Correlated Pseudorandomness from the Hardness of Quasi-Abelian Decoding

Maxime Bombar, Geoffroy Couteau, Alain Couvreur, Clément Ducros

Correlated Pseudorandomness from the Hardness of Quasi-Abelian Decoding

Maxime Bombar[1,2], Geoffroy Couteau[3,4], Alain Couvreur[2,1], and Clément Ducros[4,2] ⋆

[1] Laboratoire LIX, École Polytechnique, Institut Polytechnique de Paris, 1 rue Honoré d'Estienne d'Orves, 91120 Palaiseau Cedex
[2] INRIA
[3] CNRS
[4] IRIF, Université Paris Cité
{maxime.bombar, alain.couvreur}@inria.fr
{couteau, cducros}@irif.fr

Abstract. Secure computation often benefits from the use of correlated randomness to achieve fast, non-cryptographic online protocols. A recent paradigm put forth by Boyle *et al.* (CCS 2018, Crypto 2019) showed how *pseudorandom correlation generators* (PCG) can be used to generate large amounts of useful forms of correlated (pseudo)randomness, using minimal interactions followed solely by local computations, yielding *silent* secure two-party computation protocols (protocols where the preprocessing phase requires almost no communication). Furthermore, *programmable* PCG's can be used similarly to generate multiparty correlated randomness to be used in silent secure N-party protocols. Previous works constructed very efficient (non-programmable) PCG's for correlations such as random oblivious transfers. However, the situation is less satisfying for the case of *random oblivious linear evaluation* (OLE), which generalises oblivious transfers over large fields, and are a core resource for secure computation of arithmetic circuits. The state-of-the-art work of Boyle *et al.* (Crypto 2020) constructed programmable PCG's for OLE, but their work suffers from two important downsides: (1) it only generates OLE's over *large fields*, and (2) it relies on a relatively new "splittable" ring-LPN assumption, which lacks strong security foundations.
In this work, we construct new programmable PCG's for the OLE correlation, that overcome both limitations. To this end, we introduce the *quasi-abelian syndrome decoding problem* (QA-SD), a family of assumptions which generalises the well-established quasi-cyclic syndrome decoding assumption. Building upon QA-SD, we construct new programmable PCG's for OLE's over any field $\mathbb{F}_q$ with $q > 2$. Our analysis also sheds light on the security of the ring-LPN assumption used in Boyle *et al.* (Crypto 2020). Using our new PCG's, we obtain the first efficient N-party silent secure computation protocols for computing general arithmetic circuit over $\mathbb{F}_q$ for any $q > 2$.

Keywords: Pseudorandom correlation generators, oblivious linear evaluation, quasi-abelian codes, silent secure computation

⋆ This work was funded by the French Agence Nationale de la Recherche through the France 2030 ANR Projects ANR-22-PECY-003 SecureCompute and ANR-22-PETQ-0008 PQ-TLS, the ANR BARRACUDA (ANR-21-CE39-0009-BARRACUDA), the ANR SCENE (ANR-20-CE39-0001-SCENE), and by the DIM RFSI through the project LICENCED.

Table of Contents

1 Introduction

Correlated randomness is a powerful resource in secure computation. Following the seminal work of Beaver [Bea92], many lightweight, concretely efficient secure computation protocols have been designed in a model where the parties have access to long trusted correlated random strings: $\Omega(n)$-length instances of a simple correlation enable securely computing circuits with n gates. Depending on the setting, various correlations are used: for example, oblivious transfer (OT) correlations are used for two-party (semi-honest) secure computation of Boolean circuits, and oblivious

linear evaluation (OLE) correlations, which generalize OT over arbitrary fields, enable 2-party semi-honest secure computation of arithmetic circuits. Eventually, n-party Beaver triples enable n-party semi-honest secure computation of arithmetic circuits, and authenticated Beaver triples enable maliciously secure computation of arithmetic circuits.

Since protocols in the correlated randomness paradigm are lightweight and very efficient, they gave rise to a popular, two-stage approach: first, the parties run an input-independent *preprocessing phase*, which securely generates and distributes the correlated strings, and second, these strings are consumed by an *online* protocol. Traditional approaches for implementing the preprocessing phase had $\Omega(n)$ communication [IKNP03, DPSZ12, KPR18] and formed the efficiency bottleneck of the overall protocol. The situation changed recently with a new approach, introduced in [BCG+17, BCGI18, BCG+19b] and further refined in many subsequent works [BCG+19a, SGRR19, BCG+20b, BCG+20a, YWL+20, CRR21, BCG+22], with appealing efficiency features such as a one-time, $o(n)$-communication phase followed solely by local computation. At the heart of this approach is the notion of *pseudorandom correlation generators* (PCG's). Informally, a PCG has two algorithms: $\mathsf{Gen}(1^\lambda)$ outputs two *short correlated keys* $(\mathsf{k}_0, \mathsf{k}_1)$, and $R_\sigma \leftarrow \mathsf{Expand}(\mathsf{k}_\sigma)$ stretches k_σ into a long string R_σ, such that (R_0, R_1) is a pseudorandom instance of a target correlation. PCG's enable an efficient, two-stage *silent* preprocessing phase:

1. First, the parties securely distribute the short PCG seeds, using a small amount of work and communication (often independent of the circuit size).
2. Second, the parties locally stretch the PCG's into long correlated pseudorandom strings: this part is the bulk of the computation, but does not require any further communication among the parties.

This is the model of secure computation with *silent preprocessing* (or *silent secure computation* in short), where most of the preprocessing phase is pushed offline. Previous works gave efficient constructions of PCG's for various correlations such as OT's [BCG+19a, SGRR19, CRR21, BCG+22], vector-OLE [BCGI18], OLE's over large fields [BCG+20b], authenticated Beaver triples [BCG+20b] and many more. These PCG's all build upon a common template, which combines function secret sharing (FSS) for simple function classes with suitable variants of the syndrome decoding assumption.

1.1 PCG's: State of the Art and Challenges

Very efficient constructions of PCG's for the OT correlations have been proposed [BCG+19a, SGRR19, CRR21, BCG+22]. The most recent constructions (see [CRR21, BCG+22]) allow to generate millions of random OT's per second on one core of a standard laptop. Combined with the GMW protocol, they effectively enable extremely efficient two-party secure computation of Boolean circuits in the semi-honest model, with minimal communication in the preprocessing phase (a few dozen of kilobytes, independent of the circuit size), followed by cheap local computation, and a fast online phase (exchanging four bits per AND gate).

The situation, however, is much less satisfactory in essentially all other standard settings of secure computation, where the OT correlation is not the best choice of correlation[5], and one of the major open problems in this line of work is to improve this state of affair. Concretely, when targeting any one of *multiparty* computation (with $N > 2$ parties), *arithmetic* computation (for arithmetic circuits over a field $\mathbb{F}$ of size $|\mathbb{F}| > 2$), or *malicious* security, the best-known PCG-based solutions lag way behind the state of the art for 2-party, semi-honest secure computation of Boolean circuits. At a high level, the problem is twofold:

– Secure computation of arithmetic circuits requires the OLE correlation rather than the OT correlation, and the constructions of [BCG+19a, SGRR19, CRR21, BCG+22] are inherently limited to the OT correlation. To handle OLE, a fundamentally different approach is required.

[5] While the OT correlation is complete even for N-party malicious secure computation of arithmetic circuits, its use induces large overheads in the online phase: an $\Omega(N^2)$ communication overhead for handling N parties, an $\Omega(\log^2 |\mathbb{F}|)$ overhead for handling larger fields $\mathbb{F}$, and an $\Omega(\lambda)$ overhead for handling malicious parties. In contrast, other choices of correlated randomness can avoid each of these overheads.

– Additionally, handling $N > 2$ parties or achieving malicious security both require the underlying PCG for OLE (or OT) to satisfy a property known as *programmability* (at a high level, programmability allows both to generate N-party correlations from $O(N^2)$ 2-party correlations, which is required because all known PCG's are inherently restricted to the 2-party setting, and to *authenticate* 2-party correlations with a MAC, which is needed for malicious security). Unfortunately, the constructions of [BCG+19a, SGRR19, CRR21, BCG+22] cannot (by design) achieve programmability.

These two limitations were addressed in the recent work of [BCG+20b], which introduced the first (reasonably efficient) construction of programmable PCG for the OLE correlation. While not as efficient as the best known PCG's for OT, it can produce around 10^5 OLE's per second on a standard laptop. However, the result of [BCG+20b] suffers from two important downsides:

– it can only produce OLE's over *large enough fields* (concretely, the field size must be larger than the circuit size). This leaves open the question of designing efficient programmable PCG's for OLE over small fields.
– it relies on a relatively new *ring*-LPN *with splittable polynomial* assumption which states, in essence, that $(a, as + e)$ is hard to distinguish from (a, b), where a, b are random polynomials from a ring $\mathcal{R} = \mathbb{F}_p[X]/(P(X))$ where P splits into $\deg(P)$ linear factors, and s, e are random *sparse* polynomials from $\mathcal{R}$. The ring-LPN assumption was introduced a decade ago in [HKL+12] to build efficient authentication protocols, and it has received some attention from the cryptography community [BL12, DP12, LP15, GJL15, BCG+20b, BCD22]. However, so far, we lack both a principled understanding of *which* choice of the underlying polynomial P yield solid instances (beyond the observation that reducible polynomials seem to enable more efficient attacks [GJL15, BCG+20b]), and a general methodology to argue the security of ring-LPN assumptions.

At a high level, the construction of PCG for OLE from [BCG+20b] proceeds by generating a single large pseudorandom OLE correlation over a polynomial ring $\mathcal{R} = \mathbb{F}_p[X]/(P(X))$, assuming the hardness of the ring-LPN assumption over $\mathcal{R}$. When P splits into $N = \deg(P)$ linear factors, the Chinese Remainder Theorem permits to convert this large OLE correlation over $\mathcal{R}$ into N OLE correlations over $\mathbb{F}_p$ (by reducing it modulo each of the factors of P). Note that the condition that P splits requires $|\mathbb{F}_p| \geqslant N$, hence the restriction to large fields. Because the ring-LPN assumption with a splittable polynomial is relatively new, the authors also provided a broad overview of its security against standard attacks and provided an ad-hoc analysis of the relation between the choice of the polynomial P and the security strength of this assumption.

1.2 Our Contributions

In this work, we put forth and analyze a new general family of cryptographic assumptions related to the hardness of decoding codes defined over group algebras. A problem called *quasi-abelian syndrome decoding* (QA-SD). Our family of assumptions builds upon quasi-abelian codes, a well-known family of codes in algebraic coding theory. It generalizes both the ring-LPN assumption from [BCG+20b] under some conditions on the underlying choice of polynomial and the quasi-cyclic syndrome decoding assumption. The latter assumption was in particular used in several recent works [ABB+17, AMBD+18, MAB+18, BCG+19a], including prominent submissions to the NIST post-quantum competition. We show that working over group algebras presents several advantages:

1. a broad family of possible instantiations;
2. a rich structure that allows stronger security foundations;
3. a group algebra contains a canonical basis given by the group itself, providing a canonical notion of sparsity.

Building on our new family of assumptions, we overcome both downsides of the recent work of [BCG+20b] and obtain PCG's for OLE's over *general fields* with *solid security foundations*. In more details:

A Template for Building New PCG's. We revisit and generalize the approach of [BCG+20b] for building pseudorandom correlation generators for OLE from ring-LPN. We show that any choice of quasi-abelian code yields a PCG for OLE over a group algebra $\mathcal{R}$ under the corresponding QA-SD assumption. We identify natural instances of our framework such that the group algebra $\mathcal{R}$:

1. supports fast operations via generalizations of the Fast Fourier Transform (which allows to achieve efficiency comparable to that of [BCG+20b]), and
2. is isomorphic to a product $\mathbb{F}_q \times \cdots \times \mathbb{F}_q$ of N copies of $\mathbb{F}_q$ for arbitrary small $q > 2$ and arbitrary large N and therefore yields an efficient PCG for generating N copies of OLE over $\mathbb{F}_q$ for any $q > 2$.

Therefore, we obtain new constructions of efficient programmable PCG over small fields, circumventing the main limitation of the work of [BCG+20b]. Our PCG's enable for the first time secure computation of arithmetic circuits over fields $\mathbb{F}$ of any size $|\mathbb{F}| > 2$ in the silent preprocessing model. This holds for two or more parties, in the semi-honest or in the malicious setting. The concrete efficiency of our construction is comparable to that of [BCG+20b] (we refer the reader to Table 1 for details on the seed size and stretch of our PCG's). Concretely, our costs are essentially identical, up to the fact that [BCG+20b] uses FFT's over cyclotomic rings, while our generalization to arbitrary field relies on a generic FFT. Because FFT's over cyclotomic rings have been thoroughly optimized in hundreds of papers, we expect that using generic FFT's will be noticeably slower. Still, we identify some concrete FFT-friendly choices of quasi-abelian codes where fast FFT algorithms comparable to cyclotomic FFT's could in principle be designed. We leave the concrete optimization of these FFT algorithms to future work.

Strong Security Foundations. Building upon recent results on the minimum distance of quasi-abelian codes, we give evidence that the assumptions from our family cannot be broken by any attack from the *linear test framework* [BCG+20a, CRR21], a broad framework that encompasses essentially all known attacks on LPN and syndrome decoding (including ISD, Gaussian elimination, BKW, and many more). Our approach also sheds light on the security of the ring-LPN assumption. In essence, a conceptual message from our new approach is that some choices of P in the ring $\mathbb{F}_q[X]/(P(X))$ yield an instance of QA-SD, and as such inherit our arguments of resistance against linear attacks. In contrast, other (seemingly very similar) choices of P yield instances that are *completely broken* by linear attacks. This suggests that choosing instantiations of the ring-LPN assumption should be done with care, and our framework yields a way to do it with strong security guarantees.

As a contribution of independent interest, we also complement our security analysis by showing, for all concrete instantiations of our framework that we use in our new PCG constructions, a search-to-decision reduction for the underlying assumption. Therefore, we reduce the security of all our new PCG's to (instances of) the *search* QA-SD assumption.

The Case of $\mathbb{F}_2$. Perhaps intriguingly, the most natural way to instantiate our framework goes all the way to $\mathbb{F}_3$, but breaks down over $\mathbb{F}_2$. We prove a theorem that states that this is in fact inherent to the approach. Basically, the reason why the construction is not adaptable to $\mathbb{F}_2$ is due to the fact that the product ring $\mathbb{F}_2^N = \mathbb{F}_2 \times \cdots \times \mathbb{F}_2$ has only one invertible element and hence can never be realised as a group algebra but in the irrelevant case of $N = 1$. We then discuss a general methodology toward circumventing this limitation over $\mathbb{F}_2$. While our approach falls short of providing a full-fledged solution, it highlights a possible avenue towards the intriguing goal of one day getting an efficient programmable PCG for OLE's over $\mathbb{F}_2$.

Applications. Building upon our new programmable PCG's, we obtain

– (via Beaver triples) secure N-party computation of arithmetic circuits over $\mathbb{F}_q$, for any $q > 2$, with silent preprocessing and communication $N^2 \cdot \mathsf{poly}(\lambda) \cdot \log s$ bits (preprocessing phase) plus $2Ns$ field elements (online phase), where s is the number of multiplication gates. The silent preprocessing phase involves $O(N\mathsf{poly}(\lambda)s \log s)$ work per party. For small numbers of parties, the $N^2 \cdot \mathsf{poly}(\lambda) \cdot \log s$ is dominated by the $2Ns$ field elements for values of s as low as 2^{25}.

– (via circuit-dependent correlated randomness) secure N-party computation of a batch of T arithmetic circuits over $\mathbb{F}_q$, for any $q > 2$, with silent preprocessing and communication $N^2 \cdot \mathsf{poly}(\lambda) \cdot s \log T$ bits (preprocessing phase) plus NTs field elements (online phase) , where s is the number of multiplication gates in each circuit. The silent preprocessing phase involves $O(N\mathsf{poly}(\lambda)sT \log T)$ work per party.

As in [BCG+20b], our protocols extend to the malicious setting by generating *authenticated* correlated randomness instead, which our PCG's allow as well, and using a maliciously secure seed distribution protocol. Since the extension to authenticated correlated randomness and the seed distribution protocols in [BCG+20b] are oblivious to the concrete choice of underlying ring $\mathcal{R}$, they directly apply to our new PCG's from QA-SD.

1.3 Related Works

Traditional constructions of OLE protocols require communication for each OLE produced. The work of [Gil99] requires $\Omega(\log |\mathbb{F}|)$ string-OT's per OLE[6]. OLE's can also be produced using state-of-the-art protocols based on homomorphic encryption [KPR18, HIMV19], *e.g.* producing 64MB worth of OLE's requires about 2GB of communication with Overdrive [KPR18]. A recent direct construction of OLE from Ring-LWE has also been described in [BEPU+20]. Using their construction, generating a batch of OLE's has an amortized communication of about 8 elements of $\mathbb{F}$ over a large enough field.

PCG's for OLE's allow removing most of the communication overhead, by generating a large number of pseudorandom OLE's using sublinear communication. The work of [BCG+20b], which is our starting point, has a computational cost comparable to that of recent OLE protocols [KPR18], but a considerably lower communication ; however, it only works over large fields. There has been several attempts to build PCG's for OLE's over small fields, but all suffer from severe downsides. The work of [BCG+19b] describes a PCG construction that combines BGV-based somewhat homomorphic encryption (under ring-LWE) and a new, ad-hoc variant of the multivariate quadratic assumption with sparse secrets. Their PCG's require very large seed sizes and are only efficient when generating huge batches ([BCG+19b] estimates about 7.000 OLE's per second using a 3GB seed size when producing 17GB worth of triples).

In an appendix, the work of [BCG+20b] shows that the standard variant of syndrome decoding with quasi-cyclic code yields a PCG for OLE's over arbitrary fields (including small fields). At a high level, the construction uses the fact that given two pseudorandom vectors $\mathbf{x}^\intercal = \mathbf{H} \cdot \mathbf{e}_x^\intercal$ and $\mathbf{y} = \mathbf{H} \cdot \mathbf{e}_y^\intercal$, generating shares of their pointwise products (*i.e.* a batch of pseudorandom OLE correlations) reduces to generating shares of the diagonal of $\mathbf{x}^\intercal \cdot \mathbf{y} = \mathbf{H} \cdot (\mathbf{e}_x^\intercal \cdot \mathbf{e}_y) \cdot \mathbf{H}^\intercal$, and the term $(\mathbf{e}_x^\intercal \cdot \mathbf{e}_y)$ can be shared efficiently with FSS for point functions. However, the computational cost of generating n OLE's this way scales as $\Omega(n^2 \log n)$ (ignoring $\mathsf{poly}(\lambda)$ factors), which makes it entirely impractical in practice (the sublinearity in these protocols only "kicks in" for values of n above about 2^{30}).

Eventually, two recent works on PCG's [BCG+20a, BCG+22] have introduced new variants of syndrome decoding called respectively *variable-density* and *expand-accumulate* LPN. Each of these variants can actually be used to construct programmable PCG's for OLE over small fields (though that was not their primary purpose: VDLPN was introduced to construct pseudorandom correlation *functions*, and EALPN to obtain more efficient "online-offline" PCG's for OT). The intuition is that both assumptions can be formulated as the hardness of distinguishing $\mathbf{H} \cdot \mathbf{e}^\intercal$ from random, where $\mathbf{H}$ is a *sparse* matrix, and the noise distribution is such that the term $(\mathbf{e}_x^\intercal \cdot \mathbf{e}_y)$ can still be shared efficiently using some appropriate FSS. In this case, extracting the diagonal of $\mathbf{H} \cdot (\mathbf{e}_x^\intercal \cdot \mathbf{e}_y) \cdot \mathbf{H}^\intercal$ does not require computing the full square matrix, and scales only as $\mathsf{poly}(\lambda) \cdot \tilde{\Omega}(n)$. However, the hidden costs remain prohibitively large. Concretely, for both the EALPN assumption and the VDLPN assumption, the row-weight of $\mathbf{H}$ must grow as $\lambda \cdot \log n$ [BCG+20a, BCG+22, CD23] (for some specific security parameter λ), hence the cost of generating n OLE's boils down to $\lambda^2 \cdot \log^2 n$ invocations of an FSS scheme, where the concrete security parameter λ must be quite large: the recent analysis of [CD23] estimates $\lambda \approx 350$. For $n = 2^{30}$, this translates to around 10^8 invocation of an FSS scheme for *each* OLE produced, which is nowhere near practical.

[6] This approach crucially requires structured OT's, hence we cannot remove the communication by using pseudorandom OT's.

1.4 Organization

We provide a technical overview of our results in Section 2, and preliminaries in Section 3. Section 4 is devoted to introducing group algebras, quasi-abelian codes, and our new QA-SD family of assumptions. Section 5 uses our new QA-SD assumption to build programmable PCG's, adapting and generalising the template of [BCG+20b]. Section 6 covers the concrete security analysis of QA-SD against various known attacks, and in particular against *folding attacks*, which exploit the structure of the assumption to reduce the dimension of the instances. Finally, in Section 7 we elaborate on the applications of our new PCG's to secure computation. Appendix A provides more detailed preliminaries on FSS and PCG's. Appendix B complements our study of QA-SD by providing a search-to-decision reduction for the subset of the QA-SD family used to construct our PCG's. Appendix C provides some background on function field theory, which is used in the analysis of some of our results. Appendix C.3 adds background on the Carlitz module, which is at the heart of our (ultimately unsuccessful) attempt to extend our framework to OLE's over $\mathbb{F}_2$. Appendix D covers our approach for building OLE's over $\mathbb{F}_2$ and identifies the missing ingredient.

2 Technical Overview

2.1 Generating Pseudorandom Correlations: a Template

A general template to construct PCG's was put forth in [BCGI18], and further refined in subsequent works. At a high level, the template combines two ingredients: a method that uses *function secret sharing* to generate a *sparse* version of the target correlation, and a carefully chosen linear code for which the syndrome decoding problem is conjectured to be intractable. To give a concrete example let us consider the task of generating an OLE correlation over a large polynomial ring $\mathcal{R} = \mathbb{F}_p[X]/(P)$, where P is some degree-N split polynomial, and $\mathbb{F}_p$ is a field. In a ring-OLE correlation, each party P_σ receives $(x_\sigma, y_\sigma) \in \mathcal{R}^2$ for $\sigma = 0, 1$, which are random conditioned on $x_0 + x_1 = y_0 \cdot y_1$.

Sparse correlations from FSS. Informally, FSS for a function class $\mathcal{F}$ allows to share functions $f : \{0,1\}^\ell \mapsto \mathbb{G}$ (where $\mathbb{G}$ is some group) from $\mathcal{F}$ into $(f_0, f_1) \leftarrow \mathsf{Share}(f)$ such that

(1) f_σ hides f (computationally), and
(2) for any $x \in \{0,1\}^\ell$, $f_0(x) + f_1(x) = f(x)$.

Since FSS can always be achieved trivially by sharing the truth table of f, one typically wants the shares to be compact (*i.e.* not much larger than the description of f). Efficient constructions of FSS from a length-doubling pseudorandom generator are known for some simple function classes, such as *point functions* (functions $f_{\alpha,\beta}$ that evaluate to β on $x = \alpha$, and to 0 otherwise). FSS for point functions can be seen as a succinct way to privately share a long unit vector. More generally, FSS for t-point functions yield a succinct protocol for privately sharing a long t-sparse vector.

An FSS for multipoint functions immediately gives a strategy to succinctly distribute a *sparse* ring-OLE correlation: sample two random t-sparse polynomials y_0, y_1 (*i.e.* polynomials with t nonzero coefficients in the standard basis), and define f to be the t^2-point function whose truth table are the coefficients of $y_0 \cdot y_1$ (over $\mathbb{F}_p[X]$). Each party P_σ receives $\mathsf{k}_\sigma = (y_\sigma, f_\sigma)$, where $(f_0, f_1) = \mathsf{Share}(f)$. With standard constructions of multipoint FSS, the size of k_σ is $O(t^2 \cdot \log N)$ (ignoring λ and $\log p$ terms): whenever t is small, this is an exponential improvement over directly sharing $y_0 \cdot y_1$ (which would yield keys of length $O(N)$).

From sparse to pseudorandom using syndrome decoding. It remains to convert the sparse correlation into a pseudorandom correlation. This step is done non-interactively, by locally *compressing* the sparse correlation using a suitable linear mapping. Viewing the compressed vector as the syndrome of a linear code $\mathcal{C}$ (the compressive linear mapping is the parity-check matrix of $\mathcal{C}$). The mapping must satisfy two constraints: it should be *efficient* (linear or quasi-linear in its input size), and its output on a sparse vector should be *pseudorandom*. Fortunately, decades of research on coding theory have provided us with many linear mappings which are conjectured to satisfy the latter;

the corresponding assumptions are usually referred to as (variants of) the *syndrome decoding* (SD) assumption, or as (variants of) the *learning parity with noise* (LPN) assumption[7].

Going back to our example, we will use two instances $(x^0_\sigma, y^0_\sigma)_{\sigma\in\{0,1\}}$ and $(x^1_\sigma, y^1_\sigma)_{\sigma\in\{0,1\}}$ of a sparse ring-OLE correlation. Fix a random element $a \xleftarrow{\$} \mathcal{R}$. Each party P_σ defines

$$y_\sigma \leftarrow (1, \mathbf{a}) \cdot (y^0_\sigma, y^1_\sigma)^\intercal = y^0_\sigma + a \cdot y^1_\sigma \bmod P(X).$$

The assumption that y_σ is indistinguishable from random is known in the literature as the *ring-LPN assumption*, and has been studied in several previous works [BCG+20b] (for an appropriate choice of P, it is also equivalent to the quasi-cyclic syndrome decoding assumption, used in NIST submissions such as BIKE [ABB+17] and HQC [MAB+18]). Furthermore, using FFT, the mapping can be computed in time $\tilde{O}(N)$. Then, observe that we have

$$y_0 y_1 = (y^0_0 + a \cdot y^1_0) \cdot (y^0_1 + a \cdot y^1_1) = y^0_0 \cdot y^0_1 + a \cdot (y^0_1 \cdot y^1_0 + y^1_0 \cdot y^1_1) + a^2 \cdot (y^1_0 \cdot y^1_1),$$

where the polynomials $y^0_0 \cdot y^0_1, y^0_1 \cdot y^1_0, y^1_0 \cdot y^1_1$, and $y^1_0 \cdot y^1_1$ are all t^2-sparse. Hence, each of these four polynomials can be succinctly shared using FSS for a t^2-point function. Therefore, shares of $y_0 y_1$ can be reconstructed using a local linear combination of shares of sparse polynomials, which can be distributed succinctly using FSS for multipoint functions.

Wrapping up. The final PCG looks as follows: each party P_σ gets (y^0_σ, y^1_σ) together with four FSS shares of t^2-point functions whose domain correspond to these four terms. The PCG key size scales as $O(t^2 \log N)$ overall. Expanding the keys amounts to locally computing the shares of the sparse polynomial products (four evaluations of the FSS on their entire domain, in time $O(N)$) and a few $\tilde{O}(N)$-time polynomial multiplications with a and a^2 (which are public parameters). Observe that when P splits into N linear factors over $\mathbb{F}_p[X]$, a single pseudorandom ring-OLE correlation as above can be locally transformed into N instances of pseudorandom OLE's over $\mathbb{F}_p$: this is essentially the construction of PCG for OLE of [BCG+20b]. However, this requires p to be larger than N, restricting the construction to generating OLE's over large fields. Furthermore, the requirement of a splitting P makes the construction rely on a less-studied variant of ring-LPN.

2.2 Quasi-Abelian Codes to the Rescue

We start by abstracting out the requirement of the construction of [BCG+20b]. In coding theoretic terms, the hardness of distinguishing $(a, a \cdot e + f)$ with sparse (e, f) is an instance of the (decisional) *syndrome decoding problem* with respect to a code with parity check matrix $(1, a)$. At a high level, and sticking to the coding-theoretic terminology, we need a ring $\mathcal{R}$ such that

1. the (decisional) syndrome decoding problem with respect to the matrix $(1, a)$ is intractable with high probability over the random choice of $a \xleftarrow{\$} \mathcal{R}$;
2. given *sparse* elements (e, f) of $\mathcal{R}$, it is possible to succinctly share the element $e \cdot f \in \mathcal{R}$;
3. operations on $\mathcal{R}$, such as products, can be computed efficiently (*i.e.* in time quasilinear in the description length of elements of $\mathcal{R}$);
4. eventually, $\mathcal{R}$ is isomorphic to $\mathbb{F} \times \cdots \times \mathbb{F}$ for some target field $\mathbb{F}$ of interest.

We identify *quasi-abelian codes* as a family of codes that simultaneously satisfy all the above criteria. At a high level, a quasi-abelian code of index ℓ has codewords of the form

$$\{(\mathbf{m}\boldsymbol{\Gamma}_1, \ldots, \mathbf{m}\boldsymbol{\Gamma}_\ell) \mid \mathbf{m} = (m_1, \ldots, m_\ell) \in (\mathbb{F}_q[G])^k\},$$

[7] The name LPN historically refers to the hardness of distinguishing oracle access to samples $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s}\rangle + e)$ (for a fixed secret $\mathbf{s}$) from samples $(\mathbf{a}, b)$ where $\mathbf{a}, \mathbf{s}$ are random vectors, e is a biased random bit, and b is a uniform random bit. This becomes equivalent to the syndrome decoding assumption when the number of calls to the oracle is *a priori bounded*, hence the slight abuse of terminology. Since we will mostly use tools and results from coding theory in this work, we will use the standard coding theoretic terminology "syndrome decoding" to refer to the variant with bounded oracle access, which is the one used in all works on PCG's.

where each $\mathbf{\Gamma}_i$ is an element of $\mathbb{F}_q[G]^k$. Here, $\mathbb{F}_q[G]$ denotes the *group algebra*:

$$\mathbb{F}_q[G] \stackrel{\text{def}}{=} \left\{ \sum_{g \in G} a_g g \mid a_g \in \mathbb{F}_q \right\},$$

where G is a finite abelian group. Quasi-abelian codes generalise quasi-cyclic codes in a natural way: a quasi-cyclic code is obtained by instantiating G with $\mathbb{Z}/n\mathbb{Z}$. We define the *quasi-abelian syndrome decoding problem* (QA-SD) as the natural generalisation of the syndrome decoding problem to quasi-abelian codes. This encompasses both quasi-cyclic syndrome decoding and plain syndrome decoding. The properties of quasi-abelian codes have been thoroughly studied in algebraic coding theory. We elaborate below on why quasi-abelian codes turn out to be precisely the right choice given our constraints 1–4 above.

Security Against Linear Tests. The linear test framework from [BCG+20a, CRR21] provides a unified way to study the resistance of LPN-style and syndrome decoding-style assumptions against a wide family of *linear* attacks, which includes most known attacks on LPN and syndrome decoding. We refer the reader to Section 3.2 for a detailed coverage. At a high level, in our setting, security against linear attacks boils down to proving that $(1, a)$ generates a code with large minimum distance. On one hand, a recent result of Fan and Lin [FL15] proves that quasi-Abelian codes asymptotically meet the Gilbert-Varshamov bound when the code length goes to infinity and the underlying group is fixed. On the other hand, Gaborit and Zémor [GZ06] prove a similar result when the size of the group goes to infinity but restricted to the case where the group is cyclic. We conjecture an extension of Gaborit and Zémor result to arbitrary abelian groups. The latter conjecture entails that the QA-SD problem cannot be broken by any attack from the linear test framework, for any choice of the underlying group G. This is the key to circumvent the restrictions of [BCG+20b].

Distribution of Products of Sparse Elements. Using quasi-abelian codes, the ring $\mathcal{R}$ is therefore a group algebra $\mathbb{F}_q[G]$. Now, given $e = \sum_{g \in G} e_g g$ and $f = \sum_{g \in G} f_g g$ any two t-sparse elements of $\mathcal{R}$ (that is, such that $(e_g)_{g \in G}$ and $(f_g)_{g \in G}$ have Hamming weight t), the product $e \cdot f$ can be rewritten as

$$e \cdot f = \sum_{e_g, f_h \neq 0} e_g f_h \cdot gh,$$

which is a t^2-sparse element of the group algebra. In other words, the product of two sparse elements in a group algebra is always a sparse element. In the context of building PCG's, this implies that we can directly distribute elements $ef \in \mathcal{R}$ using Sum of Point Function Secret Sharing (SPFSS) for t^2-point functions. This allows us to generalise the template PCG construction of [BCG+20b] to the setting of arbitrary quasi-abelian code, with essentially the same efficiency (in a sense, the template is "black-box" in the ring: it only relies on the ability to distribute sparse elements via FSS).

We note that our generalised template differs slightly from the approach of [BCG+20b]: in this work, the authors work over rings of the form $\mathcal{R} = \mathbb{F}_p[X]/(P(X))$, where P is some polynomial. However, in general, this ring is not a group algebra, and the product of sparse elements of $\mathcal{R}$ might not be sparse. They circumvented this issue by sharing directly the product over $\mathbb{F}_p[X]$ (where the product of sparse polynomials remains sparse) and letting the parties reduce locally modulo P. Doing so, however, introduces a factor 2 overhead in the expansion (and a slight overhead in the seed size). Our approach provides a cleaner solution, using a structure where sparsity is natively preserved through products inside the ring.

Fast Operations on Group Algebras. We observe that, by folklore results, operations over a group algebra $\mathbb{F}_q[G]$ admit an FFT algorithm (using a general form of the FFT which encompasses both the original FFT of Cooley and Tuckey, and the Number Theoretic Transform). When using this general FFT, setting $G = \mathbb{Z}/2^t\mathbb{Z}$ recovers the usual FFT from the literature. In full generality, given any abelian group G of cardinality n with $\gcd(n, q) = 1$ and exponent d, if $\mathbb{F}_q$ contains a

primitive d-th root of unity, then the Discrete Fourier Transform and its inverse can be computed in time $O(n \cdot \sum_i p_i)$, where the p_i are the prime factors appearing in the Jordan-Hölder series of G; we refer the reader to Section 4.3 for a more detailed coverage. For several groups of interest in our context, this appears to yield very efficient FFT variants. For example, setting $q = 3$ and $G = (\mathbb{Z}/2\mathbb{Z})^d$, the resulting FFT is a d-dimensional FFT over $\mathbb{F}_3$ and it can be computed in time $O(n \cdot \log n)$ (the group algebra $\mathbb{F}_3[(\mathbb{Z}/2\mathbb{Z})^d]$ is the one that yields a PCG for n copies of OLE over $\mathbb{F}_3$).

We note that FFT's over cyclotomic rings, such as those used in [BCG+20b], have been heavily optimised in hundreds of papers, due to their wide use (among other things) in prominent cryptosystems. As such, it is likely that even over "FFT-friendly" choices of group algebras, such as $\mathbb{F}_3[(\mathbb{Z}/2\mathbb{Z})^d]$, the general FFT construction described above will be in practice significantly less efficient than the best known FFT's implementations over cyclotomic rings. Hence, computationally, we expect that state-of-the-art implementations of the PCG of [BCG+20b] over large fields $\mathbb{F}$ using a cyclotomic ring $\mathcal{R}$ for the ring-LPN assumption will be noticeably faster than state-of-the-art implementations of our approach to generate OLE's over a small field, such as $\mathbb{F}_3$. There is however nothing inherent to this: the efficiency gap stems solely from the years of effort that have been devoted to optimising FFT's over cyclotomic rings, but we expect that FFT's over other FFT-friendly group algebra such as $\mathbb{F}_3[(\mathbb{Z}/2\mathbb{Z})^d]$ could be significantly optimised in future works. We hope that our applications to silent secure computation over general fields will motivate such studies in the future.

From Quasi-Abelian Codes to OLE's over $\mathbb{F}_q$. Our general PCG template allows to generate a pseudorandom OLE over an arbitrary group algebra $\mathbb{F}_q[G]$. Then, when using $G = (\mathbb{Z}/(q-1)\mathbb{Z})^d$, we have that $\mathbb{F}_q[G] \simeq \mathbb{F}_q^n$ (with $n = (q-1)^d$). Therefore, a single pseudorandom OLE over $\mathbb{F}_q[G]$ can be *locally* converted by the parties into $(q-1)^d$ copies of a pseudorandom OLE over $\mathbb{F}_q$. Furthermore, for these concrete choices of G, we complement our security analysis by proving a search-to-decision reduction, showing that the decision QA-SD problem over $\mathbb{F}_q[G]$ with $G = (\mathbb{Z}/(q-1)\mathbb{Z})^d$ is as hard as the *search* QA-SD problem. This provides further support for the security of our instantiations.

In addition, our framework provides a way to investigate different instantiations of the ring-LPN problem through the lens of quasi-abelian codes. This turns out to play an important role in understanding the basis for the security of ring-LPN: seemingly very similar choices of the underlying polynomial can yield secure instances in one case, and completely broken instances in the other case. While the work of [BCG+20b] gave a heuristic cryptanalysis of ring-LPN, it fails to identify the influence of the choice of the polynomial.

Concretely, consider the ring $\mathcal{R} = \mathbb{F}_q[X]/(P(X))$ with either $P(X) = X^{q-1} - 1$ or $P(X) = X^q - X$. The latter is a natural choice, as it has the largest possible number of factors over $\mathbb{F}_q$ (which controls the number of OLE's produced over $\mathbb{F}_q$). $\mathcal{R} = \mathbb{F}_q[X]/(P(X))$ with $P(X) = X^{q-1} - 1$ is a group algebra, and the ring-LPN assumption with ring $\mathcal{R}$ reduces to QA-SD($\mathcal{R}$). Hence, it is secure against all attacks from the linear test framework (and admits a search-to-decision reduction) by our analysis. On the other hand, ring-LPN over the ring $\mathcal{R} = \mathbb{F}_q[X]/(P(X))$ with $P(X) = X^q - X$ does not fit in our framework, and turns out to be *completely broken* by a simple linear attack: given (a, b) where b is either random or equal to $a \cdot e + f \bmod X^q - X$, it holds that $e(0) = f(0) = 0 \bmod X^q - X$ with high probability, because $e(0) = f(0) = 0$ over $\mathbb{F}_q[X]$ with high probability (since e, f are sparse, their constant coefficient is likely to be zero), and reduction modulo $X^q - X$ does not change the constant coefficient. Hence, the adversary can distinguish b from random simply by computing $b(0)$ (since $b(0)$ is nonzero with probability $(q-1)/q$ for a random b).

The above suggests that settling for $\mathcal{R} = \mathbb{F}_q[X]/(X^{q-1} - 1)$ is a conservative choice to instantiate the PCG of [BCG+20b] with strong security guarantees. We note that [BCG+20b] recommended instead $\mathcal{R} = \mathbb{F}_p[X]/(X^n + 1)$ with n being a power of 2 and p a large prime for efficiency reasons (since it is a cyclotomic ring, it admits fast FFT's). We believe that a natural generalisation of our framework should also encompass this ring, and allow proving that it also yields a flavor of ring-LPN which is immune to linear attacks. However, this is beyond the scope of our paper, and we leave it to future work.

Considerations on the Case of $\mathbb{F}_2$. Interestingly, the aforementioned instance allows generating many OLE's over $\mathbb{F}_q$ for any $q > 2$; for $q = 2$, however, the term $n = (q-1)^d$ becomes equal to 1;

that is, we only get a single OLE over $\mathbb{F}_2$ this way. This is in fact inherent to our approach: the product ring $\mathbb{F}_2^n$ has only one invertible element, and therefore can never be realised as a group algebra unless $n = 1$. Hence, somewhat surprisingly, our general approach circumvents the size limitation of [BCG+20b] and gets us all the way to $\mathbb{F}_3$ or any larger field, but fails to provide a construction in the (particularly interesting) case $\mathbb{F}_2$.

Motivated by this limitation of our framework, we devise a strategy to further generalise our approach through the theory of algebraic function fields (in essence, our generalisation is to quasi-abelian codes what quasi-negacyclic codes are to quasi-cyclic codes; we note that this is also close in spirit to the instance chosen in [BCG+20b]: for their main candidate, they suggest using the ring $\mathcal{R} = \mathbb{F}_p[X]/(X^n + 1)$, which is a module over a group algebra and yields a *quasi-negacyclic code*). Alas, we did not manage to get a fully working candidate. At a (very) high level, our generalised framework produces pseudorandom elements $x = a \odot e_x + 1 \odot f_x$ and $y = a \odot e_y + 1 \odot f_y$ where e_x, e_y, f_x, f_y are sparse. However, the product $\odot$ is now *not* the same product as the group algebra product $x \cdot y$. Concretely, to share $x \cdot y$, we need to share terms of the form $(u \odot e) \cdot (v \odot f)$ (where u, v can be a or 1). However, unlike the case of our previous instantiation, this does not rewrite as a term of the form $uv \cdot ef$ (which we could then share by sharing the sparse term ef, as uv is public). Still, we believe that our approach could serve as a baseline for future works attempting to tackle the intriguing problem of building efficient programmable PCG's for OLE over $\mathbb{F}_2$. In particular, our unsuccessful attempts show that to get such a PCG, it suffices to find a way to succinctly share terms of the form $(u \odot e) \cdot (v \odot f)$ where u, v are public, and e, f are sparse. While FSS do not provide an immediate solution to this problem, this reduces the goal to a "pure MPC problem" which could admit an efficient solution.

Concrete Cryptanalysis. Eventually, we complement our study by a concrete analysis of the security of our assumptions. As in previous works, the bounds derived from the resistance to linear attacks are quite loose, because they cover a *worst-case* choice of linear attack. We cover standard attacks, such as information set decoding. A particularity of both ring-LPN with splittable polynomial and our new family of QA-SD assumption is that it grants the adversary some additional freedom: the adversary can, informally, transform a QA-SD instance into an instance with reduced dimension (in the case of ring-LPN, by reducing modulo factors of P; for QA-SD, by quotienting by subgroups of G). This turns out to be equivalent to the concept of *folding attacks*, which have been recently studied both in the context of code-based cryptography [CT19] and of lattice-based cryptography [BCV20]. We analyse the effect of folding attacks on our instances and discuss the impact on our parameter choices. In particular, the instances of QA-SD used in our PCG construction closely resemble the Multivariate LWE assumption (with sparse noise instead of small-magnitude noise), which was shown in [BCV20] to be broken by folding attacks. We note (but this is well-known [CT19]) that folding attacks are much less devastating on LPN- and syndrome decoding-style assumptions, essentially because folding yields a very slight increase of the noise magnitude in the LWE setting (the sum of LWE error terms has small magnitude), but increases the noise rate very quickly in the coding setting (the sum of sparse noises very quickly becomes dense).

3 Preliminaries

Function Secret Sharing. Function secret sharing (FSS), introduced in [BGI15, BGI16], allows to succinctly share functions. An FSS scheme splits a secret function $f : I \to \mathbb{G}$, where $\mathbb{G}$ is some Abelian group, into two functions f_0, f_1, each represented by a key K_0, K_1, such that: (1) $f_0(x) + f_1(x) = f(x)$ for every input $x \in I$, and (2) each of K_0, K_1 individually hides f.

An SPFSS is an FSS scheme for the class of *sums of point functions*: functions of the form $f(x) = \sum_i f_{s_i,y_i}(x)$ where each $f_{s_i,y_i}(\cdot)$ evaluates to y_i on s_i, and to 0 everywhere else. As in previous works, we will use efficient constructions of SPFSS in our constructions of PCGs. Such efficient constructions are known from any length-doubling pseudorandom generator [BGI16]. We refer the reader to Appendix A for more details on FSS and SPFSS.

Pseudorandom Correlation Generators. A pseudorandom correlation generator (PCG) for some target ideal correlation takes as input a pair of short, correlated seeds and outputs long cor-

related pseudorandom strings, where the expansion procedure is deterministic and can be applied locally. In slightly more details, a PCG is a pair $(\mathsf{Gen}, \mathsf{Expand})$ such that $\mathsf{Gen}(1^\lambda)$ produces a pair of short seeds $(\mathsf{k}_0, \mathsf{k}_1)$ and $\mathsf{Expand}(\sigma, \mathsf{k}_\sigma)$ outputs a string R_σ. A PCG is *correct* if the distribution of the pairs (R_0, R_1) output by $\mathsf{Expand}(\sigma, \mathsf{k}_\sigma)$ for $\sigma = 0, 1$ is indistinguishable from a random sample of the target correlation. It is *secure* if the distribution of $(\mathsf{k}_{1-\sigma}, R_\sigma)$ is indistinguishable from the distribution obtained by first computing $R_{1-\sigma}$ from $\mathsf{k}_{1-\sigma}$, and sampling a uniformly random R_σ conditioned on satisfying the target correlation with $R_{1-\sigma}$ (for both $\sigma = 0$ and $\sigma = 1$). In this work, we will mostly consider the OLE correlation, where the parties P_0, P_1 receive random vectors $\mathbf{x}_0, \mathbf{x}_1 \in \mathbb{F}^n$ respectively, together with random shares of $\mathbf{x}_0 * \mathbf{x}_1$, where $*$ denotes the component-wise (*i.e.* Schur) product.

Eventually, *programmable* PCG's allow generating multiple PCG keys such that part of the correlation generated remains the same across different instances. Programmable PCG's are necessary to construct n-party correlated randomness from the 2-party correlated randomness generated via the PCG. Informally, this is because when expanding n-party shares (e.g. of Beaver triples) into a sum of 2-party shares, the sum will involve many "cross terms"; using programmable PCG's allows maintaining consistent pseudorandom values across these cross terms. We refer the reader to Appendix A for more details on PCG's and programmable PCG's.

3.1 Syndrome Decoding Assumptions

The syndrome decoding assumption over a field $\mathbb{F}$ states, informally, that no adversary can distinguish $(\mathbf{H}, \mathbf{H} \cdot \mathbf{e}^\intercal)$ from $(\mathbf{H}, \mathbf{b}^\intercal)$, where $\mathbf{H}$ is sampled from the set of parity-check matrices of some family of linear codes, and $\mathbf{e}$ is a *noise vector* sampled from some distribution over $\mathbb{F}$-vectors and typically sparse. The vector $\mathbf{b}$ is a uniform vector over $\mathbb{F}^n$. More formally, we define the SD assumption over a ring $\mathcal{R}$ with dimension k, code length n, w.r.t. a family $\mathcal{F}$ of linear codes, and a noise distribution $\mathcal{D}$:

Definition 1 (Syndrome Decoding). *Let $k, n \in \mathbb{N}$, and let $\mathcal{F} = \mathcal{F}_{n,k} \subset \mathcal{R}^{(n-k)\times n}$ be a family of parity-check matrices of codes over some ring $\mathcal{R}$. Let $\mathcal{D}$ be a noise distribution over $\mathcal{R}^n$. The $(\mathcal{D}, \mathcal{F}, \mathcal{R})$-$\mathsf{SD}(k, n)$ assumption states that*

$$\{(\mathbf{H}, \mathbf{H} \cdot \mathbf{e}^\intercal) \mid \mathbf{H} \xleftarrow{\$} \mathcal{F}, \mathbf{e} \xleftarrow{\$} \mathcal{D}\} \overset{\mathrm{c}}{\approx} \{(\mathbf{H}, \mathbf{b}^\intercal) \mid \mathbf{H} \xleftarrow{\$} \mathcal{F}, \mathbf{b} \xleftarrow{\$} \mathcal{R}^n\},$$

where "$\overset{\mathrm{c}}{\approx}$" denotes the computational indistiguishability.

Denoting t a parameter which governs the average density of nonzero entries in a random noise vector, common choices of noise distribution are Bernoulli noise (each entry is sampled from a Bernoulli distribution with parameter t/n), exact noise (the noise vector is uniformly random over the set of vectors of Hamming weight t), and regular noise (the noise vector is a concatenation of t random unit vectors). The latter is a very natural choice in the construction of pseudorandom correlation generators as it significantly improves efficiency [BCGI18, BCG+19b, BCG+19a] without harming security (to the best of our knowledge; the recent work [BØ23] being efficient for very low code rates, which is not our setting).

Many codes are widely believed to yield secure instances of the syndrome decoding assumption, such as setting $\mathbf{H}$ to be a uniformly random matrix over $\mathbb{F}_2$ (the standard SD assumption), the parity-check matrix of an LDPC code [Ale03] (the "Alekhnovich assumption"), a quasi-cyclic code (as used in several recent submissions to the NIST post-quantum competition, see e.g. [ABB+17, AMBD+18, MAB+18] and in previous works on pseudorandom correlation generators, such as [BCG+19a]), Toeplitz matrices [GRS08, LM13] and more. All these variants generalize naturally to larger fields (and are conjectured to remain secure over arbitrary fields).

In the context of PCG's, different codes enable different applications: advanced PCG constructions, such as PCGs for OLE, require codes with structure. When designing new PCGs, it is common to rely on syndrome decoding for codes which have not been previously analyzed in the literature – hence, unlike the ones listed above, they did not withstand years or decades of cryptanalysis. To facilitate the systematic analysis of new proposals, recent works [BCG+20a, CRR21] have put forth a framework to automatically establish the security of new variants of the syndrome decoding assumption against a large class of standard attacks.

3.2 The Linear Test Framework

The linear test framework provides a unified template to analyze the security of variants of the LPN or syndrome decoding assumption against the most common attacks. It was first put forth explicitly in [BCG+20a, CRR21] (but similar observations were implicit in many older works). Concretely, an attack against syndrome decoding in the linear test framework proceeds in two stages:

1. First, a matrix $\mathbf{H}$ is sampled from $\mathcal{F}$, and fed to the (unbounded) adversary $\mathcal{A}$. The adversary returns a (nonzero) *test vector* $\mathbf{v} = \mathcal{A}(\mathbf{H})$.
2. Second, a noise vector $\mathbf{e}$ is sampled. The *advantage* of the adversary $\mathcal{A}$ in the linear test game is the bias of the induced distribution $\mathbf{v} \cdot \mathbf{H} \cdot \mathbf{e}^\intercal$.

To formalize this notion, we recall the definition of the bias of a distribution:

Definition 2 (Bias of a Distribution). *Given a distribution $\mathcal{D}$ over $\mathbb{F}^n$ and a vector $\mathbf{u} \in \mathbb{F}^n$, the bias of $\mathcal{D}$ with respect to $\mathbf{u}$, denoted $\mathsf{bias}_{\mathbf{u}}(\mathcal{D})$, is equal to*

$$\mathsf{bias}_{\mathbf{u}}(\mathcal{D}) = |\mathbb{P}_{\mathbf{x}\sim\mathcal{D}}[\mathbf{u}\cdot\mathbf{x}^\intercal = 0] - \mathbb{P}_{\mathbf{x}\sim\mathcal{U}_n}[\mathbf{u}\cdot\mathbf{x}^\intercal = 0]| = \left|\mathbb{P}_{\mathbf{x}\sim\mathcal{D}}[\mathbf{u}\cdot\mathbf{x}^\intercal = 0] - \frac{1}{|\mathbb{F}|}\right|,$$

where $\mathcal{U}_n$ denotes the uniform distribution over $\mathbb{F}^n$. The bias of $\mathcal{D}$, denoted $\mathsf{bias}(\mathcal{D})$, is the maximum bias of $\mathcal{D}$ with respect to any nonzero vector $\mathbf{u}$.

We say that an instance of the syndrome decoding problem is *secure against linear test* if, with very high probability over the sampling of $\mathbf{H}$ in step 1, for any possible adversarial choice of $\mathbf{v} = \mathcal{A}(\mathbf{H})$, the bias of $\mathbf{v} \cdot \mathbf{H} \cdot \mathbf{e}^\intercal$ induced by the random sampling of $\mathbf{e}$ is negligible. Intuitively, the linear test framework captures any attack where the adversary is restricted to computing a linear function of the syndrome $\mathbf{b}^\intercal = \mathbf{H} \cdot \mathbf{e}^\intercal$, but the choice of the linear function itself can depend arbitrarily on the code. Hence, the adversary is restricted in one dimension (it has to be linear in $\mathbf{b}^\intercal$), but can run in unbounded time given $\mathbf{H}$.

The core observation made in [BCG+20a, CRR21] (and also implicit in previous works) is that almost all known attacks against syndrome decoding (including, but not limited to, attacks based on Gaussian elimination and the BKW algorithm [BKW00, Lyu05, LF06, EKM17] and variants based on covering codes [ZJW16, BV16, BTV16, GJL20], the ISD family of information set decoding attacks [Pra62, Ste88, FS09, BLP11, MMT11, BJMM12, MO15, EKM17, BM18], statistical decoding attacks [AJ01, FKI06, Ove06, DT17], generalized birthday attacks [Wag02, Kir11], linearization attacks [BM97, Saa07], attacks based on finding low weight code vectors [Zic17], or on finding correlations with low-degree polynomials [ABG+14, BR17]) fit in the above framework. Therefore, provable resistance against linear test implies security against essentially all standard attacks.

Security Against Linear Tests. Resistance against linear test is a property of both the code distribution (this is the "with high probability over the choice of $\mathbf{H}$" part of the statement) and of the noise distribution (this is the "the bias of the distribution induced by the sampling of $\mathbf{e}$ is low" part of the statement). It turns out to be relatively easy to give sufficient conditions for resistance against linear tests. At a high level, it suffices that

1. the *code generated by* $\mathbf{H}$ has large minimum distance, and
2. for any large enough subset S of coordinates, with high probability over the choice of $\mathbf{e}$, one of the coordinates of $\mathbf{e}$ indexed by S will be nonzero.

The above characterization works for any noise distribution whose nonzero entries are uniformly random over $\mathcal{R} \setminus \{0\}$, which is the case for all standard choices of noise distributions. To see why these conditions are sufficient, recall that the adversarial advantage is the bias of $\mathbf{v} \cdot \mathbf{H} \cdot \mathbf{e}^\intercal$. By condition (2), if the subset S of nonzero entries of $\mathbf{v} \cdot \mathbf{H}$ is sufficiently large, then $\mathbf{e}$ will "hit" one of these entries with large probabilities, and the output will be uniformly random. But the condition that S is sufficiently large translates precisely to the condition that $\mathbf{v} \cdot \mathbf{H}$ has large Hamming weight for any possible (nonzero) vector $\mathbf{v}$, which is equivalent to saying that $\mathbf{H}$ generates a code with large minimum distance. We recall the formalization below:

Definition 3 (Security against Linear Tests). *Let $\mathcal{R}$ be a ring, and let $\mathcal{D}$ denote a noise distribution over $\mathcal{R}^n$. Let $\mathcal{F} \subset \mathcal{R}^{(n-k)\times k}$ be a family of (parity-check matrices of) linear codes. Let $\varepsilon, \eta : \mathbb{N} \mapsto [0,1]$ be two functions. We say that the $(\mathcal{D}, \mathcal{F}, \mathcal{R})$-$\mathsf{SD}(k,n)$ problem is* (ε, η)-secure against linear tests *if for any (possibly inefficient) adversary $\mathcal{A}$ which, on input $\mathbf{H}$ outputs a nonzero $\mathbf{v} \in \mathcal{R}^n$, it holds that*

$$\Pr[\mathbf{H} \xleftarrow{\$} \mathcal{F}, \mathbf{v} \xleftarrow{\$} \mathcal{A}(\mathbf{H}) \ : \ \mathsf{bias}_{\mathbf{v}}(\mathcal{D}_{\mathbf{H}}) \geqslant \varepsilon(\lambda)] \leqslant \eta(\lambda),$$

where λ denotes the security parameter and $\mathcal{D}_{\mathbf{H}}$ denotes the distribution which samples $\mathbf{e} \leftarrow \mathcal{D}$ and outputs $\mathbf{H} \cdot \mathbf{e}^{\intercal}$.

The *minimum distance* of a matrix $\mathbf{H}$, denoted $\mathsf{d}(\mathbf{H})$, is the minimum weight of a nonzero vector in its row-span. Then, we have the following straightforward lemma:

Lemma 4. *Let $\mathcal{D}$ denote a noise distribution over $\mathcal{R}^n$. Let $\mathcal{F} \subset \mathcal{R}^{(n-k)\times k}$ be a family of parity-check matrices of linear codes. Then for any integer $d \in \mathbb{N}$, the $(\mathcal{D}, \mathcal{F}, \mathcal{R})$-$\mathsf{SD}(k,n)$ problem is (ε_d, η_d)-secure against linear tests, where*

$$\varepsilon_d = \max_{wt(\mathbf{v})>d} \mathsf{bias}_{\mathbf{v}}(\mathcal{D}), \qquad \text{and} \qquad \eta_d = \Pr_{\mathbf{H} \xleftarrow{\$} \mathcal{F}}[\mathsf{d}(\mathbf{H}) \geqslant d].$$

The proof is folklore, and can be found e.g. in [CRR21]. For example, using either Bernoulli, exact, or regular noise distributions with expected weight t, for any $\mathbf{v}$ of weight at least d, the bias against $\mathbf{v}$ is bounded by $e^{-2td/n}$. Hence, if the code is a good code (*i.e.* $d = \Omega(n)$), the bias is of the form $2^{-\Omega(t)}$.

When security against linear attacks does not suffice. There are two important cases where security against linear test does not yield security against *all* attacks.

1. When the code is strongly algebraic. For example, Reed-Solomon codes, which have a strong algebraic structure, have high dual minimum distance, but can be decoded efficiently with the Welch–Berlekamp algorithm, hence they do not lead to a secure syndrome decoding instance (and indeed, Welch–Berlekamp does not fit in the linear test framework).
2. When the noise is structured (e.g. for regular noise) and the code length is at least quadratic in the dimension. This opens the door to algebraic attacks such as the Arora-Ge attack [AG11] or the recent attack from Briaud and Øygarden [BØ23]. However, when $n = O(k)$ (which is the case in all our instances), these attacks do not apply.

The above are, as of today, the only known cases where security against linear attacks is known to be insufficient. Algebraic decoding techniques have a long history and are only known for very restricted families of codes, and the aforementioned algebraic attacks typically never applies in the $n = O(k)$ regime which we usually consider for PCG's. Therefore, a reasonable rule of thumb is that a variant of syndrome decoding yields a plausible assumption if (1) it provably resists linear attacks, and (2) finding an algebraic decoding algorithm is a longstanding open problem.

4 Group Algebras and Quasi-Abelian Codes

4.1 Quasi-Abelian Codes

Quasi-abelian codes have been first introduced in [Was77], and, since then, have been extensively studied in coding theory.

Group Algebras. Let $\mathbb{F}_q$ denote the finite field with q elements, and let G be a finite abelian group of cardinality n. The group algebra of G with coefficients in $\mathbb{F}_q$ is the free algebra with generators G. More precisely, it is the set $\mathbb{F}_q[G]$ of formal linear combinations

$$\mathbb{F}_q[G] \stackrel{\text{def}}{=} \left\{ \sum_{g \in G} a_g g \;\middle|\; a_g \in \mathbb{F}_q \right\},$$

endowed with an $\mathbb{F}_q-$vector space structure in the natural way, and the multiplication is given by the convolution:

$$\left(\sum_{g\in G} a_g g\right)\left(\sum_{g\in G} b_g g\right) \stackrel{\text{def}}{=} \sum_{g\in G}\left(\sum_{h\in G} a_h b_{h^{-1}g}\right) g.$$

It is readily seen that $\mathbb{F}_q[G]$ is commutative if and only if the group G is abelian, which will always be the case in this article.

Once an ordering $g_0, \ldots, g_{n-1}$ of the elements of G is chosen, the group algebra $\mathbb{F}_q[G]$ is isomorphic (as an $\mathbb{F}_q$–linear space) to $\mathbb{F}_q^n$ via $\varphi\colon \sum_{i=0}^{n-1} a_i g_i \mapsto (a_0, \ldots, a_{n-1})$. This isomorphism is not canonical since it depends on the ordering, but changing it only leads to a permutation of the coordinates, and many groups (especially *abelian* groups) come with a canonical ordering. This isomorphism allows us to endow $\mathbb{F}_q[G]$ with the Hamming metric, making φ an *isometry*: The weight $\mathrm{wt}(a))$ of $a \in \mathbb{F}_q[G]$ is defined as the Hamming weight of $\varphi(a)$ (Note that changing the ordering of the group does not impact the weight of an element, which is thus well-defined).

Example 5. The simplest example to have in mind is the case of cyclic groups.

- Let $G = \{1\}$ be the trivial group with one element. Then the group algebra $\mathbb{F}_q[G]$ is isomorphic to the finite field $\mathbb{F}_q$.
- Let $G = \mathbb{Z}/n\mathbb{Z}$ be the cyclic group with n elements. Assuming that q is coprime to n, it is easy to see that the group algebra $\mathbb{F}_q[G]$ is nothing else than the usual polynomial ring $\mathbb{F}_q[X]/(X^n - 1)$. The isomorphism is given by $k \mapsto X^k$ extended by linearity.

Remark 6. The above example shows that our framework will only be a generalisation of known constructions. This generality will be crucial though, because all the instances we introduce in the present article and which will be proved to resist to linear attacks will arise from group algebras.

Example 5 shows that the group algebra of a cyclic group can be seen as a (quotient of a) polynomial ring in one variable. For a general finite abelian group, this is not always so simple, however there is also an explicit nice representation. This uses the following standard fact from the theory of group algebras.

Proposition 7. *Let G_1, G_2 be two finite groups. Then*

$$\mathbb{F}_q[G_1 \times G_2] \simeq \mathbb{F}_q[G_1] \otimes_{\mathbb{F}_q} \mathbb{F}_q[G_2].$$

Example 8. Let $G = \mathbb{Z}/n\mathbb{Z} \times \mathbb{Z}/m\mathbb{Z}$. Then, Proposition 7 entails that

$$\begin{aligned}\mathbb{F}_q[G] = \mathbb{F}_q[\mathbb{Z}/n\mathbb{Z}] \otimes_{\mathbb{F}_q} \mathbb{F}_q[\mathbb{Z}/m\mathbb{Z}] &= \mathbb{F}_q[X]/(X^n - 1) \otimes_{\mathbb{F}_q} \mathbb{F}_q[X]/(X^m - 1)\\ &= \mathbb{F}_q[X, Y]/(X^n - 1, Y^m - 1).\end{aligned}$$

This isomorphism can actually be made explicit by $(k, \ell) \mapsto X^k Y^\ell$ extended by linearity.

Remark 9. More generally, since it is well–known that any finite abelian group G is a product of cyclic group $\mathbb{Z}/d_1\mathbb{Z} \times \cdots \times \mathbb{Z}/d_r\mathbb{Z}$, the previous statement asserts that the group algebra $\mathbb{F}_q[G]$ is isomorphic to a quotient of a multivariate polynomial ring, namely:

$$\mathbb{F}_q[G] = \mathbb{F}_q[\mathbb{Z}/d_1\mathbb{Z} \times \cdots \times \mathbb{Z}/d_r\mathbb{Z}] \simeq \mathbb{F}_q[X_1, \ldots, X_r]/(X_1^{d_1} - 1, \ldots, X_r^{d_r} - 1).$$

Quasi-Abelian Codes. Let $\ell > 0$ be any positive integer, and consider the free $\mathbb{F}_q[G]-$module of rank ℓ:

$$(\mathbb{F}_q[G])^\ell \stackrel{\text{def}}{=} \mathbb{F}_q[G] \oplus \cdots \oplus \mathbb{F}_q[G] = \Big\{(a_1, \ldots, a_\ell) \mid a_i \in \mathbb{F}_q[G]\Big\}.$$

Any $\mathbb{F}_q[G]-$submodule of $(\mathbb{F}_q[G])^\ell$ is called a *quasi-group code* of index ℓ of G (or quasi-G code). When the group G is abelian, a quasi-G code is called *quasi-abelian*. More precisely, given a matrix

$$\mathbf{\Gamma} = \begin{pmatrix} \gamma_{1,1} & \cdots & \gamma_{1,\ell} \\ \vdots & \ddots & \vdots \\ \gamma_{k,1} & \cdots & \gamma_{k,\ell} \end{pmatrix} \in (\mathbb{F}_q[G])^{k\times\ell},$$

the quasi-G code defined by $\mathbf{\Gamma}$ is

$$\mathcal{C} \stackrel{\text{def}}{=} \{\mathbf{m\Gamma} = (\mathbf{m\Gamma}_1, \ldots, \mathbf{m\Gamma}_\ell) \mid \mathbf{m} = (m_1, \ldots, m_\ell) \in (\mathbb{F}_q[G])^k\},$$

where $\mathbf{\Gamma}_i$ denotes the column $\begin{pmatrix} \gamma_{1,i} \\ \vdots \\ \gamma_{k,i} \end{pmatrix}$ and $\mathbf{m\Gamma}_i = m_1\gamma_{1,i} + \cdots + m_k\gamma_{k,i} \in \mathbb{F}_q[G]$. The matrix $\mathbf{\Gamma}$ is said to be *systematic* if it is of the form $\mathbf{\Gamma} = \left(I_k \mid \mathbf{\Gamma}'\right)$, where $\mathbf{\Gamma}' \in (\mathbb{F}_q[G])^{k\times(\ell-k)}$ and $I_k \in (\mathbb{F}_q[G])^{k\times k}$ is the diagonal matrix with values 1_G.

Let $a \in \mathbb{F}_q[G]$ and choose an ordering $g_0, \ldots, g_{n-1}$ of the elements of G. Through the aforementioned isomorphism φ, the element a can be represented as a vector $(a_0, \ldots, a_{n-1}) \in \mathbb{F}_q^n$. Now, consider the matrix

$$\mathbf{A} = \begin{pmatrix} \varphi(a \cdot g_0) \\ \vdots \\ \varphi(a \cdot g_{n-1}) \end{pmatrix} \in \mathbb{F}_q^{n\times n},$$

where each row is the vector representation of a shift of a by some element $g_i \in G$. In short, the matrix $\mathbf{A}$ is the matrix representing the multiplication–by–a map $m \mapsto am$ in $\mathbb{F}_q[G]$ in the basis $(g_0, \ldots, g_{n-1})$. An easy computation shows that for $m, a \in \mathbb{F}_q[G]$, the vector representation of the product $m \cdot a$ is the vector-matrix product

$$\varphi(m)\mathbf{A} = (m_0, \ldots, m_{n-1}) \begin{pmatrix} \varphi(a \cdot g_0) \\ \vdots \\ \varphi(a \cdot g_{n-1}) \end{pmatrix}.$$

In other words, any quasi-group code $\mathcal{C}$ of index ℓ can be seen as a linear code of length $\ell \times n$ over $\mathbb{F}_q$. The $\mathbb{F}_q[G]-$module structure endows $\mathcal{C}$ with an additional action of the group G on each block of length n; and $\mathcal{C}$ (seen as a linear code over $\mathbb{F}_q$), admits a generator matrix formed out by $k \times \ell$ square blocks of size n.

Example 10. Let us continue with Example 5.

- If $G = \{1\}$, then any linear code is a quasi-G code.
- If $G = \mathbb{Z}/n\mathbb{Z}$ and q is coprime to n. An element of $\mathbb{F}_q[G] \simeq \mathbb{F}_q[X]/(X^n - 1)$ is a polynomial of degree at most n which can be represented by the vector of its coefficients, and any product $m(X) \cdot a(X) \in \mathbb{F}_q[G]$ can be represented by the *circulant* vector-matrix product

$$\begin{pmatrix} m_0 & m_1 & \ldots & m_{n-1} \end{pmatrix} \begin{pmatrix} a_0 & a_1 & \ldots & a_{n-1} \\ a_{n-1} & a_0 & \ldots & a_{n-2} \\ \vdots & & & \vdots \\ a_1 & a_{n-1} & \ldots & a_0 \end{pmatrix} \in \mathbb{F}_q^n.$$

  For simplicity, assume that $k = 1$ and $\ell = 2$. Then, a quasi-$\mathbb{Z}/n\mathbb{Z}$ code of index 2 is defined over $\mathbb{F}_q$ by a double-circulant generator matrix

$$\left(\begin{array}{cccc|cccc} a_0 & a_1 & \ldots & a_{n-1} & b_0 & b_1 & \ldots & b_{n-1} \\ a_{n-1} & a_0 & \ldots & a_{n-2} & b_{n-1} & b_0 & \ldots & b_{n-2} \\ \vdots & & & \vdots & \vdots & & & \vdots \\ a_1 & a_{n-1} & \ldots & a_0 & b_1 & b_{n-1} & \ldots & b_0 \end{array}\right).$$

  In other words, a quasi-$\mathbb{Z}/n\mathbb{Z}$ code is nothing else than a usual *quasi-cyclic* code with block length n.

4.2 Duality for Quasi-Abelian Codes

When dealing with codes, it may be easier to use the language of parity-check matrices, especially when considering random codes. In this section, we show that this also extends naturally to quasi-abelian codes.

Let G be an abelian group. The algebra $\mathbb{F}_q[G]$ is naturally endowed with an inner product $\langle \cdot, \cdot \rangle$ defined as follows:

$$\left\langle \sum_{g \in G} a_g g, \sum_{g \in G} b_g g \right\rangle \stackrel{\text{def}}{=} \sum_{g \in G} a_g b_g,$$

which is simply the usual inner product over $\mathbb{F}_q^n$ (this does not depend on the ordering of G). This inner product can be naturally extended to $(\mathbb{F}_q[G])^\ell$:

$$\langle (a_1, \ldots, a_\ell), (b_1, \ldots, b_\ell) \rangle \stackrel{\text{def}}{=} \sum_{i=1}^{\ell} \langle a_i, b_i \rangle,$$

and the notion of the dual $\mathcal{C}^\perp$ of a code $\mathcal{C}$ extends to quasi-abelian codes:

$$\mathcal{C}^\perp \stackrel{\text{def}}{=} \left\{ x \in (\mathbb{F}_q[G])^\ell \mid \langle x, c \rangle = 0 \quad \forall c \in \mathcal{C} \right\}.$$

Proposition 11. *Let G be a finite abelian group and let $\mathcal{C}$ be a quasi-G code of index ℓ. Then $\mathcal{C}^\perp$ is also a quasi-G code of index ℓ.*

Proof. There needs only to prove that $\mathcal{C}^\perp$ is kept invariant by the action of $\mathbb{F}_q[G]$.

For any $a = \sum_{g \in G} a_g g \in \mathbb{F}_q[G]$, define $\bar{a} \stackrel{\text{def}}{=} \sum_{g \in G} a_g g^{-1} \in \mathbb{F}_q[G]$ and $\sigma(a) \stackrel{\text{def}}{=} a_{1_G} \in \mathbb{F}_q$ where 1_G denotes the identity element of G. The map $a \mapsto \bar{a}$ is clearly an automorphism of $\mathbb{F}_q[G]$ of order 2, and $\sigma : \mathbb{F}_q[G] \mapsto \mathbb{F}_q$ is a linear form. Moreover, for $a, b \in \mathbb{F}_q[G]$, a simple computation shows that $\langle a, b \rangle = \sigma(a\bar{b})$.

Now, let $x = (x_1, \ldots, x_\ell) \in \mathcal{C}^\perp$. For any $c = (c_1, \ldots, c_\ell) \in \mathcal{C}$ and any $a \in \mathbb{F}_q[G]$,

$$\langle x \cdot a, c \rangle = \sum_{i=1}^{\ell} \sigma((x_i a)\bar{c_i}) = \sum_{i=1}^{\ell} \sigma(x_i \overline{(c_i \bar{a})}) = \langle x, c \cdot \bar{a} \rangle = 0,$$

where in the last equality we used the fact that $c \cdot \bar{a} \in \mathcal{C}$ since $\mathcal{C}$ is an $\mathbb{F}_q[G]-$module. This concludes the proof of the proposition. □

Example 12. Consider a quasi-abelian code $\mathcal{C}$ of index 2, with a systematic generator matrix $\mathbf{\Gamma} = (1 \mid a)$. Then, $\mathcal{C}$ admits a parity-check matrix of the form $\mathbf{H} = (\bar{a} \mid -1)$.

4.3 Fast-Fourier Transform and Encoding

This Section recalls Fast Fourier Transform algorithms in a general setting. This encompasses the usual FFT introduced by Cooley and Tuckey in 1965[CT65][8] or the Number Theoretic Transform (NTT) algorithm with which the reader might be more familiar. For a detailed presentation in the group algebra setting (see [Obe07]).

Let G be a finite abelian group[9] of cardinality n, $\mathbb{F}_q$ a finite field with q elements, and consider the group algebra $\mathbb{F}_q[G]$. As explained above, encoding a quasi-G code amounts to computing multiplications in $\mathbb{F}_q[G]$ which can be done using Discrete Fourier Transform algorithms (DFT) when $\gcd(n, q) = 1$. Indeed, in this case Maschke theorem ensures that $\mathbb{F}_q[G]$ is semisimple, *i.e.* $\mathbb{F}_q[G]$ is isomorphic to a direct product of finite fields[10], where the product is now done componentwise. DFT-based algorithms to compute the products of two elements of $\mathbb{F}_q[G]$ always follow the same strategy:

1. Compute the forward map $\mathbb{F}_q[G] \to \mathbb{F}_{q^{\ell_1}} \times \cdots \times \mathbb{F}_{q^{\ell_r}}$ [11].
2. Compute the componentwise products.
3. Compute the inverse map $\mathbb{F}_{q^{\ell_1}} \times \cdots \times \mathbb{F}_{q^{\ell_r}} \to \mathbb{F}_q[G]$.

[8] Although such an algorithm was already probably known by Gauss.

[9] Recall than in this work we restrict ourselves to the abelian setting, though a Fourier Transform theory exists also for non-abelian group algebras, making use of the theory of characters.

[10] This uses the abelianity of G, in general $\mathbb{F}_q[G]$ is a direct product of matrix algebras

[11] This is what is usually called the Discrete Fourier Transform.

Fast Fourier Transform (FFT) algorithms correspond to the case where steps 1 and 3 can be done efficiently (typically in $O(n\log(n))$ operations in $\mathbb{F}_q$ compared to a quadratic *naive* approach.) This operation is all the more efficient when $\ell_i = 1$ for all i. This happens when $\mathbb{F}_q$ contains a primitive d-th root of unity, where $d = \exp(G)$ is the *exponent* of G, *i.e.* the *lcm* of the orders of all elements of G. For our applications, this will always be the case.

Recall that any finite group G has a Jordan-Hölder composition series:

$$\{1_G\} = G_0 \lhd G_1 \lhd \cdots \lhd G_r = G$$

such that the quotients G_{i+1}/G_i (called the *factors* of the series) are simple groups (*i.e.* in the abelian setting they are isomorphic to some $\mathbb{Z}/p_i\mathbb{Z}$ where p_i is a prime.), and this composition series is uniquely defined, up to equivalence (*i.e.* all Jordan-Hölder series have same length and the factors are the same up to permutation).

Proposition 13 ([Obe07, Section 5]). *Consider a finite abelian group G of cardinality n with $\gcd(n, q) = 1$, and exponent d. Assume that $\mathbb{F}_q$ contains a primitive d-th root of unity. Let $p_1, \ldots, p_r$ denote all the primes (possibly non distinct) appearing in the Jordan-Hölder series of G (in particular $n = p_1 \cdots p_r$). Then the Discrete Fourier Transform (and its inverse) in $\mathbb{F}_q[G]$ can be computed in $O(n \times (p_1 + \cdots + p_r))$ operations in $\mathbb{F}_q$.*

Example 14. Proposition 13 encompasses well-known FFT's from the literature.

- The usual FFT corresponds to $G = \mathbb{Z}/2^t\mathbb{Z}$. In this case, a composition series is given by

  $$G_0 = \{0\} \subset \cdots \subset G_i = 2^{t-i}\mathbb{Z}/2^t\mathbb{Z} \subset \cdots \subset G_t = G = \mathbb{Z}/2^t\mathbb{Z},$$

  and each factor G_{i+1}/G_i is isomorphic to $\mathbb{Z}/2\mathbb{Z}$, and with the above proposition we recover the usual complexity in $O(2^t \times t) = O(n\log(n))$. However, $\mathbb{F}_q$ needs to be large enough to contain a primitive 2^t-th root of unity[12].

- Consider the finite field $\mathbb{F}_3$ and the group $G = (\mathbb{Z}/2\mathbb{Z})^t$. Example 8 entails that

  $$\mathbb{F}_3[G] \simeq \mathbb{F}_3[X_1, \ldots, X_t]/(X_1^2 - 1, \ldots, X_t^2 - 1).$$

  A composition series of G is given by

  $$G_0 = \{0\}^t \subset \cdots \subset G_i = (\mathbb{Z}/2\mathbb{Z})^i \times \{0\}^{t-i} \subset \cdots \subset G_t = G = (\mathbb{Z}/2\mathbb{Z})^t,$$

  and the FFT can also be computed in time $O(2^t \times t) = O(n\log(n))$. This is nothing else than a t-dimensional FFT in $\mathbb{F}_3$.

Remark 15. Proposition 13 is *asymptotic*, although efficient implementations exist for several groups and fields. They are particularly efficient when G admits a Jordan-Hölder composition series with groups of index 2, such as in the above two examples, which allows a simple divide-and-conquer approach. For a more precise description of Multivariate FFT algorithms (see [vdHLS13, Section 2.2]).

4.4 The Quasi-Abelian Decoding Problem

In this section, we introduce computationally hard problems related to random quasi-abelian codes. They are variants of the Syndrome Decoding Problem, restricted to this class of codes.

Let G be a finite abelian group and $\mathbb{F}_q$ a finite field with q elements. Given an integer $t \in \mathbb{N}$, we denote by $\mathcal{D}_t(\mathbb{F}_q[G])$ a noise distribution over $\mathbb{F}_q[G]$ such that $\mathbb{E}[\mathrm{wt}(x)] = t$ when $x \xleftarrow{\$} \mathcal{D}_t$, and $\mathcal{D}_{t,n}(\mathbb{F}_q[G]) \stackrel{\text{def}}{=} \mathcal{D}_t(\mathbb{F}_q[G])^{\otimes n}$ will denote its n-fold tensorization, *i.e.* $\mathbf{e} \xleftarrow{\$} \mathcal{D}_{t,n}(\mathbb{F}_q[G])$ is $\mathbf{e} \in \mathbb{F}_q[G]^n$ and its coordinates are drawn independently according to $\mathcal{D}_t(\mathbb{F}_q[G])$. A *random* quasi-$G$ code of index 2, in *systematic form*, will be a quasi-G code whose parity-check matrix $\mathbf{H} \in (\mathbb{F}_q[G])^2$ is of the form $\mathbf{H} = (\mathbf{1} \mid \mathbf{a})$, where a is uniformly distributed over $\mathbb{F}_q[G]$. Equivalently, it is the dual of the code generated by $\mathbf{H}$. The search Quasi-Abelian Syndrome Decoding problem is defined as follows:

[12] When the characteristic of $\mathbb{F}_q$ is not too large, an approach based on the Frobenius Fast Fourier Transform can also be exploited to remove this fact.

Definition 16 ((Search) QA-SD problem). *Given* $\mathbf{H} = (\mathbf{1} \mid \mathbf{a})$ *a parity-check matrix of a random systematic quasi-abelian code, a target weight* $t \in \mathbb{N}$ *and a syndrome* $\mathbf{s} \in \mathbb{F}_q[G]$*, the goal is to recover an error* $\mathbf{e} = (\mathbf{e}_1 \mid \mathbf{e}_2)$ *with* $\mathbf{e}_i \xleftarrow{\$} \mathcal{D}_t(\mathbb{F}_q[G])$ *such that* $\mathbf{H}\mathbf{e}^T = \mathbf{s}$*,* i.e. $\mathbf{e}_1 + \mathbf{a} \cdot \mathbf{e}_2 = \mathbf{s}$.

The problem also has a decisional version.

Definition 17 ((Decisional) QA-SD problem). *Given a target weight* t*, the goal of this decisional* QA-SD *problem is to distinguish, with a non-negligible advantage, between the distributions*

$$\begin{array}{ll} \mathcal{D}_0 : & (\mathbf{a}, \mathbf{s}) \quad \text{where } \mathbf{a}, \mathbf{s} \xleftarrow{\$} \mathbb{F}_q[G] \\ \mathcal{D}_1 : & (\mathbf{a}, \mathbf{a} \cdot \mathbf{e}_1 + \mathbf{e}_2) \quad \text{where } \mathbf{a} \xleftarrow{\$} \mathbb{F}_q[G] \text{ and } \mathbf{e}_i \xleftarrow{\$} \mathcal{D}_t(\mathbb{F}_q[G]). \end{array}$$

Both assumptions above generalize immediately to the case of parity-check matrices with more columns and/or rows of blocks. When $\mathbf{H} = (\mathbf{1} \mid \mathbf{a_1} \mid \cdots \mid \mathbf{a_{c-1}})$, for some parameter c, this corresponds to what has been called Module-LPN in the literature. This corresponds to the hardness of syndrome decoding for a quasi-abelian code of larger rate $(c-1)/c$. We call (search, decisional) QA-SD$(c, \mathcal{R})$ this natural generalization of QA-SD.

The QA-SD assumption states that the above decisional problem should be hard (for appropriate parameters). When the group G is the trivial group, this is the usual *plain* SD-assumption, while when the group G is cyclic[13], this is the QC-SD assumption at the core of Round 4 NIST submissions BIKE and HQC. Those problems, especially their search version, have been studied for over 50 years by the coding theory community and to this day, no efficient algorithm is known to decode a random quasi-abelian code. This is even listed as an open research problem in the most recent Encyclopedia of Coding Theory (from 2021) [Wil21, Problem 16.10.5].

Remark 18. In Definition 17, we consider quasi-abelian codes with a parity-check matrix in systematic form. Indeed, assume $\mathbf{H} = (\mathbf{a}_1 \mid \mathbf{a}_2) \in \mathbb{F}_q[G]^{1\times 2}$. A syndrome of $\mathbf{H}$ will be of the form $\mathbf{s} = \mathbf{a}_1\mathbf{e}_1 + \mathbf{a}_2\mathbf{e}_2$, and therefore is contained in the ideal $\mathcal{I} = (\mathbf{a}_1, \mathbf{a}_2)$ of $\mathbb{F}_q[G]$ generated by $\mathbf{a}_1$ and $\mathbf{a}_2$[14]. Therefore, when this ideal is *not* the full ring, there is an obvious bias. When working over a large field $\mathbb{F}_q$, elements of $\mathbb{F}_q[G]$ are invertible with high probability, and therefore $\mathcal{I} = \mathbb{F}_q[G]$ with overwhelming probability. On the other hand, this is not true anymore when working over small fields. Using parity-check matrices in systematic form ensures that $1_G \in \mathcal{I}$, which removes the bias. This is a standard definition (see for instance [AAB+22a, AAB+22b]), though not always formulated like that in the literature.

4.5 Security Analysis

In this paragraph, we provide evidence for the QA-SD-assumption. Note first that for $G = \{1\}$ it is nothing but the SD-assumption, which is well established. Moreover, we argue for security of QA-SD against linear tests (Definition 3). With Lemma 4 in hand, it suffices to show that given the parity-check matrix $\mathbf{H}$ of a quasi-G code $\mathcal{C}$, the minimum distance of the code *generated by* $\mathbf{H}$, *i.e.* the *dual* of $\mathcal{C}$, is large with high probability (over the choice of $\mathbf{H}$). Note that when $G = \{1\}$, it is well–known that random codes are good, *i.e.* meet the Gilbert-Varshamov (GV) bound (see for instance [Pie67, BF02, Deb23]).

Proposition 19 (Gilbert-Varshamov). *Let* $0 < \delta < 1 - \frac{1}{q}$*. Let* $\varepsilon > 0$*, and let* $\mathcal{C}$ *be a random code of rate* $\frac{k}{n} \leqslant (1 - h_q(\delta) - \varepsilon)$*. Then,*

$$\mathbb{P}\left(d_{min}(\mathcal{C}) > \delta n\right) \geqslant 1 - q^{-\varepsilon n},$$

where the probability is taken over the uniform choice of a generator matrix of $\mathcal{C}$*, and* h_q *denotes the* q*-ary entropy function*

$$h_q(x) \stackrel{def}{=} -x \log_q\left(\frac{x}{q-1}\right) - (1-x)\log_q(1-x).$$

[13] and $\gcd(q, |G|) = 1$

[14] Beware that $\mathbb{F}_q[G]$ is not necessarily principal.

For the past 50 years, it has been a long trend of research in coding theory to extend such a result for more general quasi-abelian codes. For the class of quasi-cyclic codes which are, by far, the most used quasi-abelian codes in cryptography, a GV-like bound was introduced by Kasami in [Kas74]. Gaborit and Zémor even showed in [GZ06] that various families of random double-circulant codes asymptotically satisfied a logarithmic improvement on this bound. More recently, this state of affairs was extended by Fan and Lin in [FL15] to *any* quasi-abelian code, even in the modular case where $char(\mathbb{F}_q)$ is *not* coprime to $|G|$. The proof of this result makes use of the theory of representations of finite abelian groups in $\mathbb{F}_q$.

Theorem 20 ([FL15, Theorem 2.1]). *Let G be a finite abelian group, and let $(\mathcal{C}_\ell)_\ell$ be a sequence of random quasi-G codes of length $\ell \in \mathbb{N}$ and rate $r \in (0,1)$. Let $\delta \in (0, 1 - \frac{1}{q})$. Then,*

$$\lim_{\ell \to \infty} \mathbb{P}\left(\frac{d_{min}(\mathcal{C}_\ell)}{|G|} > \delta\ell\right) = \begin{cases} 1 & \text{if } r < 1 - h_q(\delta); \\ 0 & \text{if } r > 1 - h_q(\delta); \end{cases}$$

and both limits converge exponentially fast. The above probability is taken over the uniform choice of a generator matrix $\mathbf{G}_\ell \in \mathbb{F}_q[G]^{k \times \ell}$ of $\mathcal{C}_\ell$.

As it is often the case in coding theory, this result is stated asymptotically, but the convergence speed could be made more precise, the exponent depends on $|G|$: the larger the group G, the higher this probability. Actually, to assert the resistance of QA-SD against linear attacks, it would be more relevant to consider the regime where k, ℓ are constant and $|G|$ goes to infinity as it is done in [GZ06] but such a development is out of reach of this article and we leave it as a conjecture. There is a caveat though. Indeed, as it was noticed in Remark 18, in the case of constant k, ℓ and growing $|G|$ there is a bias in the QA-SD distribution when the ideal generated by the blocks in the input parity-check matrix is not the full ring. This corresponds to the parity-check matrix not being *full-rank* when seen as a matrix over $\mathbb{F}_q[G]$. In this case, the minimum distance could drop, but heuristically a random quasi-G code will have a minimum distance linear in its length as long as this bias is removed, which is the case in our setting since we enforce the systematic form.

Example 21. In order to produce OLE's over the field $\mathbb{F}_p$, [BCG+20b] proposed to use a ring $\mathcal{R}$ of the form $\mathbb{F}_p[X]/(F(X))$ where $F(X)$ is totally split in $\mathbb{F}_p$.

- The choice of polynomial F that maximizes the number of OLE would be the polynomial $F(X) = X^p - X$ which has precisely all its roots in $\mathbb{F}_p$ (This is *not* the choice recommended by the authors, but is still allowed in their framework). However, this ring does not fit in our setting, and in fact the SD-problem in this ring is vulnerable to a very simple linear attack: given (a, b) where b is either random or equal to $a \cdot e + f \bmod X^p - X$, it holds that $e(0) = f(0) = 0 \bmod X^q - X$ with high probability, because $e(0) = f(0) = 0$ over $\mathbb{F}_p[X]$ with high probability (since e, f are sparse, their constant coefficient is likely to be zero), and reduction modulo $X^p - X$ does not change the constant coefficient. Hence, the adversary can distinguish b from random simply by computing $b(0)$ (since $b(0)$ is nonzero with probability $(p-1)/p$ for a random b).
- However, by simply removing the X factor and setting $F(X) = X^{p-1} - 1$, which would yield $p-1$ copies of $\mathbb{F}_p$ instead of p, the ring $\mathcal{R} = \mathbb{F}_p[X]/(X^{p-1} - 1)$ is nothing else than the group ring $\mathbb{F}_p[\mathbb{F}_p^\times]$ and totally fits in our framework. In particular, it resists linear attacks. Note that the previous evaluation at 0 does no longer make sense.

5 Pseudorandom Correlation Generators from QA-SD

In the following we always consider $\mathcal{R} = \mathbb{F}_q[G] = \left\{\sum_{g \in G} a_g g \mid a_g \in \mathbb{F}_q\right\}$, with G an abelian group. We refer to $\mathcal{R}_t$ as the set of ring elements of $\mathcal{R}$ of maximum weight t.

5.1 A Template for Programmable PCG for OLE from QA-SD

Theorem 22. *Let $\mathcal{R} = \mathbb{F}_q[G]$. Assume that SPFSS is a secure FSS scheme for sums of point functions, and that the QA-SD$(c, \mathcal{R})$ assumption holds. Then there exists a generic construction scheme to construct a PCG to produce one OLE correlation (described on Fig. 3). If the SPFSS is based on a PRG : $\{0,1\}^\lambda \to \{0,1\}^{2\lambda+2}$ via the PRG-based construction from [BGI16], we obtain:*

- *Each party's seed has maximum size around :* $(ct)^2 \cdot ((\log|G| - \log t + 1) \cdot (\lambda + 2) + \lambda + \log q) + ct(\log|G| + \log q)$ *bits*
- *The computation of* Expand *can be done with at most* $(2 + \lfloor(\log q)/\lambda\rfloor)|G|c^2 t$ PRG *operations, and* $O(c^2|G|\log|G|)$ *operations in* $\mathbb{F}_q$.

The protocol, adapted from the work of Boyle et al. [BCG+20b], is described on Fig. 3. We first present an overview. Remind that an instance of the OLE correlation consists in giving a random value $x_\sigma \in \mathcal{R}$ to party P_σ as well as an additive secret sharing of $x_0 \cdot x_1 \in \mathcal{R}$ to both. Formally:

$$\left\{((x_0, z_0), (x_1, z_1)) | x_0, x_1, z_0 \xleftarrow{\$} \mathcal{R}, z_1 + z_0 = x_0 \cdot x_1\right\}.$$

The core idea of the protocol is to give the two parties a random vector $\mathbf{e_0}$ or $\mathbf{e_1} \in \mathcal{R}_t^c$, where each element of the vector is sparse. In addition, parties have access to a vector $\mathbf{a} = (1, \dot{\mathbf{a}})$, with $\dot{\mathbf{a}} = (a_1, \cdots, a_{c-1})$, a vector of random elements of $\mathcal{R}$. We see the vector $\mathbf{e}_\sigma$ of party P_σ as an error vector. Using the vector $\mathbf{a}$, parties can locally extend their error vector and construct $x_\sigma = \langle \mathbf{a}, \mathbf{e_\sigma}\rangle$, which is pseudorandom under QA-SD.

We want to give the parties shares of $x_0 \cdot x_1$. Note that $x_0 \cdot x_1$ is a degree 2 function in $(\mathbf{e_0}, \mathbf{e_1})$; therefore, it suffices to share $\mathbf{e_0} \otimes \mathbf{e_1}$. We underline a property of the sparse elements in $\mathcal{R}_t$. Let e, f be sparse elements. This means that there exist sets $S_e, S_f \subset G$, such that $e = \sum_{g\in S_e} e_g g, f = \sum_{g\in S_f} f_g g$ with $e_g, f_g \in \mathbb{F}_q$ and $|S_e| = |S_f| = t \leqslant |G|$. It follows that the product of $e \cdot f$ can be expressed using only $S_e \cdot S_f \stackrel{\text{def}}{=} \{gh \mid g \in S_e,\ h \in S_f\}$ as basis. We conclude with $|S_e \cdot S_f| < |S_e| \cdot |S_f| = t^2$, to deduce that the product of sparse vectors in $\mathcal{R}$ also gives us sparse vectors (with sparsity t^2 instead of t). We note that here, we deviate from the original construction of [BCG+20b]: over a ring of the form $\mathbb{F}_q[X]/P(X)$ where P is some polynomial, it is not generally true that the product of sparse elements remains sparse. This is circumvented in [BCG+20b] by sharing the product over $\mathbb{F}_q[X]$ instead, and reducing locally. When using group algebras as we do, the product preserves sparsity and we can share the product directly within $\mathbb{F}_q[G]$, which is slightly more efficient.

This result enables us to express each element of $\mathbf{e_0} \otimes \mathbf{e_1}$ as a sum of t^2 point functions. Then, we rely on SPFSS (Definition 36). Recall that an SPFSS takes as input a sequence of points as well as a vector of values, and produces two keys that can be use to find shares of the sum of the implicit point functions. When a party evaluates its key at each point in the domain, it obtains a pseudorandom secret sharing of the coefficients of the sparse element in $\mathcal{R}_t$. The protocol uses c^2 elements of $\mathcal{R}_t$ as a result of the tensor product. This means that we need c^2 instances of SPFSS for t^2 point functions. This gives us a seed size of $O(\lambda(ct)^2 \log|G|) = O(\lambda^3 \log|G|)$.

Proof (of Theorem 22). First, we argue the correctness of the protocol. The coefficient vectors $\mathbf{b}^\mathbf{i}_\sigma, \mathbf{A}^\mathbf{i}_\sigma$ define a random element in $\mathcal{R}_t$. We can rewrite the product of two of these elements as follows:

$$e_0^i \cdot e_1^j = \sum_{k,l\in[0..t)} \mathbf{b}^\mathbf{i}_\mathbf{0}[k] \cdot \mathbf{b}^\mathbf{j}_\mathbf{1}[l]\mathbf{A}^\mathbf{i}_\mathbf{0}[k]\mathbf{A}^\mathbf{j}_\mathbf{1}[l].$$

This can indeed be described by a sum of point functions. From this point, $\mathbf{u} = \mathbf{u_0} + \mathbf{u_1}$, then $\mathbf{u} = \mathbf{e_0} \otimes \mathbf{e_1}$, each entry being equal to one of those $e_0^i \cdot e_1^j$. The party obtains z_σ as an output, and we can verify:

$$z_0 + z_1 = \langle \mathbf{a} \otimes \mathbf{a}, \mathbf{u_0} + \mathbf{u_1}\rangle = \langle \mathbf{a} \otimes \mathbf{a}, \mathbf{e_0} \otimes \mathbf{e_1}\rangle = \langle \mathbf{a} \otimes \mathbf{e_0}\rangle \cdot \langle \mathbf{a} \otimes \mathbf{e_1}\rangle = x_0 \cdot x_1.$$

The next-to-last equality is straightforward to check. Note that here, $\langle \mathbf{a}, \mathbf{e_\sigma}\rangle$ is a QA-SD sample, with fixed random $\mathbf{a}$ and independent secret e_σ. We now briefly show sketch security (the analysis is essentially identical to [BCG+20b] since the construction is "black-box" in the ring $\mathcal{R}$, we sketch it for completeness). As the two cases are symmetrical, we assume $\sigma = 1$. Let $(k_0, k_1) \xleftarrow{\$} \mathsf{PCG.Gen}(1^\lambda)$ with associated expanded outputs (x_0, z_0) and (x_1, z_1), we need to show that

$$\{(k_1, x_0, z_0)\} \equiv \left\{(k_1, \tilde{x_0}, \tilde{z_0}) \mid \tilde{x_0} \xleftarrow{\$} \mathcal{R}, \tilde{z_0} = \tilde{x_0} \cdot x_1 - z_1\right\}.$$

To show this, we use a sequence of hybrid distributions.

– Replace z_0 by $x_0 \cdot x_1 - z_1$.
– Step by step replace each the FSS key $K_1^{i,j}$ in k_1 by a simulated key generated only with the range and the domain of the function. Due of the correctness and the security properties of the FSS scheme, this distribution is indistinguishable from the original distribution.
– Replace x_0 by a fresh $\tilde{x}_0$. It is also impossible to distinguish this distribution from the previous one, since the $K_1^{i,j}$ are now completely independent of x_0, and we can rely on the QA-SD assumption.
– Reverse step 2 by using the FSS security property once again. □

Regarding the size of the different parameters, we use the optimization suggested in [BCG+20b], such as assuming that the QA-SD assumption holds also for *regular error distributions* (we note that our proof of resistance against linear tests holds for very general noise distributions, and in particular for the regular noise distribution). We can thus reduce the seeds size to $(ct)^2 \cdot ((\log |G| - \log t + 1) \cdot (\lambda + 2) + \lambda + \log q) + ct(\log N + \log q)$ bits ; and the number of PRG calls in Expand down to $(2 + \lfloor (\log q)/\lambda \rfloor)|G|c^2t$. Note that to achieve security, choosing $ct = O(\lambda)$ is sufficient. The number of PRG calls can be further reduced to $O(|G|c^2)$ using batch codes to implement the SPFSS.

Theorem 23. *The* PCG *construction for* OLE *from Fig. 3 is programmable.*

Proof. In order to show that our PCG is programmable we have to transform it a little, as the Gen functionality takes additional inputs (ρ_0, ρ_1) in the programmability definition. In our case, we can choose $\rho_\sigma = \left\{\mathbf{A}^{\mathbf{i}}_{\boldsymbol{\sigma}}, \mathbf{b}^{\mathbf{i}}_{\boldsymbol{\sigma}}\right\}$. In this way, as explained in the description of the protocol, the additional input of the players can be seen as a vector of elements in $\mathcal{R}_t$, $\mathbf{e}_{\boldsymbol{\sigma}} = (\mathbf{e}^{\mathbf{0}}_{\boldsymbol{\sigma}}, \cdots, \mathbf{e}^{\mathbf{c-1}}_{\boldsymbol{\sigma}})$. Because $x_\sigma = \langle \mathbf{a}, \mathbf{e}_{\boldsymbol{\sigma}} \rangle$, the players can compute their first input locally, after expanding their ρ_σ into e_σ. This defines functions ϕ_σ, and proves the programmability property. The proof of the correctness property is the same as in the proof of the Theorem 22. The programmable security property can be proven with s sequence of hybrid distribution as in the proof of Theorem 22, using the reduction to FSS scheme and the QA-SD assumption. □

Distributed Seed Generation The protocol described in Fig. 3 assumes that a trusted dealer has given the parties their seed. What we want to do in practice is to achieve the Gen phase via a distributive setup protocol.

Functionality $\mathsf{QA\text{-}SD}_{\mathsf{OLE-Setup}}$

PARAMETERS: Security parameter 1^λ, $\mathsf{PCG}_{\mathsf{OLE}} = (\mathsf{PCG}_{\mathsf{OLE}}.\mathsf{Gen}, \mathsf{PCG}_{\mathsf{OLE}}.\mathsf{Expand})$ as per Fig. 3
FUNCTIONALITY:

1. Sample $(k_0, k_1) \leftarrow \mathsf{PCG}_{\mathsf{OLE}}.\mathsf{Gen}(1^\lambda)$.
2. Output k_σ to party P_σ for $\sigma \in \{0, 1\}$

Fig. 1. Generic functionality for the distributed setup of OLE PCG seeds

Theorem 24 (From [BCG+20b]). *There exists a protocol securely realizing the functionality* $\mathsf{QA\text{-}SD}_{\mathsf{OLE-Setup}}$ *of Fig. 1 against malicious adversaries, with complexity:*

- *Communication costs per party dominated by* $(ct)^2 \cdot ((2\lambda + 3) \log 2|G| + (9t + 2) \log(q - 1))$.
- *Computation is dominated by* $2|G|$ PRG *evaluations.*

Taking $ct = O(\lambda)$ is enough to achieve exponential security. With this we can conclude a general result:

Theorem 25. *Let G be a group, and $\mathcal{R} = \mathbb{F}_q[G]$. Suppose that* SPFSS *is a secure FSS scheme for sums of point functions, and the* $\mathsf{QA\text{-}SD}(c, \mathcal{R})$ *assumption. Then there exists a protocol securely realizing the* $\mathsf{QA\text{-}SD}_{\mathsf{OLE-All}}$ *functionality over the ring $\mathcal{R}$ with the following parameters*

- *Communication costs and size of the seed :* $O(\lambda^3 \log |G|)$.
- *Computation costs :* $O(\lambda |G|)$ PRG *evaluations and* $O(c^2 |G| \log |G|)$ *operations in* $\mathbb{F}_q$.

Functionality $\mathsf{QA\text{-}SD}_{\mathsf{OLE-All}}$

PARAMETERS: Security parameter, a group G, and a ring $\mathcal{R} = \mathbb{F}_q[G]$.
FUNCTIONALITY:
If both parties are honest:

- Sample $x_0, x_1 \leftarrow \mathcal{R}$
- Sample $z_0 \xleftarrow{\$} \mathcal{R}$ and let $z_1 = x_0 \cdot x_1 - z_0$.
- Output (x_σ, z_σ) to party P_σ for $\sigma \in \{0, 1\}$.

If party P_σ is corrupted:

- Wait for input $(x_\sigma, z_\sigma) \in \mathcal{R}^2$ from the adversary.
- Sample $x_{1-\sigma} \leftarrow \mathcal{R}$ and set $z_{1-\sigma} = x_0 \cdot x_1 - z_\sigma$
- Output $(x_{1-\sigma}, z_{1-\sigma})$ to the honest party.

Fig. 2. OLE Functionality with Corruption

Construction $\mathsf{QA\text{-}SD}_{\mathsf{OLE}}$

PARAMETERS: Security parameter λ, noise weight $t = t(\lambda)$, compression factor $c \geqslant 2$, G a finite abelian group, $\mathcal{R} = \mathbb{F}_q[G]$. An FSS scheme (SPFSS.Gen,SPFSS.FullEval) for sums of t^2 point functions, with domain $[0..|G|)$ and range $\mathbb{F}_q$.
PUBLIC INPUT: $c - 1$ random ring elements $a_1, \cdots, a_{c-1} \in \mathcal{R}$.
CORRELATION: After expansion, outputs $(x_0, z_0) \in \mathcal{R}^2$ and $(x_1, z_1) \in \mathcal{R}^2$ where $z_0 + z_1 = x_0 \cdot x_1$
Gen : On input 1^λ :

1. For $\sigma \in \{0, 1\}$ and $i \in [0..c)$, sample random vectors $\mathbf{A}^{\mathbf{i}}_{\boldsymbol{\sigma}} \leftarrow (g_1, \cdots, g_t)_{g_i \in G}$ and $\mathbf{b}^{\mathbf{i}}_{\boldsymbol{\sigma}} \leftarrow (\mathbb{F}_q^*)^t$.
2. For each $i, j \in [0..c)$, sample FSS keys $(K_0^{i,j}, K_1^{i,j}) \xleftarrow{\$} \mathsf{SPFSS.Gen}(1^\lambda, \mathbf{A}^{\mathbf{i}}_{\mathbf{0}} \otimes \mathbf{A}^{\mathbf{j}}_{\mathbf{1}}, \mathbf{b}^{\mathbf{i}}_{\mathbf{0}} \otimes \mathbf{b}^{\mathbf{j}}_{\mathbf{1}})$.
3. Let $\mathsf{k}_\sigma = ((K_\sigma^{i,j})_{i,j \in [0..c)}, ((\mathbf{A}_{\boldsymbol{\sigma}}{}^i, \mathbf{b}^{\mathbf{i}}_{\boldsymbol{\sigma}})_{i \in [0..c)})$.
4. Output $(\mathsf{k}_0, \mathsf{k}_1)$.

Expand : On input $(\sigma, \mathsf{k}_\sigma)$:

1. Parse k_σ as $((K_\sigma^{i,j})_{i,j \in [0..c)}, ((\mathbf{A}^{\mathbf{i}}_{\boldsymbol{\sigma}}, \mathbf{b}^{\mathbf{i}}_{\boldsymbol{\sigma}})_{i \in [0..c)})$
2. For $i \in [0..c)$, define the element of $\mathcal{R}_t$
$$e^i_\sigma = \sum_{j \in [0..t)} \mathbf{b}^{\mathbf{i}}_{\boldsymbol{\sigma}}[j] \cdot \mathbf{A}^{\mathbf{i}}_{\boldsymbol{\sigma}}[j].$$
3. Compute $x_\sigma = \langle \mathbf{a}, \mathbf{e}_{\boldsymbol{\sigma}} \rangle$, where $\mathbf{a} = (1, a_1, \cdot, a_{c-1}), \mathbf{e}_{\boldsymbol{\sigma}} = (e^0_\sigma, \cdots, e^{c-1}_\sigma)$.
4. For $i, j \in [0..c)$, compute $u_{\sigma, i+cj} \leftarrow \mathsf{SPFSS.FullEval}(\sigma, K_\sigma^{i,j})$ and view it as a c^2 vector of element in $\mathcal{R}_{t^2}$.
5. Compute $z_\sigma = \langle \mathbf{a} \otimes \mathbf{a}, \mathbf{u}_{\boldsymbol{\sigma}} \rangle$.
6. Output x_σ, z_σ .

Fig. 3. PCG for OLE over $\mathcal{R}$, based on QA-SD

5.2 Instantiating the Group Algebra

In this section we instantiate our general result with a concrete construction of a PCG for OLE correlation over $\mathbb{F}_q$. Remind that $G = \prod_{i=1}^n \mathbb{Z}/q_i\mathbb{Z}$, $q_i \geqslant 2$. Using Proposition 7 from previous

section:

$$\mathbb{F}_q[G] = \mathbb{F}_q\left[\prod_{i=1}^{n}\mathbb{Z}/q_i\mathbb{Z}\right] \simeq \mathbb{F}_q[\mathbb{Z}/q_1\mathbb{Z}] \otimes_{\mathbb{F}_q} \cdots \otimes_{\mathbb{F}_q} \mathbb{F}_q[\mathbb{Z}/q_n\mathbb{Z}]$$
$$\simeq \bigotimes_{i=1}^{n}\mathbb{F}_q[X_i]/(X_1^{q_i}-1) \simeq \mathbb{F}_q[X_1,..,X_n]/(X_1^{q_1}-1,..,X_n^{q_n}-1).$$

Batch-OLE over $\mathbb{F}_q$. In the following we let all the q_i be all equal to $q-1$. We therefore use $\mathcal{R} = \mathbb{F}_q[G] \simeq \mathbb{F}_q[X_1,..,X_n]/(X_1^{q-1}-1,..,X_n^{q-1}-1)$. Remark that the elements of $\mathbb{F}_q^*$ are the roots of the polynomial $X_i^{q-1}-1$. Therefore, we can write $X_i^{q-1}-1 = \prod_{a\in\mathbb{F}_p^*}(X_i-a)$, for all $1 \leqslant i \leqslant n$ and, by the Chinese Remainder Theorem, we get

$$\mathbb{F}_q[X_1,..,X_n]/(X_1^{q-1}-1,..,X_n^{q-1}-1) \simeq \prod_{i=1}^{T}\mathbb{F}_q.$$

where $T=(q-1)^n$ is the number of elements in the group. We can apply our protocol to construct a PCG for the OLE correlation in $\mathcal{R}$. This single OLE over $\mathcal{R}$ can be transformed in T different instances of OLE over $\mathbb{F}_q$. We get:

Theorem 26. *Suppose that* SPFSS *is a secure* FSS *scheme for sums of point functions and that the* QA-SD *assumption holds. Let* $\mathcal{R} = \mathbb{F}_q[X_1,..,X_n]/(X_1^{q-1}-1,..,X_n^{q-1}-1)$*, and* $T=(q-1)^n$*. We can construct a* PCG *producing* T *instances for* OLE *over* $\mathbb{F}_p$*, using the* $\mathsf{QA\text{-}SD}_{\mathsf{OLE}}$ *construction. The parameters we obtain are the following.*

- *Each party's seed has size at most:* $(ct)^2\cdot((n\log(q-1)-\log t+1)\cdot(\lambda+2)+\lambda+\log q)+ct(n\log(q-1)+\log q)$ *bits*
- *The computation of* Expand *can be done with at most* $(2+\lfloor(\log q)/\lambda\rfloor)n\log(q-1)c^2t$ PRG *operations, and* $O(c^2(q-1)^n n\log(q-1))$ *operations in* $\mathbb{F}_q$*.*

Concrete Parameters. We report on Table 1 a set of concrete parameters for our new programmable PCGs from QA-SD, when generating T instances of a pseudorandom OLE over $\mathbb{F}_q$, chosen according to the analysis of of Section 6. We note that our concrete security parameters are very close to the parameters of [BCG+20b]. This stems from two points:

First, [BCG+20b] conservatively chose security bounds based on existing attacks over $\mathbb{F}_2$, even though their instantiation is over $\mathbb{F}_p$ with $\log p \approx 128$ (and known attacks on syndrome decoding are less efficient over larger fields). One of the reason for this was to get conservative estimates (syndrome decoding over large fields was less investigated, and attacks could improve in the future); another motivation is that over $\mathbb{F}_2$, tools have been implemented to automatically evaluate the resistance against various flavors of ISD (whose exact cost can be quite tedious to analyze). Because our PCGs can handle fields as low as $\mathbb{F}_3$, and to avoid having to pick different parameters for each field size, we also based our analysis on known bounds for $\mathbb{F}_2$.

Second, the main difference between our analysis and that of [BCG+20b] is that we must consider folding attacks, which are considerably more diverse in our setting (since an attacker can construct a reduced instance by quotienting with *any* subgroup G', of which there are many). Yet, the *effect* of folding on security does not depend on the fine details of the subgroup G', but only on the *size* of G', which allows to compute the new dimension and the reduced noise weight (via a generalized piling-up lemma). This does not differ significantly from the case of ring-LPN over cyclotomic rings considered in [BCG+20b], since there the adversary could reduce the dimension to any power of two of their choice: our setting allows the adversary to be slightly more fine grained in its dimension reduction (*i.e.* the adversary is not restricted to a power of two), but this does not make a significant difference on the concrete attack cost (essentially because close dimensions yield near-identical noise reduction via the piling-up lemma, and do not have significantly different impact on the concrete attack cost beyond that).

As our table illustrates, our PCG's offer a non-trivial stretch (computed as the ratio between the seed size and the size of storing the output OLE's) from a target number $T=2^{25}$ of OLE's.

Table 1. Concrete parameters and seed sizes (per party, counted in bits) for our PCG for OLE over $\mathbb{F}_q$ from QA-SD($\mathcal{R}$), using $\mathcal{R} = \mathbb{F}_q[(\mathbb{Z}/(q-1)\mathbb{Z})^n]$, $\lambda = 128$, target number $T = (q-1)^n$ of OLE's, syndrome compression factor $c \in \{2, 4\}$, and number of noisy coordinates t. 'Stretch', computed as $2T/(\text{seed size})$, is the ratio between storing a full random OLE (i.e., $2T$ field elements) and the smaller PCG seed. The parameter k denotes the dimension of the SD instance after folding, and t' the (expected) noise weight of the folded instance (when heuristically choosing the best possible folding for the adversary). #PRG calls is computed as $4 \cdot Tct$. Parameters are chosen to achieve λ-bits of security against known attacks, according to the analysis of Section 6.

T	c	t	(k, t')	Seed size	Stretch	# R-mults	#PRG calls
2^{25}	2	152	$(2^8, 121)$	$2^{26.0}/\log q$	$\log q$	4	$2^{28.2} \cdot \log q$
2^{25}	4	64	$(3 \cdot 2^8, 60)$	$2^{23.6}/\log q$	$5.3 \log q$	16	$2^{28.0} \cdot \log q$
2^{30}	2	152	$(2^8, 121)$	$2^{26.3}/\log q$	$26 \log q$	4	$2^{33.2} \cdot \log q$
2^{30}	4	64	$(3 \cdot 2^8, 60)$	$2^{24.0}/\log q$	$128 \log q$	16	$2^{33.0} \cdot \log q$
2^{35}	2	152	$(2^8, 121)$	$2^{26.6}/\log q$	$676 \log q$	4	$2^{38.2} \cdot \log q$
2^{35}	4	64	$(3 \cdot 2^8, 60)$	$2^{24.3}/\log q$	$3327 \log q$	16	$2^{38.0} \cdot \log q$

Discussions on Efficient FFTs. Operations over the group algebra can be accelerated using the generalized FFT. Here, we briefly remark that some specific values of q yield "FFT-friendly" instances, where the generalized FFT algorithm is extremely efficient (and could even be competitive with the more well-known FFT over cyclotomic rings, with proper optimizations): this is the case whenever $q-1$ is a power of 2, since it enables a very efficient divide and conquer algorithm. For example, this is the case over $\mathbb{F}_3[(\mathbb{Z}/2\mathbb{Z})^{2^n}]$, where the FFT reduces to a 2^n-dimensional FFT over $\mathbb{F}_3$.

From Decision-QA-SD to Search-QA-SD. In Appendix B, we give a reduction from the search version of QA-SD to the decision version for all instances over $\mathcal{R} = \mathbb{F}_q[G]$ where $G = (\mathbb{Z}/(q-1)\mathbb{Z})^n$, which is the group which we use to obtain PCG's for OLE's over $\mathbb{F}_q^{(q-1)^n}$. This provides further support for the security of our PCG schemes, by showing that their security reduces to the *search* QA-SD assumption. More precisely, we prove the following theorem:

Theorem 27. *Let q, t be two integers, and let $G \stackrel{def}{=} (\mathbb{Z}/(q-1)\mathbb{Z})^t$. Let $n \stackrel{def}{=} |G| = (q-1)^t$ and $w \in \{0, \ldots, n\}$ be an admissible weight, and let ψ be an error distribution over $\mathcal{R} \stackrel{def}{=} \mathbb{F}_q[G]$ such that $\mathbb{E}[wt(\mathbf{x})] = w$ when $\mathbf{x}$ is sampled according to ψ. Let $\mathbf{s} \in \mathbb{F}_q[G]$ be a fixed secret.*

Suppose that there exists a distinguisher $\mathcal{A}$ between $(\mathbf{a}, \mathbf{y}^{unif})$ and $(\mathbf{a}, \mathbf{a}\cdot\mathbf{s}+\mathbf{e})$ where $\mathbf{a}, \mathbf{y}^{unif} \leftarrow \mathcal{R}$ and $\mathbf{e} \leftarrow \psi$. Denote by τ its running time and ε its distinguishing advantage. Then, there exists an algorithm that recovers $\mathbf{s} \in \mathcal{R}$ (with an overwhelming probability in n) in time

$$O\left(n^4 \times \frac{1}{\varepsilon^2} \times q \times \tau\right).$$

6 Concrete Cryptanalysis

In this section, we discuss the concrete security of QA-SD. Most of the attacks we discuss in this section fit in the framework of linear tests, and are therefore asymptotically ruled out by our proof of resistance against linear tests. However, while the concrete bounds of the proof are reasonable (in the sense that choosing parameters from these bounds would yield instances that can be reasonably used in practice), they are overly pessimistic. This stems from the fact that the linear test framework rules out *all linear attacks* (even inefficient ones); equivalently, it considers that the adversary can always find a vector $\mathbf{v}$ that minimizes $\text{wt}(\mathbf{v}\cdot\mathbf{H})$. However, in practice, *finding* the vector $\mathbf{v}$ that minimizes $\text{wt}(\mathbf{v} \cdot \mathbf{H})$ is a hard problem. Indeed, this problem, when instantiated with arbitrary codes is known to be NP–complete [Var97] and is commonly assumed to be hard in average and the best know algorithm to solve this search problem are nothing but the algorithms solving SD, *i.e.* all the known variants of ISD.

When choosing concrete parameters, all previous works that rely on LPN or SD choose instead to use parameters derived using the *best possible* $\mathbf{v}$ which can be obtained using existing linear attacks, such as ISD. For all known concrete linear attacks, two codes whose duals have the same minimum distance will yield the same resistance (measured as $\mathrm{wt}(\mathbf{v}\cdot\mathbf{H})$) against these attacks. In other words, these attacks, which are combinatorial in nature, only rely at their core on the distance properties of the code and *not* on its general structure. To get an apple-to-apple efficiency comparison with the state of the art, the natural rule of thumb is therefore to choose parameters similar to those chosen for variants of syndrome decoding with the same minimum distance property: this heuristic was explicitely advocated in [CRR21, Section 1.4]. In our setting, since quasi-abelian codes meet the GV bound (*i.e.* have typically the same minimum distance as random linear codes), this translates to choosing parameters comparable to those of the standard syndrome decoding problem with random codes.

In our context, this would however be too aggressive, since there are known ways in which an attacker *can* exploit the structure of the code. First, because our codes are quasi-abelian codes and hence, according to Remark 9, they can be regarded as codes over a quotient of a multivariate polynomial ring. Therefore, an attacker can reduce the word modulo some ideal of the ring, in order to generate an instance of a "smaller" decoding problem. This approach has been considered in [CT19] in the code–based setting and in [BCV20] in the lattice setting. This point of view has been considered in [BCG$^+$20b] when studying the security of OLE's generated using instances of Ring-LPN.

The parameters should therefore be chosen such that any such "reduced instance" remains intractable. Second, due to the quasi-abelian structure of our codes, one can apply the DOOM attack from [Sen11] to obtain a speedup by a factor $\sqrt{|G|}$, where G denotes the underlying abelian group of the group algebra.

Our setting. In the following, we focus on linear attacks against the QA-SD(n,k) assumption instantiated over a ring $\mathcal{R} = \mathbb{F}_q[X_1,\dots,X_d]/(X_1^{q-1}-1,\dots,X_d^{q-1}-1)$. Our point is to distinguish pairs $((a_1,\dots,a_c), a_1s_1+\cdots+a_cs_c+e)$ (with possibly $c=1$), where $a \xleftarrow{\$} \mathcal{R}$ and $s_1,\dots,s_c,e \in \mathcal{R}$ are sparse with respect to the basis of monomials. As already mentioned in Section 4.4, the search version of the problem is equivalent to solving the QA-SD problem. That is to say solving a decoding problem of the form

$$\left(\ \mathbf{A}_1 \mid \cdots \mid \mathbf{A}_c \mid 1\ \right) \begin{pmatrix} \mathbf{s}_1 \\ \vdots \\ \mathbf{s}_c \\ \mathbf{e} \end{pmatrix} = 0,$$

where the $\mathbf{A}_i$'s are the matrix representations in the basis of monomials of the multiplication–by–a_i maps in $\mathcal{R}$ and the $\mathbf{s}_i$'s and $\mathbf{e}$ are the unknown vector representations of the s_i's and e in this basis, *i.e.* are unknown sparse vectors.

In terms of code parameters, the group codes have length $n = (c+1)\dim_{\mathbb{F}_q}\mathcal{R}$ and dimension $k = c\dim_{\mathbb{F}_q}\mathcal{R}$. Therefore, we always have $k \geqslant \frac{n}{2}$ with equality when $c=1$. In this setting, attacks such as Arora-Ge [AG11] (which require $n = \Omega(k^2)$) or BKW (which require n to be subexponential in k, or $n = \Omega(k^{1+\varepsilon})$ using the sample-efficient variant of [Lyu05]) do not apply. Furthermore, our codes have rate $c/(c+1)$ with $c \geqslant 1$. In particular, this implies that the recent results on Statistical Decoding 2.0 [CDMT23], which improves over ISD when the code rate is sufficiently small, will not yield an efficient attack on our setting (for rates above 1/2, SD 2.0 is always outperformed by ISD).

6.1 Instance Projection via Quotient

As previously mentioned, a manner to solve the problem is to solve the search QA-SD$(\mathcal{R})$ problem, where $\mathcal{R} = \mathbb{F}_q[X_1,\dots,X_d]/(X_1^{q-1}-1,\dots,X_d^{q-1}-1)$. Given an instance (a,b) of QA-SD$(\mathcal{R})$, an attacker may construct a new decoding instance with smaller length and dimension. In full generality, the attacker can pick any ideal $I \subseteq \mathbb{F}_q[X_1,\dots,X_d]$ containing $(X_1^{q-1}-1,\dots,X_d^{q-1}-d)$ and represented by a Gröbner basis, and construct a new instance $(a',b') \leftarrow (a,b) \bmod I$, where the mod operation is the reduction modulo I with respect to the chosen Gröbner basis. For instance,

one can choose a sequence $(F_1(X_1), \ldots, F_d(X_d))$ of factors of $X_1^{q-1} - 1, \ldots, X_d^{q-1} - 1$ and reduce modulo them.

However, in general, the projection modulo an arbitrary ideal I can significantly increase the noise. The way the noise increases is highly dependent from the density of the generators of I. For example, if $\mathcal{R} = \mathbb{F}_q[X_1, X_2]/(X_1^{q-1} - 1, X_2^{q-1} - 1)$ and the attacker reduces modulo $I = (F_1(X_1), F_2(X_2))$ where F_1, F_2 are respective factors of $X_1^{q-1} - 1$ and $X_2^{q-1} - 1$ of respective Hamming weight, say, 3 and 5, the noise rate can increase by a factor up to $(3-1)\cdot(5-1) = 8$. Therefore, we expect this approach to be useful (to the attacker) only when the noise increase is very small.

Heuristically the best possible projections of $\mathcal{R}$ regarded as the group algebra $\mathbb{F}_q[G]$ seem to be the projections arising from quotients of G. Namely, given a subgroup H of G the canonical map $G \to G/H$ induces a morphism of algebras

$$\pi_H : \begin{cases} \mathbb{F}_q[G] & \longrightarrow & \mathbb{F}_q[G/H] \\ \sum_{g\in G} a_g\ g & \longmapsto & \sum_{\bar{g}\in G/H} (\sum_{h\in H} a_{gh})\bar{g}. \end{cases}$$

From a coding theoretic point of view, this operation is nothing but summing up the entries of a codeword whose index are in a same orbit under the action of H. This operation sends a code of length $(c+1)|G|$ and dimension $c|G|$ onto a code of length $(c+1)|G/H|$ and dimension $c|G/H|$. Moreover, a noisy codeword $c + e$ is sent onto $\pi_H(c) + \pi_H(e)$ and the weight of $\pi_H(e)$ is bounded from above by the weight of e. In summary, the length and dimensions of the code are divided by $|H|$ while the weight of the error is preserved or slightly reduced since some entries of e may sum up to 0.

Such projections seem optimal in terms of limiting the growth of the noise.

Remark 28. From the ring theoretic point of view, the map π_H can be regarded as a quotient map of $\mathcal{R} = \mathbb{F}_q[G]$ modulo the ideal generated by all the elements $(h - e_G)$ where $h \in H$ and e_G denotes the unit element of the group G.

Example 29. Following the spirit of [BCG$^+$20b] consider the case

$$\mathcal{R} = \mathbb{F}_q[X]/(X^{q-1} - 1) \simeq \mathbb{F}_q[\mathbb{Z}/(q-1)\mathbb{Z}].$$

In this situation, for any $\ell|(q-1)$, one can consider the subgroup $H = \ell\mathbb{Z}/(q-1)\mathbb{Z}$. The corresponding projection can be made explicit as

$$\pi_H : \begin{cases} \mathbb{F}_q[X]/(X^{q-1} - 1) & \longrightarrow & \mathbb{F}_q[X]/(X^{\frac{q-1}{\ell}} - 1) \\ \sum_{i=0}^{q-2} a_i X^i & \longmapsto & \sum_{i=0}^{\frac{q-1}{\ell}-1} \left(\sum_{j\equiv i \bmod \ell} a_j\right) X^i. \end{cases}$$

In short, we sum up the entries of the codeword whose indexes are congruent modulo ℓ.

Example 30. This example is in the spirit of the attacks on multivariate Ring-LWE [BCV20]. Consider the ring $\mathcal{R} = \mathbb{F}_q[\mathbb{Z}/n\mathbb{Z} \times \mathbb{Z}/n\mathbb{Z}] \simeq \mathbb{F}_q[X, Y]/(X^n - 1, Y^n - 1)$ and consider the subgroup

$$H \stackrel{\text{def}}{=} \{(x, x) \mid x \in \mathbb{Z}/n\mathbb{Z}\} \subseteq G = \mathbb{Z}/n\mathbb{Z} \times \mathbb{Z}/n\mathbb{Z}.$$

Here the projection map can be made explicit as

$$\pi_H : \begin{cases} \mathbb{F}_q[X, Y]/(X^n - 1, Y^n - 1) & \longrightarrow & \mathbb{F}_q[X]/(X^n - 1) \\ \sum_{i,j=0}^{n-1} a_{ij} X^i Y^j & \longmapsto & \sum_{i=0}^{n-1} \left(\sum_{u+v\equiv i \bmod n} a_{uv}\right) X^i. \end{cases} \tag{1}$$

This approach is considered in [BCV20] to provide an attack on multivariate Ring-LWE. In the coding theoretic context, this approach is analysed in depth in [CT19] where the projection map is called *folding*.

Computing the new noise weight. Following [BCG+20b], we consider an instance $(a, ae + f)$ where each sparse vector e, f has been sampled as a sum of $t/2$ random monomials. This distribution is very close to the original distribution, and its choice significantly simplifies the analysis. It also slightly favor the attacker (since the expected number of noisy entries will now be slightly below t due to possible collisions). In this setting, the expected noise rate t' can be computed fairly simply. Let $R_{m,\ell}$ be the random variable counting the number of nonzero coefficients in a polynomial with m coefficients over $\mathbb{F}_q$ computed as the sum of ℓ random monomials. Note that $t' = \mathbb{E}[R_{n,t}]$, where n is the code length. Then, we have

$$\mathbb{E}[R_{m,\ell+1}] = \left(1 - \frac{\mathbb{E}[R_{m,\ell}]}{m}\right) \cdot (\mathbb{E}[R_{m,\ell}] + 1) + \frac{\mathbb{E}[R_{m,\ell}]}{m} \cdot \left(\mathbb{E}[R_{m,\ell}] - \frac{1}{q-1}\right),$$

since adding a new random monomial increases the number of nonzero coefficients by 1 if it falls in a position with a zero coefficient, and decreases the expected number of nonzero coefficients by $1/(q-1)$ otherwise (since this is the probability, when summing two random elements of $\mathbb{F}_q^*$, to get 0). Solving the recurrence relation gives

$$t' = \frac{n \cdot (q-1)}{q} \cdot \left(1 - \left(1 - \frac{q}{n \cdot (q-1)}\right)^{\ell}\right).$$

In the rest of the analysis, we will cover standard attacks on syndrome decoding on instances of a given noise rate and dimension. Then, when choosing concrete parameters, we will estimate the attacker cost as the minimum cost of solving any instance obtained by reducing $\mathbb{F}_q[G]$ to $\mathbb{F}_q[G/H]$, estimating the reduced noise parameter t' using the formula above. We note that this approach ignores the possibility that for a given instance, t' ends up being much smaller than its expected value, which would yield some weak instances of the problem. As in [BCG+20b], we observe that this can be avoided by changing the structure of the noise using rejection sampling: one can re-sample the noise vectors until the weight t' of the reduced instance over $\mathbb{F}_q[G/H]$ (using the best possible choice of $|H|$ for the attacker with the attacks covered below) is at least its expected value (on average, since the probability of having $\mathbb{E}[t'] \leqslant t'$ is $1/2$, this reduces by at most a single bit the entropy of the noise vector).

6.2 Information Set Decoding

In this section, we cover standard linear attacks against syndrome decoding. The most advanced attacks in this category are the information set decoding (ISD) attacks, initially introduced by Prange [Pra62] and subsequently refined in a long sequence of works [Ste88, FS09, BLP11, MMT11, BJMM12, MO15]. Evaluating precisely the effect of each attack on a given instance is complex and tedious, but a general lower bound on the attack cost was derived in [HOSS18], based on similar analysis given in [FS09, Sen11, HS13, TS16]. These lower bounds build upon the common structure of most ISD variants. In general, the cost of modern ISD algorithms for a code with parity-check matrix $\mathbf{H}$ over $\mathbb{F}_2$, with dimension k, code length n, and t noisy coordinates, is lower bounded by

$$\min_{p_1,p_2} \left\{ \frac{\min\left\{2^k, \binom{n}{t}\right\}}{\binom{k-p_2}{t-p_1}} \cdot \left(\frac{K_1 + K_2}{\binom{k+p_2}{p_1}} + \frac{t \cdot (k-p_2)}{2^{p_2}} \right) \right\},$$

where (p_1, p_2) satisfy $0 \leqslant p_2 \leqslant k/2$ and $0 \leqslant p_1 \leqslant k + p_2$, K_1 denotes the cost of Gaussian elimination on a submatrix of $\mathbf{H}$ with $n - p_2$ columns, and K_2 denotes the running time of a specific sub-algorithm, which varies accross different attacks. As in [BCG+20b], we assume that performing Gaussian elimination on the submatrix of $\mathbf{H}$ can be done in time $K_1 \approx (k-p_2)^2 \log(k-p_2)$, because $\mathbf{H}$ is a structured matrix. According to the analysis of [HOSS18], K_2 can be lower bounded by $K_2 \geqslant \binom{(k+p_2)/2}{p_1/8}$ for algorithm of [BJMM12]. As in [BCG+20b], [BJMM12] seems to provide the best efficiency in our setting (more recent algorithms have large hidden constants that render them less practical, or improve over [BJMM12] only for very high noise rates).

The above analysis is restricted to the case of $\mathbb{F}_2$, which is the easiest to attack using ISD. Over larger fields, one can use the above costs as a lower bound for the true cost of the attack, but as the field size grows, this lower bound becomes quite loose. Indeed, this bound was used to pick

concrete parameters in [BCG+20b], but a recent preprint [LWYY22] estimates that the parameters recommended in [BCG+20b] for 80 bits of security actually achieve 92-112 bits of security, while the parameters recommended for 128 bits of security actually achieve 133-171 bits of security. In our setting, however, our PCG's can be instantiated over fields as small as $\mathbb{F}_3$, in which the costs should be much closer to the lower bounds used in [BCG+20b].

We note that a detailed analysis of ISD over larger fields was given in a recent paper [BCDL19]. However, for the sake of avoiding to compute different QA-SD parameters for each possible field size $\mathbb{F}_q$, we stick in this paper to the conservative lower bound that stems from the analysis over $\mathbb{F}_2$.

In [CT19] a study of the combination of ISD with the *folding* operation is studied and precises how the use of folding improves the complexity of the decoder. It turns out that for small errors rates, which is precisely our setting, the use of folding does not represent a significant improvement.

6.3 Prange and statistical decoding (Low-Weight Parity-Check)

We also consider other standard linear attacks, such as Prange decoding algorithm [Pra62] and low-weight parity checks [Zic17, AJ01, FKI06, Ove06, DT17] which leads to the so-called *statistical decoding*. The former, which is just the original ISD algorithm, consists in guessing k noise-free equations and solving the resulting system. It has the advantage over more recent ISD algorithms that it does not depend on the field size. The latter is also often more efficient than ISD in our setting. This is because ISD is a search attack, and executing the attack involves solving a linear system in each iteration of the attack. Since typical PCG applications have huge dimensions (e.g. $k \approx 2^{30}$), this polynomial cost turns out to have a significant impact on the overall runtime of the attack (even though ISDs have the lowest exponent in the exponential part of the attack). Low-weight parity checks, however, work by directly finding many $\mathbf{v}$ such that $\mathbf{v} \cdot \mathbf{H}$ has low weight, and declare $\mathbf{b}$ to be a syndrome decoding instance if the set $\{\mathbf{v} \cdot \mathbf{b}^\intercal\}$ contains too many zeroes. In other words, these attacks directly target the decision variant of syndrome decoding (on which our PCG's rely) and require computing only an inner product per iteration, rather than solving a large linear system. Concretely, the cost of Prange (when $\mathbf{H}$ is a structured matrix) is given by $O\left(1/(1-\frac{t}{n})^k \cdot k^2 \log k\right)$ arithmetic operations, and the cost of the low-weight parity check attack is $O\left(n/(k-1)^t \cdot k\right)$ arithmetic operations (see [BCGI18, BCG+20b]).

6.4 Algebraic Decoding Attacks

An important line of work in code–based cryptography consists in recovering a hidden algebraic structure of a code which permits to decode. See for instance [Wie10, CGG+14, COT17, CMP17, CLT19]. In general such attacks rest on the fact that the public code $\mathcal{C}$ or some of its subcodes has a peculiar behaviour with respect to the component wise product. Namely that the "square of $\mathcal{C}$", *i.e.* the span of the component wise products of any two words of $\mathcal{C}$ has small dimension compared to the square of a random code.

Note that codes sharing this feature of having a "small square" benefit from an efficient decoding algorithm usually referred to as *Error Locating Pairs decoder* [Pel92]. See [Cou21, Section 4] for further details. Therefore, if a random quasi–group code had a small square compared to random codes, then one could deduce an algebraic decoder for quasi–group codes which is a longstanding open question: even when restricting to the case of cyclic codes!

Algebraic attacks exploit the structure of the underlying code to decode it efficiently. Many such algebraic decoding attacks have been devised in the literature, and fall in a unified framework developed in [Pel92, Kot92] based on componentwise product of codes. Examples of such attacks include [PMMM11, MP12, FGO+13, CGGU+13, MMP14] (and many more), and were often used to break some variants of the McEliece cryptosystem. In our context, though, algebraic decoding of quasi-group codes is a well-known and long-standing open problem: it has been studied for over 50 years in the coding theory community, and to this day no efficient algorithm is known to decode a random quasi-abelian code. This is listed as an open research problem in the most recent Encyclopedia of Coding Theory (from 2021) [Wil21, Problem 16.10.5].

6.5 Attacks on Multivariate LWE

As already mentioned in Example 30, an attack on multivariate Ring-LWE is presented in [BCV20]. This attack is based on a projection of the form $\mathbb{F}_q[G] \to \mathbb{F}_q[G/H]$ as described in Section 6.1. The attack is particularly efficient since applying a map of the form (1) has a very limited impact on the Euclidean norm and hence has a limited impact on the noise term. In the coding theoretic setting, the situation is very different since the Hamming weight of the error is more or less preserved but then the relative weight, *i.e.* the ratio $\frac{t}{n}$ is more or less multiplied by a term $|H|$. Therefore, reducing with respect to a too large subgroup H leads to shorter codes but provides intractable instances of the decoding problem.

6.6 Decoding One-Out-Of Many

For a code equipped with a non trivial permutation group, which is an obvious feature of quasi-abelian codes, the decoding problem can be made easier using Sendrier's *Decoding One Out of Many* (DOOM) paradigm [Sen11]. Indeed, consider a quasi–abelian code $\mathcal{C} \subseteq \mathbb{F}_q[G]^\ell$ and a noisy codeword $y = c + e$ with $c \in \mathcal{C}$ and $e \in \mathbb{F}_q[G]^\ell$ of low weight. Then, for any $g \in G$, we get another instance of the decoding problem with an error term of the same weight:

$$g \cdot y = g \cdot c + g \cdot e.$$

Here $g \cdot c \in \mathcal{C}$ and $\mathrm{wt}(g \cdot e)) = \mathrm{wt}(e))$. Therefore, given a single instance of QA-SD we naturally deduce $|G|$ instances and solving one of them immediately solves the other ones. Thus, from [Sen11], solving one out of $|G|$ instances of SD permits to divide the work factor of any decoder by $\sqrt{|G|}$. Therefore the cost of ISD should be divided by $\sqrt{|G|}$ and the cost of the composition of a projection $\mathbb{F}_q[G] \to \mathbb{F}_q[G/H]$ with ISD should be divided by $\sqrt{|G/H|}$.

7 Applications to Secure Computation

In this part, we explain some of the main applications of our new PCG's to secure computation. To provide bounds, we will use the following restatement of Theorem 25 in the case $\mathcal{R} = \mathbb{F}_q[X_1, .., X_n]/(X_1^{q-1} - 1, .., X_n^{q-1} - 1)$.

Theorem 31. *Suppose that* SPFSS *is a secure* FSS *scheme for sums of point functions and that the* QA-SD *assumption holds. Let* $\mathcal{R} = \mathbb{F}_q[X_1, .., X_n]/(X_1^{q-1} - 1, .., X_n^{q-1} - 1)$*, and* $T = (q-1)^n$*. We can construct a* PCG *producing* T *instances for* OLE *over* $\mathbb{F}_p$*, using the* $\mathsf{QA\text{-}SD}_{\mathsf{OLE}}$ *construction with the following parameters*

- *Communication costs and size of the seed :* $O(\lambda^3 \log T)$.
- *Computation costs :* $O(\lambda T)$ PRG *evaluations and* $O(c^2 T \log T)$ *operations in* $\mathbb{F}_q$.

First, as explained in [BCG+20b], a PCG generating N multiplication triples can be derived from a PCG generating $2N$ OLE.

Extension to multiplication triples. The OLE correlation gives a secret to each party P_0 and P_1 and an additive secret-sharing of the product of the two secrets. The OLE correlation is interesting in its own right and can be used directly in some applications, but in general, the multiplication triple correlation is used. A (2-party) multiplication triple, gives the parties additive shares of random elements a and b, and shares of the product $a \cdot b$. The main advantage of multiplication triples is their usefulness in the setting of 2-party computation of arithmetic circuits over $\mathbb{F}_q$.

In this setting, each multiplication gate can be evaluated by consuming a single multiplication triple, and with communication costs of two $\mathbb{F}_q$ elements per party - the additions are free in this setting. Using two instances of an OLE correlation we can obtain an instance of a multiplication triple correlation. Let $a = a_0 + a_1, b = b_0 + b_1$ and $c = ab = a_0b_0 + a_0b_1 + a_1b_0 + a_1b_1$, we can distribute a_σ, b_σ to party P_σ and run two independent OLE instances to obtain the secret share of the cross terms a_0b_1 and a_1b_0. As the party P_σ can locally compute $a_\sigma b_\sigma$ it gets a correct sharing of ab. Note that we obtain the correlation in a black-box way. Thus, a PCG generating N multiplication triples can be derived from a PCG generating $2N$ OLE.

7.1 Application : (N-party) multiplication triples generation for arithmetic circuit

Theorem 32. *Assume the existence of oblivious transfers and* QA-SD($\mathcal{R}$) *assumption, where* $\mathcal{R} = \mathbb{F}_q[X_1, \cdots, X_n]/(X_1^{q-1} - 1, \cdots, X_n^{q-1} - 1) \simeq \mathbb{F}_q \times \cdots \times \mathbb{F}_q$, *with* $q \geqslant 3$. *Let* $T = (q-1)^n$. *There exists a semi-honest* N*-party protocol for securely evaluating an arithmetic circuit* C *over* $\mathbb{F}_q$ *with* T *multiplication gates, in the preprocessing model, such that:*

- *The preprocessing phase has communication cost* $\tilde{c}(N, \lambda, T) = O(\lambda^3 \cdot N^2 \cdot \log(2T))$, *and computation cost* $\dot{c}(N, \lambda, T) = O(N^2 \cdot \lambda \cdot 2T)$ PRG *calls;* $O(N^2 \cdot 2T \log(2T))$ *operations in* $\mathbb{F}_q$.
- *The online phase is non-cryptographic and communication cost* $2 \cdot N \cdot T$ *elements of* $\mathbb{F}_q$.

Proof. Consider the parties $P_1, \cdots, P_N$. First, remark that programmability enables parties to generate "correlated" (2-party) multiplication triples, which can used to obtain N -party multiplication triples in the following way.

- The party P_i gets two random values (x_i, y_i). We define $X = \sum_i x_i$ and $Y = \sum_j y_j$.
- Each pair of parties $(P_i, P_j)_{1 \leqslant i,j \leqslant N, i \neq j}$ performs the programmable protocol for (2-party) multiplication triples with programmable inputs (x_i, y_j), and obtains shares of $x_i \cdot y_j$. We indicate the share of P_i as $\langle x_i \cdot y_j \rangle_i$.
- Let $K_i = \sum_{j=1}^{N} \langle x_i \cdot y_j \rangle_i + \langle x_i \cdot y_j \rangle_i + x_i \cdot y_i$. The K_i are shares of the product

$$X \cdot Y = \sum_{1 \leqslant i,j \leqslant N} x_i \cdot y_j = \sum_{i=1}^{N} K_i$$

The parties use the QA-SD$_{\mathsf{OLE}}$ to generate short seeds of each of the $N \cdot (N-1)$ (2-party) multiplication triples they need. In the online phase, they locally expand the seeds to obtain T instances of (N-party) multiplication triples. The parties can execute the (N-party) GMW protocol using the multiplication triples, and evaluate the circuit.

Using Theorem 31 we obtain the cost of preprocessing for generating the $2T$ OLE over $\mathbb{F}_p$, namely $O(\lambda^3 \cdot \log(2T))$ in communication cost, and $\dot{c}(N, \lambda, T) = O(N^2 \lambda 2T)$ PRG calls ; $O(\lambda^2 \cdot 2T \log(2T))$ operations in $\mathbb{F}_q$ in computation cost.

The cost of communication in the online phase is simply derived from the GMW algorithm using the multiplication triples. For each multiplication gate, each party must send two field elements, resulting in a cost of $2 \cdot N \cdot T$.

7.2 Secure Computation with Circuit-Dependent Preprocessing

Circuit-dependent preprocessing is a variation of the standard Beaver's circuit randomization technique with multiplication triples. It has been investigated in recent works, such as [DNNR17, Cou19]. The idea is to preprocess multiplications in a way that depends on the structure of the circuit and leads to an online phase that requires just *one opening per multiplication gate*, instead of two when using multiplication triples. PCG's for OLE's do not directly enable reducing the preprocessing phase of secure computation with circuit-dependent correlated randomness: at a high level, this stems from the fact that since the correlated randomness depends on the topology of the circuit, it cannot be compressed beyond the description size of this topology. Nevertheless, PCG's enable *batch* secure computation (*i.e.* securely computing many copies of the same circuit on different input) with silent preprocessing in the circuit-dependent correlated randomness setting, by using PCG's to compress a batch of correlations for a given gate accross all circuits.

Theorem 33. *Assume the existence of oblivious transfer and the* QA-SD($\mathcal{R}$) *assumption, where* $\mathcal{R} = \mathbb{F}_q[X_1, \cdots, X_n]/(X_1^{q-1} - 1, \cdots, X_n^{q-1} - 1) \simeq \mathbb{F}_q \times \cdots \times \mathbb{F}_q$, *with* $q \geqslant 3$. *Let* $T = (q-1)^n$. *There exists a semi-honest 2-party protocol for securely evaluating* T *copies of an arithmetic circuit* C *over* $\mathbb{F}$ *with* S *multiplication gates, in the preprocessing model, such that:*

- *The preprocessing phase has communication cost* $c(T, \lambda, S) = O(\lambda^3 \cdot S \cdot \log(2T))$ *and a computation cost* $\dot{c}(T, \lambda, S) = O(\lambda \cdot S \cdot 2T)$ PRG *calls ;* $O(S \cdot 2T \log(2T))$ *operations in* $\mathbb{F}_q$.
- *The onTine phase is non-cryptographic and communication costs* $2 \cdot S \cdot T$ *elements of* $\mathbb{F}$.

Proof. Let C be an arithmetic circuit over $\mathbb{F}$ consisting of fan-in two addition and multiplication gates. Each wire w is assigned a mask r_w during the offline phase. The masks are designed as follows.

- if w is an input wire, r_w is chosen at random.
- if w is the output wire of a multiplication gate, $r_w \leftarrow \mathbb{F}$ is chosen at random
- if w is the output wire of an addition gate with input wires u and v, then $r_w = r_u + r_v$.
- for each multiplication gate, we assigned a value $s_{u,v}$, such that on input wires u and v, $s_{u,v} = r_u \cdot r_v$.

The masks are not known by the parties, but they obtain random additive shares of each r_w for any input and output wire of multiplication gates, as well as $s_{u,v}$ for the multiplication gates.

When the online phase begins, both parties hide their secret values with random masks. The party that is not the one giving the input for a given wire w gives to the other one his shares $\langle r_w \rangle$. The invariant of the online phase is that through the protocol, for each wire, parties know exactly the value $x + r_x$ where r_x is the mask of this wire (for which parties have additive sharing), and x is the real value that is computed by the circuit passing through the wire w. The invariant is preserved through each gate because of the following:

- For an addition gate, parties know $x + r_x$ and $y + r_y$. Then the parties add locally those values to obtain $x + y + r_x + r_y$, with $r_x + r_y$ being indeed the output mask for the addition gate.
- For a multiplication gate with r_w denoting its output wire's mask, parties know $x + r_x$ and $y + r_y$. The parties can locally compute their share $\langle (x + r_x) \cdot r_y + (y + r_y) \cdot r_x + r_x \cdot r_y + r_w \rangle$ (the formula can be a little bit different if we are not in $\mathbb{F}_2$). By both exchanging one bit of information, they reconstitute that value. Adding up $(x + r_x) \cdot (y + r_y)$, they obtain in clear $x \cdot y + r_w$ where r_w is the mask of the output wire of this multiplication gate.

In the end, we have to perform $2S$ different calls to our PCG to create the (2-party) multiplication triples seeds. Again we use Theorem 31 to get the estimation of the costs in communication and space, per instances, and we multiply it by S. In the online phase, we gain a factor 2 in communication because each party only has to send a bit of information for each of the multiplication gates. As there are $S \cdot T$ multiplication gates in total, the communication cost in the online phase is $2 \cdot S \cdot T$. □

References

AAB+22a. Carlos Aguilar Melchor, Nicolas Aragon, Paulo Barreto, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Santosh Ghosh, Shay Gueron, Tim Güneysu, Rafael Misoczki, Edoardo Persichetti, Jan Richter-Brockmann, Nicolas Sendrier, Jean-Pierre Tillich, Valentin Vasseur, and Gilles Zémor. BIKE. Round 4 Submission to the NIST Post-Quantum Cryptography Call, v. 5.1, October 2022.

AAB+22b. Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jurjen Bos, Jean-Christophe Deneuville, Arnaud Dion, Philippe Gaborit, Jérôme Lacan, Edoardo Persichetti, Jean-Marc Robert, Pascal Véron, Gilles Zémor, and Jurjen Bos. HQC. Round 4 Submission to the NIST Post-Quantum Cryptography Call, October 2022. `https://pqc-hqc.org/`.

ABB+17. Nicolas Aragon, Paulo Barreto, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Shay Gueron, Tim Guneysu, Carlos Aguilar Melchor, et al. Bike: Bit flipping key encapsulation. 2017.

ABG+14. Adi Akavia, Andrej Bogdanov, Siyao Guo, Akshay Kamath, and Alon Rosen. Candidate weak pseudorandom functions in AC^0 *o* MOD_2. In Moni Naor, editor, *ITCS 2014: 5th Conference on Innovations in Theoretical Computer Science*, pages 251–260, Princeton, NJ, USA, January 12–14, 2014. Association for Computing Machinery.

AG11. Sanjeev Arora and Rong Ge. New algorithms for learning in presence of errors. In *International Colloquium on Automata, Languages, and Programming*, pages 403–415. Springer, 2011.

AJ01. Abdulrahman Al Jabri. A statistical decoding algorithm for general linear block codes. In *IMA International Conference on Cryptography and Coding*, pages 1–8. Springer, 2001.

Ale03. Michael Alekhnovich. More on average case vs approximation complexity. In *44th Annual Symposium on Foundations of Computer Science*, pages 298–307, Cambridge, MA, USA, October 11–14, 2003. IEEE Computer Society Press.

AMBD+18. Carlos Aguilar-Melchor, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, and Gilles Zémor. Efficient encryption from random quasi-cyclic codes. *IEEE Transactions on Information Theory*, 64(5):3927–3943, 2018.

BCD22. Maxime Bombar, Alain Couvreur, and Thomas Debris-Alazard. On codes and learning with errors over function fields. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022, Part II*, volume 13508 of *Lecture Notes in Computer Science*, pages 513–540, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Heidelberg, Germany.

BCDL19. Rémi Bricout, André Chailloux, Thomas Debris-Alazard, and Matthieu Lequesne. Ternary syndrome decoding with large weight. In Kenneth G. Paterson and Douglas Stebila, editors, *SAC 2019: 26th Annual International Workshop on Selected Areas in Cryptography*, volume 11959 of *Lecture Notes in Computer Science*, pages 437–466, Waterloo, ON, Canada, August 12–16, 2019. Springer, Heidelberg, Germany.

BCG+17. Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, and Michele Orrù. Homomorphic secret sharing: Optimizations and applications. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017: 24th Conference on Computer and Communications Security*, pages 2105–2122, Dallas, TX, USA, October 31 – November 2, 2017. ACM Press.

BCG+19a. Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Peter Rindal, and Peter Scholl. Efficient two-round OT extension and silent non-interactive secure computation. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019: 26th Conference on Computer and Communications Security*, pages 291–308, London, UK, November 11–15, 2019. ACM Press.

BCG+19b. Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators: Silent OT extension and more. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 489–518, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany.

BCG+20a. Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Correlated pseudorandom functions from variable-density LPN. In *61st Annual Symposium on Foundations of Computer Science*, pages 1069–1080, Durham, NC, USA, November 16–19, 2020. IEEE Computer Society Press.

BCG+20b. Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators from ring-LPN. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020, Part II*, volume 12171 of *Lecture Notes in Computer Science*, pages 387–416, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Heidelberg, Germany.

BCG+22. Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Nicolas Resch, and Peter Scholl. Correlated pseudorandomness from expand-accumulate codes. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022, Part II*, volume 13508 of *Lecture Notes in Computer Science*, pages 603–633, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Heidelberg, Germany.

BCGI18. Elette Boyle, Geoffroy Couteau, Niv Gilboa, and Yuval Ishai. Compressing vector OLE. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018: 25th Conference on Computer and Communications Security*, pages 896–912, Toronto, ON, Canada, October 15–19, 2018. ACM Press.

BCV20. Carl Bootland, Wouter Castryck, and Frederik Vercauteren. On the security of the multivariate ring learning with errors problem. In *ANTS-XIV, Fourteenth Algorithmic Number Theory Symposium, Proceedings*, volume 4 of *Open Book Series*, pages 57–71. Mathematical Sciences Publishers, 2020.

Bea92. Donald Beaver. Efficient multiparty protocols using circuit randomization. In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 420–432, Santa Barbara, CA, USA, August 11–15, 1992. Springer, Heidelberg, Germany.

BEPU+20. Carsten Baum, Daniel Escudero, Alberto Pedrouzo-Ulloa, Peter Scholl, and Juan Ramón Troncoso-Pastoriza. Efficient protocols for oblivious linear function evaluation from ring-LWE. In Clemente Galdi and Vladimir Kolesnikov, editors, *SCN 20: 12th International Conference on Security in Communication Networks*, volume 12238 of *Lecture Notes in Computer Science*, pages 130–149, Amalfi, Italy, September 14–16, 2020. Springer, Heidelberg, Germany.

BF02. Alexander Barg and G. David Forney. Random codes: Minimum distances and error exponents. *IEEE Trans. Inf. Theory*, 48(9):2568–2573, 2002.

BGI15. Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part II*, volume

9057 of *Lecture Notes in Computer Science*, pages 337–367, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany.

BGI16. Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing: Improvements and extensions. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016: 23rd Conference on Computer and Communications Security*, pages 1292–1303, Vienna, Austria, October 24–28, 2016. ACM Press.

BJMM12. Anja Becker, Antoine Joux, Alexander May, and Alexander Meurer. Decoding random binary linear codes in $2^{n/20}$: How $1 + 1 = 0$ improves information set decoding. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 520–536, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany.

BKW00. Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. In *32nd Annual ACM Symposium on Theory of Computing*, pages 435–440, Portland, OR, USA, May 21–23, 2000. ACM Press.

BL12. Daniel J Bernstein and Tanja Lange. Never trust a bunny. In *International Workshop on Radio Frequency Identification: Security and Privacy Issues*, pages 137–148. Springer, 2012.

BLP11. Daniel J. Bernstein, Tanja Lange, and Christiane Peters. Smaller decoding exponents: Ball-collision decoding. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 743–760, Santa Barbara, CA, USA, August 14–18, 2011. Springer, Heidelberg, Germany.

BM97. Mihir Bellare and Daniele Micciancio. A new paradigm for collision-free hashing: Incrementality at reduced cost. In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 163–192, Konstanz, Germany, May 11–15, 1997. Springer, Heidelberg, Germany.

BM18. Leif Both and Alexander May. Decoding linear codes with high error rate and its impact for LPN security. In Tanja Lange and Rainer Steinwandt, editors, *Post-Quantum Cryptography - 9th International Conference, PQCrypto 2018*, pages 25–46, Fort Lauderdale, Florida, United States, April 9–11, 2018. Springer, Heidelberg, Germany.

BØ23. Pierre Briaud and Morten Øygarden. A new algebraic approach to the regular syndrome decoding problem and implications for PCG constructions. Cryptology ePrint Archive, Paper 2023/176, 2023. `https://eprint.iacr.org/2023/176`.

BR17. Andrej Bogdanov and Alon Rosen. Pseudorandom functions: Three decades later. Cryptology ePrint Archive, Report 2017/652, 2017. `https://eprint.iacr.org/2017/652`.

BTV16. Sonia Bogos, Florian Tramer, and Serge Vaudenay. On solving lpn using bkw and variants. *Cryptography and Communications*, 8(3):331–369, 2016.

BV16. Sonia Bogos and Serge Vaudenay. Optimization of LPN solving algorithms. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology – ASIACRYPT 2016, Part I*, volume 10031 of *Lecture Notes in Computer Science*, pages 703–728, Hanoi, Vietnam, December 4–8, 2016. Springer, Heidelberg, Germany.

CD23. Geoffroy Couteau and Clément Ducros. Pseudorandom correlation functions from variable-density LPN, revisited. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *PKC 2023: 26th International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 13941 of *Lecture Notes in Computer Science*, pages 221–250, Atlanta, GA, USA, May 7–10, 2023. Springer, Heidelberg, Germany.

CDMT23. Kévin Carrier, Thomas Debris-Alazard, Charles Meyer-Hilfiger, and Jean-Pierre Tillich. Statistical decoding 2.0: Reducing decoding to lpn. In *Advances in Cryptology–ASIACRYPT 2022: 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5–9, 2022, Proceedings, Part IV*, pages 477–507. Springer, 2023.

CGG+14. Alain Couvreur, Philippe Gaborit, Valérie Gauthier-Umaña, Ayoub Otmani, and Jean-Pierre Tillich. Distinguisher-based attacks on public-key cryptosystems using Reed-Solomon codes. *Des. Codes Cryptogr.*, 73(2):641–666, 2014.

CGGU+13. Alain Couvreur, Philippe Gaborit, Valérie Gauthier-Umana, Ayoub Otmani, and Jean-Pierre Tillich. Distinguisher-based attacks on public-key cryptosystems using reed-solomon codes. *arXiv preprint arXiv:1307.6458*, 2013.

CL17. Sunil K. Chebolu and Keir Lockridge. Fuchs' problem for dihedral groups. *Journal of Pure and Applied Algebra*, 221(4):971–982, 2017.

CLT19. Alain Couvreur, Matthieu Lequesne, and Jean-Pierre Tillich. Recovering short secret keys of RLCE in polynomial time. In Jintai Ding and Rainer Steinwandt, editors, *Post-Quantum Cryptography 2019*, volume 11505 of *LNCS*, pages 133–152, Chongquing, China, May 2019. Springer.

CMP17. Alain Couvreur, Irene Márquez–Corbella, and Ruud Pellikaan. Cryptanalysis of McEliece cryptosystem based on algebraic geometry codes and their subcodes. *IEEE Trans. Inform. Theory*, 63(8):5404–5418, 8 2017.

Con. Keith Conrad. Carlitz extensions. `https://kconrad.math.uconn.edu/blurbs/gradnumthy/carlitz.pdf`.

COT17. Alain Couvreur, Ayoub Otmani, and Jean-Pierre Tillich. Polynomial time attack on wild McEliece over quadratic extensions. *IEEE Trans. Inform. Theory*, 63(1):404–427, 1 2017.

Cou19. Geoffroy Couteau. A note on the communication complexity of multiparty computation in the correlated randomness model. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part II*, volume 11477 of *Lecture Notes in Computer Science*, pages 473–503, Darmstadt, Germany, May 19–23, 2019. Springer, Heidelberg, Germany.

Cou21. Alain Couvreur. How arithmetic and geometry make error correcting codes better. Preprint, October 2021.

Cox21. Nicholas Coxon. Fast transforms over finite fields of characteristic two. *J. Symb. Comput.*, 104:824–854, 2021.

CRR21. Geoffroy Couteau, Peter Rindal, and Srinivasan Raghuraman. Silver: Silent VOLE and oblivious transfer from hardness of decoding structured LDPC codes. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part III*, volume 12827 of *Lecture Notes in Computer Science*, pages 502–534, Virtual Event, August 16–20, 2021. Springer, Heidelberg, Germany.

CT65. James W. Cooley and John W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation*, 19:297–301, 1965. URL: `http://cr.yp.to/bib/entries.html#1965/cooley`.

CT19. Rodolfo Canto-Torres and Jean-Pierre Tillich. Speeding up decoding a code with a non-trivial automorphism group up to an exponential factor. In *Proc. IEEE Int. Symposium Inf. Theory - ISIT 2019*, pages 1927–1931, 2019.

Deb23. Thomas Debris-Alazard. Code-based cryptography: Lecture notes, 2023. `https://arxiv.org/abs/2304.03541`.

DNNR17. Ivan Damgård, Jesper Buus Nielsen, Michael Nielsen, and Samuel Ranellucci. The TinyTable protocol for 2-party secure computation, or: Gate-scrambling revisited. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 167–187, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany.

DP12. Ivan Damgård and Sunoo Park. How practical is public-key encryption based on LPN and ring-LPN? Cryptology ePrint Archive, Report 2012/699, 2012. `https://eprint.iacr.org/2012/699`.

DPSZ12. Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 643–662, Santa Barbara, CA, USA, August 19–23, 2012. Springer, Heidelberg, Germany.

DT17. Thomas Debris-Alazard and Jean-Pierre Tillich. Statistical decoding. In *2017 IEEE International Symposium on Information Theory (ISIT)*, pages 1798–1802. IEEE, 2017.

EKM17. Andre Esser, Robert Kübler, and Alexander May. LPN decoded. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part II*, volume 10402 of *Lecture Notes in Computer Science*, pages 486–514, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany.

FGO$^+$13. Jean-Charles Faugere, Valérie Gauthier-Umana, Ayoub Otmani, Ludovic Perret, and Jean-Pierre Tillich. A distinguisher for high-rate mceliece cryptosystems. *IEEE Transactions on Information Theory*, 59(10):6830–6844, 2013.

FKI06. Marc PC Fossorier, Kazukuni Kobara, and Hideki Imai. Modeling bit flipping decoding based on nonorthogonal check sums with application to iterative decoding attack of mceliece cryptosystem. *IEEE Transactions on Information Theory*, 53(1):402–411, 2006.

FL15. Yun Fan and Liren Lin. Thresholds of random quasi-abelian codes. *IEEE Transactions on Information Theory*, 61(1):82–90, 2015.

FS09. Matthieu Finiasz and Nicolas Sendrier. Security bounds for the design of code-based cryptosystems. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 88–105, Tokyo, Japan, December 6–10, 2009. Springer, Heidelberg, Germany.

GI14. Niv Gilboa and Yuval Ishai. Distributed point functions and their applications. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 640–658, Copenhagen, Denmark, May 11–15, 2014. Springer, Heidelberg, Germany.

Gil99. Niv Gilboa. Two party RSA key generation. In Michael J. Wiener, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 116–129, Santa Barbara, CA, USA, August 15–19, 1999. Springer, Heidelberg, Germany.

GJL15. Qian Guo, Thomas Johansson, and Carl Löndahl. A new algorithm for solving ring-lpn with a reducible polynomial. *IEEE Transactions on Information Theory*, 61(11):6204–6212, 2015.

GJL20. Qian Guo, Thomas Johansson, and Carl Löndahl. Solving LPN using covering codes. *Journal of Cryptology*, 33(1):1–33, January 2020.

GRS08. Henri Gilbert, Matthew J. B. Robshaw, and Yannick Seurin. Good variants of HB+ are hard to find. In Gene Tsudik, editor, *FC 2008: 12th International Conference on Financial Cryptography and Data Security*, volume 5143 of *Lecture Notes in Computer Science*, pages 156–170, Cozumel, Mexico, January 28–31, 2008. Springer, Heidelberg, Germany.

GZ06. Philippe Gaborit and Gilles Zémor. Asymptotic improvement of the Gilbert-Varshamov bound for linear codes. In *Proc. IEEE Int. Symposium Inf. Theory - ISIT 2006*, pages 287–291, Seattle, USA, June 2006.

Hay74. David R Hayes. Explicit class field theory for rational function fields. *Transactions of the American Mathematical Society*, 189:77–91, 1974.

HIMV19. Carmit Hazay, Yuval Ishai, Antonio Marcedone, and Muthuramakrishnan Venkitasubramaniam. LevioSA: Lightweight secure arithmetic computation. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019: 26th Conference on Computer and Communications Security*, pages 327–344, London, UK, November 11–15, 2019. ACM Press.

HKL+12. Stefan Heyse, Eike Kiltz, Vadim Lyubashevsky, Christof Paar, and Krzysztof Pietrzak. Lapin: An efficient authentication protocol based on ring-LPN. In Anne Canteaut, editor, *Fast Software Encryption – FSE 2012*, volume 7549 of *Lecture Notes in Computer Science*, pages 346–365, Washington, DC, USA, March 19–21, 2012. Springer, Heidelberg, Germany.

HOSS18. Carmit Hazay, Emmanuela Orsini, Peter Scholl, and Eduardo Soria-Vazquez. TinyKeys: A new approach to efficient multi-party computation. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part III*, volume 10993 of *Lecture Notes in Computer Science*, pages 3–33, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Heidelberg, Germany.

HS13. Yann Hamdaoui and Nicolas Sendrier. A non asymptotic analysis of information set decoding. *IACR Cryptology ePrint Archive*, 2013:162, 2013.

HVB16. Yi Hong, Emanuele Viterbo, and Jean-Claude Belfiore. The two-modular fourier transform of binary functions. *IEEE Transactions on Information Theory*, 62(5):2813–2826, 2016.

IKNP03. Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 145–161, Santa Barbara, CA, USA, August 17–21, 2003. Springer, Heidelberg, Germany.

Kas74. T. Kasami. A gilbert-varshamov bound for quasi-cycle codes of rate 1/2 (corresp.). *IEEE Transactions on Information Theory*, 20(5):679–679, 1974.

Kir11. Paul Kirchner. Improved generalized birthday attack. Cryptology ePrint Archive, Report 2011/377, 2011. `https://eprint.iacr.org/2011/377`.

Kot92. Ralf Kotter. An unified description of an error locating procedure for linear codes. *Proc. IAACCT, Voneshta Voda, Bulgaria*, 1992.

KPR18. Marcel Keller, Valerio Pastro, and Dragos Rotaru. Overdrive: Making SPDZ great again. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part III*, volume 10822 of *Lecture Notes in Computer Science*, pages 158–189, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany.

LCK+18. Wen-Ding Li, Ming-Shing Chen, Po-Chun Kuo, Chen-Mou Cheng, and Bo-Yin Yang. Frobenius additive fast fourier transform. In Manuel Kauers, Alexey Ovchinnikov, and Éric Schost, editors, *Proceedings of the 2018 ACM on International Symposium on Symbolic and Algebraic Computation, ISSAC 2018, New York, NY, USA, July 16-19, 2018*, pages 263–270. ACM, 2018.

LF06. Éric Levieil and Pierre-Alain Fouque. An improved LPN algorithm. In Roberto De Prisco and Moti Yung, editors, *SCN 06: 5th International Conference on Security in Communication Networks*, volume 4116 of *Lecture Notes in Computer Science*, pages 348–359, Maiori, Italy, September 6–8, 2006. Springer, Heidelberg, Germany.

LHW18. Runzhou Li, Qin Huang, and Zulin Wang. Encoding of non-binary quasi-cyclic codes by lin-chung-han transform. In *2018 IEEE Information Theory Workshop (ITW)*, pages 1–5, 2018.

LM13. Vadim Lyubashevsky and Daniel Masny. Man-in-the-middle secure authentication schemes from LPN and weak PRFs. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology*

– *CRYPTO 2013, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 308–325, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany.

Lor21. Dino Lorenzini. *An invitation to arithmetic geometry*, volume 9. American Mathematical Society, 2021.

LP15. Helger Lipmaa and Kateryna Pavlyk. Analysis and implementation of an efficient ring-LPN based commitment scheme. In Michael Reiter and David Naccache, editors, *CANS 15: 14th International Conference on Cryptology and Network Security*, Lecture Notes in Computer Science, pages 160–175, Marrakesh, Morocco, December 10–12, 2015. Springer, Heidelberg, Germany.

LPR10. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23, French Riviera, May 30 – June 3, 2010. Springer, Heidelberg, Germany.

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

Lyu05. Vadim Lyubashevsky. The parity problem in the presence of noise, decoding random linear codes, and the subset sum problem. In *Approximation, randomization and combinatorial optimization. Algorithms and techniques*, pages 378–389. Springer, 2005.

MAB+18. Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loıc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Edoardo Persichetti, Gilles Zémor, and IC Bourges. Hamming quasi-cyclic (hqc). *NIST PQC Round*, 2:4–13, 2018.

MMP14. Irene Márquez-Corbella, Edgar Martínez-Moro, and Ruud Pellikaan. On the unique representation of very strong algebraic geometry codes. *Designs, Codes and Cryptography*, 70(1-2):215–230, 2014.

MMT11. Alexander May, Alexander Meurer, and Enrico Thomae. Decoding random linear codes in $\tilde{\mathcal{O}}(2^{0.054n})$. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 107–124, Seoul, South Korea, December 4–8, 2011. Springer, Heidelberg, Germany.

MO15. Alexander May and Ilya Ozerov. On computing nearest neighbors with applications to decoding of binary linear codes. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 203–228, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany.

MP12. Irene Márquez-Corbella and Ruud Pellikaan. Error-correcting pairs for a public-key cryptosystem. *arXiv preprint arXiv:1205.3647*, 2012.

Noe32. Emmy Noether. Normalbasis bei Körpern ohne Höhere Verzweigung. *J. Reine Angew. Math.*, 167:147–152, 1932.

Obe07. Ulrich Oberst. The fast fourier transform. *SIAM Journal on Control and Optimization*, 46(2):496–540, 2007.

Ove06. Raphael Overbeck. Statistical decoding revisited. In Lynn Margaret Batten and Reihaneh Safavi-Naini, editors, *ACISP 06: 11th Australasian Conference on Information Security and Privacy*, volume 4058 of *Lecture Notes in Computer Science*, pages 283–294, Melbourne, Australia, July 3–5, 2006. Springer, Heidelberg, Germany.

Pel92. Ruud Pellikaan. On decoding by error location and dependent sets of error positions. *Discrete Math.*, 106–107:368–381, 1992.

Pie67. John N. Pierce. Limit distribution of the minimum distance of random linear codes. *IEEE Trans. Inform. Theory*, 13(1):595–599, 1967.

PMMM11. Ruud Pellikaan, Irene Márquez-Corbella, and Edgar Martínez-Moro. Evaluation of public-key cryptosystems based on algebraic geometry codes. In *Third International Castle Meeting on Coding Theory and Applications (3ICMTA, Cardona Castle, Barcelona, Spain, pages 199–204*, 2011.

Pra62. Eugene Prange. The use of information sets in decoding cyclic codes. *IRE Transactions on Information Theory*, 8(5):5–9, 1962.

Ros02. Michael Rosen. *Number Theory in Function Fields*. Graduate Texts in Mathematics. Springer, 2002.

Saa07. Markku-Juhani Olavi Saarinen. Linearization attacks against syndrome based hashes. In K. Srinathan, C. Pandu Rangan, and Moti Yung, editors, *Progress in Cryptology - INDOCRYPT 2007: 8th International Conference in Cryptology in India*, volume 4859 of *Lecture Notes in Computer Science*, pages 1–9, Chennai, India, December 9–13, 2007. Springer, Heidelberg, Germany.

Sen11. Nicolas Sendrier. Decoding one out of many. In *Post-Quantum Cryptography 2011*, volume 7071 of *LNCS*, pages 51–67, 2011.

SGRR19. Phillipp Schoppmann, Adrià Gascón, Leonie Reichert, and Mariana Raykova. Distributed vector-OLE: Improved constructions and implementation. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019: 26th Conference on Computer and Communications Security*, pages 1055–1072, London, UK, November 11–15, 2019. ACM Press.

Ste88. Jacques Stern. A method for finding codewords of small weight. In *International Colloquium on Coding Theory and Applications*, pages 106–113. Springer, 1988.

Sti09. Henning Stichtenoth. *Algebraic function fields and codes*, volume 254 of *Graduate Texts in Mathematics*. Springer-Verlag, Berlin, second edition, 2009.

TS16. Rodolfo Canto Torres and Nicolas Sendrier. Analysis of information set decoding for a sub-linear error weight. In *International Workshop on Post-Quantum Cryptography*, pages 144–161. Springer, 2016.

Var97. Alexander Vardy. The intractability of computing the minimum distance of a code. *IEEE Trans. Inform. Theory*, 43(6):1757–1766, November 1997.

vdHLS13. Joris van der Hoeven, Romain Lebreton, and Éric Schost. Structured FFT and TFT: symmetric and lattice polynomials. In Manuel Kauers, editor, *International Symposium on Symbolic and Algebraic Computation, ISSAC'13, Boston, MA, USA, June 26-29, 2013*, pages 355–362. ACM, 2013.

Vil06. Gabriel Daniel Villa Salvador. *Topics in the Theory of Algebraic Function Fields*. Springer, 2006.

Wag02. David Wagner. A generalized birthday problem. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 288–303, Santa Barbara, CA, USA, August 18–22, 2002. Springer, Heidelberg, Germany.

Was77. Siri Krishan Wasan. Quasi-abelian codes. 1977.

Wie10. Christian Wieschebrink. Cryptanalysis of the Niederreiter public key scheme based on GRS subcodes. In *Post-Quantum Cryptography 2010*, volume 6061 of *LNCS*, pages 61–72. Springer, 2010.

Wil21. Wolfgang Willems. *Codes in group algebras*, chapter 16. Chapman and Hall/CRC, 2021.

YWL+20. Kang Yang, Chenkai Weng, Xiao Lan, Jiang Zhang, and Xiao Wang. Ferret: Fast extension for correlated OT with small communication. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020: 27th Conference on Computer and Communications Security*, pages 1607–1626, Virtual Event, USA, November 9–13, 2020. ACM Press.

Zic17. Lior Zichron. Locally computable arithmetic pseudorandom generators. Master's thesis, School of Electrical Engineering, Tel Aviv University, 2017.

ZJW16. Bin Zhang, Lin Jiao, and Mingsheng Wang. Faster algorithms for solving LPN. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part I*, volume 9665 of *Lecture Notes in Computer Science*, pages 168–195, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany.

Appendices

A Additional Preliminaries

A.1 Function Secret Sharing

Function secret sharing (FSS), introduced in [BGI15, BGI16], allows to succinctly share functions. In this section, we largely follow the presentation from the preliminaries of [BCG+20b] (in particular, the definitions are reproduced almost verbatim from [BCG+20b]). An FSS scheme splits a secret function $f : I \to \mathbb{G}$, where $\mathbb{G}$ is some Abelian group, into two functions f_0, f_1, each represented by a key K_0, K_1, such that: (1) $f_0(x) + f_1(x) = f(x)$ for every input $x \in I$, and (2) each of K_0, K_1 individually hides f.

Definition 34 (Function Secret Sharing). *Let $\mathcal{C} = \{f : I \to \mathbb{G}\}$ be a class of function descriptions, where the description of each f specifies the input domain I and an Abelian group $(\mathbb{G}, +)$ as the output domain. A (2-party)* function secret sharing *(*FSS*) scheme for $\mathcal{C}$ is a pair of algorithms* $\mathsf{FSS} = (\mathsf{FSS.Gen}, \mathsf{FSS.Eval})$ *with the following syntax:*

- $\mathsf{FSS.Gen}(1^\lambda, f)$ *is a Probabilistic Polynomial Time (PPT) algorithm that given security parameter λ and description of $f \in \mathcal{C}$ outputs a pair of keys (K_0, K_1). We assume that the keys specify I and $\mathbb{G}$.*
- $\mathsf{FSS.Eval}(b, K_b, x)$ *is a polynomial-time algorithm that, given a key K_b for party $b \in \{0, 1\}$, and an input $x \in I$, outputs a group element $y_b \in \mathbb{G}$.*

The scheme should satisfy the following requirements:

- **Correctness:** *For any $f \in \mathcal{C}$ and $x \in I$, we have*

$$\Pr\left[(K_0, K_1) \xleftarrow{\$} \mathsf{FSS.Gen}(1^\lambda, f) \;\middle|\; \sum_{b\in\{0,1\}} \mathsf{FSS.Eval}(b, K_b, x) = f(x)\right] = 1.$$

- **Security:** *For any $b \in \{0, 1\}$, there exists a PPT simulator* Sim *such that for any polynomial-size function sequence $f_\lambda \in \mathcal{C}$, the distributions $\{(K_0, K_1) \xleftarrow{\$} \mathsf{FSS.Gen}(1^\lambda, f_\lambda) \mid K_b\}$ and $\{K_b \xleftarrow{\$} \mathsf{Sim}(1^\lambda, \mathsf{Leak}(f_\lambda))\}$ are computationally indistinguishable.*

In the constructions we use, the leakage function $\mathsf{Leak} : \{0, 1\}^* \to \{0, 1\}^*$ *is given by* $\mathsf{Leak}(f_\lambda) = (I, \mathbb{G})$*, namely it outputs a description of the input and output domains of f.*

We also define a full-domain evaluation algorithm, $\mathsf{FSS.FullEval}(b, K_b)$, which outputs a vector of $|I|$ group elements, corresponding to running Eval on every element x in the domain I. For the type of FSS we consider, FSS.FullEval is significantly faster than the generic solution of running $|I|$ instances of Eval. We will use FSS for point functions and sums of point functions, as defined below.

Definition 35 (Distributed Point Function (DPF) [GI14, BGI15]). *Denote by $[n]$ the set of integers $\{0, \ldots, n-1\}$. For an Abelian group $\mathbb{G}$, $\alpha \in [n]$, and $\beta \in \mathbb{G}$, the* point function $f_{\alpha,\beta}$ *is the function $f_{\alpha,\beta} : [n] \to \mathbb{G}$ defined by $f_{\alpha,\beta}(x) = 0$ whenever $x \neq \alpha$, and $f_{\alpha,\beta}(x) = \beta$ if $x = \alpha$. A* distributed point function *(DPF) is an* FSS *scheme for the class of point functions $\{f_{\alpha,\beta} : [n] \to \mathbb{G} \mid \alpha \in [n], \beta \in \mathbb{G}\}$.*

The best known DPF construction [BGI16] can use any pseudorandom generator (PRG) $G : \{0, 1\}^\lambda \to \{0, 1\}^{2\lambda+2}$ and has the following efficiency features. For $m = \lceil \frac{\log |\mathbb{G}|}{\lambda+2} \rceil$, the key generation algorithm Gen invokes G at most $2(\lceil \log n \rceil + m)$ times, the evaluation algorithm Eval invokes G at most $\lceil \log n \rceil + m$ times, and the full-domain evaluation algorithm FullEval invokes G at most $n \cdot (1 + m)$ times. The size of each key is at most $\lceil \log n \rceil \cdot (\lambda + 2) + \lambda + \lceil \log_2 |\mathbb{G}| \rceil$ bits. We will use a simple and generic extension of DPF to sums of point functions.

Definition 36 (FSS for sum of point functions (SPFSS)). *For $S = (s_1, \ldots, s_t) \in [n]^t$ and $\mathbf{y} = (y_1, \ldots, y_t) \in \mathbb{G}^t$, define the* sum of point functions *$f_{S,\mathbf{y}} : [n] \to \mathbb{G}$ by*

$$f_{S,\mathbf{y}}(x) = \sum_{i=1}^{t} f_{s_i,y_i}(x).$$

An SPFSS *scheme is an* FSS *scheme for the class of sums of point functions.*

Note that for $S = (s_1, \ldots, s_t)$, the function $f_{S,\mathbf{y}}$ is non-zero on *at most* t points. If the elements of S are distinct, $f_{S,\mathbf{y}}$ coincides with a *multi-point function* for the set of points in S. A simple realization of SPFSS is by summing t independent instances of DPF. This will typically be good enough for our purposes. To simplify notation, when generating keys for a scheme $\mathsf{SPFSS} = (\mathsf{SPFSS.Gen}, \mathsf{SPFSS.Eval})$, we write $\mathsf{SPFSS.Gen}(1^\lambda, S, \mathbf{y})$, instead of explicitly writing $f_{S,\mathbf{y}}$.

A.2 Pseudorandom Correlation Generators

We recall the notion of pseudorandom correlation generator (PCG) from [BCG+19b]. At a high level, a PCG for some target ideal correlation takes as input a pair of short, correlated seeds and outputs long correlated pseudorandom strings, where the expansion procedure is deterministic and can be applied locally. The definitions below are taken almost verbatim from [BCG+20b].

Definition 37 (Correlation generator). *A PPT algorithm $\mathcal{C}$ is called a* correlation generator, *if $\mathcal{C}$ on input 1^λ outputs a pair of elements in $\{0,1\}^n \times \{0,1\}^n$ for $n \in \mathsf{poly}(\lambda)$.*

The security definition of PCG's requires the target correlation to satisfy a technical requirement, which roughly says that it is possible to efficiently sample from the conditional distribution of R_0 given $R_1 = r_1$ and vice versa. It is easy to see that this is true for the correlations considered in this paper.

Definition 38 (Reverse-sampleable correlation generator). *Let $\mathcal{C}$ be a correlation generator. We say $\mathcal{C}$ is* reverse sampleable *if there exists a PPT algorithm* RSample *such that for $\sigma \in \{0,1\}$ the correlation obtained via:*

$$\{(R'_0, R'_1) \,|(R_0, R_1) \xleftarrow{\$} \mathcal{C}(1^\lambda), R'_\sigma := R_\sigma, R'_{1-\sigma} \xleftarrow{\$} \mathsf{RSample}(\sigma, R_\sigma)\}$$

is computationally indistinguishable from $\mathcal{C}(1^\lambda)$.

Definition 39 (Pseudorandom Correlation Generator (PCG)). *Let $\mathcal{C}$ be a reverse-sampleable correlation generator. A* pseudorandom correlation generator (PCG) for $\mathcal{C}$ *is a pair of algorithms* $(\mathsf{PCG.Gen}, \mathsf{PCG.Expand})$ *with the following syntax:*

- $\mathsf{PCG.Gen}(1^\lambda)$ *is a PPT algorithm that given a security parameter λ, outputs a pair of seeds $(\mathsf{k}_0, \mathsf{k}_1)$;*
- $\mathsf{PCG.Expand}(\sigma, \mathsf{k}_\sigma)$ *is a polynomial-time algorithm that given a party index $\sigma \in \{0,1\}$ and a seed k_σ, outputs a bit string $R_\sigma \in \{0,1\}^n$.*

The algorithms $(\mathsf{PCG.Gen}, \mathsf{PCG.Expand})$ *should satisfy the following:*

- **Correctness.** *The correlation obtained via:*

  $$\{(R_0, R_1) \mid (\mathsf{k}_0, \mathsf{k}_1) \xleftarrow{\$} \mathsf{PCG.Gen}(1^\lambda), R_\sigma \leftarrow \mathsf{PCG.Expand}(\sigma, \mathsf{k}_\sigma) \text{ for } \sigma \in \{0,1\}\}$$

  is computationally indistinguishable from $\mathcal{C}(1^\lambda)$.
- **Security.** *For any $\sigma \in \{0,1\}$, the following two distributions are computationally indistinguishable:*

  $$\{(\mathsf{k}_{1-\sigma}, R_\sigma) \mid (\mathsf{k}_0, \mathsf{k}_1) \xleftarrow{\$} \mathsf{PCG.Gen}(1^\lambda), R_\sigma \leftarrow \mathsf{PCG.Expand}(\sigma, \mathsf{k}_\sigma)\} \text{ and}$$
  $$\{(\mathsf{k}_{1-\sigma}, R_\sigma) \mid (\mathsf{k}_0, \mathsf{k}_1) \xleftarrow{\$} \mathsf{PCG.Gen}(1^\lambda), R_{1-\sigma} \leftarrow \mathsf{PCG.Expand}(\sigma, \mathsf{k}_{1-\sigma}), R_\sigma \xleftarrow{\$} \mathsf{RSample}(\sigma, R_{1-\sigma})\}$$

  where RSample *is the reverse sampling algorithm for correlation $\mathcal{C}$.*

Note that PCG.Gen could simply output a sample from $\mathcal{C}$. To avoid this trivial construction, we also require that the seed size is significantly shorter than the output size.

Programmable PCG's. At a high level, a programmable PCG allows generating multiple PCG keys such that part of the correlation generated remains the same accross different instances. Programmable PCG's are necessary to construct n-party correlated randomness from the 2-party correlated randomness generated via the PCG. Informally, this is because when expanding n-party shares (e.g. of Beaver triples) into a sum of 2-party shares, the sum will involve many "cross terms"; using programmable PCG's allows maintaining consistent pseudorandom values accross these cross terms. We recall the formal definition below.

Definition 40 (Programmable PCG). *A tuple of algorithms* $\mathsf{PCG} = (\mathsf{PCG.Gen}, \mathsf{PCG.Expand})$ *following the syntax of a standard* PCG*, but where* $\mathsf{PCG.Gen}(1^\lambda)$ *takes additional random inputs* $\rho_0, \rho_1 \in \{0,1\}^\kappa$*, for a fixed parameter* κ *of size* $\mathsf{poly}(\lambda)$*, is a* programmable PCG *for a simple bilinear 2-party correlation* C_e^n *(specified by a bilinear pairing* $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ *for some groups* $\mathbb{G}_1, \mathbb{G}_2$ *and* $\mathbb{G}_T$*) if the following holds:*

- ***Correctness.*** *The correlation obtained via:*

$$\left\{ ((R_0,S_0),(R_1,S_1)) \middle| \begin{array}{c} \rho_0,\rho_1 \xleftarrow{\$} \{0,1\}^\kappa, (k_0,k_1) \leftarrow \mathsf{PCG.Gen}(1^\lambda,\rho_0,\rho_1), \\ (R_\sigma,S_\sigma) \leftarrow \mathsf{PCG.Expand}(\sigma,k_\sigma) \text{ for } \sigma \in \{0,1\} \end{array} \right\}$$

  is computationally indistinguishable from $C_e^n(1^\lambda)$*.*
- ***Programmability*** *There exist public efficiently computable functions* $\phi_0 : \{0,1\}^* \to \mathbb{G}_1^n$*,* $\phi_1 : \{0,1\}^* \to \mathbb{G}_2^n$ *such that*

$$\Pr\left[\begin{array}{c} \rho_0,\rho_1 \xleftarrow{\$} \{0,1\}^\kappa, (k_0,k_1) \leftarrow \mathsf{PCG.Gen}(1^\lambda,\rho_0,\rho_1) \\ (R_0,S_0) \leftarrow \mathsf{PCG.Expand}(0,k_0), \\ (R_1,S_1) \leftarrow \mathsf{PCG.Expand}(1,k_1) \end{array} : \begin{array}{c} R_0 = \phi_0(\rho_0) \\ R_1 = \phi_1(\rho_1) \end{array} \right] \geqslant 1 - \mathsf{negl}(\lambda),$$

  where $e : \mathbb{G}_1^n \times \mathbb{G}_2^n \to \mathbb{G}_T^n$ *is the bilinear map obtained by applying* e *componentwise.*
- ***Programmable security*** *The following pair of distributions are computationally indistinguishable*

$$\left\{ (k_1,(\rho_0,\rho_1)) \;\middle|\; \rho_0,\rho_1 \xleftarrow{\$} \{0,1\}^\kappa, (k_0,k_1) \leftarrow \mathsf{PCG.Gen}(1^\lambda,\rho_0,\rho_1) \right\} \text{ and}$$

$$\left\{ (k_1,(\rho_0,\rho_1)) \;\middle|\; \rho_0,\rho_1,\tilde{\rho_0} \xleftarrow{\$} \{0,1\}^\kappa, (k_0,k_1) \leftarrow \mathsf{PCG.Gen}(1^\lambda,\tilde{\rho_0},\rho_1) \right\}$$

  as well as the pair of distributions:

$$\left\{ (k_0,(\rho_0,\rho_1)) \;\middle|\; \rho_0,\rho_1 \xleftarrow{\$} \{0,1\}^\kappa, (k_0,k_1) \leftarrow \mathsf{PCG.Gen}(1^\lambda,\rho_0,\rho_1) \right\} \text{ and}$$

$$\left\{ (k_0,(\rho_0,\rho_1)) \;\middle|\; \rho_0,\rho_1,\tilde{\rho_1} \xleftarrow{\$} \{0,1\}^\kappa, (k_0,k_1) \leftarrow \mathsf{PCG.Gen}(1^\lambda,\rho_0,\tilde{\rho_1}) \right\}.$$

B From Decision-QA-SD to Search-QA-SD

In this section, we describe a reduction from the search version of QA-SD to the decision version, in the concrete chosen instantiations (all instances over $\mathcal{R} = \mathbb{F}_q[G]$ where $G = (\mathbb{Z}/(q-1)\mathbb{Z})^n$, which is the group we use to obtain PCG's for OLE's over $\mathbb{F}_q^{(q-1)^n}$). This reduction is actually a natural extension to that of [BCD22] to the multivariate setting, and essentially applies in extreme regime of low rate, *i.e.* more in the LPN regime. In [BCD22], the authors introduced a new problem they called Function Field Decoding Problem (FF-DP) that we recall below, which is the analogue of Ring-LWE with function fields instead of number fields (See Section C for a quick reminder on the theory of algebraic function fields).

Let $K/\mathbb{F}_q(T)$ be a function field with constant field $\mathbb{F}_q$ and ring of integers $\mathcal{O}_K$, and let $Q(T) \in \mathbb{F}_q[T]$ be irreducible. Let $\mathfrak{P} \stackrel{\text{def}}{=} Q\mathcal{O}_K$ be the ideal of $\mathcal{O}_K$ generated by Q. FF-DP is parametrized by a secret element $\mathbf{s} \in \mathcal{O}_K/\mathfrak{P}$, and a noise distribution ψ over $\mathcal{O}_K/\mathfrak{P}$ which is a finite set.

Definition 41 (FF-DP distribution). *A sample* $(\mathbf{a}, \mathbf{b}) \in \mathcal{O}_K/\mathfrak{P} \times \mathcal{O}_K/\mathfrak{P}$ *is distributed according to the* FF-DP *distribution modulo* $\mathfrak{P}$*, with secret* $\mathbf{s}$ *and noise distribution* ψ *if*

- $\mathbf{a}$ *is uniformly distributed over* $\mathcal{O}_K/\mathfrak{P}$*;*
- $\mathbf{b} = \mathbf{a} \cdot \mathbf{s} + \mathbf{e}$ *where* $\mathbf{e}$ *is distributed according to* ψ*.*

A sample drawn according to this distribution will be denoted by $(\mathbf{a}, \mathbf{b}) \xleftarrow{\$} \mathcal{F}_{\mathbf{s},\psi}$*.*

In its search version, the goal of FF-DP is to recover the secret $\mathbf{s}$ given access to enough samples.

Definition 42 (FF-DP (search version)). *Let* $\mathbf{s} \in \mathcal{O}_K/\mathfrak{P}$*, and let* ψ *be a probability distribution over* $\mathcal{O}_K/\mathfrak{P}$*. An instance of* FF-DP *consists in an oracle giving access to independent samples* $(\mathbf{a}, \mathbf{b}) \xleftarrow{\$} \mathcal{F}_{\mathbf{s},\psi}$*. The goal is to recover* $\mathbf{s}$*.*

In its decision version, the goal is to distinguish between the FF-DP distribution and the uniform over $\mathcal{O}_K/\mathfrak{P} \times \mathcal{O}_K/\mathfrak{P}$.

Definition 43 (FF-DP (decision version)). *Let* $\mathbf{s}$ *be drawn* uniformly *at random in* $\mathcal{O}_K/\mathfrak{P}$*, and let* ψ *be a noise distribution over* $\mathcal{O}_K/\mathfrak{P}$*. Define the following two distributions:*

- $\mathcal{D}_0 : (\mathbf{a}, \mathbf{b})$ *uniformly distributed over* $\mathcal{O}_K/\mathfrak{P} \times \mathcal{O}_K/\mathfrak{P}$*.*
- $\mathcal{D}_1 : (\mathbf{a}, \mathbf{a} \cdot \mathbf{s} + \mathbf{e})$ *distributed according to the* FF-DP *distribution* $\mathcal{F}_{\mathbf{s},\psi}$*.*

Let $b \in \{0, 1\}$*. Given access to an oracle* $\mathcal{O}_b$ *providing independent samples from distribution* $\mathcal{D}_b$*, the goal of the decision* FF-DP *is to recover* b*.*

Recall that a distinguisher between two distributions $\mathcal{D}_0$ and $\mathcal{D}_1$ is a PPT algorithm $\mathcal{A}$ that takes as input an oracle $\mathcal{O}_b$ corresponding to distribution $\mathcal{D}_b$ with $b \in \{0, 1\}$ and outputs a bit $\mathcal{A}(\mathcal{O}_b) \in \{0, 1\}$. The distinguisher wins when $\mathcal{A}(\mathcal{O}_b) = b$. Its distinguishing advantage is defined as:

$$Adv_{\mathcal{A}}(\mathcal{D}_0, \mathcal{D}_1) \stackrel{\text{def}}{=} \frac{1}{2}\Big(\mathbb{P}(\mathcal{A}(\mathcal{O}_b) = 1 \mid b = 1) - \mathbb{P}(\mathcal{A}(\mathcal{O}_b) = 1 \mid b = 0)\Big)$$

and satisfies

$$\mathbb{P}(\mathcal{A}(\mathcal{O}_b) = b) = \frac{1}{2} + Adv_{\mathcal{A}}(\mathcal{D}_0, \mathcal{D}_1).$$

The crucial remark of [BCD22] was to notice that some structured variants of the decoding problem could be somehow *lifted* to the function field setting, and could be directly seen as instances of FF-DP.

Example 44. Let $n \in \mathbb{N}$ and consider the polynomial

$$F(T, X) \stackrel{\text{def}}{=} X^n + T - 1 \in \mathbb{F}_q(T)[X].$$

Eisenstein criterion proves that F is irreducible over $\mathbb{F}_q[T]$. Define the function field

$$K \stackrel{\text{def}}{=} \mathbb{F}_q(T)[X]/(F(T, X)).$$

Computing partial derivatives shows that the curve defined by $F(T, X)$ is non-singular, and therefore $\mathbb{F}_q[T, X]/(F(T, X))$ is the full ring of integers $\mathcal{O}_K$ of K (see for instance [Lor21, Chapter VII]). Now, let $Q(T) \stackrel{\text{def}}{=} T \in \mathbb{F}_q[T]$. Then,

$$\mathcal{O}_K/T\mathcal{O}_K = \mathbb{F}_q[T, X]/(X^n + T - 1, T) = \mathbb{F}_q[X]/(X^n - 1) = \mathbb{F}_q[\mathbb{Z}/n\mathbb{Z}].$$

Therefore, QA-SD with the group $\mathbb{Z}/n\mathbb{Z}$ can be seen as an instanciation of FF-DP with the function field $K = \mathbb{F}_q(T)[X]/(X^n + T - 1)$, and modulus $Q(T) = T$.

A general search-to-decision reduction for FF-DP would therefore immediately provide a search-to-decision reduction for many variants of QA-SD. However, adapting the reduction of [LPR10] the authors of [BCD22] were only able to give such a reduction with addition algebraic constraints on K and $\mathfrak{P}$. More precisely, they gave the following theorem

Theorem 45 (Search to decision reduction for FF-DP). *Let $K/\mathbb{F}_q(T)$ be a Galois function field of degree n with field of constants $\mathbb{F}_q$, and denote by $\mathcal{O}_K$ its ring of integers. Let $Q(T) \in \mathbb{F}_q[T]$ be an irreducible polynomial. Consider the ideal $\mathfrak{P} \stackrel{def}{=} Q\mathcal{O}_K$. Assume that $\mathfrak{P}$ does not ramify in $\mathcal{O}_K$, and denote by f its inertia degree. Let ψ be a probability distribution over $\mathcal{O}_K/\mathfrak{P}$, closed under the action of $\mathrm{Gal}(K/\mathbb{F}_q(T))$, meaning that if $\mathbf{e} \leftarrow \psi$, then for any $\sigma \in \mathrm{Gal}(K/\mathbb{F}_q(T))$, we have $\sigma(\mathbf{e}) \leftarrow \psi$. Let $\mathbf{s} \in \mathcal{O}_K/\mathfrak{P}$.*

Suppose that we have an access to $\mathcal{F}_{\mathbf{s},\psi}$ and there exists a distinguisher between the uniform distribution over $\mathcal{O}_K/\mathfrak{P}$ and the FF-DP distribution with uniform secret and error distribution ψ, running in time t and having an advantage ε. Then there exists an algorithm that recovers $\mathbf{s} \in \mathcal{O}_K/\mathfrak{P}$ (with an overwhelming probability in n) in time

$$O\left(\frac{n^4}{f^3} \times \frac{1}{\varepsilon^2} \times q^{f \deg(Q)} \times t\right).$$

Unfortunately, not all group algebras arise from Galois extensions of function fields. Nonetheless, based on the analogy between cyclotomic number fields and the Carlitz modules, they proposed to instantiate their reduction with $K = \mathbb{F}_q(T)[\Lambda_T] = \mathbb{F}_q(T)[X]/(X^{q-1} + T)$, and modulus $Q(T) = T + 1$. The theory of Carlitz extensions ensures that $\mathcal{O}_K = \mathbb{F}_q[T][X]/(X^{q-1} + T)$, and therefore

$$\mathcal{O}_K/(T+1)\mathcal{O}_K = \mathbb{F}_q[T, X]/(T + 1, X^{q-1} + T) = \mathbb{F}_q[X]/(X^{q-1} - 1) = \mathbb{F}_q[\mathbb{Z}/(q-1)\mathbb{Z}].$$

The key point is the fact that $\mathrm{Gal}(K/\mathbb{F}_q(T)) = \mathbb{F}_q^\times$ and an element $\zeta \in \mathbb{F}_q^\times$ acts on $P(X) \in \mathbb{F}_q[X]/(X^{q-1}-1)$ by $\zeta \cdot P(X) \stackrel{\text{def}}{=} P(\zeta X)$. In particular, the Galois group keeps invariant the support of *any* element, and therefore any distribution that only depends on the weight is Galois invariant. Theorem 45 immediately yields a search-to-decision reduction for QA-SD instantiated with the group $G = \mathbb{Z}/(q-1)\mathbb{Z}$.

Extension to the multivariate setting. Consider the group $G = (\mathbb{Z}/(q-1)\mathbb{Z})^t$, and let $\mathcal{R} \stackrel{\text{def}}{=} \mathbb{F}_q[G] = \mathbb{F}_q[X_1, \ldots, X_t]/(X_1^{q-1} - 1, \ldots, X_t^{q-1} - 1)$. Using the heavy machinery of inverse Galois theory, it is possible to find a Galois extension of $\mathbb{F}_q(T)$ with Galois group G. However, this would induce a large overhead in the complexity of the reduction. Instead, in this case, building on the case of $\mathbb{F}_q[\mathbb{Z}/(q-1)\mathbb{Z}]$, we can directly describe the reduction and get Theorem 27 from Section 5.2.

The reduction works as follows. Recall that by the Chinese Remainder Theorem,

$$\mathcal{R} = \prod_{(\zeta_1,\ldots,\zeta_t) \in (\mathbb{F}_q^\times)^t} \mathbb{F}_q[X_1, \ldots, X_t]/(X_i - \zeta_i),$$

and fix an ordering of $(\mathbb{F}_q^\times)^t$, which yields an ordering $\mathfrak{I}_1, \ldots, \mathfrak{I}_r$ of the ideals in the above decomposition (where $r = (q-1)^t$):

$$\mathcal{R} = \prod_{i=1}^{r} \mathbb{F}_q[X_1, \ldots, X_t]/\mathfrak{I}_i.$$

Let $w \in \{0, \ldots, (q-1)^t\}$ and $\mathbf{s} \in \mathcal{R}$. Consider a noise distribution $\psi = \psi_w$ over $\mathcal{R}$ such that $\mathbb{E}[\mathrm{wt}(x)] = w$ when x is sampled according to ψ. A sample $(\mathbf{a}, \mathbf{b})$ is distributed according to $\mathcal{F}_{\mathbf{s},\psi}$ if $\mathbf{a}$ is uniformly distributed in $\mathcal{R}$, and $\mathbf{b} = \mathbf{a} \cdot \mathbf{s} + \mathbf{e}$ where $\mathbf{e} \stackrel{\$}{\leftarrow} \psi$.

The idea of the reduction is to recover the secret modulo one of the factors, and then using the action of some group recover the full secret. We keep a high level, the first steps of the reduction following *exactly* the same path as that of [BCD22]. The only difference resides in the last step and the considered group action.

Step 1: Randomizing the secret. In the *decision* version, the secret $\mathbf{s}$ is supposed to be *uniformly* distributed over $\mathcal{R}$, while in the *search* version, the secret is *fixed*. In other words, the decision version is an average-case problem, while the search version is worst-case. Fortunately, the secret can be easily randomized by sampling some $\mathbf{s}'$ uniformly at random in $\mathcal{R}$. Now, for each sample $(\mathbf{a}, \mathbf{b}) \leftarrow \mathcal{F}_{\mathbf{s},\psi}$ with a fixed secret $\mathbf{s}$, we can build the sample $(\mathbf{a}, \mathbf{b} + \mathbf{a} \cdot \mathbf{s}')$ which is distributed according to $\mathcal{F}_{\mathbf{s}+\mathbf{s}',\psi}$, and the secret is now uniformly distributed. Feeding the latter sample to a distinguisher, allows to creates a distinguisher for a fixed secret, with exactly the same advantage.

Step 2: Hybrid argument. A sample $(\mathbf{a}, \mathbf{b})$ is said to follow the hybrid distribution $\mathcal{H}_i$ if it is of the form $(\mathbf{a}', \mathbf{b}' + \mathbf{h})$ where $\mathbf{h}$ is uniformly distributed modulo $\mathfrak{I}_j$ for $j \leqslant i$, and is 0 modulo $\mathfrak{I}_j$ for $j > i$. Such an $\mathbf{h}$ is easily constructed using the Chinese Remainder Theorem. In particular, $\mathcal{H}_0 = \mathcal{F}_{s,\psi}$ and $\mathcal{H}_r$ is the uniform distribution over $\mathcal{R}$. A simple hybrid argument proves that a distinguisher between $\mathcal{H}_0$ and $\mathcal{H}_r$ with advantage ε can be turned into a distinguisher between $\mathcal{H}_{i_0}$ and H_{i_0+1} for some i_0, with advantage at least $\frac{\varepsilon}{r}$.

Step 3: Guess and search. Given i_0, the idea is to make a guess g_{i_0} for $\mathbf{s}$ modulo $\mathfrak{I}_{i_0}$ and to use the previous distinguisher to tell whether this guess is correct, or not. Define $\mathbf{g}, \mathbf{h}$ and $\mathbf{v} \in \mathcal{R}$ such that:

$$\mathbf{g} = \begin{cases} g_{i_0} & \mod \mathfrak{I}_{i_0} \\ 0 & \text{elsewhere} \end{cases} \qquad \mathbf{v} = \begin{cases} \text{random} & \mod \mathfrak{I}_{i_0} \\ 0 & \text{elsewhere} \end{cases}$$

and $\mathbf{h}$ is uniformly distributed modulo $\mathfrak{I}_j$ for $j \leqslant i_0 + 1$ and 0 elsewhere. Then, for each sample $(\mathbf{a}, \mathbf{b})$, we can build the sample $(\mathbf{a}', \mathbf{b}')$ where

$$\begin{cases} \mathbf{a}' = \mathbf{a} + \mathbf{v} \\ \mathbf{b}' = \mathbf{b} + \mathbf{h} + \mathbf{v} \cdot \mathbf{g} = \mathbf{a}' \cdot \mathbf{s} + \mathbf{e} + \mathbf{h} + \mathbf{v}(\mathbf{g} - \mathbf{s}). \end{cases}$$

Using the fact that all the factors $\mathbb{F}_q[X_1, \ldots, X_t]/\mathfrak{I}_j$ are finite fields (isomorphic to $\mathbb{F}_q$), it is easily seen that $(\mathbf{a}', \mathbf{b}')$ is distributed according to $\mathcal{H}_{i_0}$ when the guess is correct, and according to $\mathcal{H}_{i_0+1}$ otherwise. Therefore, by the Chernoff-Hoeffding bound, using our distinguisher $\Theta(n(r/\varepsilon)^2)$ times, one can detect if the guess is correct or not with probability at least $1 - 2^{-\Theta(n)}$. An exhaustive search on $\mathbb{F}_q$ yields the value of $\mathbf{s} \mod \mathfrak{I}_{i_0}$.

Step 4: A group action This is the only step that changes from the reduction of [BCD22]. We need to find a group $\widehat{G}$ that will replace the Galois group of their reduction in permuting the factors.

Inspired by the univariate example, let $\widehat{G} \stackrel{\text{def}}{=} (\mathbb{F}_q^\times)^t$. It acts on $\mathcal{R}$ by:

$$(\zeta_1, \ldots, \zeta_t) \cdot P(X_1, \ldots, X_t) \stackrel{\text{def}}{=} P(\zeta_1 X_1, \ldots, \zeta_t X_t).$$

The key observation here is that

1. This action keeps invariant the support of elements in $\mathcal{R}$. In particular, the distribution ψ is invariant under the action of $\widehat{G}$.
2. $\widehat{G}$ acts transitively on the factors:
   The action of $(\zeta_1, \ldots, \zeta_t)$ maps the ideal $(X_1 - \gamma_1, \ldots, X_t - \gamma_t)$ onto $(X_1 - \zeta_1^{-1}\gamma_1, \ldots, X_t - \zeta_t^{-1}\gamma_t)$.

Now, in order to recover $\mathbf{s} \mod \mathfrak{I}_j$ for $j \neq i_0$, it suffices to take the (unique) element $\mathbf{z} \in \widehat{G}$ such that $\mathbf{z} \cdot \mathfrak{I}_j = \mathfrak{I}_{i_0}$, and for any sample $(\mathbf{a}, \mathbf{b} = \mathbf{as} + \mathbf{e})$ we can build $(\mathbf{z} \cdot \mathbf{a}, \mathbf{z} \cdot \mathbf{b} = (\mathbf{z} \cdot \mathbf{a})(\mathbf{z} \cdot \mathbf{s}) + \mathbf{z} \cdot \mathbf{e})$. Note that $\mathbf{a}'$ (resp. $\mathbf{z} \cdot \mathbf{e}$) is still uniformly distributed over $\mathcal{R}$ (resp. distributed according to ψ since it is $\widehat{G}$-invariant). In other words, $(\mathbf{a}', \mathbf{b}')$ is distributed according to $\mathcal{F}_{\mathbf{z}\cdot\mathbf{s},\psi}$, and repeating the first three steps of the reduction will yield $\mathbf{z} \cdot \mathbf{s} \mod \mathfrak{I}_{i_0}$, which is equal to $\mathbf{s} \mod \mathbf{z}^{-1} \cdot \mathfrak{I}_{i_0} = \mathbf{s} \mod \mathfrak{I}_j$, which concludes the reduction.

C Algebraic number theory in function fields

There is a well-known analogy between the theory of finite extensions of $\mathbb{Q}$, the so-called *number fields*, and that of finite separable extensions of $\mathbb{F}_q(T)$, the field of rational functions with coefficients in a finite field $\mathbb{F}_q$. The latter algebraic extensions are called *function fields*, because they can be realized as fields of rational functions on curves over finite fields. In this section, we recall the minimal requirements about the arithmetic of function fields that are needed in the sequel. A dictionnary summarizing the analogies between function fields and number fields is represented in Table 2 below.

C.1 Algebraic function fields.

Starting from a finite field $\mathbb{F}_q$, a *function field* is a finite extension K of $\mathbb{F}_q(T)$ of the form

$$K = \mathbb{F}_q(T)[X]/(P(T, X)),$$

Number fields	Function fields
$\mathbb{Q}$	$\mathbb{F}_q(T)$
$\mathbb{Z}$	$\mathbb{F}_q[T]$
Prime numbers $q \in \mathbb{Z}$	Irreducible polynomials $Q \in \mathbb{F}_q[T]$
$K = \mathbb{Q}[X]/(f(X))$	$K = \mathbb{F}_q(T)[X]/(f(T,X))$
$\mathcal{O}_K$ = Integral closure of $\mathbb{Z}$ *Dedekind* domain	$\mathcal{O}_K$ = Integral closure of $\mathbb{F}_q[T]$ *Dedekind* domain
characteristic 0	**characteristic > 0**

Table 2. A Number-Function fields analogy

where $P(T,X) \in \mathbb{F}_q(T)[X]$ is irreducible. The field $K \cap \overline{\mathbb{F}}_q$ is referred to as *the field of constants* of K. In general, this is a (finite) extension of $\mathbb{F}_q$, but when $\mathbb{F}_q$ is the full field of constants of K, the extension $K/\mathbb{F}_q(T)$ is said to be *geometric*. This is equivalent for the modulus $P(T,X)$ to be irreducible, even regarded as a polynomial in $\overline{\mathbb{F}_q}(T)[X]$ ([Sti09, Cor, 3.6.8]). This will always be assumed in our setting.

Similarly to the number field setting, the integral closure of $\mathbb{F}_q[T]$ in K is called the *ring of integers* of K, and denoted by $\mathcal{O}_K$. This is a *Dedeking domain*. In particular, for any ideal $\mathfrak{P}$ of $\mathcal{O}_K$, there exist unique prime ideals $\mathfrak{P}_i$ and integers e_i such that $\mathfrak{P} = \mathfrak{P}_1^{e_1} \dots \mathfrak{P}_r^{e_r}$, and the quotients $\mathcal{O}_K/\mathfrak{P}_i$ are finite extensions of $\mathbb{F}_q$. When the ideal $\mathfrak{P}$ is of the form $P\mathcal{O}_K$ where $P(T) \in \mathbb{F}_q[T]$ is an irreducible polynomial, the primes $\mathfrak{P}_i$ are said to be *lying above* P[15]. The extension degrees $f_i \stackrel{\text{def}}{=} [\mathcal{O}_K/\mathfrak{P}_i \colon \mathbb{F}_q[T]/(P(T))] = [\mathcal{O}_K/\mathfrak{P}_i \colon \mathbb{F}_{q^{\deg P}}]$ are called the *inertia degrees* of P, and e_i are known as its *ramification* indexes. When the e_i's are all equal to 1, the extension is said to be *unramified* at P. In that case, the Chinese Remainder Theorem entails that $\mathcal{O}_K/\mathfrak{P}$ is isomorphic to $\prod_{i=1}^r \mathcal{O}_K/\mathfrak{P}_i$ which is a product of finite fields. All those quantities are related through the well-known formula

$$n \stackrel{\text{def}}{=} [K : \mathbb{F}_q(T)] = \sum_{i=1}^{r} e_i f_i. \tag{2}$$

C.2 Galois extensions.

Recall that the extension $K/\mathbb{F}_q(T)$ is said to be *Galois* when the automorphism group

$$\mathrm{Aut}(K/\mathbb{F}_q(T)) \stackrel{\text{def}}{=} \{\sigma \colon K \mapsto K \mid \sigma \text{ is an isomorphism with } \sigma(a) = a \ \ \forall a \in \mathbb{F}_q(T)\}$$

has cardinality $[K : \mathbb{F}_q(T)]$. In that case, this group is usually denoted by $\mathrm{Gal}(K/\mathbb{F}_q(T))$ and known as the *Galois* group of K. Galois extensions whose Galois group is abelian are called *abelian* extensions. This Galois group adds more symmetry to the function field. More specifically, G keeps $\mathcal{O}_K$ globally invariant and given an irreducible polynomial $Q(T) \in \mathbb{F}_q[T]$, it acts transitively on the prime ideals lying above Q (*i.e.* it permutes the factors). In particular, all the ramification indexes e_i (resp. the inertia degrees f_i) are equal, and denoted by e (resp. f):

$$Q\mathcal{O}_K = (\mathfrak{P}_1 \dots \mathfrak{P}_r)^e,$$

[15] Rigourously, there exists another prime element in $\mathbb{F}_q(T)$, which is $1/T$. This element does not belong to $\mathbb{F}_q[T]$, and corresponds to the point at infinity on the projective line. To take into account this additional point, we could consider the ring $\mathbb{F}_q[1/T]$ (or its localization $(\mathbb{F}_q[1/T])_{1/T}$ to avoid redundancy), and its integral closure $\mathcal{O}_{K,\infty}$ in K. It is also a Dedekind domain, and the primes of $\mathcal{O}_{K,\infty}$ lying above $(1/T)$ are known as the *places at infinity*. The main difference with the number field setting being that this place at infinity plays a similar role as the other primes (which are called *finite places* in opposition), while in number fields the *places at infinity* are called *archimedean places* would correspond to the complex embeddings of K in $\mathbb{C}$.

and Equation (2) simply becomes $n = efr$. In this work, we sometimes need to work with different extensions. When the context is not clear, we will put the irreducible polynomial in index, and the considered function field in brackets: $e_Q(K)$ and $f_Q(K)$. Another consequence is that the action of G on $\mathcal{O}_K$ is well defined on the quotient $\mathcal{O}_K/Q\mathcal{O}_K$ and simply permutes the factors $\mathcal{O}_K/\mathfrak{P}_i^e$. The *decomposition group* $D_{\mathfrak{P}_i/Q}$ of $\mathfrak{P}_i$ over Q is the subgroup of Galois automorphisms keeping $\mathfrak{P}_i$ globally invariant

$$D_{\mathfrak{P}_i/Q} \stackrel{\text{def}}{=} \{\sigma \in G \mid \sigma(\mathfrak{P}_i) = \mathfrak{P}_i\}.$$

It has cardinality $e \times f$. When K is unramified at Q, the ring $\mathcal{O}_K/\mathfrak{P}_i$ is the finite field $\mathbb{F}_{q^{f \deg(Q)}}$ and the action of $D_{\mathfrak{P}_i/Q}$ is that of the Frobenius endomorphism: the reduction modulo $\mathfrak{P}_i$ yields an isomorphism

$$D_{\mathfrak{P}_i/Q} \simeq \mathrm{Gal}(\mathbb{F}_{q^{f \deg(Q)}}/\mathbb{F}_{q^{\deg(Q)}}).$$

The decomposition groups of all the primes above Q are conjugate in $\mathrm{Gal}(K/\mathbb{F}_q(T))$: For any $i \neq j$ there exists $\sigma \in G$ such that $D_{\mathfrak{P}_i/Q} = \sigma D_{\mathfrak{P}_j/Q}\sigma^{-1}$. In particular, when the extension is *abelian*, they are all equal and referred to as the *decomposition group* of Q, and denoted by D_Q. The subfield

$$L \stackrel{\text{def}}{=} K^{D_Q} = \{x \in K \mid \sigma(x) = x \quad \forall \sigma \in D_Q\}$$

of all elements of K fixed *pointwise* by D_Q is called the *decomposition field* of Q. It is an algebraic function field, with ring of integers $\mathcal{O}_L = \mathcal{O}_K^{D_Q}$ consisting in all the elements of $\mathcal{O}_K$ pointwise fixed by D_Q. Moreover, it is a Galois extension with Galois group G/D_Q. This is the largest subextension of K in which Q *totally splits*[16] (*i.e.* $f_Q(K^{D_Q}) = e_Q(K^{D_Q}) = 1$ and $r_Q(K^{D_Q}) = r_Q(K) = r$).

$$\begin{array}{lccc}
(Q) = \mathfrak{P}_1 \dots \mathfrak{P}_r \subset & \mathcal{O}_K & \text{———} & K \qquad \mathcal{O}_K/\mathfrak{P}_i = \mathbb{F}_{q^{f \deg(Q)}} \\
& | & & | \\
(Q) = \mathfrak{p}_1 \dots \mathfrak{p}_r \subset & \mathcal{O}_K^{D_Q} & \text{———} & K^{D_Q} \qquad \mathcal{O}_K/\mathfrak{p}_i = \mathbb{F}_{q^{\deg(Q)}} \\
& | & & | \\
(Q) \subset & \mathbb{F}_q[T] & \text{———} & \mathbb{F}_q(T) \qquad \mathbb{F}_q[T]/Q = \mathbb{F}_{q^{\deg(Q)}}
\end{array}$$

C.3 The Carlitz module

In classical algebraic number theory, the cyclotomic number fields play a major role. For instance, all abelian extensions of $\mathbb{Q}$ can be realized as subfields of some cyclotomic number fields. This is known as the Kronecker-Webber Theorem, and is the cornerstone of the very important class field theory.

In the theory of algebraic function fields, the analogues of the cyclotomic extensions of $\mathbb{Q}$ are known as the *Carlitz extensions*. They were discovered by Carlitz in the late 1930's and the analogy with the cyclotomic number fields was made explicit by his student Hayes about 40 years later in [Hay74] to give an analogue of the Kronecker-Webber Theorem for the rational function field $\mathbb{F}_q(T)$. This result was later generalized by Drinfeld and Goss to yield a complete solution to Kronecker's Jugendtraum[17] for function fields, *i.e.* an explicit class field theory. In the number field setting, such an explicit construction is only known for $\mathbb{Q}$, via the cyclotomic number fields, and for imaginary quadratic number fields, via the theory of elliptic curves with complex multiplication.

In this section, we just give a quick presentation of the Carlitz modules, keeping the same notations as [BCD22]. We refer to [BCD22, § V] paper for a self-contained presentation (without proofs). For an in-depth exposition, the interested reader can refer to [Ros02, Chapter 12], [Vil06, Chapter 12], or the survey [Con].

A dictionnary summarizing the analogies between cyclotomic number fields and Carlitz extensions is given in Table 3.

[16] Hence the name decomposition field

[17] "childhood dream" in German.

$\mathbb{Q}$	$\mathbb{F}_q(T)$
$\mathbb{Z}$	$\mathbb{F}_q[T]$
Prime numbers $q \in \mathbb{Z}$	Irreducible polynomials $Q \in \mathbb{F}_q[T]$
$\mu_m = \langle \zeta \rangle \simeq \mathbb{Z}/m\mathbb{Z}$ (groups)	$\Lambda_M = \langle \lambda \rangle \simeq \mathbb{F}_q[T]/(M)$ (modules)
$d \mid m \Leftrightarrow \mu_d \subset \mu_m$ (subgroups)	$D \mid M \Leftrightarrow \Lambda_D \subset \Lambda_M$ (submodules)
$a \equiv b \mod m \Rightarrow \zeta^a = \zeta^b$	$A \equiv B \mod M \Rightarrow [A](\lambda) = [B](\lambda)$
$K = \mathbb{Q}[\zeta]$ $\mathcal{O}_K = \mathbb{Z}[\zeta]$	$K = \mathbb{F}_q(T)[\lambda]$ $\mathcal{O}_K = \mathbb{F}_q[T][\lambda]$
$\mathrm{Gal}(K/\mathbb{Q}) \simeq (\mathbb{Z}/m\mathbb{Z})^\times$	$\mathrm{Gal}(K/\mathbb{F}_q(T)) \simeq (\mathbb{F}_q[T]/(M))^\times$
Cyclotomic	**Carlitz**

Table 3. Analogies between cyclotomic and Carlitz

If one wants to build cyclotomic extensions of $\mathbb{F}_q(T)$, the most natural idea is to mimic the construction of cyclotomic number fields and to add roots of unity to $\mathbb{F}_q(T)$. However, the crucial difference with $\mathbb{Q}$ is that roots of unity are already *algebraic* over $\mathbb{F}_q$, and adjoining them to $\mathbb{F}_q(T)$ only yields a function field of the form $\mathbb{F}_{q^m}(T)$, *i.e.* an extension of the constants.

Instead, one needs to look deeper into the algebraic structure adjoined to $\mathbb{Q}$. Notice that roots of unity form an abelian group, that is to say a $\mathbb{Z}$-module. More precisely, consider the action of $\mathbb{Z}$ on $\overline{\mathbb{Q}}^\times$ by exponentiation: $m \cdot z \stackrel{\text{def}}{=} z^m$. Then, the m-th roots of unity are nothing else than the *torsion elements* of the action of $m \in \mathbb{Z}$:

$$\mu_m = \{z \in \overline{\mathbb{Q}^\times} \mid m \cdot z = 1\}.$$

At a high level, the philosophy behind the construction of Carlitz extension is to replace $\mathbb{Z}$ by $\mathbb{F}_q[T]$ when *that makes sense*, and therefore *abelian groups* by $\mathbb{F}_q[T]$*-modules*. In particular, the analogue of the exponentiation will be a new action of $\mathbb{F}_q[T]$ on $\overline{\mathbb{F}_q(T)}$, called the *Carlitz action*. This yields another structure of $\mathbb{F}_q[T]$-module on $\overline{\mathbb{F}_q(T)}$, which is called the *Carlitz module*. If $M \in \mathbb{F}_q[T]$, the elements of M–torsion are denoted $\Lambda_M \stackrel{\text{def}}{=} \{\lambda \in \overline{\mathbb{F}_q(T)} \mid M \cdot \lambda = 0\}$, and form a *cyclic* $\mathbb{F}_q[T]$-module, generated by some element denoted λ_0, which is an analogue of a *primitive* root of unity. The Carlitz extension by M will then be $\mathbb{F}_q(T)[\Lambda_M] = \mathbb{F}_q(T)[\lambda_0]$. It is a *Galois* extension of $\mathbb{F}_q(T)$ of Galois group isomorphic to $(\mathbb{F}_q[T]/(M))^\times$.

One key fact about Carlitz extensions is that their ring of integers is simply $\mathbb{F}_q[T][\lambda_0]$ and the decomposition of primes is well understood:

Theorem 46 ([Ros02, Th. 12.10]). *Let $M \in \mathbb{F}_q[T]$, $M \neq 0$, and let $Q \in \mathbb{F}_q[T]$ be a monic, irreducible polynomial. Consider the Carlitz extension K_M and let $\mathcal{O}_M$ denote its ring of integers. Then,*

- *If Q divides M, then $Q\mathcal{O}_M$ is totally ramified.*
- *Otherwise, let f be the smallest integer f such that $Q^f \equiv 1 \mod M$. Then $Q\mathcal{O}_M$ is unramified and has inertia degree f. In particular, Q splits completely if and only if $Q \equiv 1 \mod M$.*

Those results are completely analogue to the cyclotomic case.

D The Curious Case of $\mathbb{F}_2$

In Section 5, we showed how to produce batch OLE's over all finite fields $\mathbb{F}_q$ for $q \geqslant 3$. However, this approach cannot be applied as is to build OLE's over $\mathbb{F}_2$. The most natural approach to mimic previous construction is to consider the ring $\mathbb{B}_n \stackrel{\text{def}}{=} \mathbb{F}_2[X_1, \ldots, X_n]/(X_i^2 - X_i)$ of Boolean functions,

for which efficient algorithmics exist and which is isomorphic to a direct product of n copies of $\mathbb{F}_2$. However, there is a strong bias which is very similar to the one mentioned in Example 21. Indeed suppose we are given a pair $(a, as+e)$ where $a \xleftarrow{\$} \mathbb{B}_n$ and s, e are sparse with respect to the basis of monomials. Then, the constant term of both s, e is very likely to be zero and the constant term is nothing but their evaluation at 0. Moreover, the evaluation at 0 map commutes with the reduction modulo $((X_i^2 - X_i))_i$. Consequently one can evaluate our sample at 0 and the result is highly biased since $(as + e)(0) = a(0)s(0) + e(0)$ and hence is 0 whenever s, e both vanish at 0 which is highly probable. Here again, we have a distinguisher on codes from a multivariate ring which is *not* a group algebra.

More generally, we have the following simple, but powerful, impossibility result.

Theorem 47 (Impossibility result). *Let G be a finite group and let $\mathcal{R} = \mathbb{F}[G]$ be its group algebra with coefficients in a finite field $\mathbb{F}$. Assume that $\mathcal{R}$ is isomorphic, as algebra, to $\mathbb{F}_2^N$ for $N \geqslant 1$. Then, $N = 1$ and $G = \{1\}$.*

Proof. Note that G can be embedded in the invertible elements $\mathcal{R}^\times$ of $\mathcal{R}$. Indeed, any $g \in G$, when regarded as an element of $\mathcal{R}$ is invertible, with inverse g^{-1}. In particular, $|G| \leqslant |\mathcal{R}^\times|$. But the algebra isomorphism $\mathcal{R} \simeq \mathbb{F}_2^N$ induces a group isomorphism $\mathcal{R}^\times \simeq \mathbb{F}_2^\times \times \cdots \times \mathbb{F}_2^\times = \{(1, \ldots, 1)\}$. In particular, $|\mathcal{R}^\times| = 1$, and $|G| = 1$, *i.e.* $G = \{1\}$, which concludes the proof.

Theorem 47 shows that we cannot adapt directly our approach based on QA-SD to efficiently build OLE's over $\mathbb{F}_2$. In this Section though, we propose a way to overcome this limitation. In a nutshell, our approach is to consider the group algebra $\mathbb{F}_2[G]$ of some well-chosen finite abelian group G, as was done previously, such that there is an isomorphism of *modules* between $\mathbb{F}_2[G]$ and $\mathbb{F}_2^N$, but not of *algebras*. It turns out that this approach is not so different from the proposal of [BCG+20b] which uses the ring $\mathbb{F}_p[X]/(P(X))$ where $P(X) = X^{2^\ell} + 1$ is a *cyclotomic polynomial* and p is a prime such that $p \equiv 1 \mod 2^{\ell+1}$. Indeed, our proposal uses the theory of *Carlitz extensions* (see Section C.3) which are function fields analogues of cyclotomic number fields.

D.1 An attempt based on the Carlitz module.

In [BCG+20b], the authors propose to use a cyclotomic ring modulo some prime p. The natural idea to mimic their construction would be to make use of Carlitz extensions.

Consider the rational function field $\mathbb{F}_2(T)$, endowed with the Carlitz action, and let

$$K_\ell \stackrel{\text{def}}{=} \mathbb{F}_2(T)[\Lambda_{T^{\ell+1}}]$$

for some positive integer ℓ to be detailed later. The theory of Carlitz modules asserts that K_ℓ is a Galois extension of $\mathbb{F}_2(T)$ of degree 2^ℓ, and of Galois group

$$G \stackrel{\text{def}}{=} \left(\mathbb{F}_2[T]/(T^{\ell+1})\right)^\times.$$

The first idea that comes to mind is to find an irreducible modulus $Q(T) \in \mathbb{F}_2[T]$ that splits completely in $\mathcal{O}_K$ so that

$$\mathcal{O}_K/Q\mathcal{O}_K \simeq \mathbb{F}_{2^{\deg(Q)}} \times \cdots \times \mathbb{F}_{2^{\deg(Q)}} \simeq \mathbb{F}_2 \times \cdots \times \mathbb{F}_2.$$

On the one hand, this shows that a necessary condition is $\deg(Q) = 1$. On the other hand, by Theorem 46, the ideal $Q\mathcal{O}_{K_\ell}$ splits completely if and only if $Q \equiv 1 \mod T^{\ell+1}$. In particular, $\deg(Q)$ needs to be large enough, and both conditions are incompatible.

Therefore, one needs to relax some of the hypotheses above in order to make this idea somehow work. Clearly, the first condition ($\deg(Q) = 1$) cannot be released, because all factors of $\mathcal{O}_K/(Q\mathcal{O}_K)$ are extension fields of $\mathbb{F}_{2^{\deg(Q)}}$, of dimension the inertia degree of the ideal generated by Q. Therefore, the only condition that can be relaxed is the second one.

Let $Q \in \mathbb{F}_2[T]$ be an irreducible polynomial of degree 1. There are only two possibilities, namely $Q = T$ or $Q = T + 1$. However, by Theorem 46, T ramifies in $\mathcal{O}_K$, and therefore the only possible choice for Q is $T+1$. Now, we need to compute the inertia degree. By the aformentionned theorem, it is characterized by the multiplicative order of $T + 1$ modulo $T^{\ell+1}$. It is not hard to see that it is the least power of 2 greater (or equal) than $\ell + 1$.

In the sequel, we make a concrete choice for the parameter ℓ. Assume that we want to produce 2^{20} OLE's correlations. This is an estimation of the order of magnitude of the number of multiplicative gates in a concrete arithmetic circuit. The number of OLE's produced being the number of factors, we need to set $\ell > 20$. As we will see, setting $\ell = 25$ is enough.

Indeed, the least power of 2 greater than 26 is $32 = 2^5$. Therefore, $(T+1)$ has inertia degree 32 in $\mathcal{O}_K$. Theorem 46 and Equation 2 entail that

$$\mathcal{O}_K/(T+1)\mathcal{O}_K \equiv \underbrace{\mathbb{F}_{2^{32}} \times \cdots \times \mathbb{F}_{2^{32}}}_{2^{20} \text{ times}}.$$

With only Carlitz extensions, this is the best that we can produce[18]. However, we are not required to use the *full* Carlitz extension: We could consider an intermediate one, that would cancel the inertia. This is precisely the *decomposition field*:

Proposition 48. *Let $\ell = 25$, and let K be the Carlitz extension by $T^{\ell+1} \stackrel{def}{=} T^{26}$. Let D_{T+1} be the decomposition group of $T+1$, and let $L \stackrel{def}{=} K^{D_{T+1}}$ denote the fixed field by D_{T+1}. Then we have:*

$$\mathcal{O}_L/(T+1)\mathcal{O}_L = \underbrace{\mathbb{F}_2 \times \ldots \mathbb{F}_2}_{2^{20}\ times}.$$

Although we will not provide it here because it would only obfuscate the speech, $\mathcal{O}_K$ has an explicit description of the form

$$\mathcal{O}_K = \mathbb{F}_2[X]/(1 + P(X))$$

where P is a linearized polynomial of degree 2^{25}, *i.e.* the only monomials that appear in P are powers of 2. In particular, P has a very sparse description. On the other hand, the ring $\mathcal{O}_L$ does not seem to inherit this property. It is only defined as the subring of $\mathcal{O}_K$ fixed by D_{T+1}, and we need to understand how D_{T+1} acts on $\mathcal{O}_K$. Recall that by definition, D_{T+1} acts as the Frobenius of each of the factors of $\mathcal{O}_K/(T+1)\mathcal{O}_K$. In other words, $\mathcal{O}_L/(T+1)\mathcal{O}_L$ is the subring of $\mathcal{O}_K/(T+1)\mathcal{O}_K$ fixed by the Frobenius on each factor (after applying the Chinese Remainder Theorem). Note that the action of D_{T+1} can be directly understood on $\mathcal{O}_K$ (before CRT): Indeed, it is isomorphic to the cyclic group (of order 32) generated by $(T+1) \in \left(\mathbb{F}_2[T]/T^{26}\right)^\times$, where $T+1$ acts on $F(T,X) \in \mathcal{O}_K$ by $F(T,(T+1)\cdot X)$ via the Carlitz action on the second variable. In other words, $\mathcal{O}_L$ is the subring of $\mathcal{O}_K$ fixed by this Carlitz action.

D.2 Building OLE's.

It suffices to build *one* OLE over $\mathcal{O}_L/(T+1)\mathcal{O}_L$ to generate 2^{20} OLE's over $\mathbb{F}_2$. Let

$$\mathcal{R} \stackrel{\text{def}}{=} \mathcal{O}_L/(T+1)\mathcal{O}_L.$$

Following [BCG+20b], in order to build an OLE over $\mathcal{R}$, we could generate $U, V \in \mathcal{R}$ pseudorandom such that they admit a sparse description of the form $U = a \cdot e_1 + f_1$ and $V = a \cdot e_2 + f_2$ with e_i, f_i somehow *sparse*. However, there are two issues here:

- How to assert pseudorandomness here ?
- What does it mean to have a sparse description in $\mathcal{R}$?

If in [BCG+20b] the sparsity is well defined in the *canonical* basis, it is not clear what basis to choose in $\mathcal{R}$. Note that $\mathcal{O}_K/(T+1)\mathcal{O}_K$ admits a monomial basis (this is a consequence of the fact that K being a Carlitz extension, $\mathcal{O}_K$ is generated over $\mathbb{F}_q[T]$ by a unique element λ_0), but it is no longer true for $\mathcal{R}$.

However, as it was recalled in [BCD22], since $T+1$ is not ramified in $\mathcal{O}_L$, a result of Noether (see [Noe32] for the original paper (in German)) entails that $\mathcal{O}_L$ admits a local normal basis, *i.e.* there exists $a \in \mathcal{R}$ such that $(\sigma(a))_{\sigma \in \Gamma}$ forms an $\mathbb{F}_2$-basis of $\mathcal{R}$, where

$$\Gamma = \mathrm{Gal}(L/\mathbb{F}_2(T)) = \left(\mathbb{F}_2[T]/(T^{26})\right)^\times/(T+1)$$

[18] With finite places

is the Galois group of L. In fact, here, the normal basis is easy to find: Indeed, Γ acts transitively on the factors of $\mathcal{R}$. This means that starting from $e_1 = (1, 0, \ldots, 0)$, $\sigma(e_1)$ is another element of the canonical basis of $\mathbb{F}_2^{2^{20}}$ for all $\sigma \in \Gamma$. In particular, e_1 generates a normal basis of $\mathbb{F}_2^{2^{20}}$, and therefore its inverse through the CRT generates a normal basis of $\mathcal{R}$. Let us call this polynomial $\varepsilon(X)$.

It is tantalizing to define the sparsity with respect to this basis. Moreover, the existence of this normal basis has a powerful consequence. Indeed, let $a \in \mathcal{R}$. Written in the normal basis, we have that

$$a = \sum_{\sigma \in \Gamma} a_\sigma \sigma(\varepsilon) = \underbrace{\left(\sum_{\sigma \in \Gamma} a_\sigma \sigma \right)}_{\stackrel{\text{def}}{=} A} (\varepsilon(X)).$$

In other words, we can write $a \in \mathcal{R}$ as $A(\varepsilon(X))$ where A now belongs to the group algebra $\mathbb{F}_2[\Gamma]$. This exactly means that $\mathcal{R} = \mathbb{F}_2[\Gamma] \cdot \varepsilon$, *i.e.* that $\mathcal{R}$ is a free module of rank one over $\mathbb{F}_2[\Gamma]$, *i.e.* that $\mathcal{R}$ is *isomorphic* to the group algebra $\mathbb{F}_2[\Gamma]$ as *modules*.

D.3 QA-SD to the rescue.

With this result in hand, it is very appealing to define our OLE over $\mathbb{F}_2[\Gamma]$, and *then only* map it to $\mathcal{R}$, since hardness of QA-SD would provide security.

Proposition 49. *Let a be uniformly distributed over $\mathbb{F}_2[\Gamma]$ and $e, f \in \mathbb{F}_2[G]$ of Hamming weight t. Then, $a \cdot e + f$ is pseudorandom assuming the hardness of* QA-SD *over $\mathbb{F}_2[\Gamma]$.*

Remark 50. Note that Theorem 20 also holds in the modular setting, and therefore it holds for $\mathbb{F}_2[\Gamma]$. In particular, according to our analysis, QA-SD with this instantiation is secure against linear tests.

Now, since $\mathcal{R}$ is isomorphic (as a module) to $\mathbb{F}_2[G]$, if $U \in \mathbb{F}_2[\Gamma]$ is pseudorandom, then $U(\varepsilon) \in \mathcal{R}$ is pseudorandom. Everything seems to be there for building an OLE over $\mathcal{R}$: Let $U, V \in \mathcal{R}$ be such that $U = (a \cdot e_1 + f_1)(\varepsilon(X))$ and $V = (a \cdot e_2 + f_2)(\varepsilon(X))$ with a uniformly distributed over $\mathbb{F}_2[G]$, and e_i, f_i sparse (as elements of $\mathbb{F}_2[G]$). Proposition 49 entails that U and V are pseudorandom in $\mathcal{R}$. Following [BCG+20b], if we can distribute additive shares of the product $U \cdot V \in \mathcal{R}$, we would win. However, here the operations do not commute, and we cannot use FSS for point functions to distribute shares of the cross products. Indeed,

$$\begin{aligned} U \times V &= (a \cdot e_1 + f_1)(\varepsilon) \times (a \cdot e_2 + f_2)(\varepsilon) \\ &= (a \cdot e_1)(\varepsilon) \times (a \cdot e_2)(\varepsilon) + (a \cdot e_1)(\varepsilon) \times (f_2)(\varepsilon) + (a \cdot e_2)(\varepsilon) \times (f_1)(\varepsilon) \\ &\quad + (f_1)(\varepsilon) \cdot (f_2)(\varepsilon), \end{aligned}$$

and if every term admits a sparse presentation, it is not clear to us how to distribute additive shares of them.

D.4 A note on efficiency.

Even if the previous problem is solved, there remains the question of efficiency. Indeed, fast encoding of quasi-abelian codes, *i.e.* fast multiplication in the group algebra, is usually done through the Fast Fourier Transform, which does not extend *a priori* to the modular setting since it is not semisimple. However, a recent work of Hong, Viterbo and Belfiore ([HVB16]) developped a *modular* FFT over $\mathbb{F}_2$ for the specific group $(\mathbb{Z}/2\mathbb{Z})^s$. Their algorithm is particularly efficient because it only involves *additions*, and could be optimized on hardware.

Our group Γ is a little bit more complicated (see [CL17, Proposition 2.4]):

$$\Gamma \stackrel{\text{def}}{=} \left(\mathbb{F}_2[T]/T^{26}\right)^\times/(T+1) \stackrel{\text{def}}{=} (\mathbb{Z}/2\mathbb{Z})^6 \times (\mathbb{Z}/4\mathbb{Z})^3 \times (\mathbb{Z}/8\mathbb{Z}) \times (\mathbb{Z}/16\mathbb{Z}).$$

However, note that $\mathbb{F}_2[\mathbb{Z}/2^k\mathbb{Z}] \simeq \mathbb{F}_2[X]/(X^{2^k})$. In other words, multiplication in this group algebra can be thought as a truncated multiplication in $\mathbb{F}_2[X]$. Now, many algorithms have been

developped as analogues of FFT in characteristics 2. They are known as additive fast Fourier transform, and even benefit from very efficient implementation [Cox21, LHW18]. If they actually work over extensions of $\mathbb{F}_2$, recent works such as [LCK+18] suggest that multiplying polynomials over $\mathbb{F}_2$ could be made very efficient.

Finally, as the group algebra of a direct product, $\mathbb{F}_2[\Gamma]$ is a tensor product, *i.e.* isomorphic to a multivariate ring, where the degrees of the variables are bounded by the corresponding power of 2. The existence of multivariate FFT also suggests the existence of efficient multivariate *additive* FFT in characteristics 2.

Moreover, this description is very naive, and further work may actually directly design efficient algorithms for multiplication in modular group algebras over $\mathbb{F}_2$ in the spirit of what has been done for $(\mathbb{Z}/2\mathbb{Z})^s$.