Stationary Syndrome Decoding for Improved PCGs

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

[1] Georgia Institute of Technology, Atlanta, GA, USA
kolesnikov@gatech.edu
[2] Visa Research and MIT, Cambridge, MA, USA
[3] Visa Research, Foster City, CA, USA

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

© International Association for Cryptologic Research 2025
Y. Tauman Kalai and S. F. Kamara (Eds.): CRYPTO 2025, LNCS 16000, pp. 284–317, 2025.
https://doi.org/10.1007/978-3-032-01855-7_10

$\boldsymbol{e} \in \mathbb{F}^n$ 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:

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

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. We refer to [KPRR25] for more detail.

Degree 2 Correlations. The setup for degree 2 correlations from Ring-LPN [BCG+20, BBC+24, LXYY25], 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. The recent implementation of [RR] demonstrates that the dominant cost is simply generating the noise vector using $O(nt)$ calls to a PRG. 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. $O(\mathsf{poly}(t'))$. Alternatively, [BGH+25] gives a distributed multi-point function, which requires only $O(n)$ calls to a PRG and $O(nt)$ simple operations. However, the concrete costs of generating the distributed point function keys remain very high and likely require new protocol improvements.

To reduce the high concrete cost of these protocols, we propose using SSD, 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 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. We believe this is an extremely promising direction for future exploration. We refer to [KPRR25] for more detail.

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 Sect. 3, we review existing works on syndrome decoding (SD) and related attacks. In this section, 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 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 Sect. 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, $(\mathbf{G}, \mathbf{G}\boldsymbol{e})$ 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 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.

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 $\boldsymbol{e} \in \mathbb{F}^n$, $(\boldsymbol{A}, \boldsymbol{A}\boldsymbol{s} + \boldsymbol{e})$ 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 $(n, m, \mathbb{F}, \mathcal{D}, \mathcal{H})$-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. For those less familiar with agebraic geometry we provide a self-contained introduction in [KPRR25].

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 (see [KPRR25]). 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 Fig. 1.

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

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.

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:*

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

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

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} (see [KPRR25]). The key difference is that d_{reg} is usually used for homogeneous systems solved with Gröbner bases, 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 consider 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 [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 $j_1 < j_2 \in [n/t]$.[1] 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 $\mathbf{G}\boldsymbol{e} - \boldsymbol{b} = 0$.*

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

– $\mathcal{S}$ *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 d_{wit} 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 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 rerunning 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. More 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.

Like SD (Sect. 3.4), the SSD assumption is false for some choices of $\mathcal{G}$ and $\mathcal{L}$. This definition serves as template that we make concrete in Sect. 5 and 6. We give concrete parameters in Sect. 7 for when we believe that SSD holds.

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* $(n, m, q, \mathbb{F}, \mathcal{L}, \mathcal{H})$-$\mathsf{SLPN}$ *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 Sect. 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 (Sect. 4.1) and discuss SSD's implications for the performance of pseudorandom correlation generators (Sect. 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 Sect. 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.[2] 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 Sect. 3.6 and Fig. 1), when applied to the SD problem with regular

[2] 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 Sect. 5 for formal detail.

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 [KPRR25]. 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 $[\![\boldsymbol{v}\Delta]\!]$.

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) or a distributed point function (DPF), where the input to the ith PPRF is the index of the non-zero in $\boldsymbol{e}_i$ 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).

Without going into detail, the PPRF protocol generates a GGM tree with n' leaves. A version of the GGM tree will be held by both parties and is generated by applying a PRG to the value of each parent node and assigning the result to its children. In total, $2n'$ PRG calls are made to evaluate the full tree. Additionally, the parties perform an OT for each level of the GGM tree, which allows the receiver to know all but one of the seeds at each level. These missing seeds all lie on the path from the root to the leaf node that corresponds to the non-zero of $\boldsymbol{e}_i$. The expansion of the GGM tree is largely independent of the value assigned to the leaf, in this case a sharing of Δ. SSD can be used to optimize this process by only expanding the tree once and then repeatedly derandomizing the leaf value for each instance, a $2\times$ reduction in the number of PRG calls and requiring no additional OTs. See [KPRR25] for a more detailed description.

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* $(n', m', \mathbb{F}, \mathcal{D}, \mathcal{H}')$-LPN *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 $\boldsymbol{v} \cdot (\mathbf{As} + \mathbf{e})$ is distinguishable from uniform. We consider two cases.

Non-codeword $\boldsymbol{v}$. Let us define the code $\mathcal{C} = \{\boldsymbol{c} \in \mathbb{F}^n : \boldsymbol{cA} = 0\} = \{\boldsymbol{mG} : \boldsymbol{m} \in \mathbb{F}^k\}$. 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, $\boldsymbol{v}^\mathsf{T}\boldsymbol{As} = \boldsymbol{u}^\mathsf{T}\boldsymbol{s} = r$ 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 $\max_{v \notin \mathcal{C}}(\mathsf{bias}_{\boldsymbol{v}}(\boldsymbol{As} + \boldsymbol{e})) = 0$.

Codeword $\boldsymbol{v}$. As just described, when $\boldsymbol{v}$ is a codeword the randomness contributed by $\boldsymbol{s}$ vanishes. That is, $\boldsymbol{v}^\mathsf{T}(\boldsymbol{As} + \boldsymbol{e}) = \boldsymbol{v}^\mathsf{T}\boldsymbol{e}$. 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

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

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 $\boldsymbol{v} \cdot (\boldsymbol{As} + \boldsymbol{e})$ 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 Sect. 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 $\mathbf{G}'_i := \left(\mathbf{G}_{i,*,L_1}, \dots \mathbf{G}_{i,*,L_t}\right) \in \mathbb{F}^{m \times t}$ 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 Sect. 6.1. We continue by computing our system's Hilbert series in Sect. 6.2, use it to estimate the witness degree in Sect. 6.3, repeat the same procedure for the hybrid approach in Sect. 6.4, and then evaluate the attack's impact on SSD's security in Sect. 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 $q^2 t \binom{n/t}{2} = q^2 n^2/2t - q^2 n/2$ 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 (Sect. 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 (Sect. 3.6). To see why, consider $f_1 := \mathbf{e}_{1,1,1}\mathbf{e}_{1,1,2}$ and $f_2 := \mathbf{e}_{1,1,2}\mathbf{e}_{1,1,3}$ 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 Sect. 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

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

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 $\mathcal{R}^{(\mathsf{h})} := \{r_1, \ldots, r_{qk}\}$ 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 $\langle \mathcal{S}^{(\mathsf{h})}, r_1, \ldots, r_j \rangle$. 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

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

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 (Sect. 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. We defer the presentation of Modeling 4 to [KPRR25] and note that it follows a similar methodology.

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 $\{\mathbf{x}_1 - a_1, \ldots, \mathbf{x}_n - a_n\}$.[3] 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.

[3] 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.

As in [BØ23], our Assumption 1 and [KPRR25]'s Assumption 2 associated with Modeling 4 imply that the relevant Macaulay matrix has maximal rank. Now, consider the Hilbert series from Theorem 4 and 5 (see [KPRR25]) 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 (see [KPRR25]) prior to truncation. We have experimentally verified these estimates as we discuss in Sect. 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 2. 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 $\overline{\mathbf{P}_{f,\mu}^{-1}}$ 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 2. *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 2, 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 Sect. 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 Sect. 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 Sect. 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 (Fig. 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 implies that the attack is not able to take advantage of the additional information and becomes slower 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 (Sect. 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 Fig. 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 Parameter Selection

We use our findings in Sect. 5 on SSD security against linear tests to choose the appropriate amount of noise t for Fig. 4 and 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 $t \approx 60$ gives 128 bits of security against known linear attacks, while our algebraic attacks in Sect. 6.5 suggest $t \approx 60$ depending on n. Moreover, the best attack strategy is to simply attack a single instance.

The main difference between SD and SSD parameters arises in small fields, e.g. $\mathbb{F}_2$. In this case, since our noise is sampled uniformly instead of set to 1, one must increase the noise to compensate. This difference vanishes over large fields, and fixed parameters for SD and SSD appear to give the same security level.

This suggests that one can instantiate SSD using the same parameters as SD for large fields, or increase the noise by the statistical security parameter for small fields so that each instance is expected to have at least as many non-zeros as in SD. However, for the time being, we opt for being conservative and choosing parameters based on the linear test framework.

8 Experimental Evaluation

Implementation & Setting. We present a detailed description of how to adapt existing PCG protocols in [KPRR25]. The crux for degree 1 PCG protocols is (1) generating a sharing of $\Delta \boldsymbol{e}$ and (2) compressing it by the generator matrix $\mathbf{G}$. SSD optimizes the generation of $[\![\Delta \boldsymbol{e}]\!]$ across q PCG instances. Thus, we implement our SSD-based $[\![\Delta \boldsymbol{e}]\!]$ 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.

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

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.

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 (Fig. 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 (Fig. 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 Sect. 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 Fig. 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 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 fixed qk, it is preferable to create more smaller instances (i.e. large q and small k) rather than fewer large ones. This is further exacerbated by the fact that for small instances, multiplying by $\mathbf{G}$ is also 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. Akavia, A., Bogdanov, A., Guo, S., Kamath, A., Rosen, A.: Candidate weak pseudorandom functions in ac0 $\odot$ mod2. In: *Proceedings of the 5th Conference on Innovations in Theoretical Computer Science*, ITCS '14, pp. 251–260, New York, NY, USA (2014). Association for Computing Machinery

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

ANO+22. Abram, D., Nof, A., Orlandi, C., Scholl, P., Shlomovits, O.: Low-bandwidth threshold ECDSA via pseudorandom correlation generators. In: *2022 IEEE Symposium on Security and Privacy*, pp. 2554–2572. IEEE Computer Society Press (2022)

AS22. Abram, D., Scholl, P.: Low-communication multiparty triple generation for SPDZ from ring-LPN. In: Hanaoka, G., Shikata, J., Watanabe, Y. (eds.) PKC 2022. Part I, volume 13177 of LNCS, pp. 221–251. Springer, Heidelberg (2022)

BBC+24. Bombar, M., Bui, D., Couteau, G., Couvreur, A., Ducros, C., Schreiber, S.S.: FOLEAGE: $\mathbb{F}_4$OLE-based multi-party computation for boolean circuits. In: LNCS, pp. 69–101 (2024)

BCG+19a. Boyle, E., et al.: Efficient two-round OT extension and silent non-interactive secure computation. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J., (eds.) *ACM CCS 2019*, pp. 291–308. ACM Press (2019)

BCG+19b. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y., Kohl, L., Scholl, P.: Efficient pseudorandom correlation generators: silent OT extension and more. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. Part III, volume 11694 of LNCS, pp. 489–518. Springer, Heidelberg (2019)

BCG+20. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y., Kohl, L., Scholl, P.: Efficient pseudorandom correlation generators from ring-LPN. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020. Part II, volume 12171 of LNCS, pp. 387–416. Springer, Heidelberg (2020)

BCG+22. Boyle, E., et al.: Correlated pseudorandomness from expand-accumulate codes. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022. Part II, volume 13508 of LNCS, pp. 603–633. Springer, Heidelberg (2022)

BCGI18. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y.: Compressing vector OLE. In: Lie, D., Mannan, M., Backes, M., Wang, X., (eds.), *ACM CCS 2018*, pp. 896–912. ACM Press (2018)

BCP97. Bosma, W., Cannon, J., Playoust, C.: 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. Baum, C., Dittmer, S., Scholl, P., Wang, X.: Sok: vector ole-based zero-knowledge protocols. Des. Codes Crypt. **91**(8), 3527–3561 (2023)

Ber68. Berlekamp, E.R.: *Algebraic Coding Theory*. McGraw-Hill Series in Systems Science. McGraw-Hill (1968)

BFKL94. Blum, A., Furst, M.L., Kearns, M.J., Lipton, R.J.: Cryptographic primitives based on hard learning problems. In: Stinson, D.R. (ed.) CRYPTO'93. LNCS, vol. 773, pp. 278–291. Springer, Heidelberg (1994)

BGH+25. Boyle, E., Gilboa, N., Hamilis, M., Ishai, Y., Tu, Y.: Improved constructions for distributed multi-point functions . In: *2025 IEEE Symposium on Security and Privacy (SP)*, pp. 2414–2432, Los Alamitos, CA, USA (2025). IEEE Computer Society

BJMM12. Becker, A., Joux, A., May, A., Meurer, A.: Decoding random binary linear codes in $2^{n/20}$: How $1 + 1 = 0$ improves information set decoding. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 520–536. Springer, Heidelberg (2012)

BKW03. Blum, A., Kalai, A., Wasserman, H.: 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: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 743–760. Springer, Heidelberg (2011)

BM18. Both, L., May, A.: Decoding linear codes with high error rate and its impact for LPN security. In: Lange, T., Steinwandt, R. (eds.) PQCrypto 2018. LNCS, vol. 10786, pp. 25–46. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-79063-3_2

BØ23. Briaud, P., Øygarden, M.: A new algebraic approach to the regular syndrome decoding problem and implications for PCG constructions. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023. Part V, volume 14008 of LNCS, pp. 391–422. Springer, Heidelberg (2023)

BR17. Bogdanov, A., Rosen, A.: *Pseudorandom Functions: Three Decades Later*, pp. 79–158. Springer International Publishing, Cham (2017)

CCJ23. Carozza, E., Couteau, G., Joux, A.: Short signatures from regular syndrome decoding in the head. In Carmit Hazay and Martijn Stam, editors, Advances in Cryptology - EUROCRYPT **2023**, 532–563 (2023)

CKPS00. Courtois, N., Klimov, A., Patarin, J., Shamir, A.: Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 392–407. Springer, Heidelberg (2000)

CRR21. Couteau, G., Rindal, P., Raghuraman, S.: Silver: silent VOLE and oblivious transfer from hardness of decoding structured LDPC codes. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021. LNCS, vol. 12827, pp. 502–534. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-84252-9_17

DILO22. Dittmer, S., Ishai, Y., Steve, L., Ostrovsky, R.: Authenticated garbling from simple correlations. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022. Part IV, volume 13510 of LNCS, pp. 57–87. Springer, Heidelberg (2022)

Ds17. Doerner, J., shelat, A..: Scaling ORAM for secure computation. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D., (eds.). *ACM CCS 2017*, pp. 523–535. ACM Press (2017)

ES24. Esser, A., Santini, P.: Not just regular decoding: asymptotics and improvements of regular syndrome decoding attacks. In Leonid Reyzin and Douglas Stebila, editors, Advances in Cryptology - CRYPTO **2024**, 183–217 (2024)

FJR22. Feneuil, T., Joux, A., Rivain, M.: Syndrome decoding in the head: shorter signatures from zero-knowledge proofs. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022. Part II, volume 13508 of LNCS, pp. 541–572. Springer, Heidelberg (2022)

FKI07. Fossorier, M.P.C., Kobara, K., Imai, H.: Modeling bit flipping decoding based on nonorthogonal check sums with application to iterative decoding attack of mceliece cryptosystem. *IEEE Trans. Info. Theory* **53**(1), 402–411 (2007)

FS09a. Finiasz, M., Sendrier, N.: Security bounds for the design of code-based cryptosystems. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 88–105. Springer, Heidelberg (2009)

FS09b. Flajolet, P., Sedgewick, R.: Analytic Combinatorics. Cambridge University Press (2009)
GGM84. Goldreich, O., Goldwasser, S., Micali, S..: How to construct random functions (extended abstract). In: *25th FOCS*, pp. 464–479. IEEE Computer Society Press (1984)
HOSS18. Hazay, C., Orsini, E., Scholl, P., Soria-Vazquez, E.: TinyKeys: a new approach to efficient multi-party computation. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018. Part III, volume 10993 of LNCS, pp. 3–33. Springer, Heidelberg (2018)
Jab01. Jabri, A.A.: A statistical decoding algorithm for general linear block codes. In: Honary, B., (ed.), *Cryptography and Coding*, pp. 1–8, Berlin, Heidelberg (2001). Springer Berlin Heidelberg
KPRR25. Kolesnikov, V., Peceny, S., Raghuraman, S., Rindal, P.: Stationary syndrome decoding for improved PCGs. Cryptology ePrint Archive, Paper 2025/295 (2025)
LWYY22. Liu, H., Wang, X., Yang, K., Yu, Y.: 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
LXYY25. Li, Z., Xing, C., Yao, Y., Yuan, C.: Efficient pseudorandom correlation generators for any finite field. In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 145–175. Springer (2025)
Lyu05. Lyubashevsky, V.: The parity problem in the presence of noise, decoding random linear codes, and the subset sum problem. In: Chekuri, C., Jansen, K., Rolim, J.D.P., Trevisan, L., (eds.) *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, pp. 378–389 (2005)
MMT11. May, A., Meurer, A., Thomae, E.: Decoding random linear codes in $\tilde{\mathcal{O}}(2^{0.054n})$. In: Lee, D.H., Wang, X., (eds.), *ASIACRYPT 2011*, vol. 7073 of *LNCS*, pp. 107–124. Springer, Heidelberg (2011)
MO15. May, A., Ozerov, I.: On computing nearest neighbors with applications to decoding of binary linear codes. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. Part I, volume 9056 of LNCS, pp. 203–228. Springer, Heidelberg (2015)
Pra62. Prange, E.: The use of information sets in decoding cyclic codes. In: *IRE Transactions on Information Theory* (1962)
Roy22. Roy, L.: SoftSpokenOT: quieter OT extension from small-field silent VOLE in the minicrypt model. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022. Part I, volume 13507 of LNCS, pp. 657–687. Springer, Heidelberg (2022)
RR. Rindal, P., Roy, L.: libOTe: an efficient, portable, and easy to use Oblivious Transfer Library. https://github.com/osu-crypto/libOTe
RRT23. Raghuraman, S., Rindal, P., Tanguy, T.: Expand-convolute codes for pseudorandom correlation generators from LPN. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023. Part IV, volume 14084 of LNCS, pp. 602–632. Springer, Heidelberg (2023)
RS21. Rindal, P., Schoppmann, P.: VOLE-PSI: fast OPRF and circuit-PSI from vector-OLE. In: Canteaut, A., Standaert, F.-X. (eds.) EUROCRYPT 2021. Part II, volume 12697 of LNCS, pp. 901–930. Springer, Heidelberg (2021)
SGRR19. Schoppmann, P., Gascón, A., Reichert, L., Raykova, M.: Distributed vector-OLE: Improved constructions and implementation. In: Cavallaro,

L., Kinder, J., Wang, X., Katz, J., (eds.) *ACM CCS 2019*, pp. 1055–1072. ACM Press (2019)

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

Ste89. Stern, J.: A method for finding codewords of small weight. In: Cohen, G., Wolfmann, J., (eds.) *Coding Theory and Applications*, pp. 106–113, Berlin, Heidelberg (1989). Springer Berlin Heidelberg

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

WYKW21. Weng, C., Yang, K., Katz, J., Wang, X.: 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*, pp. 1074–1091. IEEE (2021)

WYY+22. Weng, C., Yang, K., Yang, Z., Xie, X., Wang, X.: AntMan: interactive zero-knowledge proofs with sublinear communication. In: Yin, H., Stavrou, A., Cremers, C., Shi, E.,(eds.) *ACM CCS 2022*, pp. 2901–2914. ACM Press (2022)

YSWW21. Yang, K., Sarkar, P., Weng, C., Wang, X.: QuickSilver: efficient and affordable zero-knowledge proofs for circuits and polynomials over any field. In: Vigna, G., Shi, E., (eds.) *ACM CCS 2021*, pp. 2986–3001. ACM Press (2021)