Fast Pseudorandom Correlation Functions from Sparse LPN*

Lennart Braun[1], Geoffroy Couteau[1], Kelsey Melissaris[2**], Mahshid Riahinia[3***], and Elahe Sadeghi[4]

[1] Université Paris Cité, CNRS, IRIF, F-75013, Paris, France
[2] Chalmers University of Technology, Göteborg, Sweden
[3] DIENS, École normale supérieure, CNRS, Paris, France
[4] University of Texas at Austin, Texas, USA
{couteau,lbraun}@irif.fr,kelseym@chalmers.se,mahshid.riahinia@ens.fr,elahesadeghi@utexas.edu

Abstract. We introduce a new and efficient pseudorandom correlation function whose security reduces to the sparse LPN assumption in the random oracle model. Our construction is the first to achieve high concrete efficiency while relying on well-established assumptions: previous candidates either required introducing new assumptions, or had poor concrete performances. We complement our result with an in-depth analysis of the sparse LPN assumption, providing new insight on how to evaluate the strength of concrete sets of parameters.

* This article is the full version of an article with the same name [BCMRS25], © IACR 2025, to appear at Asiacrypt 2025.
** The majority of this research was done while Kelsey Melissaris was employed by Aarhus University, Denmark, and was visiting IRIF, Paris.
*** This work was done while Mahshid Riahinia was visiting IRIF, Paris.

Table of Contents

Fast Pseudorandom Correlation Functions from Sparse LPN . . . 1
Lennart Braun, Geoffroy Couteau, Kelsey Melissaris, Mahshid Riahinia, and Elahe Sadeghi
1 Introduction . . . 3
Silent Secure Computation. . . . 3
Pseudorandom Correlation Functions . . . 3
Our Contribution. . . . 3
1.1 Pseudorandom Correlation Functions: A Brief Overview . . . 3
1.2 The Challenge of Finding a Usable PCF for Real-World MPC . . . 4
1.3 Our Results . . . 5
1.4 On the Sparse LPN Assumption . . . 6
1.5 Technical Overview . . . 7
Our Approach. . . . 8
A Concrete Analysis of Sparse LPN. . . . 9
Additional Optimizations. . . . 10
Open Questions. . . . 10
2 Preliminaries . . . 11
2.1 Notation . . . 11
2.2 Learning Parity with Noise . . . 11
2.3 Distributed Point Functions . . . 12
2.4 Pseudorandom Correlation Functions . . . 12
2.5 Vector Oblivious Linear Evaluation . . . 13
3 A PCF for VOLE . . . 14
3.1 Properties . . . 14
Key sizes. . . . 14
DMPF Evaluations. . . . 16
Asymptotic Parameters and Security. . . . 16
3.2 Pseudorandom VOLE-Correlated Outputs . . . 16
Correctness. . . . 16
Pseudorandomness. . . . 17
3.3 Security . . . 20
Security for $\sigma = 0$. . . . 21
Security for $\sigma = 1$. . . . 21
4 Sparse LPN Parameters and Efficiency . . . 22
4.1 Attacking Sparse LPN . . . 23
4.2 Security From High Dual Distance . . . 23
4.3 Computing the Dual Distance . . . 24
Security in the Linear Test Framework. . . . 24
A Detailed Analysis of the Dual Distance of $\mathbf{A}$ over $\mathbb{F}_2$. . . . 24
Analysis Over a General Field. . . . 26
Numbers. . . . 27
4.4 Concrete Parameter Sets . . . 29
4.5 Computing the Dual Distance Over a General Field . . . 30
Hamming Weight of the Sum of Two Vectors. . . . 30
Recursive Distribution for w Vectors. . . . 31
Usefulness. . . . 32
References . . . 33

1 Introduction

Modern secure computation protocols fundamentally rely on trusted sources of correlated randomness, typically in the form of random oblivious transfers (Alice gets two random bits (s_0, s_1) and Bob gets (b, s_b) for a random b), vector-OLE (Alice gets vectors $(\mathbf{u}, \mathbf{v}) \in \mathbb{F}^n \times \mathbb{F}^n$, Bob gets $(\Delta, \mathbf{u} \cdot \Delta + \mathbf{v})$ for $\Delta \in \mathbb{F}$), or Beaver triples [Bea92] (the parties get shares of $(a, b, a \cdot b)$ for random a, b). The generation of the correlated randomness can be delegated to a (partially) trusted dealer [Bea97; BGHIN25], at the cost of making stronger trust assumptions, or securely distributed among the participants in a preprocessing phase. Traditional approaches for the latter, based on oblivious transfers [GMW87; FKOS15; KOS16] or somewhat homomorphic encryption [DPSZ12; KPR18], incur a significant communication overhead that forms a bottleneck cost for the overall efficiency.

Silent Secure Computation. A recent paradigm put forth in [BCGI18; Boy+19b], called *silent secure computation*, overcomes this bottleneck by designing methods to securely generate a large amount of *pseudorandom* correlations using a minimal amount of interaction. In recent years, this paradigm has gained significant traction and has become the cornerstone of many efficient secure computation protocols such as private set intersection [RS21; BC23; RR22], threshold signatures [ANOSS22; KOR23; FHKRS25], zero-knowledge proofs [YWLZW20; WYKW21; DIO21; YSWW21; BBMS22], privacy-preserving machine learning [HLHD22; RSWP23], and more. At the heart of the silent secure computation paradigm is the use of primitives that allow a set of participants given short correlated seeds to *locally* expand them, without any interaction, into long correlated pseudorandom strings.

Pseudorandom Correlation Functions (PCFs) are perhaps the most versatile primitive for silent secure computation. Informally, a PCF distributes a pair of correlated keys that allows the participants to generate *on-the-fly* any target amount of correlated (pseudo)randomness. PCFs extend the notion of *pseudorandom correlation generators* (PCGs). PCGs allow two parties, given short correlated seeds, to locally generate long correlated pseudorandom strings. However, unlike this extension in PCGs can be done only once and to an a-priori bounded amount of correlated randomness. PCFs have the appealing feature that, after a one-time setup to securely generate the keys, the participants never need to interact anymore to generate correlated randomness in any future protocol: they can instead locally extract from their keys any amount of correlated randomness they want, upon need. Due to their appealing features, pseudorandom correlation functions have been suggested in recent works as the right primitive of choice for some standard MPC tasks such as threshold signatures [KOR23; FHKRS25].

Yet, in spite of their theoretical benefits, pseudorandom correlation functions remain largely absent from state-of-the-art secure computation protocols, where they are usually dismissed in favor of their (less versatile but) better-understood and more efficient counterpart, pseudorandom correlation generators (PCGs). As we argue next, this stems from the fact that the current state of the art in PCFs is largely unsatisfying: existing constructions broadly fall in one of two categories comprising candidates that are either *too slow* or *based on poorly-understood assumptions*.

Our Contribution. In this work, our main contribution is a new construction of pseudorandom correlation functions for the OT and VOLE correlations that strikes a much better balance than previous candidates. Our construction reduces to the sparse-LPN assumption, a decades-old and well-studied cryptographic assumption, and generates correlations at a rate that matches that of the fastest known constructions in all realistic parameter ranges.

1.1 Pseudorandom Correlation Functions: A Brief Overview

A (2-party) pseudorandom correlation function for a correlation $\mathcal{Y}$ (that is, a distribution over pairs (y_0, y_1)) is a pair $(\mathsf{KeyGen}, \mathsf{Eval})$ where KeyGen generates a pair of keys (k_0, k_1) and Eval, given (b, k_b, x) for any $b \in \{0, 1\}$ and input x, outputs a value y_b. A PCF is *pseudorandom* if pairs (y_0, y_1) are indistinguishable from a random sample from $\mathcal{Y}$, and *secure* if given k_0, for any x, $y_1 = \mathsf{Eval}(1, k_1, x)$ looks random conditioned on $y_0 = \mathsf{Eval}(0, k_0, x)$ (and conversely). The notion was first introduced in [Boy+20a], together with a general template. As noted in [Boy+20a], a generic instantiation of the template from

LWE (for a large class of additive correlations) is implied by spooky encryption [DHRW16], but it does not yield to a practically feasible construction. To achieve better efficiency, [Boy+20a] put forth an instantiation (for the vector-OLE and oblivious transfer correlations) from a new variant of the learning parity with noise assumption – called variable-density LPN (VDLPN) – and provided a preliminary security analysis. The analysis was subsequently refined in [CD23], who identified (and fixed) a flaw in the original analysis and derived much tighter security estimates. A sample parameters from [CD23] indicates that an instance with keys of size 3 MB can produce up to 2^{30} oblivious transfer correlations, where the cost of generating each correlation is dominated by $1.8 \cdot 10^5$ calls to AES, which translates (using modern hardware with AES instructions) to a few hundred evaluations per second: definitely feasible, but not fast enough for most applications (as typical secure computation protocols evaluate circuits in a size range $10^5 \sim 10^9$, hence require a similar amount of correlated randomness).

A faster PCF was introduced in a follow-up work [Boy+22], using the same template, but introducing another new variant of the LPN assumption – called Expand-Accumulate LPN (EALPN). This variant came with several benefits over the original VDLPN-based construction, including a tighter security analysis against standard attacks on LPN and a much faster runtime: an instance with keys of size below 500 kB can produce up to 2^{30} oblivious transfer correlations, where a per-correlation cost, dominated by 1500 calls to AES, is about three orders of magnitude faster than the previous construction. Both [Boy+20a; Boy+22] left open the goal of building (reasonably efficient) PCFs from well-understood previously-studied assumptions.

The work of [OSY21] introduced a new paradigm for constructing PCFs, combining an encryption scheme with "linear decryption in the exponent" and dense ciphertext space with a *distributed discrete logarithm* procedure. Instantiating their template with Paillier yields a PCF for the VOLE correlation (over a large ring $\mathbb{Z}_N$), while using Goldwasser-Micali yields a PCF for OTs. Both schemes come with the benefit of relying only on well-known assumptions (respectively DCR and QR), but this comes at the cost of requiring expensive exponentiations over a large ring (one per correlation for VOLE, and n for OTs, where n is a security parameter). In practice, this makes them much slower than the EALPN variant (around one OT/s with the QR-based construction, and about 100 VOLEs/s with the DCR-based construction). The recent work of [BCMPR24] introduced yet another famework that builds PCFs from constrained pseudorandom functions (albeit only for the OT correlation), and obtain a construction that strikes a reasonable balance: it relies on the power-DDH assumption (a relatively well-known assumption) together with the security of a weak pseudorandom function with a specific "inner-product membership" structure. The latter can be instantiated either from the recent weak PRF candidate from [BIPSW18], or from a variant of the Goldreich pseudorandom generator [Gol00; AR16], an assumption that has received a significant amount of attention, albeit not in the parameter range considered in [BCMPR24] (and indeed, a new attack tailored to their parameter range was later described in [FLLL24], compromising some of their candidate parameters). For generating up to 2^{40} OTs, the construction of [BCMPR24] has key size around 30 kB (when updating the parameters to account for the attack of [FLLL24]) and can produce around 21k OTs/s (on one core of an M1 Pro, according to the implementation of [CDDKS24]). Eventually, we mention in passing the work of [CDDKS24] that provides a blazing-fast PCF from a similar Goldreich PRG assumption (millions of evaluations per second), but only for a weaker *list-OT* correlation that can be converted into standard OTs using communication (and therefore does not enable fully silent secure computation with an optimal online phase).

1.2 The Challenge of Finding a Usable PCF for Real-World MPC

As outlined in the above overview, early constructions of PCFs typically relied on introducing new assumptions (with some minimal amount of preliminary cryptanalysis), leaving a lot of uncertainties regarding the actual security of the assumption and the choice of concrete parameters. As a matter of fact, these assumptions and their parameters have often been called into questions. In the related context of pseudorandom correlation generators (a less versatile variant of PCFs where the generation of correlated randomness is "one shot"), an efficient proposal based on a new LPN variant for so-called *Silver codes* [CRR21] was later broken in [RRT23]. The original analysis of VDLPN in [Boy+20a] was later found to be flawed in [CD23], and the new analysis of [CD23] requires using a different variant of VDLPN to obtain tight security bounds. As for EALPN, some parameter sets from [Boy+22] were based on a conjecture that was later invalidated in [RRT23]. Eventually, the original parameters of the BIPSW weak PRF candidate [BIPSW18] have been significantly revised in the past few years due to improved

cryptanalysis [CCKK22; JMN23], and the Goldreich PRG variant used in the PCF of [BCMPR24] uses a set of parameters that was later broken in [FLLL24].

As an ongoing research program, this is a healthy dynamic, where new approaches motivate the study of new primitives (or of new parametrizations of existing primitives) and attract cryptanalysis efforts, furthering our understanding of these primitives and their design principles. However, this falls short of providing candidates that can already be trusted, and companies implementing and deploying secure computation protocols should (and do) avoid relying on immature assumptions. Unfortunately, the landscape of PCFs based on well-established, "mature" assumptions is also unsatisfying: existing constructions are simply way too slow – for example, the fastest known PCF for OT correlations from a well-established assumption (quadratic residuosity) can output around one OT/s on one core of a standard laptop, when almost all applications require at minima a few *thousands* OT correlations (and often up to millions or billions). In addition, none of the PCFs based on well-established assumptions offer post-quantum security.

This state of affairs makes it clear why, in spite of their attracting feature (enabling on-demand, near-unbounded offline preprocessing for secure computation), PCFs remain mostly as of today a purely academic theoretical subject of study. This stands in stark contrast with pseudorandom correlation *generators*, that do admit constructions that are simultaneously fast and based on well-established assumptions, and as a result are being implemented in multiple libraries [Gal19; WMK16; RR; DHKS25], deployed in real-world applications (e.g. TLSNotary [Prib] or Sign [Eth]), and in commercial solutions (such as zkPass [zkP23], zkTLS [Pria]).

1.3 Our Results

In this work, we introduce a new pseudorandom correlation function. Our PCF represents a significant improvement over the previous state of affairs: it relies on a well-established assumption (sparse LPN), offers very good concrete efficiency (competitive with the fastest PCFs based on immature assumptions), and enables generating both VOLE correlations (the cornerstone of many modern zero-knowledge proof systems, such as [YWLZW20; WYKW21; DIO21; YSWW21; BBMS22]) and OT correlations.

From a *theoretical* point of view, our PCF is the first PCF from a well-established variant of LPN, and the first (efficient) PCF from a well-established post-quantum assumption, putting it on par with state-of-the-art constructions of pseudorandom correlation generators. Compared with previous constructions from VDLPN, EALPN, DCR, or QR, our PCF comes with a theoretical limitation: while previous constructions were plausibly secure against a (subexponential-time) adversary requesting a subexponential number of PCF evaluations, our new PCF is only secure (against subexponential-time adversaries) for up to $n^{\log n/\log\log n}$ queries, which is superpolynomial (n is the LPN dimension; other parameter tradeoffs are available). Nevertheless, for usual applications, even asymptotically, polynomially many samples would suffice, and the superpolynomial versus subexponential gap in the number of queries matters mostly from a theoretical point of view. The effect of this limitation is minor in practice: as we show, our PCF admits efficient concrete instantiations for a target number of samples in the typical range $2^{30} \sim 2^{40}$ (the same parameter range considered in all previous works).

We complement our result with a number of *practical* contributions. First, we revisit the state-of-the-art cryptanalysis of sparse LPN. In essence, the sparse LPN assumption states that a pair $(\mathbf{A}, \mathbf{A}\cdot\mathbf{s}+\mathbf{e})$ is indistinguishable from $(\mathbf{A}, \mathbf{y})$, where $\mathbf{A} \xleftarrow{\$} \mathbb{F}^{m\times n}$ is a random sparse matrix (i.e., the rows are sparse vectors), $\mathbf{e} \in \mathbb{F}^m$ is a random sparse vector (i.e., most coordinates are zero), and $\mathbf{s} \xleftarrow{\$} \mathbb{F}^n$, $\mathbf{y} \xleftarrow{\$} \mathbb{F}^m$ are random vectors. Previous works related the security of sparse LPN to the dual distance of random sparse matrices (the smallest Hamming weight of a non-zero vector in the kernel of the matrix), and relied on an analysis from [MST03; Boy+22] to prove that it grows linearly with the dimension with probability $1-o(1)$. However, their purely asymptotic analysis does not provide any meaningful bound on the dual distance for concrete parameters of interest. We remedy this situation by deriving a recursive formula to tightly estimate the dual distance of random sparse matrices, and provide a Python script to compute it explicitly for matrices over $\mathbb{F}_2$. Based on this, we compute (for any target set of concrete parameters) the smallest sparsity (the row-weight of the matrix) required to achieve a linear dual distance with a target statistical failure probability. This allows us to derive precise security estimates for our PCF; as a byproduct, our script can also be used to revisit the parameter selection of previous works, and compare empirically the strength of various instantiations. In particular, since our construction uses a recursive approach with a different parameter set per level, we optimize efficiency by selecting the best

possible sparsity parameter independently for each level. Second, we show that our PCF admits a natural time-sample tradeoffs: when generating more samples, the per-sample cost decreases significantly. This feature was not present in previous candidates. Third, the concrete efficiency of our PCF is maximized when we allow the participants to maintain a short state, i.e., a counter that parties increment after generating each sample. We introduce a simple strategy to obtain a stateless variant with the same concrete efficiency, at the cost of tweaking slightly the noise distribution in sparse LPN.

We provide a comparison of our new PCF to the state of art in Table 1. There we use GAR to denote the Goldreich-Applebaum-Raykov weak pseudorandom function [AR16], BIPSW denotes the candidate from [BIPSW18], and pDDH the power DDH assumption. Given a bound $m = 2^{30}$ on the number of samples, our new construction outperforms all previous PCFs, even those based on immature assumptions, by a significant margin; for $m = 2^{35}$, it remains faster than the fastest known PCF. For larger values of m, its efficiency decreases quickly: generating 2^5 times more samples is about 6 times slower. Nevertheless, its efficiency in realistic parameter ranges coupled with its security from well-established assumptions makes it an appealing choice for real-world applications. The numbers reported on Table 1 are based on plausible but somewhat optimistic parameters; we also report more conservative parameters on Table 3 (that still provide good performances). Eventually, we observe that in a batch setting, where a large number of PCF evaluations are computed simultaneously, our PCF shines even more: to provide a single data point, we estimate a throughput of about 245k evaluations per second when making batches of 10k evaluations, almost a 2.5× improvement over the non-batched version. This tradeoff is inherent to our construction: other PCFs do not benefit from computing PCF evaluations in batches. We note that when using a PCF for securely computing a circuit, the evaluation of the PCF in large batches (of size proportional to the circuit size) is actually the default use case.

Table 1. Comparison between our new PCF with existing PCF construction for 2^{30} or 2^{35} samples. For all constructions based on LPN variants, runtimes estimates are extrapolated by counting the number of PRG calls and using the benchmark of [MSY21] for the cost of instantiating the PRG with fixed-key AES on a machine with hardware support for AES. The estimates ignore other costs and could be off by a small factor, but provide apple-to-apple comparison among the LPN-based variants. The cost of the DCR and QR-based variants is extrapolated by counting exponentiations and using standard benchmarks for exponentiations over RSA groups.

Construction	Assumptions⋆	PQ	Correlations/s		VOLE	OT
			$m = 2^{30}$	$m = 2^{35}$		
[Boy+20a]	LWE	✓	✗	✗	✓	✓
[Boy+20a; CD23]	VDLPN	✓	1k	667	✓	✓
[Boy+22]	EALPN	✓	118k†	84k	✓	✓
[OSY21]	DCR	✗	100	100	✓	✗
[OSY21]	QR	✗	1	1	✓	✓
[BCMPR24]	BIPSW + pDDH	✗	15k‡	15k‡	✗	✓
[BCMPR24]	GAR + pDDH	✗	21k‡	21k‡	✗	✓
Our work	Sparse LPN	✓	507k	100k	✓	✓

⋆ All constructions additionally assume the ROM.
† [Boy+22] also provides an *aggressive* parameter set using a heuristic reasoning based on the assumption that codewords of relative weight less than 0.02 are hard to find in Expand-Accumulate codes, but this assumption was later invalidated in [RRT23].
‡ Throughput measured on one core of an M1 Pro, taken from [CDDKS24, Table 2].

1.4 On the Sparse LPN Assumption

The learning parity with noise (LPN) assumption with dimension n, number of samples m, and noise weight t states that given $(\mathbf{A}, \mathbf{A} \cdot \mathbf{s} + \mathbf{e})$, where $\mathbf{A} \xleftarrow{\$} \mathbb{F}_2^{m \times n}$, $\mathbf{s} \xleftarrow{\$} \mathbb{F}_2^n$, and $\mathbf{e}$ is a random sparse vector (typically either with entries sampled independently from a Bernoulli distribution with probability $p = t/m$, or sampled uniformly from the set of t-sparse vectors), it is infeasible to retrieve $\mathbf{s}$. In coding-theoretic

terms, LPN with a bounded number of samples is the problem of decoding a noisy codeword for a random linear code. The decision version states that given $\mathbf{A}$, it is hard to distinguish whether $\mathbf{b} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$ or $\mathbf{b} \xleftarrow{\$} \mathbb{F}_2^m$.

The sparse LPN assumption is a variant of LPN where the matrix $\mathbf{A}$ is sparse: its rows are independently sampled as random k-sparse vectors, for some sparsity parameter k. The sparse LPN assumption was formally introduced in [Ale03], but its roots are much older: in coding theory, it is the problem of decoding noisy codewords from the dual of an LDPC codes. Since its introduction, sparse LPN has found a wide variety of applications, such as hardness of approximation [Ale03], local pseudorandom generators [AIK06] hardness of learning [BCM25], public-key encryption [ABW10], pseudorandom correlation generators [BCGI18], efficient zero-knowledge proofs [YSWW21], homomorphic secret sharing [DIJL23], somewhat homomorphic encryption [CHKV25], constant-overhead secure computation [BCM25], and indistinguishability obfuscation [RVV24]. Consequently, its security has been thoroughly investigated over the years [MST03; FKO06; Sch08; ABW10; BSV19; CRR21; DJ24]. In particular, [ABW10; BSV19] proved search-to-decision reductions for sparse LPN, and [CRR21], building upon earlier works [NN90; MST03], introduced a general heuristic to reason about the resistance of LPN variants (including sparse LPN) against a large class of attacks.

Our PCF is based on a conservative flavor of the sparse LPN assumption: namely, that sparse LPN with dimension n, number of samples $m = n^{1+\varepsilon}$ (where ε is an arbitrary constant), noise rate $t/m = n^{-\delta(n)}$, and sparsity $k(n) = \mathsf{polylog}(n)$, cannot be broken in time $\mathsf{poly}(n)$ for $\delta(n) = 1 - o(1)$ (where $o(1)$ can be an arbitrary function, e.g. $1/\log\log n$). The result holds for any standard LPN noise distribution (Bernoulli noise, fixed-weight noise, regular noise). In this regime, the best known attacks run in time $\exp(-\Omega(n^{o(1)}))$ except with probability at most $1/n^{\mathsf{polylog}(n)}$.[5]

For our concrete instantiations, as in all previous works on pseudorandom correlation generators [BCGI18; Boy+19b; Boy+19a; SGRR19; Boy+20c; YWLZW20; CRR21] and functions [Boy+20a; Boy+22], we use a regular noise distribution, where the weight-t noise is sampled as a concatenation of t unit vectors. The hardness of LPN with regular noise has also a long history in cryptography: it was introduced in [AFS03] and received a significant attention from cryptanalysts [FGS07; MDCY11; BLPS11; BØ23; ES24; LWYY24; NMSÜ25]. Beyond PCGs and PCFs, it has also been used to design secure computation protocols [HOSS18] and signature schemes [CCJ23; BCCGJ24; Cui+24; OTX24]. In fact, the security of sparse LPN with regular noise is already assumed in several deployed solutions and commercial products [Prib; Eth; zkP23; Pria].

1.5 Technical Overview

Our starting point is the "primal" construction of pseudorandom correlation generator from [BCGI18], that we briefly recall. It relies on distributed multi-point functions (DMPF), a primitive that can be constructed efficiently from a length-doubling PRG [GI14; BGI15; BGI16]. At a high level, an DMPF allows to compress a t-sparse length-m vector $\mathbf{e}$ into a pair of keys (k_0, k_1) of size $O(\lambda \cdot t \cdot \log m)$ (where λ is some security parameter) such that two parties can non-interactively compute shares of $\mathbf{e}_i$, given respectively k_0, k_1 along with a common index i while the keys reveal no information about $\mathbf{e}$.

Given an DMPF, [BCGI18] introduces a PCG for the vector oblivious linear evaluation (VOLE) correlation. A *subfield* VOLE correlation of dimension n over a field $\mathbb{F}_{p^r}$ consists of a random tuple $((\mathbf{u}, \mathbf{v}), (\Delta, \mathbf{w})) \in (\mathbb{F}_p^n \times \mathbb{F}_{p^r}^n) \times (\mathbb{F}_{p^r} \times \mathbb{F}_{p^r}^n)$ such that $\mathbf{w} - \mathbf{v} = \Delta \cdot \mathbf{u}$. Now, given a vector $\mathbf{x}$, let $\langle \mathbf{x} \rangle = (\langle \mathbf{x} \rangle_0, \langle \mathbf{x} \rangle_1)$ denote subtractive shares of $\Delta \cdot \mathbf{x}$: a VOLE correlation can therefore be written as $[\![\mathbf{u}]\!] := ((\mathbf{u}, \langle \mathbf{u} \rangle_0), (\Delta, \langle \mathbf{u} \rangle_1))$. The construction of [BCGI18] proceeds as follows: the PCG key consists of a short VOLE correlation $[\![\mathbf{s}]\!] = \big((\mathbf{s}, \langle \mathbf{s} \rangle_0), (\Delta, \langle \mathbf{s} \rangle_1)\big)$ of size n, together with DMPF keys (k_0, k_1) succinctly encoding shares of $\Delta \cdot \mathbf{e}$ for $\Delta \xleftarrow{\$} \mathbb{F}_{p^r}$ and a random t-sparse vector $\mathbf{e} \in \mathbb{F}_p^m$. The first party is additionally given $\mathbf{e}$, and the second is given Δ. Using the DMPF, the parties compute shares $\langle \mathbf{e} \rangle$ of $\Delta \cdot \mathbf{e}$ and expand the short VOLE correlation $[\![\mathbf{s}]\!]$ into a longer (pseudorandom) correlation $[\![\mathbf{u}]\!] = \mathbf{A} \cdot [\![\mathbf{s}]\!] + [\![\mathbf{e}]\!]$ by encoding the vector with a random linear code $\mathbf{A} \in \mathbb{F}_p^{m \times n}$, where $m > n$, and adding the sparse error VOLE correlation $[\![\mathbf{e}]\!]$. Under the hardness of LPN with matrix $\mathbf{A}$, $\mathbf{u} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$ is indistinguishable from random

[5] The inverse-superpolynomial failure probability corresponds to the possibility that $\mathbf{A}$ is a "bad" matrix that admits an easy distinguisher: this holds for example whenever two rows of $\mathbf{A}$ are identical, which happens with probability $1/n^{O(k)}$.

from the viewpoint of the second party. Ignoring logarithmic factors and $\mathsf{poly}(\lambda)$ factors, the key size grows with $t + n$, and the PCG produces a correlation of length at most m.

When using a k-sparse matrix $\mathbf{A}$, security reduces to the sparse LPN assumption, and the PCG allows evaluating each entry of the pseudorandom correlation in time independent of m (concretely, each entry is computed from k entries of $[\![\mathbf{s}]\!]$ and evaluating the DMPF on the entry index). The main limitation is the stretch: for security, LPN requires $t \cdot n \gg m$ (otherwise, n randomly chosen equations are likely to be noise-free, which implies a trivial attack via Gaussian elimination), hence this PCG can at most provide a sub-quadratic expansion.

Our Approach. Our approach to obtain a longer stretch is simply to iterate this construction L times as follows:

$$[\![\mathbf{u}]\!] = \mathbf{A}^{(L)} \cdot \Bigg(\underbrace{\cdots \Big(\mathbf{A}^{(2)} \cdot \underbrace{\big(\mathbf{A}^{(1)} \cdot [\![\mathbf{s}^{(0)}]\!] + [\![\mathbf{e}^{(1)}]\!] \big)}_{=:[\![\mathbf{s}^{(1)}]\!]} + [\![\mathbf{e}^{(2)}]\!] \Big) \cdots}_{\substack{=:[\![\mathbf{s}^{(2)}]\!] \\ \vdots \\ =:[\![\mathbf{s}^{(L-1)}]\!]}} \Bigg) + [\![\mathbf{e}^{(L)}]\!].$$

Using *sparse* LPN, we can choose the matrices $\mathbf{A}^{(\ell)}$ for $\ell = 1, \ldots, L$ to be sparse, i.e., such that each row of $\mathbf{A}^{(\ell)}$ has k non-zero entries. Thus, one entry of $[\![\mathbf{u}]\!]$ can be computed from only $k^{L-\ell+1}$ entries of the intermediate secret vector $[\![\mathbf{s}^{(\ell-1)}]\!]$ for any level $\ell \in [L]$, and in particular, from k^L entries of $[\![\mathbf{s}^{(0)}]\!]$.

For simplicity of exposition, let us fix a security parameter λ, a noise weight $t = \mathsf{poly}(\lambda)$, and assume that we set $m = n \cdot t/\lambda$ (this is in line with known cryptanalysis that run in time $\exp(\Omega(n \cdot t/m)) = \exp(\Omega(\lambda))$) at each of the L levels of the recursion. With this choice of parameters, starting with a secret vector of size $n = m^{(0)}$, each iteration expands it by a factor of $\frac{t}{\lambda}$ to length $m^{(\ell)} = m^{(\ell-1)} \cdot \frac{t}{\lambda}$. The key observation is that for such choices of parameters, we can achieve superpolynomial stretch whenever $k = \mathsf{polylog}(n)$: setting $L = \log n / \log k = \Omega(\log n / \log\log n)$, each entry can be individually computed in time $k^L = n$ while the length of the final correlation is $m^{(L)} = n \cdot (t/\lambda)^L = \mathsf{poly}(\lambda)^{\log n / \log\log n}$, which is superpolynomial.

The above iterative construction does not immediately yield a PCF, as storing the matrices is infeasible whenever the stretch becomes superpolynomial. To obtain a PCF from LPN-style assumptions, previous works [Boy+20a; Boy+22] would first build a *weak* PCF, where outputs $y_b = \mathsf{Eval}(b, K_B, x)$ look like random samples from the target correlation provided that the input x is random. Then, x is used as a seed to sample a random row of the LPN matrix. However, in our setting, this does not quite work, as we must sample a row of the product $\prod_{\ell=0}^{r} \mathbf{A}^{(L-\ell)} = \mathbf{A}^{(L)} \cdot \mathbf{A}^{(L-1)} \cdots \mathbf{A}^{(L-r)}$ for $r = 0$ to $L-1$, from x. In particular, the challenge is to ensure consistency across inputs: all sampled vectors must consistently be rows of $\prod_\ell \mathbf{A}^{(\ell)}$ with respect to random sparse matrices $\mathbf{A}^{(\ell)}$ (specified on-the-fly).

To address this issue, we adopt the following strategy: we define the i-th row of each matrix $\mathbf{A}^{(\ell)} \in \mathbb{F}_p^{m^{(\ell)} \times m^{(\ell-1)}}$ with $\ell \in [L-1]$ via a random oracle $\mathsf{H}_A^{(\ell)}$ as

$$\mathbf{A}_i^{(\ell)} = \big((j_1, a_{i,j_1}), \ldots, (j_k, a_{i,j_k}) \big) \leftarrow \mathsf{H}_A^{(\ell)}(i)$$

such that $\mathbf{A}^{(\ell)} \sim \mathsf{SparseCodeGen}(k, m^{(\ell)}, m^{(\ell-1)})$. Moreover, to avoid restricting the domain of our PCF to be $m^{(L)}$ and allow a larger domain $\mathcal{X}$ instead, we build the matrix $\mathbf{A}^{(L)}$ for the outermost level, dynamically by using the random oracle to sample one row $\mathbf{A}_x^{(L)} \leftarrow \mathsf{H}_A^{(L)}(x)$ for every input x. Note that if we restrict the number of allowed queries to be at most $m^{(L)}$, we still obtain a matrix $\mathbf{A}^{(L)} \in \mathbb{F}_p^{m^{(L)} \times m^{(L-1)}}$ with the correct distribution.

The key observation is that once the matrices are defined this way, given an input x, the corresponding row of each product $\prod_{\ell=0}^{r} \mathbf{A}^{(L-\ell)}$ (for $r = 0$ to $L-1$) can be computed using a k-ary hash tree:

Level L: Set $\mathbf{A}_x^{(L)} = \big((j_{L,1}, a^{(L)}_{x|j_{L,1}}), \ldots, (j_{L,k}, a^{(L)}_{x|j_{L,k}}) \big) \leftarrow \mathsf{H}_A^{(L)}(x)$.

Level $L-1$: For $i_1 \in [k]$, set $\mathbf{A}_{j_{L,i_1}}^{(L-1)}$ as

$$\big((j_{L-1,i_1,1}, a^{(L-1)}_{j_{L,i_1}|j_{L-1,i_1,1}}), \ldots, (j_{L-1,i_1,k}, a^{(L-1)}_{j_{L,i_1}|j_{L-1,i_1,k}}) \big) \leftarrow \mathsf{H}_A^{(L-1)}(j_{L,i_1}).$$

...

Level $L-\ell$, $\ell \in [L-1]$: For all $\mathbf{i} = (i_1, \ldots, i_\ell) \in [k]^\ell$, set $\mathbf{A}^{(L-\ell)}_{j_{L-\ell+1,\mathbf{i}}}$ as

$$\left((j_{L-\ell,\mathbf{i},1}, a^{(L-\ell)}_{j_{L-\ell+1,\mathbf{i}}|j_{L-\ell,\mathbf{i},1}}), \ldots, (j_{L-\ell,\mathbf{i},k}, a^{(L-\ell)}_{j_{L-\ell+1,\mathbf{i}}|j_{L-\ell,\mathbf{i},k}})\right) \leftarrow \mathsf{H}^{(L-\ell)}_A(j_{L-\ell+1,\mathbf{i}}).$$

To compute the k^r-sparse vector corresponding to the xth row $\mathbf{a}^{(L-r)}_x$ of the product $\prod_{\ell=0}^{r} \mathbf{A}^{(L-\ell)}$, i.e., $\mathbf{a}^{(L-r)}_x = \mathbf{A}^{(L)}_x \cdot \prod_{\ell=1}^{r} \mathbf{A}^{(L-\ell)}$, for $r \in [1, L)$ one proceeds as follows:

Case $\mathbb{F}_2$: After traversing the tree to level $L-r$, $\mathbf{a}^{(L-r)}_x$ is computed by xoring the vectors $\mathbf{A}^{(L-r)}_{j_{L-r+1,\mathbf{i}}}$ for all $\mathbf{i} = (i_1, \ldots, i_r) \in [k]^r$.

Case $\mathbb{F}_p$: For general fields $\mathbb{F}_p$, we start with $\mathbf{a}^{(L)}_x := \mathbf{A}^{(L)}_x$ and iteratively compute, for $r \in [1, L)$, $\mathbf{a}^{(L-r)}_x = \mathbf{a}^{(L-r+1)}_x \cdot \mathbf{A}^{(L-r)}$ as a linear combination of k^{r-1} vectors which each are k-sparse.

Given an arbitrary input x, this k-ary hash tree can be computed on-the-fly in time at most $k^L = \mathsf{poly}(n)$, hence the resulting PCF is efficient. Eventually, from the input x, we also compute the entry of $[\![\mathbf{e}^{(L)}]\!]$ to add to the final result using a random oracle $\mathsf{H}_B(x)$. As $\mathbf{e}$ has superpolynomial size, it holds with overwhelming probability that all indices will be distinct; furthermore, since the matrix distribution of sparse LPN is invariant by shuffling of the rows, this does not change the underlying assumption even when the noise vector distribution is not (as for, e.g., a regular vector) since this translates to reordering the noisy samples (with a public reordering). Security follows via L hybrids (resulting in a factor-L security loss), invoking sparse LPN at each step to replace $[\![\mathbf{s}^{(r)}]\!]$ with a random vector, starting with $r = 1$. The right choice of L, is the smallest providing sufficiently many PCF outputs for an application.

PCF for Oblivious Transfer. Note that subfield VOLE over $\mathbb{F}_{2^\kappa}$ is identical to *correlated* OT of κ-bit strings. A correlation robust hash function can be used to convert such a correlation non-interactively into *random* OTs [IKNP03; Boy+19b].

A Concrete Analysis of Sparse LPN. An important component of the security analysis of sparse LPN lies in the proof that random sparse matrices have large *dual distance*: that is, all non-zero vectors in their kernel must have large Hamming weight. Indeed, given a low-weight vector $\mathbf{v}$ such that $\mathbf{v}^\intercal \cdot \mathbf{A} = 0$, it holds that $\mathbf{v}^\intercal \cdot (\mathbf{A} \cdot \mathbf{s} + \mathbf{e}) = \mathbf{v}^\intercal \cdot \mathbf{e}$ is zero with significant probability if $\mathbf{v}$ is sufficiently sparse, which provides a distinguisher. An important heuristic used in parameter selections is that having a large dual distance is also a sufficient condition for sparse LPN to be essentially as secure as standard LPN. We refer the reader to Section 4 for a more in-depth discussion on this heuristic and the rationale behind it.

Previous work relied on a loose asymptotic lower bound on the dual distance of random sparse matrices, which did not provide any meaningful way to efficiently compare between different parameter sets (as they might have vastly different concrete dual distance). In our setting, this is especially unsatisfying due to the recursive nature of our construction: we have L levels of recursion and a different parameter set at each level (though asymptotically, all levels can be implied by a single parameter set, this is suboptimal when picking concrete parameters). To remedy this situation, we provide a methodology to compute tight estimates of the dual distance $\mathsf{dd}(\mathbf{A})$ of a random sparse $m \times n$ matrix $\mathbf{A}$. At a high level, our approach works as follows: fix a vector $\mathbf{v}$ of Hamming weight w and consider $\Pr[\mathbf{v}^\intercal \cdot \mathbf{A}] = 0$ over the random choice of $\mathbf{A}$. This event is equivalent to sampling $k \cdot w$ unit vectors and xoring them[6]. To compute the probability $p_{w \cdot k, 0}$ of this event, we derive a general formula for computing the probability $p_{W,u}$ that W unit vectors xor to a vector of weight u. We do so by separating the first two unit vectors from the remaining $W-2$, and by a case analysis:

- (Case 1) the first two vectors xor to 0 (probability $1/n$). Then, the remaining $W-2$ vector must xor to u (probability $p_{W-2,u}$).
- (Case 2) the first two vectors xor to a weight-2 vector (probability $1 - 1/n$).
  - (Case 2.a) the $W-2$ remaining vectors xor to a weight-$(u-2)$ vector and the non-zero entries do not collide with the entries of the first two vectors (probability $p_{W-2,u-2} \cdot \binom{n-2}{u-2} / \binom{n}{u-2}$)

[6] This makes the simplifying assumption that the rows of $\mathbf{A}$ are sampled as the xor of k independent unit vectors of length n, which differs slightly from the uniform distribution over k-sparse vector. Similar simplifications are commonly done to guarantee independence and simplify analysis, and the bounds obtained are heuristically expected to be essentially identical when using random k-sparse rows.

- (Case 2.b) the $W-2$ remaining vectors xor to a weight-u vector and exactly one the non-zero entries collides with an entry of the first two vectors (probability $p_{W-2,u} \cdot 2 \cdot \binom{n-2}{u} / \binom{n}{u}$)
- (Case 2.c) the $W-2$ remaining vectors xor to a weight-$(u+2)$ vector and the non-zero entries collide with both entries of the first two vectors (probability $p_{W-2,u+2} \cdot \binom{n-2}{u} / \binom{n}{u+2}$)

From this analysis, we derive a recursive formula for computing $p_{W,0}$ explicitly, and derive a tight lower bound D on $\mathsf{dd}(\mathbf{A})$ via a union bound: D is the largest value such that $\Pr[\sum_{w=1}^{D} \binom{m}{w} \cdot p_{w \cdot k,0}] \leq 2^{\rho}$, where ρ is a statistical security parameter. We use this formula to compute tight security estimates for our parameter choices and compare them to previous instantiations of sparse LPN.

An important consequence of our tighter analysis is that it lets us choose *independently* the optimal value of the sparsity parameter k that suffices to achieve a given target dual distance $\mathsf{dd}(\mathbf{A})$ with probability $1 - 2^{\rho}$ for each level of the recursion. Hence, the computational cost of our procedure is reduced from $O(k^L)$ to $O(\prod_{\ell=1}^{L} k^{(\ell)})$, where the $k^{(\ell)}$ are the optimal k's for each level. This way we obtain up to a factor 2 improvement with our suggested parameter sets (see Table 3 for concrete numbers).

Additional Optimizations. We make the following simple observation: existing DMPF have the property that evaluating them on their full domain has a much lower per-evaluation cost than computing each evaluation independently (this stems from the fact that visiting all nodes of a complete binary tree of depth d requires visiting 2^{d+1} nodes, but going from the root to each leaf independently costs $d \cdot 2^d$ in total). In addition, in our construction, fully evaluating one level reduces significantly the per-evaluation cost: concretely, if the first level is fully computed and stored, each evaluation costs $O(\prod_{i=2}^{L} k^{(i)})$ instead of $O(\prod_{i=1}^{L} k^{(i)})$, saving a factor $k^{(1)}$. This suggests the following optimization: when users want to evaluate our PCF on a sufficiently large batch of values, it becomes more efficient to first precompute a full evaluation of the first level before evaluating the PCF. In our experiments, this approach improves over naive direct evaluation as soon as the users want to do as little as 1500 PCF evaluations (most applications require much more), and as the batch size increases, the speedup over naive evaluation approaches a factor 5.

Eventually, we revisit the use of a state in concrete PCF instantiations. Asymptotically, our PCF (as previous ones) is stateless. However, when targeting concrete parameters, an issue arises: on the L-th level our PCF requires adding an entry $e_i^{(L)}$ of a noise vector $\mathbf{e}^{(L)}$ to the output. In the asymptotic analysis, i is picked uniformly at random, and collisions occur only with negligible probability as $\mathbf{e}^{(L)}$ has superpolynomial size. However, when fixing a target bound of e.g. 2^{30} PCF evaluations, using random indices would create collisions after about 2^{15} evaluations, causing security issues. The usual strategy in previous works [Boy+20a; Boy+22] is to let users maintain a counter that they update and store after each evaluation. For settings where maintaining a state can be undesirable, we put forth an alternative: fix a very sparse matrix $\mathbf{B}$ such that $\mathbf{B}$ is full rank with high probability, and let the parties randomly pick entries of $\mathbf{B} \cdot \mathbf{e}^{(L)}$ as the noise of each evaluation. Collisions are now unlikely to occur as $\mathbf{B}$ is unlikely to be low rank, and security reduces to sparse LPN with a new *sparse-product* noise distribution (in light of all state-of-the-art cryptanalysis, this alternative noise distribution does not appear to have any impact on security, though of course caution is always warranted when using a new construction – this last optimization departs from our main construction, where the aim is to use only well-established assumptions). Concretely, to achieve *statelessness*, we replace $[\![\mathbf{e}^{(L)}]\!]$ with $\mathbf{B} \cdot [\![\mathbf{e}^{(L)}]\!]$ where $\mathbf{B} \in \mathbb{F}_p^{m^{(L)} \times m^{(L)}}$ is again a sparse matrix whose rows are computed by $\mathbf{B}_x \leftarrow \mathsf{H}_B(x)$, where x is a random input (sampled via coin tossing or obtained via a trusted source of randomness).

Open Questions. We mention two interesting open questions:

(1) Is it possible to build a PCF with fully-subexponential security (in runtime and queries) from sparse LPN? Our PCF only achieves subexponential security up to superpolynomially-many queries, a lower asymptotic security level compared to previous candidates.
(2) Is it possible to get rid of the random oracle model by settling for a *weak* PCF (that is only required to be secure on random inputs)? Intriguingly, all previous PCF candidates were secure in the standard model when targeting only weak security, as the random oracle was used only as a generic strategy to boost from weak to strong by hashing the input (the hash-then-evaluate paradigm [BCEKM24]). However, in our setting, the random oracle is used adaptively to sample rows of a product of matrices

via a k-ary hash tree, and it is unclear how to replace the RO with standard model primitives. The reduction from sparse-LPN requires programming the RO, and it is unclear how the reduction would work otherwise; for instance, using a PRF instead of a RO (with key known to all parties) requires a new "sparse-LPN with pseudorandom matrix" assumption, a plausible but non-standard variant of LPN. A more detailed discussion appears in [BCEKM24].

2 Preliminaries

2.1 Notation

We denote by λ and ρ the computational and statistical security parameters, respectively. We denote by $\mathsf{negl}(\lambda)$ a negligible function in λ and by $\mathsf{poly}(\lambda)$ a polynomial function in λ. PPT refers to probabilistic polynomial-time. For integers $a \leq b$ we write $[a, b] := \{a, \ldots, b\}$, $[a, b) := [a, b-1]$, and $[b] := [1, b]$. We use bold for vectors $\mathbf{v}$ and matrices $\mathbf{M}$, and $\omega(\mathbf{v})$ for the Hamming weight of $\mathbf{v}$.

2.2 Learning Parity with Noise

We consider the so-called primal variant of learning parity with noise (LPN) that we parametrize by the distributions of the linear code and of the noise vector: Let $\mathsf{CodeGen}$ be a probabilistic polynomial time code generation algorithm such that $\mathsf{CodeGen}(m, n, \mathbb{F})$ outputs a matrix $\mathbf{A} \in \mathbb{F}^{m \times n}$, and let $\mathsf{Noise}_t(m, \mathbb{F})$ be an efficiently sampleable noise distribution which outputs t-sparse vectors in $\mathbb{F}^m$. We denote by $\mathcal{V}_k^n(\mathbb{F})$ the set of vectors in $\mathbb{F}^n$ with Hamming weight k. For a vector $\mathbf{v} \in \mathcal{V}_k^n(\mathbb{F})$, we parse $\mathbf{v} = ((\alpha_i, \beta_i))_{i \in [k]}$ which means that at index $\alpha_i \in [n]$, the non-zero value of $\mathbf{v}$ is β_i. We denote by $\mathcal{M}_k^{m \times n}(\mathbb{F})$ the set of $m \times n$ matrices that are k-sparse, i.e., where each row is in $\mathcal{V}_k^n(\mathbb{F})$.

Definition 1 (Regular Noise Distribution). Let $t, m \in \mathbb{N}$ such that $t \mid m$. Define $\mathsf{Reg}_t(m, \mathbb{F})$ to be the uniform distribution of vectors $\mathbf{e} = [\mathbf{e}_1, \ldots, \mathbf{e}_t] \in \mathbb{F}^m$ conditioned on each $\mathbf{e}_i \in \mathbb{F}^{m/t}$ having Hamming weight $\omega(\mathbf{e}_i) = 1$.

Definition 2 (Sparse-Product Noise Distribution). Let $t, m, \beta \in \mathbb{N}$ such that $t \mid m$. and $\beta \in [m]$. Let $D^{\mathsf{sp}}_{\mathsf{Noise}_t, \beta}(m, \mathbb{F})$ be the distribution over $\mathbb{F}^{m \times m} \times \mathbb{F}^m$ obtained by sampling $\mathbf{B} \xleftarrow{\$} \mathcal{M}_\beta^{m \times m}(\mathbb{F})$ and $\mathbf{e} \leftarrow \mathsf{Noise}_t(m, \mathbb{F})$, and outputting $(\mathbf{B}, \mathbf{B} \cdot \mathbf{e})$.

Definition 3 (Sparse Random Code). Let $k, n, m \in \mathbb{N}$ such that $3 \leq k \leq n$. Define $\mathsf{SparseCodeGen}(k, m, n, \mathbb{F})$ to output a random matrix $\mathbf{A} \xleftarrow{\$} \mathcal{M}_k^{m \times n}(\mathbb{F})$.

Definition 4 (Regular Random Code). Let $k, n, m \in \mathbb{N}$ such that $3 \leq k \leq n$ and $k \mid n$. Define $\mathsf{RegularCodeGen}(k, m, n, \mathbb{F})$ to output a random matrix $\mathbf{A} \in \mathbb{F}^{m \times n}$, such that for every $j \in [m]$ row $\mathbf{a}_j = [\mathbf{a}_{j,1}, \mathbf{a}_{j,2}, \ldots, \mathbf{a}_{j,k}]$ is the concatenation of k vectors $\mathbf{a}_{j,i} \in \mathbb{F}^{n/k}$ with Hamming weights $\omega(\mathbf{a}_{j,i}) = 1$.

Definition 5 (Learning Parity with Noise (LPN) Assumption). Let λ be a security parameter, secret size $n = n(\lambda)$, output size $m = m(\lambda)$, noise sparsity $t = t(\lambda)$. Let $\mathbb{F}$ be a finite field, $\mathsf{CodeGen}$ a code generation algorithm, and Noise a noise distribution. We say that the Learning Parity with Noise (LPN) problem is hard relative to $\mathsf{CodeGen}$ and Noise if

$$(\mathbf{A}, \mathbf{A} \cdot \mathbf{s} + \mathbf{e}) \approx_c (\mathbf{A}, \mathbf{y}),$$

where $\mathbf{A} \leftarrow \mathsf{CodeGen}(m, n, \mathbb{F})$, $\mathbf{s} \xleftarrow{\$} \mathbb{F}^n$, $\mathbf{e} \leftarrow \mathsf{Noise}_t(m, \mathbb{F})$, and $\mathbf{y} \xleftarrow{\$} \mathbb{F}^m$. We denote by $\mathsf{LPN}(n, m, \mathsf{CodeGen}, \mathsf{Noise}_t, \mathbb{F})$ the assumption with these parameters and introduce the following specific variants of the assumption:

- When instantiated with $\mathsf{SparseCodeGen}(k, \cdot, \cdot, \cdot)$ and Reg_t, we call it *sparse* LPN with *regular* noise and denote it by $\mathsf{SparseLPN}(k, \cdot, \cdot, \mathsf{Reg}_t, \cdot)$ for some $k \in [3, n]$.
- When instantiated with $D^{\mathsf{sp}}_{\mathsf{Noise}_t, \beta}$ for some $\beta \in [1, m]$, the assumption states that the following two tuples are indistinguishable

$$(\mathbf{A}, \mathbf{A} \cdot \mathbf{s} + \mathbf{B} \cdot \mathbf{e}, \mathbf{B}) \approx_c (\mathbf{A}, \mathbf{y}, \mathbf{B}),$$

where $(\mathbf{B}, \mathbf{B} \cdot \mathbf{e}) \xleftarrow{\$} D^{\mathsf{sp}}_{\mathsf{Noise}_t, \beta}(m, \mathbb{F})$, and $\mathbf{A}, \mathbf{s}$ and $\mathbf{y}$ are defined as before.

2.3 Distributed Point Functions

Definition 6 (t-Point Function). Let $n, t \in \mathbb{N}$ and $\mathbb{G}$ be an additive group. Let $(\boldsymbol{\alpha}, \boldsymbol{\beta}) = (\alpha_i, \beta_i)_{i\in[t]}$, with $\alpha_i \in \{0,1\}^n$ distinct and $\beta_i \in \mathbb{G}$. A t-point function $f_{(\boldsymbol{\alpha},\boldsymbol{\beta})} \colon \{0,1\}^n \to \mathbb{G}$ is defined as follows:

$$f_{(\boldsymbol{\alpha},\boldsymbol{\beta})}(x) = \begin{cases} \beta_i & \text{if } x = \alpha_i \text{ for some } i \in [t], \\ 0 & \text{otherwise.} \end{cases}$$

Definition 7 (Distributed Multi-Point Function, [Boy+20b, Def 5.1 (adapted)]). A distributed multi-point function (DMPF) for the class of t-point functions

$$\mathcal{F} = \{f_{(\boldsymbol{\alpha},\boldsymbol{\beta})} : \{0,1\}^{n(\lambda)} \to \mathbb{G} \mid (\boldsymbol{\alpha}, \boldsymbol{\beta}) = (\alpha_i, \beta_i)_{i\in[t]} \in (\{0,1\}^{n(\lambda)}, \mathbb{G})^t\}$$

is a pair of PPT algorithms $(\mathsf{DMPF.Gen}, \mathsf{DMPF.Eval})$ with the following syntax:

$\mathsf{DMPF.Gen}(1^\lambda, f_{(\boldsymbol{\alpha},\boldsymbol{\beta})}, n)$ is a PPT algorithm that on input 1^λ, the t-point function $f_{(\boldsymbol{\alpha},\boldsymbol{\beta})}$, and domain length $n = n(\lambda)$, outputs a key pair $(k_0^{\mathsf{DMPF}}, k_1^{\mathsf{DMPF}})$.

$\mathsf{DMPF.Eval}(\sigma, k_\sigma^{\mathsf{DMPF}}, x)$ is a deterministic PT algorithm that takes as input $\sigma \in \{0,1\}$, the associated key k_σ, and the input $x \in \{0,1\}^n$, and outputs $y_\sigma \in \mathbb{G}$.

The scheme should satisfy the following properties:

- **Correctness:** For any t-point function $f_{(\boldsymbol{\alpha},\boldsymbol{\beta})} \in \mathcal{F}$ and $x \in \{0,1\}^n$:

$$\Pr[y_0 + y_1 = f_{(\boldsymbol{\alpha},\boldsymbol{\beta})}(x) \mid (k_0^{\mathsf{DMPF}}, k_1^{\mathsf{DMPF}}) \leftarrow \mathsf{DMPF.Gen}(1^\lambda, f_{(\boldsymbol{\alpha},\boldsymbol{\beta})}, n)] = 1$$

  where $y_0 \leftarrow \mathsf{DMPF.Eval}(0, k_0^{\mathsf{DMPF}}, x)$ and $y_1 \leftarrow \mathsf{DMPF.Eval}(1, k_1^{\mathsf{DMPF}}, x)$.
- **Security:** For any $\sigma \in \{0,1\}$, there exists a PPT simulator Sim such that for any polynomial-size $f_{(\boldsymbol{\alpha},\boldsymbol{\beta})} \in \mathcal{F}$,

$$\{(\sigma, f_{(\boldsymbol{\alpha},\boldsymbol{\beta})}, k_\sigma^{\mathsf{DMPF}}) \mid (k_0^{\mathsf{DMPF}}, k_1^{\mathsf{DMPF}}) \leftarrow \mathsf{DMPF.Gen}(1^\lambda, f_{(\boldsymbol{\alpha},\boldsymbol{\beta})}, n)\}$$

  is computationally indistinguishable from the case where instead $k_\sigma^{\mathsf{DMPF}} \leftarrow \mathsf{Sim}(1^\lambda, \mathsf{Leak}_\sigma(f_{(\boldsymbol{\alpha},\boldsymbol{\beta})}))$, and $\mathsf{Leak}_\sigma(f_{(\boldsymbol{\alpha},\boldsymbol{\beta})})$ leaks the domain and range of the multi-point function by default, i.e. $(n, \mathbb{G})$.
- **Pseudorandom Outputs:** Let $n \in \mathbb{N}$. For any $\sigma \in \{0,1\}$, $N = \mathsf{poly}(\lambda)$, poly-size t-point function $f = f_{(\boldsymbol{\alpha},\boldsymbol{\beta})} \in \mathcal{F}$, and set of n-bit inputs $(x_i)_{i\in[N]}$, the following distribution is indistinguishable from the uniform distribution over $\mathbb{G}^N$:

$$\{(y_i)_{i\in[N]} \mid y_i \leftarrow \mathsf{DMPF.Eval}(\sigma, k_\sigma^{\mathsf{DMPF}}, x_i)\}$$

  where $(k_0^{\mathsf{DMPF}}, k_1^{\mathsf{DMPF}}) \leftarrow \mathsf{DMPF.Gen}(1^\lambda, f_{(\boldsymbol{\alpha},\boldsymbol{\beta})}, n)$.

For better concrete efficiency of our PCF, we can use *known-index* DMPFs that reveal $\boldsymbol{\alpha}$ to one of the parties.

2.4 Pseudorandom Correlation Functions

We recall the definition of pseudorandom correlation functions (PCF) for reverse-sampleable correlations. At a high level, a correlation $\mathcal{Y}$ is reverse-sampleable if pairs sampled from $\mathcal{Y}$ are indistinguishable from simulated samples output by $\mathcal{Y}.\mathsf{RSample}$. An overview of PCFs can be found in Section 1.1.

Definition 8 (Reverse-Sampleable Correlation). Let $1 \le \ell_0(\lambda), \ell_1(\lambda) \le \mathsf{poly}(\lambda)$ be output-length functions. Let $\mathcal{Y}$ be a probabilistic algorithm that, on input 1^λ, returns a pair of outputs $(y_0, y_1) \in \{0,1\}^{\ell_0(\lambda)} \times \{0,1\}^{\ell_1(\lambda)}$, defining a correlation on the outputs.

We say that $\mathcal{Y}$ defines a *reverse-sampleable* correlation if there exists a probabilistic polynomial time algorithm $\mathcal{Y}.\mathsf{RSample}$ which takes as input 1^λ, $\sigma \in \{0,1\}$, and $y_\sigma \in \{0,1\}^{\ell_\sigma(\lambda)}$, and outputs $y_{1-\sigma} \in \{0,1\}^{\ell_{1-\sigma}(\lambda)}$, such that for all $\sigma \in \{0,1\}$ the following distributions are statistically close:

$$\{(y_0, y_1) \colon (y_0, y_1) \xleftarrow{\$} \mathcal{Y}(1^\lambda)\}, \text{ and}$$
$$\{(y_0, y_1) \colon (y_0', y_1') \xleftarrow{\$} \mathcal{Y}(1^\lambda), y_\sigma \leftarrow y_\sigma', y_{1-\sigma} \leftarrow \mathcal{Y}.\mathsf{RSample}(1^\lambda, \sigma, y_\sigma)\}.$$

Definition 9 (Pseudorandom Correlation Function, [Boy+20a, Def. 4.4]). Let $\mathcal{Y}$ be a reverse-sampleable correlation with output length functions $\ell_0(\lambda), \ell_1(\lambda)\colon \mathbb{N} \to \mathbb{N}$, and for $\lambda \leq n(\lambda) \leq \mathsf{poly}(\lambda)$, let $n(\lambda)\colon \mathbb{N} \to \mathbb{N}$ be an input length function. Let $\mathcal{Y}.\mathsf{RSample}$ be the reverse sampling algorithm of $\mathcal{Y}$ and $(\mathsf{PCF}.\mathsf{Gen}, \mathsf{PCF}.\mathsf{Eval})$ be a pair of algorithms with the following syntax:

$\mathsf{PCF}.\mathsf{Gen}(1^\lambda)$ is a PPT algorithm that on input 1^λ, outputs a pair of keys $(k_0^{\mathsf{PCF}}, k_1^{\mathsf{PCF}})$. We assume that λ can be inferred from the keys.

$\mathsf{PCF}.\mathsf{Eval}(\sigma, k_\sigma^{\mathsf{PCF}}, x)$ is a deterministic PT algorithm that on input $\sigma \in \{0,1\}$, a key k_σ^{PCF} and an input value $x \in \{0,1\}^{n(\lambda)}$, outputs $y_\sigma \in \{0,1\}^{\ell_\sigma(\lambda)}$.

Let $N, B\colon \mathbb{N} \to \mathbb{N}$ and $\varepsilon(\lambda)\colon \mathbb{N} \to [0,1]$. We say that $(\mathsf{PCF}.\mathsf{Gen}, \mathsf{PCF}.\mathsf{Eval})$ is an (N, B, ε)-secure *(strong) pseudorandom correlation function (PCF)* for $\mathcal{Y}$, if the following conditions hold for the security games in Figure 1:

- **Pseudorandom $\mathcal{Y}$-correlated outputs:** For every non-uniform adversary $\mathcal{A}$ of size $B(\lambda)$, it holds that for all large enough $\lambda \in \mathbb{N}$,
$$|\Pr[\mathsf{Exp}^{\mathsf{pr}}_{\mathcal{A},0}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{pr}}_{\mathcal{A},1}(\lambda) = 1]| \leq \varepsilon(\lambda)$$
where $\mathcal{A}$ makes at most $N(\lambda)$ queries to the oracle $\mathcal{O}_b^{\mathsf{pr}}$.
- **Security:** For every $\sigma \in \{0,1\}$ and every non-uniform adversary $\mathcal{A}$ of size $B(\lambda)$, it holds that for all large enough $\lambda \in \mathbb{N}$,
$$|\Pr[\mathsf{Exp}^{\mathsf{sec}}_{\mathcal{A},\sigma,0}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{sec}}_{\mathcal{A},\sigma,1}(\lambda) = 1]| \leq \varepsilon(\lambda)$$
where $\mathcal{A}$ makes at most $N(\lambda)$ queries to $\mathcal{O}_b^{\mathsf{sec}}$ oracle.

We call $(\mathsf{PCF}.\mathsf{Gen}, \mathsf{PCF}.\mathsf{Eval})$ *secure* if it is $(\mathsf{poly}, \mathsf{poly}, \mathsf{negl})$-secure.

Figure 1: PCF Security Games

Let $(\mathsf{Gen}, \mathsf{Eval})$ be a PCF for a reverse-sampleable correlation $\mathcal{Y}$.

(Strong) Pseudorandom $\mathcal{Y}$-Correlated Outputs

$\underline{\mathsf{Exp}^{\mathsf{pr}}_{\mathcal{A},b}(\lambda)}$:

$(k_0^{\mathsf{PCF}}, k_1^{\mathsf{PCF}}) \leftarrow \mathsf{Gen}(1^\lambda)$
$\mathcal{Q} := \emptyset$
$b' \leftarrow \mathcal{A}^{\mathcal{O}_b^{\mathsf{pr}}(\cdot)}(1^\lambda)$
Output b'

$\underline{\mathcal{O}_0^{\mathsf{pr}}(x)}$:

If $(x, y_0, y_1) \in \mathcal{Q}$**:**
  Return (y_0, y_1)
Else:
  $(y_0, y_1) \leftarrow \mathcal{Y}(1^\lambda)$
  $\mathcal{Q} := \mathcal{Q} \cup \{(x, y_0, y_1)\}$
Return (y_0, y_1)

$\underline{\mathcal{O}_1^{\mathsf{pr}}(x)}$:

For $\sigma \in \{0,1\}$**:**
  $y_\sigma \leftarrow \mathsf{Eval}(1^\lambda, \sigma, k_\sigma^{\mathsf{PCF}}, x)$
Return (y_0, y_1)

(Strong) PCF Security

$\underline{\mathsf{Exp}^{\mathsf{sec}}_{\mathcal{A},\sigma,b}(\lambda)}$:

$(k_0^{\mathsf{PCF}}, k_1^{\mathsf{PCF}}) \leftarrow \mathsf{Gen}(1^\lambda)$
$\mathcal{Q} := \emptyset$
$b' \leftarrow \mathcal{A}^{\mathcal{O}_b^{\mathsf{sec}}(\cdot)}(1^\lambda, \sigma, k_\sigma^{\mathsf{PCF}})$
Output b'

$\underline{\mathcal{O}_0^{\mathsf{sec}}(x)}$:

$y_\sigma \leftarrow \mathsf{Eval}(\sigma, k_\sigma^{\mathsf{PCF}}, x)$
$y_{1-\sigma} \leftarrow \mathcal{Y}.\mathsf{RSample}(1^\lambda, \sigma, y_\sigma)$
Return $y_{1-\sigma}$

$\underline{\mathcal{O}_1^{\mathsf{sec}}(x)}$:

$y_{1-\sigma} \leftarrow \mathsf{Eval}(1-\sigma, k_{1-\sigma}^{\mathsf{PCF}}, x)$
Return $y_{1-\sigma}$

2.5 Vector Oblivious Linear Evaluation

Definition 10 ((Random) Subfield VOLE Correlation). A subfield vector oblivious linear evaluation (VOLE) correlation of dimension n over a finite field $\mathbb{F}_{p^r}$ consists of a tuple $((\mathbf{u}, \mathbf{v}), (\Delta, \mathbf{w})) \in (\mathbb{F}_p^n \times \mathbb{F}_{p^r}^n) \times (\mathbb{F}_{p^r} \times \mathbb{F}_{p^r}^n)$ such that $\mathbf{w} - \mathbf{v} = \mathbf{u} \cdot \Delta$. A random VOLE correlation VOLE^n_Δ samples such a tuple for a given $\Delta \in \mathbb{F}_{p^r}$ and $\mathbf{u} \xleftarrow{\$} \mathbb{F}_p^n, \mathbf{v} \xleftarrow{\$} \mathbb{F}_{p^r}^n$. Write VOLE^n for the correlation, where $\Delta \xleftarrow{\$} \mathbb{F}_{p^r}$.

Notation. When Δ is clear from the context, we use the following notation as a shorthand: Let $\mathbf{x} \in \mathbb{F}_p^n$ (or $\in \mathbb{F}_p$) be a vector (or scalar). First, we use $\langle \mathbf{x} \rangle$ to denote a subtractive secret sharing of $\Delta \cdot \mathbf{x}$, i.e., $\langle \mathbf{x} \rangle = (\langle \mathbf{x} \rangle_0, \langle \mathbf{x} \rangle_1)$ such that $\langle \mathbf{x} \rangle_1 - \langle \mathbf{x} \rangle_0 = \Delta \cdot \mathbf{x}$. Then, we use $[\![\mathbf{x}]\!]$ to denote the full VOLE correlation with $\mathbf{x}$, i.e., $[\![\mathbf{x}]\!] = \big((\mathbf{x}, \langle \mathbf{x} \rangle_0), (\Delta, \langle \mathbf{x} \rangle_1)\big)$. We define linear operations on VOLE-authenticated data such as $[\![\mathbf{z}]\!] = \alpha \cdot [\![\mathbf{x}]\!] + [\![\mathbf{y}]\!] + \beta$ as follows:

$$[\![\mathbf{z}]\!]_0 = \alpha \cdot [\![\mathbf{x}]\!]_0 + [\![\mathbf{y}]\!]_0 + \beta = \big(\alpha \cdot \mathbf{x} + \mathbf{y} + \beta, \alpha \cdot \langle \mathbf{x} \rangle_0 + \langle \mathbf{y} \rangle_0\big) \qquad \text{and}$$
$$[\![\mathbf{z}]\!]_1 = \alpha \cdot [\![\mathbf{x}]\!]_1 + [\![\mathbf{y}]\!]_1 + \beta = \big(\Delta, \alpha \cdot \langle \mathbf{x} \rangle_1 + \langle \mathbf{y} \rangle_1 + \Delta \cdot \beta\big).$$

Remark 1. We note that a random VOLE correlation is *reverse sampleable*. We can define the algorithm $\mathsf{VOLE}^n_\Delta.\mathsf{RSample}(1^\lambda, \sigma, y_\sigma)$ as follows: If $\sigma = 0$, parse $y_0 = [\![\mathbf{u}]\!]_0 = (\mathbf{u}, \langle \mathbf{u} \rangle_0)$. Then compute $\langle \mathbf{u} \rangle_1 := \langle \mathbf{u} \rangle_0 + \mathbf{u} \cdot \Delta$, and output $[\![\mathbf{y}]\!]_1 := (\Delta, \langle \mathbf{u} \rangle_1)$. Otherwise, if $\sigma = 1$, parse $y_1 = [\![\mathbf{u}]\!]_1 = (\Delta, \langle \mathbf{u} \rangle_1)$. Sample $\mathbf{u} \xleftarrow{\$} \mathbb{F}_p^n$, compute $\langle \mathbf{u} \rangle_0 := \langle \mathbf{u} \rangle_1 - \mathbf{u} \cdot \Delta$, and output $[\![\mathbf{u}]\!]_0 := (\mathbf{u}, \langle \mathbf{u} \rangle_0)$. $\mathsf{VOLE}^n.\mathsf{RSample}(1^\lambda, \sigma, y_\sigma)$ is defined similarly, but samples $\Delta \xleftarrow{\$} \mathbb{F}_{p^r}$ for $\sigma = 0$.

3 A PCF for VOLE

We now describe our PCF construction for subfield VOLE correlations. We denote by $[\![\mathbf{x}]\!]$ a VOLE correlation of the form $\langle \mathbf{x} \rangle_1 - \langle \mathbf{x} \rangle_0 = \Delta \cdot \mathbf{x}$, as introduced in Section 2.5. Our construction follows the outline given in Section 1.5, and is formally presented in Figure 2.

The high-level idea behind our PCF construction is to start with an LPN-PCG for VOLE correlation and obtain a PCF by stretching the length of that correlation. Starting with a short VOLE correlation $\langle \mathbf{s}^{(0)} \rangle$, we expand this correlation by computing $\langle \mathbf{s}^{(\ell)} \rangle = \mathbf{A}^{(\ell)} \cdot \langle \mathbf{s}^{(\ell-1)} \rangle + \langle \mathbf{e}^{(\ell)} \rangle$ iteratively by using a random stretching linear code $\mathbf{A}^{(\ell)}$ as many times as needed. In particular, using the sparse-LPN assumption at each iteration ℓ, we sample a sparse matrix $\mathbf{A}^{(\ell)} \in \mathbb{F}^{m^{(\ell)} \times m^{(\ell-1)}}$ along with a sparse vector $\mathbf{e}^{(\ell)} \in \mathsf{Reg}_t(m^{(\ell)}, \mathbb{F}_p)$ to obtain a super-polynomial stretch of the VOLE correlation while maintaining a logarithmic key-size in terms of the security parameter, and enabling the independent computation of each entry of the output.

Section 3.1 gives details about the key sizes and evaluation costs. We state the following theorem about our construction and prove it in Sections 3.2 and 3.3.

Theorem 1 (Our construction is a secure PCF for subfield VOLE). *Let $L, t \in \mathbb{N}$. Let Q_A, Q_B denote fixed a priori bounds on the number of queries to $\mathsf{H}_A^{(L)}$, and H_B, respectively, and let $Q_\mathcal{O}$ be a fixed a priori bound on the number of queries to either oracles $\mathcal{O}_1^{\mathsf{pr}}$ or $\mathcal{O}_1^{\mathsf{sec}}$ (see Figure 1). The protocol presented in Figure 2 is a secure pseudorandom function with VOLE-correlated outputs (see Definition 9), under the assumptions that*

1. *$\mathsf{H}_A^{(\ell)}$ and H_B are random oracles for all $\ell \in [L]$,*
2. *$\mathsf{SparseLPN}(k^{(\ell)}, m^{(\ell-1)}, m^{(\ell)}, \mathsf{Reg}_t, \mathbb{F}_p)$ is hard for $\ell \in [L-1]$,*
3. *$\mathsf{SparseLPN}(k^{(L)}, m^{(L-1)}, Q_{A^L} + Q_B + Q_\mathcal{O}, D^{\mathsf{sp}}_{\mathsf{Reg}_t,\beta}, \mathbb{F}_p)$ is hard, and*
4. *$\mathsf{DMPF}^{(\ell)}$ is a secure DMPF for t-point function with pseudorandom outputs in $\mathbb{F}_{p^r}$ for $\ell \in [L]$.*

3.1 Properties

Key sizes. The size of each PCF key k_σ^{PCF} differs for $\sigma = 0$ and $\sigma = 1$. In particular, k_0^{PCF} includes $[\![\mathbf{s}^{(0)}]\!] \in \mathbb{F}_p^{m^{(0)}} \times \mathbb{F}_{p^r}^{m^{(0)}}$, L DMPF keys, and L vectors $(\mathbf{e}^{(\ell)})_{\ell \in [L]}$ that are t-sparse and regular.[7] The total size of k_0^{PCF} can thus be computed as

$$|k_0^{\mathsf{PCF}}| = m^{(0)} \cdot (r+1) \cdot \log p + \sum_{\ell=1}^{L} |k_{0,(\ell)}^{\mathsf{DMPF}}| + L \cdot t \cdot r \cdot \log p.$$

The other PCF key, k_1^{PCF} consists of $[\![\mathbf{s}^{(0)}]\!]_1 \in \mathbb{F}_{p^r} \times \mathbb{F}_{p^r}^{m^{(0)}}$, and L DMPF keys. We therefore have

$$|k_1^{\mathsf{PCF}}| = (m^{(0)} + 1) \cdot r \cdot \log p + \sum_{\ell=1}^{L} |k_{1,(\ell)}^{\mathsf{DMPF}}|.$$

[7] Note that we must only count the *values* of the non-zero entries $\mathbf{e}^{(\ell)}$. The indices are already included in $k_{0,(\ell)}^{\mathsf{DMPF}}$.

Figure 2: PCF for Subfield VOLE from Sparse LPN

– Let $\mathbb{F}_{p^r}$ be a finite field and let $\mathcal{X}$ denote the domain of the PCF.
– Let $L, t, m^{(0)}, \ldots, m^{(L)}, k^{(1)}, \ldots, k^{(L)}, k^B \in \mathbb{N}$, such that $t < m^{(1)}$, $m^{(0)} < \cdots < m^{(L)}$, $k^{(\ell)} \leq m^{(\ell-1)}$ for $\ell \in [L]$, $k^B \leq m^{(L)}$.
– Let $\mathsf{DMPF}^{(\ell)} = (\mathsf{DMPF}^{(\ell)}.\mathsf{Gen}, \mathsf{DMPF}^{(\ell)}.\mathsf{Eval})$ be a distributed multi-point function scheme for the class of t-point functions $[m^{(\ell)}] \to \mathbb{F}_p$ for $\ell \in [L]$.
– Let $\mathsf{H}_A^{(1)}, \ldots, \mathsf{H}_A^{(L)}$ and H_B be random oracles such that

$$\mathsf{H}_A^{(\ell)} \colon [m^{(\ell)}] \to \mathcal{V}_{k^{(\ell)}}^{m^{(\ell-1)}}(\mathbb{F}_p) \text{ for } \ell \in [L-1], \qquad \mathsf{H}_A^{(L)} \colon \mathcal{X} \to \mathcal{V}_{k^{(L)}}^{m^{(L-1)}}(\mathbb{F}_p), \qquad \mathsf{H}_B \colon \mathcal{X} \to \mathcal{V}_{k^B}^{m^{(L)}}(\mathbb{F}_p).$$

$\mathsf{PCF}.\mathsf{Gen}(1^\lambda)$**:**

1. Sample $\Delta \xleftarrow{\$} \mathbb{F}_{p^r}$, $[\![\mathbf{s}^{(0)}]\!] \leftarrow \mathsf{VOLE}_\Delta^{m^{(0)}}$.
2. For $\ell \in [L]$ sample $\mathbf{e}^{(\ell)} = \left((\alpha_i^{(\ell)}, \beta_i^{(\ell)})\right)_{i \in [t]} \leftarrow \mathsf{Reg}_t(m^{(\ell)}, \mathbb{F}_p)$, and run
$$(k_{0,(\ell)}^{\mathsf{DMPF}}, k_{1,(\ell)}^{\mathsf{DMPF}}) \leftarrow \mathsf{DMPF}^{(\ell)}.\mathsf{Gen}\left(1^\lambda, \left((\alpha_i^{(\ell)}, \Delta \cdot \beta_i^{(\ell)})\right)_{i \in [t]}\right).$$
3. Output $k_0^{\mathsf{PCF}} = \left([\![\mathbf{s}^{(0)}]\!]_0, \left(\mathbf{e}^{(\ell)}, k_{0,(\ell)}^{\mathsf{DMPF}}\right)_{\ell \in [L]}\right)$, $k_1^{\mathsf{PCF}} = \left([\![\mathbf{s}^{(0)}]\!]_1, \left(k_{1,(\ell)}^{\mathsf{DMPF}}\right)_{\ell \in [L]}\right)$.

$\mathsf{PCF}.\mathsf{Eval}(\sigma, k_\sigma^{\mathsf{PCF}}, x \in \mathcal{X})$**:**

1. Compute $\mathbf{A}_x^{(L)} \leftarrow \mathsf{H}_A^{(L)}(x)$ and $\mathbf{B}_x \leftarrow \mathsf{H}_B(x)$.
2. For $(j, \cdot) \in \mathbf{A}_x^{(L)}$ compute $[\![s_j^{(L-1)}]\!]_\sigma \leftarrow \mathsf{EvalRec}(\sigma, k_\sigma^{\mathsf{PCF}}, L-1, j)$.
3. For $(j, \cdot) \in \mathbf{B}_x$ compute $\langle e_j^{(L)} \rangle_\sigma \leftarrow \mathsf{DMPF}.\mathsf{Eval}(\sigma, k_{\sigma,(L)}^{\mathsf{DMPF}}, j)$.[a]
4. Compute and return
$$[\![s_x^{(L)}]\!]_\sigma = \sum_{(j,a_j) \in \mathbf{A}_x^{(L)}} a_j \cdot [\![s_j^{(L-1)}]\!]_\sigma + \sum_{(j,b_j) \in \mathbf{B}_x} b_j \cdot [\![e_j^{(L)}]\!]_\sigma \ .$$

$\mathsf{EvalRec}(\sigma, k_\sigma^{\mathsf{PCF}}, \ell \in [0, L-1], i \in [m^{(\ell)}])$**:**

1. If $\ell = 0$, return $[\![s_i^{(0)}]\!]_\sigma$.
2. Otherwise compute $\mathbf{A}_i^{(\ell)} \leftarrow \mathsf{H}_A^{(\ell)}(i)$.
3. For $(j, \cdot) \in \mathbf{A}_i^{(\ell)}$ compute $[\![s_j^{(\ell-1)}]\!]_\sigma \leftarrow \mathsf{EvalRec}(\sigma, k_\sigma^{\mathsf{PCF}}, \ell-1, j)$.
4. Compute $\langle e_i^{(\ell)} \rangle_\sigma \leftarrow \mathsf{DMPF}.\mathsf{Eval}(\sigma, k_{\sigma,(\ell)}^{\mathsf{DMPF}}, i)$.
5. Compute and return
$$[\![s_i^{(\ell)}]\!]_\sigma = \sum_{(j,a_j) \in \mathbf{A}_i^{(\ell)}} a_j \cdot [\![s_j^{(\ell-1)}]\!]_\sigma + [\![e_i^{(\ell)}]\!]_\sigma.$$

[a] Recall that $[\![e_j^{(\ell)}]\!]_0 = (e_j^{(\ell)}, \langle e_j^{(\ell)} \rangle_0)$ and $[\![e_j^{(\ell)}]\!]_1 = (\Delta, \langle e_j^{(\ell)} \rangle_1)$ for $\ell \in [L]$.

With the standard known-index DMPF construction from GGM PPRFs (where Party 0 knows the indices and holds the corresponding punctured keys), we have $|k^{\mathsf{DMPF}}_{0,(\ell)}| = t \cdot \big((\lambda+1)\cdot\log(m^{(\ell)}/t) + r\cdot\log p\big)$, and $|k^{\mathsf{DMPF}}_{1,(\ell)}| = \lambda \cdot t$ for $\ell \in [L]$.

DMPF Evaluations. The main computational cost of our PCF stems from the DMPF evaluations. Note that we can rewrite the output $\langle \mathbf{u} \rangle$ of the PCF as follows:

$$\begin{aligned}\langle \mathbf{u} \rangle &= \mathbf{A}^{(L)} \cdot \left(\cdots \left(\mathbf{A}^{(2)} \cdot \left(\mathbf{A}^{(1)} \cdot \langle \mathbf{s}^{(0)} \rangle + \langle \mathbf{e}^{(1)} \rangle \right) + \langle \mathbf{e}^{(2)} \rangle \right) \cdots \right) + \mathbf{e}^{(L)} \\ &= \left(\prod_{\ell=0}^{L-1} \mathbf{A}^{(L-\ell)} \right) \cdot \langle \mathbf{s}^{(0)} \rangle + \sum_{\ell=1}^{L-1} \left(\prod_{j=0}^{L-\ell-1} \mathbf{A}^{(L-j)} \right) \cdot \langle \mathbf{e}^{(\ell)} \rangle + \mathbf{B} \cdot \langle \mathbf{e}^{(L)} \rangle.\end{aligned}$$

Since the matrices $\mathbf{A}^{(\ell)}$ are $k^{(\ell)}$-sparse and $\mathbf{B}$ is k^B-sparse, the computation of a single entry of $\langle \mathbf{u} \rangle$ depends on $\prod_{j=\ell+1}^{L} k^{(j)}$ entries of $\langle \mathbf{e}^{(\ell)} \rangle$ and k^B entries of $\mathbf{e}^{(L)}$. The entries of $\langle \mathbf{e}^{(\ell)} \rangle$ are computed by evaluating the corresponding DMPF with domain $m^{(\ell)}$. Thus we need $\sum_{\ell=1}^{L-1} \prod_{j=\ell+1}^{L} k^{(j)} + k^B$ DMPF evaluations in total.

Asymptotic Parameters and Security. For each $\ell \in [L]$, the vector $\mathbf{s}^{(\ell)} \in \mathbb{F}^{m^{(\ell)}}$ is a sparse-LPN instance computed as $\mathbf{s}^{(\ell)} = \mathbf{A}^{(\ell)} \cdot \mathbf{s}^{(\ell-1)} + \mathbf{e}^{(\ell)}$, where $\mathbf{e}^{(\ell)}$ has a Hamming weight of t and $\mathbf{A}^{(\ell)} \in \mathbb{F}^{m^{(\ell)} \times m^{(\ell-1)}}$ is k-sparse. We take $n = m^{(0)}$ as security parameter and outline a candidate parameter set (several tradeoffs are possible). The parameter set must satisfy the following constraints:

- For efficiency, we require $k^L = \mathsf{poly}(n)$ (evaluation must run in polynomial time)
- For security, we observe that as long as $t \cdot m^{(\ell-1)} \geq n^{\varepsilon} \cdot m^{(\ell)}$ for some constant $0 < \varepsilon < 1$, the best known attacks against sparse-LPN with sparsity $k = \omega(1)$ must run in time $2^{\tilde{O}(n^{\varepsilon})}$ to achieve non-negligible advantage [Lyu05; BJMM12; MO15; BM18].

For concreteness, we fix $t = n$, $\varepsilon = 1/2$, and $k = \log(n) = \omega(1)$. To enforce the security condition, we set $m^{(0)} = n$ and $m^{(\ell)} = m^{(\ell-1)} \cdot \sqrt{n}$; note that $t \cdot m^{(\ell-1)} = n \cdot m^{(\ell-1)} \geq \sqrt{n} \cdot m^{(\ell)} = \sqrt{n} \cdot \sqrt{n} \cdot m^{(\ell-1)}$. In this regime and for all known attacks, attackers must run in time $2^{\tilde{O}(\sqrt{n})}$ or else have advantage bounded by $n^{-O(\log n)}$. For efficiency, we set $L = \log(n)/\log(k) = \log n/\log\log n$, so that $k^L = n$.

With these parameters, starting from PCF keys of size $\mathsf{poly}(n)$, the PCF is secure for up to $m^{(L)} = n^{1+L/2} = n^{O(\log n/\log\log n)}$ queries. That is, these parameters achieve security with inverse-superpolynomial advantage against subexponential-time adversaries making at most superpolynomially-many queries to the PCF.

3.2 Pseudorandom VOLE-Correlated Outputs

As per Definition 9, proving security of the construction in Figure 2 requires proving that the outputs are pseudorandom VOLE-correlated, i.e., proving that, for any input x, the distribution of the outputs of PCF.Eval are computationally indistinguishable from random VOLE_{Δ}-correlated pairs for some $\Delta \in \mathbb{F}_{p^r}$:

$$\left\{ \left(\mathsf{PCF.Eval}(1^\lambda, 0, k_0^{\mathsf{PCF}}, x), \mathsf{PCF.Eval}(1^\lambda, 1, k_1^{\mathsf{PCF}}, x) \right) \right\}_x \sim_c \left\{ \mathsf{VOLE}^1_{\Delta} \right\}_x$$

Thus, it suffices to prove that the outputs of PCF.Eval are, a) VOLE^1_{Δ}-correlated, and b) are indistinguishable from uniformly random VOLE tuples.

Correctness. We prove that for all security parameters $\lambda \in \mathbb{N}$, all key pairs $\left(k_0^{\mathsf{PCF}}, k_1^{\mathsf{PCF}}\right) \leftarrow \mathsf{PCF.Gen}(1^\lambda)$ and all inputs $x \in \mathcal{X}$, the outputs of PCF.Eval are VOLE^1_{Δ}-correlated:

$$\left([\![s_x^{(L)}]\!]_0 = \left(s_x^{(L)}, \left\langle s_x^{(L)} \right\rangle_0 \right), [\![s_x^{(L)}]\!]_1 = \left(\Delta, \left\langle s_x^{(L)} \right\rangle_1 \right) \right) = \begin{pmatrix} \mathsf{PCF.Eval}(0, k_0^{\mathsf{PCF}}, x), \\ \mathsf{PCF.Eval}(1, k_1^{\mathsf{PCF}}, x) \end{pmatrix}.$$

Let the parameters be as in Figure 2. We show by induction over $\ell \in [0, L]$ that the elements of $[\![\mathbf{s}^{(\ell)}]\!]$, which are (implicitly) computed by PCF.Eval, are VOLE_Δ-correlated where Δ is contained in k_1^{PCF}.

Base Case ($\ell = 0$): This case is immediate, as Step 1 of PCF.Gen explicitly samples $[\![\mathbf{s}^{(0)}]\!]$ from the correlation $\mathsf{VOLE}_\Delta^{m^{(0)}}$.

Inductive step ($\ell - 1 \Rightarrow \ell$): Let $\ell \in [L-1]$ and suppose all entries $[\![s_i^{(\ell-1)}]\!]$ are VOLE_Δ^1-correlated, for $i \in [m^{(\ell-1)}]$. Note that $[\![e_i^{(\ell)}]\!]$ is VOLE_Δ^1-correlated due to the correctness of $\mathsf{DMPF}^{(\ell)}$. Then EvalRec computes

$$[\![s_i^{(\ell)}]\!]_\sigma = \sum_{(j,a_j)\in \mathbf{A}_x^{(\ell)}} a_j \cdot [\![s_j^{(\ell-1)}]\!]_\sigma + [\![e_j^{(\ell)}]\!]_\sigma$$

and by linearity of the VOLE correlation, $[\![s_i^{(\ell)}]\!]$ is also VOLE_Δ^1 correlated.

Final step ($L - 1 \Rightarrow L$): The last step follows by the same reasoning; the only difference is that we include multiple terms $[\![e_j^{(L)}]\!]$ in the computation. Thus the final output $[\![s_x^{(L)}]\!]$ is also VOLE_Δ^1-correlated.

Pseudorandomness. The proof of pseudorandomness of the outputs of our PCF consists of proving computational indistinguishability of the real and ideal experiments $\mathsf{Exp}^{\mathsf{pr}}_{\mathcal{A},1}$ and $\mathsf{Exp}^{\mathsf{pr}}_{\mathcal{A},0}$ as defined in Figure 1. We do so by a defining the following sequence of hybrids:

Hyb_0 is identical to the real-world experiment, $\mathsf{Exp}_1^{\mathsf{pr}}$ in Definition 9.

Hyb_1 is identical to Hyb_0, except:

$\mathcal{O}_1^{\mathsf{pr}}(x)$**:** in this hybrid, the oracle

1. Reads Δ from k_1^{PCF}.
2. Computes $y_0 \leftarrow \mathsf{PCF.Eval}(0, k_0^{\mathsf{PCF}}, x)$.
3. Computes $y_1 \leftarrow \mathsf{VOLE}_\Delta^1.\mathsf{RSample}(y_0)$.
4. Returns (y_0, y_1).

The PCF outputs are VOLE-correlated pairs (proven immediately above), and VOLE is reverse-sampleable (Definition 8). Therefore, Hyb_1 is perfectly indistinguishable from Hyb_0.
Note that, as the game no longer makes use of k_1^{PCF}, we only need to adjust the PCF algorithms (executed by the game) when called with argument $\sigma = 0$ in the following hybrids.

$\mathsf{Hyb}_2^0 := \mathsf{Hyb}_1$ is identical to the previous hybrid.

Hyb_2^l for $l \in [0, L-1]$: algorithms PCF.Gen and PCF.Eval are modified in each successive hybrid to replace the VOLE correlations computed by the inner l iterations by ideal, uniformly random VOLE samples:

$\mathsf{PCF.Gen}(1^\lambda)$**:** Lazily sample $[\![\mathbf{s}^{(l)}]\!] \leftarrow \mathsf{VOLE}_\Delta^{m^{(l)}}$, and include these shares in the PCF keys.[8]

$\mathsf{PCF.EvalRec}(\sigma = 0, k_0^{\mathsf{PCF}}, \ell, i)$**:** If $\ell = l$, return $[\![s_i^{(l)}]\!]_0$. Otherwise, recursively compute the output. Note that EvalRec with $\ell < l$ is no longer called.

Hyb_2^L**:** is identical to Hyb_2^{L-1}, except:

$\mathsf{PCF.Eval}(\sigma = 0, k_0^{\mathsf{PCF}}, x)$**:** lazily samples $[\![s_x^{(L)}]\!] \leftarrow \mathsf{VOLE}_\Delta^1$ and returns $[\![s_x^{(L)}]\!]_0$ for each query x.

$\mathsf{Hyb}_3 := \mathsf{Hyb}_2^L$**:** is identical to the ideal-world experiment $\mathsf{Exp}_0^{\mathsf{pr}}$ in Definition 9.

It remains to show that Hybrid Hyb_2^l is indistinguishable from Hyb_2^{l-1} for all $l \in [L]$. We prove this for $l \in [L-1]$ in Lemma 2 and for $l = L$ in Lemma 3 by applying the pseudorandomness of the DMPF and the corresponding LPN assumption for each level.

Lemma 2. *Let $l \in [L-1]$. Hybrids Hyb_2^{l-1} and Hyb_2^l are computationally indistinguishable under the assumptions that:*

1. *$\mathsf{H}_A^{(l)}$ is a random oracle,*
2. *$\mathsf{SparseLPN}(k^{(l)}, m^{(l-1)}, m^{(l)}, \mathsf{Reg}_t, \mathbb{F}_p)$ is hard, and*
3. *$\mathsf{DMPF}^{(l)}$ is a DMPF with pseudorandom outputs.*

[8] To ensure the game stays efficient, the algorithm instead lazily samples the entries of $[\![\mathbf{s}^{(l)}]\!]$ when they are first accessed.

Proof. Recall that the difference between these hybrids consist in replacing

$$[\![s_i^{(l)}]\!] = \sum_{(j,a_j)\in \mathbf{A}_i^{(l)}} a_j \cdot [\![s_j^{(l-1)}]\!] + [\![e_i^{(l)}]\!] \qquad \text{with} \qquad [\![s_i^{(l)}]\!] \leftarrow \mathsf{VOLE}_\Delta^1.$$

Since we only need to take care of the changes for $\sigma = 0$, this is equivalent to replacing

$$s_i^{(l)} = \sum_{(j,a_j)\in \mathbf{A}_i^{(l)}} a_j \cdot s_j^{(l-1)} + e_i^{(l)} \qquad \text{with} \qquad s_i^{(l)} \xleftarrow{\$} \mathbb{F}_p$$

$$\langle s_i^{(l)}\rangle_0 = \sum_{(j,a_j)\in \mathbf{A}_i^{(l)}} a_j \cdot \langle s_j^{(l-1)}\rangle_0 + \langle e_i^{(l)}\rangle_0 \qquad \text{with} \qquad \langle s_i^{(l)}\rangle_0 \xleftarrow{\$} \mathbb{F}_{p^r}$$

At a high level, we step-by-step make the following changes to prove indistinguishability:

1. use LPN to replace $s_i^{(l)}$ by a uniformly random $s_i^{(l)} \xleftarrow{\$} \mathbb{F}_p$,
2. use pseudorandomness of the DMPF to replace $\langle e_i^{(l)}\rangle_0$ by a uniformly random $\langle e_i^{(l)}\rangle_0 \xleftarrow{\$} \mathbb{F}_{p^r}$,
3. and finally replace $\langle s_i^{(l)}\rangle$ by a uniformly random $\langle s_i^{(l)}\rangle_0 \xleftarrow{\$} \mathbb{F}_{p^r}$.

In more detail:

$\mathsf{Hyb}_2^{l-1,0}$ is identical to Hyb_2^{l-1}.

$\mathsf{Hyb}_2^{l-1,1}$**:** In this hybrid, PCF.EvalRec, input i, computes $[\![s_i^{(l)}]\!]_0$ similarly to the previous hybrid, except for Step 4a below:

PCF.EvalRec$(\sigma = 0, k_0^{\mathsf{PCF}}, \ell, i)$**:** If $\ell \neq l$, behave as before. Otherwise if $\ell = l$:

1. Compute $\mathbf{A}_i^{(l)} \leftarrow \mathsf{H}_A^{(l)}(i)$.
2. For $(j, \cdot) \in \mathbf{A}_i^{(l)}$ compute $[\![s_j^{(l-1)}]\!]_0 \leftarrow \mathsf{EvalRec}(0, k_0^{\mathsf{PCF}}, l-1, j)$.
3. Compute $\langle e_i^{(l)}\rangle_0 \leftarrow \mathsf{DMPF.Eval}(0, k_{0,(l)}^{\mathsf{DMPF}}, i)$.
4. (a) Sample $s_i^{(l)} \xleftarrow{\$} \mathbb{F}_p$.
   (b) Compute
   $$\langle s_i^{(l)}\rangle_0 = \sum_{(j,a_j)\in \mathbf{A}_i^{(L)}} a_j \cdot \langle s_j^{(l-1)}\rangle_0 + \langle e_i^{(l)}\rangle_0.$$
   (c) Set and return $[\![s_i^{(l)}]\!]_0 = (s_i^{(l)}, \langle s_i^{(l)}\rangle_0)$.

We defer showing the indistinguishability of $\mathsf{Hyb}_2^{l-1,1}$ and $\mathsf{Hyb}_2^{l-1,0}$ to Claim 1 below.

$\mathsf{Hyb}_2^{l-1,2}$**:** In this hybrid we make the following change to the EvalRec algorithm:

PCF.EvalRec$(\sigma = 0, k_0^{\mathsf{PCF}}, \ell, i)$**:** If $\ell = l$:

3. Sample $\langle e_i^{(l)}\rangle_0 \xleftarrow{\$} \mathbb{F}_{p^r}$.

As the underlying $\mathsf{DMPF}^{(l)}$ has pseudorandom outputs, each share $\langle e_x^{(l)}\rangle_0$ is computationally indistinguishable from a randomly chosen field element. Therefore, $\mathsf{Hyb}_2^{l-1,2}$ is computationally indistinguishable from $\mathsf{Hyb}_2^{l-1,1}$.

$\mathsf{Hyb}_2^{l-1,3}$**:** This hybrid applies the final modification to PCF.EvalRec.

PCF.EvalRec$(\sigma = 0, k_0^{\mathsf{PCF}}, \ell, i)$**:** If $\ell = l$:

4. (b) Sample $\langle s_i^{(l)}\rangle_0 \xleftarrow{\$} \mathbb{F}_{p^r}$.

In other words, EvalRec now samples and returns $[\![s_i^{(l)}]\!]_0 \xleftarrow{\$} \mathbb{F}_p \times \mathbb{F}_{p^r}$. Since $\langle e_x^{(l)}\rangle_0$ was already replaced by a uniformly random value in $\mathsf{Hyb}_2^{l-1,2}$ (and not used or output otherwise), $\langle s_x^{(l)}\rangle_0$ is already distributed uniformly random. Thus, we can just directly sample $\langle s_x^{(l)}\rangle_0 \xleftarrow{\$} \mathbb{F}_{p^r}$.

Note that $\mathsf{Hyb}_2^{l-1,3} = \mathsf{Hyb}_2^{l}$. Therefore, combining all of the indistinguishability arguments above, we have $\mathsf{Hyb}_2^{l-1} \approx_c \mathsf{Hyb}_2^{l}$, for all $l \in [L-1]$. Now we prove the deferred indistinguishability between $\mathsf{Hyb}_2^{l-1,1}$ and $\mathsf{Hyb}_2^{l-1,0}$:

Claim 1. Assuming the hardness of $\mathsf{SparseLPN}(k^{(l)}, m^{(l-1)}, m^{(l)}, \mathsf{Reg}_t, \mathbb{F}_p)$, hybrids $\mathsf{Hyb}_2^{l-1,1}$ and $\mathsf{Hyb}_2^{l-1,0}$ are computationally indistinguishable.

Proof. Let $\mathcal{A}$ be a PPT adversary that can distinguish between these two hybrids. We construct an adversary $\mathcal{B}$ against the SparseLPN instance, as follows: The task of $\mathcal{B}$ is, given a pair $(\mathbf{A}, \mathbf{y})$ such that $\mathbf{A} \xleftarrow{\$} \mathcal{M}_{k^{(l)}}^{m^{(l)} \times m^{(l-1)}}(\mathbb{F}_p)$ and $\mathbf{y} \in \mathbb{F}_p^{m^{(l)}}$, to distinguish whether $\mathbf{y} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$, for some $\mathbf{s} \xleftarrow{\$} \mathbb{F}_p^{m^{(l-1)}}$ and $\mathbf{e} \xleftarrow{\$} \mathsf{Reg}_t(m^{(l)}, \mathbb{F}_p)$, or $\mathbf{y} \xleftarrow{\$} \mathbb{F}_p^{m^{(l)}}$. Since $m^{(l)}$ might be superpolynomially large, we assume that $\mathcal{B}$ can ask iteratively query samples of the form $(\mathbf{A}_i, y_i)$ for $i \in [m^{(l)}]$, where each row $\mathbf{A}_i \in \mathcal{V}_{k^{(l)}}^{m^{(l-1)}}(\mathbb{F}_p)$ has a polynomially sized representation.

The adversary $\mathcal{B}$ uses these samples to program the oracles $\mathsf{H}_A^{(l)}$ and $\mathcal{O}_1^{\mathsf{pr}}$ accessed by $\mathcal{A}$:[9]

$\mathsf{H}_A^{(l)}(i)$: Given $i \in [m^{(l)}]$, $\mathcal{B}$ receives the sample $(\mathbf{A}_i, y_i)$ from the SparseLPN challenger and returns $\mathbf{A}_i$.

$\mathcal{O}_1^{\mathsf{pr}}(\cdot)$: We change the computation of $\mathsf{PCF}.\mathsf{EvalRec}(\sigma = 0, k_0^{\mathsf{PCF}}, \ell, i)$ where $i \in [m^{(\ell)}]$ as follows:

$\mathsf{PCF}.\mathsf{EvalRec}\left(\sigma = 0, k_0^{\mathsf{PCF}}, \ell, i\right)$: If $\ell = l$:

4. (a) $\mathcal{B}$ receives the sample $(\mathbf{A}_i, y_i)$ from the SparseLPN challenger and sets $s_i^{(l)} := y_i$.

Note that if the challenges received by $\mathcal{B}$ are SparseLPN structured, then $\mathcal{B}$ simulates Hybrid $\mathsf{Hyb}_2^{l-1,0}$ for $\mathcal{A}$. Otherwise, if they are random, $\mathcal{B}$ simulates Hybrid $\mathsf{Hyb}_2^{l-1,1}$. Hence, if $\mathcal{A}$ can distinguish between these two hybrids, $\mathcal{B}$ can solve the sparse-LPN challenges for the aforementioned parameters. Therefore the hybrids are computationally indistinguishable. ■

This concludes the proof of Lemma 2. □

Lemma 2 covers the indistinguishability of the first $L-1$ pairs of hybrids. We now prove that also the last two hybrids in this sequence are indistinguishable:

Lemma 3. *Let $Q_A, Q_B, Q_{\mathcal{O}}$ denote a-priori fixed bounds on the number of queries issued by $\mathcal{A}$ to $\mathsf{H}_A^{(L)}$, H_B and $\mathcal{O}_1^{\mathsf{pr}}$, respectively (see Figure 1). Hybrids Hyb_2^{L-1} and Hyb_2^{L} are computationally indistinguishable under the assumptions that:*

1. *$\mathsf{H}_A^{(L)}$ and H_B are random oracles,*
2. *$\mathsf{SparseLPN}(k^{(L)}, m^{(L-1)}, Q_A + Q_B + Q_{\mathcal{O}}, D_{\mathsf{Reg}_t,\beta}^{\mathsf{sp}}, \mathbb{F}_p)$ is hard, and*
3. *$\mathsf{DMPF}^{(L)}$ is a DMPF with pseudorandom outputs.*

Proof. The proof follows along the lines of the proof of Lemma 2, but needs to account for the different flavor of LPN that is used on the last level. We prove indistinguishability via the following sequence of hybrids.

$\mathsf{Hyb}_2^{L-1,0}$ is identical to Hyb_2^{L-1}.

$\mathsf{Hyb}_2^{L-1,1}$: In this hybrid, $\mathsf{PCF}.\mathsf{Eval}$, on input x, computes $[\![s_x^{(L)}]\!]_0$ similarly to the previous hybrid, except for Step 4a below:

$\mathsf{PCF}.\mathsf{Eval}(\sigma = 0, k_0^{\mathsf{PCF}}, x)$:

1. Compute $\mathbf{A}_x^{(L)} \leftarrow \mathsf{H}_A^{(L)}(x)$ and $\mathbf{B}_x \leftarrow \mathsf{H}_B(x)$.
2. For $(j, \cdot) \in \mathbf{A}_x^{(L)}$ compute $[\![s_j^{(L-1)}]\!]_0 \leftarrow \mathsf{EvalRec}(0, k_0^{\mathsf{PCF}}, L-1, j)$.
3. For $(j, \cdot) \in \mathbf{B}_x$ compute $\langle e_j^{(L)} \rangle_0 \leftarrow \mathsf{DMPF}.\mathsf{Eval}(0, k_{0,(L)}^{\mathsf{DMPF}}, j)$.
4. (a) Sample $s_x^{(L)} \xleftarrow{\$} \mathbb{F}_p$.
   (b) Compute
   $$\langle s_x^{(L)} \rangle_0 = \sum_{(j,a_j) \in \mathbf{A}_x^{(L)}} a_j \cdot \langle s_j^{(L-1)} \rangle_0 + \sum_{(j,b_j) \in \mathbf{B}_x} b_j \cdot \langle e_j^{(L)} \rangle_0.$$
   (c) Set and return $[\![s_x^{(L)}]\!]_0 = (s_x^{(L)}, \langle s_x^{(L)} \rangle_0)$.

[9] The output of the oracles is consistent because $\mathcal{B}$ receives the same sample $(\mathbf{A}_i, y_i)$ for each index $i \in [m^{(l)}$ from the challenger.

We defer showing the indistinguishability of $\mathsf{Hyb}_2^{L-1,1}$ and $\mathsf{Hyb}_2^{L-1,0}$ to Claim 2 below.

$\mathsf{Hyb}_2^{L-1,2}$**:** In this hybrid we make the following change to the Eval algorithm:

$\mathsf{PCF.Eval}\left(\sigma=0,k_0^{\mathsf{PCF}},x\right)$**:** 3. For $(j,\cdot)\in\mathbf{B}_x$ sample $\langle e_j^{(L)}\rangle_0 \xleftarrow{\$} \mathbb{F}_{p^r}$.

Due to the pseudorandomness of the outputs of the underlying $\mathsf{DMPF}^{(L)}$, each share $\langle e_j^{(L)}\rangle_0$ is computationally indistinguishable from a randomly chosen field element. Therefore, $\mathsf{Hyb}_2^{L-1,2}$ is computationally indistinguishable from $\mathsf{Hyb}_2^{L-1,1}$.

$\mathsf{Hyb}_2^{L-1,3}$**:** This hybrid applies the final modification to PCF.Eval.

$\mathsf{PCF.Eval}\left(\sigma=0,k_0^{\mathsf{PCF}},x\right)$**:** 4. (b) Sample $\langle s_x^{(L)}\rangle_0 \xleftarrow{\$} \mathbb{F}_{p^r}$.

Since the $\langle e_j^{(L)}\rangle_0$ were already replaced by a uniformly random values in $\mathsf{Hyb}_2^{L-1,2}$ and are multiplied by non-zero coefficients b_j during the computation of $\langle s_x^{(L)}\rangle_0$, the latter is already distributed uniformly random. Thus, we can just directly sample $\langle s_x^{(L)}\rangle_0 \xleftarrow{\$} \mathbb{F}_{p^r}$.

Note that $\mathsf{Hyb}_2^{L-1,3}$ is the same as Hyb_2^{L}. Therefore, combining all of the indistinguishability arguments made above, we have $\mathsf{Hyb}_2^{L-1} \approx_c \mathsf{Hyb}_2^{L}$. Now we prove the deferred indistinguishability between $\mathsf{Hyb}_2^{L-1,1}$ and $\mathsf{Hyb}_2^{L-1,0}$:

Claim 2. Let $Q_A, Q_B, Q_{\mathcal{O}}$ denote a-priori fixed bounds on the number of queries issued by $\mathcal{A}$ to $\mathsf{H}_A^{(L)}$, H_B and $\mathcal{O}_1^{\mathsf{pr}}$, respectively (see Figure 1). Assuming the hardness of $\mathsf{SparseLPN}(k^{(L)}, m^{(L-1)}, Q_{A^+}Q_B + Q_{\mathcal{O}}, D^{\mathsf{sp}}_{\mathsf{Reg}_t,\beta}, \mathbb{F}_p)$, hybrids $\mathsf{Hyb}_2^{L-1,1}$ and $\mathsf{Hyb}_2^{L-1,0}$ are computationally indistinguishable.

Proof. The proof is very similar to the proof of Claim 1. Let $\mathcal{A}$ be a PPT adversary that can distinguish between these two hybrids. Let $q := Q_{A^+}Q_B + Q_{\mathcal{O}}$ and note that we can assume q to be polynomial in λ as $\mathcal{A}$ is polynomially bounded. We construct an adversary $\mathcal{B}$ as follows: The task of $\mathcal{B}$ is, given a tuple $(\mathbf{A},\mathbf{y},\mathbf{B})$ such that $\mathbf{A} \xleftarrow{\$} \mathcal{M}_{k^{(L)}}^{q\times m^{(L-1)}}(\mathbb{F}_p)$, $\mathbf{y}\in\mathbb{F}_p^q$, and $\mathbf{B} \xleftarrow{\$} \mathcal{M}_{k^B}^{q\times m^{(L-1)}}(\mathbb{F}_p)$, decide whether whether $\mathbf{y} = \mathbf{A}\cdot\mathbf{s}+\mathbf{B}\cdot\mathbf{e}$, for some $\mathbf{s}\xleftarrow{\$}\mathbb{F}_p^{m^{(L-1)}}$ and $\mathbf{e}\xleftarrow{\$}\mathsf{Reg}_t(m^{(L)},\mathbb{F}_p)$, or $\mathbf{y}\xleftarrow{\$}\mathbb{F}_p^q$. Note that due to the sparsity of $\mathbf{A}$ and $\mathbf{B}$, the challenge tuple has a polynomially sized representation.

The adversary $\mathcal{B}$ uses these samples to program the oracles $\mathsf{H}_A^{(L)}$, H_B, and $\mathcal{O}^{\mathsf{pr}}$ accessed by $\mathcal{A}$:
Let T an initially empty look-up table and $\mathsf{cnt} := 0$.

$\mathsf{H}_A^{(L)}(x)$**:** Given $x\in\mathcal{X}$, if $(x,i)\in\mathsf{T}$ for some $i\in[q]$ return $\mathbf{A}_i$. Otherwise, increment $\mathsf{cnt} := \mathsf{cnt}+1$, add (x,cnt) to T, and respond with $\mathbf{A}_{\mathsf{cnt}}$.

$\mathsf{H}_B(x)$**:** Given $x\in\mathcal{X}$, if $(x,i)\in\mathsf{T}$ for some $i\in[q]$ return $\mathbf{B}_i$. Otherwise, increment $\mathsf{cnt} := \mathsf{cnt}+1$, add (x,cnt) to T, and respond with $\mathbf{B}_{\mathsf{cnt}}$.

$\mathcal{O}_1^{\mathsf{pr}}(x)$**:** We change the computation of $\mathsf{PCF.Eval}(\sigma=0,k_0^{\mathsf{PCF}},x)$ where $x\in\mathcal{X}$ as follows:

$\mathsf{PCF.Eval}\left(\sigma=0,k_0^{\mathsf{PCF}},x\right)$**:**

4. (a) Find $(x,i)\in\mathsf{T}$ for some $i\in[q]$ and set $s_x^{(L)} := y^*$.[10]

Note that if the challenges received by $\mathcal{B}$ are SparseLPN structured, then $\mathcal{B}$ simulates Hybrid $\mathsf{Hyb}_2^{L-1,0}$ for $\mathcal{A}$. Otherwise, if they are random, $\mathcal{B}$ simulates Hybrid $\mathsf{Hyb}_2^{L-1,1}$. Hence, if $\mathcal{A}$ can distinguish between these two hybrids, $\mathcal{B}$ can solve the sparse-LPN challenges for the aforementioned parameters. Therefore the hybrids are computationally indistinguishable. ■

This concludes the proof of Lemma 3. □

By combining Lemmas 2 and 3 we have proven the first part of Theorem 1, the pseudorandomness property of our PCF.

3.3 Security

The proof of the security property of our PCF consists of proving computational indistinguishability of the real and ideal experiments $\mathsf{Exp}^{\mathsf{sec}}_{\mathcal{A},\sigma,1}$ and $\mathsf{Exp}^{\mathsf{sec}}_{\mathcal{A},\sigma,0}$ as defined in Figure 1 for both $\sigma=0$ and $\sigma=1$.

[10] Note that there will be some entry $(x,i)\in\mathsf{T}$ already, since PCF.Eval had already queried $\mathsf{H}_A^{(L)}$ and H_B on x in the previous steps.

Security for $\sigma = 0$. Consider the following sequence of hybrids:

Hyb_0 is identical to the real-world experiment $\mathsf{Exp}^{\mathsf{sec}}_{\mathcal{A},\sigma=0,b=1}(\lambda)$.

Hyb_1**:** In this hybrid, we change the oracle $\mathcal{O}_1^{\mathsf{sec}}(x)$ as follows.

1. Parse $k_1^{\mathsf{PCF}} = \left(\llbracket \mathbf{s}^{(0)} \rrbracket_1, \left(k^{\mathsf{DMPF}}_{1,(\ell)}\right)_{\ell\in[L]}\right)$, where $\llbracket \mathbf{s}^{(0)} \rrbracket_1 = (\Delta, \langle \mathbf{s}^{(0)} \rangle_1)$.
2. For each query x:
   (a) Compute $y_0 \leftarrow \mathsf{PCF.Eval}(\sigma = 0, k_0^{\mathsf{PCF}}, x)$, and parse $y_0 = \llbracket s_x^{(L)} \rrbracket_0 = (s_x^{(L)}, \langle s_x^{(L)} \rangle_0)$.
   (b) Set $y_1 := \llbracket s_x^{(L)} \rrbracket_1 = (\Delta, \langle s_x^{(L)} \rangle_0 + s_x^{(L)} \cdot \Delta) = \mathsf{VOLE}^1_\Delta.\mathsf{RSample}(\sigma = 0, y_0)$.
   (c) Return y_1.

Recall that our PCF correctly outputs VOLE-correlated pairs (proved in Section 3.2), and that the VOLE_Δ correlation is reverse-sampleable (see Remark 1). Moreover, the output of $\mathsf{VOLE}_\Delta.\mathsf{RSample}(\cdot)$ for a fixed Δ is actually deterministic. Therefore, this hybrid behaves identically to Hyb_0 and is perfectly indistinguishable. Since Hyb_1 is the same as $\mathcal{O}_0^{\mathsf{sec}}(x)$, which concludes the proof for $\sigma = 0$.

Security for $\sigma = 1$. The proof of security against this party is more involved. Consider the following sequence of hybrids.

Hyb_0 is identical to the real-world experiment $\mathsf{Exp}^{\mathsf{sec}}_{\mathcal{A},\sigma=1,b=1}(\lambda)$.

Hyb_1**:** In this hybrid, we change the oracle $\mathcal{O}_1^{\mathsf{sec}}(x)$ as follows.

1. Compute $y_1 = \left(\Delta, \langle s_x^{(L)} \rangle_1\right) \leftarrow \mathsf{PCF.Eval}(\sigma = 1, k_1^{\mathsf{PCF}}, x)$.
2. Compute $(s_x^{(L)}, _) \leftarrow \mathsf{PCF.Eval}(\sigma = 0, k_0^{\mathsf{PCF}}, x)$ (and ignore the second part of the output).
3. Compute $\langle s_x^{(L)} \rangle_0 := \langle s_x^{(L)} \rangle_1 - s_x^{(L)} \cdot \Delta$.
4. Return $y_0 = (s_x^{(L)}, \langle s_x^{(L)} \rangle_0)$.

Since our PCF correctly outputs VOLE-correlated pairs and $\langle s_x^{(L)} \rangle_0$ is completely determined by $\langle s_x^{(L)} \rangle_1$, $s_x^{(L)}$, and Δ, the modification in this hybrid does not change its behavior. Therefore, it is perfectly indistinguishable from Hyb_0.

$\mathsf{Hyb}_2^0 := \mathsf{Hyb}_1$**.**

Hyb_2^l for $l \in [L-1]$: In this game, we change the algorithm $\mathsf{PCF.Gen}$ and $\mathsf{PCF.EvalRec}$ used by the game as follows:

$\mathsf{PCF.Gen}(1^\lambda)$**:**

1. Compute $(k_0^{\mathsf{PCF}}, k_1^{\mathsf{PCF}})$ as before and read Δ from k_1^{PCF}.
2. Modify k_0^{PCF} to include a lazily sampled version of $\llbracket \mathbf{s}^{(l)} \rrbracket_0$ that behaves as follows whenever $\llbracket s_i^{(l)} \rrbracket_0$ is accessed for $i \in [m^{(l)}]$:
   (a) Compute $\llbracket s_i^{(l)} \rrbracket_1 \leftarrow \mathsf{PCF.EvalRec}(\sigma = 1, k_1^{\mathsf{PCF}}, l, i)$.
   (b) Compute and return $\llbracket s_i^{(l)} \rrbracket_0 \leftarrow \mathsf{VOLE}^1_\Delta.\mathsf{RSample}(\sigma = 1, \llbracket s_i^{(l)} \rrbracket_1)$.

$\mathsf{PCF.EvalRec}(\sigma = 0, k_0^{\mathsf{PCF}}, \ell, i)$**:**

1. If $\ell = l$, return $\llbracket s_i^{(\ell)} \rrbracket_0$.
2. Otherwise, behave as usual.

Hyb_2^L**:** For any input $x \in \mathcal{X}$, instead of running $y_0 \leftarrow \mathsf{PCF.Eval}(\sigma = 0, k_0^{\mathsf{PCF}}, x)$, we first run $y_1 \leftarrow \mathsf{PCF.Eval}(\sigma = 1, k_1^{\mathsf{PCF}}, x)$ and then reversely sample and return $y_0 \leftarrow \mathsf{RSample}(\sigma = 1, y_1)$. In this hybrid, we change the oracle $\mathcal{O}_1^{\mathsf{sec}}(x)$ as follows.

1. Compute $y_1 = \left(\Delta, \langle s_x^{(L)} \rangle_1\right) \leftarrow \mathsf{PCF.Eval}(\sigma = 1, k_1^{\mathsf{PCF}}, x)$.
2. $y_0 = \mathsf{VOLE}^1_\Delta.\mathsf{RSample}(\sigma = 1, y_1)$. where Δ is read from k_1^{PCF}.

$\mathsf{Hyb}_3 := \mathsf{Hyb}_2^L$ is identical to the ideal experiment $\mathsf{Exp}_0^{\mathsf{sec}}$ of Definition 9.

It remains to show that Hybrid Hyb_2^l is indistinguishable from Hyb_2^{l-1} for all $l \in [L]$. We prove this for $l \in [L-1]$ in Lemma 4 and for $l = L$ in Lemma 5.

Lemma 4. *Let $l \in [L-1]$. Hybrids Hyb_2^{l-1} and Hyb_2^l are computationally indistinguishable under the assumptions that*

1. *$\mathsf{H}_A^{(l)}$ is a random oracle,*
2. *$\mathsf{SparseLPN}(k^{(l)}, m^{(l-1)}, m^{(l)}, \mathsf{Reg}_t, \mathbb{F}_p)$ is hard, and*
3. *$\mathsf{DMPF}^{(l)}$ is a secure DMPF for t-point functions.*

Proof. We prove the indistinguishability of these two games through the following sequence of hybrid games:

$\mathsf{Hyb}_2^{l-1,0} := \mathsf{Hyb}_2^{l-1}$

$\mathsf{Hyb}_2^{l-1,1}$**:** We modify the $\mathsf{PCF.Gen}$ algorithm to generate the DMPF keys for level l independently of the noise vector by applying the simulator from Definition 7:

$$k_{l,(1)}^{\mathsf{DMPF}} \leftarrow \mathsf{DMPF}^{(\ell)}.\mathsf{Sim}^{(l)}(1^\lambda, \sigma = 1, \mathsf{Leak}_1(\mathbf{e}^{(l)} \cdot \Delta)).$$

Note that due to the changes in Hyb_1, $\langle s_x^{(L)} \rangle_0$ is computed directly from $\langle s_x^{(L)} \rangle_1$, $s_x^{(L)}$, and Δ. Therefore, the intermediate values $\langle s_i^{(l)} \rangle_0$ and thus $k_{l,(0)}^{\mathsf{DMPF}}$ are no longer used by $\mathcal{O}_1^{\mathsf{sec}}(x)$. Hence, we can apply the security property of $\mathsf{DMPF}^{(l)}$ to argue that the two hybrids remain indistinguishable to the adversary who only sees $k_{l,(1)}^{\mathsf{DMPF}}$.

$\mathsf{Hyb}_2^{l-1,2}$**:** In this hybrid, we modify $\mathsf{PCF.EvalRec}$ on level l:

$\mathsf{PCF.EvalRec}(\sigma = 0, k_0^{\mathsf{PCF}}, \ell, i)$**:** If $\ell = l$, sample $s_i^{(l)} \xleftarrow{\$} \mathbb{F}_p$. Otherwise behave as before (cf. hybrid $\mathsf{Hyb}_2^{l-1,1}$ in the proof of Lemma 2).

The proof of indistinguishability to the previous hybrid works the same way as in the corresponding step of the proof of Lemma 2, by invoking the hardness of $\mathsf{SparseLPN}(k^{(l)}, m^{(l-1)}, m^{(l)}, \mathsf{Reg}_t, \mathbb{F}_p)$.

Note that $\mathsf{Hyb}_2^{l-1,2}$ is the same as Hyb_2^l, where the whole secret of the previous level is reverse-sampled. Therefore, combining all of the indistinguishability arguments above, we have $\mathsf{Hyb}_2^{l-1} \approx_c \mathsf{Hyb}_2^l$, which concludes the proof of Lemma 4. □

Lemma 5. *Let $Q_A, Q_B, Q_{\mathcal{O}}$ denote a-priori fixed bounds on the number of queries issued by $\mathcal{A}$ to $\mathsf{H}_A^{(L)}$, H_B and $\mathcal{O}_1^{\mathsf{sec}}$, respectively (see Figure 1). Hybrids Hyb_2^{L-1} and Hyb_2^L are computationally indistinguishable under the assumptions that*

1. *$\mathsf{H}_A^{(L)}$ and H_B are random oracles,*
2. *$\mathsf{SparseLPN}(k^{(L)}, m^{(L-1)}, Q_A Q_B + Q_{\mathcal{O}}, D^{\mathsf{sp}}_{\mathsf{Reg}_t,\beta}, \mathbb{F}_p)$ is hard, and*
3. *$\mathsf{DMPF}^{(L)}$ is a secure DMPF for t-point functions.*

Proof. The proof follows similar steps as the proof of Lemma 4, while applying the same modifications proved in Lemma 3 with respect to the matrix $\mathbf{B}_x$ used for LPN with the sparse-product noise. □

By combining Lemmas 4 and 5 we have proven the second part of Theorem 1, the security property of our PCF. This concludes our security proof.

Remark 2. In the security and pseudorandomness proofs of our PCF, the analysis of the last level is different due to the use of a sparse matrix $\mathbf{B}$, used to select noise entries for arbitrary inputs in a stateless manner. In the asymptotic analysis, this matrix can be replaced by the identity matrix, and therefore, the proofs will follow from the sparse LPN assumption (with regular noise, instead of sparse-product noise). Yet, we keep the analysis as above for generality. In particular, the use of this matrix is useful to get statelessness in concrete instantiations. (See the discussion in *Additional Optimization* in Section 1.5).

4 Sparse LPN Parameters and Efficiency

In this section, we provide an in-depth concrete security analysis of sparse LPN, revisiting and tightening standard heuristics to select parameters.

4.1 Attacking Sparse LPN

Broadly speaking, there are two orthogonal ways to attack sparse LPN:

(1) exploiting the structure of the matrix to identify weaknesses specific to this matrix distribution, or
(2) ignoring the matrix distribution and attacking sparse LPN using any generic LPN attack (that is oblivious to how the matrix is sampled).

At a high level, in all previous works where concrete parameters are chosen for sparse LPN, the parameter selection proceeds always as follows: first, provide some heuristic argument as to why (1) can be dismissed, and (2) select parameters using state-of-the-art tools for evaluating the security of standard (non-sparse) LPN instances. In the rest of this section, we will focus on (1), as (2) refers simply to the state-of-the-art cryptanalysis of the standard LPN assumption, and revisiting the security of LPN in its full generality would be way outside of the scope of this work.

The argument for (1) relies on two ingredients: a conjecture (made explicitly or implicitly in multiple works) that LPN variants (such as sparse LPN) are as secure as standard LPN as long as *the dual of the code generated by* $\mathbf{A}$ *is a good code* (equivalently, there is no non-zero vector $\mathbf{v}$ in the left kernel of $\mathbf{A} \in \mathbb{F}^{m \times n}$ with Hamming weight $w \ll n$ – we call *dual distance* and denote $\mathsf{dd}(\mathbf{A})$ the smallest weight of any non-zero vector in $\mathbf{A}$'s kernel), and a proof that a random sparse matrix has large dual distance with high probability. Concretely, this relies on the following lemma (here stated for $\mathbb{F}_2$ and reproduced almost verbatim from [BCM25, Thm. 4.5]):

Lemma 6 (Most sparse matrices have large dual distance [BCM25]). *For any $k \geq \omega(1)$ and any constant $\eta \in (0,1)$, there is a function $\gamma(k) = k^{1+o(1)}$ such that for any large enough n, for any $m \leq n^{(1-\eta)k/2+\eta}$,*

$$\Pr\left[\mathbf{A} \xleftarrow{\$} \mathsf{SparseCodeGen}(k,m,n,\mathbb{F}_2) \ : \ \mathsf{dd}(\mathbf{A}) \geq \frac{n^\eta}{\gamma(k)} - 1\right] \geq 1 - \left(\frac{\gamma(k)}{n^\eta}\right)^{k-2}.$$

The proof of Lemma 6 generalizes an earlier analysis of [MST03] for a more restricted set of parameters, and can be found in multiple papers [CRR21; BCM25; DIJL23; Boy+23].

4.2 Security From High Dual Distance

The conjecture that a high dual distance is "all that is needed" was first made explicitly in [ADINZ17]; see also [BCM25] for a more precisely quantified version of the conjecture. Note that the conjecture is not fully formal: there are known counter-examples (typically, Reed-Solomon codes), but they are highly algebraic codes tailor-made to admit algebraic decoders. The conjecture remains nevertheless plausible for essentially all codes that do not possess this specific algebraic structure (and in particular all "combinatorial" codes). The works of [Boy+20a; CRR21] provide support for this conjecture by introducing the *linear test framework*. They observed that most (but not all) attacks on LPN fall in this framework, and that high dual distance provably suffices to be secure against attacks from the linear test framework (see Lemma 5.3 in [BCM25] for a precise statement of this result). These results have been regularly invoked to back-up the heuristic approach to parameter selection outlined previously.

There is, however, a subtle but important discrepancy between the guarantees offered by the proof of security in the linear test framework, and the concrete parameter selection: the proof of security against linear tests shows that, for a large class of attacks, the runtime of the attack must at least grow as $\exp(\mathsf{dd}(\mathbf{A}) \cdot t/m)$ (where t is the weight of the error vector). For *standard* LPN, where $\mathbf{A}$ is a random matrix, $\mathsf{dd}(\mathbf{A})$ approaches (rapidly) $n/2$ when $m \gg n$ (more precisely, $\mathsf{dd}(\mathbf{A})$ is given by the Gilbert-Varshamov bound). But for other matrix distributions, $\mathsf{dd}(\mathbf{A})$ can be much smaller: following precisely the linear test framework would therefore require computing *precisely* $\mathsf{dd}(\mathbf{A})$ and scaling up t by the ratio $(n/2)/\mathsf{dd}(\mathbf{A})$ to match the security of LPN with a random code. This is never done in practice: first, in previous work, no tight bound on $\mathsf{dd}(\mathbf{A})$ is known (Lemma 6 provides only a very loose bound), and second, applying such a scaling factor would make t huge, rendering the construction inefficient.

If an adversary could efficiently find vectors of Hamming weight $\mathsf{dd}(\mathbf{A})$ in the kernel of $\mathbf{A}$, then the bound $\exp(\mathsf{dd}(\mathbf{A}) \cdot t/m)$ of the linear test framework would be essentially tight, since the attack given these vectors $\mathbf{v}_i$ is efficient (given $(\mathbf{A}, \mathbf{b})$ check whether $\mathbf{v}_i^\intercal \cdot \mathbf{b} = 0$ for noticeable less than half of

the $\mathbf{v}_i$'s). Therefore, in addition to the security argument against linear tests, these works implicitly rely on the following informal assumption: *provided that* $\mathsf{dd}(\mathbf{A})$ *is not too small in the first place, it should be infeasible for an efficient adversary to find vectors with Hamming weight significantly smaller than* $n/2$ *in the kernel of* $\mathbf{A}$. This assumption – that low-weight vectors in the kernel of $\mathbf{A}$ are hard to find even when they exist – was explicitly required and studied in multiple works [ABW10; AHIKV17; DJ24]; it was called the *binary-SVP* assumption in [AHIKV17]. In light of the current state of the art, this is a very reasonable assumption. However, because Lemma 6 does not provide a meaningful bound for concrete parameters due to its looseness, the use of a flavor of binary-SVP in the parameter selection process is never *quantified*, making it hard to assess whether a set of parameters should be considered conservative or aggressive, or even to compare two parameter sets. The right parameter here is the quantity $\mathsf{dd}(\mathbf{A}) \cdot t/m$, that measures the gap between the security offered by the inexistence of vectors of weight below $\mathsf{dd}(\mathbf{A})$ in $\mathbf{A}$'s kernel, and the security that stems from the hardness of finding such vectors (where t, m are chosen "as if" $\mathsf{dd}(\mathbf{A})$ was equal to the optimal value $n/2$, typically by setting $(n/2) \cdot t/m$ to be some security parameter).

4.3 Computing the Dual Distance

In this work, motivated by the above discussion, we revisit the standard parameter selection process and introduce a much tighter methodology to derive a bound on $\mathsf{dd}(\mathbf{A})$. To this end, we derive formulae (indexed by a weight parameter w) that compute a close estimate of the probability that a vector of Hamming weight w is in the kernel of $\mathbf{A}$. To evaluate the formula (that does not admit a simple closed-form) for the $\mathbb{F}_2$ case, we wrote a (Python) script[11], and computed (a reasonably tight lower bound on) the concrete dual distance that our parameters achieve with good probability over the random choice of the sparse matrix $\mathbf{A}$ (we fix a statistical failure probability $2^{-\rho}$ to either 2^{-40} or 2^{-64}). Based on this, we were able to derive concrete parameters for each level of our iterative PCF construction, while providing strong support for the claim that all parameter sets provide a comparable security level. Using our tool, we also computed the dual distance of sparse LPN instance used in previous works [YWLZW20; WYKW21; YSWW21], allowing to compare precisely the conservativeness of our parameters compared to these works. Below, we elaborate on our analysis. For simplicity, we describe the analysis in the slightly easier setting where the matrix $\mathbf{A}$ is a *regular* sparse matrix, whose rows are sampled as a concatenation of k random unit vectors. However, the analysis extends immediately to the case where the each row of $\mathbf{A}$ is sampled as the sum of k random unit vectors.

Security in the Linear Test Framework. For concreteness, we briefly recall a high-level intuition of the linear test framework. Informally, for the matrix $\mathbf{A} \leftarrow \mathsf{CodeGen}(m, n, \mathbb{F})$, the goal of an unbounded adversary $\mathcal{A}$ in an attack is to find a non-zero vector $\mathbf{v}$ for the distribution $\mathcal{D}_{\mathbf{A}} = \{\mathbf{As} + \mathbf{e} \mid \mathbf{e} \leftarrow \mathsf{Noise}_t, \mathbf{s} \leftarrow \mathbb{F}^n\}$ such that the probability $\mathbf{v}^\top \cdot \mathbf{x} = 0$ for $\mathbf{x} \leftarrow \mathcal{D}_{\mathbf{A}}$ is noticeably different from $\frac{1}{2}$. A variant of LPN is said to be secure against linear tests if the number of such vectors is low with high probability, meaning that an $\mathbf{A} \leftarrow \mathsf{CodeGen}$ is likely to be "good."

Intuitively, if $\mathbf{v}^\top \mathbf{A} \neq \mathbf{0}$ then $\mathbf{v}^\top \mathbf{As}$ is uniform. Conversely, when $\mathbf{v}^\top \mathbf{A} = \mathbf{0}$, the sparsity of $\mathbf{v}$ determines the probability that $\mathbf{v}^\top \cdot \mathbf{e} = \mathbf{0}$; for sparse noise distributions, the sparser $\mathbf{v}$ the further the product $\mathbf{v}^\top \cdot \mathbf{x}$ is biased towards 0. Towards parameter selection, we bound the probability that a non-zero $\mathbf{v} \in \mathbb{F}^m$ with low Hamming weight $\omega(\mathbf{v})$ exists such that $\mathbf{v}^\top \mathbf{A} = \mathbf{0}$ for an $\mathbf{A} \leftarrow \mathsf{RegularCodeGen}(m, n, k, \mathbb{F})$ by $2^{-\rho}$.

A Detailed Analysis of the Dual Distance of A over $\mathbb{F}_2$. We fix the parameters; for a single level ℓ, let $k := k^{(\ell)}$, $m := m^{(\ell)} = m(\cdot), n := m^{(\ell-1)}$ and, for ease of analysis, $\mathbb{F} := \mathbb{F}_2$. Consider an unbounded adversary $\mathcal{A}$ in the following linear test attack:

1. Sample $\mathbf{A} \leftarrow \mathsf{RegularCodeGen}(m, n, k, \mathbb{F}_2)$,
2. The adversary outputs a non-zero vector $\mathbf{v} \leftarrow \mathcal{A}(\mathbf{A})$ such that $\omega(\mathbf{v}) = 2w$.[12]
3. The adversary wins if $\mathbf{v}^\top(\mathbf{As} + \mathbf{e}) = 0$ for $\mathbf{s} \leftarrow \mathbb{F}_2^n$, $\mathbf{e} \leftarrow \mathsf{Reg}_t(m, \mathbb{F}_2)$ (or $\mathbf{e} \leftarrow D^{\mathsf{sp}}_{\mathsf{Reg}_t,\beta}(m, \mathbb{F}_2)$ for $\ell = L$).

[11] https://github.com/IRIFCrypto/dualdistance/

[12] We can assume $\omega(\mathbf{v})$ to be even because a vector of odd Hamming weight can never cancel out $\mathbf{A}$: the xor of an odd number of vectors with odd Hamming weight again has odd Hamming weight.

To determine the advantage of $\mathcal{A}$, we compute the probability that there exists a vector $\mathbf{v}$ with Hamming weight $0 < \omega(\mathbf{v}) < d$, as in Eq (1), in terms of $p_{w,0}$, the probability that the xor of $2w$ random unit vectors of length $N := \frac{n}{k}$ is $\mathbf{0}$:

$$\Pr_{\mathbf{A}}\left[\exists \mathbf{v} \in \mathbb{F}^m : \begin{matrix} 0 < \omega(\mathbf{v}) < d \\ \mathbf{v}^\top \mathbf{A} = \mathbf{0} \end{matrix}\right] \leq \sum_{w=1}^{\lfloor (d-1)/2 \rfloor} \Pr_{\mathbf{A}}\left[\exists \mathbf{v} \in \mathbb{F}^m : \begin{matrix} \mathbf{v}^\top \mathbf{A} = \mathbf{0} \\ \omega(\mathbf{v}) = 2w \end{matrix}\right] \tag{1}$$

$$\leq \sum_{w=1}^{\lfloor (d-1)/2 \rfloor} \binom{m}{2w} \Pr_{\mathbf{A}}\left[\mathbf{v}^\top \mathbf{A} = \mathbf{0}\right] \tag{2}$$

$$= \sum_{w=1}^{\lfloor (d-1)/2 \rfloor} \binom{m}{2w} (p_{w,0})^k \tag{3}$$

The first inequality, Eq (1), is the result of computing the probability that there exists a vector $\mathbf{v}$ of Hamming weight $2w = \omega(\mathbf{v})$ which cancels a random $\mathbf{A}$, and applying the union bound over $w \in [\lfloor (d-1)/2 \rfloor]$.

Observe that, as the rows of $\mathbf{A}$ are independent, the probability that $\mathbf{v}^\top \mathbf{A} = \mathbf{0}$ for any Hamming weight $2w$, is independent of the choice of $\mathbf{v}$ with Hamming weight $\omega(\mathbf{v}) = 2w$. The second inequality, Eq (2), is the result of computing the probability that a *fixed* vector $\mathbf{v}$ of Hamming weight $2w$ cancels $\mathbf{A}$, and applying the union bound over all $\binom{m}{2w}$ choices of $\mathbf{v}$.

As $\mathbf{A}$ is k-regular, each row of $\mathbf{A}$ is independently sampled as $\mathsf{Reg}_k(n, \mathbb{F})$. Therefore, we can focus on a single block of unit vectors. Let $p_{w,u}$ be the probability that the xor of $2w$ random unit vectors of length $N := \frac{n}{k}$ has Hamming weight $2u$, so when $u = 0$ then the xor is $\mathbf{0}$. The final equality, Eq (3), is the probability that each of the k blocks of each row cancels out. Before we give the general formula for $p_{w,u}$, we note the following edge cases:

$$p_{1,0} = \frac{1}{N}, \qquad p_{w,w} = \prod_{i=1}^{2w-1} \frac{N-i}{N}, \qquad p_{w,u} = 0 \text{ for } u > w.$$

Now we recursively compute $p_{w,u}$:

$$p_{w,u} := \Pr_{\mathbf{a}_i}\left[\omega\left(\bigoplus_{i \in [2w]} \mathbf{a}_i\right) = 2u\right] \tag{4}$$

$$= \Pr_{\mathbf{a}_i}\left[\underbrace{\begin{pmatrix} a_1 \oplus a_2 = \mathbf{0} \\ \wedge \\ \omega\left(\bigoplus_{i \in [3,2w]} \mathbf{a}_i\right) = 2u \end{pmatrix}}_{\mathtt{E}_1} \vee \underbrace{\begin{pmatrix} a_1 \oplus a_2 \neq \mathbf{0} \\ \wedge \\ \omega\left(\bigoplus_{i \in [2w]} \mathbf{a}_i\right) = 2u \end{pmatrix}}_{\mathtt{E}_2}\right] \tag{5}$$

$$= \underbrace{\frac{p_{w-1,u}}{N}}_{\mathtt{E}_1} + \underbrace{\frac{N-1}{N}}_{\mathtt{E}_{2a}} \left(\underbrace{c_{u,-1} \cdot p_{w-1,u-1}}_{\mathtt{E}_{2b.i}} + \underbrace{c_{u,0} \cdot p_{w-1,u}}_{\mathtt{E}_{2b.ii}} + \underbrace{c_{u,1} \cdot p_{w-1,u+1}}_{\mathtt{E}_{2b.iii}}\right) \tag{6}$$

$$= \frac{p_{w-1,u}}{N} + \frac{1}{N^2} \begin{pmatrix} (N-2u+2)(N-2u+1) \cdot p_{w-1,u-1} \\ + 4u(N-2u) \cdot p_{w-1,u} \\ + (2u+2)(2u+1) \cdot p_{w-1,u+1} \end{pmatrix} \tag{7}$$

Our scenario can be conceptualized via a balls and bins analysis; each index $j \in [N]$ is a bin, and each unit vector $\mathbf{a}_i \in \mathbb{F}_2^N$ is a ball $i \in [2w]$ which lands in bin j_i if $\mathbf{a}_i[j_i] = 1$. Rephrasing, we compute $p_{w,u}$ as the probability that $2u$ of the N bins are each hit by an odd number of the $2w$ balls.

As in Eq (5), the probability $p_{w,u}$ can be expressed as the probability that one of two events occurs, $p_{w,u} = \Pr[\mathtt{E}_1 \vee \mathtt{E}_2]$, where

$\mathtt{E}_1$**:** (a) the first two balls land in the same bin, and (b) of the remaining $2(w-1)$, an odd number of balls land in $2u$ distinct bins,

$$\Pr[\mathtt{E}_1] = \Pr[\mathtt{E}_{1a} \wedge \mathtt{E}_{1b}] = \left(\frac{1}{N}\right) \cdot p_{w-1,u}.$$

$\mathtt{E}_2$: (a) the first two balls land in two distinct bins, and (b) exactly $2u$ distinct bins contain an odd number of the $2w$ balls,

$$\Pr[\mathtt{E}_2] = \Pr[\mathtt{E}_{2a} \wedge \mathtt{E}_{2b}] = \left(\frac{N-1}{N}\right) \cdot \Pr[\mathtt{E}_{2b}].$$

For the event $\mathtt{E}_2$, we assume that the first pair of balls landed in distinct bins $B_1 = \{j_1, j_2\} \subset [N]$, and denote by $B_{[2,w]} \subseteq [N]$ the set of bins that contain an odd number of the remaining balls. As shown in Eq (6), event $\mathtt{E}_{2b}$ occurs in one of three scenarios $(\mathtt{E}_{2b.i}, \mathtt{E}_{2b.ii}, \mathtt{E}_{2b.iii})$.

$\mathtt{E}_{2b.i}$: $\left|B_{[2,w]}\right| = 2(u-1)$ and $B_{[2,w]} \cap B_1 = \emptyset$, meaning that

- an odd number of the remaining $2(w-1)$ balls land in $2(u-1)$ of the bins,
- and these $2(u-1)$ bins do not contain the first two balls:

$$\Pr[\mathtt{E}_{2b.i} \mid \mathtt{E}_{2a}] = c_{u,-1} \cdot p_{w-1,u-1} \text{ for } c_{u,-1} := \frac{\binom{N-2}{2(u-1)}}{\binom{N}{2(u-1)}} = \frac{(N-2u+2)(N-2u+1)}{N(N-1)}.$$

$\mathtt{E}_{2b.ii}$: $\left|B_{[2,w]}\right| = 2u$ and $\left|B_{[2,w]} \cap B_1\right| = 1$, meaning

- an odd number of the remaining $2(w-1)$ balls land in $2u$ of the bins,
- and one of these $2u$ bins contains one of the first two balls:

$$\Pr[\mathtt{E}_{2b.ii} \mid \mathtt{E}_{2a}] = c_{u,0} \cdot p_{w-1,u} \text{ for } c_{u,0} := 2 \cdot \frac{\binom{N-2}{2u-1}}{\binom{N}{2u}} = \frac{4u\,(N-2u)}{N\,(N-1)}.$$

$\mathtt{E}_{2b.iii}$: $\left|B_{[2,w]}\right| = 2\,(u+1)$ and $B_1 \subset B_{[2,w]}$, meaning

- an odd number of the remaining $2(w-1)$ balls land in $2(u+1)$ of the bins,
- and two of these $2(u+1)$ bins each contain one of the first two balls:

$$\Pr[\mathtt{E}_{2b.iii} \mid \mathtt{E}_{2a}] = c_{u,1} \cdot p_{w-1,u+1} \text{ for } c_{u,1} := \frac{\binom{N-2}{2u}}{\binom{N}{2(u+1)}} = \frac{(2u+2)(2u+1)}{N(N-1)}.$$

A formula for the desired probability appears in Eq (7), which we compute recursively to solve for D as follows:

$$D := \arg\max_{d>0} \left(\sum_{w=1}^{\lfloor (d-1)/2 \rfloor} \binom{m}{2w} \cdot (p_{w,0})^k \leq 2^{-\rho} \right).$$

Note that the analysis extends immediately to the case where each row of $\mathbf{A}$ is a sum of k independent length-n unit vectors, by replacing $p_{w,0}^k$ with $p_{w \cdot k,0}$ in the above formula.

Analysis Over a General Field. Our analysis above exploits properties of $\mathbb{F}_2$ (e.g., cancellations are more likely compared to a general field and only even-weight vectors matter). This allows us to run our implementation to run up to large values of n (see below), but makes it incompatible with general fields $\mathbb{F}_p$.

However, we expect the bound obtained through our analysis over $\mathbb{F}_2$ to provide a lower bound on that over larger fields. Intuitively, as the field grows, canceling rows gets harder; it no longer suffices for balls to land in the same bins but they must have opposite values. All previous works assume that the bound does not degrade as the field size grows and a common heuristic is, therefore, to use the bounds from the analysis over $\mathbb{F}_2$ [ADINZ17; BCGI18]. We note that the asymptotic formulas from previous works give (loose) bounds for arbitrary fields [CRR21].

For completeness we provide formulae to bound the dual distance of $\mathbf{A} \xleftarrow{\$} \mathsf{SparseCodeGen}()$ over a general field $\mathbb{F}_p$ in Section 4.5. Using these to obtain concrete bounds, however, results in computational complexity at least cubic with n which not feasible for the parameter ranges that we are interested in.

Numbers. We implemented our algorithm for $\mathbb{F}_2$ to find D and $\delta := D/n$ for different choices of (n, m, k) and statistical security parameter ρ and list the results in Table 2. Here δ is the dual distance relative to the LPN dimension n and allows us to compare two settings with different dimensions. Our results show that for a fixed ratio n/m and sparsity k, δ converges quickly and monotonically as n grows. Therefore, for values of n, that would be too large to run on our computer, we used the δ of the largest n with the same ratio and sparsity that we could compute (it provides a very tight lower bound on the real δ).

Note that the numbers for D and δ provided in the conference version of this paper [BCMRS25] were too small by a factor 2. Table 2 has been corrected and extended, and our parameter choices in Section 4.4 have been updated accordingly.

Table 2. Lower bounds D on the dual distance $\mathsf{dd}(\mathbf{A})$ and $\delta := D/n$, for different parameter sets (n, m, k) and statistical security parameter ρ. $D = 0$ indicates that we can only guarantee a trivial bound.

n	$k=5$		$k=6$		$k=7$		$k=8$		$k=9$		$k=10$	
	D	δ	D	δ	D	δ	D	δ	D	δ	D	δ
$s = m/n = 32$, $\rho = 40$												
2^{12}	0	0	0	0	0	0	0	0	272	0.066 4	302	0.073 7
2^{13}	0	0	0	0	0	0	480	0.058 6	554	0.067 6	612	0.074 7
2^{14}	0	0	0	0	780	0.047 6	968	0.059 1	1 118	0.068 2	1 234	0.075 3
2^{15}	0	0	0	0	1 574	0.048 0	1 948	0.059 4	2 244	0.068 5	2 478	0.075 6
2^{16}	0	0	0	0	3 158	0.048 2	3 904	0.059 6	4 498	0.068 6	4 964	0.075 7
2^{17}	0	0	4 504	0.034 4	6 324	0.048 2	7 818	0.059 6	9 006	0.068 7	9 938	0.075 8
2^{18}	0	0	9 022	0.034 4	12 660	0.048 3	15 648	0.059 7	18 020	0.068 7	19 886	0.075 9
2^{19}	0	0	18 058	0.034 4	25 332	0.048 3	31 304	0.059 7	36 050	0.068 8	39 778	0.075 9
2^{20}	0	0	36 126	0.034 5	50 676	0.048 3	62 618	0.059 7	72 108	0.068 8	79 566	0.075 9
2^{21}	39 716	0.018 9	72 266	0.034 5	101 362	0.048 3	125 244	0.059 7	144 226	0.068 8	159 142	0.075 9
2^{22}	79 446	0.018 9	144 542	0.034 5	202 732	0.048 3	250 496	0.059 7	288 460	0.068 8	318 294	0.075 9
2^{23}	158 908	0.018 9	289 096	0.034 5	405 476	0.048 3	501 002	0.059 7	576 930	0.068 8	636 596	0.075 9
2^{24}	317 832	0.018 9	578 204	0.034 5	810 960	0.048 3	1 002 014	0.059 7	1 153 870	0.068 8	1 273 198	0.075 9
2^{25}	635 678	0.018 9	1 156 420	0.034 5	1 621 932	0.048 3	2 004 036	0.059 7	2 307 750	0.068 8	2 546 406	0.075 9
2^{26}	1 271 368	0.018 9	2 312 850	0.034 5	3 243 872	0.048 3	4 008 080	0.059 7	4 615 508	0.068 8	5 092 820	0.075 9
$s = m/n = 32$, $\rho = 64$												
2^{12}	0	0	0	0	0	0	0	0	0	0	0	0
2^{13}	0	0	0	0	0	0	0	0	0	0	0	0
2^{14}	0	0	0	0	0	0	0	0	0	0	1 230	0.075 1
2^{15}	0	0	0	0	0	0	0	0	2 238	0.068 3	2 472	0.075 4
2^{16}	0	0	0	0	0	0	0	0	4 492	0.068 5	4 960	0.075 7
2^{17}	0	0	0	0	0	0	7 812	0.059 6	9 000	0.068 7	9 934	0.075 8
2^{18}	0	0	0	0	0	0	15 642	0.059 7	18 016	0.068 7	19 880	0.075 8
2^{19}	0	0	0	0	25 326	0.048 3	31 298	0.059 7	36 044	0.068 7	39 774	0.075 9
2^{20}	0	0	0	0	50 668	0.048 3	62 612	0.059 7	72 104	0.068 8	79 562	0.075 9
2^{21}	0	0	0	0	101 354	0.048 3	125 238	0.059 7	144 220	0.068 8	159 138	0.075 9
2^{22}	0	0	0	0	202 726	0.048 3	250 490	0.059 7	288 456	0.068 8	318 288	0.075 9
2^{23}	0	0	289 088	0.034 5	405 468	0.048 3	500 996	0.059 7	576 926	0.068 8	636 590	0.075 9
2^{24}	0	0	578 196	0.034 5	810 954	0.048 3	1 002 008	0.059 7	1 153 866	0.068 8	1 273 194	0.075 9
2^{25}	0	0	1 156 410	0.034 5	1 621 924	0.048 3	2 004 030	0.059 7	2 307 744	0.068 8	2 546 400	0.075 9
2^{26}	0	0	2 312 842	0.034 5	3 243 866	0.048 3	4 008 074	0.059 7	4 615 502	0.068 8	5 092 814	0.075 9

4.4 Concrete Parameter Sets

Parameter Choices. Based on the discussions of the previous sections, we now propose concrete parameter sets for the $\lambda = 128$ bit security level. We use an initial secret of size $n = m^{(0)} = 2^{15}$ and apply our iterative construction with $L \in \{3, 4\}$ levels and stretch $s = 2^5$ resulting in $m^{(1)} = 2^{20}$, $m^{(2)} = 2^{25}$, $m^{(3)} = 2^{30}$, and $m^{(4)} = 2^{35}$. We use the LPN estimator[13] of [LWYY24] to find a value of $t = 2700$ providing at least λ bits of security for each level.[14]

For each setting, we provide two sets of sparsity parameters based on our observations above: The first is more conservative. For each LPN dimension $n = m^{(\ell-1)}$, we find in Table 2 the value of $k = k^{(\ell)}$ that results in $D > 0$ and target a value of $\delta \approx 0.059$ or higher. This choice of δ targets security guarantees comparable to sparse LPN instances used in other works [YWLZW20; WYKW21; YSWW21]: for our parameters, we have $\mathsf{dd}(\mathbf{A}) \cdot t/m \approx 5$, while [YWLZW20; WYKW21; YSWW21] have $\mathsf{dd}(\mathbf{A}) \cdot t/m \approx 6.5$ (cf. Section 4.2).

For the second set, we take the smallest k that results in $D > 0$, as we observe a sharp transition from $D = 0$ to $\delta > 0.018$ in our experiments. This corresponds to making the stronger (but still plausible) assumption that as soon as $\mathsf{dd}(\mathbf{A})$ grows as $\delta \cdot n$ for a not-too-small constant δ (large enough that existing algorithms for finding low-weight vectors in the kernel of a matrix are infeasible for a weight $\delta \cdot n$), the fastest attack remains a direct attack on the standard LPN assumption.

Stateful vs. Stateless. As in previous works [Boy+20a; Boy+22], we assume in the concrete instantiation that the parties maintain a state (a counter i) and use it as input to the PCF to avoid input collisions.[15] Thus we set $\mathcal{X} = [m^{(L)}]$, let $\mathsf{H}_B(i)$ output the ith unit vector, and evaluate the PCF on input i to obtain the ith sample.

If a state is undesirable, the parties can instead use a sparse-product noise distribution and sample the rows of $\mathbf{B}$ randomly, with virtually no impact on concrete efficiency (but at the cost of making a new assumption): picking $\mathbf{B}$ to have row-weight 9 suffices to achieve a 2^{-40} collision probability[16] and adds only 8 DMPF evaluations to the total. Additionally, the stateless version requires that the parties are given access to a trusted source of randomness as inputs (or, alternatively, that they run a coin-tossing protocol). In most applications, we expect maintaining a counter as a state to the simplest solution.

Precomputation and Batched Evaluation. Note that each PCF evaluation depends on $\prod_{j=\ell+1}^{L} k^{(j)}$ entries of the error vector $[\![\mathbf{e}^{(\ell)}]\!]_\sigma$, which are computed by evaluating the corresponding DMPF. Thus most of the DMPF evaluation are related to $[\![\mathbf{e}^{(1)}]\!]_\sigma$ and we can avoid them by precomputing $[\![\mathbf{e}^{(1)}]\!]_\sigma$ (and storing it as part of the PCF key). This precomputation costs one full evaluation of the DMPF with domain size $m^{(1)}$ (which can be implemented faster than $m^{(1)}$ separate evaluations), and $|\langle \mathbf{e}^{(1)} \rangle_\sigma| = m^{(1)} \cdot r \cdot \log p$ bits of storage. Now each PCF evaluation costs only $\sum_{\ell=2}^{L-1} \prod_{j=\ell+1}^{L} k^{(j)} + k^B$ DMPF evaluations. More concretely, for $L = 3$, this reduces the number of DMPF evaluations per PCF evaluation from $k^{(2)} \cdot k^{(3)} + k^{(3)} + k^B$ to $k^{(3)} + k^B$.

Similarly, when the PCF needs to be evaluated on large batches of inputs – and there is sufficient memory available – it quickly becomes faster to first compute the whole vector $\langle \mathbf{e}^{(1)} \rangle_\sigma$ and then use it for the whole batch of evaluations. For example, for $L = 4$, $\rho = 40$, using sparsity $(7, 6, 5, 5)$, precomputation outperforms direct computation as soon as the batch size exceeds 2^{11}.

Key Sizes and Runtime Estimates. For the parameter choices detailed above, we give the PCF key sizes as well as the required PRG evaluations in Table 3.

We convert the latter into runtime estimates as follows: On a modern x86 CPU, fixed-key AES can achieve a throughput of 1.3 cycles per byte [MSY21] or 20.8 cycles per AES call. We obtain runtimes by

[13] https://lpnestimator.com/

[14] Since the dimension increases in each level, we could choose smaller values of t. For example, on level $\ell = 2$, $t = 1675$ would be sufficient according to the LPN estimator. This results in a tradeoff between key size and evaluation costs.

[15] I.e., associating the same row of the virtual vector $\mathbf{A}^{(L)} \cdot \mathbf{s}^{(L-1)} + \mathbf{e}^{(L)}$ with two different inputs. In the asymptotic analysis, such input collisions happen with negligible probability since $m^{(L)}$ is superpolynomially large in n.

[16] For $\mathbf{B} \in \mathbb{F}_2^{m^{(L)} \times m^{(L)}}$ with $m^{(L)} = 2^{35}$ and row-weight 9, an adversary making 2^{128} queries to H_B obtains a collision with probability at most $(2^{128})^2 / \binom{2^{35}}{9} \approx 2^{-40}$.

assuming a CPU frequency of 3.8 GHz. For Eval and PrepEval we only need one call to AES per PRG evaluations, since only one half of the PRG output is required. For the preprocessing, we need both halves and, therefore, have to count two calls to AES per PRG evaluation.

Table 3. Key sizes and computation costs for our VOLE PCF over $\mathbb{F}_{2^{128}}$, with parameters set to $\lambda = 128$, $n = 2^{15}$, $t = 2700$, $s = 32$, $k^B = 1$. Preprocessing of the innermost layer requires 1 382 400 PRG evaluations which can be done in $\approx 15\,133$ µs and has size 16 384 KiB. The gray rows show how the corresponding computation cost *without* tuning the sparsity parameter at each level.

L	$(k^{(\ell)})_\ell$	Key Sizes (KiB)		Computation (# PRG Eval, µs)			
		$\lvert k_0^{\mathsf{PCF}}\rvert$	$\lvert k_1^{\mathsf{PCF}}\rvert$	Eval		PrepEval	
$\rho = 40$							
3	(8, 8, 8)	2812	638	707	3.87	131	0.72
	(7, 6, 5)			359	1.97	89	0.49
	(7, 7, 7)			558	3.05	117	0.64
4	(8, 8, 8, 8)	4002	680	5 680	31.09	1 072	5.87
	(7, 6, 5, 5)			1 819	9.96	469	2.57
	(7, 7, 7, 7)			3 930	21.51	843	4.61
$\rho = 64$							
3	(9, 8, 8)	2812	638	707	3.87	131	0.72
	(9, 7, 6)			481	2.63	103	0.56
	(9, 9, 9)			874	4.78	145	0.79
4	(8, 8, 8, 8)	4002	680	5 680	31.09	1 072	5.87
	(9, 7, 6, 6)			2 910	15.93	642	3.51
	(9, 9, 9, 9)			7 890	43.19	1 329	7.27

4.5 Computing the Dual Distance Over a General Field

In Section 4.3 we gave explicit formulae to compute the dual distance of regular random codes $\mathbf{A} \xleftarrow{\$} \mathsf{RegularCodeGen}(k, m, n, \mathbb{F}_2)$ (Definition 4) over $\mathbb{F}_2$. For completeness, we now provide an analysis for arbitrary sparse codes $\mathbf{A} \xleftarrow{\$} \mathsf{SparseCodeGen}(k, m, n, \mathbb{F}_p)$ (Definition 3) over arbitrary prime fields $\mathbb{F}_p$.

Let $p_{w,u}$ be the probability that the sub of w random weight-k vectors of length n has Hamming weight u.[17] While we are ultimately interested in $p_{w,0}$, we need the general case as well for the recursive formula to work.

From there, we can compute a lower bound D on the minimum distance via a union bound over all attack vectors $\mathbf{v} \in \mathbb{F}_p^m$ of weight $\omega(\mathbf{v}) = w$. This bound D is the largest number such that the probability (over the choice of the matrix $\mathbf{A}$) that there exists a $\mathbf{v}$ with $\omega(\mathbf{v}) < D$ is at most $2^{-\rho}$:

$$D = \arg\max_{d>0} \left(\sum_{w=1}^{d-1} \binom{m}{w} \cdot p_{w,0} \cdot (p-1)^w \leq 2^{-\rho} \right).$$

In the following let $0 \leq a, b, c \leq n$ with $c \leq a + b$.

Hamming Weight of the Sum of Two Vectors. Define

$$G_{a,b}(c) := \Pr_{\mathbf{u},\mathbf{v}}[\omega(\mathbf{u} + \mathbf{v}) = c \mid \omega(\mathbf{u}) = a \wedge \omega(\mathbf{v}) = b],$$

where $\mathbf{u}$ and $\mathbf{v}$ are independently sampled uniformly at random from $\mathbb{F}_p^n$. In order to compute $G_{a,b}(c)$, we need to account for (i) the overlap size of two vectors, and (ii) cancellations in their sum.

[17] Note that in contrast to the $\mathbb{F}_2$ case, we now also have to consider attack vectors $\mathbf{v}$ of odd Hamming weight, so the meaning of $p_{w,u}$ differs slightly compared to Section 4.3.

Overlap Size. Let $R_{a,b}$ denote the size of the overlap of $\mathbf{u}$ and $\mathbf{v}$, i.e., the number of indices where both $\mathbf{u}$ and $\mathbf{v}$ have non-zero entries:

$$R_{a,b} := |\operatorname{sup}(\mathbf{u}) \cap \operatorname{sup}(\mathbf{v})|.$$

The size of the overlap of $\mathbf{u}$ and $\mathbf{v}$ follows a hypergeometric distribution:

$$\Pr[R_{a,b} = r] = \frac{\binom{a}{r} \cdot \binom{n-a}{b-r}}{\binom{n}{b}}, \qquad \text{for } r \in [\max(0, a+b-n), \min(a,b)].$$

Cancellations. To account for cancellations, note that for each coordinate in the overlap $i \in \operatorname{sup}(\mathbf{u}) \cap \operatorname{sup}(\mathbf{v})$ independently it holds that[18]

$$\Pr[u_i + v_i = 0] = \frac{1}{p-1}, \qquad \text{and} \qquad \Pr[u_i + v_i \neq 0] = \frac{p-2}{p-1}.$$

Thus the number of cancellations C_r (in r overlapping coordinates) is binomially distributed ($C_r \sim \mathsf{Bin}(r, \frac{1}{p-1})$) and satisfies

$$\Pr[C_r = i] = \binom{r}{i} \cdot \left(\frac{1}{p-1}\right)^i \cdot \left(\frac{p-2}{p-1}\right)^{r-i} \qquad \text{for } i \in [0, r].$$

As we are interested in computing the probability of $\omega(\mathbf{u}+\mathbf{v}) = c$, we need to set i accordingly, i.e.,

$$(a-r) + (b-r) + (r-i) = c \implies i = a+b-c-r.$$

Finally, combining the above gives the following formula for $G_{a,b}(c)$:

$$\begin{aligned} G_{a,b}(c) &= \sum_{r=r_{\min}}^{r_{\max}} \Pr[R_{a,b} = r] \cdot \Pr[C_r = a+b-c-r] \\ &= \sum_{r=r_{\min}}^{r_{\max}} \frac{\binom{a}{r} \cdot \binom{n-a}{b-r}}{\binom{n}{b}} \cdot \binom{r}{a+b-c-r} \cdot \left(\frac{1}{p-1}\right)^{a+b-c-r} \cdot \left(\frac{p-2}{p-1}\right)^{2r-(a+b-c)} \end{aligned}$$

where

$$r_{\min} = \max\left(0, a+b-n, \left\lceil \frac{a+b-c}{2} \right\rceil\right) \quad \text{and} \quad r_{\max} = \min(a, b, a+b-c).$$

Note that the probability $G_{a,b}(c) = 0$ unless $c \in [|a-b|, \min(a+b, n)]$.

Recursive Distribution for w Vectors. Equipped with $G_{a,b}(c)$ from above, we can now derive a recursive formula for $p_{w,u}$, i.e., the probability that the sum of w random vectors of weight k and length n has Hamming weight u:

Base Case $w = 1$:

$$p_{1,u} = \begin{cases} 1 & \text{if } u = k \\ 0 & \text{if } u \neq k \end{cases}$$

For the recurrence, we consider the following two cases:

Even w: A list of w weight-k vectors sums to a vector of weight u when the first $w/2$ vectors sum to a vector of weight $a \in [0, n]$, the last $w/2$ vectors sum to a vector of weight $b \in [0, n]$, and the two resulting vectors (of weight a and b, respectively) sum to a vector of weight u. That is,

$$p_{w,u} = \sum_{a=0}^{n} \sum_{b=0}^{n} p_{w/2,a} \cdot p_{w/2,b} \cdot G_{a,b}(u).$$

Odd w: In this case, we instead sum a single weight-k vector with the sum of the remaining $w-1$ vectors, and compute the probability that the latter sum to a vector of weight $a \in [0, n]$ and the final sum has weight u. That is,

$$p_{w,u} = \sum_{a=0}^{n} p_{w-1,a} \cdot G_{a,k}(u).$$

[18] While for $\mathbb{F}_2$ cancellation always occurs, we need to consider both cases for general fields $\mathbb{F}_p$.

Usefulness. Using the formulae in this section, we can compute lower bounds on the dual distance of $\mathbf{A}$ for general fields $\mathbb{F}_p$. Note that the complexity of the computation is at least cubic in the LPN dimension n and quickly becomes infeasible for the large dimensions that we are interested in.

Acknowledgements. This research was supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement number 101115790 (OBELiSC), the French Agence Nationale de la Recherche (ANR) under the France 2030 ANR Project ANR-22-PECY-003 SecureCompute, and the Danish Independent Research Council under Grant-ID DFF-2064-00016B (YOSO). We thank the Aarhus Crypto group for computational resources and Sebastian Hasler for pointing out a mistake in the asymptotic analysis.

References

[ABW10] Benny Applebaum, Boaz Barak, and Avi Wigderson. *Public-key cryptography from different assumptions*. In: *42nd Annual ACM Symposium on Theory of Computing*. June 2010. DOI: 10.1145/1806689.1806714.

[ADINZ17] Benny Applebaum, Ivan Damgård, Yuval Ishai, Michael Nielsen, and Lior Zichron. *Secure Arithmetic Computation with Constant Computational Overhead*. In: *Advances in Cryptology – CRYPTO 2017, Part I*. Aug. 2017. DOI: 10.1007/978-3-319-63688-7_8.

[AFS03] Daniel Augot, Matthieu Finiasz, and Nicolas Sendrier. *A Fast Provably Secure Cryptographic Hash Function*. Cryptology ePrint Archive, Report 2003/230. 2003. URL: https://eprint.iacr.org/2003/230.

[AHIKV17] Benny Applebaum, Naama Haramaty, Yuval Ishai, Eyal Kushilevitz, and Vinod Vaikuntanathan. *Low-Complexity Cryptographic Hash Functions*. In: *ITCS 2017: 8th Innovations in Theoretical Computer Science Conference*. Jan. 2017. DOI: 10.4230/LIPIcs.ITCS.2017.7.

[AIK06] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. *On Pseudorandom Generators with Linear Stretch in NC^0*. In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 9th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2006 and 10th International Workshop on Randomization and Computation, RANDOM 2006, Barcelona, Spain, August 28-30 2006, Proceedings*. 2006. DOI: 10.1007/11830924_25.

[Ale03] Michael Alekhnovich. *More on Average Case vs Approximation Complexity*. In: *44th Annual Symposium on Foundations of Computer Science*. Oct. 2003. DOI: 10.1109/SFCS.2003.1238204.

[ANOSS22] Damiano Abram, Ariel Nof, Claudio Orlandi, Peter Scholl, and Omer Shlomovits. *Low-Bandwidth Threshold ECDSA via Pseudorandom Correlation Generators*. In: *2022 IEEE Symposium on Security and Privacy*. May 2022. DOI: 10.1109/SP46214.2022.9833559.

[AR16] Benny Applebaum and Pavel Raykov. *Fast Pseudorandom Functions Based on Expander Graphs*. In: *TCC 2016-B: 14th Theory of Cryptography Conference, Part I*. Oct. 2016. DOI: 10.1007/978-3-662-53641-4_2.

[BBMS22] Carsten Baum, Lennart Braun, Alexander Munch-Hansen, and Peter Scholl. *Moz$\mathbb{Z}_{2^k}$arella: Efficient Vector-OLE and Zero-Knowledge Proofs over $\mathbb{Z}_{2^k}$*. In: *Advances in Cryptology – CRYPTO 2022, Part IV*. Aug. 2022. DOI: 10.1007/978-3-031-15985-5_12.

[BC23] Dung Bui and Geoffroy Couteau. *Improved Private Set Intersection for Sets with Small Entries*. In: *PKC 2023: 26th International Conference on Theory and Practice of Public Key Cryptography, Part II*. May 2023. DOI: 10.1007/978-3-031-31371-4_7.

[BCCGJ24] Dung Bui, Eliana Carozza, Geoffroy Couteau, Dahmun Goudarzi, and Antoine Joux. *Faster Signatures from MPC-in-the-Head*. In: *Advances in Cryptology – ASIACRYPT 2024, Part I*. Dec. 2024. DOI: 10.1007/978-981-96-0875-1_13.

[BCEKM24] Chris Brzuska, Geoffroy Couteau, Christoph Egger, Pihla Karanko, and Pierre Meyer. *Instantiating the Hash-Then-Evaluate Paradigm: Strengthening PRFs, PCFs, and OPRFs*. In: *SCN 24: 14th International Conference on Security in Communication Networks, Part II*. Sept. 2024. DOI: 10.1007/978-3-031-71073-5_5.

[BCGI18] Elette Boyle, Geoffroy Couteau, Niv Gilboa, and Yuval Ishai. *Compressing Vector OLE*. In: *ACM CCS 2018: 25th Conference on Computer and Communications Security*. Oct. 2018. DOI: 10.1145/3243734.3243868.

[BCM25] Dung Bui, Geoffroy Couteau, and Nikolas Melissaris. *Structured-Seed Local Pseudorandom Generators and their Applications*. In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2025, August 11-13, 2024, UC Berkeley, California, USA*. Aug. 2025.

[BCMPR24] Dung Bui, Geoffroy Couteau, Pierre Meyer, Alain Passelègue, and Mahshid Riahinia. *Fast Public-Key Silent OT and More from Constrained Naor-Reingold*. In: *Advances in Cryptology – EUROCRYPT 2024, Part VI*. May 2024. DOI: 10.1007/978-3-031-58751-1_4.

[BCMRS25] Lennart Braun, Geoffroy Couteau, Kelsey Melissaris, Mahshid Riahinia, and Elahe Sadeghi. *Fast Pseudorandom Correlation Functions from Sparse LPN*. In: Dec. 2025.

[Bea92] Donald Beaver. *Efficient Multiparty Protocols Using Circuit Randomization*. In: *Advances in Cryptology – CRYPTO'91*. Aug. 1992. DOI: 10.1007/3-540-46766-1_34.

[Bea97] Donald Beaver. *Commodity-Based Cryptography (Extended Abstract)*. In: *29th Annual ACM Symposium on Theory of Computing*. May 1997. DOI: 10.1145/258533.258637.

[BGHIN25] Elette Boyle, Niv Gilboa, Matan Hamilis, Yuval Ishai, and Ariel Nof. *Preprocessing for Life: Dishonest-Majority MPC with a Trusted or Untrusted Dealer*. In: *2025 IEEE Symposium on Security and Privacy*. May 2025.

[BGI15] Elette Boyle, Niv Gilboa, and Yuval Ishai. *Function Secret Sharing*. In: *Advances in Cryptology – EUROCRYPT 2015, Part II*. Apr. 2015. DOI: 10.1007/978-3-662-46803-6_12.

[BGI16] Elette Boyle, Niv Gilboa, and Yuval Ishai. *Function Secret Sharing: Improvements and Extensions*. In: *ACM CCS 2016: 23rd Conference on Computer and Communications Security*. Oct. 2016. DOI: 10.1145/2976749.2978429.

[BIPSW18] Dan Boneh, Yuval Ishai, Alain Passelègue, Amit Sahai, and David J. Wu. *Exploring Crypto Dark Matter: New Simple PRF Candidates and Their Applications*. In: *TCC 2018: 16th Theory of Cryptography Conference, Part II*. Nov. 2018. DOI: 10.1007/978-3-030-03810-6_25.

[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: *Advances in Cryptology – EUROCRYPT 2012*. Apr. 2012. DOI: 10.1007/978-3-642-29011-4_31.

[BLPS11] Daniel J. Bernstein, Tanja Lange, Christiane Peters, and Peter Schwabe. *Really Fast Syndrome-Based Hashing*. In: *AFRICACRYPT 11: 4th International Conference on Cryptology in Africa*. July 2011. DOI: 10.1007/978-3-642-21969-6_9.

[BM18] Leif Both and Alexander May. *Decoding Linear Codes with High Error Rate and Its Impact for LPN Security*. In: *Post-Quantum Cryptography - 9th International Conference, PQCrypto 2018*. Apr. 2018. DOI: 10.1007/978-3-319-79063-3_2.

[BØ23] Pierre Briaud and Morten Øygarden. *A New Algebraic Approach to the Regular Syndrome Decoding Problem and Implications for PCG Constructions*. In: *Advances in Cryptology – EUROCRYPT 2023, Part V*. Apr. 2023. DOI: 10.1007/978-3-031-30589-4_14.

[Boy+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: *ACM CCS 2019: 26th Conference on Computer and Communications Security*. Nov. 2019. DOI: 10.1145/3319535.3354255.

[Boy+19b] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. *Efficient Pseudorandom Correlation Generators: Silent OT Extension and More*. In: *Advances in Cryptology – CRYPTO 2019, Part III*. Aug. 2019. DOI: 10.1007/978-3-030-26954-8_16.

[Boy+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*. Nov. 2020. DOI: 10.1109/FOCS46700.2020.00103.

[Boy+20b] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. *Correlated Pseudorandom Functions from Variable-Density LPN*. Cryptology ePrint Archive, Report 2020/1417. 2020. URL: https://eprint.iacr.org/2020/1417.

[Boy+20c] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. *Efficient Pseudorandom Correlation Generators from Ring-LPN*. In: *Advances in Cryptology – CRYPTO 2020, Part II*. Aug. 2020. DOI: 10.1007/978-3-030-56880-1_14.

[Boy+22] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Nicolas Resch, and Peter Scholl. *Correlated Pseudorandomness from Expand-Accumulate Codes*. In: *Advances in Cryptology – CRYPTO 2022, Part II*. Aug. 2022. DOI: 10.1007/978-3-031-15979-4_21.

[Boy+23] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Nicolas Resch, and Peter Scholl. *Oblivious Transfer with Constant Computational Overhead*. In: *Advances in Cryptology – EUROCRYPT 2023, Part I*. Apr. 2023. DOI: 10.1007/978-3-031-30545-0_10.

[BSV19] Andrej Bogdanov, Manuel Sabin, and Prashant Nalini Vasudevan. *XOR Codes and Sparse Learning Parity with Noise*. In: *30th Annual ACM-SIAM Symposium on Discrete Algorithms*. Jan. 2019. DOI: 10.1137/1.9781611975482.61.

[CCJ23] Eliana Carozza, Geoffroy Couteau, and Antoine Joux. *Short Signatures from Regular Syndrome Decoding in the Head*. In: *Advances in Cryptology – EUROCRYPT 2023, Part V*. Apr. 2023. DOI: 10.1007/978-3-031-30589-4_19.

[CCKK22] Jung Hee Cheon, Wonhee Cho, Jeong Han Kim, and Jiseung Kim. *Adventures in crypto dark matter: attacks, fixes and analysis for weak pseudorandom functions*. In: *Designs, Codes and Cryptography* 8 (2022). DOI: 10.1007/s10623-022-01071-x.

[CD23] Geoffroy Couteau and Clément Ducros. *Pseudorandom Correlation Functions from Variable-Density LPN, Revisited*. In: *PKC 2023: 26th International Conference on Theory and Practice of Public Key Cryptography, Part II*. May 2023. DOI: 10.1007/978-3-031-31371-4_8.

[CDDKS24] Geoffroy Couteau, Lalita Devadas, Srinivas Devadas, Alexander Koch, and Sacha Servan-Schreiber. *QuietOT: Lightweight Oblivious Transfer with a Public-Key Setup*. In: *Advances in Cryptology – ASIACRYPT 2024, Part II*. Dec. 2024. DOI: 10.1007/978-981-96-0888-1_7.

[CHKV25] Henry Corrigan-Gibbs, Alexandra Henzinger, Yael Tauman Kalai, and Vinod Vaikuntanathan. *Somewhat Homomorphic Encryption from Linear Homomorphism and Sparse LPN*. In: *Advances in Cryptology – EUROCRYPT 2025, Part II*. May 2025. DOI: 10.1007/978-3-031-91124-8_1.

[CRR21] Geoffroy Couteau, Peter Rindal, and Srinivasan Raghuraman. *Silver: Silent VOLE and Oblivious Transfer from Hardness of Decoding Structured LDPC Codes*. In: *Advances in Cryptology – CRYPTO 2021, Part III*. Aug. 2021. DOI: 10.1007/978-3-030-84252-9_17.

[Cui+24] Hongrui Cui, Hanlin Liu, Di Yan, Kang Yang, Yu Yu, and Kaiyi Zhang. *ReSolveD: Shorter Signatures from Regular Syndrome Decoding and VOLE-in-the-Head*. In: *PKC 2024: 27th International Conference on Theory and Practice of Public Key Cryptography, Part I*. Apr. 2024. DOI: 10.1007/978-3-031-57718-5_8.

[DHKS25] Henri Dohmen, Robin Hundt, Nora Khayata, and Thomas Schneider. *SEEC: Memory Safety Meets Efficiency in Secure Two-Party Computation*. In: *ASIACCS 25: 20th ACM Symposium on Information, Computer and Communications Security*. June 2025. DOI: 10.1145/3708821.3736224.

[DHRW16] Yevgeniy Dodis, Shai Halevi, Ron D. Rothblum, and Daniel Wichs. *Spooky Encryption and Its Applications*. In: *Advances in Cryptology – CRYPTO 2016, Part III*. Aug. 2016. DOI: 10.1007/978-3-662-53015-3_4.

[DIJL23] Quang Dao, Yuval Ishai, Aayush Jain, and Huijia Lin. *Multi-party Homomorphic Secret Sharing and Sublinear MPC from Sparse LPN*. In: *Advances in Cryptology – CRYPTO 2023, Part II*. Aug. 2023. DOI: 10.1007/978-3-031-38545-2_11.

[DIO21] Samuel Dittmer, Yuval Ishai, and Rafail Ostrovsky. *Line-Point Zero Knowledge and Its Applications*. In: *ITC 2021: 2nd Conference on Information-Theoretic Cryptography*. July 2021. DOI: 10.4230/LIPIcs.ITC.2021.5.

[DJ24] Quang Dao and Aayush Jain. *Lossy Cryptography from Code-Based Assumptions*. In: *Advances in Cryptology – CRYPTO 2024, Part III*. Aug. 2024. DOI: 10.1007/978-3-031-68382-4_2.

[DPSZ12] Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. *Multiparty Computation from Somewhat Homomorphic Encryption*. In: *Advances in Cryptology – CRYPTO 2012*. Aug. 2012. DOI: 10.1007/978-3-642-32009-5_38.

[ES24] Andre Esser and Paolo Santini. *Not Just Regular Decoding: Asymptotics and Improvements of Regular Syndrome Decoding Attacks*. In: *Advances in Cryptology – CRYPTO 2024, Part VI*. Aug. 2024. DOI: 10.1007/978-3-031-68391-6_6.

[Eth] Ethereum Foundation. *Sign. MPC-TLS*. https://docs.sign.global/case-study/mpc-tls.

[FGS07] Matthieu Finiasz, Philippe Gaborit, and Nicolas Sendrier. *Improved fast syndrome based cryptographic hash functions*. In: *Proceedings of ECRYPT Hash Workshop*. 2007.

[FHKRS25] Sebastian Faust, Carmit Hazay, David Kretzler, Leandro Rometsch, and Benjamin Schlosser. *Non-interactive threshold BBS+ from pseudorandom correlations*. In: *Cryptographers' Track at the RSA Conference*. 2025.

[FKO06] Uriel Feige, Jeong Han Kim, and Eran Ofek. *Witnesses for non-satisfiability of dense random 3CNF formulas*. In: *47th Annual Symposium on Foundations of Computer Science*. Oct. 2006. DOI: 10.1109/FOCS.2006.78.

[FKOS15] Tore Kasper Frederiksen, Marcel Keller, Emmanuela Orsini, and Peter Scholl. *A Unified Approach to MPC with Preprocessing Using OT*. In: *Advances in Cryptology – ASIACRYPT 2015, Part I*. Nov. 2015. DOI: 10.1007/978-3-662-48797-6_29.

[FLLL24] Ximing Fu, Mo Li, Shihan Lyu, and Chuanyi Liu. *Bit-fixing Correlation Attacks on Goldreich's Pseudorandom Generators*. Cryptology ePrint Archive, Report 2024/1594. 2024. URL: https://eprint.iacr.org/2024/1594.

[Gal19] Galois, Inc. *swanky: A suite of rust libraries for secure computation*. https://github.com/GaloisInc/swanky. 2019.

[GI14] Niv Gilboa and Yuval Ishai. *Distributed Point Functions and Their Applications*. In: *Advances in Cryptology – EUROCRYPT 2014*. May 2014. DOI: 10.1007/978-3-642-55220-5_35.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. *How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority*. In: *19th Annual ACM Symposium on Theory of Computing*. May 1987. DOI: 10.1145/28395.28420.

[Gol00] Oded Goldreich. *Candidate One-Way Functions Based on Expander Graphs*. Cryptology ePrint Archive, Report 2000/063. 2000. URL: https://eprint.iacr.org/2000/063.

[HLHD22] Zhicong Huang, Wen-jie Lu, Cheng Hong, and Jiansheng Ding. *Cheetah: Lean and Fast Secure Two-Party Deep Neural Network Inference*. In: *USENIX Security 2022: 31st USENIX Security Symposium*. Aug. 2022. URL: https://www.usenix.org/conference/usenixsecurity22/presentation/huang-zhicong.

[HOSS18] Carmit Hazay, Emmanuela Orsini, Peter Scholl, and Eduardo Soria-Vazquez. *TinyKeys: A New Approach to Efficient Multi-Party Computation*. In: *Advances in Cryptology – CRYPTO 2018, Part III*. Aug. 2018. DOI: 10.1007/978-3-319-96878-0_1.

[IKNP03] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. *Extending Oblivious Transfers Efficiently*. In: *Advances in Cryptology – CRYPTO 2003*. Aug. 2003. DOI: 10.1007/978-3-540-45146-4_9.

[JMN23] Thomas Johansson, Willi Meier, and Vu Nguyen. *Differential cryptanalysis of Mod-2/Mod-3 constructions of binary weak PRFs*. In: *2023 IEEE International Symposium on Information Theory (ISIT)*. 2023.

[KOR23] Yashvanth Kondi, Claudio Orlandi, and Lawrence Roy. *Two-Round Stateless Deterministic Two-Party Schnorr Signatures from Pseudorandom Correlation Functions*. In: *Advances in Cryptology – CRYPTO 2023, Part I*. Aug. 2023. DOI: 10.1007/978-3-031-38557-5_21.

[KOS16] Marcel Keller, Emmanuela Orsini, and Peter Scholl. *MASCOT: Faster Malicious Arithmetic Secure Computation with Oblivious Transfer*. In: *ACM CCS 2016: 23rd Conference on Computer and Communications Security*. Oct. 2016. DOI: 10.1145/2976749.2978357.

[KPR18] Marcel Keller, Valerio Pastro, and Dragos Rotaru. *Overdrive: Making SPDZ Great Again*. In: *Advances in Cryptology – EUROCRYPT 2018, Part III*. Apr. 2018. DOI: 10.1007/978-3-319-78372-7_6.

[LWYY24] Hanlin Liu, Xiao Wang, Kang Yang, and Yu Yu. *The Hardness of LPN over Any Integer Ring and Field for PCG Applications*. In: *Advances in Cryptology – EUROCRYPT 2024, Part VI*. May 2024. DOI: 10.1007/978-3-031-58751-1_6.

[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*. 2005. ISBN: 978-3-540-31874-3.

[MDCY11] Mohammed Meziani, Özgür Dagdelen, Pierre-Louis Cayrel, and Sidi Mohamed El Yousfi Alaoui. *S-FSB: An improved variant of the FSB hash family*. In: *International Conference on Information Security and Assurance*. 2011.

[MO15] Alexander May and Ilya Ozerov. *On Computing Nearest Neighbors with Applications to Decoding of Binary Linear Codes*. In: *Advances in Cryptology – EUROCRYPT 2015, Part I*. Apr. 2015. DOI: 10.1007/978-3-662-46800-5_9.

[MST03] Elchanan Mossel, Amir Shpilka, and Luca Trevisan. *On e-Biased Generators in NC0*. In: *44th Annual Symposium on Foundations of Computer Science*. Oct. 2003. DOI: 10.1109/SFCS.2003.1238188.

[MSY21] Jean-Pierre Münch, Thomas Schneider, and Hossein Yalame. *VASA: Vector AES Instructions for Security Applications*. In: *ACSAC '21: Annual Computer Security Applications Conference, Virtual Event, USA, December 6 - 10, 2021*. 2021. DOI: 10.1145/3485832.3485897.

[NMSÜ25] Miguel Cueto Noval, Simon-Philipp Merz, Patrick Stählin, and Akin Ünal. *On the Soundness of Algebraic Attacks Against Code-Based Assumptions*. In: *Advances in Cryptology – EUROCRYPT 2025, Part VI*. May 2025. DOI: 10.1007/978-3-031-91095-1_14.

[NN90] Joseph Naor and Moni Naor. *Small-bias Probability Spaces: Efficient Constructions and Applications*. In: *22nd Annual ACM Symposium on Theory of Computing*. May 1990. DOI: 10.1145/100216.100244.

[OSY21] Claudio Orlandi, Peter Scholl, and Sophia Yakoubov. *The Rise of Paillier: Homomorphic Secret Sharing and Public-Key Silent OT*. In: *Advances in Cryptology – EUROCRYPT 2021, Part I*. Oct. 2021. DOI: 10.1007/978-3-030-77870-5_24.

[OTX24] Ying Ouyang, Deng Tang, and Yanhong Xu. *Code-Based Zero-Knowledge from VOLE-in-the-Head and Their Applications: Simpler, Faster, and Smaller*. In: *Advances in Cryptology – ASIACRYPT 2024, Part V*. Dec. 2024. DOI: 10.1007/978-981-96-0935-2_14.

[Pria] Primus. *What is Primus. zkTLS*. https://docs.primuslabs.xyz/data-verification/tech-intro.

[Prib] Privacy and Scaling Exploration. *TLSNotary*. https://docs.tlsnotary.org/.

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

[RR22] Srinivasan Raghuraman and Peter Rindal. *Blazing Fast PSI from Improved OKVS and Subfield VOLE*. In: *ACM CCS 2022: 29th Conference on Computer and Communications Security*. Nov. 2022. DOI: 10.1145/3548606.3560658.

[RRT23] Srinivasan Raghuraman, Peter Rindal, and Titouan Tanguy. *Expand-Convolute Codes for Pseudorandom Correlation Generators from LPN*. In: *Advances in Cryptology – CRYPTO 2023, Part IV*. Aug. 2023. DOI: 10.1007/978-3-031-38551-3_19.

[RS21] Peter Rindal and Phillipp Schoppmann. *VOLE-PSI: Fast OPRF and Circuit-PSI from Vector-OLE*. In: *Advances in Cryptology – EUROCRYPT 2021, Part II*. Oct. 2021. DOI: 10.1007/978-3-030-77886-6_31.

[RSWP23] Mayank Rathee, Conghao Shen, Sameer Wagh, and Raluca Ada Popa. *ELSA: Secure Aggregation for Federated Learning with Malicious Actors*. In: *2023 IEEE Symposium on Security and Privacy*. May 2023. DOI: 10.1109/SP46215.2023.10179468.

[RVV24] Seyoon Ragavan, Neekon Vafa, and Vinod Vaikuntanathan. *Indistinguishability Obfuscation from Bilinear Maps and LPN Variants*. In: *TCC 2024: 22nd Theory of Cryptography Conference, Part IV*. Dec. 2024. DOI: 10.1007/978-3-031-78023-3_1.

[Sch08] Grant Schoenebeck. *Linear Level Lasserre Lower Bounds for Certain k-CSPs*. In: *49th Annual Symposium on Foundations of Computer Science*. Oct. 2008. DOI: 10.1109/FOCS.2008.74.

[SGRR19] Phillipp Schoppmann, Adrià Gascón, Leonie Reichert, and Mariana Raykova. *Distributed Vector-OLE: Improved Constructions and Implementation*. In: *ACM CCS 2019: 26th*

Conference on Computer and Communications Security. Nov. 2019. DOI: 10.1145/3319535.3363228.

[WMK16] Xiao Wang, Alex J. Malozemoff, and Jonathan Katz. *EMP-toolkit: Efficient MultiParty computation toolkit*. https://github.com/emp-toolkit. 2016.

[WYKW21] Chenkai Weng, Kang Yang, Jonathan Katz, and Xiao Wang. *Wolverine: Fast, Scalable, and Communication-Efficient Zero-Knowledge Proofs for Boolean and Arithmetic Circuits*. In: *2021 IEEE Symposium on Security and Privacy*. May 2021. DOI: 10.1109/SP40001.2021.00056.

[YSWW21] Kang Yang, Pratik Sarkar, Chenkai Weng, and Xiao Wang. *QuickSilver: Efficient and Affordable Zero-Knowledge Proofs for Circuits and Polynomials over Any Field*. In: *ACM CCS 2021: 28th Conference on Computer and Communications Security*. Nov. 2021. DOI: 10.1145/3460120.3484556.

[YWLZW20] Kang Yang, Chenkai Weng, Xiao Lan, Jiang Zhang, and Xiao Wang. *Ferret: Fast Extension for Correlated OT with Small Communication*. In: *ACM CCS 2020: 27th Conference on Computer and Communications Security*. Nov. 2020. DOI: 10.1145/3372297.3417276.

[zkP23] zkPass Team. *zkPass Protocol based on TLS, MPC and ZKP. Technical Whitepaper 2.0*. Tech. rep. 2023.