (4) *BKW-style* solvers [9], which combine an iterative reduction (with noise growth) and a final recovery step (often WHT or enumeration) once k' is sufficiently small. We discuss these in greater detail in section A.

In our complexity analysis we consider the LPN-ISD algorithm of Tran and Vaudenay [55] (Alg. 5). Define $P = P(k,\tau,w_1,w_2) = \sum_{i=w_1}^{w_2} \binom{k}{i} \tau^i (1-\tau)^{k-i}$. Now, let T_{sparse} denote the complexity of the *sparse-secret* reduction (given by Tran and Vaudenay in Appendix A.1). Then, the (classical) time complexity is

$$T_{\mathrm{ISD}} = \frac{1}{P}\left(T_{\mathrm{sparse}} + k(\mathsf{Q}-k)\sum_{i=w_1}^{w_2}\binom{k}{i}\right). \tag{1}$$

2.4 LPN-Based Symmetric Encryption and Authentication Schemes

LPN-C. Gilbert, Robshaw, and Seurin introduced LPN-C [29], a symmetric encryption scheme based on the LPN assumption. The construction combines a public binary linear error-correcting code with a secret matrix, and adds random Bernoulli noise during encryption.

Definition 4 (LPN-C [29]). *Let $r, n, d \in \mathbb{N}$, and let $C : \mathbb{F}_2^r \to \mathbb{F}_2^n$ be a binary linear code with minimum distance d and a decoder $D : \mathbb{F}_2^n \to \mathbb{F}_2^r \cup \{\perp\}$, where $\perp$ denotes decoding failure. The symmetric encryption scheme* LPN-C *is then defined as follows:*

- ***Key Generation:*** *Output $S \xleftarrow{\$} \mathbb{F}_2^{k \times n}$.*
- ***Encryption:*** *On input message $\boldsymbol{x} \in \mathbb{F}_2^r$, sample $\boldsymbol{a} \xleftarrow{\$} \mathbb{F}_2^k$ and $\boldsymbol{\nu} \xleftarrow{\$} \mathrm{Ber}_\tau^n$, set*

$$\boldsymbol{y} \leftarrow C(\boldsymbol{x}) \oplus \boldsymbol{a}^T S \oplus \boldsymbol{\nu}\,,$$

  and output the ciphertext $(\boldsymbol{a}, \boldsymbol{y})$.
- ***Decryption:*** *On input $(\boldsymbol{a}, \boldsymbol{y}) \in \mathbb{F}_2^k \times \mathbb{F}_2^n$, output $D(\boldsymbol{y} \oplus \boldsymbol{a}^T S)$.*

Under the assumption that C is indistinguishable from a random linear code, the security of LPN-C can be reduced directly to the hardness of LPN [29].

LPN-C with Secret Toeplitz Matrix. Gilbert et al. [29] already discussed, and the HB# protocol [28] further promoted reducing the key size by replacing the uniformly random secret $S \in \mathbb{F}_2^{k \times n}$ with a Toeplitz matrix. Concretely, for a Toeplitz secret $S = (s_{ij})$ with $s_{ij} = t_{i-j}$, the key can be stored using only $k + n - 1$ bits instead of kn bits. While the structure substantially compresses the key size, its concrete security impact remains subtle and not fully understood. We discuss the impact of using a secret Toeplitz matrix in LPN-C on one of our attacks in Appendix D.1.

MLPN-C. MLPN-C generalizes LPN-C [29] to extension fields and was introduced by Fouque, Hadjibeyli, and Kirchner [24] specifically for FHE-friendly symmetric encryption. The aim is to design symmetric primitives whose decryption circuit has very small multiplicative depth, enabling efficient homomorphic evaluation. The scheme is defined analogously to LPN-C, with the secret key S now sampled from $\mathbb{F}_{2^\ell}^{k \times n}$ and ciphertext components interpreted accordingly. The resulting scheme is referred to as MLPN-C.

Table 2: Suggested LPN-C parameter sets targeting 80-bit security [29]

LPN-C Parameters	k	τ	n	r	d	$t = \lfloor \frac{d-1}{2} \rfloor$	$\Pr[\mathrm{HW}(\boldsymbol{\nu}) > t]$
Parameter set 1	512	1/8	80	27	21	10	0.42
Parameter set 2	512	1/8	160	42	42	20	0.44
Parameter set 3	768	1/20	80	53	9	4	0.37
Parameter set 4	768	1/20	160	99	17	8	0.41
Parameter set 5	768	1/20	160	75	25	12	0.06

Definition 5 (MLPN-C [24]). *Let $r, n, \ell \in \mathbb{N}$, and let $C : \mathbb{F}_{2^\ell}^r \to \mathbb{F}_{2^\ell}^n$ be a linear code with minimum distance d and a decoder $D : \mathbb{F}_{2^\ell}^n \to \mathbb{F}_{2^\ell}^r \cup \{\perp\}$, where $\perp$ denotes decoding failure. The symmetric encryption scheme* MLPN-C *is then defined as follows:*

- ***Key Generation:*** *Output $S \xleftarrow{\$} \mathbb{F}_{2^\ell}^{k \times n}$.*
- ***Encryption:*** *On input $\boldsymbol{x} \in \mathbb{F}_{2^\ell}^r$, sample $\boldsymbol{a} \xleftarrow{\$} \mathbb{F}_{2^\ell}^k$ and $\boldsymbol{\nu} \xleftarrow{\$} \mathrm{Ber}_\tau^{\ell \times n}$, set*

$$\boldsymbol{y} \leftarrow C(\boldsymbol{x}) \oplus \boldsymbol{a}^T S \oplus \boldsymbol{\nu}\,,$$

  (interpreting $\boldsymbol{\nu} \in \mathbb{F}_2^{\ell \times n}$ as an element of $\mathbb{F}_{2^\ell}^n$), and output ciphertext $(\boldsymbol{a}, \boldsymbol{y})$.
- ***Decryption:*** *On input $(\boldsymbol{a}, \boldsymbol{y}) \in \mathbb{F}_{2^\ell}^k \times \mathbb{F}_{2^\ell}^n$, output $D(\boldsymbol{y} \oplus \boldsymbol{a}^T S)$.*

For a message of r bits, MLPN-C yields a ciphertext of $n + k$ bits. Fouque et al. [24] propose using the 3-repetition code to realize a multiplication circuit of depth 2. For $a, b, c \in \mathbb{F}_{2^\ell}$ with encoding $(a, b, c) \in \mathbb{F}_{2^\ell}^3$, one computes $ab + bc + ac$ as the decoding. The additional factor $2^{\ell-1}$ can be folded into the encoding to improve efficiency in homomorphic encryption. Just like LPN-C, the security of MLPN-C can be reduced to the hardness of LPN under the assumption that the code C is indistinguishable from a random linear code [24].

Variants of LPN-C and MLPN-C: Handling Decryption Errors. The scheme LPN-C fails to decrypt whenever the error exceeds the decoding capacity of the underlying code C, namely whenever $\mathrm{HW}(\nu) > t = \left\lfloor \frac{d-1}{2} \right\rfloor$, the decoder outputs $\perp$. Table 2 lists the originally suggested parameter sets for LPN-C targeting 80-bit security, together with their associated decryption errors [29]. For the first four sets, more than 1 in 3 ciphertexts will fail to decrypt. Obviously, such high failure rates may impact the practicality of the scheme. Two main directions have been discussed in the literature to address this.

Variant 1 (Unrestricted Bernoulli noise). The first option is essentially to treat decryption failures as a necessary evil to live with. Here, the noise vector ν is sampled as in the original scheme, i.e., $\nu \xleftarrow{\$} \mathrm{Ber}_\tau^n$, without constraints on its Hamming weight. This preserves the reduction to the standard LPN problem and maintains the usual i.i.d. noise distribution. In practice, however, it may be difficult to hide that such an error has occurred based on behaviors/side-channels from the communicating parties. The noise parameter τ can be chosen such that decryption failures occur only with low probability, but this also enables more efficient attacks. To counter these attacks, Fouque et al. [24] implement a *delinearization step* to increase the algebraic degree of the ciphertext components, thereby raising attack complexity. We do not treat Variant 1 of LPN-C further in this work, but will return to its delinearization variant in section 5.

Variant 2 (Rejection sampling/bounded-weight noise). Here, the encryptor repeatedly samples ν until the condition $\mathrm{HW}(\nu) \leq t$. This ensures that there are no decryption failures, and thus the correctness of decryption. Variant 2 introduces two important consequences that are central to our cryptanalysis:

(i) *Statistical effect.* Conditioning on $\mathrm{HW}(\nu) \leq t$ changes the noise distribution of accepted ciphertexts and yields a strictly smaller *effective* noise rate. As we show in section 3, this directly improves standard LPN-solving algorithms.
(ii) *Algebraic effect.* Since every accepted ciphertext satisfies $\mathrm{HW}(\nu) \leq t$, any product of $t+1$ ciphertext components must contain at least one error-free term. In section 4, we show that these polynomial relations have additional structure specific to LPN-C, and we give tight bounds on the dimension of the resulting linear space.

2.5 On the HB Authentication Protocol Family

The idea of bounding the Hamming weight of the noise vector is not unique to LPN-C: Similar ideas have appeared in several earlier LPN-based constructions, such as in the HB family of LPN-based symmetric-key authentication protocols. This line of research started with the HB protocol [36], a simple and lightweight authentication protocol with a security reduction to standard LPN. Later variants, including HB+ [38] and HB# [28], aimed to further optimize security and efficiency. These schemes attracted much attention due to their simplicity, strong security guarantees, and lightweight implementation profile, making them suitable for constrained settings such as RFID tags.

Katz, Shin, and Smith [39] discuss an HB variant in which the prover ensures that the sampled noise vector has Hamming weight at most $\lceil \tau n \rceil$. This modification

ensures perfect completeness, since the tag never produces more errors than the verifier can tolerate. It also improves the concrete security bounds, as the parameter controlling acceptance can be set precisely to τn rather than analyzed using tail bounds of a Bernoulli distribution. However, as Katz et al. note,

it remains unclear how the hardness of the resulting fixed-weight LPN variant compares to the standard LPN_τ problem. This illustrates that bounding the Hamming weight of the noise vector is a recurring theme for LPN-based primitives, yet cryptanalytic implications remain uncertain.

In another direction, while the HB and HB+ protocols are vulnerable to meet in the middle attacks (which are considered in this model different than "active attack"), the HB# protocol resists those. This begs the question whether our results can be applied to it. In principle, the answer is yes, but the effect is negligible for the parameter ranges that have typically been discussed for HB#: the probability that a randomly sampled error vector would cause false rejection (the equivalent of a decryption error in LPN-C) is simply too small. Similarly, applying the results of section 4 seems to be infeasible, as the degree of the polynomial grows beyond 100 for the suggested parameters.

3 Analysis of LPN-C with Rejection Sampling

When LPN-C is implemented with rejection sampling, such that all error vectors satisfy $\mathrm{HW}(\boldsymbol{\nu}) \leq t$, the attacker effectively faces a smaller noise rate than the original rate τ. We refer to this reduced rate as the *effective noise rate* τ_{eff}, as given in Proposition 1. Bogos et al. [10] briefly note that bounding the Hamming weight of the noise vector may negatively affect the security. In this section, we quantify this effect. To our knowledge, this is the first work to explicitly turn this rejection-sampling-induced reduction in the effective noise rate into an LPN-C key-recovery attack. In general, an attacker can then recover the first column of the secret key more efficiently by interpreting it as an LPN instance with noise rate τ_{eff} in place of τ. Furthermore, having recovered the first column of the secret key (and thus the first element of every noise vector), the effective noise rate of the remaining noise elements is reduced further, enabling an iterated key-recovery attack with successively smaller noise rates, leading to significant speed-up compared to solving every column independently (see Table 4).

3.1 Reduction to LPN Instances with Smaller Noise Level

An LPN-C ciphertext will fail to decrypt whenever $\mathrm{HW}(\boldsymbol{\nu}) > t$, where $t = \lfloor \frac{d-1}{2} \rfloor$ is the decoding radius of the code C. Since elements of $\boldsymbol{\nu}$ are i.i.d., the failure probability is given by the following formula.

Definition 6. *Let $n, t \in \mathbb{N}$ and $\tau \in [0, 1/2)$, and let $\boldsymbol{\nu}$ be an error vector distributed according to Ber_τ^n. Then the probability that the Hamming weight is within the threshold $t < n$ is given by*

$$W_\tau(n,t) \coloneqq \Pr[\boldsymbol{\nu} \xleftarrow{\$} \mathrm{Ber}_\tau^n \,:\, \mathrm{HW}(\boldsymbol{\nu}) \leq t] = \sum_{i=0}^{t} \binom{n}{i} \tau^i (1-\tau)^{n-i}\,.$$

Let us now focus on recovering the first column of the secret matrix S by solving the corresponding LPN instance.

Proposition 1. *Let $k, n, t \in \mathbb{N}$ and $\tau \in (0, 1/2)$. Consider an* LPN-C *instance with parameters (k, τ, n) for which error vectors are guaranteed to have Hamming weight at most $t < n$. Then a passive attacker observing plaintext-ciphertext pairs can recover one column of the secret key by solving an $\mathrm{LPN}_{k,\tau_{\mathit{eff}}}$ instance, where $\tau_{\mathit{eff}} \in [0, 1/2)$ is given by*

$$\tau_{\mathit{eff}} = \tau_{\mathit{eff}}(\tau, n, t) \coloneqq \frac{\tau \cdot W_\tau(n-1, t-1)}{W_\tau(n, t)} < \tau\,,$$

Proof. An LPN-C attacker can transform a plaintext-ciphertext pair $(\boldsymbol{x}, (\boldsymbol{a}, \boldsymbol{y}))$ into n LPN samples $(\boldsymbol{a}, \boldsymbol{y}_i) = (\boldsymbol{a}, \langle \boldsymbol{a}, \boldsymbol{s}_i \rangle \oplus \nu_i)$ by computing $C(\boldsymbol{x}) \oplus \boldsymbol{y} = \boldsymbol{a}S \oplus \boldsymbol{\nu}$, and treating each row of the secret matrix $S[i]$ as an individual LPN secret $\boldsymbol{s}_i$.

For error vector $\boldsymbol{\nu} = (\nu_1, \ldots, \nu_n)$, let E_1 denote the event that $\nu_1 = 1$, and let Acc denote the event that $\boldsymbol{\nu}$ is accepted by the rejection sampling procedure, i.e. the event $\mathrm{HW}(\boldsymbol{\nu}) \leq t$. We want to calculate the probability that the first element of the vector contains an error, conditioned on the instance having been accepted, i.e. $\Pr[\mathsf{E}_1 \mid \mathsf{Acc}] \coloneqq \tau_{\mathrm{eff}}$.

We have from Definition 6 that

$$\Pr[\mathsf{Acc}] = W_\tau(n, t) = \sum_{i=0}^{t} \binom{n}{i} \tau^i (1-\tau)^{n-i}\,.$$

Next, note that if the first entry of $\boldsymbol{\nu}$ is 1, then the remaining $n-1$ elements can contain at most $t-1$ errors before the sample would have been rejected. Hence

$$\Pr[\mathsf{Acc} \mid \mathsf{E}_1] = W_\tau(n-1, t-1). \tag{2}$$

Meanwhile, the first entry of the error vector is 1 with probability τ by definition:

$$\Pr[\mathsf{E}_1 \cap \mathsf{Acc}] = \Pr[\mathsf{E}_1] \cdot \Pr[\mathsf{Acc} \mid \mathsf{E}_1] = \tau \cdot W_\tau(n-1, t-1)\,, \tag{3}$$

The desired probability is then given by

$$\tau_{\mathrm{eff}} = \Pr[\mathsf{E}_1 \mid \mathsf{Acc}] = \frac{\Pr[\mathsf{E}_1 \cap \mathsf{Acc}]}{\Pr[\mathsf{Acc}]} = \frac{\tau \cdot W_\tau(n-1, t-1)}{W_\tau(n, t)}\,. \tag{4}$$

Although the accepted error vector is not distributed as an i.i.d. Bernoulli vector with parameter τ_{eff}, each fixed coordinate has marginal error probability τ_{eff}. Since different ciphertexts are sampled independently, the errors observed in any fixed column are independent Bernoulli samples with this parameter.

To show that $\tau_{\mathrm{eff}} < \tau$ whenever $\tau > 0$, it suffices to observe that $W_\tau(n-1, t-1) < W_\tau(n, t)$ for all values of n, $t < n$, which can be verified directly from Definition 6. (Intuitively, increasing the threshold by 1 makes it less likely to be met, even when sampling an additional bit.) □

We calculate the effective error rate τ_{eff} for the LPN-C parameter sets 1–5 (Table 2); results are presented in Table 3. We observe a reduction in the effective

Table 3: Original and effective noise rates for the LPN-C parameter sets from Table 2. The relative change is computed as $\left(\frac{\tau_{\text{eff}}-\tau}{\tau}\right) \cdot 100\%$.

Set	$\tau \to \tau_{\text{eff}}$	Relative change
1	$0.125 \to 0.1000$	-20.0%
2	$0.125 \to 0.1064$	-14.88%
3	$0.05 \to 0.0349$	-30.2%
4	$0.05 \to 0.0385$	-22.96%
5	$0.05 \to 0.0477$	-4.6%

error rate for all five parameter sets, with the decrease being particularly significant for parameter sets 1–4. More generally, the relative decrease becomes more noticeable as the rejection threshold decreases.

The effect of the reduced noise is a significant speedup in recovering the first column of the secret matrix via LPN solvers, as illustrated in Table 1. For example, the estimated attack complexity of ISD against parameter set 1 drops by more than 20 bits, from $2^{123.07}$ to $2^{102.37}$. The same parameter set drops by 10 bits, from $2^{76.5}$ to $2^{66.5}$ when using the estimates for state-of-the-art LPN (quantum) solvers due to Tran and Vaudenay [54], well below the targeted 80 bits of security. (Specifically, for this parameter range, Tran and Vaudenay find the best-performing classical solver to be the SFT algorithm, and the best-performing quantum solver to be a quantum ISD algorithm.)

3.2 An Iterative Key-Recovery Attack

We next extend the attack strategy presented in subsection 3.1 by observing that the noise reduction effect compounds: Once a column S_i has been recovered, the corresponding error bits ν_i become known for all available samples. Provided access to enough samples, one can then restrict to only those samples with $\nu_i = 1$ (which, recall, must have passed rejection sampling), further reducing the effective noise rate for the remainder of the noise vector.

Iterating this procedure gives a sequence of LPN instances whose effective noise rate decreases with each solved column. As illustrated in Table 4 for ISD, this has the effect of reducing the complexity of full-key recovery from that of solving each column independently, $\mathsf{T} = n \cdot \mathsf{T}_1$, to $\mathsf{T} \approx \mathsf{T}_1$. (Here, T is the complexity of recovering the full matrix, and T_1 is that of recovering only the first column.) In other words, the complexity of recovering the full matrix will now be completely dominated by that of recovering the first column, justifying a focus on the cost of recovering just the first column when estimating security.

Table 4: Per-iteration ISD attack complexity $\log_2 \mathsf{T}_i$ given by Equation 1 for LPN-C parameter sets 1 and 2 from Table 2 (see Table 5 for sets 3–5). The first row of each set shows the complexity using the original noise rate τ; subsequent rows use the iterated effective noise rates from Proposition 2. Let T_0 denote the complexity of the ISD attack using noise rate τ. The bottom rows compare $\log_2(\sum_i \mathsf{T}_i)$ with the independent-column baseline $\log_2(n \cdot \mathsf{T}_0)$. All complexity values are given in $\log_2$.

Parameter Set 1			Parameter Set 2					
i	$\tau_{\text{eff}}^{(i)}$	T_i	i	$\tau_{\text{eff}}^{(i)}$	T_i	i	$\tau_{\text{eff}}^{(i)}$	T_i
–	0.125	123.07	–	0.125	123.07			
1	0.0999	102.37	1	0.1064	107.67	11	0.0617	72.17
2	0.0932	96.96	2	0.1030	104.89	12	0.0562	67.96
3	0.0857	90.96	3	0.0994	101.96	13	0.0505	63.64
4	0.0773	84.31	4	0.0954	98.73	14	0.0446	59.21
5	0.0682	77.19	5	0.0913	95.43	15	0.0386	54.76
6	0.0584	69.64	6	0.0868	91.83	16	0.0325	50.30
7	0.0479	61.68	7	0.0822	88.18	17	0.0263	45.84
8	0.0368	53.44	8	0.0773	84.31	18	0.0199	41.35
9	0.0251	44.99	9	0.0723	80.39	19	0.0134	36.95
10	0.0128	36.56	10	0.0671	76.34	20	0.0068	32.81
Total	$\sum_i \mathsf{T}_i = \mathbf{102.4}$		**Total**		$\sum_i \mathsf{T}_i = \mathbf{107.9}$			
Baseline	$n \cdot \mathsf{T}_0 = 129.4$		Baseline		$n \cdot \mathsf{T}_0 = 130.4$			

Proposition 2. *Let $k, n, t, i \in \mathbb{N}$ and $\tau \in (0, 1/2)$. Consider an instance of* LPN-C *with parameters (k, τ, n) for which error vectors are guaranteed to have Hamming weight at most $t < n$. Then a passive attacker observing plaintext-ciphertext pairs, having recovered the first $i-1$ columns of the secret key S, can recover the ith column by solving an $\mathrm{LPN}_{k,\tau_{\mathit{eff}}^{(i)}}$ instance, where the ith iterated effective noise rate $\tau_{\mathit{eff}}^{(i)} \in (0, 1/2)$ is given by*

$$\tau_{\mathit{eff}}^{(i)} = \tau_{\mathit{eff}}^{(i)}(\tau, n, t) = \frac{\tau\, W_\tau(n-i, t-i)}{W_\tau(n-(i-1), t-(i-1))}\,.$$

Further, for all $i \le t$, $\tau_{\mathit{eff}}^{(i+1)} < \tau_{\mathit{eff}}^{(i)}$, and for all $i > t$, $\tau_{\mathit{eff}}^{(i)} = 0$.

Let us give a high-level overview of the attack procedure before providing a formal proof. Let $\ell = 1, \ldots, \mathsf{Q}$ enumerate the observed LPN-C plaintext-ciphertext pairs, and let $\boldsymbol{a}^{(\ell)} \in \mathbb{F}_2^k$ and $\boldsymbol{\nu}^{(\ell)} \xleftarrow{\$} \mathrm{Ber}_\tau^n$ be the LPN sample and error vector associated with the ℓth plaintext/ciphertext-pair $(\boldsymbol{x}^{(\ell)}, (\boldsymbol{a}^{(\ell)}, \boldsymbol{y}^{(\ell)}))$. We are attacking schemes that implement rejection sampling with threshold t, so for all ℓ, we are guaranteed that $\mathrm{HW}(\boldsymbol{\nu}^{(\ell)}) \le t$. Proposition 1 then tells us that the first column S_1 of the secret matrix can be recovered by running an $\mathrm{LPN}_{k,\tau_{\text{eff}}^{(1)}}$ solver on the available samples (where $\tau_{\text{eff}}^{(1)} \coloneqq \tau_{\text{eff}}$).

Table 5: Per-iteration ISD attack complexity $\log_2 \mathsf{T}_i$ for LPN-C parameter sets 3, 4, and 5 from Table 2. The notation is the same as in Table 4. All complexity values are given in $\log_2$.

Parameter Set 3			Parameter Set 4			Parameter Set 5		
i	$\tau_{\text{eff}}^{(i)}$	T_i	i	$\tau_{\text{eff}}^{(i)}$	T_i	i	$\tau_{\text{eff}}^{(i)}$	T_i
–	0.05	83.37	–	0.05	83.37	–	0.05	83.37
1	0.0349	66.34	1	0.0385	70.36	1	0.0477	80.75
2	0.0280	58.73	2	0.0350	66.45	2	0.0463	79.16
3	0.0198	49.86	3	0.0310	62.02	3	0.0444	77.01
4	0.0104	40.10	4	0.0266	57.20	4	0.0420	74.30
			5	0.0219	52.11	5	0.0391	71.03
			6	0.0168	46.69	6	0.0356	67.12
			7	0.0114	41.11	7	0.0316	62.69
			8	0.0058	35.67	8	0.0272	57.85
						9	0.0223	52.54
						10	0.0171	47.00
						11	0.0117	41.41
						12	0.0060	35.85
Total $\sum_i \mathsf{T}_i$		**66.3**	**Total** $\sum_i \mathsf{T}_i$		**70.5**	**Total** $\sum_i \mathsf{T}_i$		**81.3**
Baseline $n \cdot \mathsf{T}_0$		89.7	Baseline $n \cdot \mathsf{T}_0$		90.7	Baseline $n \cdot \mathsf{T}_0$		90.7

Define the following index subsets:

$$I_1 \coloneqq \left\{ \ell \in \{1, \ldots, \mathsf{Q}\} \mid \mathrm{HW}(\boldsymbol{\nu}^{(\ell)}) \leq t \right\}, \tag{5}$$

$$I_i \coloneqq \left\{ \ell \in I_{i-1} \mid \nu_{i-1}^{(\ell)} = 1 \right\} \text{ for } 2 \leq i \leq t+1, \tag{6}$$

i.e., I_i (inductively) indexes samples whose first $i-1$ noise elements are all 1.

Once S_1 is recovered (via Proposition 1), we learn $\nu_1^{(\ell)}$ for all samples. Next, form I_2 by discarding samples for which $\nu_1^{(\ell)} = 0$. Then for every $\ell \in I_2$, we will have that $\mathrm{HW}\big(\boldsymbol{\nu}^{(\ell)}[2..n]\big) \leq t-1$. By conditioning on the sample having been accepted and $\nu_1^{(\ell)} = 1$, we get the updated effective error rate $\tau_{\text{eff}}^{(2)}$ for the element $\nu_2^{(\ell)}$. At this point, we can recover the second column S_2 by running an $\mathrm{LPN}_{k,\tau_{\text{eff}}^{(2)}}$ solver on the subset of LPN samples given by I_2.

Iterating this procedure gives us a strategy for recovering the full matrix S: At step $i \geq 2$, discard any sample with $\nu_{i-1}^{(\ell)} = 0$ to form the next subset I_i, then run an $\mathrm{LPN}_{k,\tau_{\text{eff}}^{(i)}}$ solver on the samples in I_i to recover S_i. If this procedure reaches $i = t+1$ with $I_{t+1} \neq \emptyset$, then any surviving samples must have all remaining error elements equal 0, making the remaining columns of S solvable by Gaussian elimination. Note, however, that for $I_{t+1} \neq \emptyset$ to occur, the attacker needs enough samples to survive the successive conditioning steps. In expectation, this requires roughly $\left(\prod_{j=1}^{t} \tau_{\text{eff}}^{(j)}\right)^{-1}$ plaintext/ciphertext pairs, see section B for details. This may be prohibitive for large t, though t is often significantly smaller than n, with the smallest suggested value being $t = 4$ for parameter set 3.

Table 6: Lower bounds on the number of queries $\log_2 \mathsf{Q}$ needed for $\mathbb{E}(|I_{i+1}|) \geq 1$ after iteration i of the iterative key recovery attack on the LPN-C parameter sets of Table 2, for the first, halfway-to-final, and final iteration. The bottom row gives the minimum query requirement to carry the attack through all t iterations (highlighted in bold). Full table of per-iteration values is given in section E.

i	Query lower bounds ($\log_2 \mathsf{Q}$)				
	Set 1 ($t = 10$)	Set 2 ($t = 20$)	Set 3 ($t = 4$)	Set 4 ($t = 8$)	Set 5 ($t = 12$)
1	3.3	3.2	4.8	4.7	4.4
$t/2$	17.9	35.2	10.0	19.8	27.4
t	**42.7**	**86.1**	**22.2**	**45.1**	**62.7**

A more careful estimate yields the lower bounds listed in Table 6 (see section B for details). We indeed get the comparatively small lower bound $\mathsf{Q} \geq 2^{22.2}$ for parameter set 3, well within the capabilities of modern attackers. In contrast, parameter set 2 requires $\mathsf{Q} \geq 2^{86.1}$, which is clearly unrealistic. However, we can still gain efficiency by carrying out the attack over the first few iterations, then solving the remaining columns with the new noise parameter. For example, if one runs the attack against parameter set 2 through three iterations, the lower bound on the query complexity is only $2^{47.7}$, yet the total attack complexity goes down from 115 bits to $\log_2(\mathsf{T}_1 + \mathsf{T}_2 + (n-2) \cdot \mathsf{T}_3) = 109.73$ bits.

Proof. Let $\mathsf{E}_i \coloneqq \{\nu_1 = \ldots = \nu_i = 1\}$ denote the event that the first i noise elements are all 1 (where $\mathsf{E}_0 = \text{true}$), and let Acc denote the event that a sampled error vector $\boldsymbol{\nu} \xleftarrow{\$} \mathrm{Ber}_\tau^n$ is accepted by the rejection sampling procedure, i.e., $\mathrm{HW}(\boldsymbol{\nu}) \leq t$. For iteration $i \leq t$, we want calculate the probability

$$\Pr[\nu_i = 1 \mid \mathsf{Acc} \cap \mathsf{E}_{i-1}] \coloneqq \tau_{\text{eff}}^{(i)} .$$

First, extending the argument from Proposition 1, we get that for every $0 \leq i < t$,

$$\Pr[\mathsf{Acc} \cap \mathsf{E}_i] = \Pr[\mathsf{E}_i] \cdot \Pr[\mathsf{Acc} \mid \mathsf{E}_i] = \tau^i \; W_\tau(n-i, t-i) . \tag{7}$$

Base case: Proposition 1 covers the case $i = 1$.
Inductive step: Assume the claim holds for step $i - 1 < t$. Then, for step i:

$$\begin{aligned}
\Pr[\nu_i = 1 \mid \mathsf{Acc} \cap \mathsf{E}_{i-1}] &= \Pr[\mathsf{E}_i \mid \mathsf{Acc} \cap \mathsf{E}_{i-1}] \\
&= \frac{\Pr[\mathsf{Acc} \cap \mathsf{E}_{i-1} \cap \mathsf{E}_i]}{\Pr[\mathsf{Acc} \cap \mathsf{E}_{i-1}]} = \frac{\Pr[\mathsf{Acc} \cap \mathsf{E}_i]}{\Pr[\mathsf{Acc} \cap \mathsf{E}_{i-1}]} \\
&\overset{(7)}{=} \frac{\tau^i \; W_\tau(n-i, t-i)}{\tau^{i-1} \; W_\tau(n-(i-1), t-(i-1))} \\
&= \frac{\tau \; W_\tau(n-i, t-i)}{W_\tau(n-(i-1), t-(i-1))} .
\end{aligned}$$

That $\tau_{\text{eff}}^{(i)}(\tau, n, t) \in (0, 1/2)$ and $\tau_{\text{eff}}^{(i)}(\tau, n, t) < \tau_{\text{eff}}^{(i-1)}(\tau, n, t)$ follows by the same argument as in Proposition 1. For $i > t$, $\tau_{\text{eff}}^{(i)} = \Pr[\nu_i = 1 \mid \mathsf{Acc} \cap \mathsf{E}_{i-1}] = 0$. □

Table 7: Experimental ISD (Prange) results for Parameter Set 3, running on 50 cores for $\sim 10k$ iterations per second. Expected iterations: $(1/(1-\tau_{\text{eff}}^{(i)}))^k$. Based on 5 runs each.

i	$\tau_{\text{eff}}^{(i)}$	Iterations / Wall time			
		Expected	Observed Min	Observed Median	Observed Max
2	0.0280	3.0B / 3.4 days	113.1M / 3.1h	4.5B / 5.2 days	9.8B / 11.4 days
3	0.0198	4.7M / 8 min	750,301 / 77s	4.9M / 8.5 min	18.3M / 31.3 min
4	0.0104	3,070 / 0.4s	1,067 / 0.2s	7,471 / 1.0s	24,051 / 2.9s

Pseudocode for the iterative key-recovery attack is given in section C. We also provide complexities for the iterated attack when using LF2 [11] and quantum ISD [54] as the underlying solver in section E.

Experiments. As a proof-of-concept experiment, we ran Prange's ISD solver against parameter set 3 ($k = 768$, $\tau = 1/20$, $t = 4$) with the second, third, and fourth iterated noise rates $\tau_{\text{eff}}^{(i)}$ in place of τ. These correspond to recovering the final three columns of the secret matrix after the first column has already been recovered. We used the implementation of Esser et al. [19] to run the attack on a Linux server using 50 cores. The results are given in Table 7 and seem to be relatively consistent with our estimates.

4 Analysis of LPN-C via Modified Arora-Ge Attacks

In section 3, we used the condition on bounding the Hamming weight of the error vector $\boldsymbol{\nu}$ in a statistical way. The algebraic attack developed in this section relies on the same bounded-weight condition, albeit in a different way. Instead of treating the bound as influencing the noise distribution, we use it to provide a guarantee that higher-degree relations will vanish.

Fouque et al. [24, Section 4.2] already observed this effect, namely that bounding the Hamming weight of the error vector implies that any product of $t + 1$ entries from a ciphertext vector must contain at least one error-free term; thereby suggesting the possibility of an Arora–Ge type attack. They also discuss the possibility of using Gröbner bases to speed up the attack, but did not fully explore the details of either direction.

We develop these ideas throughout this section. In particular, we find that analyzing LPN-C requires more than directly applying the standard Arora–Ge heuristics, owing to structural properties that arise specifically from LPN-C. We also conclude that Gröbner bases are unlikely to improve upon this attack vector. We begin with a brief reminder of the linearization technique, which provides a useful background for the analysis to follow.

Short Background on Linearization Attacks. Algebraic attacks against cryptosystems typically proceed by turning the problem of recovering a secret value into that of solving a system of multivariate polynomial equations. Consider a system of polynomial equations

$$\mathcal{F} = (f_1, \ldots, f_m) \subset R = \mathbb{F}_q[x_1, \ldots, x_n],$$

where each f_i has total degree d. The most general tool for analysing such systems is through the use of Gröbner-basis algorithms (which we will return to in section 6). For the time being, we will be content with noting that for n and d fixed, algebraic attacks generally become easier with increasing m. This holds up until the point when the number of equations m is at least as large as the number of possible monomials in $\mathcal{F}$. In this case we can apply a method known as *linearization*, where we view each monomial as a linear variable, and solve the resulting system directly using linear algebra techniques. Since $\mathcal{F}$ is of degree d, there are at most $\binom{n}{d}$ possible monomials in the system for $q = 2$, and hence the linearization approach is estimated to require $\mathcal{O}\left(\binom{n}{d}^{\omega}\right)$ operations in $\mathbb{F}_2$, where $2 \leq \omega \leq 3$ is the linear algebra constant. For $q > d$, analogous estimates hold for $\binom{n+d}{d}$ possible monomials.

4.1 An Adapted Arora–Ge Attack

We view the secret $S \in \mathbb{F}_2^{k \times n}$ as kn indeterminates

$$S_w = \begin{bmatrix} s_{11} & s_{12} & \ldots & s_{1n} \\ s_{21} & \ddots & & \\ \vdots & & & \\ s_{k1} & s_{k2} & \ldots & s_{kn} \end{bmatrix}$$

and work in the boolean polynomial ring

$$\mathcal{B} := \mathbb{F}_2[s_{ij}]/\langle s_{ij}^2 + s_{ij} \rangle_{1 \leq i \leq k,\ 1 \leq j \leq n}. \tag{8}$$

Given an intercepted plaintext $\boldsymbol{x} \in \mathbb{F}_2^r$, and its associated ciphertext $(\boldsymbol{a}, \boldsymbol{y})$, define for each $j \in \{1, \ldots, n\}$ the affine-linear polynomial

$$f_j = [C(\boldsymbol{x})]_j + y_j + (a_1 s_{1j} + \ldots + a_k s_{kj}) \in \mathcal{B}. \tag{9}$$

If the error vector has Hamming weight at most t, then among the first $t+1$ entries at least one satisfies $f_j = 0$, and hence we have an "Arora—Ge-like" equation:

$$F_{\boldsymbol{x},(\boldsymbol{a},\boldsymbol{y})} = \prod_{j=1}^{t+1} f_j = 0, \tag{10}$$

which is satisfied by the true solution $S_w = S$. With multiple plaintext-ciphertext pairs, an attacker can now generate further polynomials of this form. In order to

see how many such polynomials are needed to linearize the system, we need to investigate the space spanned by the $F_{\boldsymbol{x},(\boldsymbol{a},\boldsymbol{y})}$ polynomials. To this end we define the set generated by the possible Arora–Ge equations as follows:

$$\mathcal{S}' = \left\{ F_{\boldsymbol{x},(\boldsymbol{a},\boldsymbol{y})} \in \mathcal{B} \mid (\boldsymbol{x},(\boldsymbol{a},\boldsymbol{y})) \text{ is a valid plaintext-ciphertext pair} \right\}. \tag{11}$$

Recall that the elements of $\mathcal{S}'$ can naturally be seen as vectors over $\mathbb{F}_2$ (with the possible monomials as basis elements). We write $\mathcal{S} = \mathrm{Span}(\mathcal{S}')$ for the linear space spanned by these vectors. The complexity of a linearization attack on these equations will be determined by the dimension of this space.

A monomial-count upper bound. A natural starting point is to bound the dimension of $\mathcal{S}'$ by counting the number of possible monomials. Note that each polynomial $F_{\boldsymbol{x},(\boldsymbol{a},\boldsymbol{y})}$ has degree at most $t+1$ and involves only the $(t+1)k$ variables in the first $t+1$ columns of S_w. Moreover, $F_{\boldsymbol{x},(\boldsymbol{a},\boldsymbol{y})}$ is block-multilinear: in any monomial, at most one variable can be taken from each column, since each factor f_j contributes either its constant term or a single variable from column j. Thus, a degree-i monomial is obtained by choosing i columns among the $t+1$ and then one of the k variables in each chosen column, giving

$$\dim(\mathcal{S}) \;\le\; \sum_{i=0}^{t+1} \binom{t+1}{i} k^i \;=\; (k+1)^{t+1}. \tag{12}$$

While this bound does indeed count the number of possible monomials in $\mathcal{S}$, it turns out to provide only a loose upper bound on the actual dimension of $\mathcal{S}$, due to the structural dependencies among polynomials in $F_{\boldsymbol{x},(\boldsymbol{a},\boldsymbol{y})}$. We illustrate this effect with the following example.

A Toy Example. Consider an instance of LPN-C with parameters $k = 3$ and $n = 2$, and error-correction capacity $t = 1$. For convenience, we denote the columns of the secret matrix $S_w \in \mathbb{F}^{3\times 2}$ by the variables s and z, i.e. $S_w = \begin{bmatrix} s_1 & s_2 & s_3 \\ z_1 & z_2 & z_3 \end{bmatrix}^\top$.

For a plaintext-ciphertext pair $\boldsymbol{x},(\boldsymbol{a},\boldsymbol{y})$, we write $C(\boldsymbol{x}) = (p_1,p_2)$. By Equation 9, we have $f_1 = p_1 + y_1 + a_1s_1 + a_2s_2 + a_3s_3$ and $f_2 = p_2 + y_2 + a_1z_1 + a_2z_2 + a_3z_3$. Then, the polynomial $F_{\boldsymbol{x},(\boldsymbol{a},\boldsymbol{y})}$ from Equation 10 can be expressed as

$$\begin{aligned} F_{\boldsymbol{m},(\boldsymbol{a},\boldsymbol{y})} = f_1 \cdot f_2 = {} & a_1s_1z_1 + a_1a_2(s_1z_2 + s_2z_1) + a_1a_3(s_1z_3 + s_3z_1) + a_2s_2z_2 \\ & + a_2a_3(s_2z_3 + s_3z_2) + a_3s_3z_3 + (p_1 - y_1)(a_1z_1 + a_2z_2 + a_3z_3) \\ & + (p_2 - y_2)(a_1s_1 + a_2s_2 + a_3s_3) + K, \end{aligned} \tag{13}$$

where K is a known constant depending on p_1, p_2, y_1, y_2. To highlight the different components of this polynomial, we have written the a_i coefficients in *red*;

the unknown indeterminates in *blue* and known constants (which have been influenced by the error ν_i) in green. This error dependency among the green constants is a subtle fact worth emphasizing. Indeed, without this error the factors $(p_i - y_i)$ could have been expressed purely in terms of the sampled a_j's and indeterminates, and there would be further dependencies among the linear terms in Equation 13. However, the inclusion of the errors means that the coefficients $(p_i - y_i)a_j$ may differ even for the same choice of $\boldsymbol{m}$ and $\boldsymbol{a}$, and hence we have to treat them as (somewhat) independent as we sample from $\mathcal{S}$.

It follows that Equation 13 can be rewritten as the linear form

$$\sum_{i=1}^{6} c_i u_i + \sum_{j=1}^{3} d_j s_j + \sum_{l=1}^{3} l_l z_l + K, \tag{14}$$

for suitable constants $c_i, d_j, l_l, K \in \mathbb{F}_2$, by introducing the change-of-variables

$$u_1 = s_1 z_1, \qquad u_2 = s_1 z_2 + s_2 z_1, \qquad u_3 = s_1 z_3 + s_3 z_1,$$
$$u_4 = s_2 z_2, \qquad u_5 = s_2 z_3 + s_3 z_2, \qquad u_6 = s_3 z_3.$$

We can moreover observe that these equations span a vector space of at most $\dim(\mathcal{S}) = 13$ (by including the constant term K). Note that is strictly smaller than the upper bound in Equation 12, which yields 16 in this case.

4.2 Bounding the Linear Space $\mathcal{S}$

Building on the observation from the above toy example, we now develop a tighter bound on the dimension of $\mathcal{S}$. As a first step, we present a useful lemma, to be used in the subsequent proof of our bound on $\mathcal{S}$. We adapt the convention that $\binom{a}{b} = 0$ for any integer a and $b \leq 0$.

Lemma 1 (Counting Monomials). *Let $\boldsymbol{a} = \{a_1, \ldots, a_k\}$ and $\boldsymbol{b} = \{b_1, \ldots, b_n\}$ be sets of variables, and consider a polynomial ring $R = \mathbb{F}_q[\boldsymbol{a}, \boldsymbol{b}]$ and polynomial*

$$H(\boldsymbol{a}, \boldsymbol{b}) = \prod_{j=1}^{\ell} (h_j(\boldsymbol{a}) + b_j) \in R,$$

where each $h_j(\boldsymbol{a})$ is a polynomial of degree ρ with no constant term.

Then, the total number of monomials in R that can appear in the expansion of H is upper bounded by

$$\sum_{i=0}^{\ell} \binom{\ell}{i} \left[\binom{k + \rho i}{\rho i} - \binom{k + i - 1}{i - 1} \right]. \tag{15}$$

Proof. Fix $m \in \{0, \ldots, \ell\}$ and a subset $T \subseteq \{1, \ldots, \ell\}$ with $|T| = m$. Note that any term in $H(\boldsymbol{a}, \boldsymbol{b})$ will be of the form

$$\Big(\prod_{j \in T} b_j \Big) \cdot \Big(\prod_{j \notin T} h_j(\boldsymbol{a}) \Big). \tag{16}$$

The factor $\prod_{j\notin T} h_j(\boldsymbol{a})$ is a product of $\ell-m$ polynomials, each polynomial having only monomials of degree between 1 and ρ. Hence, the degrees of the monomials in the a-product are between $\ell-m$ and $\rho(\ell-m)$.

Recall that the number of monomials in k variables of degree *exactly* d is $\binom{k+d-1}{d}$, and that the number of monomials in k variables of total degree *at most* d is $\binom{k+d}{d}$. We can therefore bound the number of monomials in Equation 16 as being at most

$$\begin{aligned}\sum_{d=\ell-m}^{\rho(\ell-m)} \binom{k+d-1}{d} &= \sum_{d=0}^{\rho(\ell-m)} \binom{k+d-1}{d} - \sum_{d=0}^{(\ell-m)-1} \binom{k+d-1}{d} \\ &= \binom{k+\rho(\ell-m)}{\rho(\ell-m)} - \binom{k+(\ell-m)-1}{(\ell-m)-1}.\end{aligned}$$

The factor $\prod_{j\in T} b_j$ is a square-free monomial of degree m in the b-variables, and, as T ranges over all m-subsets, there are exactly $\binom{\ell}{m}$ distinct b-monomials. Since the choices for the a- and b-parts are independent, the block with subsets of size m will contribute at most

$$\binom{\ell}{m}\left[\binom{k+\rho(\ell-m)}{\rho(\ell-m)} - \binom{k+(\ell-m)-1}{(\ell-m)-1}\right]$$

monomials. Summing over all $0 \le m \le \ell$ and re-indexing with $i = \ell - m$ gives the final result. □

We now have the necessary tools to derive a bound on the dimension of $\mathcal{S}$, as stated in the following result.

Proposition 3. *Consider an instance of LPN-C, where k is the number of columns in the secret key and t the error correcting capability. Moreover, write $\mathcal{S} = Span(\mathcal{S}')$, where $\mathcal{S}'$ is as defined in Equation 11. Then, the vector space dimension of $\mathcal{S}$ is subject to the following upper bound*

$$\dim(\mathcal{S}) \le \sum_{i=0}^{t+1} \binom{t+1}{i}\binom{k+i-1}{i}. \tag{17}$$

Proof. Each element in $\mathcal{S}'$ is a product of the polynomials $f_j = \langle \boldsymbol{a}, S_j\rangle + b_j$, and hence the idea will be to apply Lemma 1 with $h_j(a) = \langle \boldsymbol{a}, S_j\rangle$. Indeed, since the equations $F_{\boldsymbol{x},(\boldsymbol{a},\boldsymbol{y})}$ can only differ up to $\boldsymbol{a}, \boldsymbol{b}$, their maximal span cannot exceed the number of monomials predicted in Lemma 1, which in our case is

$$\sum_{m=0}^{t+1} \binom{t+1}{m}\left[\binom{k+(t+1-m)}{(t+1-m)} - \binom{k+(t+1-m)-1}{(t+1-m)-1}\right].$$

Using Pascal's identity, the expression within the brackets simplifies as follows: $\binom{k+(t+1-m)}{t+1-m} - \binom{k+(t+1-m)-1}{(t+1-m)-1} = \binom{k+(t+1-m)-1}{t+1-m}$, and reindexing with $i = t+1-m$ gives us the claimed bound. □

Note that applying Proposition 3 to the case $k = 3$, $t = 1$ gives an upper bound of 13, which is exactly what we observed in the toy example of subsection 4.1.

Once the dimension of $\mathcal{S}$ is known, a solution for this linear space can be recovered in time $\mathcal{O}\left(\dim(\mathcal{S})^{\omega}\right)$ with query complexity $\mathcal{O}\left(\dim(\mathcal{S})\right)$, where ω denotes the linear algebra constant. As some of these entries correspond to the associated entries of the secret matrix (as, e.g., seen in Equation 14), this immediately recovers $t + 1$ columns of the secret key.

Remark 1. The expression on the right hand side of Equation 17 is known in the combinatorics literature as the *asymmetric Delannoy numbers*, which have been studied in relation to the enumeration of certain lattice paths. There seems to be no known closed-formula expression, but we refer the reader to [35] for further properties and recursion relations.

4.3 Further Discussion and Alterations

Recovering More than One Column. Similar to what we saw in subsection 3.2, the above attack can also be improved once the first column(s) have been recovered. For instance, one may restrict to plaintext-ciphertext pairs that have an error in the recovered column and then apply Proposition 3 to the resulting system with $t - 1$.

Restricting the Arora–Ge Systems. We can also restrict the Arora—Ge systems: The dimension in Proposition 3 depends on the observed vectors $\boldsymbol{a}$, which opens up the possibilities for data-time trade-offs. Concretely, fix an integer $1 \leq s \leq k$, and only keep plaintext-ciphertext pairs for which the first s entries of $\boldsymbol{a}$ are zero. This allows us to directly apply Proposition 3 with $k - s$ number of effective columns, and the attack proceeds accordingly with the reduced dimension.

Secret Toeplitz Matrix. Gilbert et al. [28,29] propose using a Toeplitz matrix as the secret key to reduce the key size in HB+ and LPN-C. Clearly, if the secret is Toeplitz, then recovering a single column trivially reveals the entire key. One might therefore hope that Toeplitz structure also benefits our algebraic attack, however, we have not found this to be the case. More precisely, we have not found this structure to be compatible with the linear space $\mathcal{S}$ explored in Proposition 3. We defer the technical explanation to Appendix D.1.

On the Use of Gröbner Bases. Finally, previous sections discussed a particular kind of LPN setting where we can avoid errors by generating high-degree polynomials. It is natural to ask whether the analysis can be improved through the use of Gröbner bases, along the lines of [2,52]. We note that this was also discussed by Fouque et al. [24]. While this can lead to attacks using less data complexity, it comes at the cost of a massive increase in time complexity, as the concise structure of Proposition 3 will be lost at higher degrees.

We provide further reasoning in Appendix D.2.

Complexity and Comparison with Other Attacks. When t is small relative to k, we expect that the sum Equation 17 is well-estimated by its highest-degree term $\binom{k+t}{t+1}$, and the time complexity grows on the order of $\mathcal{O}\left(k^{t\omega}\right)$. Consequently, we expect this attack to be competitive in practice only for very small t.

Writing (Q, T) for the query and time complexity of the attack from subsection 4.2 and using $\omega = 2.8$ (Strassen's algorithm [53]), we estimate the complexities for the three parameter sets in Table 2 as $(\log_2 \mathsf{Q}, \log_2 \mathsf{T}) = (74, 208), (125, 351)$ and $(41, 115)$, respectively. Thus, while the attack is generally not competitive with the methods presented in section 3 in terms of time complexity, it can nevertheless offer improvements in query complexity for certain parameters.

5 Analysis of Delinearized LPN-C Variants

We now shift our attention to the LPN-C variants with delinearization steps that was proposed in [24]. Note that this will include shifting our focus to the MLPN-C problem (Definition 5). In fact, the ensuing analysis will not depend on the characteristic of the field, and hence we will work with the more general case of $\mathbb{F}_q$ where q can be any prime power (and we recover MLPN-C as the special case $q = 2^\ell$ for some integer ℓ).

5.1 The Delinearization Method

The main difference from the LPN-C problem we studied in section 4 (beyond the change of fields) is the inclusion of a delinearization step. This adds an extra step to the encryption by applying a non-linear function $F : \mathbb{F}_q^n \longrightarrow \mathbb{F}_q^n$. To remain efficient in the hybrid homomorphic setting, it was proposed to build F through a composition $F = G_r \circ G_{r-1} \circ \cdots \circ G_1$, where each $G_i : \mathbb{F}_q^n \longrightarrow \mathbb{F}_q^n$ can be written as a polynomial of low degree. In particular, [24] suggests using $r = 2$, and quadratic G_i polynomials. Encryption closely resembles that of Definition 5 (and, consequently, that of LPN-C), except that $F(\cdot)$ is applied to $\boldsymbol{a}S$, i.e.,

$$y = E(\boldsymbol{x}) + F(\boldsymbol{a}S) + \boldsymbol{\nu}.$$

Decryption now generalizes naturally by applying $F(\cdot)$ in a symmetrical manner.

The idea behind the delinearization step is to enhance the security against possible attacks. Indeed, $F(\boldsymbol{a}S)$ is now represented by non-linear polynomials in the secret key S, which drastically increases the complexity of algebraic attacks, as well as more direct approaches solving the underlying LPN/LWE problem. Indeed, this allows [24] to propose parameter sets with a significantly smaller error rate τ, and no upper bound on the hamming weight of error vectors (as the ensuing probability of decryption errors will be minuscule).

The security of this construction crucially depends on the choice of delinearization function F, and in the following analysis we cover two cases:

Case I. Here, F is the composition of generic polynomials G_i. That is, there is no particular structure for an attacker to exploit, beyond the length of the composition and degree of the involved polynomials. (Recall in particular that such polynomials G_i are expected to form regular systems, as we will see in Definition 7.)

Case II. This is the function suggested in [24]. We use underlined notation to distinguish the respective functions from those of Case I, i.e., $\underline{F} = \underline{G}_r \circ \cdots \circ \underline{G}_1$. We moreover write the functions as vectors of polynomial mappings, i.e., $\underline{G}_i = (\underline{g}_{i,1}, \ldots, \underline{g}_{i,n})$, where $\underline{g}_{i,j} : \mathbb{F}_q^n \to \mathbb{F}_q$. For each pair i, j, we choose three integers l_1, l_2, l_3 in $[1, n]$, and define

$$\underline{g}_{i,j}(x_1, \ldots, x_n) = x_{l_1} x_{l_2} + x_{l_3}. \tag{18}$$

The benefit of this choice is to minimize the number of multiplications (recalling the motivation of the hybrid homomorphic setting). Fouque et al. [24] did not further expand on how to choose the various integer triples (l_1, l_2, l_3), beyond ensuring that entries do not linearly cancel out across different j's.

Suggested Parameters. [24, Section 4.2] suggests two parameter sets using $\underline{F}$ at degree $d^r = 2^2$. The sets are $(k, \tau) = (80, 6 \cdot 10^{-5})$ and $(k, \tau) = (512, 4 \cdot 10^{-9})$. Note that, due to the nonlinear function F, the small error rates are still sufficient to add security against attacks that rely on a large number of samples (as we'll, e.g., see in the next section).

Both parameter sets are believed to achieve 80-bit security, where the best proposed attack is based on solving Gröbner bases at degree 5. (We also remark that the number of secret key columns n was not discussed in the security analysis.)

5.2 Arora-Ge in the Delinearized Setting

The Arora-Ge attack can (not surprisingly) be adapted to delinearized MLPN-C. Since there is no longer a bound on the Hamming weight of a single codeword, we might as well fix a single entry to focus on. As a result, the attack will be closer to the setting initially envisioned by Arora and Ge [4].

We start by adapting Equation 10 and Equation 11 to suit this setting. We fix a set of small error values $\{\nu_0 = 0, \nu_1, \ldots, \nu_{l-1}\}$, and define

$$FD_{l,\boldsymbol{x},(\boldsymbol{a},\boldsymbol{y})} = \prod_{j=0}^{l-1} (F(\boldsymbol{a}S) + E(\boldsymbol{x}) - \boldsymbol{y} + \nu_j) = 0, \tag{19}$$

and the corresponding set

$$\mathcal{SD}'_l = \left\{ FD_{l,\boldsymbol{x},(\boldsymbol{a},\boldsymbol{y})} \mid \boldsymbol{x}, (\boldsymbol{a}, \boldsymbol{y}) \text{ is a valid plaintext-ciphertext pair} \right\}. \tag{20}$$

The following result can be seen as the analogue of Proposition 3, adapted to the delinearized setting.

Proposition 4. *Consider an instance of MLPN-C with delinearization, as described in subsection 5.1. where k is the number of columns in the secret key and t the error correcting capability. Moreover, let $\mathcal{SD}'_l$ be as defined in Equation 20. Then the vector space dimension of $\mathcal{SD}_l = \mathrm{Span}(\mathcal{SD}'_l)$ is subject to the following upper bound.*

$$\dim(\mathcal{SD}_l) \leq \sum_{i=0}^{l} \binom{l}{i} \left[\binom{k + d^r i}{d^r i} - \binom{k+i-1}{i-1} \right]. \tag{21}$$

Proof. The proof follows exactly as in Proposition 3, i.e., by applying Lemma 1 with $h_j(\boldsymbol{a}) = F(\boldsymbol{aS})$ and $\rho = d^r$ (noting that any constant term from F can be included in b_j). □

The attack proceeds in the usual manner, and we can bound its complexity using $\dim(\mathcal{SD}_l)$, similar to what we saw in subsection 4.2. Note that in addition to the linear algebra complexity, we now also have to take into account the success probability for the attack. Hence, we estimate the complexity to be

$$(\dim(\mathcal{SD}_l))^{\omega}\, e^{\tau \cdot \dim(\mathcal{SD}_l)} \tag{22}$$

field operations, where we have used the approximation $(1-\tau)^m \approx e^{-\tau m}$.

Observe that since $F(\mathbf{aS})$ is not linear, the degree (and hence complexity) of the system quickly explodes with larger l. In the proposed parameters, with F of degree 4, we find that the best approach is to keep $l = 1$. This amounts to the straightforward linearization attack, where the attacker attempts to solve the polynomial system from a fixed i_0-th output across m plaintext-ciphertext pairs, hoping that they are all error-free. Note that Proposition 4 still holds for the $l = 1$ case, and indeed yields a non-trivial bound that is generally smaller than the number of monomials in F (similar to what we observed in subsection 4.1). We estimate the attack complexity on the suggested parameters in Table 8.

5.3 On the (In)applicability of Gröbner Basis Improvements

Similar to the discussion in subsection 4.3, we next argue why the methods of the previous subsection are unlikely to be improved by Gröbner basis calculations. Note that this observation is more impactful for delinearized LPN-C, as [24] identified that the use of Gröbner basis methods leads to the most efficient attack, and suggested parameters accordingly.

As discussed above, we expect that the best algebraic approach is to use equation (19) with $l = 1$. We now explore this attack when the number of equations is $m < dim(\mathcal{SD}_l)$. The probability for (one iteration) of this approach to work is $(1-\tau)^m$, which results in a trade-off between probability and polynomial solving complexity when it comes to choosing m.

A Study in Degree Five. In the following we restrict to the setting where the equations are of degree 4, and the associated Gröbner basis computation

Table 8: Estimated time complexities against the proposed parameters for delinearized MLPN-C with expected 80-bit security.

k	τ	$\log_2(m_{5,\text{prev}})$	$\log_2(m_4)$	$\log_2(\mathsf{T}_4)$	$\log_2(m_5)$	$\log_2(\mathsf{T}_5)$
80	$6 \cdot 10^{-5}$	19	21	209	24	1,856
512	$4 \cdot 10^{-9}$	29	31	80	35	287

only involves polynomials up to degree 5, which captures both the suggested parameters, as well as the best attack discussed in [24, Section 4.2]. We trust that this restricted analysis suffices to convince the reader that Gröbner bases will not provide improvements for the delinearized MLPN-C.

As argued in Appendix D.2, we do not see a way where the structure of $\mathcal{SD}$ explored in subsection 5.2 generalizes to higher degrees: Restricting to the $\underline{F}$ function of Case II, where the components are defined as in (18), and focusing on the monomials of highest degree, we see that the homogeneous degree 4 part of all entries in $\underline{F}$ is of the form

$$(\langle \boldsymbol{a}, S_{i_1} \rangle) \cdot (\langle \boldsymbol{a}, S_{i_2} \rangle) \cdot (\langle \boldsymbol{a}, S_{i_3} \rangle) \cdot (\langle \boldsymbol{a}, S_{i_4} \rangle) \,. \tag{23}$$

In other words, it is multilinear w.r.t. four columns of S. To exclude weak cases, we furthermore assume that i_1, i_2, i_3, i_4 are pairwise distinct for all entries of $\underline{F}$.

Now suppose that we want to generate a degree 5 polynomial by multiplying with variables from S_{i_1}. This yields mk polynomials in monomials of degree 5 where the S_{i_1}-variables have degree 2; and the variables from columns i_2, i_3, i_4 are of degree 1. In order to ensure that we have enough equations to generically cancel out the degree 5 monomials, we then require

$$m \geq \binom{k+1}{2} k^2 \,. \tag{24}$$

(We could also multiply with variables from the other columns, but this results in a disjoint sets of monomials, and hence not affect the above observation.) Let us use this bound with equality as the value of m, and lower bound the number columns of the Macaulay matrix we have to solve by the number of aforementioned degree 5 monomials. Similar to (22), we can now conservatively estimate the complexity of a system solving approach at degree 5 by

$$\left(\binom{k+1}{2} k^3 \right)^{\omega} e^{\tau \binom{k+1}{2} k^2} \tag{25}$$

field operations. Due to our simplification of only focusing on the homogeneous degree 4 part, we expect this to yield a loose lower bound for the complexity. The bound also holds in case I for regular F (although we expect it to be even looser, as the homogeneous degree 4 part is denser than Equation 23).

Revisiting the Suggested Parameters. In Table 8 we apply the algebraic attacks discussed so far to the suggested parameters. Since both parameters used $\underline{F}$ of degree 4, we denote m_4 (resp. m_5) as the number of equations needed to solve at degree 4 (resp. 5), given by Equation 21 (resp. Equation 24). We let T_4 and T_5 be the respective complexity estimates from Equation 22 and Equation 25, where we used (the very conservative) $\omega = 2$ for a consistent comparison with [24], and $m_{5,\text{prev}}$ is the number of equations [24] estimate is needed to solve at degree 5.

Table 8 shows that earlier estimates for m_5 are significantly smaller than our estimates. This, in turn, contributes to a massive discrepancy in estimated attack complexity: while the attack at degree 5 was previously estimated close to 2^{80} field operations, we now estimate that 2^{1856} and 2^{287} operations are required.

Somewhat counterintuitively, we find that m_4 is smaller than m_5 in the two cases, essentially owing to the structure from Proposition 4. With $\mathsf{T}_4 = 2^{209}$, we believe that the security of the first parameter set has been greatly *under*estimated. The second set obtains an estimated 2^{80} time complexity in this analysis, but with the use of $\omega = 2$ we still believe there is ample margin here for 80-bit security.

6 A Dual Algebraic Attack

In the previous section we looked at methods that sample plaintext-ciphertext pairs and solve for the entries in S directly via linearization or Gröbner bases techniques. This resulted in complexities that mainly depended on k, the number of columns in S. In this section we present a different algebraic attack that instead mainly depends on n, the number of rows in S. We start with a brief refresher on state-of-the-art methods for solving determined polynomial systems.

Background: Solving Determined systems. Like in section 4, we consider a general polynomial system $\mathcal{F} = (f_1, \ldots, f_n) \subset R = \mathbb{F}_q[x_1, \ldots, x_n]$ in degree d that is *determined*, i.e. the number of equations and variables are the same. The solving procedure now involves the computation of Gröbner bases for the ideal $\langle \mathcal{F} \rangle$. We refer the reader to [15] for the fundamentals on Gröbner bases, and to [6, Section 2 and 3] for a recent overview of the procedure and its complexity.

For our purposes, it suffices to point out that the bottleneck of the solving method consists of two steps: *i*) computing a Gröbner basis of $\langle \mathcal{F} \rangle$ in some "computation-friendly" monomial order $\prec$; and *ii*) changing the monomial order of this basis from $\prec$ to the lexicographic order, from where a solution can easily be recovered. Step *i*) is typically done via algorithms such as F_4 [21], but its complexity is generally hard to predict unless $\mathcal{F}$ is a *regular* polynomial system. We note that this regularity is a generic condition, and most randomly generated polynomial systems are expected to behave this way, (although as seen in Lemma 2, subsection 6.3, one should be careful in assuming that this extends to *structured* systems arising in cryptography).

Definition 7 (Regular Polynomial Systems). *A homogeneous polynomial system $\mathcal{F} = (f_1, \ldots, f_n) \subseteq R = \mathbb{F}_q[x_1, \ldots, x_n]$ is said to be* regular *if $f_1 \neq 0$ and f_i is not a zero-divisor in $R/\langle f_1, \ldots, f_{i-1}\rangle$ for all $2 \leq i \leq n$.*

An inhomogeneous *polynomial system $\mathcal{F}$ is said to be regular if its top homogeneous part, $F^h = (f_1^h, \ldots, f_k^h)$, is regular.*

If $\mathcal{F}$ is regular, then we have to compute with polynomials up to a degree D given by the Macaulay bound (left side of Equation 26), and the complexity of F_4 can be bounded by performing linear algebra operations at this degree [5]. That is, we can bound the complexity of step i) by:

$$D = 1 + n(d-1), \qquad \mathsf{T}_1 = \mathcal{O}\left(\binom{D+n}{n}^{\omega}\right) . \tag{26}$$

There are also dedicated algorithms for step ii) [22]. The complexity of these algorithms reduces to linear algebra on a matrix the size of $d_{\langle\mathcal{F}\rangle}$, which is the *degree of the ideal* $\langle\mathcal{F}\rangle$. Again, $d_{\langle\mathcal{F}\rangle}$ is hard to determine in general, but for regular $\mathcal{F}$ it is given by Bézout's Theorem (left side of Equation 27), making the complexity of step ii)

$$d_{\langle\mathcal{F}\rangle} = d^n, \qquad \mathsf{T}_2 = \mathcal{O}\left(d_{\langle\mathcal{F}\rangle}^{\omega}\right) . \tag{27}$$

6.1 Overview of the Attack

The core idea of the second algebraic attack is to search for outputs where the delinearization function F has a unique preimage, and use these to derive linear constraints on S. Specifically, suppose that we have an error-free plaintext-ciphertext pair $(\boldsymbol{m}, (\boldsymbol{a}, \boldsymbol{y}))$ (i.e. $\boldsymbol{\nu} = 0$), and recall that $\boldsymbol{a}$ gets mapped by the composition $\boldsymbol{a} \longmapsto \boldsymbol{a}S \longmapsto F(\boldsymbol{a}S)$. If F has a unique preimage $\boldsymbol{b} \in \mathbb{F}_q^n$ at $C(\boldsymbol{m}) - \boldsymbol{y}$, then $\boldsymbol{a}S = \boldsymbol{b}$. This allows for the following two-step approach:

1. For a number of samples $(\boldsymbol{m}, (\boldsymbol{a}, \boldsymbol{y}))$, solve the system of equations

$$F(\boldsymbol{z}) = C(\boldsymbol{m}) - \boldsymbol{y}\,, \tag{28}$$

   with respect to the indeterminates $\boldsymbol{z} = z_1, \ldots, z_n$. If the system has a unique solution $\boldsymbol{b} \in \mathbb{F}_q^n$, then the pair $(\boldsymbol{a}, \boldsymbol{b})$ is stored in a list $\mathcal{L}$.
2. When $|\mathcal{L}| \gtrsim k$ we can recover S by solving the linear system of equations

$$\boldsymbol{a}S = \boldsymbol{b}, \qquad \text{for } (\boldsymbol{a}, \boldsymbol{b}) \in \mathcal{L}\,. \tag{29}$$

Step 2 fails to recover the correct values of S if the samples used to construct $\mathcal{L}$ included errors. If this is the case, the attacker simply reruns the two steps with different samples until the correct S is recovered. (Note that checking the validity of a solution of Equation 29 takes negligible time compared to the two steps.)

Remark 2. One may wonder whether a similar attack is applicable to LPN-C without a delinearization step, but this is not the case. Indeed, the comparable algebraic restriction we used in Equation 10, simply considers the possible $t+1$ products among its inputs. By construction, all vectors of Hamming weight $\leq t$ satisfy this restriction.

Number of Solving Steps. The running time of the above method is mainly determined by solving systems of the form in Equation 28 a number of times. The best choice of solving algorithm depends on n, $\mathbb{F}_q$ and the specific choice of F. We will further explore this question in subsection 6.2 and subsection 6.3. For now we are interested in estimating the number of systems to be solved.

The list $\mathcal{L}$ constructed in Step 1 requires k instances where Equation 28 yields a unique solution. Moreover, the probability for a determined random polynomial system over a prime field to have a unique solution is proven to asymptotically (in n) be $1/e$ [25]; this is also observed to hold when n is small [25, Sect. 4]. Finally, the probability for $\mathcal{L}$ to be error-free is given by $(1-\tau)^{kn}$. Hence, if we assume that Equation 28 behaves like a random system in terms of its solutions, we expect the attack to require solving $(1-\tau)^{-kn}ek$ polynomial systems.

6.2 Case I: Regular Delinearization

We consider the problem of solving a single instance of the system of Equation 28 when F is as in Case I of subsection 5.1. It is helpful to explicitly write out the vectorial components of this equation. That is, for a fixed tuple $(c_1, \dots c_n) \in \mathbb{F}_q^n$, we define the polynomial system

$$\mathcal{F} = (f_i - c_i)_{1 \leq i \leq n} \subseteq F_q[x_1, \dots, x_n] \ , \tag{30}$$

where $f_1, \dots, f_n$ are the degree d^r polynomials defining F. As a first step, we are interested in computing a Gröbner basis (in the grevlex order) for the associated ideal $\langle \mathcal{F} \rangle$. In order to apply the machinery described in the beginning of section 6, we need that $\mathcal{F}$ is regular. This can be challenging to verify in practice – especially when d^r is large – but can be guaranteed by the choosing the intermediate functions G_i to be regular, as seen in the following.

Proposition 5 (Theorem 104, [49]). *Let $\mathcal{F}$ be as defined in Equation 30, and suppose that for each $1 \leq i \leq r$ the n polynomials describing G_i form a regular system over $\mathbb{F}_q[x_1, \dots, x_n]$. Then $\mathcal{F}$ is regular over $\mathbb{F}_q[x_1, \dots, x_n]$.*

For a regular $\mathcal{F}$, the Macaulay bound yields a degree of regularity of $n(d^r-1)+1$, and hence we estimate the time complexity to solve $\mathcal{F}$ to be

$$\mathcal{O}\left(\binom{nd^r+1}{n}^{\omega}\right). \tag{31}$$

However, a better strategy may be to take advantage of the fact that F is constructed as the composition of polynomials of low degree. Indeed, we will see that this can be exploited by computing a series of smaller Gröbner bases that can be combined into a Gröbner basis for the ideal $\langle \mathcal{F} \rangle$.

A Second Modeling. To further develop this strategy, we define a second polynomial system, and introduce some notation. We write $\mathbf{y} = \{y_{i,j}\}_{1 \leq i \leq r, 1 \leq j \leq n}$, i.e., a set of rn indeterminates, and denote its associated polynomial ring by $\mathbb{F}_q[\mathbf{y}]$. For each of the functions $G_i : \mathbb{F}_q^n \to \mathbb{F}_q^n$ comprising F, we let $g'_{i,1}, \ldots, g'_{i,n}$ be the associated polynomials of degree d in n variables. Finally, we define the polynomial system $\mathcal{G} = (g_{i,j})_{1 \leq i \leq r, 1 \leq j \leq n} \subseteq \mathbb{F}_q[\mathbf{y}]$ as

$$\begin{aligned} g_{i,j} &= g'_{i,j}(y_{i,1}, \ldots, y_{i,n}) - y_{i+1,j}, && \text{for } 1 \leq i \leq r-1, 1 \leq j \leq n \\ g_{r,j} &= g'_{r,j}(y_{r,1}, \ldots, y_{r,n}) - c_j, && \text{for } 1 \leq j \leq n. \end{aligned} \tag{32}$$

It is now straightforward to verify that $\mathcal{F}$ and $\mathcal{G}$ essentially share the same solution set by identifying $x_1, \ldots, x_n$ with $y_{1,1}, \ldots, y_{1,n}$, which uniquely determines the remaining variables in $\mathbf{y}$. Therefore, we can focus our attention on solving the latter polynomial system.

Proposition 6. *For each $1 \leq i \leq n$, let $\mathcal{G}$ be as defined in Equation 32, and let $(g'_{i,1}, \ldots, g'_{i,n})$ be regular over $\mathbb{F}_q[y_{i,1}, \ldots, y_{i,n}]$. Moreover, we write B_i for the Gröbner basis of $\langle g_{i,1}, \ldots, g_{i,n} \rangle \subseteq \mathbb{F}_q[\mathbf{y}]$, with respect to some fixed, graded monomial order $<$. Then*

$$\mathcal{B} = \bigcup_{i=1}^{r} B_i$$

is a Gröbner basis of $\mathcal{G}$ with respect to the same order $\prec$.

Proof. Note that since each G_i forms a regular system, there are no non-trivial degree fall polynomials in $\langle G_i \rangle$. In particular, for any $h \in B_i$ we have $LM(h) \in \mathbb{F}_q[y_{i,1}, \ldots, y_{i,n}]$, where $LM(h)$ denotes the leading monomial of h under the graded order $\prec$.

Now let $h_1 \in B_{i_1}, h_2 \in B_{i_2}$ be distinct elements, and let $S(h_1, h_2) \to_{\mathcal{B}} 0$ denote that their S-polynomial reduces to 0 modulo $\mathcal{B}$ (which follows from [15, Ch. 2 §6 and §9]). If $i_1 \neq i_2$, then h_1 and h_2 will have coprime leading monomials by the above observation, and $S(h_1, h_2) \to_{\mathcal{B}} 0$ by [15, Ch. 2 §9, Proposition 4]. If $i_1 = i_2$ then $S(h_1, h_2) \to_{\mathcal{B}} 0$ by the refined Buchberger's Criterion [15, Ch. 2 §9, Theorem 3] since B_{i_1} is a Gröbner basis. Since all S-polynomials of distinct pairs in $\mathcal{B}$ reduces to 0, we have that $\mathcal{B}$ is a Gröbner basis of $\langle \mathcal{B} \rangle = \langle \mathcal{G} \rangle$ by another application of Buchberger's Criterion. □

By Proposition 6 we can solve $\mathcal{F}$ by computing r Gröbner bases, each requiring about $\mathcal{O}\left(\binom{nd+1}{n}^{\omega}\right)$ field operations, as opposed to one large basis as in Equation 31. We note from Equation 32 that only $g_{r,1}, \ldots, g_{r,n}$ will change with as we differ the tuples $(c_1, \ldots, c_n)$. We can therefore precompute the Gröbner bases $B_1, \ldots, B_{r-1}$, and need only compute B_r for each iteration of the attack. Finally, we need also count the cost of changing the monomial order from Equation 27, where the degree of the ideal $\langle \mathcal{F} \rangle$ will be d^{rn}. Summing everything up yields the following complexity estimate for the dual algebraic attack with regular F:

$$\mathsf{T} = \mathcal{O}\left(r\binom{nd+1}{n}^{\omega} + (1-\tau)^{-kn} ek \left(\binom{nd+1}{n}^{\omega} + d^{rn\omega}\right)\right). \tag{33}$$

6.3 Case II: The Function $\underline{F}$

We now turn our attention to the second case considered in subsection 5.1, namely with delinearization function $\underline{F} = \underline{G}_r \circ \cdots \circ \underline{G}_1$, and the $\underline{G}_i$'s are defined in Equation 18. An immediate question is whether $\underline{F}$ forms a regular system, so that the analysis of subsection 6.2 can be directly applied. We answer this in the negative with the following result.

Lemma 2. *The n polynomials in R describing $\underline{F}$, as defined in subsection 5.1, does not form a regular sequence.*

Proof. Recall from Definition 7 that regularity only depends on the homogeneous components of $\underline{F}$ of highest degree, and that for a polynomial f we write f^h to mean its top homogeneous part. Now write $(m_1, \ldots, m_n)$ for the homogeneous degree d^{r-1} components of the polynomial mapping $\underline{G}_{r-1} \circ \cdots \circ \underline{G}_1$. We observe that there must be at least two polynomials in $\underline{F}$ that share the same m-factor in their highest degree part. That is, there exists suitable indices i_1, i_2, i_3 and $j_0 \neq j_1$ such that $\underline{g}^h_{r,j_0} = m_{i_1} m_{i_2}$ and $\underline{g}^h_{r,j_1} = m_{i_1} m_{i_3}$. But this means that

$$m_{i_2} \underline{g}^h_{r,j_1} = 0 \in R/\langle \underline{g}^h_{r,j_0} \rangle \qquad \text{and} \qquad m_{i_2} \notin \langle \underline{g}^h_{r,j_0} \rangle \ , \tag{34}$$

which shows that $\underline{F}$ fails to be regular by Definition 7. □

Although this lemma shows that we cannot directly apply Proposition 6, we still believe that the underlying idea of splitting the Gröbner basis computation into that of the smaller bases $B_1, \ldots, B_r$ is expected to work. Indeed, we observe from the proof of Proposition 6 that the regularity assumption can be eased as long as the leading monomials of the elements in B_{i_0} only contains variables $y_{i_0,j}$. With this in mind, we can inspect how the zero-divisor constructed in the proof of Lemma 2 corresponds to a degree fall polynomial in the affine case for B_1. By considering the modeling of Equation 32, with the $\underline{g}_{i,j}$'s of Equation 18 in place $g'_{i,j}$, we have the polynomials $g_{1,j_1} = y_{1,l_1} y_{1,l_2} + y_{1,l_4} - y_{2,j_1}$, and $g_{1,j_2} = y_{1,l_1} y_{1,l_3} + y_{1,l_5} - y_{2,j_2}$, and the zero divisor in Equation 34 will correspond to the polynomial

$$y_{1,l_3} g_{1,j_1} - y_{1,l_2} g_{1,j_2} = y_{1,l_3} \left(y_{1,l_4} - y_{2,j_1} \right) - y_{1,l_2} \left(y_{1,l_5} - y_{2,j_2} \right) \ .$$

The leading monomial of this polynomial is then still in the $y_{1,j}$-variables for any monomial order $\prec$ satisfying $y_{1,j} > y_{2,j'}$. As such, we still expect a variant of Proposition 6 to hold as long as the linear terms of Equation 18 are carefully chosen in a manner that avoids further cancellations in the $y_{i_0,j}$-terms.

We remark that the degree fall polynomials caused by Equation 34 is likely to benefit the attacker, as they are expected to cause the individual Gröbner bases $B_1, \ldots, B_r$ to be solved at a smaller degree than the respective Macaulay bounds, hence yielding smaller complexities than what was predicted in subsection 6.2. Similarly, we expect their presence to cause the Bézout bound to only yield a loose upper bound. To conclude, we expect that the complexity to solve

the polynomial system from $\underline{F}$ is loosely upper bounded by Equation 33, and that the exact complexity will greatly vary depending on the choices of indices in Equation 18.

Solving for Smaller Fields The analysis in subsection 6.2 and in this section implicitly assumes that the field size $\mathbb{F}_q$ is large enough that solving strategies based on guessing (some of) the variables will not be feasible. Smaller values of q gives the attacker increased freedom in terms of solving strategies. A particular example is the Crossbred algorithm [37] which, for $\underline{F}$, can also be adapted to take advantage of the zero divisors in the proof of Lemma 2 (see, e.g., [43, Section 4.3] for an analogous example).

7 On the Potential Future of Delinearized MLPN-C

The presented cryptanalysis on delinearized MLPN-C gives a two-sided story. On the one hand, the analysis of section 5 suggests that previous analysis has underestimated the security of this construction. Indeed, the attacks on the first parameter set in Table 8 are comfortably above the claimed 80-bit security.

On the other hand, the main observation of the analysis in section 6 is that n, the number of rows in the secret matrix, must also be carefully chosen for the delinearized variants of MLPN-C to remain secure. Recall that the role of n was not discussed in the initial security analysis of [24]. Indeed, with the choice of $\omega = 2$ used in this analysis, we find using Equation 33 that we need $n \geq 19$ for the complexity of this attack against the first parameter set in Table 8 to go beyond 2^{80}, and $n \geq 18$ for the second parameter set. Moreover, these numbers are mainly for the delinearization function F of case I; extra care needs to be taken with the choice of $\underline{F}$ in case II, as discussed in subsection 6.3.

Lessons From Other FHE-Friendly Encryption Schemes. As noted in the introduction, the zoo of FHE-friendly encryption schemes has greatly expanded since the introduction of delinearized MLPN-C. We point to a couple of ideas from this field that may particularly complement MLPN-C.

Some designs have adopted truncation at the output to prevent MitM attacks [18,30,34]. If this can be adopted for delinearized MLPN-C, it would go a long way to eliminate the attack vector in section 6, which crucially relies on the ability to find preimages of the delinearization function.

While Fouque et al. [24] viewed decryption errors as something to be avoided, added errors are treated more like a feature in Rubato [34], which targets approximate homomorphic encryption. We wonder if there are benefits to be had by designing delinearized MLPN-C variants for this setting.

In conclusion, we believe that delinearized MLPN-C may still be valuable to the area of FHE-friendly encryption due to its efficiency and simplicity of design. Our work suggests that the space for parameters and design variants remain largely unexplored: we leave this as an open problem for future work.

References

1. A. Akavia. *Learning noisy characters, multiplication codes, and cryptographic hardcore predicates*. PhD thesis, Massachusetts Institute of Technology, 2008.
2. M. R. Albrecht, C. Cid, J.-C. Faugère, and L. Perret. Algebraic Algorithms for LWE. Cryptology ePrint Archive, Paper 2014/1018, 2014.
3. M. R. Albrecht, C. Rechberger, T. Schneider, T. Tiessen, and M. Zohner. Ciphers for MPC and FHE. In *Advances in Cryptology — Proceedings of EUROCRYPT 2015*, volume 9056 of *Lecture Notes in Computer Science*, pages 430–454. Springer, 2015.
4. S. Arora and R. Ge. New algorithms for learning in presence of errors. In *Proceedings of International Colloquium on Automata, Languages, and Programming (ICALP) 2011*, volume 6755 of *Lecture Notes in Computer Science*, pages 403–415. Springer, 2011.
5. M. Bardet, J.-C. Faugére, B. Salvy, and B.-Y. Yang. Asymptotic Behaviour of the Degree of Regularity of Semi-Regular Polynomial Systems. In *Proceedings of MEGA*, 2005.
6. A. Bariant, A. Boeuf, A. Lemoine, I. Manterola Ayala, M. Øygarden, L. Perrin, and H. Raddum. The Algebraic FreeLunch: Efficient Gröbner Basis Attacks Against Arithmetization-Oriented Primitives. In *Advances in Crypotology — Proceedings of CRYPTO 2024*, volume 14923 of *Lecture Notes in Computer Science*, pages 139–173. Springer, 2024.
7. M. R. Bender, J.-C. Faugère, and E. Tsigaridas. Towards Mixed Gröbner Basis Algorithms: the Multihomogeneous and Sparse Case. In *Proceedings of the ACM International Symposium on Symbolic and Algebraic Computation (ISSAC) 2018*, pages 71–78, 2018.
8. D. J. Bernstein and T. Lange. Never trust a bunny. In *Proceedings of Radio Frequency Identification. Security and Privacy Issues (RFIDSec) 2012*, volume 7739 of *Lecture Notes in Computer Science*, pages 137–148. Springer, 2012.
9. A. Blum, A. Kalai, and H. Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *Journal of the ACM (JACM)*, 50(4):506–519, 2003.
10. S. Bogos, D. Korolija, T. Locher, and S. Vaudenay. Towards efficient lpn-based symmetric encryption. In *International Conference on Applied Cryptography and Network Security*, pages 208–230. Springer, 2021.
11. S. Bogos, F. Tramer, and S. Vaudenay. On solving lpn using bkw and variants: implementation and analysis. *Cryptography and Communications*, 8(3):331–369, 2016.
12. A. Canteaut, S. Carpov, C. Fontaine, T. Lepoint, M. Naya-Plasencia, P. Paillier, and R. Sirdey. Stream ciphers: A practical solution for efficient homomorphic-ciphertext compression. *Journal of Cryptology*, 31(3):885–916, 2018.
13. C. Cid, D. Elkouss, and M. Goulão. Superposition Attacks Against LPN-Based Authentication Protocols. *Cryptology ePrint Archive*, 2025.
14. O. Cosseron, C. Hoffmann, P. Méaux, and F.-X. Standaert. Towards case-optimized hybrid homomorphic encryption: Featuring the elisabeth stream cipher. In *Advances in Cryptology — Proceedings of ASIACRYPT 2022*, volume 13793 of *Lecture Notes in Computer Science*, pages 32–67. Springer, 2022.
15. D. Cox, J. Little, and D. O'shea. *Ideals, Varieties, and Algorithms*. Springer, 4th edition, 1992.

16. D. Dachman-Soled, H. Gong, H. Kippen, and A. Shahverdi. BKW meets Fourier new algorithms for LPN with sparse parities. In *Proceedings of Theory of Cryptography Conference (TCC) 2021*, volume 13043 of *Lecture Notes in Computer Science*, pages 658–688. Springer, 2021.
17. C. Dobraunig, M. Eichlseder, L. Grassi, V. Lallemand, G. Leander, E. List, F. Mendel, and C. Rechberger. Rasta: a cipher with low AND-depth and few ANDs per bit. In *Advances in Cryptology — Proceedings of CRYPTO 2018*, volume 10991 of *Lecture Notes in Computer Science*, pages 662–692. Springer, 2018.
18. C. Dobraunig, L. Grassi, L. Helminger, C. Rechberger, M. Schofnegger, and R. Walch. Pasta: A case for hybrid homomorphic encryption. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2023(3):30–73, 2023.
19. A. Esser, R. Kübler, and A. May. LPN decoded. In *Advances in Cryptology — Proceedings of CRYPTO 2017*, volume 10402 of *Lecture Notes in Computer Science*, pages 486–514. Springer, 2017.
20. J.-C. Faugère, M. Safey El Din, and P.-J. Spaenlehauer. Gröbner bases of bihomogeneous ideals generated by polynomials of bidegree (1,1): Algorithms and complexity. *Journal of Symbolic Computation*, 46(4):406–437, 2011.
21. J.-C. Faugére. A new efficient algorithm for computing Gröbner bases (F_4). *Journal of pure and applied algebra*, 139(1-3):61–88, 1999.
22. J.-C. Faugére and C. Mou. Sparse FGLM algorithms. *Journal of Symbolic Computation*, 80:538–569, 2017.
23. P.-A. Fouque, B. Hadjibeyli, and P. Kirchner. Homomorphic evaluation of lattice-based symmetric encryption schemes. In *Proceedings of Computing and Combinatorics Conference (COCOON) 2016*, volume 9797 of *Lecture Notes in Computer Science*, pages 269–280. Springer, 2016.
24. P.-A. Fouque, B. Hadjibeyli, and P. Kirchner. Homomorphic Evaluation of Lattice-Based Symmetric Encryption Schemes. Cryptology ePrint Archive, Paper 2018/653, 2018.
25. G. Fusco and E. Bach. Phase transition of multivariate polynomial systems. *Mathematical Structures in Computer Science*, 19(1):9–23, 2009.
26. S. D. Galbraith, J. Laity, and B. Shani. Finding Significant Fourier Coefficients: Clarifications, Simplifications, Applications and Limitations. *Chicago Journal of Theoretical Computer Science*, 2018, 2018.
27. C. Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford university, 2009.
28. H. Gilbert, M. J. Robshaw, and Y. Seurin. $HB^{\#}$: Increasing the Security and Efficiency of HB^{+}. In *Advances in Cryptology — Proceedings of EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 361–378. Springer, 2008.
29. H. Gilbert, M. J. Robshaw, and Y. Seurin. How to encrypt with the LPN problem. In *Proceedings of International Colloquium on Automata, Languages and Programming (ICALP) 2008*, volume 5126 of *Lecture Notes in Computer Science*, pages 679–690. Springer, 2008.
30. L. Grassi, F. Liu, C. Rechberger, F. Schmid, R. Walch, and Q. Wang. Minimize the Randomness in Rasta-Like Designs: How Far Can We Go? Cryptology ePrint Archive, Paper 2024/791, 2024.
31. Q. Guo, T. Johansson, and C. Löndahl. Solving LPN using covering codes. In *Advances in Cryptology — Proceedings of ASIACRYPT 2014*, volume 8873 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2014.

32. Q. Guo, T. Johansson, and C. Löndahl. Solving LPN using covering codes. *Journal of Cryptology*, 33(1):1–33, 2020.
33. J. Ha, S. Kim, W. Choi, J. Lee, D. Moon, H. Yoon, and J. Cho. Masta: an HE-friendly cipher using modular arithmetic. *IEEE Access*, 8:194741–194751, 2020.
34. J. Ha, S. Kim, B. Lee, J. Lee, and M. Son. Rubato: Noisy Ciphers for Approximate Homomorphic Encryption. In *Advances in Cryptology — Proceedings of EUROCRYPT 2022*, Lecture Notes in Computer Science, pages 581–610. Springer, 2022.
35. G. Hetyei. Central Delannoy numbers and balanced Cohen-Macaulay complexes. *Annals of Combinatorics*, 10(4):443–462, 2006.
36. N. J. Hopper and M. Blum. Secure Human Identification Protocols. In *Advances in Cryptology — Proceedings of ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 52–66. Springer, 2001.
37. A. Joux and V. Vitse. A crossbred algorithm for solving Boolean polynomial systems. In *Proceedings of Number-Theoretic Methods in Cryptology (NuTMiC) 2017*, volume 10737 of *Lecture Notes in Computer Science*, pages 3–21. Springer, 2017.
38. A. Juels and S. A. Weis. Authenticating Pervasive Devices with Human Protocols. In *Advances in Cryptology — Proceedings of CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 293–308. Springer, 2005.
39. J. Katz, J. S. Shin, and A. Smith. Parallel and concurrent security of the hb and hb+ protocols. *Journal of cryptology*, 23(3):402–421, 2010.
40. P. Kirchner. Improved generalized birthday attack. *Cryptology ePrint Archive*, 2011.
41. P. J. Lee and E. F. Brickell. An observation on the security of McEliece's public-key cryptosystem. In *Advances in Cryptology – EUROCRYPT '88*, volume 330 of *Lecture Notes in Computer Science*, pages 275–280. Springer, 1988.
42. É. Levieil and P.-A. Fouque. An improved LPN algorithm. In *Proceedings of Security and Cryptography for Networks (SCN) 2006*, volume 4116 of *Lecture Notes in Computer Science*, pages 348–359. Springer, 2006.
43. F. Liu, M. Mahzoun, M. Øygarden, and W. Meier. Algebraic Attacks on RAIN and AIM Using Equivalent Representations. *IACR Transactions on Symmetric Cryptology*, 2023(4):166–186, 2023.
44. A. May, A. Meurer, and E. Thomae. Decoding random linear codes in $\tilde{O}(2^{0.054n})$. In *Advances in Cryptology – ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 107–124. Springer, 2011.
45. P. Méaux, C. Carlet, A. Journault, and F.-X. Standaert. Improved filter permutators for efficient FHE: better instances and implementations. In *Proceedings of INDOCRYPT 2019*, volume 11898 of *Lecture Notes in Computer Science*, pages 68–91. Springer, 2019.
46. P. Méaux, A. Journault, F.-X. Standaert, and C. Carlet. Towards stream ciphers for efficient FHE with low-noise ciphertexts. In *Advances in Cryptology — Proceedings of EUROCRYPT 2016*, volume 9665 of *Lecture Notes in Computer Science*, pages 311–343. Springer, 2016.
47. M. Naehrig, K. Lauter, and V. Vaikuntanathan. Can homomorphic encryption be practical? In *Proceedings of the 3rd ACM Cloud Computing Security Workshop (CCSW)*, pages 113–124, 2011.
48. K. Pietrzak. Cryptography from learning parity with noise. In *Proceedings of SOFSEM 2012: Theory and Practice of Computer Science*, volume 7147 of *Lecture Notes in Computer Science*, pages 99–114. Springer, 2012.

49. M. Poppe. An Algebraic Cryptanalysis of the Atrapos Permutations. Bachelor's thesis, Radboud University Nijmegen, 2025. `https://mdjp.eu/bsc-thesis-atrapos`.
50. E. Prange. The use of information sets in decoding cyclic codes. *IRE Transactions on Information Theory*, 8(5):5–9, 1962.
51. R. L. Rivest, L. Adleman, M. L. Dertouzos, et al. On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180, 1978.
52. M. J. Steiner. The Complexity of Algebraic Algorithms for LWE. In *Advances in Cryptology —Proceedings of EUROCRYPT 2024*, volume 14653 of *Lecture Notes in Computer Science*, pages 375–403. Springer, 2024.
53. V. Strassen. Gaussian elimination is not Optimal. *Numerische mathematik*, 13(4):354–356, 1969.
54. B. Tran and S. Vaudenay. Solving the learning parity with noise problem using quantum algorithms. In *Progress in Cryptology – AFRICACRYPT 2022*, volume 13503 of *Lecture Notes in Computer Science*, pages 295–322. Springer, 2022.
55. B. Tran and S. Vaudenay. Solving the Learning Parity with Noise Problem Using Quantum Algorithms. In *Proceedings of AFRICACRYPT 2022*, volume 13503 of *Lecture Notes in Computer Science*, pages 295–322. Springer, 2022.
56. B. Zhang, L. Jiao, and M. Wang. Faster algorithms for solving LPN. In *Advances in Cryptology — Proceedings of EUROCRYPT 2016*, volume 9665 of *Lecture Notes in Computer Science*, pages 168–195. Springer, 2016.

A LPN Solvers

We briefly summarize the main solver families that commonly appear as the final step of such reduction chains.

BKW-style algorithms. The seminal work of Blum, Kalai, and Wasserman introduced the BKW algorithm [9], an iterative collision-based procedure that reduces the effective dimension by canceling blocks of secret bits at the cost of amplifying the noise. Concretely, BKW partitions the k secret bits into blocks and repeatedly combines samples so as to eliminate entire blocks, yielding after sufficiently many rounds a reduced LPN instance. The remaining secret can then be recovered by a final step, typically using correlation-based techniques (e.g., Walsh–Hadamard methods) or, if the reduced dimension is small, by brute-force enumeration with an appropriate scoring test.

Over time, BKW has been refined in several directions. Levieil and Fouque [42] optimized the final recovery using fast Walsh–Hadamard/Fourier techniques. Kirchner [40] suggested transforming the problem into a systematic form where each secret bit appears as an observed symbol perturbed by noise. Bernstein and Lange [8] studied the Ring-LPN variant and observed that exploiting additional structure can further reduce the cost of attacks; their techniques also apply to standard LPN instances via reductions. More recently, Guo et al. [31] introduced coded-BKW, which employs covering codes to improve the dimension reduction at each step, yielding more efficient attacks. Zhang et al. [56] extended this approach by embedding cascaded perfect codes into the BKW procedure, achieving improved attack complexities across a wide range of parameters, directly impacting parameter recommendations for schemes such as LPN-C. Finally, BKW*-type optimizations streamline reduction-chain choices and parameter search, yielding tighter analyses and improved practical performance [11].

Linear-system solving via Gaussian elimination. A generic approach for attacking an LPN instance is to treat it as a noisy linear system and attempt to derive the secret from an invertible subsystem of k equations. Concretely, select an index set $I \subseteq [\mathsf{Q}]$ of size k such that the row-restricted matrix $A_I \in \mathbb{F}_2^{k\times k}$ is invertible, and let $\boldsymbol{b}_I \in \mathbb{F}_2^k$ be the associated oracle bits. In the noiseless case, the unique solution is $\boldsymbol{s} = A_I^{-1}\boldsymbol{b}_I$. In the noisy setting, Gaussian elimination yields a candidate $\hat{\boldsymbol{s}} = A_I^{-1}\boldsymbol{b}_I$, which is then validated on additional samples via a hypothesis test: one distinguishes Ber_τ-noise from uniform by checking whether the residual $A\hat{\boldsymbol{s}} \oplus \boldsymbol{b}$ has unusually small Hamming weight.

Information-Set Decoding (ISD). An *information set* is an index set $I \subseteq [\mathsf{Q}]$ of size k such that the corresponding row-restricted matrix A_I has full rank. Intuitively, these k samples contain enough independent linear information to uniquely determine the secret in the absence of noise. Information-set decoding (ISD), originating from Prange's seminal idea [50], is a *search-and-decode* framework that repeats: sample (or construct) an information set, perform an inner decoding step, and validate the resulting candidate secret on fresh samples. In

the LPN setting, the inner decoding step is often Gaussian elimination, possibly augmented with limited error enumeration/correction to boost success probability per trial. More advanced ISD variants enumerate structured sets of candidates or partial error patterns (e.g., Lee–Brickell [41] and MMT-style [44] techniques), trading extra computation and memory for fewer trials. These techniques do not exploit the full structure of LPN, but can be relevant in very small-noise regimes where elimination-based decoding is competitive.

Fourier-analytic solvers (WHT, SFT). A different family of solvers exploits correlations induced by the secret. For a transcript $(A, \boldsymbol{b})$ and $y \in \mathbb{F}_2^k$, define the correlation score

$$W(y) \;=\; \sum_{i=1}^{\mathsf{Q}} (-1)^{\langle a_i, y\rangle + b_i} \;=\; \mathsf{Q} - 2 \cdot \mathrm{HW}(Ay \oplus \boldsymbol{b}).$$

In regimes where the coefficient corresponding to $y = \boldsymbol{s}$ dominates (e.g., after sufficient dimension reduction and for large enough Q), spectral methods aim to output $\arg\max_y |W(y)|$. The Walsh–Hadamard transform (WHT) computes the full spectrum and returns the maximizer. Significant Fourier Transform (SFT) methods instead search directly for heavy Fourier coefficients without evaluating the entire spectrum. Conceptually, WHT and SFT rely on the same correlation phenomenon; the difference is algorithmic: WHT evaluates all coefficients, whereas SFT performs an adaptive search to locate the significant coefficient corresponding to the secret.

B Sample Requirements for Iterated Key Recovery

Suppose we have Q error vectors $\boldsymbol{\nu}^{(\ell)} = (\nu_1^{(\ell)}, \ldots, \nu_n^{(\ell)})$ satisfying $\mathrm{HW}(\boldsymbol{\nu}^{(\ell)}) \le t$, and let $I_1 = \{1, \ldots, Q\}$ as in Equation 5. By Proposition 2, iteration i of the cascading attack recovers column S_i by solving an $\mathrm{LPN}_{k,\tau_{\mathrm{eff}}^{(i)}}$ instance on the sample subset I_i, after which samples with $\nu_i^{(\ell)} = 0$ are discarded to form I_{i+1}. Since $\nu_i^{(\ell)}$ is distributed according to $\mathrm{Ber}_{\tau_{\mathrm{eff}}^{(i)}}$ conditioned on membership in I_i, the expected size of I_{i+1} is

$$\mathbb{E}\left(|I_{i+1}|\right) = |I_i| \cdot \tau_{\mathrm{eff}}^{(i)}\,.$$

Unrolling the recursion from $|I_1| = \mathsf{Q}$ gives

$$\mathbb{E}\left(|I_{i+1}|\right) = \mathsf{Q} \cdot \prod_{j=1}^{i} \tau_{\mathrm{eff}}^{(j)}\,.$$

Hence, the number of usable samples decreases iteratively in each round, with the rate of reduction governed by the sequence of parameters $\tau_{\mathrm{eff}}^{(i)}$.

For the attack to proceed past iteration i (i.e. to have samples remaining for iteration $i+1$, or for Gaussian elimination if $i = t$), we need $\mathbb{E}\left(|I_{i+1}|\right) \ge 1$, which requires

$$\mathsf{Q} \geq \prod_{j=1}^{i} \frac{1}{\tau_{\text{eff}}^{(j)}} \quad \Longleftrightarrow \quad \log_2 \mathsf{Q} \geq -\sum_{j=1}^{i} \log_2 \tau_{\text{eff}}^{(j)} \,.$$

In particular, to carry the attack through all t iterations (after which the remaining columns are recoverable by Gaussian elimination), the attacker needs at least

$$\log_2 \mathsf{Q} \geq -\sum_{j=1}^{t} \log_2 \tau_{\text{eff}}^{(j)}$$

samples (although practical attacks may require significantly higher sample numbers).

Table 6 lists the cumulative query requirement $\log_2 \mathsf{Q}$ after each iteration i for each of the three parameter sets. We stress that even with limited samples, significant speed-ups may still be accessible by carrying out the attack through fewer than t iterations.

C Pseudocode for Iterated Key Recovery

Let $\mathcal{A}$ be a generic LPN solver (e.g., a BKW-variant or an ISD-inspired method). At column i of the LPN-C secret matrix, we invoke $\mathcal{A}$ with noise parameter $\tau_{\text{eff}}^{(i)}(\tau, n, t)$ on the filtered sample set I_i; the per-column cost is therefore the cost of solving $\text{LPN}_{k,\tau_{\text{eff}}^{(i)}}$ under the available sample budget.

As explained in subsection 3.2, this yields a natural column-wise key recovery strategy, where each column i (for $1 \leq i \leq t$) is attacked as an independent LPN instance with its own effective noise parameter $\tau_{\text{eff}}^{(i)}(\tau, n, t)$, each smaller than the last. Once the t-th column has been recovered, the remaining columns are recovered by Gaussian elimination. The procedure is summarized below.

Algorithm 1 Sequential Full-Key Recovery for LPN-C

Require: Instance parameters (k, τ, n, t); Q accepted samples $(\boldsymbol{a}^{(\ell)}, \boldsymbol{y}^{(\ell)})$.
Ensure: Recovered secret matrix $S = [S_1 \ \ldots \ S_n]$.

Initialize $I_0 \leftarrow \{1, \ldots, \mathsf{Q}\}$ ▷ indices of accepted samples
for i **from** 1 **to** t **do**
  if $1 < i$ **then**
    $I_i \leftarrow \{\ell \in I_{i-1} \mid \nu_{i-1}^{(\ell)} = 1\}$ ▷ discard samples with $\nu_{i-1}^{(\ell)} = 0$
  else
    $I_i \leftarrow I_{i-1}$ ▷ no change; keep current active set
  end if
  Form LPN samples $\{(\boldsymbol{a}^{(\ell)}, b_i^{(\ell)})\}_{\ell \in I_i}$, where $b_i^{(\ell)} = \langle \boldsymbol{a}^{(\ell)}, S_i \rangle \oplus \nu_i^{(\ell)}$
  Solve: $S_i \leftarrow \mathcal{A}\Big(k, \tau_{\text{eff}}^{(i)}, \{(\boldsymbol{a}^{(\ell)}, b_i^{(\ell)})\}_{\ell \in I_i}\Big)$
  Compute $\nu_i^{(\ell)}$ for all $\ell \in I_i$ ▷ needed for next round
end for
Use Gaussian elimination on the remaining linear equations to recover the rest of S.
return S

D Further Discussion on section 4

D.1 LPN-C with Secret Toeplitz Matrix

We now expand on subsection 4.3. In our attack, Toeplitz structure can only help if it creates many additional collisions when we collect terms by identical a-monomials; such collisions would reduce the number of distinct grouped coefficients, and hence the effective dimension of the resulting linear system. We did not observe this effect: for the low degrees relevant to our attack, collisions appear negligible, so the equation count, and thus the complexity, remains essentially unchanged.

We consider the Toeplitz variant suggested in HB# and LPN-C [28,29], where the secret $S = (s_{ij})$ satisfies $s_{ij} = t_{i-j}$ and can be stored using $n + k - 1$ bits instead of nk. In our linearization, after grouping by identical a-monomials, the resulting coefficient for a given group takes the form

$$\sum_{\pi \in Perm(r)} \prod_{l=1}^{d} t_{j_l - i_{\pi(l)}},$$

so only the multiset of differences $(j_l - i_{\pi(l)})_{l=1}^{d}$ matters. The attack would speed up only if many such grouped sums *collide*. We do not know a closed form for counting these collisions; empirically (and for low degrees where it could matter) they are negligible.

Toy example. With a 3×2 Toeplitz slice

$$S = \begin{bmatrix} s_{11} & s_{12} \\ s_{21} & s_{22} \\ s_{31} & s_{32} \end{bmatrix} = \begin{bmatrix} t_0 & t_{-1} \\ t_1 & t_0 \\ t_2 & t_1 \end{bmatrix}, \qquad f_1 = \sum_i a_i s_{i1}, \quad f_2 = \sum_i a_i s_{i2},$$

the degree-2 grouping gives

$$f_1 f_2 = \underbrace{a_1 a_2}_{\text{same } a\text{-mono}} (t_0^2 + t_{-1} t_1) + \underbrace{a_1 a_3}_{\text{same } a\text{-mono}} (t_0 t_1 + t_{-1} t_2) + \cdots$$

The a-monomials coincide (e.g., $a_1 a_2$), but their associated sums in the t_ℓ almost always remain distinct and for larger values of k, n there is no closed form formula to compute how often they coincide. Thus, the number of effective linear equations—and hence the attack complexity—remains essentially unchanged compared to a random S (up to small constants).

D.2 On the (In)Applicability of Gröbner Basis Methods for LPN-C

In this appendix we briefly discuss why Gröbner basis methods are unlikely to improve the attack in section 4. We will assume that the reader is familiar with some of the fundamental concepts and underlying theory, which can be found in [15].

Gröbner Bases Versus Linearization. State-of-the-art Gröbner basis algorithms, such as F_4 [21], perform linear algebra operations on S-polynomials of increasing degrees. As our polynomial system contains equations of degree $t+1$ (i.e., of the form of Equation 10), our Gröbner basis algorithm will, as a minimum, work with a linearized system at degree $t+1$. This linear system will have dimension $D_{\mathcal{S}} = \dim(\mathcal{S})$ (as bounded in Proposition 3), and we can therefore expect this approach to be lower bounded by $\mathcal{O}(D_{\mathcal{S}}^{\omega})$, which is the same complexity we expected in subsection 4.1 when the attacker has access to $\approx D_{\mathcal{S}}$ plaintext-ciphertext pairs. In particular, we do not expect Gröbner basis techniques to outperform linearization in this scenario.

As such, Gröbner basis methods only seem viable in cases where the amount of plaintext-ciphertext pairs are limited. Indeed, if this number is smaller than $D_{\mathcal{S}}$, but larger than $k(t+1)$, it will not be possible to directly linearize the system, but we can still expect that a unique solution can be found by solving at a larger degree, at a greater computation cost. This can thus be viewed as a time/data tradeoff. A special variation of this is the probabilistic variant when the attacker considers equations as in Equation 10, but of smaller degrees than $t+1$. (This does not seem to yield an efficient strategy with the relatively large error rates τ used for LPN-C, although it is more viable in the setting of section 5.)

Solving at Higher Degrees. Let us then briefly consider the scenario where the number of equations available to an attacker is smaller than $D_{\mathcal{S}}$. In this case the attacker will have to construct equations of degree $> t+1$, and we do not see a way to preserve the concise vector space structure of $(\mathcal{S})$. Instead, the polynomial system should be viewed as a system of affine multilinear polynomials in $t+1$ sets of k variables[7].

We are not aware of good estimates for solving these kinds of structured polynomial systems. The works that do exist on solving *generic multilinear systems*, such as [7,20], would imply that they will be easier to solve than general estimates applied to systems of degree $t+1$ in $k(t+1)$ variables would suggest [5]. A more dedicated analysis of these systems is beyond the scope of this paper, and we will be content with the above conclusion that the (time) complexity will exceed that of subsection 4.1.

E Additional Complexity Tables for Iterated Key Recovery

Table 9–Table 10 show the estimated per-iteration time complexities for iterative key recovery on the first three parameter sets of Table 2, when using LF2 [11] and quantum ISD [54] as the LPN solver, respectively. Table 11 shows the expected sample requirements for carrying out the attack through i iterations for all five sets.

[7] An exception here is when the secret matrix has a Toeplitz form, as in Appendix D.1. We view this as out of scope for this discussion.

Table 9: Per-iteration LF2 attack complexity $\log_2 \mathsf{T}_i$ for each LPN-C parameter set, with optimized parameters (a, b) from [11]. The first row of each set shows the complexity using the original noise rate τ; subsequent rows use the iterated effective noise rates from Proposition 2.

Set 1 (k=512, n=80, t=10)				Set 2 (k=512, n=160, t=20)								Set 3 (k=768, n=80, t=4)			
i	(a,b)	$\tau_{\text{eff}}^{(i)}$	$\log_2 \mathsf{T}_i$	i	(a,b)	$\tau_{\text{eff}}^{(i)}$	$\log_2 \mathsf{T}_i$	i	(a,b)	$\tau_{\text{eff}}^{(i)}$	$\log_2 \mathsf{T}_i$	i	(a,b)	$\tau_{\text{eff}}^{(i)}$	$\log_2 \mathsf{T}_i$
–	$(7,74)$	0.125	87.40	–			87.40					–	$(9,86)$	0.05	100.35
1	$(7,74)$	0.0999	87.40	1	$(7,74)$	0.1064	87.40	11	$(8,64)$	0.0617	77.59	1	$(9,86)$	0.0349	100.35
2	$(7,74)$	0.0932	87.40	2	$(7,74)$	0.1030	87.40	12	$(8,64)$	0.0562	77.59	2	$(9,86)$	0.0280	100.35
3	$(7,74)$	0.0857	87.40	3	$(7,74)$	0.0994	87.40	13	$(8,64)$	0.0505	77.59	3	$(10,77)$	0.0198	91.50
4	$(8,70)$	0.0773	83.59	4	$(7,74)$	0.0954	87.40	14	$(8,64)$	0.0446	77.59	4	$(11,70)$	0.0104	84.63
5	$(8,64)$	0.0682	77.59	5	$(7,74)$	0.0913	87.40	15	$(8,64)$	0.0386	77.59				
6	$(8,64)$	0.0584	77.59	6	$(7,74)$	0.0868	87.40	16	$(9,57)$	0.0325	70.76				
7	$(8,64)$	0.0479	77.59	7	$(7,74)$	0.0822	87.40	17	$(9,57)$	0.0263	70.76				
8	$(8,64)$	0.0368	77.59	8	$(8,70)$	0.0773	83.59	18	$(9,57)$	0.0199	70.76				
9	$(9,57)$	0.0251	70.76	9	$(8,65)$	0.0723	78.59	19	$(10,52)$	0.0134	65.91				
10	$(10,52)$	0.0128	65.91	10	$(8,64)$	0.0671	76.34	20	$(11,47)$	0.0068	61.05				
Total			**89.0**	**Total**							**90.2**	**Total**			**101.3**
$n \cdot \mathsf{T}_1$			93.7	$n \cdot \mathsf{T}_1$							94.7	$n \cdot \mathsf{T}_1$			106.7

Table 10: Per-iteration quantum ISD (Q-ISD) attack complexity $\log_2 \mathsf{T}_i$ for each LPN-C parameter set, using the iterated effective noise rates from Proposition 2.

Set 1 (k=512, n=80, t=10)			Set 2 (k=512, n=160, t=20)						Set 3 (k=768, n=80, t=4)		
i	$\tau_{\text{eff}}^{(i)}$	$\log_2 \mathsf{T}_i$	i	$\tau_{\text{eff}}^{(i)}$	$\log_2 \mathsf{T}_i$	i	$\tau_{\text{eff}}^{(i)}$	$\log_2 \mathsf{T}_i$	i	$\tau_{\text{eff}}^{(i)}$	$\log_2 \mathsf{T}_i$
1	0.0999	66.43	1	0.1064	69.10	11	0.0617	51.21	1	0.0349	49.09
2	0.0932	63.70	2	0.1030	67.70	12	0.0562	49.09	2	0.0280	45.26
3	0.0857	60.67	3	0.0994	66.22	13	0.0505	46.91	3	0.0198	40.80
4	0.0773	57.32	4	0.0954	64.59	14	0.0446	44.68	4	0.0104	35.49
5	0.0682	53.74	5	0.0913	62.93	15	0.0386	42.44			
6	0.0584	49.93	6	0.0868	61.11	16	0.0325	40.19			
7	0.0479	45.92	7	0.0822	59.27	17	0.0263	37.89			
8	0.0368	41.77	8	0.0773	57.32	18	0.0199	35.45			
9	0.0251	37.44	9	0.0723	55.35	19	0.0134	32.99			
10	0.0128	32.77	10	0.0671	53.31	20	0.0068	30.51			
Total		**66.7**	**Total**					**69.8**	**Total**		**49.2**
$n \cdot \mathsf{T}_1$		72.8	$n \cdot \mathsf{T}_1$					76.4	$n \cdot \mathsf{T}_1$		55.4

Table 11: Lower bounds on the number of queries $\log_2 \mathsf{Q}$ needed for $\mathbb{E}(|I_{i+1}|) \geq 1$ after each iteration i of the cascading attack on the LPN-C parameter sets of Table 2. The last row of each set gives the minimum query requirement to carry the attack through all t iterations (highlighted in bold).

Parameter Set 1		Parameter Set 2				Parameter Set 3		Parameter Set 4		Parameter Set 5	
i	$\log_2 \mathsf{Q}$	i	$\log_2 \mathsf{Q}$	i	$\log_2 \mathsf{Q}$	i	$\log_2 \mathsf{Q}$	i	$\log_2 \mathsf{Q}$	i	$\log_2 \mathsf{Q}$
1	3.3	1	3.2	11	39.2	1	4.8	1	4.7	1	4.4
2	6.7	2	6.5	12	43.4	2	10.0	2	9.5	2	8.8
3	10.3	3	9.8	13	47.7	3	15.7	3	14.5	3	13.3
4	14.0	4	13.2	14	52.2	4	**22.2**	4	19.8	4	17.9
5	17.9	5	16.7	15	56.9			5	25.3	5	22.6
6	22.0	6	20.2	16	61.8			6	31.2	6	27.4
7	26.3	7	23.8	17	67.1			7	37.6	7	32.4
8	31.1	8	27.5	18	72.7			8	**45.1**	8	37.6
9	36.4	9	31.3	19	78.9			9	43.0		
10	**42.7**	10	35.2	20	**86.1**			10	48.9		
								11	55.3		
								12	**62.7**		