Stationary Syndrome Decoding for Improved PCGs

Vladimir Kolesnikov[1], Stanislav Peceny[1],
Srinivasan Raghuraman[2], and Peter Rindal[3]

[1] Georgia Institute of Technology
[2] Visa Research and MIT
[3] Visa Research

Abstract. Syndrome decoding (SD), and equivalently Learning Parity with Noise (LPN), is a fundamental problem in cryptography, which states that for a field $\mathbb{F}$, some compressing public matrix $\mathbf{G} \in \mathbb{F}^{k \times n}$, and a secret sparse vector $\boldsymbol{e} \in \mathbb{F}^n$ sampled from some *noise* distribution, $\mathbf{G}\boldsymbol{e}$ is indistinguishable from uniform. Recently, the SD has gained significant interest due to its use in pseudorandom correlation generators (PCGs).

In pursuit of better efficiency, we propose a new assumption called *Stationary Syndrome Decoding* (SSD). In SSD, we consider q correlated noise vectors $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_q \in \mathbb{F}^n$ and associated instances $\mathbf{G}_1\boldsymbol{e}_1, \ldots, \mathbf{G}_q\boldsymbol{e}_q$ where the noise vectors are restricted to having non-zeros in the same small subset of t positions $L \subset [n]$. That is, for all $i \in L$, $\boldsymbol{e}_{j,i}$ is uniformly random, while for all other i, $\boldsymbol{e}_{j,i} = 0$.

Although naively reusing the noise vector renders SD and LPN insecure via simple Gaussian elimination, we observe known attacks do not extend to our correlated noise. We show SSD is unconditionally secure against so-called linear attacks, e.g., advanced information set decoding and representation techniques (Esser and Santini, Crypto 2024). We further adapt the state-of-the-art nonlinear attack (Briaud and Øygarden, Eurocrypt 2023) to SSD and demonstrate both theoretically and experimentally resistance to the attack.

We apply SSD to PCGs to amortize the cost of noise generation protocol. For OT and VOLE generation, each instance requires $O(t)$ communication instead of $O(t \log n)$. For suggested parameters, we observe a 1.5× improvement in the running time or between 6 and 18× reduction in communication. For Beaver triple generation using Ring LPN, our techniques have the potential for substantial amortization due to the high concrete overhead of the Ring LPN noise generation.

1 Introduction

Syndrome Decoding (SD), or equivalently Learning Parity with Noise (LPN), are standard assumptions in code-based cryptography. SD states that for some public matrix $\mathbf{G} \in \mathbb{F}^{k \times n}$, where $k < n$, and a secret sparse vector $\boldsymbol{e} \in \mathbb{F}^n$ with a Hamming weight $|\boldsymbol{e}| \approx t$, $\mathbf{G}\boldsymbol{e}$ is pseudorandom. Recently, SD has seen increased interest due to its applications in secure multi-party computation (MPC)

[RS21,DILO22,ANO+22,AS22,BDSW23], zero knowledge [YSWW21,WYKW21] [WYY+22,BDSW23], and post-quantum signatures [FJR22,CCJ23].

MPC enables parties that do not trust each other to compute on their private data without disclosing any information to the other parties. MPC has gained relevance in both academia and industry (e.g., for online auctions, electronic voting, privacy-preserving machine learning); its potential remains largely untapped due to the significantly higher costs compared to plaintext computation.

Most efficient MPC protocols work in the preprocessing model, where parties first preprocess cryptographic material that is independent of both the function and its inputs. This preprocessing phase typically involves generating random instances of oblivious transfers (OTs), oblivious linear evaluations (OLE), vector oblivious linear evaluations (VOLE), or Beaver triples. Once the function and inputs are known, the parties compute the function securely using the correlated randomness of the preprocessed material.

Preprocessing is often the computational bottleneck in MPC. A recent line of work on pseudorandom correlation generators (PCGs) shows great promise in significantly reducing preprocessing costs. PCGs offer sublinear communication and compelling computational overheads. While they have led to substantial improvements, PCGs are still far from matching the cost of honest majority or plaintext computation. *In this work, one of our goals is to reduce the cost of MPC by minimizing the costs associated with PCGs.*

State-of-the-art PCG constructions intimately rely on the Syndrome Decoding (SD) assumption. These PCGs make use of this assumption by having the parties interactively generate a secret sharing of the sparse vector $\boldsymbol{e}$ and then locally compute a sharing of $\mathbf{G}\boldsymbol{e}$. This final sharing forms, in part, the preprocessed materials. The computational overhead of this process has two main parts; generating the secret $\boldsymbol{e}$ with sublinear communication and performing the multiplication $\mathbf{G}\boldsymbol{e}$. Consequently, existing works optimize PCGs either by adding structure to $\mathbf{G}$ to accelerate the multiplication or by changing the distribution of $\boldsymbol{e}$ to improve the efficiency of generating a sharing of it. In our work, we take the latter approach. We amortize the cost of computing $\boldsymbol{e}$ across q SD instances.

To achieve this, we propose and cryptanalyze a new assumption called the Stationary Syndrome Decoding (SSD). Intuitively, SSD allows for reusing the non-zero coordinates of a noise vector $\boldsymbol{e}$ across multiple SD instances, provided that the noise at each coordinate is uniformly drawn for each SD instance. For example, consider $\mathbb{F}_7$, $n = 12$, and $t = 4$. The noise vectors could be:

$$\begin{aligned}
\boldsymbol{e}_1 &= \big(0\ 1\ 0\ 3\ 0\ 0\ 0\ 0\ 6\ 3\ 0\ 0\big)\,,\\
\boldsymbol{e}_2 &= \big(0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 2\ 4\ 0\ 0\big)\,,\\
&\ldots,\\
\boldsymbol{e}_q &= \big(0\ 4\ 0\ 6\ 0\ 0\ 0\ 0\ 0\ 3\ 0\ 0\big)
\end{aligned}$$

We refer to this assumption as stationary because the noise positions remain fixed across instances, e.g. at positions $L = \{2, 4, 9, 10\}$ in the example above. Note that the noisy values of $\boldsymbol{e}_i$ are sampled uniformly from $\mathbb{F}$, *including zero.*

Without delving into details, the stationary nature of the noise positions enables us to reuse the bulk of the work required to generate a sharing of $\boldsymbol{e}$, such as the OTs and the GGM tree expansion. This can indirectly speed up the time to multiply $\mathbf{G}\boldsymbol{e}$ due to better parameter selection.

We believe the resilience of SSD to known LPN/SD attacks to be both unanticipated and significant. While we are not the first to consider the hardness of LPN and SD with structured noise, our assumption arguably has more implications due to the noise being highly correlated. This suggests the intriguing possibility of LPN and SD with other highly correlated noise distributions and the potential impact they can have on various constructions.

1.1 Contribution

Our core contribution is the introduction of the Stationary Syndrome Decoding assumption. First, we provide supporting evidence for its hardness. We show that SSD is resilient against all linear attacks [Pra62,Ste89,Jab01,BKW03,Lyu05] [FKI07,ABG+14,BR17,ES24] and adaptations of the state-of-the-art algebraic attacks of [BØ23].

Then, we show that SSD has significant efficiency implications for pseudorandom correlation generators (PCGs). In particular, we start with the communication and computation implications for degree-1 correlations, such as VOLE, OT, and binary OLE. Next, we discuss the same implications for degree-2 correlations, such as general OLE and Beaver triples. We also show that SSD can result in much better cache/memory utilization of PCGs.

Degree 1 Correlations. Let n be the length and t the Hamming weight of the noise vector $\boldsymbol{e}$. To generate q noise vectors with the SSD assumption, our construction requires a single "base VOLE" of size tq and $t\log_2(n/t)$ base OTs. To generate q noise vectors using the standard SD construction also requires the same size base VOLE correlation, but q times more base OTs. I.e., we can reuse all OTs across the q instances. As a result, our construction requires an amortized $2\log_2(n/t)$ times less data to be sent when generating the sharing of $\boldsymbol{e}$. Our experiments show $6.4-7.5\times$ reduction in communication for OT and $10.7-17.6\times$ for VOLE, for a standard choice of parameters.

Degree 2 Correlations. The setup for degree 2 correlations from Ring-LPN [BCG+20], e.g. Beaver triples over $\mathbb{F}_p$, is significantly more complex and requires the generation of a weight $t' := t^2$ noise vector (product of two t-sparse vectors), in contrast to the simpler degree 1 setup, which only requires a weight t noise vector. Moreover, the natural implementation requires $O(nt)$ local work instead of $O(n)$ in the degree 1 case. Some works starting with [SGRR19,BCG+20] suggested the use of so called batch codes, such as cuckoo hashing, to reduce this overhead back to $O(n)$ at the expense of more work being performed in MPC, e.g. $O(\mathsf{poly}(t'))$. However, the concrete costs remain high. We suggest that SSD can be used, similarly to the degree 1 case, to amortize this expensive setup across the q instances. In particular, the vast majority of the work performed

within the MPC can be performed once and then reused across the q instances. This brings down the amortized overheads from at least $O(t' \log(n/t) + \mathsf{poly}(t'))$ to $O(t')$ communication and from $O(nt)$ to $O(n)$ computation.

Cache and Memory Utilization. To achieve the desired sublinear communication overhead of PCGs, it has been necessary to generate the correlations in large batches, e.g. $n \approx 2^{20}$. However, SSD reduces this necessity as it is more communication efficient, and therefore allows for a smaller n while achieving the same relative communication overhead. This in turn results in non-trivial performance improvements as the overall protocol can better fit into CPU cache. Our implementation shows that by executing $q = 2^5$ instances of size $n = 2^{15}$ binary OLEs rather than a large batch of $n = 2^{20}$, we can reduce wall-clock time for degree 1 correlations by 1.5 times while achieving the same relative communication overhead. Moreover, we expect speedup for degree 2 correlation to be even larger due to the larger amount of local computation along with the more complex setup protocol.

2 Related Work

In Section 3, we review existing works on syndrome decoding (SD) and related attacks. In this section, however, we focus on approaches aimed at improving pseudorandom correlation generators (PCGs), which serve as the motivating application of SD in our work. The majority of PCGs use essentially the same protocols [BCGI18,BCG+19b,BCG+19a,BCG+20,CRR21,BCG+22,RRT23]. They generate a secret sharing of a sparse vector $\boldsymbol{e}$ times either a scalar Δ or in [BCG+20] by another sparse vector. This secret sharing is then compressed by a linear function $\mathbf{G}$. Thus, to improve PCG performance the focus has been to improve the generation of the shared sparse vector and the time required to multiply by $\mathbf{G}$. All prior works have focused on improving the generation of a *single* SD instance.

Thus far, the most impactful optimization for generating the secret sharing of $\boldsymbol{e}$ [AFS05,HOSS18,Ds17,BCGI18,SGRR19,BCG+19b,BCG+20,BCG+22] has been the idea of the regular noise distribution [AFS05] that replaces the Bernoulli/exact distributions (see Section 3.4). In the context of secure computation, this optimization was first used by TinyKeys [HOSS18] and later by [BCGI18] for PCGs. The regular noise distribution is so beneficial because it allows the generation of the sharing of $\Delta\boldsymbol{e}$ to consist of $O(n)$ AES calls across t distributed point functions. This is in contrast with other noise distributions that require $O(t)$ distributed point functions and as much as $O(tn)$ AES calls, or the use of more complicated batch codes such as cuckoo hashing [SGRR19].

Another line of work tries to optimize the time it takes to multiply matrix $\mathbf{G}$ and $\boldsymbol{e}$ [BCGI18,BCG+19b,BCG+19a,BCG+20,CRR21,BCG+22,RRT23]. The original LPN assumption states that $\mathbf{G}$ is uniform, and therefore $\mathbf{G} \cdot \boldsymbol{e}$ requires $O(n^2)$ time. However, it is widely accepted that $\mathbf{G}$ can be replaced by the generator matrix of a code with high minimum distance, unless that code has strongly

algebraic structure, such as the Reed-Solomon code, which can be efficiently decoded with the Berlekamp-Massey algorithm [Ber68]. By changing $\mathbf{G}$ to be a generator matrix, it is theoretically possible to compute $\mathbf{G} \cdot \boldsymbol{v}$ for any $\boldsymbol{v}$ in $O(n)$ time. However, practical considerations typically result in time $O(nc)$ where c is somehow related to $\log n$ or the security parameter. [BCGI18,BCG+19b,BCG+19a] propose to rely on LDPC or quasi-cyclic codes as their security is well-studied and they offer reasonable performance. [BCG+22] propose using a type of turbo code that they call expand-accumulate, which not only significantly improves costs but also provably has high minimum distance. [RRT23] further improve on this code by replacing accumulation with a so-called convolution, which significantly increases minimum distance, and hence allows for more favorable parameters and performance. Our work differs from all these works in that we amortize PCG cost across *multiple* SD instances. I.e., we amortize out the cost of generating the noise vectors $\boldsymbol{e}$ across q instances.

3 Preliminaries

3.1 Notation

Most of our notation follows standard conventions. Specifically, matrices are denoted using bold, non-italic, uppercase letters, while vectors are represented by bold, italic, lowercase letters. E.g., $\mathbf{G}$ is a matrix and $\boldsymbol{v}$ is a vector. We index matrices and vectors with subscript and use 1-based indexing, e.g., $\mathbf{M}_{i,j}$ or $\boldsymbol{v}_i$. $|\boldsymbol{v}|$ represents the Hamming weight of a vector. $||\boldsymbol{v}||$ represents the length of a vector or size of a set. $\boldsymbol{v}||\boldsymbol{u}$ denotes the concatenation of vectors $\boldsymbol{v}, \boldsymbol{u}$. $\boldsymbol{v} \odot \boldsymbol{u}$ and $\boldsymbol{v} \cdot \boldsymbol{u}$ denote the component-wise and dot product multiplications, respectively. $[a, b]$ denotes the sequence of natural numbers $a, \ldots, b$ and $[n]$ the sequence $[1, n]$. $[a, b]_{\mathbb{R}}$ denotes the inclusive range from a to b over the real numbers. κ is the computational security parameter. $[\![x]\!]$ denotes a two-out-of-two secret sharing of $x \in \mathbb{F}$. A sender party holds a share $[\![x]\!]_{\mathsf{s}} \in \mathbb{F}$, while the receiver party holds $[\![x]\!]_{\mathsf{r}} \in \mathbb{F}$ such that $x = [\![x]\!]_{\mathsf{s}} + [\![x]\!]_{\mathsf{r}}$.

We denote variables of multivariate polynomials in bold, non-italic font, e.g., $\mathbf{x}, \mathbf{e}, \mathbf{z}$. A polynomial $f \in \mathbb{F}[\mathbf{x}]$ is in italics. When given concrete values, we denote $\mathbf{x}$ as $\boldsymbol{x}$. We use the graded lexicographic monomial ordering, where $\mathbf{x}^{\boldsymbol{\alpha}} > \mathbf{x}^{\boldsymbol{\beta}}$ if and only if $\sum_i \alpha_i > \sum_i \beta_i$, or $\sum_i \alpha_i = \sum_i \beta_i$ and the left-most non-zero entry of $\boldsymbol{\alpha} - \boldsymbol{\beta}$ is positive. f_i will denote the ith smallest monomial in f. Let $\mathsf{LM}(f)$ denote the *largest monomial* in f (without its coefficient) and $\mathsf{LT}(f)$ denote the *largest term* in f (with its coefficient). A set $\mathcal{F} \subseteq \mathbb{F}[\mathbf{x}]$ of polynomials will be in calligraphy font (along with distributions). $\mathcal{M} := \{\mathbf{x}^{\boldsymbol{\alpha}} : \boldsymbol{\alpha} \in \mathbb{N}^n\} = \{\mathbf{x}_1^{\alpha_1} \cdot \ldots \cdot \mathbf{x}_n^{\alpha_n} : \boldsymbol{\alpha} \in \mathbb{N}^n\}$ denotes the set of monomials.

3.2 Distributions and Bias

A distribution $\mathcal{D}$ is associated with a set X and each $x \in X$ is associated with a probability $p(x) \in [0, 1]_{\mathbb{R}}$ s.t. $\sum_{x \in X} p(x) = 1$. Let $\mathsf{Dist}[X]$ denote the set of

all probability distributions over set X. We can sample an element x from $\mathcal{D}$, denoted as $x \leftarrow \mathcal{D}$, such that $\Pr[x \leftarrow \mathcal{D}] = p(x)$. In places, we will also treat $\mathcal{D}$ as a random variable in the natural way. When X is a set, we use $x \leftarrow X$ to denote sampling from the uniform distribution $\mathcal{U}$ over X, i.e., $p(x) = 1/|X|$.

We will make use of the notion of bias, which measures how much a distribution $\mathcal{D}$ is correlated with a linear function $\boldsymbol{v}$. Given a distribution $\mathcal{D}$ over $\mathbb{F}^n$ and a non-zero vector $\boldsymbol{v} \in \mathbb{F}^n$, the bias of $\mathcal{D}$ with respect to $\boldsymbol{v}$, denoted as $\mathsf{bias}_{\boldsymbol{v}}(\mathcal{D})$, is equal to: $\mathsf{bias}_{\boldsymbol{v}}(\mathcal{D}) := \left| \mathop{\mathbb{E}}_{\boldsymbol{d} \leftarrow \mathcal{D}} [\chi(\boldsymbol{v} \cdot \boldsymbol{d})] \right|$. where $\chi : \mathbb{F} \to \mathbb{C}$ is a non-trivial character of $\mathbb{F}^n$, e.g., $\chi(x) := \exp \frac{2\pi i \mathsf{Tr}(x)}{p}$ for a field with p^k elements and field trace $\mathsf{Tr}(x) := x + x^p + ... + x^{p^{k-1}}$. In the binary case, this simplifies to $\mathsf{bias}_{\boldsymbol{v}}(\mathcal{D}) = |\Pr[\boldsymbol{v} \cdot \boldsymbol{x} = 0] - 1/2|$. By $\mathsf{bias}(\mathcal{D})$, we denote the largest bias of $\mathcal{D}$ with respect to any non-zero $\boldsymbol{v}$: $\mathsf{bias}(\mathcal{D}) = \max_{\boldsymbol{v} \neq \boldsymbol{0}} (\mathsf{bias}_{\boldsymbol{v}}(\mathcal{D}))$ We now present bias for a distribution $\sum_{i \leq t} \mathcal{D}_i$, obtained by taking t independent distributions $\mathcal{D}_1, \ldots, D_t$ over $\mathbb{F}^n$, sampling $\boldsymbol{d}_i \leftarrow \mathcal{D}_i$ and outputting the sum $\boldsymbol{d} = \sum_{i \in [t]} \boldsymbol{d}_i$. One can generalize [Shp09]'s lemma for $\mathbb{F}_2^n$ to arbitrary $\mathbb{F}^n$.

Lemma 1. *For t independent distributions $\mathcal{D}_1, \ldots, \mathcal{D}_t$ over $\mathbb{F}^n$, the bias of a distribution $\mathcal{D} := \sum_{i \in [t]} \mathcal{D}_i$ is bounded by $\mathsf{bias}(\mathcal{D}) \leq \prod_i \mathsf{bias}(\mathcal{D}_i)$ and that $\mathsf{bias}_{\boldsymbol{v}} \leq \prod_i \mathsf{bias}_{\boldsymbol{v}_i}(\mathcal{D}_i)$ for non-zero $\boldsymbol{v}_i \in \mathbb{F}^n$.*

Proof.

$$
\begin{aligned}
\mathsf{bias}(\mathcal{D}) &= \mathsf{bias}\left(\sum_i \mathcal{D}_i\right) \\
&= \max_{\boldsymbol{v} \neq 0} \left| \mathop{\mathbb{E}}_{\boldsymbol{d}_i \leftarrow \mathcal{D}_i} [\chi(\boldsymbol{v} \cdot (\boldsymbol{d}_1 + \ldots + \boldsymbol{d}_t))] \right| \\
&= \max_{\boldsymbol{v} \neq 0} \left| \mathop{\mathbb{E}}_{\boldsymbol{d}_i \leftarrow \mathcal{D}_i} [\chi(\boldsymbol{v} \cdot \boldsymbol{d}_1) \cdot \ldots \cdot \chi(\boldsymbol{v} \cdot \boldsymbol{d}_t)] \right| \qquad (1) \\
&\leq \max_{\boldsymbol{v}_1 \neq 0} \left| \mathop{\mathbb{E}}_{\boldsymbol{d}_1 \leftarrow \mathcal{D}_1} [\chi(\boldsymbol{v}_1 \cdot \boldsymbol{d}_1)] \right| \cdot \ldots \cdot \max_{\boldsymbol{v}_1 \neq 0} \left| \mathop{\mathbb{E}}_{\boldsymbol{d}_t \leftarrow \mathcal{D}_t} [\chi(\boldsymbol{v} \cdot \boldsymbol{d}_t)] \right| \qquad (2) \\
&= \prod_i \mathsf{bias}(\mathcal{D}_i)
\end{aligned}
$$

where (1) follows from $\chi(x+y) = \chi(x)\chi(y)$ and (2) follows from the independence of $\mathcal{D}_i$ and the triangle inequality. The proof of $\mathsf{bias}_{\boldsymbol{v}}(\mathcal{D})$ immediately follows. □

3.3 Coding Theory

Let $\mathcal{C}$ be a linear code that consists of a set of codewords $\boldsymbol{v}$ such that $\mathcal{C} := \{\boldsymbol{v} := \boldsymbol{x}\mathbf{G} \mid \boldsymbol{x} \in \mathbb{F}^k\}$, where $\boldsymbol{x}$ is any input of length k, $\mathbf{G} \in \mathbb{F}^{k \times n}$ is a *generator matrix* of $\mathcal{C}$, and $\mathbb{F}$ is some field. A matrix $\mathbf{H} \in \mathbb{F}^{m \times n}$, whose kernel is $\mathcal{C} := \{\boldsymbol{v} \mid \mathbf{H}\boldsymbol{v} := 0^m\}$, is called a *parity check matrix*. The code generated by $\mathbf{H}$ is called the dual code of $\mathcal{C}$. From $\mathbf{G}$, we can construct $\mathbf{H}$ and vice versa. It follows that $\mathbf{G}\mathbf{H}^T = 0$.

The *minimum distance* d of a linear code $\mathcal{C}$ represents the minimum number of positions of some codeword $\boldsymbol{v}$ that must be modified to get another codeword $\boldsymbol{v}'$. Equivalently, d can be defined as the minimum weight non-zero codeword or as the minimum number of linearly dependent columns of $\mathbf{H}$. A *dual distance* of the matrix $\boldsymbol{A}$ is the largest integer d such that every subset of d rows of $\boldsymbol{A}$ is linearly independent. It is also the minimum distance of the dual of the code generated by $\boldsymbol{A}$.

3.4 Syndrome Decoding

Syndrome Decoding (SD), or equivalently the *dual* learning parity with noise (LPN) formulation, states that for some field $\mathbb{F}$, a public matrix $\mathbf{G} \in \mathbb{F}^{k \times n}$ that is a generator of a linear error-correcting code, and a private weight-t sparse noise vector $\boldsymbol{e} \in \mathbb{F}^n$ sampled from some noise distribution, $(\mathbf{G}, \mathbf{Ge})$ is indistinguishable from $(\mathbf{G}, \boldsymbol{b})$, where $\boldsymbol{b} \leftarrow \mathbb{F}^k$ is uniformly random.

Definition 1 (Syndrome Decoding Assumption (SD)). *Syndrome Decoding is parameterized by an implicit computational security parameter κ, a field $\mathbb{F}$, dimensions $n, k \in \mathbb{N}$ with $k < n$, and distributions $\mathcal{D} \in \mathsf{Dist}[\mathbb{F}^n]$, $\mathcal{G} \in \mathsf{Dist}[\mathbb{F}^{k \times n}]$. For samples $\boldsymbol{e} \leftarrow \mathcal{D} \in \mathbb{F}^n$ and $\mathbf{G} \leftarrow \mathcal{G} \in \mathbb{F}^{k \times n}$, the $(n, k, \mathbb{F}, \mathcal{D}, \mathcal{G})$-SD assumption states:*

$$\{(\mathbf{G}, \boldsymbol{b}) \mid \boldsymbol{b} := \mathbf{G} \cdot \boldsymbol{e}\} \approx \{(\mathbf{G}, \boldsymbol{b}) \mid \boldsymbol{b} \leftarrow \mathbb{F}^k\}$$

where $\approx$ denotes computational indistinguishability.

The SD (and similarly our SSD) assumption is known to be false for some choices of $\mathcal{G}$ and $\mathcal{D}$. For example, this is the case when $\mathbf{G}$ has small minimum distance, is a Reed-Solomon code, or $\boldsymbol{e}$ is too sparse. This definition serves as template that we make concrete in Section 5 and Section 6.

Learning Parity with Noise (LPN) is a fundamental cryptographic assumption introduced by [BFKL94] and is equivalent to SD. LPN states that for some field $\mathbb{F}$, some public matrix $\boldsymbol{A} \in \mathbb{F}^{n \times m}$, a random secret vector $\boldsymbol{s} \leftarrow \mathbb{F}^m$, and a random secret weight-t sparse noise vector $\boldsymbol{e} \in \mathbb{F}^n$, $(\boldsymbol{A}, \boldsymbol{As} + \boldsymbol{e})$ is indistinguishable from $(\boldsymbol{A}, \boldsymbol{b})$, where $\boldsymbol{b} \leftarrow \mathbb{F}^n$ is uniformly random. In the original formulations both $\boldsymbol{A}$ and $\boldsymbol{s}$ were uniformly random and each position of $\boldsymbol{e}$ was sampled from the Bernoulli with parameters t/n so that in expectation $|\boldsymbol{e}| = t$. However, fruitful lines of research have shown that better performance can be achieved by adding structure to these distributions. Commonly, $\boldsymbol{A}$ is the transpose of a parity check matrix $\mathbf{H}$ of a linear error-correcting code with high minimum distance and fast encoding, e.g., [BCG+22,CRR21,RRT23]. Additionally, to improve performance of certain protocols $\boldsymbol{e}$ is often sampled from some noise distribution, e.g., regular [AFS05] as we will explain later.

Definition 2 (Learning Parity with Noise Assumption (LPN)). *Learning Parity with Noise is parameterized by an implicit computational security parameter κ, a field $\mathbb{F}$, dimensions $n, m \in \mathbb{N}$ with $n > m$, and distributions $\mathcal{D} \in \mathsf{Dist}[\mathbb{F}^n]$,*

$\mathcal{H} \in \mathsf{Dist}[\mathbb{F}^{n\times m}]$. *For samples* $\boldsymbol{e} \leftarrow \mathcal{D} \in \mathbb{F}^n$, $\boldsymbol{A} \leftarrow \mathcal{H} \in \mathbb{F}^{n\times m}$, *and* $\boldsymbol{s}_i \leftarrow \mathbb{F}^m$, *the* $(n, m, \mathbb{F}, \mathcal{D}, \mathcal{H})$-$\mathsf{LPN}$ *assumption states:*

$$\{(\boldsymbol{A}, \boldsymbol{b}) \mid \boldsymbol{b} := \boldsymbol{A} \cdot \boldsymbol{s} + \boldsymbol{e}\} \approx \{(\boldsymbol{A}, \boldsymbol{b}) \mid \boldsymbol{b} \leftarrow \mathbb{F}^n\}$$

where $\approx$ *denotes computational indistinguishability.*

LPN and SD Equivalence. Recall that for a linear error-correcting code, the matrix $\boldsymbol{A}$ from the LPN formulation is the transpose of a parity check matrix $\mathbf{H} = \boldsymbol{A}^\mathsf{T}$, while $\mathbf{G}$ from the SD formulation is the generator. The key observation is that $\mathbf{G}\mathbf{H}^\mathsf{T} = 0$. Then, $\mathbf{G}(\mathbf{H}^\mathsf{T}\boldsymbol{s}+\boldsymbol{e}) = (\mathbf{G}\mathbf{H}^T)\boldsymbol{s}+\mathbf{G}\boldsymbol{e} = \mathbf{G}\boldsymbol{e} = \boldsymbol{b}$. Similarly, given a SD sample $(\mathbf{G}, \boldsymbol{b})$, one can define the equivalent LPN instance by sampling a uniform $\hat{\boldsymbol{s}} \in \mathbb{F}^m$ and outputting $(\boldsymbol{A}, \hat{\boldsymbol{b}})$ where $\hat{\boldsymbol{b}} := \boldsymbol{A}\hat{\boldsymbol{s}} + \hat{\boldsymbol{e}}$ and $\hat{\boldsymbol{e}}$ is an arbitrary solution to $\mathbf{G}\hat{\boldsymbol{e}} = \hat{\boldsymbol{b}}$. Correctness follows from the fact that there exists some $\boldsymbol{s} \in \mathbb{F}^m$ s.t. $\boldsymbol{A}\boldsymbol{s} + \boldsymbol{e} = \boldsymbol{A}\hat{\boldsymbol{s}} + \hat{\boldsymbol{e}}$.

Noise Distributions. Three choices for the noise distribution are dominant in the literature: Bernoulli, exact, and regular. In secure computation applications, their choice greatly impacts efficiency. Let t be the desired sparsity of the error vector $\boldsymbol{e}$ of size n, consisting of elements from some field $\mathbb{F}$:

- *Bernoulli* is the classic noise distribution. Each $\boldsymbol{e}_i \in \mathbb{F}$ is sampled with $\mathsf{Ber}_{t/n}$, i.e., 0 with probability $1 - t/n$ and otherwise uniformly from $\mathbb{F} \setminus \{0\}$.
- *Exact* noise distribution evolved from the Bernoulli distribution and fixes the Hamming weight of the noise vector $|\boldsymbol{e}| = t$, I.e., $\boldsymbol{e} \leftarrow \{\boldsymbol{e} \in \mathbb{F}^n \mid |\boldsymbol{e}| = t\}$.
- *Regular* is the distribution of choice for pseudorandom correlation generators (PCG) as the noise vector is much cheaper to implement under secure computation than in the exact case (details to follow). $\boldsymbol{e}$ consists of t same-size blocks $\boldsymbol{e}_1, ..., \boldsymbol{e}_t \in \mathbb{F}^{n/t}$, where each $\boldsymbol{e}_i$ is a uniformly random unit vector. I.e., $\boldsymbol{e}_i \leftarrow \{\boldsymbol{e}_i \in \mathbb{F}^{n/t} \mid |\boldsymbol{e}_i| = 1\}$.

Interestingly, there is not a clearly best noise distribution when it comes to security. [ES24] recently showed that regular noise can actually be harder for some parameter regimes, and vice versa.

3.5 Linear Attacks

Cryptanalyzing LPN is a thriving area of research. Many attacks have been proposed of which the most effective are those based on Gaussian elimination/BKW algorithm [BKW03,Lyu05] and information set decoding (ISD) [Pra62,Ste89]. When introducing a new LPN variant, it would be laborious to prove security against each possible attack. Fortunately, the majority of known attacks fit in the *linear test framework*. These include, among others, attacks based on Gaussian elimination and the BKW algorithm [BKW03,Lyu05], attacks based on covering codes and information set decoding [Pra62,Ste89], statistical decoding [Jab01,FKI07], and finding correlations with low degree polynomials

[ABG+14,BR17]. In this framework, the adversary is given the matrix $\boldsymbol{A}$, arbitrarily preprocesses it, and then outputs a test vector $\boldsymbol{v}$. Now, the framework states that the distinguisher, who tries to distinguish the LPN sample from uniformly random, can be implemented by a simple linear test $\boldsymbol{v} \cdot (\boldsymbol{A}\boldsymbol{s} + \boldsymbol{e})$ and by checking if the output is biased, e.g., equals zero more than random chance.

One advantage of the linear test framework is that when LPN is initialized with a code that has high minimum distance d, then it cannot be distinguished from uniform except with negligible probability. Note that high minimum distance is not a necessary condition for security. It is well-known that small minimum distance can result in secure LPN as long as the *pseudominimum distance* is high. Pseudominimum distance represents the weight of the smallest codeword that is efficiently computable. In other words, small minimum distance codewords can exist as long as they cannot be efficiently found. We note that while the linear test framework covers the large majority of known attacks, there are some notable exceptions such as when the underlying code is strongly algebraic (e.g., Reed-Solomon) or the noise is structured (e.g., regular). Stationary syndrome decoding, which we introduce in this paper, has a highly structured noise, and thus requires analysis beyond the linear test framework.

To illustrate how the linear test framework captures various attacks, we next review how to cast state-of-the-art ISD algorithms as linear tests. First, the SD problem $(\mathbf{G}, \mathbf{G} \cdot \boldsymbol{e})$ is converted into its equivalent LPN formulation $(\boldsymbol{A}, \boldsymbol{b} := \boldsymbol{A}\boldsymbol{s} + \boldsymbol{e})$, see above. A subset of $m := n - k$ rows of $\boldsymbol{A}$ are selected, called an *information set* $I \subset [n]$, with the hope that this set does not intersect the non-zeros of the noise vector $\boldsymbol{e}$, i.e. $\boldsymbol{e}_i = 0$ for $i \in I$. One can then use Gaussian elimination to solve for the noisy positions of $\boldsymbol{e}$ by considering the corresponding $m \times m$ submatrix $\boldsymbol{A}'$, which consists of the rows in the information set i.e. $\boldsymbol{A}' := (\boldsymbol{A}_{I_1} /\!\!/ ... /\!\!/ \boldsymbol{A}_{I_m})$. One then solves the system $\boldsymbol{A}' \cdot \boldsymbol{s} = \boldsymbol{b}_I$ using Gaussian elimination and checks if $\boldsymbol{e}' := \boldsymbol{A}\boldsymbol{s} - \boldsymbol{b}$ is a t-sparse vector, which implies $\boldsymbol{e}' = \boldsymbol{e}$ w.h.p. This algorithm can be implemented in the linear test framework, for example, by simply checking if $\boldsymbol{e}'_1 = 0$, which is linear in the original SD problem since $\boldsymbol{e}'$ is linear in $\mathbf{G}\boldsymbol{e}$.

When ISD is naively implemented, each information set guess requires a costly Gaussian elimination step. Advanced ISD algorithms [Ste89,FS09a,BLP11] [MMT11,BJMM12,MO15,BM18,ES24] typically improve efficiency through a combination of techniques, such as reducing the overall search space, optimizing the process of finding the right information set, or amortizing the cost of Gaussian elimination by reusing partial computations across multiple *related* information sets. This avoids the need to restart the linear algebra from scratch for each attempt. While processing each individual set still takes at least $O(1)$ time, the overall approach becomes significantly more efficient. As a result, in the linear test framework, these attacks can be structured so that each test effectively runs in $O(1)$ time as well.

Importantly, we note that the noise parameter t suggested by the linear test framework tends to be highly conservative. For example, [LWYY22] shows that in SD, a choice of $t \approx 60$ should provide the same level of bit-security against

linear tests as opposed to $t \approx 170$ required by the linear test framework. In this work, we adopt a more conservative approach and choose parameters according to the linear test framework to ensure provable security guarantees. However, we are also not aware of any concrete speedup for ISD-styled algorithms when applied to SSD, which suggests more aggressive parameters could be considered.

3.6 Algebraic Preliminaries

Recent work by Briaud and Øygarden [BØ23] represents one such attack that does not fit within the linear test framework. The attack is based on algebraic geometry and leverages the regular structure of noise in LPN and SD. At the highest level, the attack represents LPN/SD as a system of linear and non-linear polynomials and then solves for the noise vector $\boldsymbol{e}$. [BØ23]'s work is a crucial prerequisite to analyze the security of our assumption against algebraic attacks. For that reason, we first provide an overview of the concepts necessary to understand [BØ23]'s attack before reviewing their attack.

Vector Spaces. Let $V \subset \mathbb{F}^d$ be a vector space of dimension d over a field $\mathbb{F}$. To qualify as a vector space, V must satisfy the following properties:

(a) Associativity: $\forall \boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w} \in V, \boldsymbol{u} + (\boldsymbol{v} + \boldsymbol{w}) = (\boldsymbol{u} + \boldsymbol{v}) + \boldsymbol{w}$.
(b) Commutativity: $\forall \boldsymbol{u}, \boldsymbol{v} \in V, \boldsymbol{u} + \boldsymbol{v} = \boldsymbol{v} + \boldsymbol{u}$.
(c) Identity: $0 \in V$ and $\forall \boldsymbol{v} \in V, \boldsymbol{v} + 0 = \boldsymbol{v}$.
(d) Inverse: $\forall \boldsymbol{v} \in V$, there exists $\boldsymbol{u} \in V$ s.t. $\boldsymbol{v} + \boldsymbol{u} = 0$.
(e) Multiplicative associativity: $\forall \boldsymbol{v} \in V, a, b \in \mathbb{F}, a(b\boldsymbol{v}) = (ab)\boldsymbol{v}$.
(f) Multiplicative identity: $\forall \boldsymbol{v} \in V, 1\boldsymbol{v} = \boldsymbol{v}$.
(g) Multiplicative distributivity: $\forall \boldsymbol{u}, \boldsymbol{v} \in V, a \in \mathbb{F}, a(\boldsymbol{u} + \boldsymbol{v}) = a\boldsymbol{u} + a\boldsymbol{v}$.
(h) Additive distributivity: $\forall \boldsymbol{v} \in V, a, b \in \mathbb{F}, (a + b)\boldsymbol{v} = a\boldsymbol{v} + b\boldsymbol{v}$.

For a subset $S \subseteq V$, the $\mathbb{F}$-span of S, denoted as $\mathsf{span}_{\mathbb{F}}(S)$ are all $\mathbb{F}$-linear combinations of elements of S, i.e., $\mathsf{span}_{\mathbb{F}}(S) := \{\sum_i a_i \boldsymbol{s}_i \mid a_i \in \mathbb{F}, \boldsymbol{s}_i \in S\}$. $B \subseteq V$ is a *basis* of V if all elements of B are linearly independent and the span of B is V. All bases B of V have the same size and $||B||$ is called the *dimension* of V. This is denoted as $\mathsf{dim}_{\mathbb{F}}(V) := ||B||$ where B is a basis of V. The dimension of V can be finite or infinite.

Graded Rings. Recall that an *Abelian group* is a set A with an operation $\circ$ that satisfies

(a) Associativity: $\forall a, b, c \in A, a \circ (b \circ c) = (a \circ b) \circ c$.
(b) Commutativity: $\forall a, b \in A, a \circ b = b \circ a$.
(c) Identity: $\exists e \in A$ such that $\forall a \in A, a \circ e = a$.
(d) Inverse: $\forall a \in A$, there exists $b \in A$ s.t. $a \circ b = e$.

For example, $\mathbb{F}$-vector spaces are Abelian groups under $+$ (as described in $\mathbb{F}$). A *ring* is a set R with two operations $+$ and $\cdot$ that satisfies

(a) R with $+$ forms an Abelian group

(b) Multiplicative associativity: $\forall a, b, c \in R, a \cdot (b \cdot c) = (a \cdot b) \cdot c$.
(c) Mulitplicative identity: $\exists 1 \in R$ such that $\forall a \in R, a \cdot 1 = 1 \cdot a = a$.
(d) Multiplicative distributivity: $\forall a, b, c \in R, a \cdot (b+c) = a \cdot b + a \cdot c$ and $(b+c) \cdot a = b \cdot a + c \cdot a$.

If $\cdot$ also commutes, we say that R is a *commutative ring*. We will typically assume that rings are commutative unless stated otherwise.

For Abelian groups A and B with operations $*$ and $\star$, the *direct sum* or *direct product* is the Abelian group $A \oplus B = A \times B$ where the underlying set is the Cartesian product $\{(a, b) : a \in A, b \in B\}$ and the operation $\circ$ is defined component-wise as $(a_1, b_1) \circ (a_2, b_2) = (a_1 * a_2, b_1 \star b_2)$. This definition immediately generalizes to the direct sum of finitely many Abelian groups. For an infinite indexed family of abelian groups $(A_i)_{i=0}^{\infty}$, their direct sum $\bigoplus_{i=0}^{\infty} A_i$ consists of the elements that have finite support, i.e., the element considered from all but finitely many A_is must be the corresponding identity element.

A *graded ring* is a ring R with two operations $+$ and $\cdot$ that can be decomposed into a direct sum $R = \bigoplus_{i=0}^{\infty} R_i$ of Abelian groups $(R_i)_{i=0}^{\infty}$ under $+$ such that $R_i \cdot R_j \subseteq R_{i+j}$ for all $i, j \geq 0$, where $R_i \cdot R_j = \{r \cdot r' : r \in R_i, r' \in R_j\}$.

Polynomials and Ideals. Let $\mathcal{A} = \mathbb{F}[\mathbf{x}]$ denote a polynomial ring in n variables $\mathbf{x} = \mathbf{x}_1, \ldots, \mathbf{x}_n$ with coefficients in a field $\mathbb{F}$ (the number of variables will be clear from context). We call a polynomial $f \in \mathcal{A}$ *homogeneous* when all its monomials have the same degree, e.g., $\mathbf{x}_1\mathbf{x}_3^2 + 2\mathbf{x}_2\mathbf{x}_3\mathbf{x}_4 + \mathbf{x}_4^3$ with $\mathsf{deg}(\mathbf{x}_1\mathbf{x}_3^2) = \mathsf{deg}(2\mathbf{x}_2\mathbf{x}_3\mathbf{x}_4) = \mathsf{deg}(\mathbf{x}_4^3) = 3$. Otherwise, f is *affine*. In that case, one of the ways in which we can turn f into a homogeneous polynomial $f^{(h)}$ (i.e., homogenize it) is by retaining only the terms of degree $\mathsf{deg}(f)$ and removing all lower-degree terms. Homogenizing polynomials is done primarily for the technical reason of studying them in the context of projective geometry, although this will not be something we will need to go into.

A subset $\mathcal{I} \subseteq \mathcal{A}$ is called an *ideal* when

(a) $0 \in \mathcal{I}$
(b) for all $f, g \in \mathcal{I}$, $f + g \in \mathcal{I}$, and
(c) for all $f \in \mathcal{I}$ and $h \in \mathcal{A}$, $hf \in \mathcal{I}$.

That is, an ideal is closed under addition and projects all items h in the ring $\mathcal{A}$ into the ideal when multiplied with an ideal element $f \in \mathcal{I}$. For our purposes, an ideal represents the set of equations that are implied by a system of equations.

For polynomials $f_1, \ldots, f_s \in \mathcal{A}$, $\langle f_1, \ldots, f_s \rangle = \{\sum_{i=1}^{s} h_i f_i : h_1, \ldots, h_s \in \mathcal{A}\}$ is an ideal and $f_1, \ldots, f_s$ are called its *generators*. Generators for an ideal are typically not unique. The ideal generated by the polynomials $f_1, \ldots, f_s$ can be seen in the following way. For unknowns $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$, consider the system of equations

$$f_1(\mathbf{x}) = \ldots = f_s(\mathbf{x}) = 0,$$

then the ideal $\langle f_1, \ldots, f_s \rangle$ contains all other polynomials g such that $g(\mathbf{x}) = 0$ is implied, e.g., $g(\mathbf{x}) = \mathbf{x}_1 \cdot f_4(\mathbf{x}) + f_1(\mathbf{x}) = 0$ is clearly implied. If we compare

with our definition of an ideal then we can check (a) $0(\mathbf{x}) = 0$. Next, (b) if $f(\mathbf{x}) = g(\mathbf{x}) = 0$, then also $(f+g)(\mathbf{x}) = 0$. Lastly, (c) if $f(\mathbf{x}) = 0$, then for all $h \in \mathcal{A}$, $(hf)(\mathbf{x}) = 0$. The notion of an ideal precisely describes the operations we can perform on the system of equations to maintain the same system.

An ideal $\mathcal{I} = \langle f_1, \ldots, f_s \rangle$ is homogeneous when its generators $f_1, \ldots, f_s$ are homogeneous. Otherwise, $\mathcal{I}$ is affine. Similar to polynomials, we can homogenize $\mathcal{I} = \langle f_1, \ldots, f_s \rangle$ by making each generator homogeneous, i.e., we replace each f_i with its homogeneous version $f_i^{(h)}$, resulting in $\mathcal{I}^{(h)} := \langle f_1^{(h)}, \ldots, f_s^{(h)} \rangle$.

An important point is that $\mathcal{A}$ as well as any ideal $\mathcal{I}$ can also be seen as $\mathbb{F}$-vector spaces. For example, B is a vector space basis of $\mathcal{A}$ or $\mathcal{I}$ if B contains all monomials in $\mathcal{A}$ or $\mathcal{I}$ respectively. As such, $\mathcal{A}$ would be an infinite dimensional $\mathbb{F}$-vector space, where each possible monomial contributes to a dimension in the vector space. The same holds for all $\mathcal{I} \neq \{0\}$. For any $d \in \mathbb{N}$, we denote $\mathcal{A}_d$ as the vector space defined as $\mathcal{A}_d := \{f \in \mathcal{A} : \mathsf{deg}(f) = d \text{ and } f \text{ is homogeneous}\} \cup \{0\}$. $\mathcal{A}_d$ forms a finite dimensional subspace of $\mathcal{A}$. Similarly, $\mathcal{I}_d := \mathcal{I} \cap \mathcal{A}_d = \{f \in \mathcal{I} : \mathsf{deg}(f) = d \text{ and } f \text{ is homogeneous}\} \cup \{0\}$ is also a vector space and forms a finite dimensional subspace of $\mathcal{A}_d$.

Quotient Rings and Spaces. Let $\mathcal{I} \subseteq \mathcal{A}$ be an ideal. We can define an equivalence relation $\sim$ on $\mathcal{A}$ using $\mathcal{I}$ as follows: $f \sim g \iff f - g \in \mathcal{I}$. It is easy to check that $\sim$ is in fact an equivalence relation. Therefore, $\sim$ partitions $\mathcal{A}$ into equivalence classes. The equivalence class corresponding to a polynomial $f \in \mathcal{A}$ is given by $[f]_{\mathcal{I}} := \{f + g : g \in \mathcal{I}\} = \mathcal{I} + f$. When clear from context we will write $[f]$. Therefore $[0] = \mathcal{I}$ and each class $[f]$ can be thought of as shifting $\mathcal{I}$ by direction f. Note that there is typically not a canonical representative element for $[f]$, unlike the integers mod p for example. We can define addition and multiplication over the equivalence classes in a natural way as follows: $[f] + [g] = [f + g]$ and $[f] \cdot [g] = [fg]$. The identities with respect to addition and multiplication are $[0]$ and $[1]$ respectively. The additive inverse of $[f]$ is $[-f]$. One can then check that the equivalence classes form a ring. This ring is called the *quotient ring* and is denoted by $\mathcal{A}/\mathcal{I} := \{[f]_{\mathcal{I}} : f \in \mathcal{A}\} = \{\mathcal{I} + f : f \in \mathcal{A}\}$. The quotient ring $\mathcal{A}/\mathcal{I}$ elements are the distinct cosets of $\mathcal{I}$. Observe also that $\mathcal{A}/\mathcal{I}$ forms an $\mathbb{F}$-vector space.

Similarly, for $d \in \mathbb{N}$, recall the vector spaces $\mathcal{I}_d \subseteq \mathcal{A}_d$. We can define an equivalence relation $\sim$ on the vector space $\mathcal{A}_d$ using $\mathcal{I}_d$ as follows: $f \sim g \iff f - g \in \mathcal{I}_d$. It is easy to check that $\sim$ is in fact an equivalence relation. Therefore, $\sim$ partitions the vector space $\mathcal{A}_d$ into equivalence classes. The equivalence class corresponding to a polynomial $f \in \mathcal{A}_d$ is given by $[f]_{\mathcal{I}_d} := \{f + g : g \in \mathcal{I}_d\}$ or just $[f]$ when $\mathcal{I}_d$ is clear from context. We can define addition and scalar multiplication over the equivalence classes in a natural way as follows: $[f] + [g] = [f + g]$ and for $\alpha \in \mathbb{F}$, $\alpha \cdot [f] = [\alpha f]$. The identity with respect to addition is $[0]$. The additive inverse of $[f]$ is $[-f]$. One can then check that the equivalence classes form an $\mathbb{F}$-vector space. This vector space is called the *quotient vector space* and is denoted by $\mathcal{A}_d/\mathcal{I}_d$ or $(\mathcal{A}/\mathcal{I})_d$. $[f]$ also does not have a canonical representative. For example, if $\mathcal{I}_2$ has a basis $\{\mathbf{x}_1\mathbf{x}_2 + \mathbf{x}_2^2, \mathbf{x}_1^2 - \mathbf{x}_2^2\}$, then we have $[\mathbf{x}_1\mathbf{x}_2] = [2\mathbf{x}_1\mathbf{x}_2 + \mathbf{x}_2^2] = [2\mathbf{x}_1\mathbf{x}_2 + \mathbf{x}_1^2]$. Both $[\mathbf{x}_1\mathbf{x}_2]$ and $[2\mathbf{x}_1\mathbf{x}_2 + \mathbf{x}_1^2]$ are reduced

and represent the same equivalence class. A geometric interpretation of this is that the cosets $\mathcal{I} + \mathbf{x}_1\mathbf{x}_2 = \mathcal{I} + 2\mathbf{x}_1\mathbf{x}_2 + \mathbf{x}_1^2$ are equal but were shifted in two different directions.

Dimension of an Ideal. Although we do not formally define the (Krull) dimension of an ideal, it describes the asymptotic growth of the size of the set of solutions to the system of equations corresponding to $\mathcal{I} = \langle f_1, \ldots, f_s \rangle$. This solution set of $\mathbf{x} \in \mathbb{F}^n$ is called the variety of $\mathcal{I}$, denoted as $\mathsf{V}(I) := \{\boldsymbol{x} : \boldsymbol{x} \in \mathbb{F}^n, f_1(\boldsymbol{x}) = \ldots = f_s(\boldsymbol{x}) = 0\}$. In general, $\mathsf{V}(I)$ is a multi-dimensional curve in an n dimensional space. For example, let $\mathcal{I} = \langle \mathbf{x}_1 - \mathbf{x}_2^2 \rangle$, and therefore the solution set lies on the *one* dimensional curve $\mathbf{x}_1 = \mathbf{x}_2^2$, the basic parabola. A two dimensional variety can be obtained by $\mathcal{I} = \langle \mathbf{x}_2 - \mathbf{x}_1^2 - \mathbf{x}_3, \mathbf{x}_4 - \mathbf{x}_1\mathbf{x}_2 \rangle$ where the variety can be defined as $\mathsf{V}(\mathcal{I}) = \{\boldsymbol{x} : \boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{F}, \boldsymbol{x}_3 = \boldsymbol{x}_2 - \boldsymbol{x}_1^2, \boldsymbol{x}_4 = \boldsymbol{x}_1\boldsymbol{x}_2\}$ which clearly is a two dimensional curve in a four dimensional space. Any non-zero dimensional ideal will be an infinite dimensional vector space as $\mathcal{I}$ does not reduce an infinite many monomials. E.g., if we order the $\mathbf{x}_k$ to eliminate larger k first, then all $\mathbf{x}_1^i$ are reduced for the first example and $\mathbf{x}_1^i\mathbf{x}_2^j$ for the second. Note the irreducable monomials are in general more complicated. Observe that it suffices to only consider the monomials because if a polynomial was not in $\mathcal{I}$ then its monomials must also not be in $\mathcal{I}$. The number of solutions in $\mathsf{V}(\mathcal{I})$ is asymptotically related to the number of the number of monomials not in $\mathcal{I}$. Informally, a d dimensional ideal $\mathcal{I}$ will have $O(i^d)$ monomials not in $\mathcal{I}$ with degree at most i. Geometrically, the dimension of $\mathcal{I}$ is the dimension of the volume left by $\mathcal{A}$ modulo $\mathcal{I}$.

Zero-dimensional Ideal. Intuitively, an ideal $\mathcal{I} \subseteq \mathcal{A}$ is zero-dimensional if $\mathsf{V}(\mathcal{I})$ is a finite set, i.e., $\mathcal{I}$ has only a finite number of solutions over $\mathbb{F}^n$. Formally, an ideal $\mathcal{I} \subseteq \mathcal{A}$ is zero-dimensional if and only if $\mathcal{A}/\mathcal{I}$ is a finite dimensional $\mathbb{F}$-vector space. That is, there is a finite sized basis $B \subset \mathcal{A}/\mathcal{I}$ such that $\mathsf{span}_{\mathbb{F}}(B) = \{\sum_i a_i \boldsymbol{b}_i \mid a_i \in \mathbb{F}, \boldsymbol{b}_i \in B\}$ is equal to $\mathcal{A}/\mathcal{I} = \{[f]_{\mathcal{I}} : f \in \mathcal{A}\} = \{\mathcal{I} + f : f \in \mathcal{A}\}$. Note that the vector space dimension is distinct from the dimension of $\mathcal{I}$ but the two are related, i.e., the vector space dimension of $\mathcal{A}/\mathcal{I}$ is finite if and only if $\mathcal{I}$ is zero-dimensional and otherwise infinite.

Thus, when trying to understand the dimension of an ideal, we are interested in all of the monomials not in $\mathcal{I}$ whose equivalence classes form an $\mathbb{F}$-vector space basis of $\mathcal{A}/\mathcal{I}$, i.e., $B \subset \{[\mathbf{x}^\alpha] : \alpha \in \mathbb{N}^n\}/\mathcal{I}$. Note that if $\mathbb{F}$ is a finite field, any ideal $\mathcal{I} \subseteq \mathcal{A}$ is zero-dimensional by definition since the set of solutions is a subset of $\mathbb{F}^n$ which is finite, or equivalently, the field equations limit the size of B.

Hilbert Function. Let $\mathcal{I}$ be a homogeneous ideal. The Hilbert function $\mathbb{N} \to \mathbb{N}$ takes an integer d as input and outputs the dimension of the quotient vector space $\mathcal{A}_d/\mathcal{I}_d$ over $\mathbb{F}$, i.e.,

$$\mathcal{HF}_{\mathcal{A}/\mathcal{I}}(d) := \mathsf{dim}_{\mathbb{F}}(\mathcal{A}_d/\mathcal{I}_d).$$

Intuitively, this function describes the asymptotic growth rate of the size of the quotient ring as we allow larger and larger monomial powers.

Hilbert Series. Also known as the Hilbert-Poincaré series, for a homogeneous ideal $\mathcal{I} \subseteq \mathcal{A}$, the Hilbert Series of a quotient ring $\mathcal{A}/\mathcal{I}$ is given by

$$\mathcal{HS}_{\mathcal{A}/\mathcal{I}}(\mathbf{z}) := \sum_{d=0}^{\infty} \mathcal{HF}_{\mathcal{A}/\mathcal{I}}(d) \cdot \mathbf{z}^d.$$

Note that $\mathcal{HS}$ is a univariate series in a single variable $\mathbf{z}$.

Hilbert Polynomial. For a homogeneous zero-dimensional ideal $\mathcal{I} \subseteq \mathcal{A}$, the Hilbert Series becomes a polynomial. Formally, since $\mathcal{A}$ is a graded ring with

$$\mathcal{A} = \bigoplus_{d=0}^{\infty} \mathcal{A}_d,$$

$\mathcal{A}/\mathcal{I}$ is also a graded ring with

$$\mathcal{A}/\mathcal{I} = \bigoplus_{d=0}^{\infty} (\mathcal{A}_d/\mathcal{I}_d).$$

Therefore, as $\mathbb{F}$-vector spaces,

$$\mathsf{dim}_{\mathbb{F}}(\mathcal{A}/\mathcal{I}) = \sum_{d=0}^{\infty} \mathcal{HF}_{\mathcal{A}/\mathcal{I}}(d).$$

By definition, since $\mathcal{I}$ is zero-dimensional, $\mathsf{dim}_{\mathbb{F}}(\mathcal{A}/\mathcal{I})$ is finite and hence there is some d_{reg}, called the degree of regularity, such that for all $d \geq d_{\mathsf{reg}}$, $\mathcal{HF}_{\mathcal{A}/\mathcal{I}}(d) = 0$ (also $\mathcal{A}_d = \mathcal{I}_d$). As we will see, we will be interested in d_{reg} or the behavior of the Hilbert series in general as this directly relates to the efficiency of the best known algebraic attacks.

Short Exact Sequences. Let $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{Z}$ be finite dimensional vector spaces. A short exact sequence of $\mathcal{X}$, $\mathcal{Y}$, $\mathcal{Z}$ is given by two maps $\gamma : \mathcal{X} \to \mathcal{Y}$ and $\delta : \mathcal{Y} \to \mathcal{Z}$ such that γ is injective, i.e., $x_1 \neq x_2 \implies \gamma(x_1) \neq \gamma(x_2)$, δ is surjective, i.e., $\mathcal{Z} = \{\delta(y) : y \in \mathcal{Y}\}$, and the kernel of δ is the image of γ, i.e., $\{y \in \mathcal{Y} : \delta(y) = 0\} = \{\gamma(x) : x \in \mathcal{X}\}$. Therefore, $\delta(\gamma(x)) = 0$ for $x \in \mathcal{X}$. We denote a short exact sequence as

$$0 \to \mathcal{X} \xrightarrow{\gamma} \mathcal{Y} \xrightarrow{\delta} \mathcal{Z} \to 0.$$

We will use a key property of their dimensions given by

$$\mathsf{dim}(\mathcal{Y}) = \mathsf{dim}(\mathcal{X}) + \mathsf{dim}(\mathcal{Z}).$$

This is easy to see via the Rank-Nullity theorem. We have

$$\mathsf{dim}(\mathcal{Y}) = \mathsf{dim}(\mathsf{ker}(\delta)) + \mathsf{dim}(\mathsf{im}(\delta)) = \mathsf{dim}(\mathsf{im}(\gamma)) + \mathsf{dim}(\mathsf{im}(\delta))$$

Since γ is injective, we have $\mathsf{dim}(\mathsf{im}(\gamma)) = \mathsf{dim}(\mathcal{X})$, and since δ is surjective, we have $\mathsf{dim}(\mathsf{im}(\delta)) = \mathsf{dim}(\mathcal{Z})$. We will make use of this property to determine the dimensions of certain vector spaces that will let us get a handle on certain Hilbert polynomials.

Regular and Semi-Regular Systems. Computing the Hilbert series of an ideal described by arbitrary polynomials is often difficult. However, they are known for some systems of polynomials. We consider two such systems. For a system of p polynomial equations $\mathcal{F} := \{f_1, \ldots, f_p\}$ in n variables, we consider a *regular* system when $p \leq n$ and a *semi-regular* system when $p > n$. Before formally defining regular and semi-regular systems, we note that a randomly selected system is with overwhelming probability (semi-)regular depending on whether $p \leq n$.

Now, a system $\mathcal{F}$ is regular when $p \leq n$, $\mathcal{F}$ is homogeneous (i.e., each f_i is homogeneous), and for all $i \in [p]$, f_i is not a zero divisor in $\mathcal{A}/\langle f_1, \ldots, f_{i-1}\rangle$ (i.e., if $gf_i = 0$ in $\mathcal{A}/\langle f_1, \ldots, f_{i-1}\rangle$, then $g = 0$ in $\mathcal{A}/\langle f_1, \ldots, f_{i-1}\rangle$). Its Hilbert series is given by Proposition 1.

Proposition 1 ([BØ23]). *Let $\mathcal{F} = \{f_1, \ldots, f_p\}$ with $\mathsf{deg}(f_i) = d_i$, $i \in [p]$, be a regular system. Then its Hilbert series is given by*

$$\mathcal{HS}_{\mathcal{A}/\langle\mathcal{F}\rangle}(z) = \frac{\prod_{i=1}^{p}(1 - z^{d_i})}{(1-z)^n}.$$

A semi-regular system is defined differently over general $\mathbb{F}$ and over $\mathbb{F}_2$. Over $\mathbb{F}$, a system $\mathcal{F}$ is semi-regular when $p > n$, $\mathcal{F}$ is homogeneous, $\mathcal{I} := \langle\mathcal{F}\rangle \neq \mathcal{A}$ is a zero-dimensional ideal (let d_{reg} be its degree of regularity), and for all $i \in [p]$, whenever $gf_i = 0$ in $\mathcal{A}/\langle f_1, \ldots, f_{i-1}\rangle$ with $\mathsf{deg}(gf_i) < d_{\mathsf{reg}}$, $g = 0$ in $\mathcal{A}/\langle f_1, \ldots, f_{i-1}\rangle$. Its Hilbert series is almost the same as in Proposition 1 except that it is truncated after the first ≤ 0 coefficient as given by Proposition 2.

Proposition 2 ([Bar04]). *Let $\mathcal{F} = \{f_1, \ldots, f_p\}$ with $\mathsf{deg}(f_i) = d_i$, $i \in [p]$, be a semi-regular system over $\mathbb{F}_q$ for $q > 2$. Then its Hilbert series is given by*

$$\mathcal{HS}_{\mathcal{A}/\langle\mathcal{F}\rangle}(z) = \frac{\prod_{i=1}^{p}(1 - z^{d_i})}{(1-z)^n},$$

truncated after the first ≤ 0 coefficient.

The semi-regularity definition over $\mathbb{F}_2$ is almost identical, except that it uses the quotient ring $\mathbb{F}_2[\mathbf{x}]/\langle \mathbf{x}_1^2, \ldots, \mathbf{x}_n^2\rangle$ instead of the polynomial ring $\mathcal{A}$. Its Hilbert series is given by Proposition 3.

Proposition 3 ([Bar04]). *Let $\mathcal{F} = \{f_1, \ldots, f_p\}$ with $\mathsf{deg}(f_i) = d_i$, $i \in [p]$, be a semi-regular system over $\mathbb{F}_2$. Then its Hilbert series is given by*

$$\mathcal{HS}_{\mathcal{A}/\langle\mathcal{F}\rangle}(z) = \frac{(1+z)^n}{\prod_{i=1}^{p}(1 + z^{d_i})},$$

truncated after the first ≤ 0 coefficient.

In general, one could extend the above to any small field by accounting for the Frobenius morphism that maps $\mathbf{x}_i^{||\mathbb{F}||} = \mathbf{x}_i$ for $i \in [n]$.

Unfortunately, the polynomial systems considered in this work cannot be simply labeled as regular or semi-regular. However, the tools presented in this section will help us determine their Hilbert series.

Note. The definitions of $\mathcal{F}$ being regular or semi-regular implicitly involve an ordering among the polynomials in $\mathcal{F}$. One may wonder whether $\mathcal{F}$ may be regular or semi-regular with respect to one ordering of $\mathcal{F}$ but not with respect to another. This is not possible when $\mathcal{F}$ is homogeneous (and in fact is possible if $\mathcal{F}$ is not homogeneous). The proofs of these facts are standard and we do not get into them in this work.

Macaulay Matrix. The Macaulay matrix is an essential primitive for solving non-linear systems of equations. Let $\mathcal{M} := \{\mathbf{x}^\alpha : \alpha \in \mathbb{N}^n\}$ be the set of all monomials. Let $\mathsf{coeff}(f, m)$ represent the coefficient of a monomial $m \in \mathcal{M}$ in a polynomial $f \in \mathcal{A}$. For finite subsets $\mathcal{F} := \{f_1, \ldots, f_p\} \subset \mathcal{A}$ and $\mathcal{S} := \{s_1, \ldots, s_q\} \subset \mathcal{M}$, the Macaulay matrix $\mathsf{Macaulay}(\mathcal{F}, \mathcal{S})$, is defined as

$$\mathsf{Macaulay}(\mathcal{F}, \mathcal{S}) := \begin{pmatrix} \mathsf{coeff}(f_1, s_1) & \ldots & \mathsf{coeff}(f_1, s_q) \\ \vdots & \ddots & \vdots \\ \mathsf{coeff}(f_p, s_1) & \ldots & \mathsf{coeff}(f_p, s_q) \end{pmatrix}$$

Note that $\mathcal{S}$ need not have all the monomials in $\mathcal{F}$.

XL Algorithm and Gröbner Bases. Techniques based on Gröbner bases and the closely related XL algorithm [CKPS00] can be used to solve systems of polynomial equations. Both approaches usually depend on the Macaulay matrix. For the systems we consider, the XL algorithm is more performant, and hence our discussion focuses on XL.

Let $\mathcal{F} := \{f_1, \ldots, f_p\}$ such that $\mathcal{F} \subseteq \mathcal{A}$ be a system of polynomial equations, i.e., $f_1(\mathbf{x}) = \ldots = f_p(\mathbf{x}) = 0$. The XL algorithm can be split into two phases. The first phase maps the non-linear system $\mathcal{F}$ to a linear system. We start by multiplying each f_i by arbitrary monomials such that the resulting polynomials are of degree at most d. d is carefully selected and input to the algorithm such that this step produces enough new equations. Note that finding the right d is often challenging and constitutes a significant effort for the approach. We then linearize the system by treating its monomials as new variables and save their coefficients in the Macaulay matrix. The second phase is standard. We proceed by solving the linear system with Gaussian elimination and obtain a polynomial in one variable. We solve this polynomial using some factorization algorithm and obtain a root, substitute, and repeat this process to solve for the remaining variables. The full algorithm is presented in Figure 1.

As briefly discussed above, for XL to be successful we need to select d carefully such that we produce in the first phase enough linearly independent equations in relation to the number of monomials. In other words, $p \geq q$ in the Macaulay matrix so that we can apply Gaussian elimination. The threshold for large enough d is called the witness degree d_{wit} and we use [BØ23]'s definition below.

Definition 3 (Witness Degree [BØ23]). *Consider an affine system of polynomials $\mathcal{F} := \{f_1, \ldots, f_p\}$ with coefficients in $\mathbb{F}$, the ideal $\mathcal{I} := \langle \mathcal{F} \rangle$, and some $d \in \mathbb{N}$. Now consider:*

$$\mathcal{I}_{\leq d} := \{h \in \mathcal{I} : \mathsf{deg}(h) \leq d\}$$

$\mathsf{XL}(\mathcal{F} = \{f_1, \ldots, f_p\} \subset \mathcal{A}, d_{\mathsf{wit}} \in \mathbb{N})$:

1. Expand the system to degree d_{wit} by computing, $\mathcal{I}_{\leq d_{\mathsf{wit}}} := \{mf_i : m \in \mathcal{M}, f_i \in \mathcal{F}, \mathsf{deg}(mf_i) \leq d_{\mathsf{wit}}\} = \{h \in \mathcal{I} : \mathsf{deg}(h) \leq d_{\mathsf{wit}}\}$.
2. Obtain the Macaulay matrix $\mathbf{M} := \mathsf{Macaulay}(\mathcal{I}_{\leq d_{\mathsf{wit}}}, \mathcal{M}_{\leq d_{\mathsf{wit}}}) \in \mathbb{F}^{p \times q}$ where $\mathcal{M}_{\leq d_{\mathsf{wit}}} := \{m \in \mathcal{M} : \mathsf{deg}(m) \leq d_{\mathsf{wit}}\}$ and $q := |\mathcal{M}_{\leq d_{\mathsf{wit}}}|$. We now have a linear system $\mathbf{M} \cdot \mathbf{z} = 0$ where $\mathbf{z}_1, \ldots, \mathbf{z}_q$ are relabelings of the $\mathcal{M}_{\leq d_{\mathsf{wit}}}$.
3. For $i \in [n]$:
   (a) Reorder $\mathbf{M}, \mathbf{z}$ such that $(\mathbf{z}_1, \ldots, \mathbf{z}_{d_{\mathsf{wit}}+1}) = (1, \mathbf{x}_i^1, \ldots, \mathbf{x}_i^{d_{\mathsf{wit}}})$.
   (b) Perform Wiedemann Gaussian elimination [Wie86] on $\mathbf{M}$ where $(\mathbf{z}_1, \ldots, \mathbf{z}_{d_{\mathsf{wit}}+1}) = (1, \mathbf{x}_i^1, \ldots, \mathbf{x}_i^{d_{\mathsf{wit}}})$ are eliminated last. Output $\perp$ if the first row of $\mathbf{M}$ is not of the form $(\mathbf{M}_{1,1}, \ldots, \mathbf{M}_{1,d_{\mathsf{wit}}+1}, 0, \ldots 0)$.
   (c) Solve the univariate polynomial system $\sum_{j \in [d_{\mathsf{wit}}+1]} \mathbf{M}_{1,j} \mathbf{x}_i^{j-1} = 0$ using factoring algorithms. Let x_i be one of the roots.
   (d) Substitute x_i in for $\mathbf{x}_i$ and simplify $\mathbf{M}$.
4. Output $(x_1, \ldots, x_n)$.

Fig. 1. The XL algorithm [CKPS00] for solving quadratic system of equations.

$$\mathcal{J}_{\leq d} := \left\{ h \in \mathcal{I} : \exists g_i \ s.t. \ h = \sum_{i \in [p]} g_i f_i; \forall i \in [p], \mathsf{deg}(g_i) \leq d - \mathsf{deg}(f_i) \right\}$$

The witness degree d_{wit} is the smallest $d \in \mathbb{N}$ for which it holds that $\mathcal{I}_{\leq d} = \mathcal{J}_{\leq d}$ and $\mathsf{LM}(\mathcal{I}_{\leq d}) = \mathsf{LM}(\mathcal{I})$, where $\mathsf{LM}(\cdot)$ (for some graded ordering) denotes the monomial ideal generated by the leading monomials of all polynomials in the input ideal.

Intuitively, d_{wit} guarantees that everything we need to know about the structure of an ideal can be captured by polynomials of degree at most d_{wit}. Note that the witness degree d_{wit} is related to the degree of regularity d_{reg}. The key difference is that d_{reg} is usually used for homogeneous systems solved with the Gröbner basis techniques, while d_{wit} is suitable also for affine systems solved with XL.

The runtime of the XL algorithm is largely determined by the cost of Gaussian elimination. As the systems we are considering have a single solution and are sparse, we can use the Wiedemann algorithm [Wie86] to perform the Gaussian elimination, and find a solution in $3 \cdot \mathsf{maxRowWeight}(\mathbf{M}) \cdot q^2$, where 3 is a constant standard in the literature (see discussion in [BØ23]), $\mathsf{maxRowWeight}(\mathbf{M}) := \mathsf{max}(\{|\mathbf{M}_i| : i \in [p]\})$ is the maximum number of non-zero entries across the rows of the Macaulay matrix $\mathbf{M}$, and q is the number of columns in the Macaulay matrix.

3.7 Algebraic Attacks

[BØ23]'s Attack. The attack solves for the noise vector $\boldsymbol{e} \in \mathbb{F}^n$ in a polynomial system representing a single instance of regular-noise SD. Recall that a regular

$\boldsymbol{e} = \boldsymbol{e}_1 || \ldots || \boldsymbol{e}_t$ can be viewed as t vectors $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_t$ each of size n/t and Hamming weight $|\boldsymbol{e}_i| = 1$. The system consists of k linear parity check equations $\mathbf{G}\boldsymbol{e} - \boldsymbol{b} = 0$ and $\binom{n/t}{2}t$ quadratic equations encoding the regular structure of $\boldsymbol{e}$, i.e., $\boldsymbol{e}_{i,j_1}\boldsymbol{e}_{i,j_2} = 0$ for all $i \in [t]$ and $j_1 < j_2 \in [n/t]$[4]. These two sets of equations represent the complete system of polynomial equations to solve regular syndrome decoding over a large field $\mathbb{F}$. Over $\mathbb{F}_2$, we additionally encode that the sum of each block equals 1, i.e., $\sum_{j\in[n/t]} \boldsymbol{e}_{i,j} - 1 = 0$, for all $i \in [t]$. We cannot do this over larger fields as we do not know the values of the non-zero coordinates. We also include field equations $\boldsymbol{e}_{i,j}^2 - \boldsymbol{e}_{i,j} = 0$, for all $i \in [t]$ and $j \in [n/t]$. More generally, we can include $\boldsymbol{e}_{i,j}^{||\mathbb{F}||} - \boldsymbol{e}_{i,j} = 0$ for any $\mathbb{F}$. However, for large $\mathbb{F}$, the degree of the field equations is much higher than d_{wit}, and hence they have no contribution.

We note that the main contribution to the polynomial system comes from the k parity check equations. Hence, the attack is more effective for instances with a non-constant rate such as in primal LPN. The presented system is for the dual setting. To attack the primal setting, we simply convert the primal instance into an equivalent dual instance and then solve for the presented polynomial system. We now present the polynomial systems for large $\mathbb{F}$ and $\mathbb{F}_2$ formally.

Modeling 1 (**Polynomial System over a Large Field** $\mathbb{F}$)**.** *Let* $(\mathbf{G}, \boldsymbol{b})$ *be a regular syndrome decoding instance over a large* $\mathbb{F}$*. Let* $\mathcal{F} := \mathcal{R} \cup \mathcal{S}$ *be a set of polynomials such that:*

- $\mathcal{R}$ *is the set of* k *linear parity check equations* $\mathbf{G}\boldsymbol{e} - \boldsymbol{b} = 0$*.*
- $\mathcal{S}$ *is the set of* $t\binom{n/t}{2}$ *quadratic equations that encode the regularity of the noise vector* $\boldsymbol{e}$*, i.e.,* $\boldsymbol{e}_{i,j_1}\boldsymbol{e}_{i,j_2} = 0$ *for all* $i \in [t]$ *and* $j_1 < j_2 \in [n/t]$*.*

Modeling 2 (**Polynomial System over** $\mathbb{F}_2$)**.** *Let* $(\mathbf{G}, \boldsymbol{b})$ *be a regular syndrome decoding instance over* $\mathbb{F}_2$*. Let* $\mathcal{F} := \mathcal{R} \cup \mathcal{S} \cup \mathcal{V} \cup \mathcal{W}$ *be a set of polynomials such that:*

- $\mathcal{R}$ *and* $\mathcal{S}$ *are the same sets as in Modeling 1.*
- $\mathcal{V}$ *is the set of* n *field equations* $\boldsymbol{e}_{i,j}^2 - \boldsymbol{e}_{i,j} = 0$ *for all* $i \in [t]$ *and* $j \in [n/t]$*.*
- $\mathcal{W}$ *is the set of* t *linear equations* $\sum_{j\in[n/t]} \boldsymbol{e}_{i,j} - 1 = 0$*, for all* $i \in [t]$*. They express that each of the* t *blocks has Hamming weight* 1*.*

[BØ23] solve these polynomial systems with XL Wiedemann. To apply XL Wiedemann, [BØ23] estimate the witness degree d_{wit} at which the polynomial system is solved. This is [BØ23]'s key contribution. It also determines the cost of XL Wiedemann as the witness degree determines the size of the Macaulay matrix.

The d_{wit} estimate is the index of the first ≤ 0 coefficient in the Hilbert series $\mathcal{HS}_{\mathcal{A}/\mathcal{I}}(z)$, where $\mathcal{I} := \langle \mathcal{F}^{(h)} \rangle$ in Modeling 1 and Modeling 2 respectively. Thus, d_{wit} can be simply retrieved if we know the Hilbert series. Recall that unfortunately Hilbert series are often difficult to compute. By using the assumption that

[4] $\boldsymbol{e}_{i,j}$ represents jth element of ith block $\boldsymbol{e}_i$ of $\boldsymbol{e}$.

the relevant Macaulay matrices have maximal rank and using the knowledge of regular and semi-regular sequences, [BØ23] arrive at the following Hilbert series.

Theorem 1 (Hilbert Series for Modeling 1). *Assuming the Macaulay matrix has maximum rank, the Hilbert series of the homogeneous ideal* $\mathcal{I} := \langle \mathcal{F}^{(h)} \rangle$*, where* $\mathcal{F}$ *is the Modeling 1 polynomial system, is*

$$\mathcal{HS}_{\mathcal{A}/\mathcal{I}}(\mathbf{z}) := (1-\mathbf{z})^k \cdot \left(1 + \frac{n}{t} \cdot \frac{\mathbf{z}}{1-\mathbf{z}}\right)^t,$$

truncated after the first ≤ 0 *coefficient.*

Theorem 2 (Hilbert Series for Modeling 2). *Assuming the Macaulay matrix has maximum rank, the Hilbert series of the homogeneous ideal* $\mathcal{I} := \langle \mathcal{F}^{(h)} \rangle$*, where* $\mathcal{F}$ *is the Modeling 2 polynomial system, is*

$$\mathcal{HS}_{\mathcal{A}/\mathcal{I}}(\mathbf{z}) := \frac{(1 + (n/t - 1)\mathbf{z})^t}{(1+\mathbf{z})^k},$$

truncated after the first ≤ 0 *coefficient.*

As presented, the complexity of the algorithm that solves Modeling 1 and Modeling 2 is too high to be competitive with more established attacks. In other words, the witness degree is too high and needs to be reduced to potentially decrease the complexity of the overall algorithm. With that in mind, [BØ23] present a hybrid approach, which consists of repeatedly guessing a few noise-free elements of $\boldsymbol{e}$ and invoking XL Wiedemann until successfully computing $\boldsymbol{e}$. More specifically, parameterized by $f \in [t]$ and $\mu \in [n/t]$, the hybrid approach guesses μ noise-free positions in the first f blocks of $\boldsymbol{e}$ (i.e., add new equations for each guessed noise-free $\boldsymbol{e}_{i,j} = 0$ to $\mathcal{F}$). Let p be the probability that the guessed positions are all noise-free. We then expect to repeat the XL Wiedemann $O(p^{-1})$ times. The hope is that the loss from having to rerun XL Wiedemann is superseded by the decreased degree at which the system is solved. As before, the degree d_{wit} is derived from the Hilbert series, which for Modeling 1 changes to

$$\mathcal{HS}_{\mathcal{A}/\mathcal{I}}(\mathbf{z}) := \left[(1-\mathbf{z})^k \cdot \left(1 + \left(\frac{n}{t} - \mu\right) \cdot \frac{\mathbf{z}}{1-\mathbf{z}}\right)^f \cdot \left(1 + \frac{n}{t} \cdot \frac{\mathbf{z}}{1-\mathbf{z}}\right)^{t-f}\right],$$

truncated after the first ≤ 0 coefficient. For Modeling 2, it changes to

$$\mathcal{HS}_{\mathcal{A}/\mathcal{I}}(\mathbf{z}) := \left[\frac{(1 + (n/t - 1 - \mu)\mathbf{z})^f \cdot (1 + (n/t - 1)\mathbf{z})^{t-f}}{(1+\mathbf{z})^k}\right],$$

also truncated after the first ≤ 0 coefficient.

4 Overview

In this work, we introduce a new assumption that we call the stationary syndrome decoding (SSD), analyze its security, and present its implications for different applications. The high-level idea of SSD is straightforward. We consider q instances of syndrome decoding (SD). SSD states that it is secure to reuse the noisy *positions* of the noise vector $\boldsymbol{e}$ across all q instances as long as their corresponding *values* are sampled uniformly for each instance. We now define SSD formally:

Definition 4 (Stationary Syndrome Decoding (SSD)). *Stationary Syndrome Decoding is parameterized by an implicit computational security parameter κ, a field $\mathbb{F}$, dimensions $n, k, q \in \mathbb{N}$ with $k < n$, and distributions $\mathcal{L} \in \mathsf{Dist}[\{0,1\}^n]$, $\mathcal{G} \in \mathsf{Dist}[\mathbb{F}^{k \times n}]$. For sample $L \leftarrow \mathcal{L}$ and for $i \in [q]$, sample $\boldsymbol{e}_i \leftarrow L \odot \mathbb{F}^n$, $\mathbf{G}_i \leftarrow \mathcal{G} \in \mathbb{F}^{k \times n}$. The $(n, k, q, \mathbb{F}, \mathcal{L}, \mathcal{G})$-SSD assumption states:*

$$\{(\mathbf{G}_i, \boldsymbol{b}_i) \mid \boldsymbol{b}_i := \mathbf{G}_i \cdot \boldsymbol{e}_i\}_{i \in [q]} \approx \{(\mathbf{G}_i, \boldsymbol{b}_i) \mid \boldsymbol{b}_i \leftarrow \mathbb{F}^k\}_{i \in [q]}$$

where $\approx$ denotes computational indistinguishability.

The applications we consider will restrict $\mathcal{L}$ to being a subset containing sparse vectors, typically with $O(\kappa)$ ones. It is not hard to show that this definition is equivalent to the following LPN-styled definition.

Definition 5 (Stationary Learning Parity with Noise (SLPN)). *Stationary Learning Parity with Noise is parameterized by an implicit computational security parameter κ, a field $\mathbb{F}$, dimensions $n, m, q \in \mathbb{N}$ with $n > m$, and distributions $\mathcal{L} \in \mathsf{Dist}[\{0,1\}^n]$, $\mathcal{H} \in \mathsf{Dist}[\mathbb{F}^{n \times m}]$. For sample $L \leftarrow \mathcal{L}$ and for $i \in [q]$, sample $\boldsymbol{e}_i \leftarrow L \odot \mathbb{F}^n$, $\boldsymbol{A}_i \leftarrow \mathcal{H} \in \mathbb{F}^{n \times m}$ and $\boldsymbol{s}_i \leftarrow \mathbb{F}^m$. The $(n, m, q, \mathbb{F}, \mathcal{L}, \mathcal{H})$-SLPN assumption states:*

$$\{(\boldsymbol{A}_i, \boldsymbol{b}_i) \mid \boldsymbol{b}_i := \boldsymbol{A}_i \cdot \boldsymbol{s}_i + \boldsymbol{e}_i\}_{i \in [q]} \approx \{(\boldsymbol{A}_i, \boldsymbol{b}_i) \mid \boldsymbol{b}_i \leftarrow \mathbb{F}^n\}_{i \in [q]}$$

where $\approx$ denotes computational indistinguishability.

In particular, the equivalence holds when $\mathbf{G}_i$ is the generator matrix for the parity check matrix $\boldsymbol{A}_i^\mathsf{T}$, see Section 3.4. Note that the more standard definitions of regular LPN and SD can be obtained simply by restricting q to be one and requiring non-zero noise. Conversely, Definition 6 shows that one can similarly reframe SSD, SLPN in terms of the standard LPN formulation $(\mathbf{G}, \mathbf{G} \cdot \boldsymbol{e}) \approx (\mathbf{G}, \$)$ with specially structured noise $\boldsymbol{e}$ and matrices $\mathbf{G}$.

In the rest of this section, we justify at a high level SSD's security (Section 4.1) and discuss SSD's implications for the performance of pseudorandom correlation generators (Section 4.2).

4.1 The Security of SSD

The majority of known attacks on SD and LPN fall into two categories: linear and non-linear. For the parameter regime that PCGs commonly use, linear attacks are typically more efficient. Interestingly, SSD enjoys provable immunity to all of these attacks. As discussed in Section 3.5, these attacks can be shown to be equivalent to sampling the generator matrices $\mathbf{G}_1, \ldots, \mathbf{G}_q \in \mathbb{F}^{k \times n}$ and invoking an adversary $\mathcal{A}(\mathbf{G})$ which outputs a test vector $\boldsymbol{v} \in \mathbb{F}^{kq}$. The distinguisher is then implemented as $\boldsymbol{v} \cdot (\boldsymbol{b}_1 || \ldots || \boldsymbol{b}_q)$. I.e., if the output is correlated with the linear function $\boldsymbol{v} = (\boldsymbol{v}_1 || \ldots || \boldsymbol{v}_q)$, then the adversary $\mathcal{A}$ wins. We provably show no $\mathcal{A}$ exists that has noticeable advantage. The core idea is that any such attacker has to essentially come up with a codeword $\boldsymbol{v}_i$ of $\mathbf{G}_i$ that does not intersect the noise.[5] However, from the distribution of the noise, this is unlikely, even if the noise is correlated. SSD is particularly interesting because it targets a weakness of linear attacks such as information set decoding (ISD). At a high level, linear attacks come down to guessing a noise-free set of positions in $\boldsymbol{e}$, which are the non-zeros of $\boldsymbol{v}_i$, and then checking for linear correlations. However, this does not allow an attacker to take advantage of the new information that SSD provides. For a linear attacker, all $\boldsymbol{v}_i$ must be codewords, and therefore the new instances are no easier to attack than the first.

The situation with respect to non-linear attacks is more complicated. State-of-the-art techniques encode the problem statement into a system of non-linear equations and use algebraic techniques for solving the system, e.g., Gröbner bases. [BØ23] recently proved bounds on the running time of the XL [CKPS00] algorithm (see Section 3.6 and Figure 1), when applied to the SD problem with regular noise. They show that when $\mathbf{G} \in \mathbb{F}^{k \times n}$ has non-constant rate, e.g., $k = (1 - \epsilon)n$, then the XL algorithm can outperform linear attacks.

Given that our system can be seen as an even more structured version of regular noise SD, it is imperative that we understand how such attacks scale when adapted to use the additional structure. To achieve this, we define a system of equations that encodes the structure of the SSD problem and then prove bounds on the required running time to solve such systems using the XL algorithm. We show that XL is not noticeably better at solving SSD compared to SD.

4.2 Pseudorandom Correlation Generator (PCG) from SSD

We now explain at a high level how SSD improves the performance of PCGs and defer the full details to Section 7. Typically, the end goal is to generate a secret sharing of a random vector $\boldsymbol{v}$ times a scalar Δ, i.e., $[\![\boldsymbol{v}\Delta]\!]$. We will first compute a secret sharing of a t-sparse vector $\boldsymbol{e}$ times Δ, i.e., $[\![\boldsymbol{e}\Delta]\!]$. The final result is obtained by computing $[\![\boldsymbol{v}\Delta]\!] = \mathbf{G}[\![\boldsymbol{e}\Delta]\!]$, i.e., $\boldsymbol{v} = \mathbf{G}\boldsymbol{e}$. $\boldsymbol{e}, \boldsymbol{v}$ will be known to the receiver while Δ is known to the sender. Many useful degree 1 PCGs, e.g., for OT, binary OLE, and VOLE, are directly obtained from $[\![\boldsymbol{v}\Delta]\!]$.

[5] Here we abuse notation and redefine $\boldsymbol{v}_i \in \mathbb{F}^n$ as a codeword of $\mathbf{G}_i$ that corresponds to a test vector $\mathbf{G}_i \boldsymbol{v}_i$ in the SD setting. See Section 5 for formal detail.

In more detail, let $n' := n/t$. The receiver samples t subvectors $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_t \in \mathbb{F}^{n'}$ of Hamming weight 1 and defines $\boldsymbol{e} := \boldsymbol{e}_1 || \ldots || \boldsymbol{e}_t \in \mathbb{F}^n$. The secret sharing $[\![\boldsymbol{e}\Delta]\!]$ is generated by evaluating t so-called punctured PRFs [GGM84,BCG+19a] (PPRF), where the input to the ith PPRF is $\boldsymbol{e}_i$ from the receiver and Δ from the sender. The output is $[\![\boldsymbol{e}_i\Delta]\!]$ and we obtain $[\![\boldsymbol{e}\Delta]\!] := [\![\boldsymbol{e}_1\Delta]\!] || \ldots || [\![\boldsymbol{e}_t\Delta]\!]$. A single instance of such PPRF protocol requires $\log_2 n'$ oblivious transfers (OTs).

Many applications require billions of correlations. However, due to memory constraints, it is often inefficient to have $n > 2^{24}$ and as such it is common to have q PCG instances each of fixed size n, e.g., $n = 2^{20}$. This comes at the cost of requiring $qt \log_2 n/t$ OTs and communication. The SSD assumption allows us to reduce the overhead back down to $t \log_2 n/t$ OTs in total and an amortized t communication per instance (i.e., $\log_2 n/t$ times less than SD). This is because the bulk of the work in the PPRF protocol is dependent on the locations of the non-zeros in $\boldsymbol{e}$ but *not their values*. As such, because SSD states that the location does not need to change, we obtain significant savings. We additionally obtain a much more cache-friendly construction that results in significant computational savings. This is because the ability to reuse the bulk of the setup makes it more attractive to use smaller values of n, which improves the cache efficiency.

We also obtain significant improvements for degree 2 PCGs such as non-binary OLEs and Beaver triples that rely on the Ring LPN assumption. This setting requires a very expensive setup to compute a sharing of two sparse polynomials $\boldsymbol{e} \cdot \boldsymbol{e}'$. If we apply the SSD assumption to this setting, the vast majority of the setup can be reused, dramatically decreasing the overhead.

5 Linear Attacks

We now demonstrate that our assumption is resilient to linear attacks. We focus on the restricted case of regular noise for efficiency. Our argument lies in showing that the linear test framework adversary gains no significant advantage from SLPN/SSD. It will be convenient to recast SSD in terms of standard SD with structured noise and structured $\mathbf{G}$.

Definition 6 (Canonical Representation). *We say* $(n', m', \mathbb{F}, \mathcal{D}, \mathcal{H}')$-SLPN *is the Canonical Representation of* $(n, m, q, \mathbb{F}, \mathcal{L}, \mathcal{H})$-SLPN *if* $n' = nq, m' = mq, \mathcal{D} = \{\boldsymbol{e}_1 || ... || \boldsymbol{e}_q : \boldsymbol{d} \leftarrow \mathcal{L}, \boldsymbol{e}_i \leftarrow \boldsymbol{d} \odot \mathbb{F}^n\}, \mathcal{H}' = \{\mathsf{diag}(\boldsymbol{A}_1, ..., \boldsymbol{A}_q) : \boldsymbol{A}_i \leftarrow \mathcal{H}\}$ *where* diag *denotes the function that places* $\boldsymbol{A}_1, ..., \boldsymbol{A}_q$ *along the diagonal of a* $n' \times m'$ *matrix.*

Similarly, we say $(n', k', \mathbb{F}, \mathcal{L}, \mathcal{G}')$-SD *is the Canonical Representation of* $(n, k, q, \mathbb{F}, \mathcal{D}, \mathcal{G})$-SSD *if* $n' = nq, k' = kq, \mathcal{D} = \{\boldsymbol{e}_1 || ... || \boldsymbol{e}_q : \boldsymbol{d} \leftarrow \mathcal{L}, \boldsymbol{e}_i \leftarrow \boldsymbol{d} \odot \mathbb{F}^n\}, \mathcal{G}' = \{\mathsf{diag}(\mathbf{G}_1, ..., \mathbf{G}_q) : \mathbf{G}_i \leftarrow \mathcal{G}\}$.

It is not hard to show that the canonical representation is equivalent.

Definition 7 (Security against Linear Tests). *Security against Linear Tests is parameterized by an implicit security parameter* κ*, a finite field* $\mathbb{F}$*, dimensions* $n, m \in \mathbb{N}$ *with* $n > m$*, and subsets* $\mathcal{D} \subset \mathbb{F}^n$*,* $\mathcal{H} \subseteq \mathbb{F}^{n \times m}$*. We say that*

the $(n, m, \mathbb{F}, \mathcal{D}, \mathcal{H})$*-LPN assumption is secure against linear attacks*

$$\Pr[\mathit{bias}(\mathcal{D}_{\boldsymbol{A}}) > \epsilon : \boldsymbol{A} \leftarrow \mathcal{H}] \leq \delta$$

where ϵ, δ *are negligible and* $\mathcal{D}_{\boldsymbol{A}}$ *is the distribution induced by* $\boldsymbol{s} \leftarrow \mathbb{F}^m, \boldsymbol{e} \leftarrow \mathcal{D}$ *and outputting the LPN sample* $\boldsymbol{A}\boldsymbol{s} + \boldsymbol{e}$*.*

We note that one can also consider a computational version of linear test by restricting $\mathcal{D}$ to being efficient. This will then correspond to using the pseudo-minimum distance in the following.

Theorem 3 (Security of SLPN against Linear Tests with Regular Noise). *Let* $\mathbb{F}$ *be a finite field,* $\mathcal{H} \subseteq \mathbb{F}^{n \times m}$ *be a set of matrices with dual distance at least* d *with probability at least* δ*,* $\mathcal{D} \in \{0,1\}^n$ *be the set of regular weight* t *vectors, then* $(n, m, q, \mathbb{F}, \mathcal{D}, \mathcal{H})$*-SLPN is secure against attacks in the* (ϵ, δ)*-linear test framework of Definition 7 (in canonical representation) where*

$$\epsilon = (1 - d/n)^t$$

Proof. To prove SLPN secure against linear attacks, we split our proof into two cases and prove them separately. First, we consider the number of LPN instances $q = 1$ and only then $q > 1$.

Non-stationary Noise, $q = 1$. Let $d \in [n]$ be the minimum number of linearly dependent rows in $\boldsymbol{A}$. We show that there does not exist a $\boldsymbol{v}$ such that $\boldsymbol{v} \cdot (\mathbf{A}\mathbf{s} + \mathbf{e})$ is distinguishable from uniform. We consider two cases.

Non-codeword $\boldsymbol{v}$. Let us define the code $\mathcal{C} = \{\boldsymbol{c} \in \mathbb{F}^n : \boldsymbol{c}\boldsymbol{A} = 0\} = \{\boldsymbol{m}\mathbf{G} : \boldsymbol{m} \in \mathbb{F}^k\}$. This implies that all $\boldsymbol{c} \in \mathcal{C}$ are mapped to zero when multiplied from the right by $\boldsymbol{A}$, that is $\boldsymbol{c}^\mathsf{T}\boldsymbol{A} = 0^m$. Conversely, for $\boldsymbol{v} \notin \mathcal{C}$ it holds that $\boldsymbol{v}^\mathsf{T}\boldsymbol{A} \neq 0^m$. Therefore, $\boldsymbol{v}^\mathsf{T}\boldsymbol{A}\boldsymbol{s} = \boldsymbol{u}^\mathsf{T}\boldsymbol{s} = r$ where $\boldsymbol{u} \in \mathbb{F}^m$ is some non-zero vector. Since $\boldsymbol{s}$ is uniform, it follows that so is r, and therefore $\max_{v \notin \mathcal{C}}(\mathsf{bias}_{\boldsymbol{v}}(\boldsymbol{A}\boldsymbol{s} + \boldsymbol{e})) = 0$.

Codeword $\boldsymbol{v}$. As just described, when $\boldsymbol{v}$ is a codeword the randomness contributed by $\boldsymbol{s}$ vanishes. That is, $\boldsymbol{v}^\mathsf{T}(\boldsymbol{A}\boldsymbol{s}+\boldsymbol{e}) = \boldsymbol{v}^\mathsf{T}\boldsymbol{e}$. To prove that the construction is secure against linear attacks we must show that $\boldsymbol{v} \cdot \boldsymbol{e}$ has negligible bias. Let $\boldsymbol{e}_i$ and $\boldsymbol{v}_i$ denote the ith regular block of $\boldsymbol{e}, \boldsymbol{v}$, respectively, and let $\mathcal{D}_i$ denote the distribution of $\boldsymbol{e}_i$. Lemma 1 states that the overall $\mathsf{bias}_{\boldsymbol{v}}(\mathcal{D})$ is bounded as $\mathsf{bias}_{\boldsymbol{v}}(\mathcal{D}) \leq \prod_{i \in [t]} \mathsf{bias}_{\boldsymbol{v}_i}(\mathcal{D}_i)$. We have $\mathsf{bias}_{\boldsymbol{v}_i}(\mathcal{D}_i) = \left| \mathbb{E}_{\boldsymbol{e}_i \leftarrow \mathcal{D}_i}[\chi(\boldsymbol{v}_i \cdot \boldsymbol{e}_i)] \right|$. Let d_i denote the Hamming weight of $\boldsymbol{v}_i$. Recall that we have one noisy location in $\boldsymbol{e}_i$ (possibly with value zero), and therefore the probability the noisy location intersects $\boldsymbol{v}_i$ is $d_i/(n/t)$. Conditioned on intersecting, $\boldsymbol{v}_i \cdot \boldsymbol{e}_i$ is uniform over $\mathbb{F}$, and therefore the bias is 0. Otherwise, $\boldsymbol{v}_i \cdot \boldsymbol{e}_i = 0$ and $\chi(\boldsymbol{v}_i \cdot \boldsymbol{e}_i) = 1$. It follows that $\mathsf{bias}_{\boldsymbol{v}_i}(\mathcal{D}_i) = 1 - d_i/(n/t)$ and

$$\max_{\boldsymbol{v} \in \mathcal{C}} (\mathsf{bias}_v(\mathcal{D})) \leq \prod_i 1 - d_i/(n/t) \leq \left(1 - \frac{d}{n}\right)^t$$

Note that this differs from the traditional regular noise bias for $q = 1$ as we allow the noise value to be 0. For $\mathbb{F}_2$ and non-zero noise, $\Pr[\boldsymbol{v}_i \cdot \boldsymbol{e}_1 = 1] = d_i/(n/t)$ and $\chi(\boldsymbol{v}_i \cdot \boldsymbol{e}_1) = -1$. It follows that $\mathbb{E}[\chi(\boldsymbol{v}_i \cdot \boldsymbol{e}_i)] = -d_i/(n/t) + (1 - d_i/(n/t)) = 1 - 2d_i/(n/t)$ and overall $\mathsf{bias}_{\boldsymbol{v}}(\mathcal{D}) \leq (1 - 2d/n)^t$.

Stationary Noise, $q > 1$. As in the $q = 1$ case, it is clear that $\boldsymbol{v} \cdot (\boldsymbol{A}\boldsymbol{s} + \boldsymbol{e})$ is uniform when $\boldsymbol{v}$ is *not* a concatenation of q codewords. This is because $\boldsymbol{v}$ does not map $\boldsymbol{s}$ to zero. Now let $\boldsymbol{v}_1, ..., \boldsymbol{v}_q$ denote the codewords of $\boldsymbol{v}$, i.e. $\boldsymbol{v}_i \boldsymbol{A}_i = 0^m$ and $\boldsymbol{v} = (\boldsymbol{v}_1 || ... || \boldsymbol{v}_q)$. As in the $q = 1$ case, if $\boldsymbol{v}$ intersects $\boldsymbol{e}$, then the result is uniformly random, and therefore has zero bias. Since (a) $\boldsymbol{v}$ is non-zero, and hence at least one $\boldsymbol{v}_i$ is non-zero, and (b) the bias of each block with non-zero $\boldsymbol{v}_i$ is at most $\left(1 - \frac{d}{n}\right)^t$, then the overall bias is at most this as well. Note that while the noisy locations between blocks do not change, their values in each block are uniformly random, and the overall bias is bounded as the maximum over the bias of each non-zero block. □

Note that given Theorem 3 and the equivalence of SLPN and SSD (see Section 4), the security of SSD against linear tests with regular noise is straightforward.

5.1 Other Linear Attacks

Given that our assumption introduces additional structure it is worth considering the existence of other attacks that could be considered linear while not fit into the linear test framework. Consider the SSD problem and the q outputs $\boldsymbol{b}_1, ..., \boldsymbol{b}_q$, i.e. $\boldsymbol{b}_i = \mathbf{G}_i \boldsymbol{A}_i \boldsymbol{s}_i + \mathbf{G}_i \boldsymbol{e}_i$ for generator $\mathbf{G}_i$ of parity check $\boldsymbol{A}_i$. There exists a hidden subset $\mathbf{G}'_1, ..., \mathbf{G}'_q$ of $\mathbf{G}_1, ..., \mathbf{G}_q$ where $\mathbf{G}'_i := \left(\mathbf{G}_{i,*,L_1}, \ldots \mathbf{G}_{i,*,L_t}\right) \in \mathbb{F}^{m \times t}$ and $\mathbf{G}_{i,*,L_j}$ is the jth "noisy column" of $\mathbf{G}_i$. We then have $\boldsymbol{b}_i = \mathbf{G}'_i \cdot \boldsymbol{c}_i$ where $\boldsymbol{c}_i \in \mathbb{F}^t$ are the t noisy values in $\boldsymbol{e}_i$, i.e. $\boldsymbol{c}_{i,j} = \boldsymbol{e}_{i,L_j}$. Similarly, we can write this as one large linear equation $\boldsymbol{b} = \mathbf{G}' \cdot \boldsymbol{c}$. with $\mathbf{G}' = \mathsf{diag}(\mathbf{G}'_1, ..., \mathbf{G}'_t)$. Given that $|\boldsymbol{b}| > |\boldsymbol{c}|$, the pseudorandomness of $\boldsymbol{b}$ clearly depends on $\mathbf{G}'$ being hidden. We consider a special case where pseudorandomness breaks down.

Consider instantiating the SSD assumption where the space of generator matrices is the singleton set $\mathcal{G} = \{\mathbf{G}\}$. Therefore $\mathbf{G}_1, ..., \mathbf{G}_q$ above are all the same. This additionally means that $\mathbf{G}'$ can be expressed as an $m \times t$ matrix as opposed to a $qm \times qt$ matrix. In particular, consider $\mathbf{C} = (\boldsymbol{c}_1, ..., \boldsymbol{c}_q), \mathbf{B} = (\boldsymbol{b}_1, ..., \boldsymbol{b}_q)$ and observe that $\mathbf{B} = \mathbf{G}' \cdot \mathbf{C}$. Although the subset corresponding to $\mathbf{G}'$ is not known, $\mathbf{G}'$ is now fixed and does not grow with q. Therefore, when $q = t$, we can assume that $\mathsf{span}(\mathbf{B}) = \mathsf{span}(\mathbf{G}')$. This also means that $\boldsymbol{b}_{t+1} \in \mathsf{span}(\boldsymbol{b}_1, ..., \boldsymbol{b}_t)$, i.e. $\boldsymbol{b}_{t+1}$ can be distinguished. Because $m < t$, one would not expect this to happen for random $\boldsymbol{b}_i$.

Although this attack is clearly in some sense linear, it is not possible to perform it in the linear attack framework. Determining the coefficients $\boldsymbol{v} \in \mathbb{F}^t$ such that $\boldsymbol{b}_{t+1} = \langle (\boldsymbol{b}_1, ..., \boldsymbol{b}_t, \boldsymbol{v}) \rangle$ is a function of the output, which the linear test adversary does not have. Linear test adversaries must first fix $\boldsymbol{v}$ before seeing $\boldsymbol{b}$. This exemplifies that although the linear test framework captures the majority of traditional attacks on LPN and SD, new attacks become possible when additional

structure is added. To prevent this relatively trivial attack, it is critical that the family of codes be large and relatively uncorrelated.

Consider $\mathcal{G} = \mathbb{F}^{m \times n}$, i.e. the uniform distribution. Then clearly the linear attack $\mathcal{A}(\boldsymbol{b}_1, ..., \boldsymbol{b}_{t+1})$ described above is impossible. Each new $\boldsymbol{b}_i$ is the sum of an independent and uniformly random subset $\mathbf{G}'_i$. Any adversary breaking security must crucially rely on the fact that the $\mathbf{G}'_i$ matrices are correlated via the secret L and public parameters $\mathbf{G}_1, ..., \mathbf{G}_q$.

More generally, existing codes used for PGCs [BCG+19a,BCG+22,RRT23] all sample their codes from an extremely large sample space. In particular, many more random bits are used to sample $\mathbf{G}_i$ than are present in $\boldsymbol{b}_i$. This suggests that it is extremely unlike that linear correlations would exist. Given that each is constructed using a randomly sampled seed, e.g. $\mathbf{G}_i = \mathsf{CodeGen}(\mathsf{H}(i))$ for a random oracle H, then such linear attacks in the output should not be feasible.

6 Algebraic Attacks

In this section, we show that SSD is resilient against the recent algebraic attack of [BØ23]. We first modify [BØ23]'s attack so that it takes advantage of the stationary noise across instances. We then show that the structured stationary noise provides negligible advantage. More specifically, we start by presenting our modified system in Section 6.1. We continue by computing our system's Hilbert series in Section 6.2, use it to estimate the witness degree in Section 6.3, repeat the same procedure for the hybrid approach in Section 6.4, and then evaluate the attack's impact on SSD's security in Section 6.5.

6.1 Formulating Our Polynomial System

Recall that we restrict the noise to be at locations $L \in \mathcal{L}$ such that the locations are regular. Consider arbitrary matrices $\mathbf{G}_i \in \mathbb{F}^{k \times n}$ sampled from $\mathcal{G}$ and error vectors $\boldsymbol{e}_i \in \mathbb{F}^n$, where each $\boldsymbol{e}_i \leftarrow L \odot \mathbb{F}^n$ is a vector with *at most* t non-zeros. That is, $\boldsymbol{e}_i = \boldsymbol{e}_{i,1} || \ldots || \boldsymbol{e}_{i,t}$ where $||\boldsymbol{e}_{i,j}|| = n/t$ and $|\boldsymbol{e}_{i,j}| \leq 1$. We are given

$$(\mathbf{G}_1, \ldots, \mathbf{G}_q, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_q),$$

where $\boldsymbol{b}_i = \mathbf{G}_i \cdot \boldsymbol{e}_i$.

We now formulate a system of polynomial equations to solve for $\boldsymbol{e}_i$ for $i \in [q]$, i.e., the system has qn unknowns. We first consider in Modeling 3 the case where SSD is parameterized over a large field $\mathbb{F}$, and then in Modeling 4 the case where it is parameterized over $\mathbb{F}_2$.

Modeling 3 (SSD **over a Large Field** $\mathbb{F}$)**.** *Let* $(\mathbf{G}_1, \ldots, \mathbf{G}_q, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_q)$ *be an* SSD *instance with regular noise locations* L *over a large* $\mathbb{F}$*. Let* $\mathcal{F} := \mathcal{R} \cup \mathcal{S}$ *be a set of polynomials such that:*

- $\mathcal{R}$ *is the linear equations* $\langle \mathbf{G}_{i,j}, \mathbf{e}_i \rangle - \boldsymbol{b}_{i,j} = 0$*, for all* $i \in [q]$ *and* $j \in [k]$*.*

– $\mathcal{S}$ *is the set of* $q^2t\binom{n/t}{2} = q^2n^2/2t - q^2n/2$ *quadratic equations* $\mathbf{e}_{i,v,j}\mathbf{e}_{i',v,j'} = 0$*, for all* $i, i' \in [q], v \in [t]$*, and* $j < j' \in [n/t]$*, implied by the structured noise constraint* L*.*

Modeling 4 optimizes the system for $\mathbb{F}_2$ by including the field equations $\mathbf{e}_{i,v,j}^2 - \mathbf{e}_{i,v,j} = 0$, for all $i \in [q]$, $v \in [t]$, and $j \in [n/t]$. This ensures that the ideal $\langle \mathcal{F} \rangle$ generated by Modeling 4 is zero-dimensional. However, for large $\mathbb{F}$, as noted prior, these equations will have no contribution.

Modeling 4 (SSD **over** $\mathbb{F}_2$)**.** *Let* $(\mathbf{G}_1, \ldots, \mathbf{G}_q, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_q)$ *be a* SSD *instance with regular noise over* $\mathbb{F}_2$*. Let* $\mathcal{F} := \mathcal{R} \cup \mathcal{S} \cup \mathcal{V}$ *be a set of polynomials such that:*

– $\mathcal{R}$ *and* $\mathcal{S}$ *are the same sets as in Modeling 3.*
– $\mathcal{V}$ *is the field equations* $\mathbf{e}_{i,v,j}^2 - \mathbf{e}_{i,v,j} = 0$ *for all* $i \in [q]$*,* $v \in [t]$*, and* $j \in [n/t]$*.*

Compared to the prior work modeling for binary fields (Modeling 2), we observe that SSD appears slightly harder because we no longer restrict $\boldsymbol{e}$ to being regular with exact weight t (i.e. the noisy positions are sampled uniformly, and thus can be zero). This eliminates t linear equations that prior attacks [ES24,BØ23] were able to use.

6.2 Computing Hilbert Series

We now compute the Hilbert series (Section 3.6) of the homogeneous ideals associated with Modeling 3 and Modeling 4. Our computation uses a template similar to [BØ23] that estimates the Hilbert series of the ideals associated with Modeling 1 and Modeling 2. Note that $\mathcal{F}$, neither in the case of a large $\mathbb{F}$ nor $\mathbb{F}_2$, is a regular or semi-regular system (Section 3.6). To see why, consider $f_1 := \mathbf{e}_{1,1,1}\mathbf{e}_{1,1,2}$ and $f_2 := \mathbf{e}_{1,1,2}\boldsymbol{e}_{1,1,3}$ that come from the structured noise constraint $\mathcal{S}$. Recall that the regularity and semi-regularity of $\mathcal{F}$ is independent of the order in which we consider the polynomials in the system. Now, $\mathbf{e}_{1,1,1}f_2 = 0$ in $\mathcal{A}/\langle f_1 \rangle$, but $\mathbf{e}_{1,1,1} \neq 0$ in $\mathcal{A}/\langle f_1 \rangle$, and hence $\mathcal{F}$ is not a regular system. If $\mathcal{F}$ were to be a semi-regular system, then its degree of regularity can be no more than 3 using the aforementioned examples of f_1, f_2. However, $\mathbf{e}_{1,1,1}\mathbf{e}_{1,2,1}\mathbf{e}_{1,3,1} \notin \langle \mathcal{F} \rangle$ (with high probability). Thus, as long as $t \geq 3$ (or even $q \geq 3$, etc.), the system is not semi-regular either. As a result, we cannot use the Hilbert series from Section 3.6 and require a more sophisticated analysis.

Hilbert Series for Modeling 3. Recall Modeling 3's polynomial system $\mathcal{F} := \mathcal{R} \cup \mathcal{S}$. We first compute by monomial counting the Hilbert series $\mathcal{HS}_{\mathcal{A}/\langle \mathcal{S} \rangle}(\mathbf{z})$ of the quotient ring $\mathcal{A}/\langle \mathcal{S} \rangle$ resulting from the structured noise equations $\mathcal{S}$. After we compute the Hilbert series corresponding to $\mathcal{S}$, we need to incorporate the linear parity-check equations $\mathcal{R}$ to get the final Hilbert series. To do that, we follow [BØ23]'s approach. We formalize as Assumption 1 an assumption that the parity-check equations $\mathcal{R}$ behave well in the quotient ring $\mathcal{A}/\langle \mathcal{S} \rangle$ formed by the structured noise equations $\mathcal{S}$. Note its similarity with the definition of semi-regularity over large $\mathbb{F}$.

Assumption 1. *Let $\mathcal{F} := \mathcal{R} \cup \mathcal{S}$ be an instance of Modeling 3 and d_{reg} be the degree of regularity of the zero-dimensional ideal $\langle \mathcal{F} \rangle$. Let $\mathcal{R}^{(\mathsf{h})} := \{r_1, \ldots, r_{qk}\}$ be the set of homogenized parity check equations. Our assumption states that for $i \in [qk]$; if $gr_i = 0$ in $\mathcal{A}/\langle \mathcal{S}, r_1, \ldots, r_{i-1} \rangle$ with $\mathsf{deg}(gr_i) < d_{\mathsf{reg}}$, then $g = 0$ in $\mathcal{A}/\langle \mathcal{S}, r_1, \ldots, r_{i-1} \rangle$.*

We now state and derive the final Hilbert series.

Theorem 4. *For $\mathcal{F}$ associated with Modeling 3, the Hilbert series of the homogeneous ideal $\langle \mathcal{F}^{(\mathsf{h})} \rangle$ under Assumption 1 is*

$$\mathcal{HS}_{\mathcal{A}/\langle \mathcal{F}^{(\mathsf{h})} \rangle}(\mathbf{z}) = (1-\mathbf{z})^{qk} \cdot \left(1 + \frac{n}{t}\left(\frac{1}{(1-\mathbf{z})^q} - 1\right)\right)^t,$$

truncated after the first ≤ 0 coefficient. We call $(1-\mathbf{z})^{qk} \cdot \left(1 + \frac{n}{t}\left(\frac{1}{(1-\mathbf{z})^q} - 1\right)\right)^t$ the generating series of $\langle \mathcal{F}^{(\mathsf{h})} \rangle$.

Theorem 4 immediately follows from the proofs of Lemma 2 and Lemma 3.

Lemma 2. *For the set $\mathcal{S}$ associated with the structural equations of Modeling 3, the Hilbert series of the homogeneous ideal $\langle \mathcal{S}^{(\mathsf{h})} \rangle$ is*

$$\mathcal{HS}_{\mathcal{A}/\langle \mathcal{S}^{(\mathsf{h})} \rangle}(\mathbf{z}) = \left(1 + \frac{n}{t}\left(\frac{1}{(1-\mathbf{z})^q} - 1\right)\right)^t.$$

Proof. Note that $\mathcal{S}$ is already homogenized, i.e., $\mathcal{S} = \mathcal{S}^{(\mathsf{h})}$. Recall that $\mathcal{HS}(\mathbf{z}) = \sum_d \mathcal{HF}(d) \cdot \mathbf{z}^d$ and the $\mathcal{HF}(d) = \mathsf{dim}(\mathcal{A}_d/\langle \mathcal{S}^{(\mathsf{h})} \rangle_d)$ is the size of the vector space basis $B_d \subset \{[\mathbf{x}^{\boldsymbol{\alpha}}] : \boldsymbol{\alpha} \in \mathbb{N}^n\}$ s.t. $\mathsf{span}(B_d) = \mathcal{A}_d/\langle \mathcal{S}^{(\mathsf{h})} \rangle_d$.

Let us first restrict our attention to a specific block $v \in [t]$ and let $\mathcal{S}_v$ be the subset of $\mathcal{S}$ that only considers block v. Because $\mathcal{S}_v = \{\mathbf{e}_{i,v,j}\mathbf{e}_{i',v,j'} : i, i' \in [q], j < j' \in [n/t]\}$, the quotient $\mathcal{A}/\langle \mathcal{S}_v^{(\mathsf{h})} \rangle$ cannot contain any monomials with $\mathbf{e}_{i,v,j}\mathbf{e}_{i',v,j'}$ as a factor. That is, for a fixed v no monomial contains more than one j index. If we then consider a specific degree d and all of the q instances, the admissible monomials are $B_{d,v} := \left\{\prod_{i\in[q]} \mathbf{e}_{i,v,j}^{\alpha_i} : j \in [n/t], \boldsymbol{\alpha} \in (\mathbb{N} \cup \{0\})^q, d = \sum_{i\in[q]} \alpha_i\right\}$. Let us count the number of such monomials for a specific $v \in [t], j \in [n/t], d > 0$. We will use a balls in bins argument to count $|\{\boldsymbol{\alpha} \in (\mathbb{N} \cup \{0\})^q : d = \sum_k \alpha_k\}|$. One can view this as counting the ways to assign d balls into q bins, i.e., bins $\alpha_1, \ldots, \alpha_q$. The number of ways to do this is $\binom{q+d-1}{d}$ and hence the number of monomials of the form $\prod_{i\in[q]} \mathbf{e}_{i,v,j}^{\alpha_i}$ for a fixed v, j is $\binom{q+d-1}{d}$. Therefore, if we consider all $j \in [n/t]$ for the fixed block v, we have $|B_{d,v}| = \frac{n}{t} \cdot \binom{q+d-1}{d}$. Thus, the Hilbert series "for one block," say v, is

$$\mathcal{HS}_{\mathcal{A}/\langle \mathcal{S}_v^{(\mathsf{h})} \rangle}(\mathbf{z}) = 1 + \frac{n}{t} \cdot \sum_{d=1}^{\infty} \binom{q+d-1}{d} \mathbf{z}^d$$

where the one is from the constant monomial. We can find a closed form expression for the infinite sum inductively as follows. Consider $q = 1$. After substituting

$q = 1$ into the infinite sum in $\mathcal{HS}_{\mathcal{A}/\langle \mathcal{S}^{(\mathsf{h})}\rangle, v}$, we get $\eta_1 = \sum_{d=1}^{\infty} \binom{1+d-1}{d}\mathbf{z}^d = \frac{\mathbf{z}}{1-\mathbf{z}}$. We claim that for arbitrary q, $\eta_q = \frac{1}{(1-\mathbf{z})^q} - 1$, which clearly holds for $q = 1$. Now, in pursuit of the inductive step, consider

$$\begin{aligned}
\mathbf{z} + \eta_q &= \mathbf{z} + \sum_{d=1}^{\infty} \binom{q+d-1}{d}\mathbf{z}^d \\
&= \mathbf{z} + q\mathbf{z} + \binom{q+1}{2}\mathbf{z}^2 + \dots \\
&= \left((q+1)\mathbf{z} + \binom{q+2}{2}\mathbf{z}^2 + \dots\right) - \left((q+1)\mathbf{z}^2 + \binom{q+2}{2}\mathbf{z}^3 + \dots\right) \quad (3) \\
&= \sum_{d=1}^{\infty} \binom{q+d}{d}\mathbf{z}^d - \sum_{d=1}^{\infty} \binom{q+d}{d}\mathbf{z}^{d+1} \\
&= (1-\mathbf{z}) \sum_{d=1}^{\infty} \binom{q+d}{d}\mathbf{z}^d \\
&= (1-\mathbf{z})\eta_{q+1}
\end{aligned}$$

where (3) follows from $\binom{q+d}{d} - \binom{q+d-1}{d-1} = \binom{q+d-1}{d}$. Therefore, $\eta_{q+1} = \frac{\mathbf{z}+\eta_q}{1-\mathbf{z}} = \frac{1}{(1-\mathbf{z})^{q+1}} - 1$. Hence, $\mathcal{HS}_{\mathcal{A}/\langle \mathcal{S}_v^{(\mathsf{h})}\rangle}(\mathbf{z}) = 1 + \frac{n}{t}\left(\frac{1}{(1-\mathbf{z})^q} - 1\right)$. Finally, a general monomial of degree d is a product of monomials for distinct blocks with the sum of their degrees equal to d. Relying on the same symbolic argument as in [FS09b], which gives the generating series of a Cartesian product, we have

$$\mathcal{HS}_{\mathcal{A}/\langle \mathcal{S}^{(\mathsf{h})}\rangle}(\mathbf{z}) = \left(1 + \frac{n}{t}\left(\frac{1}{(1-\mathbf{z})^q} - 1\right)\right)^t$$

□

Lemma 3. *For $\mathcal{F}$ associated with Modeling 3, the Hilbert series of the homogeneous ideal $\langle \mathcal{F}^{(\mathsf{h})}\rangle$ under Assumption 1 is*

$$\mathcal{HS}_{\mathcal{A}/\langle \mathcal{F}^{(\mathsf{h})}\rangle}(\mathbf{z}) = (1-\mathbf{z})^{qk} \cdot \mathcal{HS}_{\mathcal{A}/\langle \mathcal{S}^{(\mathsf{h})}\rangle}(\mathbf{z}),$$

truncated after the first ≤ 0 coefficient.

Proof. Our proof is a straightforward modification and expansion of [BØ23]. Let $\mathcal{R}^{(\mathsf{h})} := \{r_1, \dots, r_{qk}\}$ be the set of homogenized parity check equations. Let $\mathcal{I}(0)$ denote the ideal $\langle \mathcal{S}^{(\mathsf{h})}\rangle$ and $\mathcal{I}(j)$, $j \in [qk]$, denote the ideal $\langle \mathcal{S}^{(\mathsf{h})}, r_1, \dots, r_j\rangle$. We will show Assumption 1 implies that for $j \in [qk]$, $d < d_{\mathsf{reg}}$, there exists a short exact sequence

$$0 \to \mathcal{A}_{d-1}/\mathcal{I}(j-1)_{d-1} \xrightarrow{\alpha} \mathcal{A}_d/\mathcal{I}(j-1)_d \xrightarrow{\beta} \mathcal{A}_d/\mathcal{I}(j)_d \to 0.$$

We prove this as follows. Let $\alpha : \mathcal{A}_{d-1}/\mathcal{I}(j-1)_{d-1} \to \mathcal{A}_d/\mathcal{I}(j-1)_d$ be the map defined as

$$\alpha([f]) := [r_j f].$$

Firstly, α is well-defined as if $f_1 \in [f]$, then $f_1 = f + g$ for some $g \in \mathcal{I}(j-1)_{d-1}$. Then, $r_j f_1 = r_j f + r_j g$. By definition, $r_j g \in \mathcal{I}(j-1)_d$ since r_j is homogenous and linear. Therefore, $[r_j f_1] = [r_j f]$. Secondly, α is injective. Suppose $\alpha([f]) = \alpha([f'])$, i.e., $[r_j f] = [r_j f']$. This implies that $r_j f - r_j f' \in \mathcal{I}(j-1)_d$, which in turn from Assumption 1 implies that $f - f' \in \mathcal{I}(j-1)_{d-1}$, i.e., $[f] = [f']$.

Let $\beta : \mathcal{A}_d/\mathcal{I}(j-1)_d \to \mathcal{A}_d/\mathcal{I}(j)_d$ be the map defined as

$$\beta([f]) := [f].$$

Again, β is well-defined as if $f_1 \in [f]$, then $f_1 - f \in \mathcal{I}(j-1)_d$. Since $\mathcal{I}(j-1)_d \subseteq \mathcal{I}(j)_d$, $f_1 - f \in \mathcal{I}(j)_d$ and hence $[f_1] = [f]$. Also, β is trivially surjective as $[f]$ is mapped to $[f]$.

Finally, we claim that $\mathsf{ker}(\beta) = \mathsf{im}(\alpha)$. To this end, we prove two statements. First, $\mathsf{im}(\alpha) \subseteq \mathsf{ker}(\beta)$. This is because if $[h] \in \mathsf{im}(\alpha)$, there exists an f such that $[h] = [r_j f]$. Now, $\beta([h]) = \beta([r_j f]) = [0]$ as $r_j \in \mathcal{I}(j)$. Next, $\mathsf{ker}(\beta) \subseteq \mathsf{im}(\alpha)$. Suppose $[h] \in \mathsf{ker}(\beta)$. Then, $\beta([h]) = [h] = [0]$, i.e., $h \in \mathcal{I}(j)_d$. This implies that there exist $g \in \mathcal{S}^{(\mathsf{h})}$ and $f_1, \ldots, f_j$ such that $h = f_1 r_1 + \ldots + f_j r_j + g$. Let g' denote the homogenous degree-d part of g and $f_1', \ldots, f_j'$ denote the homogenous degree-$(d-1)$ parts of $f_1, \ldots, f_j$. Then, $h = f_1' r_1 + \ldots + f_j' r_j + g$, i.e, $h - f_j' r_j \in \mathcal{I}(j-1)_d$. Therefore, $[h] = [r_j f_j']$ and hence $\alpha([f_j']) = [h]$, i.e., $[h] \in \mathsf{im}(\alpha)$.

Therefore, $0 \to \mathcal{A}_{d-1}/\mathcal{I}(j-1)_{d-1} \xrightarrow{\alpha} \mathcal{A}_d/\mathcal{I}(j-1)_d \xrightarrow{\beta} \mathcal{A}_d/\mathcal{I}(j)_d \to 0$ is indeed a short exact sequence. By the Hilbert function property for short exact sequences (Section 3.6), we have

$$\mathcal{HF}_{\mathcal{A}/\mathcal{I}(j-1)}(d-1) - \mathcal{HF}_{\mathcal{A}/\mathcal{I}(j-1)}(d) + \mathcal{HF}_{\mathcal{A}/\mathcal{I}(j)}(d) = 0.$$

Let $h_{d,j} := \mathcal{HF}_{\mathcal{A}/\mathcal{I}(j)}(d) = \mathsf{dim}(\mathcal{A}_d/\mathcal{I}(j)_d)$. Then, $h_{d,j} := h_{d,j-1} - h_{d-1,j-1}$. Let $\mathcal{G}_j$ be the generating series for $h_{d,j}$, i.e., let $\mathcal{G}_j(\mathbf{z}) = \sum_{d=0}^{\infty} h_{d,j}\mathbf{z}^d$. Note that $\mathcal{G}_j = \mathcal{HS}_{\mathcal{A}/\mathcal{I}(j)}$. We have

$$\begin{aligned}
\mathcal{G}_j(\mathbf{z}) &= \sum_{d=0}^{\infty} h_{d,j}\mathbf{z}^d \\
&= \sum_{d=0}^{\infty} (h_{d,j-1} - h_{d-1,j-1})\mathbf{z}^d \\
&= \sum_{d=0}^{\infty} h_{d,j-1}\mathbf{z}^d - \sum_{d=1}^{\infty} h_{d-1,j-1}\mathbf{z}^d \\
&= \sum_{d=0}^{\infty} h_{d,j-1}\mathbf{z}^d - \mathbf{z} \cdot \sum_{d=0}^{\infty} h_{d,j-1}\mathbf{z}^d \\
&= (1-\mathbf{z}) \cdot \mathcal{G}_{j-1}(\mathbf{z}).
\end{aligned}$$

Therefore, we get $\mathcal{HS}_{\mathcal{A}/\mathcal{I}(qk)}(\mathbf{z}) = (1-\mathbf{z})^{qk} \cdot \mathcal{HS}_{\mathcal{A}/\langle \mathcal{S}^{(\mathsf{h})} \rangle}(\mathbf{z})$. Note that by definition $\mathcal{I}(qk) = \langle \mathcal{S}^{(\mathsf{h})}, r_1, \ldots, r_{qk} \rangle = \langle \mathcal{F}^{(\mathsf{h})} \rangle$. Hence, the lemma follows. □

Hilbert Series for Modeling 4. Recall that Modeling 4 adds additional structural equations $\mathcal{V}$ to $\mathcal{F}$. We adopt a similar assumption as with Modeling 3. Note its similarity with the semi-regularity over $\mathbb{F}_2$.

Assumption 2. *Let $\mathcal{F} := \mathcal{R} \cup \mathcal{S} \cup \mathcal{V}$ be an instance of Modeling 4 and d_{reg} the degree of regularity of $\langle \mathcal{F}^{(\mathsf{h})} \rangle$. Let $\mathcal{R}^{(\mathsf{h})} := \{r_1, \ldots, r_{qk}\}$ be the set of homogenized parity check equations. Our assumption states that for $i \in [qk]$, if $gr_i = 0$ in $\mathcal{A}/\langle \mathcal{S}, \mathcal{V}, r_1, \ldots, r_{i-1} \rangle$ with $\mathsf{deg}(gr_i) < d_{\mathsf{reg}}$, then $g = 0$ in $\mathcal{A}/\langle \mathcal{S}, \mathcal{V}, r_1, \ldots, r_i \rangle$.*

We now state and derive the final Hilbert series.

Theorem 5. *For $\mathcal{F}$ associated with Modeling 4, the Hilbert series of the homogeneous ideal $\langle \mathcal{F}^{(\mathsf{h})} \rangle$ under Assumption 2 is*

$$\mathcal{HS}_{\mathcal{A}/\langle \mathcal{F}^{(\mathsf{h})} \rangle}(\mathbf{z}) = \frac{1}{(1+\mathbf{z})^{qk}} \cdot \left(1 + \frac{n}{t} \cdot ((1+\mathbf{z})^q - 1)\right)^t,$$

truncated after the first ≤ 0 coefficient. We call $\frac{1}{(1+\mathbf{z})^{qk}} \cdot \left(1 + \frac{n}{t} \cdot ((1+\mathbf{z})^q - 1)\right)^t$ the generating series of $\langle \mathcal{F}^{(\mathsf{h})} \rangle$.

Theorem 5 immediately follows from the proofs of Lemma 4 and Lemma 5.

Lemma 4. *For the sets $\mathcal{S}$ and $\mathcal{V}$ associated with the structural and field equations of Modeling 4, the Hilbert series of the homogeneous ideal $\langle \mathcal{S}^{(\mathsf{h})}, \mathcal{V}^{(\mathsf{h})} \rangle$ is*

$$\mathcal{HS}_{\mathcal{A}/\langle \mathcal{S}^{(\mathsf{h})}, \mathcal{V}^{(\mathsf{h})} \rangle}(\mathbf{z}) = \left(1 + \frac{n}{t} \cdot ((1+\mathbf{z})^q - 1)\right)^t.$$

Proof. Note that $\mathcal{S}$ is already homogenized, i.e., $\mathcal{S} = \mathcal{S}^{(\mathsf{h})}$. $\mathcal{V}^{(\mathsf{h})}$ contains the qn monomials $\mathbf{e}_{i,v,j}^2$ for $i \in [q], v \in [t], j \in [n/t]$. Recall that $\mathcal{HS}(\mathbf{z}) = \sum_d \mathcal{HF}(d) \cdot \mathbf{z}^d$ and the $\mathcal{HF}(d) = \mathsf{dim}(\mathcal{A}_d/\langle \mathcal{S}^{(\mathsf{h})}, \mathcal{V}^{(\mathsf{h})} \rangle_d)$ is the size of the vector space basis $B_d \subset \{[\mathbf{x}^{\boldsymbol{\alpha}}] : \boldsymbol{\alpha} \in \mathbb{N}^n\}$ s.t. $\mathsf{span}(B_d) = \mathcal{A}_d/\langle \mathcal{S}^{(\mathsf{h})}, \mathcal{V}^{(\mathsf{h})} \rangle_d$.

Let us first restrict our attention to a specific block $v \in [t]$ and let $\mathcal{S}_v, \mathcal{V}_v$ be the subsets of $\mathcal{S}, \mathcal{V}$ that only consider block v. Because $\mathcal{S}_v = \{\mathbf{e}_{i,v,j}\mathbf{e}_{i',v,j'} : i, i' \in [q], j < j' \in [n/t]\}$ and $\mathcal{V}_v^{(\mathsf{h})} = \{\mathbf{e}_{i,v,j}^2 : i \in [q], j \in [n/t]\}$, the quotient $\mathcal{A}/\langle \mathcal{S}_v^{(\mathsf{h})}, \mathcal{V}_v^{(\mathsf{h})} \rangle$ cannot contain any monomials with $\mathbf{e}_{i,v,j}\mathbf{e}_{i',v,j'}$ or $\mathbf{e}_{i,v,j}^2$ as a factor. That is, for a fixed v no monomial contains squares or more than one j index. If we then consider a specific degree d and all of the q instances, the admissible monomials are

$$B_{d,v} := \left\{ \prod_{i \in [q]} \mathbf{e}_{i,v,j}^{\alpha_i} : j \in [n/t], \boldsymbol{\alpha} \in \{0,1\}^q, d = \sum_{i \in [q]} \alpha_i \right\}.$$

We have $|\{\boldsymbol{\alpha} \in \{0,1\}^q : d = \sum_k \alpha_k\}|$. One can view $|\{\boldsymbol{\alpha} \in \{0,1\}^q : d = \sum_k \alpha_k\}|$ as counting the ways to pick d items from a total of q, i.e., which d of $\alpha_1, \ldots, \alpha_q$ must be 1 (and the rest 0). The number of ways to do this is $\binom{q}{d}$ and hence the number of monomials of the required form for a fixed v, j is $\binom{q}{d}$. Therefore, if we

consider all $j \in [n/t]$ for the fixed block v, we have $|B_{d,v}| = \frac{n}{t} \cdot \binom{q}{d}$. Thus, the Hilbert series "for one block," say v, is

$$\mathcal{HS}_{\mathcal{A}/\langle \mathcal{S}_v^{(\mathsf{h})}, \mathcal{V}_v^{(\mathsf{h})} \rangle}(\mathbf{z}) = 1 + \frac{n}{t} \cdot \sum_{d=1}^{\infty} \binom{q}{d} \mathbf{z}^d = 1 + \frac{n}{t} \cdot ((1+\mathbf{z})^q - 1)$$

where the one is from the constant monomial.

Finally, a general monomial of degree d is a product of monomials for distinct blocks with the sum of their degrees equal to d. Relying on the same symbolic argument as in [FS09b], which gives the generating series of a Cartesian product, we have

$$\mathcal{HS}_{\mathcal{A}/\langle \mathcal{S}^{(\mathsf{h})}, \mathcal{V}^{(\mathsf{h})} \rangle}(\mathbf{z}) = \left(1 + \frac{n}{t} \cdot ((1+\mathbf{z})^q - 1)\right)^t$$

□

Lemma 5. *For $\mathcal{F}$ associated with Modeling 4, the Hilbert series of the homogeneous ideal $\langle \mathcal{F}^{(\mathsf{h})} \rangle$ under Assumption 2 is*

$$\mathcal{HS}_{\mathcal{A}/\langle \mathcal{F}^{(\mathsf{h})} \rangle}(\mathbf{z}) = \frac{\mathcal{HS}_{\mathcal{A}/\langle \mathcal{S}^{(\mathsf{h})} \rangle}(\mathbf{z})}{(1+\mathbf{z})^{qk}},$$

truncated after the first ≤ 0 coefficient.

Proof. We proceed as in the proof of Lemma 3. Let $\mathcal{R}^{(\mathsf{h})} := \{r_1, \ldots, r_{qk}\}$ be the set of homogenized parity check equations. Let $\mathcal{I}(0)$ denote the ideal $\langle \mathcal{S}^{(\mathsf{h})}, \mathcal{V}^{(\mathsf{h})} \rangle$ and $\mathcal{I}(j)$, $j \in [qk]$, denote the ideal $\langle \mathcal{S}^{(\mathsf{h})}, \mathcal{V}^{(\mathsf{h})}, r_1, \ldots, r_j \rangle$. We will show Assumption 2 implies that for $j \in [qk]$, $d < d_{\mathsf{reg}}$, there exists a short exact sequence

$$0 \to \mathcal{A}_{d-1}/\mathcal{I}(j)_{d-1} \xrightarrow{\alpha} \mathcal{A}_d/\mathcal{I}(j-1)_d \xrightarrow{\beta} \mathcal{A}_d/\mathcal{I}(j)_d \to 0.$$

We prove this as follows. Let $\alpha : \mathcal{A}_{d-1}/\mathcal{I}(j-1)_{d-1} \to \mathcal{A}_d/\mathcal{I}(j-1)_d$ be the map defined as

$$\alpha([f]) := [r_j f].$$

Firstly, α is well-defined as if $f' \in [f]$, then $f' - f \in \mathcal{I}(j)_{d-1}$, i.e., there exist $g \in \langle \mathcal{S}^{(\mathsf{h})}, \mathcal{V}^{(\mathsf{h})} \rangle$ and $f_1, \ldots, f_j$ such that $f' - f = f_1 r_1 + \ldots + f_j r_j + g$. Let g' denote the homogenous degree-$(d-1)$ part of g and $f_1', \ldots, f_j'$ denote the homogenous degree-$(d-2)$ parts of $f_1, \ldots, f_j$. Then, $f' - f = f_1' r_1 + \ldots + f_j' r_j + g$. Now, $r_j f' - r_j f = f_1' r_j r_1 + \ldots + f_j' r_j^2 + g r_j \in \mathcal{I}(j-1)_d$ as $r_j^2 \in \langle \mathcal{S}^{(\mathsf{h})}, \mathcal{V}^{(\mathsf{h})} \rangle$. Therefore, $[r_j f'] = [r_j f]$. Secondly, α is injective. Suppose $\alpha([f]) = \alpha([f'])$, i.e., $[r_j f] = [r_j f']$. This implies that $r_j f - r_j f' \in \mathcal{I}(j-1)_d$, which in turn from Assumption 2 implies that $f - f' \in \mathcal{I}(j)_{d-1}$, i.e., $[f] = [f']$.

Let $\beta : \mathcal{A}_d/\mathcal{I}(j-1)_d \to \mathcal{A}_d/\mathcal{I}(j)_d$ be the map defined as

$$\beta([f]) := [f].$$

Again, β is well-defined as if $f_1 \in [f]$, then $f_1 - f \in \mathcal{I}(j-1)_d$. Since $\mathcal{I}(j-1)_d \subseteq \mathcal{I}(j)_d$, $f_1 - f \in \mathcal{I}(j)_d$ and hence $[f_1] = [f]$. Also, β is trivially surjective as $[f]$ is mapped to $[f]$.

Finally, we claim that $\mathsf{ker}(\beta) = \mathsf{im}(\alpha)$. To this end, we prove two statements. First, $\mathsf{im}(\alpha) \subseteq \mathsf{ker}(\beta)$. This is because if $[h] \in \mathsf{im}(\alpha)$, there exists an f such that $[h] = [r_j f]$. Now, $\beta([h]) = \beta([r_j f]) = [0]$ as $r_j \in \mathcal{I}(j)$. Next, $\mathsf{ker}(\beta) \subseteq \mathsf{im}(\alpha)$. Suppose $[h] \in \mathsf{ker}(\beta)$. Then, $\beta([h]) = [h] = [0]$, i.e., $h \in \mathcal{I}(j)_d$. This implies that there exist $g \in \langle \mathcal{S}^{(\mathsf{h})}, \mathcal{V}^{(\mathsf{h})} \rangle$ and $f_1, \ldots, f_j$ such that $h = f_1 r_1 + \ldots + f_j r_j + g$. Let g' denote the homogenous degree-d part of g and $f'_1, \ldots, f'_j$ denote the homogenous degree-$(d-1)$ parts of $f_1, \ldots, f_j$. Then, $h = f'_1 r_1 + \ldots + f'_j r_j + g$, i.e, $h - f'_j r_j \in \mathcal{I}(j-1)_d$. Therefore, $[h] = [r_j f'_j]$ and hence $\alpha([f'_j]) = [h]$, i.e., $[h] \in \mathsf{im}(\alpha)$.

Therefore, $0 \to \mathcal{A}_{d-1}/\mathcal{I}(j)_{d-1} \xrightarrow{\alpha} \mathcal{A}_d/\mathcal{I}(j-1)_d \xrightarrow{\beta} \mathcal{A}_d/\mathcal{I}(j)_d \to 0$ is indeed a short exact sequence.

By the Hilbert function property for short exact sequences (Section 3.6), we have

$$\mathcal{HF}_{\mathcal{A}/\mathcal{I}(j)}(d-1) - \mathcal{HF}_{\mathcal{A}/\mathcal{I}(j-1)}(d) + \mathcal{HF}_{\mathcal{A}/\mathcal{I}(j)}(d) = 0.$$

Let $h_{d,j} := \mathcal{HF}_{\mathcal{A}/\mathcal{I}(j)}(d) = \mathsf{dim}(\mathcal{A}_d/\mathcal{I}(j)_d)$. Then,

$$h_{d,j} := h_{d,j-1} - h_{d-1,j}.$$

Let $\mathcal{G}_j$ be the generating series for $h_{d,j}$, i.e., let

$$\mathcal{G}_j(\mathbf{z}) = \sum_{d=0}^{\infty} h_{d,j} \mathbf{z}^d.$$

Note that $\mathcal{G}_j = \mathcal{HS}_{\mathcal{A}/\mathcal{I}(j)}$. We have

$$\begin{aligned}
\mathcal{G}_{j-1}(\mathbf{z}) &= \sum_{d=0}^{\infty} h_{d,j-1} \mathbf{z}^d \\
&= \sum_{d=0}^{\infty} (h_{d,j} + h_{d-1,j}) \mathbf{z}^d \\
&= \sum_{d=0}^{\infty} h_{d,j} \mathbf{z}^d + \sum_{d=1}^{\infty} h_{d-1,j} \mathbf{z}^d \\
&= \sum_{d=0}^{\infty} h_{d,j} \mathbf{z}^d + \mathbf{z} \cdot \sum_{d=0}^{\infty} h_{d,j} \mathbf{z}^d \\
&= (1 + \mathbf{z}) \cdot \mathcal{G}_j(\mathbf{z}).
\end{aligned}$$

Therefore, we get $\mathcal{HS}_{\mathcal{A}/\mathcal{I}(qk)}(\mathbf{z}) = \frac{\mathcal{HS}_{\mathcal{A}/\langle \mathcal{S}^{(\mathsf{h})}, \mathcal{V}^{(\mathsf{h})} \rangle}(\mathbf{z})}{(1+\mathbf{z})^{qk}}$. Note that by definition $\mathcal{I}(qk) = \langle \mathcal{S}^{(\mathsf{h})}, \mathcal{V}^{(\mathsf{h})}, r_1, \ldots, r_{qk} \rangle = \langle \mathcal{F}^{(\mathsf{h})} \rangle$. Hence, the lemma follows. □

6.3 Estimating Witness Degree

As described earlier, following [BØ23], we will use the witness degree d_{wit} of the input polynomial system to estimate the cost of the XL Wiedemann approach. The systems $\mathcal{F}$ in Modeling 3 and Modeling 4 admit unique solutions for the

range of parameters of interest. If $(a_1, \ldots, a_n)$ is a unique solution of the polynomial system, it has the reduced Gröbner basis $\{\mathbf{x}_1 - a_1, \ldots, \mathbf{x}_n - a_n\}$.[6] If $\mathcal{I} := \langle \mathcal{F} \rangle$, we have $\mathsf{LM}(\mathcal{I}_{\leq 1}) = \mathsf{LM}(\mathcal{I})$ and $\mathsf{dim}(\mathcal{I}_{\leq d}) = \mathsf{dim}(\mathcal{A}_{\leq d}) - 1$. In particular, we can say that d_{wit} is the smallest degree such that the rank of the Macaulay matrix is equal to the number of columns minus one.

As in [BØ23], our Assumption 1 and 2 imply that the relevant Macaulay matrix has maximal rank. Now, consider the Hilbert series from Theorem 4 and 5 prior to truncation. The coefficient in a term of degree $d < d_{\mathsf{reg}}$ is the number of columns that cannot be reduced in the Macaulay matrix of degree d. When $d \geq d_{\mathsf{reg}}$, the coefficient measures the number of "excess" rows after full reduction. We therefore estimate the witness degree d_{wit} as

$$d_{\mathsf{wit}} = \min \left\{ d \in \mathbb{N} : \sum_{j \in [d+1]} [\mathbf{z}^{j-1}](\mathcal{HS}(\mathbf{z})) \leq 0 \right\}$$

where $[\mathbf{z}^{j-1}](\mathcal{HS}(\mathbf{z}))$ denotes the coefficient of $\mathbf{z}^{j-1}$ in the Hilbert series from Theorem 4 and 5 prior to truncation. We have experimentally verified these estimates as we discuss in Section 6.5.

6.4 Hybrid Approach

Following [BØ23], we also present a hybrid approach, which consists of repeatedly guessing a few noise-free positions of L (and therefore $\boldsymbol{e}$) and invoking XL Wiedemann until successfully computing $\boldsymbol{e}$. Parameterized by $f \in [t]$ and $\mu \in [\frac{n}{t}]$, the hybrid approach guesses μ noise-free positions in the first f blocks of $\boldsymbol{e}$ and adds the equations $\boldsymbol{e}_{i,v,j} = 0$ for those f blocks v and μ positions j in every instance i to $\mathcal{F}$. Let us determine a bound $p_{f,\mu}$ on the probability that the guessed positions are all noise-free. There are at least $\frac{n}{t} - 1$ noise-free positions in each block. Therefore, the probability that the μ positions guessed in any given block are noise-free is at least $\frac{\binom{\frac{n}{t}-1}{\mu}}{\binom{\frac{n}{t}}{\mu}} = 1 - \frac{\mu t}{n}$. This means that the probability that all the positions guessed are noise-free is at least $p_{f,\mu} = \left(1 - \frac{\mu t}{n}\right)^f$. We then expect to repeat the XL Wiedemann $O(p_{f,\mu}^{-1})$ times.

We now need to derive the Hilbert series of this modified system. We discuss this hybrid approach for Modeling 3 and note that the case of Modeling 4 is similar. Consider the impact of our guessing on Lemma 2. For a fixed block v, the number of j that are not fixed by our guess is now $\frac{n}{t} - \mu$ and not $\frac{n}{t}$ in the first f blocks and still $\frac{n}{t}$ in the last $t - f$. To make Lemma 3 work, we augment Assumption 1 as in [BØ23] with Assumption 3. This ensures that fixing variables does not introduce unexpected cancelations at higher degrees among the parity-check equations in $\mathcal{R}$. For any invertible matrix $\mathbf{P}$, $f \in [t]$, and $\mu \in [\frac{n}{t}]$, let $\overline{\mathbf{P}_{f,\mu}^{-1}}$

[6] In the case of Modeling 4, the field equations ensure that the ideal is radical, and the claim from Hilbert's Nullstellensatz. In the case of Modeling 3, we assume that the system is sufficiently overdetermined to ensure this.

denote the map that applies $\mathbf{P}^{-1}$ and then fixes the initial μ variables to 0 in the last f blocks of $\mathbf{e}$.

Assumption 3. *Let $\mathcal{R}$ be the set of parity-check equations from Modeling 3. For every permutation matrix $\mathbf{P}$ which stabilizes each block of $\mathbf{e}$, $f \in [t]$, and $\mu \in [\frac{n}{t}]$, we assume $\mathcal{R}^{(\mathsf{h})} \circ \overline{\mathbf{P}_{f,\mu}^{-1}}$ satisfies Assumption 1 in the ring $\mathcal{A} \circ \overline{\mathbf{P}_{f,\mu}^{-1}}$.*

Under Assumption 3, the Hilbert series for Modeling 3 now becomes

$$\mathcal{HS}_{\mathcal{A}/\langle \mathcal{F}^{(\mathsf{h})} \rangle, \mathrm{hyb}, f, \mu}(\mathbf{z}) = (1-\mathbf{z})^{qk} \cdot \left(1 + \left(\frac{n}{t} - \mu\right)\left(\frac{1}{(1-\mathbf{z})^q} - 1\right)\right)^f \cdot \left(1 + \frac{n}{t}\left(\frac{1}{(1-\mathbf{z})^q} - 1\right)\right)^{t-f},$$

truncated after the first ≤ 0 coefficient. As before, the degree d_{wit} is derived from the Hilbert series. We refer to [BØ23] for further details.

6.5 Attack's Evaluation

We now evaluate our algebraic attack. We write scripts in the Magma Computational Algebra System V2.28-13 [BCP97] to (1) experimentally verify our Hilbert series and (2) compute the time complexity of the attack. We plan to open-source our Magma scripts along with our PCG code (see Section 8). In all our experiments, we use the free online Magma calculator, and hence our computation is restricted to ≤ 2 minutes. Thus, we were able to run our scripts only on smaller parameters.

We first experimentally verify our Hilbert series from Section 6.4 is correct. We compare the output with the output of Magma's `HilbertSeries(·)`. We verify the hybrid version of our Hilbert series over $\mathbb{F}_{101}$ and for the same systems as [BØ23] in Table 7 (we can similarly verify over $\mathbb{F}_2$). The key difference is that we also add a parameter $q \geq 1$ ($q = 1$ for [BØ23]) to our system, which represents the number of SD instances. The largest q we test for is 4 as for larger q the `HilbertSeries(·)` function exceeds our computational resources. For $\mathbb{F}_{101}$, we confirm our Hilbert series is the same for all the tested systems (q, n, k, t, f, μ), and hence Assumption 1 holds:

$$(4,30,15,5,0,0), (4,30,20,5,0,0), (3,40,20,5,0,0),$$
$$(4,40,30,5,0,0), (3,49,30,7,0,0), (3,48,30,8,0,0),$$
$$(2,40,25,10,0,0), (1,84,50,12,3,2), (3,56,30,7,2,3),$$
$$(2,56,30,7,6,3), (3,70,40,10,5,2), (4,70,40,10,5,3)$$

We next focus on the efficiency of the algebraic attack. We run 2 experiments. For both, we consider $\mathbb{F}_{2^{128}}$ and $\mathbb{F}_2$ and the XL hybrid approach of Section 6.4. In each, we find the (f, μ) that results in the most efficient attack. We pick syndrome decoding parameters n, $k = n/2$ given our computational restrictions.

In the first experiment (Figure 2), we show how the witness degree d_{wit} and the complexity of the XL Hybrid algorithm changes with increasing q. We fix $n = 2^{16}$, $k = 2^{15}$, fix $t = 69$ such that we get XL complexity of 128 with $q = 1$, and then vary $q = \{1, 2, 2^4, 2^7, 2^{10}\}$. Our results show that in fact the complexity of the XL algorithm increases with larger q. This is because the system of equations gets larger with increasing q, and hence more expensive for XL to solve. Another interesting aspect is that for all runs the best $f = t$. I.e., we should guess noise-free positions in all blocks. It is not surprising that this holds in the constant rate SD setting, where the amount of noise t is smaller than in the non-constant rate LPN setting that [BØ23] emphasize. Overall, note that to get 128-bit security we require $t = 69$, which is substantially less than what is suggested by the linear test (Section 8). This implies that the algebraic attack may not be the ideal choice for our setting.

	$\mathbb{F}_{2^{128}}$ $(t = 69)$				$\mathbb{F}_2$ $(t = 69)$			
q	d_{wit}	(f, μ)	XL	Hybrid	d_{wit}	(f, μ)	XL	Hybrid
1	2	(69,410)	128		2	(69,410)	128	
2	2	(69,410)	132		2	(69,410)	132	
2^4	2	(69,410)	144		2	(69,410)	144	
2^7	2	(69,410)	156		2	(69,410)	156	
2^{10}	2	(69,410)	168		2	(69,410)	168	

Fig. 2. This experiment shows how the witness degree d_{wit} and the complexity of the XL algorithm changes with increasing q. We set our parameters n, k following standard choices in SD. I.e., we fix $n = 2^{16}$, $k = 2^{15}$, fix $t = 69$ such that we get XL complexity of 128 with $q = 1$, and then vary $q = \{1, 2, 2^4, 2^7, 2^{10}\}$. We consider $\mathbb{F}_{2^{128}}$ and $\mathbb{F}_2$, hybrid evaluation, and search all (f, μ) for the most efficient attack for each q. Note the results for $\mathbb{F}_{2^{128}}$ and $\mathbb{F}_2$ are identical.

The first experiment showed that we get the best attack by attacking a single SD instance. In the second experiment (see Figure 3), we thus show the amount of noise t necessary to get 128 bits of security for different n, k, and q fixed to 1. We then repeat the same runs for $q = 2^{10}$ to show how t changes.

			$\mathbb{F}_{2^{128}}$						$\mathbb{F}_2$					
			$q = 1$			$q = 2^{10}$			$q = 1$			$q = 2^{10}$		
n	k	d_{wit}	(f, μ)	XL	t	(f, μ)	XL	t	(f, μ)	XL	t	(f, μ)	XL	t
2^{10}	2^9	2	(101, 4)	128	107	(58,7)	127	58	(101, 4)	128	107	(58,7)	127	58
2^{12}	2^{11}	2	(89, 20)	127	90	(46,37)	128	46	(89, 20)	127	90	(46,37)	128	46
2^{14}	2^{13}	2	(74, 192)	128	74	(34,191)	128	34	(79, 91)	128	79	(34,191)	128	34
2^{16}	2^{15}	2	(69, 410)	128	69	(24,1364)	128	24	(69, 410)	128	69	(24,1364)	128	24

Fig. 3. This experiment picks standard SD's n, k parameters used for PCGs (i.e. $k = n/2$) and computes how much noise t is necessary to get XL complexity of ≈ 128 for $q = 1$ and $q = 2^{10}$. We consider $\mathbb{F}_{2^{128}}$ and $\mathbb{F}_2$, XL hybrid evaluation, and search all (f, μ) for the most efficient attack for each parameter setting.

7 Pseudorandom Correlation Generator (PCG) from SSD

Securely Computing the Noise Vector for PCG protocols with FSS. Pseudorandom Correlation Generators (PCGs) have emerged as the preferred way to generate correlated randomness used by MPC protocols. For example, oblivious transfer (OT) and vector oblivious linear evaluation (VOLE) correlations can be generated with sublinear communication. For that reason, they are commonly referred to as *silent* OT and VOLE. These in turn enable higher level protocols such as garbled circuits, GMW, and PSI protocols. Additionally, Beaver triples for both binary and larger fields can succinctly be generated using PCG techniques.

At the heart of these constructions is the syndrome decoding assumption. If we ignore some complications, these protocols need to multiply a scalar $\Delta \in \mathbb{F}$ by the sparse noise vector $\boldsymbol{e} \in \mathbb{F}^n$, where $\Delta, \boldsymbol{e}$ are each held by a different party. The result should be a secret sharing $[\![\Delta \boldsymbol{e}]\!]$ which is then compressed by $\mathbf{G}$, i.e. $[\![\mathbf{G}\Delta \boldsymbol{e}]\!]$. Such a secret sharing represents the VOLE correlation and can locally be compiled into k OTs, where k is the codeword length. Other correlations follow a similar formula. The crux of this protocol is the generation of $[\![\Delta \boldsymbol{e}]\!]$.

Note that $\boldsymbol{e}$ with weight t can be succinctly described as t coordinates and values $\{(\alpha_i, v_i)\}_{i\in[t]}$ such that $\boldsymbol{e}_{\alpha_i} = v_i$ and otherwise is zero. This can be used to generate a secret sharing of $[\![\Delta \boldsymbol{e}]\!]$ with only *sublinear* $O(t \log n)$ communication and $O(n)$ computation. Although not strictly necessary, for the rest of this discussion we will assume $\boldsymbol{e}$ is regular, i.e. can be written as the concatenation of t length $\frac{n}{t}$ subvectors $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_t \in \mathbb{F}^{\frac{n}{t}}$.

The secret sharing of $\boldsymbol{e}$ times Δ is achieved using function secret sharing (FSS) [BGI15]. Specifically, FSS allows a party with a function f in some function class $\mathcal{F}$ to split it into function shares s_1, s_2 s.t. $s_1(i) + s_2(i) = f(i)$ for all i. Additionally, s_j alone leaks no information about f beyond its membership in $\mathcal{F}$. While it is not known how to efficiently achieve this for general function classes, the special case of point functions $f_{i,y}$ emit very efficient constructions. A point function $f_{i,y}$ is defined by $f_{i,y}(x) = 0$ for all $x \neq i$ and $f_{i,y}(i) = y$. The core idea is that given FSS of the point functions $f_{\alpha_i, v_i\Delta}$ for $i \in [t]$, the shares of $[\![\Delta \boldsymbol{e}]\!]$ can be generated as the concatenations of the individual FSS shares evaluated at all $x \in [n/t]$. That is, $\Delta \boldsymbol{e}_{i,j} = f_{\alpha_i, v_i}$.

Compute $[\![\Delta \boldsymbol{e}]\!]$ for unit vector $\boldsymbol{e}$ [BGI14]. Let us make the simplification that $\boldsymbol{e}$ is a unit vector with $\boldsymbol{e}_\alpha = 1$. As such, the point function in question is $f_{\alpha,\Delta}$. Later we will consider the general case where $\boldsymbol{e}$ has higher weight and arbitrary values.

The protocol is a simple tree-based construction inspired by [GGM84]'s idea for building a PRF from a length doubling PRG $\mathsf{G} : \{0,1\}^\kappa \to \{0,1\}^{2\times\kappa}$. The *sender*, takes $\Delta \in \mathbb{F}$ as input and samples a random seed $s \in \{0,1\}^\kappa$. The *receiver* takes the noise vector $\boldsymbol{e} \in \mathbb{F}^n$ as input. The sender with seed $s \in \{0,1\}^\kappa$ defines a binary tree with n leaves. s is assigned to the root node. The left and right child nodes are then assigned the values $(s_0, s_1) = \mathsf{G}(s)$, respectively. This is repeated for all of the descendant nodes. This tree will have $D := \lceil \log_2(n) \rceil + 1$ levels

and n leaves. Let the value assigned to the node at level $d \in [D]$ and position $j \in [2^{d-1}]$ be indexed as $s_{d,j}$.

Consider the path from the root to the leaf node α (at level D). We will call this the critical path P_α. Our first objective is for the receiver to learn all seeds $s_{i,j}$ that are *not* on the critical path. Consider the so-called co-path $\overline{P}_\alpha$ to the critical path P_α which consists of all the sibling nodes along the critical path. For example, if $n = 16, \alpha = 7$, then the critical path is $P_\alpha = ((1,1),(2,1),(3,3),(4,7))$ while the co-path is $\overline{P}_\alpha = ((2,2),(3,4),(4,8))$. Note that the co-path is not a path in the tree but a set of nodes. The interesting property is that if the receiver can learn only the seeds indexed by the co-path $\overline{P}_\alpha$, then they can use G to rederive all of the other seeds that are not on the critical path.

[BCG+19a] presented a surprisingly simple protocol for allowing the receiver to learn exactly $\{s_p \mid p \in \overline{P}_\alpha\}$. For the first level it is simple, the parties can use a 1-out-of-2 OT so that the receiver can either learn $s_{2,1}$ or $s_{2,2}$ depending on which is indexed by $\overline{P}_{\alpha,1}$. The idea is to then proceed level by level. Once at level $i \in \{3,\ldots,D\}$, observe that the receiver already knows all but two of the seeds at level i. In particular, they do not know $s_{P_{\alpha,i+1}}$ and $s_{\overline{P}_{\alpha,i}}$. However, we can no longer use a simple 1-out-of-2 OT as the sender does not know which pair of children to use.

The critical observation is that the sender can sum all of the left child seeds $l_i := \sum_j s_{i,2j-1}$ and the right child seeds $r_i := \sum_j s_{i,2j}$ and use these as OT messages. The receiver will learn the left or right sum depending on if the left or right child is indexed by $\overline{P}_{\alpha,i-1}$. That is, if $\overline{P}_{\alpha,i-1,2}$ is even, then they will learn r_i and otherwise l_i. If they learn l_i, then they can compute $s_{i,\overline{P}_{\alpha,i-1}}$ as l_i minus the sum of the left children they already know. Otherwise, they compute $s_{i,\overline{P}_{\alpha,i-1}}$ as r_i minus the sum of the right children they already know.

Observe that we almost have a secret sharing of $\boldsymbol{e}$. The sender can define their shares of $[\![\Delta \boldsymbol{e}_i]\!]$ as $-\mathsf{map}(s_{D,i})$ where $\mathsf{map} : \{0,1\}^\kappa \to \mathbb{F}$ is a function mapping bit strings to field elements. For all but the special index α, the receiver can set their share as $\mathsf{map}(s_{D,i})$. What remains is for the receiver to somehow learn the value $\mathsf{map}(s_{D,\alpha}) + \Delta$ without revealing α. However, we already have the machinery to achieve this. The sender can sum *all* the children and add Δ, i.e. $\sigma := \Delta + \sum_{j\in[n]} \mathsf{map}(s_{D,j})$. This sum is sent to the receiver who can subtract off the shares they already know to compute $\mathsf{map}(s_{D,\alpha}) + \Delta$. As such, the parties now have a valid sharing $[\![\Delta \boldsymbol{e}]\!]$.

Recall that we made two simplifications. First, we restricted the noise value $\boldsymbol{e}_\alpha$ to always be one. Removing this simplification introduces a challenge in that the sender no longer can simply use Δ in the final sums. Instead they would need to use $\Delta \boldsymbol{e}_\alpha$ but $\boldsymbol{e}_\alpha$ is a secret value known only to the receiver. This issue is solved by computing a sharing $[\![b]\!]$ for $b = \Delta \boldsymbol{e}_\alpha$ using some other secure multiplication protocol where the sender inputs Δ and the receiver inputs $\boldsymbol{e}_\alpha$ (more specifically, when we consider $t \geq 1$, using another VOLE of size t to multiply Δ with all t non-zeros in $\boldsymbol{e}$, i.e. $\boldsymbol{e}_{\alpha_1},\ldots,\boldsymbol{e}_{\alpha_t}$). The sender will now use its share $[\![b]\!]_\mathsf{s}$ in place of Δ. As such, the result will be sharing $[\![[\![b]\!]_\mathsf{s} \boldsymbol{e}']\!]$ where $\boldsymbol{e}'$ is the unit vector with

$\boldsymbol{e}'_\alpha = 1$. The receiver can locally translate this sharing $[\![[\![b]\!]_\mathsf{s}\boldsymbol{e}']\!]$ into a sharing $[\![\Delta\boldsymbol{e}]\!]$ by simply adding $[\![b]\!]_\mathsf{r}\boldsymbol{e}'$ to its share $[\![[\![b]\!]_\mathsf{s}\boldsymbol{e}']\!]_\mathsf{r}$. That is, the sender computes $[\![\Delta\boldsymbol{e}]\!]_\mathsf{s} := [\![[\![b]\!]_\mathsf{s}\boldsymbol{e}']\!]_\mathsf{s}$ and the receiver computes $[\![\Delta\boldsymbol{e}]\!]_\mathsf{r} := [\![[\![b]\!]_\mathsf{s}\boldsymbol{e}']\!]_\mathsf{r} + [\![b]\!]_\mathsf{r}\boldsymbol{e}'$.

The second simplification was that $\boldsymbol{e}$ is a unit vector as opposed to regular weight t with non-zeros at $\boldsymbol{e}_{\alpha_1}, \ldots, \boldsymbol{e}_{\alpha_t}$. However, this is easy to remedy by simply repeating the protocol t times and concatenating the shares (see [SGRR19] for the non-regular case).

Computing q stationary sharings $[\![\Delta\boldsymbol{e}_1]\!], \ldots, [\![\Delta\boldsymbol{e}_q]\!]$. Given the description above it should be clear how one can construct many stationary noise vectors. For example, one could run the same protocol except that now the element type is $\mathbb{F}^q$ as opposed to $\mathbb{F}$. In particular, let $\alpha_1, \ldots, \alpha_t$ be the stationary noise locations. The parties can first use an additional VOLE protocol to compute shares of $[\![b]\!] = \Delta\dot{(}v_{1,1}, \ldots, v_{q,t})$. The sender can then use $v'_i := ([\![b_{i,1}]\!]_\mathsf{s}, \ldots, [\![b_{i,q}]\!]_\mathsf{s})$ as the value for the i'th point function f_{α_i, v'_i}. Finally, the receiver adds their shares $([\![b_{i,1}]\!]_\mathsf{r}, \ldots, [\![b_{i,q}]\!]_\mathsf{r})$ to the position corresponding to α_i. The result is q sharings $[\![\Delta\boldsymbol{e}_1]\!], \ldots, [\![\Delta\boldsymbol{e}_q]\!]$, each correlated by $\boldsymbol{e}_i$ having noise at locations $\alpha_1, \ldots, \alpha_t$.

Moreover, the instances can all be computed together or sequentially, where the next instance $[\![\Delta\boldsymbol{e}_{i+1}]\!]$ is generated from seeds along with $O(t)$ communication, i.e. no additional oblivious transfers are required after generating $[\![\Delta\boldsymbol{e}_1]\!]$. Additionally, the prior VOLE can be used to bootstrap the next instance, eliminating the need to perform all of the tq input VOLE correlations.

If performed in parallel, our construction with SSD requires a single VOLE of size tq and $t\log_2(n/t)$ OTs. The standard construction requires a single VOLE of size tq, and $qt\log_2(n/t)$ OTs. If instead performed sequentially, where we prepare the next instance from the previous, we require just a single VOLE with t inputs and $t\log_2(n/t)$ OTs. Note we can also get $[\![\Delta_1\boldsymbol{e}_1]\!], \ldots, [\![\Delta_q\boldsymbol{e}_q]\!]$, where Δ is different for each instance, and still reuse all the OTs due to SSD.

Generalizing to OT and Binary OLE. So far, our explanation focused on VOLE. For VOLE, both $\boldsymbol{e}$ and Δ are over some arbitrary $\mathbb{F}$. For OT, $\boldsymbol{e}$ is over $\mathbb{F}_2$ and Δ is over some large field, e.g. $\mathbb{F}_{2^{128}}$. Focusing on the SD case with $q = 1$, k instances of random OT are obtained as $m_{i,0} := \mathsf{H}([\![(\mathbf{G}\Delta\boldsymbol{e})_i]\!]_\mathsf{s}), m_{i,1} := \mathsf{H}([\![(\mathbf{G}\Delta\boldsymbol{e})_i]\!]_\mathsf{s} + \Delta), a_i := (\mathbf{G}\boldsymbol{e})_i, m_{i,a_i} := \mathsf{H}([\![(\mathbf{G}\Delta\boldsymbol{e})_i]\!]_\mathsf{r})$. Binary OLE can be immediately obtained from random OT by defining $b_i := \mathsf{lsb}(m_{i,1} - m_{i,0}), [\![c_i]\!]_\mathsf{s} := \mathsf{lsb}(m_{i,0}), [\![c_i]\!]_\mathsf{r} := \mathsf{lsb}(m_{i,a_i})$.

With SD, the VOLE and OT protocols are almost identical. The main difference is that in OT the noise vector $\boldsymbol{e}$ is binary and the noisy $\boldsymbol{e}_{i,\alpha}$ *is fixed to* 1. This slightly simplifies the VOLE protocol as we can omit the multiplication of Δ with $\boldsymbol{e}_{i,\alpha}$, as described above. However, the SSD assumption states that the noise must be *uniformly random*, even in the binary case (i.e. includes 0). Therefore, $\boldsymbol{e}$ is still binary but the noisy $\boldsymbol{e}_{i,\alpha} \in \{0, 1\}$ must be uniformly sampled. As a result, for OT with SSD, we still save on the OTs across the q instances, but we need to perform a $(\mathbb{F}_2, \mathbb{F}_{2^{128}})$ subfield VOLE of size t for each instance.

Cache and Memory for Binary OLEs. Recall that binary OLEs are trivially generated from OTs and OTs are generated with an almost identical protocol to VOLE. In other words, we generate $[\![\Delta \boldsymbol{e}]\!]$ with PPRF. Also note that random binary OLEs cost only 2 bits of communication to derandomize. Ideally, we want the generation of random binary OLEs to be cheaper than the derandomization. I.e., we want each generation of random binary OLE to cost only 1 bit of communication. Say we want to generate $h = 2^{30}$ OLEs with plain SD. Naively, one might set $k = h$ and run a single large SD-based protocol to generate the OLEs. However, this would be unnecessarily inefficient. As n becomes very large, memory efficiency dominates the running time. We observe a severe performance degradation beyond $n \neq 2^{16}$. However, as k becomes smaller, the relative communication cost can increase beyond 1 bit per OLE. A reasonable compromise is $k = 2^{20}$ and run $q = h/k$ independent instances of the SD-based protocol.

However, when the SSD assumption is employed it is possible to further reduce k while staying below the 1 bit per OLE threshold. In particular, our construction requires approximately $\log_2(n/t)$ times less communication, and therefore can scale to as small as $k = 2^{14}$ with 3 bits of communication per OLE or $k = 2^{16}$ with less than 1 bit per OLE. Having a smaller k in turn allows for better computational overhead while still generating a total of $h = qk$ OLEs. See Figure 5 in Section 8 for a detailed breakdown of the effect.

Large Field OLE and Beaver Triples. State-of-the-art constructions for correlations of degree 2 are all based on Ring LPN [BCG+20]. Among other correlations, these protocols allow one to generate an OLE correlation $a_i, b_i, [\![c_i]\!]$ such that $c_i = a_i b_i$ over a large field $\mathbb{F}$, e.g. a prime field. Two of these OLEs can then be used as a Beaver triple.

Instead of multiplying a scalar Δ with a sparse vector $\boldsymbol{e}$, these protocols have two sparse vectors a', b' and interpret them as sparse polynomials. a' and b' are known to the sender and receiver, respectively. If we ignore some details, the sharing $[\![c]\!]$ is obtained by performing sparse polynomial multiplication $[\![c']\!] := a' \cdot b'$ using some secure computation technique (more detail in the following paragraph). Once the sharing $[\![c']\!]$ is obtained, these polynomails are compressed by linear functions $\mathbf{G}, \mathbf{G}'$ as $a := \mathbf{G} \cdot a', b := \mathbf{G} \cdot b', [\![c]\!] := \mathbf{G}' \cdot [\![c']\!]$. For appropriately chosen $\mathbf{G}, \mathbf{G}'$, it will hold that $a_i b_i = c_i$, where $a_i, b_i, c_i \in \mathbb{F}$, is the ith coefficient of the polynomial. The security of this protocol depends on LPN/SD holding for the specific $\mathbf{G}$ used. It is, therefore, natural to generalize this protocol to use the SSD assumption.

The core challenge in this protocol is the generation of the sparse vector/polynomial $[\![c']\!]$. Unlike the binary OLE, VOLE, and OT above, neither party knows the value of c'. However, given that a', b' are t-sparse, then c' is $t' \leq t^2$ sparse. Additionally, c' is not a regular vector/polynomial even if a', b' are. This in turn means that using relatively simply techniques $[\![c']\!]$ will require $O(nt)$ work to generate. The authors of [BCG+20] suggest that batch codes could be used to further reduce this to $O(n)$ work but at the cost of substantially increasing the amount of computation needed to be performed in MPC,

e.g. cuckoo hashing t^2 values. The exact cost of this MPC computation remains unexplored, but is certainly non-trivial.

We observe that the SSD assumption can significantly improve the amortized complexity of the protocol. Each SSD instance will sample a sparse vector/polynomial a'_i, b'_i where the sender holds a'_i and the receiver holds b'_i. The $a'_1, \ldots, a'_q$ vectors will have non-zeros in the same t locations, respectively for $b'_1, \ldots, b'_q$. The parties will input these non-zero locations into an MPC that computes

- The t^2 non-zero locations in the product polynomials c'_i.
- Using batch codes, assigning each non-zero location to a point function $f_1, \ldots, f_\tau$ for some $\tau \geq t^2$, e.g. cuckoo hash t^2 values.
- Generate and output distributed point function [Ds17] shares $[\![f_1]\!], \ldots, [\![f_\tau]\!]$.

The parties then locally expand the distributed point function shares $[\![f_1]\!], \ldots, [\![f_\tau]\!]$ to obtain q different sparse polynomials $[\![c'_q]\!], \ldots, [\![c'_q]\!]$. Depending on the batch code used, this expansion requires between $O(ntq)$ and $O(nq)$ local computation.

The critical observation is that the vast majority of the work required to be performed within the MPC is independent of the values of the coefficients and only depends on the location. As such, generating $q = 100$ instances is only slightly more expensive than $q = 1$. Moreover, batch codes such as cuckoo hashing require more work to be performed within the MPC, but are completely independent of q.

This suggests that the use of the more expensive batch codes (which reduce the local expansion time $t\times$) are indeed a good tradeoff when SSD is used because the batch code encoding is only performed once and then reused for all of the q instances. Due to the lack of an existing Ring-LPN protocol implementation and significant work required to implement it (with or without SSD), we leave the concrete evaluation of this technique to future work.

8 Experimental Evaluation

Implementation & Setting. We present a detailed description of how to adapt existing PCG protocols in Section 7. The crux for degree 1 PCG protocols is (1) generating a sharing of $\boldsymbol{\Delta e}$ and (2) compressing it by the generator matrix $\mathbf{G}$. SSD optimizes the generation of $[\![\boldsymbol{\Delta e}]\!]$ across q PCG instances. Thus, we implement our SSD-based $[\![\boldsymbol{\Delta e}]\!]$ generation. Our implementation extends the libOTe framework [RR]. We use libOTe's implementation of OT from [Roy22]'s SoftSpokenOT and VOLE from [RRT23]'s work on expand-convolute codes for the base correlations (OT, subfield VOLE for $\mathbb{F}_2$, and VOLE for $\mathbb{F}_{2^{128}}$).

We conduct our experiments on an HP Victus machine running Windows 11, equipped with a 12th Gen Intel(R) Core(TM) i7-12650H CPU at 2.30GHz and 15.6GB of usable RAM. The parties execute sequentially on the same thread and on the same laptop. The wall-clock time reflects the combined runtime of both parties. We also report the total communication summed across both parties. Each data point is sampled over 10 runs, and we present their arithmetic mean.

Protocol	q	Assumption	#OTs	#VOLE	Time (ns/o) Setup	Expand	Mult.	Comm. (b/o) Setup	Expand
OT $\boldsymbol{e} \in \mathbb{F}_2^n$ $\Delta \in \mathbb{F}_{2^{128}}$	2^4	SSD, $t = 400$	4400	6400	0.09	11.36	52.63	0.18	0.46
		SD, $t = 176$	33792	0	0.20	17.24	52.63	0.52	2.24
	2^8	SSD, $t = 400$	4400	102400	0.04	11.88	52.75	0.10	0.21
		SD, $t = 176$	540672	0	0.19	16.91	52.75	0.52	2.24
	2^{12}	SSD, $t = 400$	4400	1638400	0.04	12.11	52.73	0.10	0.20
		SD, $t = 176$	8650752	0	0.19	16.74	52.73	0.52	2.24
VOLE $\boldsymbol{e} \in \mathbb{F}_{2^{128}}^n$ $\Delta \in \mathbb{F}_{2^{128}}$	2^4	SSD, $t = 400$	4400	6400	0.53	11.55	52.63	1.18	0.46
		SD, $t = 400$	70400	6400	0.89	17.63	54.58	2.19	4.70
	2^8	SSD, $t = 400$	4400	102400	0.19	11.91	52.75	0.08	0.21
		SD, $t = 400$	1126400	102400	0.59	17.29	54.61	1.15	4.70
	2^{12}	SSD, $t = 400$	4400	1638400	0.26	12.20	52.73	0.01	0.20
		SD, $t = 400$	18022400	1638400	0.66	17.15	54.65	1.08	4.70

Fig. 4. This experiment compares the runtime and communication costs of generating OT and VOLE with SSD vs. SD. It fixes $n = 2^{19}$, $k = 2^{18}$ and varies the number of batches $q \in \{2^4, 2^8, 2^{12}\}$. The cost is split into base correlations (OT, (subfield) VOLE), expanding the seeds to get a sharing of $\Delta \boldsymbol{e}$, and multiplying the result by $\mathbf{G}$. The time (nanoseconds) and communication (bits) are expressed per output, i.e. divided by qk. t is selected such that we get 128 bits of security.

Protocol	k	Assumption	#OTs	#VOLE	Time (ns/o) Setup	Expand	Mult.	Comm. (b/o) Setup	Expand
OT $\boldsymbol{e} \in \mathbb{F}_2^n$ $\Delta \in \mathbb{F}_{2^{128}}$	2^{14}	SSD, $t = 400$	2800	409600	0.58	11.90	33.48	1.58	3.17
		SD, $t = 176$	1441792	0	2.04	32.17	33.48	5.50	24.84
	2^{16}	SSD, $t = 400$	3600	102400	0.15	11.95	49.61	0.41	0.84
		SD, $t = 176$	450560	0	0.64	20.59	49.61	1.72	7.58
	2^{18}	SSD, $t = 400$	4400	25600	0.05	11.80	52.80	0.12	0.26
		SD, $t = 176$	135168	0	0.19	17.81	52.80	0.52	2.24
VOLE $\boldsymbol{e} \in \mathbb{F}_{2^{128}}^n$ $\Delta \in \mathbb{F}_{2^{128}}$	2^{14}	SSD, $t = 400$	2800	409600	3.22	11.93	33.48	0.32	3.17
		SD, $t = 400$	2867200	409600	7.25	32.55	37.30	11.24	50.20
	2^{16}	SSD, $t = 400$	3600	102400	0.79	11.88	49.61	0.31	0.84
		SD, $t = 400$	921600	102400	2.07	21.30	52.05	3.81	15.67
	2^{18}	SSD, $t = 400$	4400	25600	0.26	11.62	52.80	0.31	0.26
		SD, $t = 400$	281600	25600	0.64	18.32	54.49	1.36	4.70

Fig. 5. This experiment compares the runtime and communication costs of generating OT and VOLE with SSD vs. SD. It fixes $qk = 2^{24}$, and varies q, k such that $(q, k) \in \{(2^{10}, 2^{14}), (2^8, 2^{16}), (2^6, 2^{18})\}$. The cost is split into base correlations (OT, (subfield) VOLE), expanding the seeds to get a sharing of $\Delta \boldsymbol{e}$, and multiplying the result by $\mathbf{G}$. The time (nanoseconds per output) and communication (bits per output), i.e. total divided by qk. t is selected such that we get 128 bits of security.

Security. We use our findings in Section 5 on SSD security against linear tests to choose the appropriate amount of noise t for Figure 4 and Figure 5. To get 128 bits of security, we need $t = 176$ for OT generated with the SD assumption and $t = 400$ otherwise. Importantly, note from Theorem 3 that q does not impact the choice of t. We also note that the parameters from the linear test framework tend

to be conservative. E.g., [LWYY22]'s work suggests that $t \approx 60$ gives 128 bits of security against known linear attacks while our algebraic attacks in Section 6.5 suggest $60 < t < 110$. We could narrow this gap by adapting existing linear attacks to SSD. For example, [ES24] recently did this for regular SD. As a first work that introduces SSD, we opt to be more conservative and select parameters according to the linear test framework to obtain provable guarantees.

Communication. We now express our communication costs analytically. Both protocols require the same constant number of rounds. The exact number depends on the instantiation but can be as small as 2.

We denote the depth of the GGM tree $d = \lceil \log_2(n/t) \rceil$. For OT with SSD, we generate dt base OTs for the GGM tree, send $256dt$ additional bits for the GGM tree, and generate subfield VOLE of size qt. As we are working over $\mathbb{F}_2$, we implement the subfield VOLE via qt base OTs. We also communicate $128tq$ bits during the expansion to get $[\![\Delta \boldsymbol{e}]\!]$. In total, we need $(d+q)t$ base OTs and $256dt + 128qt$ bits. In contrast, OT with SD requires qdt base OTs and $256qt(d+1)$ additional bits, but with a mildly smaller t. For VOLE with SSD, we also generate dt base OTs for the GGM tree, send $256dt$ additional bits for the GGM tree, and communicate $128tq$ bits during the expansion. However, we also consume qt base VOLE correlations, which cannot be implemented with base OTs as we are not working over $\mathbb{F}_2$. Thus, in total we need dt base OTs, qt VOLE correlations, and $256dt + 128qt$ bits. In contrast, VOLE with SD requires qdt base OTs, qt VOLE correlations, and $256qt(d+1)$ bits of communication.

Experiments. We consider two experiments. The first (Figure 4) fixes a batch size of $n = 2^{19}, k = 2^{18}$ and varies the number of batches $q \in \{2^4, 2^8, 2^{12}\}$. We compare our new SSD-based OT and VOLE protocols to the traditional SD protocols. We report the number of base correlations required, i.e. OTs and (subfield) VOLE. We break down the time required to compute base correlations, expand the seeds to compute $[\![\Delta \boldsymbol{e}]\!]$, and compress the error vector using expand-convolute codes [RRT23]. We report time as the number of nanoseconds per output element. We also report the communication overhead as bits per output element. The second experiment (Figure 5) reports the same quantities but for varied batch size $k \in \{2^{14}, 2^{16}, 2^{18}\}$ and a fixed $qk = 2^{24}$.

Discussion. First, we discuss observations that apply to both experiments. Note that for OT from SD we need no base VOLEs as the noisy elements of $\boldsymbol{e}$ are always 1 (recall this is not the case for OT from SSD). For all other settings, we need qt base VOLEs. Furthermore, note that for SSD and a fixed k, n, the number of base OTs stays the same for different q as all can be reused. Next, note that the setup for SSD is cheaper for OTs than for VOLEs as for OTs we generate the base VOLEs also with OTs. For VOLE, the multiplication by $\mathbf{G}$ is slightly more expensive for SD than for SSD as we cannot run the multiplication for SD over $\mathbb{F}_2$. Lastly, the parameter t differs for SD and SSD when generating OTs. We select t using our equations in Section 5 to get 128 bits of security assuming

$\mathbf{G}$ has a relative pseudominimum distance of 0.2. We now discuss observations specific to each experiment.

- **Experiment 1.** We reduce the total communication $\approx 4.3-9.4\times$ for OT and $\approx 4.2-28.6\times$ for VOLE. To compute $[\![\Delta \boldsymbol{e}]\!]$ (Setup+Expand in Figure 4), we reduce runtime $\approx 1.4-1.5\times$. For the parameters in this experiment, multiplying $[\![\Delta \boldsymbol{e}]\!]$ by $\mathbf{G}$ is the runtime bottleneck, and thus we reduce total runtime $\approx 1.1\times$ for both OT and VOLE. Our improvement can be significantly larger for different parameters (see Experiment 2).
- **Experiment 2.** We reduce total communication $\approx 6.4-7.5\times$ for OT and $\approx 10.7-17.6\times$ for VOLE. To compute $[\![\Delta \boldsymbol{e}]\!]$, we reduce runtime $\approx 1.5-2.7\times$. Notably, the runtime improvement increases for higher q and smaller k. This implies that for a fixed qk, it is preferable to create more smaller instances (i.e. large q and small k) than execute few large instances. This is further exacerbated by the fact that for small instances, multiplying by $\mathbf{G}$ also becomes cheaper (33 nanoseconds per output (ns/o) for $k = 2^{14}$ vs. 52 ns/o for $k = 2^{18}$).
  To generate 2^{24} OTs, the fastest (q, k) configuration with respect to runtime results in about 5 b/o and a total of 45 ns/o for SSD and 68 ns/o for SD resulting in $1.5\times$ improvement in throughput. For VOLE, we similarly get $\approx 1.5\times$ throughput improvement.

Acknowledgments. This work is supported in part by a Visa research award and NSF awards CNS-2246354, and CCF-2217070.

References

ABG+14. Adi Akavia, Andrej Bogdanov, Siyao Guo, Akshay Kamath, and Alon Rosen. Candidate weak pseudorandom functions in ac0 $\odot$ mod2. In *Proceedings of the 5th Conference on Innovations in Theoretical Computer Science*, ITCS '14, page 251–260, New York, NY, USA, 2014. Association for Computing Machinery.

AFS05. D. Augot, M. Finiasz, and N. Sendrier. A family of fast syndrome based cryptographic hash functions. In *Mycrypt*, 2005.

ANO+22. Damiano Abram, Ariel Nof, Claudio Orlandi, Peter Scholl, and Omer Shlomovits. Low-bandwidth threshold ECDSA via pseudorandom correlation generators. In *2022 IEEE Symposium on Security and Privacy*, pages 2554–2572. IEEE Computer Society Press, May 2022.

AS22. Damiano Abram and Peter Scholl. Low-communication multiparty triple generation for SPDZ from ring-LPN. In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *PKC 2022, Part I*, volume 13177 of *LNCS*, pages 221–251. Springer, Heidelberg, March 2022.

Bar04. M. Bardet. Étude des systèmes algébriques surdéterminés. applications aux codes correcteurs et à la cryptographie. In *Theses, Université Pierre et Marie Curie - Paris VI*, 2004.

BCG+19a. Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Peter Rindal, and Peter Scholl. Efficient two-round OT extension and silent non-interactive secure computation. In Lorenzo Cavallaro, Johannes Kinder,

XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 291–308. ACM Press, November 2019.

BCG+19b. Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators: Silent OT extension and more. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 489–518. Springer, Heidelberg, August 2019.

BCG+20. Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators from ring-LPN. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 387–416. Springer, Heidelberg, August 2020.

BCG+22. Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Nicolas Resch, and Peter Scholl. Correlated pseudorandomness from expand-accumulate codes. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 603–633. Springer, Heidelberg, August 2022.

BCGI18. Elette Boyle, Geoffroy Couteau, Niv Gilboa, and Yuval Ishai. Compressing vector OLE. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 896–912. ACM Press, October 2018.

BCP97. Wieb Bosma, John Cannon, and Catherine Playoust. The Magma algebra system. I. The user language. *J. Symbolic Comput.*, 24(3-4):235–265, 1997. Computational algebra and number theory (London, 1993).

BDSW23. Carsten Baum, Simon Dittmer, Peter Scholl, and Xiao Wang. Sok: vector ole-based zero-knowledge protocols. *Designs, Codes and Cryptography*, 91(8):3527–3561, 2023.

Ber68. Elwyn R. Berlekamp. *Algebraic coding theory*. McGraw-Hill series in systems science. McGraw-Hill, 1968.

BFKL94. Avrim Blum, Merrick L. Furst, Michael J. Kearns, and Richard J. Lipton. Cryptographic primitives based on hard learning problems. In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 278–291. Springer, Heidelberg, August 1994.

BGI14. Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 501–519. Springer, Heidelberg, March 2014.

BGI15. Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 337–367. Springer, Heidelberg, April 2015.

BJMM12. Anja Becker, Antoine Joux, Alexander May, and Alexander Meurer. Decoding random binary linear codes in $2^{n/20}$: How $1 + 1 = 0$ improves information set decoding. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 520–536. Springer, Heidelberg, April 2012.

BKW03. Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. In *Journal of the ACM*, 2003.

BLP11. Daniel J. Bernstein, Tanja Lange, and Christiane Peters. Smaller decoding exponents: Ball-collision decoding. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 743–760. Springer, Heidelberg, August 2011.

BM18. Leif Both and Alexander May. Decoding linear codes with high error rate and its impact for lpn security. In Tanja Lange and Rainer Steinwandt, editors, *Post-Quantum Cryptography - 9th International Conference, PQCrypto 2018*, pages 25–46, Heidelberg, 2018. Springer.

BØ23. Pierre Briaud and Morten Øygarden. A new algebraic approach to the regular syndrome decoding problem and implications for PCG constructions. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 391–422. Springer, Heidelberg, April 2023.

BR17. Andrej Bogdanov and Alon Rosen. *Pseudorandom Functions: Three Decades Later*, pages 79–158. Springer International Publishing, Cham, 2017.

CCJ23. Eliana Carozza, Geoffroy Couteau, and Antoine Joux. Short signatures from regular syndrome decoding in the head. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part V*, volume 14008 of *Lecture Notes in Computer Science*, pages 532–563. Springer, 2023.

CKPS00. Nicolas Courtois, Alexander Klimov, Jacques Patarin, and Adi Shamir. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 392–407. Springer, Heidelberg, May 2000.

CRR21. Geoffroy Couteau, Peter Rindal, and Srinivasan Raghuraman. Silver: Silent VOLE and oblivious transfer from hardness of decoding structured LDPC codes. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part III*, volume 12827 of *LNCS*, pages 502–534, Virtual Event, August 2021. Springer, Heidelberg.

DILO22. Samuel Dittmer, Yuval Ishai, Steve Lu, and Rafail Ostrovsky. Authenticated garbling from simple correlations. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part IV*, volume 13510 of *LNCS*, pages 57–87. Springer, Heidelberg, August 2022.

Ds17. Jack Doerner and abhi shelat. Scaling ORAM for secure computation. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 523–535. ACM Press, October / November 2017.

ES24. Andre Esser and Paolo Santini. Not just regular decoding: Asymptotics and improvements of regular syndrome decoding attacks. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part VI*, volume 14925 of *Lecture Notes in Computer Science*, pages 183–217. Springer, 2024.

FJR22. Thibauld Feneuil, Antoine Joux, and Matthieu Rivain. Syndrome decoding in the head: Shorter signatures from zero-knowledge proofs. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 541–572. Springer, Heidelberg, August 2022.

FKI07. Marc P. C. Fossorier, Kazukuni Kobara, and Hideki Imai. Modeling bit flipping decoding based on nonorthogonal check sums with application to iterative decoding attack of mceliece cryptosystem. *IEEE Transactions on Information Theory*, 53(1):402–411, 2007.

FS09a. Matthieu Finiasz and Nicolas Sendrier. Security bounds for the design of code-based cryptosystems. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 88–105. Springer, Heidelberg, December 2009.

FS09b. P. Flajolet and R. Sedgewick. Analytic combinatorics. Cambridge University Press, 2009.

GGM84. Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions (extended abstract). In *25th FOCS*, pages 464–479. IEEE Computer Society Press, October 1984.

HOSS18. Carmit Hazay, Emmanuela Orsini, Peter Scholl, and Eduardo Soria-Vazquez. TinyKeys: A new approach to efficient multi-party computation. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 3–33. Springer, Heidelberg, August 2018.

Jab01. A. Al Jabri. A statistical decoding algorithm for general linear block codes. In Bahram Honary, editor, *Cryptography and Coding*, pages 1–8, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

LWYY22. Hanlin Liu, Xiao Wang, Kang Yang, and Yu Yu. The hardness of LPN over any integer ring and field for PCG applications. Cryptology ePrint Archive, Report 2022/712, 2022. https://eprint.iacr.org/2022/712.

Lyu05. Vadim Lyubashevsky. The parity problem in the presence of noise, decoding random linear codes, and the subset sum problem. In Chandra Chekuri, Klaus Jansen, Jose D. P. Rolim, and Luca Trevisan, editors, *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, pages 378–389, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

MMT11. Alexander May, Alexander Meurer, and Enrico Thomae. Decoding random linear codes in $\tilde{\mathcal{O}}(2^{0.054n})$. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 107–124. Springer, Heidelberg, December 2011.

MO15. Alexander May and Ilya Ozerov. On computing nearest neighbors with applications to decoding of binary linear codes. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 203–228. Springer, Heidelberg, April 2015.

Pra62. Eugene Prange. The use of information sets in decoding cyclic codes. In *IRE Transactions on Information Theory*, 1962.

Roy22. Lawrence Roy. SoftSpokenOT: Quieter OT extension from small-field silent VOLE in the minicrypt model. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part I*, volume 13507 of *LNCS*, pages 657–687. Springer, Heidelberg, August 2022.

RR. Peter Rindal and Lawrence Roy. libOTe: an efficient, portable, and easy to use Oblivious Transfer Library. https://github.com/osu-crypto/libOTe.

RRT23. Srinivasan Raghuraman, Peter Rindal, and Titouan Tanguy. Expand-convolute codes for pseudorandom correlation generators from LPN. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part IV*, volume 14084 of *LNCS*, pages 602–632. Springer, Heidelberg, August 2023.

RS21. Peter Rindal and Phillipp Schoppmann. VOLE-PSI: Fast OPRF and circuit-PSI from vector-OLE. In Anne Canteaut and Fran**c**cois-Xavier Standaert, editors, *EUROCRYPT 2021, Part II*, volume 12697 of *LNCS*, pages 901–930. Springer, Heidelberg, October 2021.

SGRR19. Phillipp Schoppmann, Adrià Gascón, Leonie Reichert, and Mariana Raykova. Distributed vector-OLE: Improved constructions and implementation. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 1055–1072. ACM Press, November 2019.

Shp09. Amir Shpilka. Constructions of low-degree and error-correcting ϵ-biased generators. In *computational complexity*, 2009.

Ste89. Jacques Stern. A method for finding codewords of small weight. In Gerard Cohen and Jacques Wolfmann, editors, *Coding Theory and Applications*, pages 106–113, Berlin, Heidelberg, 1989. Springer Berlin Heidelberg.

Wie86. D. Wiedemann. Solving sparse linear equations over finite fields. In *IEEE Transac- tions on Information Theory*, 1986.

WYKW21. Chenkai Weng, Kang Yang, Jonathan Katz, and Xiao Wang. Wolverine: Fast, scalable, and communication-efficient zero-knowledge proofs for boolean and arithmetic circuits. In *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*, pages 1074–1091. IEEE, 2021.

WYY+22. Chenkai Weng, Kang Yang, Zhaomin Yang, Xiang Xie, and Xiao Wang. AntMan: Interactive zero-knowledge proofs with sublinear communication. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 2901–2914. ACM Press, November 2022.

YSWW21. Kang Yang, Pratik Sarkar, Chenkai Weng, and Xiao Wang. QuickSilver: Efficient and affordable zero-knowledge proofs for circuits and polynomials over any field. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 2986–3001. ACM Press, November 2021.

Supplementary Material

Disclaimer

Case studies, comparisons, statistics, research and recommendations are provided "AS IS" and intended for informational purposes only and should not be relied upon for operational, marketing, legal, technical, tax, financial or other advice. Visa Inc. neither makes any warranty or representation as to the completeness or accuracy of the information within this document, nor assumes any liability or responsibility that may result from reliance on such information. The Information contained herein is not intended as investment or legal advice, and readers are encouraged to seek the advice of a competent professional where such advice is required.

These materials and best practice recommendations are provided for informational purposes only and should not be relied upon for marketing, legal, regulatory or other advice. Recommended marketing materials should be independently evaluated in light of your specific business needs and any applicable laws and regulations. Visa is not responsible for your use of the marketing materials, best practice recommendations, or other information, including errors of any kind, contained in this document.