Compressing Unit-Vector Correlations via Sparse Pseudorandom Generators

Amit Agarwal[1(✉)], Elette Boyle[2,3], Niv Gilboa[4], Yuval Ishai[5], Mahimna Kelkar[6], and Yiping Ma[7]

[1] UIUC, Champaign, USA
amita2@illinois.edu
[2] Reichman University, Herzliya, Israel
[3] NTT Research, Sunnyvale, USA
[4] Ben-Gurion University, Beersheba, Israel
[5] Technion, Haifa, Israel
[6] Cornell University, Ithaca, USA
[7] University of Pennsylvania, Philadelphia, USA

Abstract. A *unit-vector (UV) correlation* is an additive secret-sharing of a vector of length B that contains 1 in a secret random position and 0's elsewhere. UV correlations are a useful resource for many cryptographic applications, including low-communication secure multiparty computation and multi-server private information retrieval. However, current practical methods for securely generating UV correlations involve a significant communication cost *per instance*, and become even more expensive when requiring security against malicious parties.

In this work, we present a new approach for constructing a *pseudorandom correlation generator* (PCG) for securely generating n independent instances of UV correlations of any polynomial length B. Such a PCG compresses the n UV instances into correlated seeds whose length is sublinear in the description size $n \cdot \log B$. Our new PCGs apply in both the honest-majority and dishonest-majority settings, and are based on a variety of assumptions. In particular, in the honest-majority case they only require "unstructured" assumptions. Our PCGs give rise to secure end-to-end protocols for generating n instances of UV correlations with $o(n)$ bits of communication. This applies even to an authenticated variant of UV correlations, which is useful for security against malicious parties. Unlike previous theoretical solutions, some instances of our PCGs offer good concrete efficiency.

Our technical approach is based on combining a low-degree *sparse pseudorandom generator*, mapping a sparse seed to a pseudorandom sparse output, with homomorphic secret sharing for low-degree polynomials. We then reduce such sparse PRGs to *local* PRGs over large alphabets, and explore old and new approaches for maximizing the stretch of such PRGs while minimizing their locality.

Finally, towards further compressing the PCG seeds, we present a new PRG-based construction of a multiparty distributed point function (DPF), whose outputs are degree-1 Shamir-shares of a secret point function. This result is independently motivated by other DPF applications.

L. Reyzin and D. Stebila (Eds.): CRYPTO 2024, LNCS 14927, pp. 346–383, 2024.
https://doi.org/10.1007/978-3-031-68397-8_11

1 Introduction

Minimizing the *communication cost* of secure multiparty computation (MPC) is a central challenge in theoretical and applied cryptography. A general approach for reducing this cost is by applying powerful cryptographic tools such as fully homomorphic encryption (FHE) [53], enabling computations on encrypted data, or homomorphic secret sharing (HSS) [25], enabling computations on secret-shared data. However, despite significant optimization efforts, the constructions of these primitives still rely on specific, "structured" cryptographic assumptions and involve a high concrete computational overhead.

A common alternative approach for improving the communication cost of practical secure computation protocols is to make use of *correlated randomness*. If correlated secret randomness is distributed to the parties during an input-independent offline phase, this can be used to dramatically reduce the costs of an input-dependent online protocol. This offline-online approach serves as a basis, e.g., for the popular SPDZ line of protocols [10,43]. However, a major challenge is to design methods for actually generating such useful correlations securely with low communication and good concrete efficiency.

This challenge is addressed by a recent line of work on *pseudorandom correlation generators* (PCGs) [15,19]. A PCG can be viewed as a natural distributed analogue of a standard cryptographic pseudorandom generator (PRG). While a PRG maps a short random seed to a long output that looks random to a computationally bounded external observer, a PCG maps two or more short *correlated* seeds to long correlated outputs which look like they were sampled from a given target correlation *even from the point of view of insiders*.

Many recent works [1,12,13,15,16,18–20,31,37,54,72,75,79] build highly efficient PCGs that avoid the use of general-purpose FHE or HSS, and instead rely mostly on different flavors of the learning parity with noise (LPN) assumption. However, these efficient PCGs are limited to simple target correlations (such as Oblivious Transfer correlations) that, in particular, cannot help bring the online communication cost below the circuit size of the function being computed. Alternative techniques [22,23,25,39,44] that do enable this, on the other hand, incur a high computational overhead, making them unattractive from a concrete efficiency viewpoint.

PCGs for Unit-Vector Correlations? One particularly useful correlation is a *unit-vector* (UV) *correlation*: an additive secret-sharing of a vector of length B over a given group that contains 1 in a secret random position and 0's elsewhere. UV correlations are an advantageous resource for many cryptographic applications. They can be directly used for reducing the communication cost of multi-server private information retrieval [34]. They can also be directly and *locally* converted, without communication, into *lookup-table* (LUT) correlations [19,60,64] (sometimes referred to as *truth-table* correlations). A LUT correlation for a function $g : \{0,1\}^b \to \{0,1\}$, with table size $B = 2^b$, reduces the online cost of securely evaluating each "gate" computing g to roughly b bits of communication. (This can be generalized to gates with longer outputs.) LUT

correlations for small values of b are very useful both in theory, for evaluating natural classes of circuits with (slightly) sublinear communication [36,42], and in practice, for obtaining significant concrete improvements in many natural applications [30,42,64,71]. This makes PCGs for UV correlations, as well as their *authenticated* variants, a highly desirable goal.

Progress in this direction was made in [19], where a concretely efficient PCG for a *single* long instance of UV or authenticated UV correlation was constructed based on a distributed point function (DPF) [26,55], which in turn can be based on any PRG. However, while this enables compressing a length-B correlation to small size $o(B)$, this does not extend to generating n independent instances of UV correlations with $o(n)$ seed size.

For smaller values of B, concretely efficient approaches for generating UV and LUT correlations via *information-theoretic* reductions to oblivious transfer were given in [30,71]. However, here the communication cost of generating n instances scales linearly not only with n but also with B. This was recently shown to be inherent for information-theoretic reductions [58].

For both the DPF-based approach and the information-theoretic approaches, generating UV correlations with security against *malicious* parties is considerably more expensive than in the semi-honest case. Even if given access to a trusted dealer, these previous approaches cannot reduce the *storage size* of n UV correlation instances below the description length $n \log B$. A concretely efficient PCG for compressing n instances of a UV correlation is thus a highly desirable goal.

1.1 Our Contributions

We present a new framework toward lightweight PCGs for n *independent instances* of unit-vector (UV) and related correlations, based on a new notion of *sparse PRGs*. We then study the underlying tools, instantiating our framework in several ways. For example, with a strong honest majority, we can obtain such PCGs from just *standard local binary PRGs* [3,56]. We present other constructions based on different combinations of (old and new) "unstructured" and "structured" assumptions. Several of these combinations are relevant to low-communication concretely efficient MPC protocols, and beyond.

We now give a more detailed account of our results, starting with the main new technical tool we introduce: a sparse PRG. In a nutshell, a sparse PRG maps a (possibly sparse) random seed to a sparse pseudorandom output. A bit more formally:

Definition 1 (Sparse PRG, informal). Let $\mathcal{R}$ be a finite ring. A (regular) B-sparse PRG over $\mathcal{R}$ with seed length k and output length n is an efficient deterministic mapping $G : \mathcal{R}^k \to \mathcal{R}^{nB}$ along with a PPT algorithm for sampling a seed x, such that $G(x)$ is indistinguishable from a concatenation of n random length-B unit vectors.

While a sparse PRG as above can be trivially constructed from any standard PRG, we will be interested in minimizing the *algebraic degree* of G, which

we will denote by d. This is motivated by the observation that PCG for UV correlations can be naturally obtained by applying any *Homomorphic Secret Sharing* (HSS) [25] scheme for degree-d polynomials to evaluate G. HSS is a form of secret sharing that enables homomorphic evaluation on shares, resulting in additive shares of the computation output. The feasibility and efficiency of such HSS schemes greatly depends on the degree of the function being homomorphically evaluated. We summarize this general approach below, and then discuss useful instantiations.

Theorem 1 (Sparse PRG + HSS ⇒ PCG for UV, informal). *Given a B-sparse PRG $G : \mathcal{R}^k \to \mathcal{R}^{nB}$ of algebraic degree d and an N-party, t-secure HSS for degree-d polynomials over $\mathcal{R}$, there is an N-party, t-secure PCG generating n length-B UV correlations, with the PCG seed size scaling linearly with k.*

From Non-binary Local PRG to Sparse PRG. The above blueprint motivates sparse PRG constructions with a low degree d and a high stretch. Our key observation is that this can be reduced to the construction of a (standard) *d-local* PRG over large alphabets, namely a PRG in which each output symbol is taken from an alphabet of size B and depends only on d input symbols. The reduction uses a sparse representation of each input symbol and output symbol of the local PRG by their characteristic vectors. Given this sparse representation, the degree of the sparse PRG coincides with the locality of the underlying local PRG.

Theorem 2 (Non-binary local PRG ⇒ sparse PRG, informal). *Given a d-local (dense) PRG $G : [B_{\mathsf{in}}]^k \to [B]^n$, there is a degree-$d$ B-sparse PRG $G' : \mathcal{R}^{kB_{\mathsf{in}}} \to \mathcal{R}^{nB}$ over any ring $\mathcal{R}$.*

We note that Lin and Tessaro [66], in the context of basing indistinguishability obfuscation on standard assumptions, used local PRGs over big input alphabets to minimize the degree of PRGs with a *dense* pseudorandom output. Motivated by very different applications, we take this approach further by using local PRGs over large *output* alphabets, and moreover are less severely affected by the barriers encountered in [66] (see the full version for discussion).

While the construction of sparse PRGs from local PRGs is very natural, we believe there are reasons to be optimistic about direct constructions of low-degree sparse PRGs that beat our current constructions based on local PRGs. We thus view sparse PRGs (as opposed to local PRGs) as the "right" abstraction, as well as a cryptographic notion of independent interest.

Sparse HSS via Multiplicative DPF. Pushing our framework even further, we observe that when using the sparse PRGs obtained via Theorem 2, we do not even require full HSS in the standard sense; rather, it would suffice to have HSS that works for *sparse inputs*. For the case of N-party 1-secure HSS for polynomials of degree $d = N - 1$, we obtain a particularly efficient construction of such sparse HSS schemes based on any standard PRG. Our construction is based on a *multiplicative* variant of a distributed point function (DPF) [26,55],

or *mult-DPF* for short. A standard DPF enables a compressed linear secret-sharing of a unit vector of length 2^ℓ (equivalently, a point function with ℓ-bit input), using $O(\lambda\ell)$-bit keys. Such DPF constructions are typically limited to the 2-party setting, where the output shares are additive and cannot be multiplied. Here we need a *d-mult-DPF* that supports a non-interactive evaluation of degree-d polynomials on the output shares.

A construction of a 1-secure, 3-party 2-mult-DPF was recently given in [32]. Other than being restricted to $d = 2$, the key size in this construction scales quadratically with ℓ. Here we present a different construction that improves the dependence on ℓ to linear and, more importantly for our purposes, applies to an arbitrary degree d.

Theorem 3 (N-party $(N-1)$-mult-DPF, informal). *For any number of parties $N \geq 3$, input length ℓ and output ring $\mathcal{R}$, there is 1-secure N-party $(N-1)$-multiplicative DPF for point functions $f_{\alpha,\beta} : \{0,1\}^\ell \to \mathcal{R}$ with per-party key size $O(N(\lambda\ell + \log|\mathcal{R}|))$, using any PRG with seed size λ. When $\mathcal{R}$ is a finite field and $|\mathcal{R}| > N$, the output can be shared using degree-1 Shamir's scheme. Otherwise it is shared using CNF (aka replicated) secret sharing.*

For $N > 3$ parties, the best prior approach was a brute-force emulation of replicated secret sharing via "tensoring" d instances of a 2-party DPF [26]. This requires $N = 2^d$ parties. In contrast, we only need $N = d + 1$ parties.

Given this tool, we can achieve degree-d HSS for *sparse* input vectors x (for some weight w) as the sum of w mult-DPFs, each sharing a single nonzero position value of x. For a degree-d computation f on the symbols of x, the parties can locally generate additive shares of $f(x)$ by expanding each the w mult-DPFs, adding the w shares for each position (yielding a multiplicative sharing of x), and then applying the degree-d multiplicativity procedure on the resulting shares.

Applying the new mult-DPF to our PCG constructions, consider beginning with a d-local PRG with stretch $k \to n$ over alphabet size B. Our mult-DPF enables us to attain $(d+1)$-party PCGs for UV correlations against one corruption, where the number of UV correlations produced is equal to the local PRG output size n, and whose seed sizes scale linearly with the local PRG seed size k and just *logarithmically* with the unit vector length B.

Our new mult-DPF is motivated by many other applications outside the realm of PCGs, such as PIR with conjunctive queries [14] (e.g., searching documents containing d specific keywords) and more.

Non-binary Local PRGs. The asymptotic and concrete efficiency of PCGs based on the above blueprint critically depends on the parameters of the underlying local PRGs. We explore candidate constructions of d-local PRGs with superlinear stretch and a polynomially bounded output alphabet size B. This includes both a simple construction from binary d-local PRGs, whose existence for $d \geq 5$ (with polynomial stretch that grows with d) is by now considered a standard cryptographic assumption [5,56,63], as well as several direct constructions that are conjectured to achieve better locality or stretch. In the latter category we consider the following candidates:

- For $d \geq 3$, an elegant d-local candidate of Barak et al. [7], in which each output is obtained by multiplying, over a simple non-Abelian group (such as the alternating group), d seed elements. The conjectured stretch is $n \approx k^{d/2}$ where k is the seed length and n is the output length. This limitation on the stretch is conjectured to be inherent to all d-local PRGs over constant-size alphabet [7]. While some instances of this construction were shown to be insecure in [46], most instances are not susceptible to this attack.
- For $d \geq 3$, a new d-local candidate that can be viewed as a *sparse* variant of the Learning with Rounding (LWR) assumption [6]. Here we have $B_{\mathsf{in}} = c \cdot B$ for some information loss parameter c ($c = 2$ by default). Each output is obtained by first applying (public) random permutations $\pi_i : \mathbb{Z}_{B_{\mathsf{in}}} \to \mathbb{Z}_{B_{\mathsf{in}}}$ to each of d random seed elements, then adding the d results over $\mathbb{Z}_{B_{\mathsf{in}}}$, and finally applying a lossy c-to-1 mapping $L : \mathbb{Z}_{B_{\mathsf{in}}} \to \mathbb{Z}_B$ mapping y to $\lfloor y/c \rfloor$. In sparse LWR, random permutations are replaced by scalar products. Other than its simplicity and the relation with a well-studied assumption, this candidate is motivated by the FFT-friendliness of the resulting sparse PRG. Here too, we conjecture $n \approx k^{d/2}$.
- A new 2-local candidate with a slightly superlinear stretch. This construction can be viewed as a natural extension of the "random local function" approach to non-binary alphabets. The limitation on the stretch follows from known limitations on the number of edges in a graph with no short cycles as well as negative results from [7,67].

Our work provides additional motivation for the study of local PRGs over large alphabets. We leave a more thorough analysis of asymptotic and concrete security to future work.

Putting Things Together: Our PCG Constructions. Through different combinations of candidate local PRG and HSS schemes, we obtain a collection of implied constructions for both the honest-majority and the 2-party setting. These are summarized at a high level in Fig. 1, and further detailed in Fig. 5 in Sect. 5.3.2. We present a selection of such corollaries for the case of a small polynomial stretch and security threshold $t = 1$.

Theorem 4 (PCGs for UV, informal). *There are N-party, 1-secure PCGs for generating $n \approx k^{1.5}$ instances of length-B UV, with the following options.*

- *$N = 6$: from Goldreich's 5-local PRG [3,56] $G : \{0,1\}^k \to \{0,1\}^n$, with PCG seed length $\approx k \log B$.*
- *$N = 4$: from the 3-local PRG candidate $G : [B]^k \to [B]^n$ of Barak et al. [7] or our generalized sparse LWR candidate $G : [2B]^k \to [B]^n$, with PCG seed length $\approx k \log B$.*
- *$N = 2$: from the previous 3-local candidates + HSS for degree-3 polynomials based on BGN [19,25], DCR [69,74], or LWE [28], with seed length $\approx kB$.*

In the case of the 2-party BGN-based construction, we also make use of correlation-robust hashing [59] (or alternatively a random oracle). To avoid the

high overhead and inverse-polynomial error of the distributed discrete-log procedure from [25] within the BGN-based scheme, we circumvent this procedure by settling for generating a *weight-1* correlation, which is the same as a UV correlation except that the nonzero entry is random. This correlation is useful for several applications, such as compressing many instances of a DPF with a small input domain and large output group. We then show how to securely convert this weight-1 correlation to a strict UV correlation using only (expected) 1 bit of communication per UV instance, much less than its description size.

An advantageous property of all our PCG constructions, inherited from the framework, is that evaluation is *incremental*. Namely, one need not expand the full PCG in one shot, but rather can perform incremental amounts of local expansion to obtain additional instances of the desired correlation.

Similarly to previous PCG constructions [15,19,20], most of our PCGs can be easily extended to generating "authenticated" variants of the target UV or LUT correlations by just doubling the (sublinear) seed size and the (linear) expansion time. Also, the PCG approach "amortizes away" the cost of malicious security (which reduces to the secure generation of a sublinear-size seed). Indeed, distributed seed generation itself can be made relatively cheap for many of our resulting PCGs. For example, for those relying on information-theoretic HSS (in the case of strong honest majority), our PCG seeds consist simply of linear secret shares of random unit vectors (corresponding to the regular sparse PRG seed), which can be generated by a circuit whose size is close to the seed length. This makes distributed seed generation via standard MPC techniques scale linearly with the seed length, and quite reasonable in practice even for the malicious case. Optimizing the concrete efficiency of distributed seed generation with mult-DPF based compressed seeds or the 2-party PCGs is left as another direction for future work.

Concrete Efficiency. To demonstrate potential practicality, we estimate the concrete efficiency of some of our PCG constructions, based on plausible conjectures about the concrete security of local PRG candidates. Our estimates are based in part on extrapolation from concrete efficiency estimates for binary 5-local PRGs [38,76–78] as well as concrete expansion parameters for random graphs [4,80]. Similarly to other new assumptions in cryptography, it will take a community effort to tune the concrete parameters and increase the level of confidence in the security. It is important to stress in this context that more conservative or aggressive parameter choices of the seed length and stretch of the local PRGs only have a proportional affect on the seed length and stretch of the derived PCG, but have almost no effect on the *throughput*, namely the number of UV instances that can be generated per second. For this reason, the potential practical value of our approach is not very sensitive to the exact choice of parameters, at least in the honest-majority setting. Indeed, one can compensate for new attacks by just increasing the seed size, without affecting the throughput. We leave a more thorough analysis of the new candidates to future work.

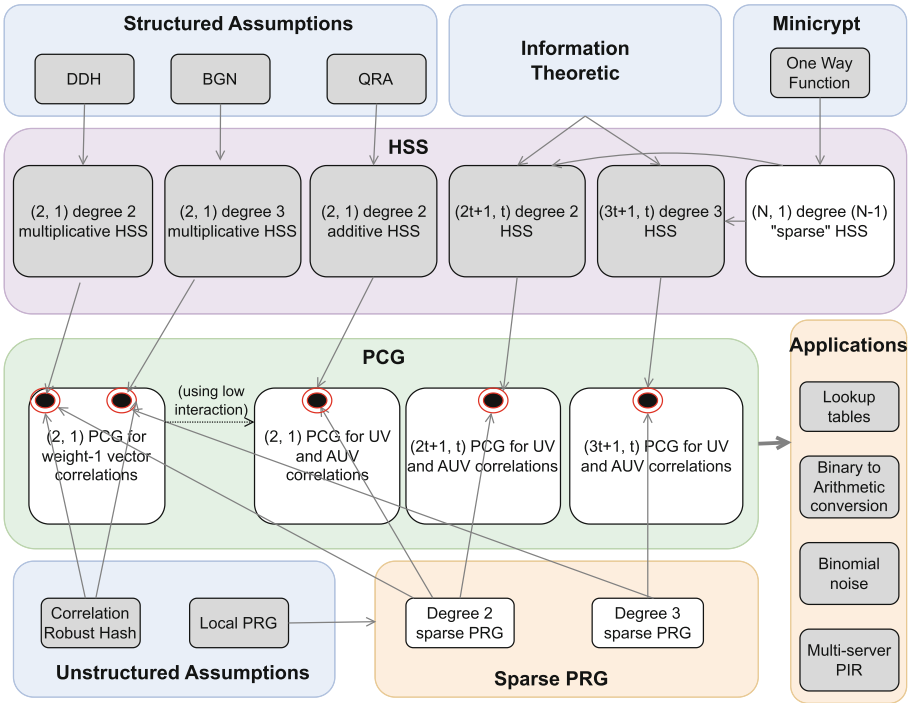

Fig. 1. Overview of our constructions. The grey boxes represent the primitives known in the literature whereas the white boxes are new to this work. "UV" and "AUV" denotes Unit-Vector and Authenticated Unit-Vector respectively. Each red circled bullet represents a unique construction obtained by combining the boxes whose arrows are incident on the bullet. (Color figure online)

To give some specific data points: In the case of 4-party (1-secure) PCGs, we consider the construction based on our new 3-local "Sparse LWR" candidate (Conjecture 3), which can leverage fast FFT algorithms with the additional boost of hardware support when the UV is shared over $\mathbb{Z}_2$. On a 2.6 GHz machine, we estimate a per-core throughput of 1.4×10^5 length-16 UVs over $\mathbb{Z}_2$ per second, with PCG seed size 300KB and compression ratio 79 (compared to the best practical alternative in this setting). Note that we do not need to batch the UV generation. When the UV length becomes larger, the throughput decreases; but we can still preserve a reasonable PCG seed size and a good compression ratio[1], especially for UVs over $\mathbb{Z}_p$ when p is large (due to our new mult-DPF construction): the throughput for length-2^{15} UVs over $\mathbb{Z}_p$ where $p = 2^{61} - 1$ (resp. $\mathbb{Z}_2$) is 8 UVs per second (resp. 24), with PCG seed size 82MB (resp. 79MB) and compression ratio 63 (resp. 3). See full version for more details.

[1] Intuitively, a compression ratio of s suggests that our PCG approach is $s\times$ better compared to the baseline approach for compressing a given number of unit vectors (more details are included in the full version).

In the 2-party case, using lattice-based packed HSS for low-degree polynomials, one can get a high throughput at the cost of long seeds (comparable to the lattice-based PCG estimates from [19]). For shorter seeds with a much lower throughput, one can use BGN-based HSS for degree-3 polynomials or DCR-based HSS for higher degree polynomials. For example, with DCR-based HSS, and with a packing technique using the Chinese Remainder Theorem, we can get an estimated single-core throughput 1 UV per second for UV length 8 with PCG seed size 102 MB and compression ratio 22. A higher throughput can be achieved at the cost of higher PCG seed size.

2 Overview of Techniques

We now give an overview of our techniques, including the general framework for constructing PCGs for UV correlations, our approach for building sparse PRGs, the multiplicative $(N, 1)$-DPF construction, and applications.

2.1 Framework: PCG for UV from Sparse PRG + HSS (Theorem 1)

The starting point for our new approach is the simple "PRG-plus-HSS" framework from [19,21] for obtaining PCGs for additive correlations. The construction is based on homomorphic secret sharing (HSS) [25], a form of secret sharing that supports homomorphic evaluation on individual shares, with additive reconstruction over some underlying Abelian group. The idea from these works is to provide parties with HSS-shares of a random PRG seed. Then these shares can be locally expanded to additive shares of any desired distribution $\mathcal{D}$ (e.g., of unit vectors), by homomorphically evaluating the function that first computes the PRG and then applies a sampling algorithm from $\mathcal{D}$ using the resulting randomness.

Despite its theoretical appeal, this unfortunately has not had much impact in terms of practical constructions, as HSS schemes with the required homomorphism capability are broadly just too expensive. Recall that the HSS must in particular support homomorphic evaluation of a PRG. Even using low-complexity PRG constructions (such as local PRGs), and with a target of generating simpler correlations, attempts at applying the brute-force HSS approach typically resulted in very high concrete costs (even given optimization efforts) [18,21].[2]

A unique exception is the LPN-based approach for PCGs initiated in [15,19], which can be cast in the above light by interpreting a piece of the construction as a (somewhat-unnatural) LPN-based PRG. Unfortunately, the approach seems strongly tied to very simple types of correlations. Loosely, the lightweight HSS in these constructions *just barely* evaluates the LPN-based PRG, and the only additional homomorphic power (required for using the PRG output to sample

[2] Composition between the PRG and even simple distribution sampling not only increases the computation degree, but also ruins "restricted-multiplication friendliness" of the computation. Several HSS schemes support "restricted" multiplications, where one multiplicand must be an original input. Thus for each multiplication of outputs from the PRG, one must be completely recomputed from scratch.

from the target distribution $\mathcal{D}$) is possibly evaluating the PRG multiplied by a constant. It is not clear how to extend the approach to generate richer correlations such as UV correlations or beyond without making the concrete costs prohibitively impractical.

Sparse PRGs and Building PCGs. Motivated by the above, we introduce and study a new primitive: *sparse* PRGs. As discussed, a sparse PRG is syntactically similar to a standard PRG, namely, a deterministic mapping that stretches a short input vector to a longer output vector. The key difference is that, instead of requiring the output vector be indistinguishable from a uniformly random vector, here we wish it to be indistinguishable from a random *sparse* vector. To this end we allow the use of a *structured* random seed. For instance, the seed can also be sparse.

We will be particularly interested in what we will refer to as *regular* sparse PRGs, where the output is indistinguishable from a random sparse vector with regular structure. In other words, it is comprised of n length-B (pseudorandom) unit vectors. The term "sparse PRG" refers to this regular variant by default.

Given such a primitive, we observe that one can obtain PCGs for n length-B UV correlations from HSS in a *direct* manner, avoiding the generic PRG and distribution-sampling procedure. Instead, one can simply HSS-evaluate the (regular) sparse PRG on the HSS-shared seed. The resulting outputs will be additive shares of the long regular sparse random vector: namely, n random length-B unit vectors as desired. This puts forth a more direct and simpler path toward obtaining *efficient* PCG constructions for UV correlations.

Handling HSS with Non-negligible Error. The described approach applies if the HSS scheme has negligible error and *additive* reconstruction. However, in several HSS constructions, relying on group-based assumptions such as Decisional Diffie Hellman (DDH) or bilinear maps, the additive reconstruction has non-negligible error. This error comes from the procedure of converting an intermediate form of (error-free) *multiplicative* sharing back to additive shares. For example, after a single multiplication in the DDH-based HSS of [25], the two parties successfully hold elements h and $h \cdot g^{ab}$ multiplicatively sharing the product ab.

As an additional contribution, we demonstrate that this intermediate multiplicative sharing can be leveraged to directly give meaningful PCG results without error. More concretely, using such a multiplicative HSS, together with correlation-robust hashing [59], we are able to directly obtain PCGs for the related *weight-1* vector correlation, where the nonzero entry is random. Weight-1 vector correlations already have many useful applications, including any application of DPFs with large (non-binary) output size.

To instead obtain the standard UV correlation, with additive shares of *1* in the nonzero positions instead of random offsets, there are two options forward. From a theory standpoint, one can append "helper" information constituting Private Information Retrieval queries to "correct" erroneous outputs, while maintaining sublinear seed size, as done in [23] (for handling non-negligible HSS error in sublinear-communication secure computation protocols).

We provide an alternative, concretely efficient process, at the expense of a small amount of communication between parties. As such, it does not constitute a strict PCG, but does achieve a relaxed notion of *PCG protocol*, as considered in [18]. Here, the PCG-type nontriviality is because the communication grows sublinearly in the output correlation size. Specifically, to generate n length-B unit vector instances over $(\mathbb{F}^B)^n$ for finite field $\mathbb{F}$, our protocol will require $O(n \log |\mathbb{F}|)$ bits of communication (sublinear for $B \in \omega(1)$).

Our PCG protocol for UV correlations begins by generating the weight-1 correlation over $\mathbb{F}$ as mentioned above. The goal will then be for the parties to identify the random $\mathbb{F}$-payload in each (length-B) weight-1 vector instance, and to divide each element of the vector out by this value (thus the need for field $\mathbb{F}$). This is done without revealing its location within the vector by sending the sum of the B corresponding shares, and is performed iteratively until a nonzero sum is reached (so that division can take place). We refer the reader to Sect. 5.3.1 for further detail.

2.2 Building Sparse PRGs

Our next goal is then to build "HSS-friendly" sparse PRGs. In particular, we will seek to minimize algebraic degree of the sparse PRG computation. Looking ahead, we will present sparse PRG candidates with degrees anywhere from 2 to 5. The degree of the sparse PRG determines the corresponding homomorphic evaluation requirement for the HSS.

A natural approach toward sparse PRGs is to simply start with a standard PRG and use the (dense) pseudorandom output to define a longer sparse output. While this works, it will not yield *low-degree* sparse PRGs. In fact, it can be shown, using a simple application of Schwartz-Zippel lemma, that any sparse PRG which consumes uniformly random seed bits must have degree $d \geq \lceil \log B \rceil$ where B is our desired sparsity parameter. This hints toward a possibility that a *non-uniform* seed distribution can assist in reducing the degree.

Low-Degree Sparse PRGs from Local PRGs (Theorem 2). We describe how to obtain a regular sparse PRG with degree d and sparsity parameter B from any d-local standard PRG with alphabet size B.

While *binary* local PRGs do not seem to be very helpful in directly constructing constant-degree sparse PRGs, it turns out that moving to the non-binary regime completely changes the landscape. Concretely, let G be a d-local (k, n) PRG over Σ of size B. The high-level idea behind the construction is simple: Associate each symbol $x \in \Sigma$ in the local PRG input seed and expanded output with its respective "sparse encoding" unit *indicator* vector $\widetilde{x} \in \{0,1\}^B$ of length $|\Sigma| = B$. Given a B-regular sparse input seed, we can then define the sparse PRG's B-regular expanded output by converting its symbols to the "dense" representation domain, applying the local PRG, and then converting the output back to "sparse" representation. See Sect. 4.2.1 for more details.

The required syntactic sparsity and output-indistinguishability requirements follow directly by construction and the pseudorandomness of the original PRG.

It remains to consider the resulting algebraic degree. For this, recall the starting PRG was d-local; each output Σ-symbol y depends on only d input Σ-symbols, $y = p(x_1, \ldots, x_d)$. An alternative way of expressing this computation is $p(x_1, \ldots, x_d) = \sum_{(u_1,\ldots,u_d)\in\Sigma^d} \left(p(u_1, \ldots, u_d) \cdot \prod_{i=1}^{d}[x_i == u_i]\right)$, where $[x_i == u_i]$ denotes the bit equal to 1 iff x_i is equal to u_i. Over a ring Σ, the computation of $[x_i == u_i]$ has high degree. However, if the input x_i is instead given in its "sparse" representation $\widetilde{x}_i \in \{0,1\}^B$, then the computation of $[x_i == u_i]$ is of degree 1 (namely, the value of the ith entry in $\widetilde{x}_i$). Observing that each $p(u_1, \ldots, u_d)$ is constant, it follows that the overall degree is d.

Related Approaches. We note that the high-level technique for reducing degree via a sparse encoding of the input is quite standard, and was used before in different cryptographic contexts (including instance hiding [8] and information-theoretic private information retrieval [34]). However, in typical applications of this technique, it either results in super-constant degree or in super-polynomial blow-up to the input size. The use of local PRGs over large alphabets is the crucial ingredient that allows up to obtain a small constant degree for any desired level of sparsity.

A related application of using local PRGs with non-binary *input alphabet* (and binary output alphabet) for reducing algebraic degree has been proposed by Lin and Tessaro [66] in the context of basing indistinguishability obfuscation on standard cryptographic assumptions. However, this approach has been essentially abandoned, because its most appealing use case required 2-local PRGs in a restrictive parameter regime that turned out to be too stringent to be realizable [7,67]. In contrast, our main applications can benefit from local PRGs with higher locality and milder stretch requirement, and moreover crucially rely on a large output alphabet.

2.3 Building the $(N, 1)$ Multiplicative DPF (Theorem 3)

We next build an N-party, 1-secure multiplicative DPF, where parties output *CNF shares* [62] of the DPF output (which can typically be converted to degree-1 Shamir shares). That is, for a DPF output value s, each party i outputs $N-1$ values $(s_j)_{j\in[N]\setminus\{i\}}$, such that $\sum_{j\in[N]} s_i = s$. Below we discuss the ideas behind the construction; see the full version for a formal treatment.

At a high level, our mult-DPF scheme adopts the full binary tree structure for DPF introduced and optimized in [24,26]. The leaves of the tree correspond to the input domain $\{0,1\}^\ell$, and an internal node at depth i corresponds to an i-bit prefix. Eval assigns a pseudorandom value in $\{0,1\}^{\lambda+2\log N}$ to each node on the path from the root to the leaf x that it is evaluating. On a node that is off the path to α, the N values computed by $\mathsf{Eval}(k_1, x), \ldots, \mathsf{Eval}(k_N, x)$ are all identical. However, that same N-tuple of values on a node that is on the path to α consists of independent pseudorandom values.

To achieve this distribution of values, each key includes a single independent seed for a PRG G, two *control elements* in the range $0, \ldots, N-1$, and $N-1$

public *correction words* in $\{0,1\}^{\lambda+2\log N}$ for each of the ℓ levels of the tree. Together, the seed and the control elements define the pseudorandom value at the root of the tree. The $N-1$ correction words for each level of the tree are the same for all the keys. The values at the two children of a node v are defined by using G to expand the seed of v to length $2(\lambda + 2\log N)$, and adding to the left (right) half of the expanded string a correction word that is identified by the left (right) control elements of v, with control element 0 corresponding to not adding any correction word.

This structure ensures that if the values generated at v by evaluating two keys k_i, k_j are identical, then the values generated at the children of v using these two keys are also identical. The identity does not depend on the specific values of the control elements and the correction words, only on the fact that they are identical in v.

Therefore, if v is a node on the path to α, and u is its unique off-path child then the goal is to choose the control elements of v and the correction words at the level of u precisely so that the pseudorandom values at u generated across all N keys are identical. The control elements in v determine which correction word is added to the states of the children of v. If the N control elements of v corresponding to its child u and taken across all keys are a permutation of $(0,\ldots,N-1)$ then the correction words for the level of u can be chosen to satisfy the above property. The correction words must be long enough to control the seed at u and the control elements of both children of an on-path node v, so that the control elements corresponding to its off-path child are identical across all keys, while the control elements in its sibling (which is on-path) form a permutation of $(0,\ldots,N-1)$. Choosing this permutation at random, together with the pseudorandom values on the path ensure that the construction satisfies the DPF secrecy requirement.

The final phase of Eval acts in a way that transforms the distribution of pseudorandom values at every leaf $x \neq \alpha$ to a CNF secret sharing of 0 over $\mathcal{R}$, and the pseudorandom values at the leaf α to a CNF secret sharing of β. To achieve that, $\mathsf{Eval}(k_i, x)$ first uses G to expand the seed at the leaf to N elements in $\mathcal{R}$, and adds the final correction words according to the control element at x. Assume that the result computed up to this point (by party i) is $(g_1,\ldots,g_{N-1}) \in \mathcal{R}^{N-1}$. For each $1 \leq i \leq N-1$, $\mathsf{Eval}(k_i, x)$ outputs the shares $(g_j)_{j\neq i}$ together with $g_N := -\sum_{i=1}^{N-1} g_i$, while $\mathsf{Eval}(k_N, x)$ outputs $(g_j)_{j\in[N-1]}$. If the parties all agree on their values of $(g_1,\ldots,g_{N-1})$, as is the case at every leaf $x \neq \alpha$, then the resulting output of the N keys is an $(N,1)$ CNF sharing of 0, regardless of the values of the control elements and of the final correction words. Therefore, the correction words can be chosen with full flexibility, to set the N parties' $(g_1,\ldots,g_{N-1})$ vectors to *any* N desired vectors, in particular such that the final post-processed outputs at α is a CNF sharing of β. (Namely, party N will receive $(r_1,\ldots,r_{N-1})$, party 1 will receive $(r_1-\beta, r_2,\ldots,r_{N-1})$ in order to compute $r_N := \beta - \sum_{i=1}^{N-1} r_i$, etc.) Note that this "programming" ability holds assuming the control elements across all keys at α form a permutation of $(0,\ldots,N-1)$, which is ensured by the previous steps.

The resulting per-party key size will be $O(N \cdot (\lambda\ell + \log|\mathcal{R}|))$, where $\mathcal{R}$ is the output ring for the CNF shares. This corresponds to N correction words in each of the ℓ levels, as well as the final leaf-level correction of the N parties' length-$(N-1)$ vectors over the output ring $\mathcal{R}$.

2.4 Applications

Our PCGs enable producing many independent pseudorandom instances of UV correlations of short to moderate length. Each such UV correlation can be viewed as a random N-party Distributed Point Function (DPF) [26,55] with feasible domain size, and as such we immediately inherit a wide range of corresponding applications. We next discuss some concrete applications.

Beyond UV Correlations. We start by observing that UV correlations can be non-interactively and securely converted to other types of useful correlations.

LUT Correlations. A particularly useful example is that of a generic small-size *lookup table* (LUT) correlation for a given function $f : \mathbb{G}_{\text{in}} \to \mathbb{G}_{\text{out}}$, consisting of additive secret shares over $\mathbb{G}_{\text{out}}$ of a randomly $\mathbb{G}_{\text{in}}$-shifted version of the truth table of f along with the shift amount (see Definition 4) [27,42,45,60]. LUT correlations have useful implications to secure computation over non-arithmetic circuits, as discussed below. It was shown in [19] that a single LUT correlation can be generated from a single UV correlation of the same length. Computationally, by using FFTs the local transformation from UV correlations to LUT correlations can be performed in quasilinear time. Our new PCG constructions for UV correlations directly provide compression of *many* LUT correlations.

Authenticated Versions. The "authenticated" version of a correlation (such as UV or LUT) corresponds to both several instances of the original correlation, as well as an additional copy where each instance is scaled by a global authenticator value α. Such correlations support secure computation protocols with *malicious* security, following the "SPDZ" paradigm [43]. Similarly to previous PCG constructions [15,19,20], most of our PCGs can be easily extended to generating authenticated variants of the target UV or LUT correlations by just doubling the (sublinear) seed size and the (linear) expansion time. The high-level idea is to HSS secret share both a randomly sampled sparse PRG seed $x \in \mathcal{R}^k$ together with $rx \in \mathcal{R}^k$ *scaled* by the desired scalar element $r \in \mathcal{R}$. The goal will be to use this to homomorphically evaluate the sparse PRG output as before, as well as r times the sparse PRG output. To do so, recall that the PRG evaluation is expressible as a degree-d polynomial. Then the r-scaled PRG computation can be expressed as the corresponding modified polynomial, where in each term one copy of an input $x_i \in \mathcal{R}$ is replaced by its scaled counterpart rx_i.

We treat these transformations in more detail in the full version.

Private Reading and Writing. Classic applications of feasible-domain DPFs include private reading applications, such as multi-server private information retrieval (PIR) [34] and other private search protocols. A single DPF enables one PIR query, with communication to each server scaling as the DPF key size. Given

a PCG for UV, the client can instead distribute a *single* compressed PCG seed to each server (in an *offline* phase). For each online PIR query, the servers expand a fresh UV instance with pseudorandom nonzero index $i \in [B]$ (computable by the client). To issue its query $q \in [B]$, the client now need only send the offset $q - i$. This corresponds to optimal online communication complexity as well as, given sublinearity of the PCG seed size in the number of instances, amortized overall communication. (See further discussion in the full version.)

In a dual fashion, our PCGs provide comparable benefits to applications of DPFs in private writing, such as in secure distributed storage [70], voting, and aggregation. This underlies, e.g., Prio-style [35] applications for private collection of aggregate statistics. We remark that the case of private writing over non-binary alphabets is particularly suited to our PCGs for *weight-1* correlations (with random instead of 1 nonzero payload). Here, the client wishes to hide both its write index q and its secret aggregation payload v. In this case, the client would send both an additive offset of the secret index $q - i$, as well as a multiplicative offset v/r to convert secret shares of an expanded pseudorandom weight-1 vector with nonzero index i and payload r to one for the targeted values.

MPC over Non-arithmetic Gates. Access to LUT correlations enable highly efficient secure computation of the corresponding LUT gates [27,42,45,60]. This has been leveraged in the literature to provide significant concrete efficiency improvements for motivated secure computation tasks (e.g., computing an AES S-box using a single LUT gate). Our PCG constructions directly provide compression of *many* LUT correlations. The compressed PCG seeds can either be provided by an outside additional party/dealer, or securely generated via a small-scale generic secure computation. For PCGs with sufficiently good stretch, the communication from the PCG seed-generation procedure will be dominated by the communication cost of consuming the PCG output. This amortization is particularly prominent in the malicious-security setting: security of the full protocol can be reduced to secure generation of the *authenticated* variant of the LUT correlations. For typical circuits, LUT-based secure evaluation can achieve (slightly) sublinear communication in the circuit size, at a moderate computational cost. For instance, consider a neural network with small weights, where LUTs are used to securely evaluate the nonlinear activation functions. We refer the reader to the full version for further discussion.

Binary-to-Arithmetic Conversion. Another useful application of unit-vector correlations (which can be viewed as a special case of LUT) is binary-to-arithmetic conversion. Here the parties have a bit x additively shared over $\mathbb{Z}_2$ and they would like to convert it into an additive sharing over $\mathbb{Z}_n$, where $n > 2$. As shown in prior works [2,52,73], this can be performed using a correlation consisting of a random bit r which is additively shared both over $\mathbb{Z}_2$ and $\mathbb{Z}_n$. Such a correlation can be obtained from UV correlations of just length $B = 2$ over the ring $\mathcal{R} = \mathbb{Z}_2 \times \mathbb{Z}_n$ and outputting the first entry of the (length-2) unit vector. For generating such UV correlations, we can use the generalized version of our Local PRG to Sparse PRG transformation which allows for Local PRGs with different input and output alphabet. Concretely, we can use a Local PRG

with arbitrary input alphabet Σ_{in} and binary output alphabet $\Sigma = \{0,1\}$. Also, note that a Local PRG with even-sized output alphabet Σ' can be converted into one where the output alphabet $\Sigma = \{0,1\}$ by simply modifying the Local PRG predicate function to map half the symbols in Σ' to 0 and the remaining to 1.

Binomial Noise. In the context of differential privacy (DP) [50] applications, one requires the *secure* generation of noise sampled from an appropriate distribution such as Gaussian. Prior works [49,51] do this by approximating Gaussian distribution via Binomial distribution. Secret shared samples of Binomial distribution are generated by first generating shares of independent Bernoulli samples and then summing them up. In terms of communication efficiency, securely generating shares of multiple Bernoulli samples forms the main bottleneck. We observe that our PCG for unit vector correlations directly gives us a way to silently generate many samples of Bernoulli (with the bias $1/B$) by simply treating the first entry in each (secret shared) output unit vector as the (secret shared) Bernoulli sample. Note that in this case if suffices to generate the first entry of each UV.

3 Preliminaries

Notation. We let $[n]$ denote the set $\{1, 2, \ldots, n\}$. Throughout the paper, we use λ to indicate a computational security parameter. By $\mathsf{poly}(\lambda)$ and $\mathsf{negl}(\lambda)$, we mean the class $\lambda^{O(1)}$ and $\frac{1}{\lambda^{\omega(1)}}$ respectively. We sometimes abuse notation and use $\mathsf{poly}(\lambda)$ and $\mathsf{negl}(\lambda)$ to refer to a specific function from the respective class. Given a security parameter λ, we use PPT to denote a probabilitic $\mathsf{poly}(\lambda)$-time algorithm and non-uniform polynomial time to denote a $\mathsf{poly}(\lambda)$-time algorithm with $\mathsf{poly}(\lambda)$-size advice (equivalently, a family of $\mathsf{poly}(\lambda)$-size circuits). We say that two distribution ensembles $X = \{X_\lambda\}_{\lambda\in\mathbb{N}}$ and $Y = \{Y_\lambda\}_{\lambda\in\mathbb{N}}$ are computationally indistinguishable, denoted by $X \approx_c Y$, if for every non-uniform polynomial-time algorithm $\mathcal{A}$ there exists a negligible function $\mathsf{negl}(\lambda)$, such that for all $\lambda \in \mathbb{N}$, we have $|\Pr[\mathcal{A}(X_\lambda) = 1] - \Pr[\mathcal{A}(Y_\lambda) = 1]| \leq \mathsf{negl}(\lambda)$.

We will denote matrices with bold capital letter (e.g. $\mathbf{A}, \mathbf{B}$, etc.) and vectors with bold lowercase letter (e.g. $\mathbf{u}, \mathbf{v}$, etc.). By default, vectors are assumed to be row vectors. We write $\mathbf{A}|_{i,j}$ to denote the $(i,j)^{\text{th}}$ entry of $\mathbf{A}$. For an ordered set/sequence s, we will use $s|_i$ to denote the i^{th} entry of s. Given a vector $\mathbf{v}$ of length $|\mathbf{v}| = n$, the notation $\mathsf{HW}(v)$ denotes the hamming weight of $\mathbf{v}$, i.e., the number of nonzero entries in $\mathbf{v}$. For strings/vectors $\mathbf{v}$, we will use $\mathbf{v}_i$ to denote the i^{th} letter/element. Given a distribution $\mathcal{D}$, we denote by $\mathsf{Im}(\mathcal{D})$ the image of $\mathcal{D}$ (i.e., its support set).

Useful Quantities. The min entropy of a random variable X with values in $\mathcal{X}$, denoted by $\mathsf{H}_\infty(X)$, is defined as $\min_{x\in\mathcal{X}} \log(\frac{1}{\Pr[X=x]})$.

3.1 Pseudorandom Generator (PRG)

Definition 2 (Pseudorandom Generator (PRG)). Let $k = k(\lambda), n = n(\lambda), s = s(\lambda)$ be polynomially bounded functions. Let $\Sigma = \{0, 1, \ldots, s-1\}$

be an alphabet of size $s(\lambda)$ ($s = 2$ by default). We say that a family of functions $G = \{G_\lambda : \Sigma^k \to \Sigma^n\}_{\lambda \in \mathbb{N}}$ is a (k, n) pseudorandom generator (PRG) over Σ if the following conditions hold:

- Nontriviality: For all sufficiently large λ, we have $n(\lambda) > k(\lambda)$;
- Efficiency: G is computable in (uniform) time $\mathsf{poly}(\lambda)$;
- Security: The following ensembles are computationally indistinguishable:

$$\{y | y \xleftarrow{\$} \Sigma^{n(\lambda)}\}_\lambda \approx_c \{G_\lambda(x) | x \xleftarrow{\$} \Sigma^{k(\lambda)}\}_\lambda$$

We will be interested in the following efficiency metrics of G:

- **Stretch:** We will consider either the *additive stretch* $n(\lambda) - k(\lambda)$, the *multiplicative stretch* $n(\lambda)/k(\lambda)$, or the *stretch function* $n(k)$ when the output length is determined by the input length.
- **Locality:** We say that G is *d-local* if for all λ and $j \in [n(\lambda)]$, the j^{th} output symbol of G_λ is a function of at most d input symbols.
- **Degree:** When Σ represents a ring $\mathcal{R}$ (or, more generally, $\Sigma \subseteq \mathcal{R}$), we say that G has *degree* d if for all λ and $j \in [n(\lambda)]$, the j^{th} output of G_λ is a multivariate polynomial (over $\mathcal{R}$) of total degree $\leq d$ in the inputs.

Remark 1. We will sometimes consider a natural generalization of the definition that allows distinct input and output alphabets. When $|\Sigma| = 2$ (for both input and output), we will refer to the PRG as a *binary* PRG.

Remark 2. We note that a PRG G with output alphabet Σ implies a PRG G' with output alphabet Σ' where $|\Sigma'|$ is a divisor of $|\Sigma|$. G' simply runs G and applies an arbitrary balanced map from $\Sigma \to \Sigma'$ on the output of G.

3.2 Homomorphic Secret Sharing (HSS)

Homomorphic secret sharing (HSS) [25] can be viewed as the natural secret-sharing analogue of fully homomorphic encryption. We will be mainly interested in HSS schemes that support the evaluation of *low-degree* multivariate polynomials on shared input vectors. Unlike some HSS schemes for polynomials from the literature [61,65], here we require by default an *additive* representation of the output, namely the output is the sum of the output shares over some publicly known Abelian group. More formally, we consider degree-d HSS over a finite ring $\mathcal{R}$. We view the ring as being implicitly defined by the security parameter λ, and assume that ring operations can be performed in time $\mathsf{poly}(\lambda)$. We refer the readers to the full version for the formal definition.

3.3 Pseudorandom Correlation Generators (PCG)

Pseudorandom correlation generators (PCGs) [19] for a target n-party correlation provide a means for locally expanding short seeds into long pseudorandom

strings with the desired correlation. In this work, we will frequently be considering *unit vector correlations*, wherein n random and independent unit vectors are additively secret shared across two or more parties. We refer the readers to the full version for the formal definition of PCG.

Low-Weight Additive Correlations. In this paper, we will focus on constructing PCGs for forms of *unit vector* and *weight-1* correlations, corresponding to additive secret shares of several independent instances of random unit or weight-1 vectors over a ring $\mathcal{R}$. We additionally consider the correlation of additive secret shares of a single long sparse vector.

Definition 3 (Useful Target Distributions). Given a block length B and a ring $\mathcal{R}$, we define the following distributions:

- $\mathcal{D}_{\mathsf{UV}}(B, \mathcal{R})$ (Unit-vector distribution): Sample a random index $j^* \leftarrow [B]$ and output a vector $\mathbf{e}_{j^*}$, where $\mathbf{e}_{j^*} \in \mathcal{R}^B$ is the unit vector with $1 \in \mathcal{R}$ at index $j^* \in [B]$ and $0 \in \mathcal{R}$ otherwise.
- $\mathcal{D}_{\mathsf{AUV}}(B, \mathcal{R}, n)$ (Authenticated unit-vector distribution): Sample a random value $r \leftarrow \mathcal{R}$ and n vectors $\mathbf{e}_1, \ldots, \mathbf{e}_n$ where $\mathbf{e}_i \leftarrow \mathcal{D}_{\mathsf{UV}}(B, \mathcal{R})$ for all $i \in [n]$. Output $(\{\mathbf{e}_i, r \cdot \mathbf{e}_i\}_{i \in [n]}, r)$.
- $\mathcal{D}_{\mathsf{wt1}}(B, \mathcal{R})$ (Weight-1 vector distribution): Sample a random index $j^* \leftarrow [B]$ and value $r \leftarrow \mathcal{R}$. Output vector $r \cdot \mathbf{e}_{j^*}$, where $\mathbf{e}_{j^*} \in \mathcal{R}^B$ is the unit vector with $1 \in \mathcal{R}$ at index $j^* \in [B]$ and $0 \in \mathcal{R}$ otherwise.
- $\mathcal{D}_{\mathsf{LUT}}(\mathbb{G}_{\mathrm{in}}, \mathbb{G}_{\mathrm{out}})$ (Lookup-table distribution): Let $\mathbb{G}_{\mathrm{in}}$ be an arbitrary abelian group of order B with group operation $+_{\mathbb{G}_{\mathrm{in}}}$ (the usual choice of $\mathbb{G}_{\mathrm{in}}$ is $\mathbb{Z}_B$ or $(\mathbb{Z}_2)^b$ where $B = 2^b$). Given a public truth table $T : \mathbb{G}_{\mathrm{in}} \to \mathbb{G}_{\mathrm{out}}$, sample a random offset $r \leftarrow \mathbb{G}_{\mathrm{in}}$. Output $(\{y_i\}_{i \in \mathbb{G}_{\mathrm{in}}}, r)$ where $y_i = T(i +_{\mathbb{G}_{\mathrm{in}}} r)$.

Definition 4 (Useful Target Correlations). Given a block length B, ring $\mathcal{R}$, and number of parties N, a correlation generator $\mathcal{C}_*$ for distribution $\mathcal{D}_*$ simply outputs N-party additive secret shares of a sample from $\mathcal{D}_*$ over the appropriate domain, where $* \in \{\mathsf{UV}, \mathsf{wt1}, \mathsf{LUT}\}$. We use a $\mathcal{C}_*^n$ notation to denote a concatenation of n independent samples from the correlation $\mathcal{C}_*$. For authenticated unit-vector, we have an additional parameter n denoting the number of instances and the correlation generator $\mathcal{C}_{\mathsf{AUV}}(B, \mathcal{R}, n)$ simply outputs N-party additive secret shares of a sample from $\mathcal{D}_{\mathsf{AUV}}(B, \mathcal{R}, n)$ over $\mathcal{R}$. When $\mathcal{R}$ is omitted, it is understood to be $\mathbb{Z}_2$.

Remark 3. Note that the correlation $\mathcal{C}_*$ w.r.t a distribution $\mathcal{D}_*$ (Definition 4) is reverse-sampleable (see the full version for details) for any $t < N$. This follows because given any $T \subseteq [N]$ shares $(R_i)_{i \in [|T|]}$ where $|T| < N$, the $\mathsf{RSample}$ algorithm can sample uniformly random $N - |T| - 1$ shares $(R'_i)_{i \in [N-|T|-1]}$ and set the last remaining share $R'_{N-|T|}$ to be $X - \sum_{i \in [|T|]} R_i - \sum_{i \in [N-|T|-1]} R'_i$. It finally outputs $\{R'_i\}_{i \in [N-|T|]}$.

Remark 4. In the full version, we show that $\mathcal{C}_{\mathsf{LUT}}(\mathbb{G}_{\mathrm{in}}, \mathbb{G}_{\mathrm{out}})$ and $\mathcal{C}_{\mathsf{UV}}(B, \mathcal{R})$ are equivalent to each other in the sense that they can be locally and securely converted from one form to another.

4 Sparse Pseudorandom Generators

In this section, we define our new notion of *sparse* PRGs and describe a general construction based on local PRGs over non-binary alphabets. Finally, we discuss different instantiations of such local PRGs, based on both existing candidates and new candidates.

4.1 Defining Sparse PRGs

At a high level, a sparse PRG naturally extends the traditional notion of PRG by requiring the output to be indistinguishable from a random "sparse" vector. We consider two notions of sparsity: a *regular* B-sparse distribution is a concatenation of random length-B unit vectors (containing 1 in a random position and 0 elsewhere), whereas a *non-regular* B-sparse distribution is a concatenation of independent Bernoulli variables with success $1/B$ (i.e., each output entry is 1 with probability $1/B$ and 0 otherwise). By default, we will focus on regular sparsity for this paper. Finally, we will be interested in minimizing the *algebraic degree* of the sparse PRG over some underlying ring $\mathcal{R}$ ($\mathbb{Z}_2$ by default). To this end, we will allow the sparse PRG seed to be sampled from an arbitrary distribution.

Definition 5 (Sparse PRG). Let λ denote a security parameter, and $k = k(\lambda)$ (seed length), $n = n(\lambda)$ (output weight), and $B = B(\lambda)$ (sparsity parameter) be positive integers. Let $\mathcal{R} = \mathcal{R}(\lambda)$ be a finite ring, where $\mathcal{R} = \mathbb{Z}_2$ by default. A (regular) sparse PRG with parameters $(k, n, B, \mathcal{R})$ is defined by a pair of algorithms $\mathsf{SPRG} = (\mathsf{SPRG.Gen}, \mathsf{SPRG.Exp})$ with the following syntax:

- $\mathsf{SPRG.Gen}(1^\lambda)$ is a PPT algorithm that given a security parameter λ outputs a seed $x \in \mathcal{R}^k$.
- $\mathsf{SPRG.Exp}(x)$ is a deterministic polynomial-time seed expansion algorithm that given a seed $x \in \mathcal{R}^k$ outputs a vector $y \in \mathcal{R}^{nB}$.

The above algorithms should satisfy the following requirements:

- Security: The output distribution of $\mathsf{SPRG.Exp}(x)$, where x is sampled from $\mathsf{SPRG.Gen}(1^\lambda)$, is computationally indistinguishable from the ideal B-sparse output distribution $\mathcal{D}_{n,B}$, defined to be the concatenation of n independently sampled random unit vectors in $\mathcal{R}^B$.
- Non-triviality: For all sufficiently large λ, we have $\mathsf{H}_\infty(\mathsf{SPRG.Gen}(1^\lambda)) \leq \mathsf{H}_\infty(\mathcal{D}_{n,B}) - 1$.

We say that SPRG has *degree* d if each output entry of $\mathsf{SPRG.Exp}(x)$ is a multivariate polynomial of degree at most d in x (over $\mathcal{R}$).

A central objective of this work is the construction of *constant-degree* sparse PRGs for any polynomial $B(\lambda)$ under plausible cryptographic assumptions.

Remark 5 (On the non-triviality requirement). The above non-triviality requirement implies that the $\mathsf{SPRG.Exp}$ must add a non-negligible amount of (pseudo-) entropy to the seed, ruling out information-theoretic constructions where computational indistinguishability is replaced by statistical indistinguishability.

Remark 6 (On different notions of stretch). Unlike the standard notion of a PRG, where the stretch is uniquely defined by input and output lengths, there are several useful notions of stretch for a sparse PRG. The most natural one, corresponding to our non-triviality requirement, compares $n \log B$ (the pseudo-entropy of the output) to the entropy of the seed. However, it will often be useful to consider a more stringent notion of stretch comparing $n \log B$ to the entropy of a *uniformly random* seed in $\mathcal{R}^k$, namely the bit-length $k \log |\mathcal{R}|$ of the seed. This corresponds to applications where the seed needs to be communicated and stored in uncompressed form. Most of our constructions of constant-degree sparse PRGs will in fact achieve superlinear stretch even under the more stringent notion.

Remark 7 (On the need for a non-uniform seed distribution). For any k-variate nonzero polynomial p of degree d over $\mathbb{Z}_2$, the probability that $p(x) = 1$ when evaluated at a random $x \in \{0,1\}^k$ is at least $1/2^d$. It follows that any sparse PRG over $\mathbb{Z}_2$ with uniform seed distribution and sparsity parameter B must have degree $d \geq \lceil \log B \rceil$. See the full version for details.

Remark 8 (On regular vs. non-regular sparsity). Our main notion of a sparse PRG considers an ideal target distribution which is *regular* in the sense that each block of B output entries contains exactly a single 1. This is motivated by a variety of cryptographic applications. However, one may also consider a natural *non-regular* variant, where each entry of the ideal target distribution $\mathcal{D}_{n,B}$ is independently sampled to be 1 with probability $1/B$ and 0 otherwise. While not as useful as the regular variant, non-regular sparse PRGs can also be motivated by applications such as secure distributed generation of secret-shared discrete Gaussians or LPN noise. Note that any regular sparse PRG with a sufficiently good stretch can be converted into a non-regular sparse PRG of the same degree by simply outputting the first entry of each block.

4.2 Low-Degree Sparse PRGs from Non-binary Local PRGs

We will now describe our main construction of low-degree sparse PRGs from local PRGs. Looking ahead, being low-degree will serve as an HSS-friendly property, enabling the construction of efficient PCGs under a variety of assumptions. We start by defining convenient notation for describing local functions. For consistency with the standard terminology of binary local PRGs, we will abuse the term "predicate" and use it even in the non-binary case.

Definition 6 (Local functions). Given a sequence Π of d-tuples $\pi_1, \ldots, \pi_n \in [k]^d$, and sequence P of predicates $p_1, \ldots, p_n : \Sigma^d \to \Sigma$, the d-local function $G_{\Pi,P} : \Sigma^k \to \Sigma^n$ is naturally defined by $G_{\Pi,P}(x) = (p_1(x_{\pi_1}), \ldots, p_n(x_{\pi_n}))$, where for any $i \in [n]$, $x_{\pi_i} \in \Sigma^d$ is the d-tuple of input symbols in x indexed by

the tuple $\pi_i \in [k]^d$. Namely, for $x = (x_1, \ldots, x_k)$, and $\pi_i = (\pi_{i,1}, \ldots, \pi_{i,d})$, we have $x_{\pi_i} = (x_{\pi_{i,1}}, \ldots, x_{\pi_{i,d}})$.

We will sometimes use a generalized version that allows for different input and output alphabets, and will sometimes refer to only a single predicate p, with the understanding that this means that $p_i = p$ for all $i \in [n]$.

4.2.1 Sparse PRG from Local PRG

We now describe a simple construction of degree-d sparse PRGs with sparsity parameter B from d-local PRGs over alphabet of size B.[3] We defer the proof of the following theorems to the full version.

Theorem 5. *Let $d \in \mathbb{N}$ and let B, k, n be polynomial in λ. Suppose there is a d-local (k, n)-PRG G over Σ of size B. Then, for any ring $\mathcal{R} = \mathcal{R}(\lambda)$, there is a $(k', n', B, \mathcal{R})$ sparse PRG G' of degree d (over $\mathcal{R}$) where $k' = k \cdot B$ and $n' = n$. Moreover, G' can be computed using $O(ndB^d)$ operations in $\mathcal{R}$.*

The formal proof is given in the full version; here we provide the construction. Let $\widetilde{u} \in \{0,1\}^B$, where $0 \in \mathcal{R}$ and $1 \in \mathcal{R}$, indicate the unit-vector encoding of a symbol $u \in \Sigma$, and let $\widetilde{u}^i$ indicate the i^{th} bit (0-indexed) of $\widetilde{u}$. For example, if $\Sigma = \{0,1,2,3\}$ and $u = 2$, then $\widetilde{u} = [0,0,1,0]$, $\widetilde{u}^0 = 0$, $\widetilde{u}^2 = 1$. Assuming we have a d-local (k, n) PRG G over Σ of size B, with a sequence of d-tuples $\pi_1, \ldots, \pi_n \in [k]^d$ and predicate $p : \Sigma^d \to \Sigma$ (Definition 6), we now provide a formal description of the sparse PRG G' with parameters $(k' = kB, n' = n, B, \mathcal{R})$ in terms of the syntax defined in Definition 5.

- $\mathsf{SPRG.Gen}(1^\lambda) \to \widetilde{x}$: Sample a uniform $x \in \Sigma^k$ and sets $\widetilde{x} = \widetilde{x_1} || \ldots || \widetilde{x_k}$, where for any $i \in [k]$, $\widetilde{x_i}$ is a length-B unit-vector encoding of x_i. Output $\widetilde{x} \in \{0,1\}^{k'}$ as the seed $(k' = kB)$, where $0 \in \mathcal{R}$ and $1 \in \mathcal{R}$.
- $\mathsf{SPRG.Exp}(\widetilde{x})$: On input $\widetilde{x} \in \{0,1\}^{k'}$, parse $\widetilde{x}$ as k length-B vectors, namely $\widetilde{x_1} || \ldots || \widetilde{x_k}$. Output $\widetilde{y} = \widetilde{y_1} || \ldots || \widetilde{y_{n'}}$, where $n' = n$ and for any $i \in [n']$, $\widetilde{y_i} \in \{0,1\}^B$ is computed in the following way:

$$\forall j \in [B] : \widetilde{y_i}^j = \sum_{\substack{(u_1,\ldots,u_d) \in \Sigma^d \\ p(u_1,\ldots,u_d) = j}} \widetilde{x_{\pi_{i,1}}}^{u_1} \cdot \ldots \cdot \widetilde{x_{\pi_{i,d}}}^{u_d} \tag{1}$$

  where the multiplications and additions are performed over $\mathcal{R}$.

Corollary 1. *In Theorem 5, if Σ is a group of size B, then sparse PRG G' can be computed using $n(d-1)B^2$ group operations.*

[3] We describe a natural generalization of our construction in the full version which allows for Local PRGs with different input and output alphabet. Here we restrict to Local PRGs with same input and output alphabet for simplicity.

Remark 9 (Further speedup using FFT). Corollary 1 gives the evaluation time using the naive convolution over groups where per convolution has cost $O(B^2)$. When Σ is instantiated with *alternating groups* (which is non-Abelian), this $O(B^2)$ term can be reduced to $O(B \log^{1.5} B)$ using FFT [47] (see details in the full version). In Sect. 4.2.3, we propose a FFT-friendly local PRG candidate (on Abelian groups) with $O(B \log B)$ evaluation time per unit vector, with additional acceleration by hardware support for polynomial multiplication over $\mathbb{Z}_2$ [57].

4.2.2 Non-Binary Local PRG from Binary Local PRG

We observe that any binary d-local PRG implies a d-local PRG over any Σ whose size is a power of 2. This fact, captured by the next lemma, will be useful for basing low-degree sparse PRGs on (by-now) standard assumptions. The proof of Lemma 1 is deferred to the full version.

Lemma 1. *Let $d, b \in \mathbb{N}$ and $B = 2^b$. Assuming an d-local (k, n) binary PRG, there exists an d-local (k, n)-PRG over $\Sigma = \{0, \ldots, B-1\}$.*

Combining Lemma 1 and Theorem 5, we get the following corollary.

Corollary 2 (Sparse PRG from binary local PRG). *Let $d \in \mathbb{N}$ and let B, k, n be polynomial in λ. Suppose there is a d-local (k, n) binary PRG G. Then, for any ring $\mathcal{R} = \mathcal{R}(\lambda)$, there is a $(k', n', B, \mathcal{R})$ sparse PRG G' of degree d where $k' = k \cdot B$ and $n' = n$.*

Conjectures on Binary Local PRGs. We provide the main conjecture here which captures the current state of the art and refer the readers to the full version for a detailed discussion.

Conjecture 1 (Binary local PRGs). There exists a 5-local binary PRG with stretch $n = k^{1.5-\epsilon}$, for any $\epsilon > 0$, and an d-local binary PRG with stretch $n = k^{\Omega(d)}$ for any constant d. In both cases, $k(\lambda) = \lambda^{O(1)}$ suffices for security against 2^{λ}-size distinguishers.

A bit more explicitly, for the case $d = 5$ it is conjectured that the "TSA predicate" $p = x_1x_2+x_3+x_4+x_5$ suffices, and for general d the "MAJ-XOR" predicate $p = \mathsf{MAJ}(x_1, \ldots, x_{\lfloor d/2 \rfloor}) + \mathsf{XOR}(x_{\lfloor d/2 \rfloor+1}, \ldots, x_d)$ suffices (where additions are modulo 2). These conjectures are supported by provable security against linear attacks and other kinds of attacks [5,68].

Combining Corollary 2 and Conjecture 1, we get the following corollary.

Corollary 3 (Degree of sparse PRG from binary local PRG). *Let $e > 1$ be a stretch parameter and $B = 2^b$ polynomial in λ. Suppose Conjecture 1 holds. Then, for any ring $\mathcal{R} = \mathcal{R}(\lambda)$, there exists a $(k, n, B, \mathcal{R})$ sparse PRG of degree $d = O(e)$, or $d = 5$ if $e < 1.5$, where $k = \lambda^{O(1)} \cdot B$ and $n = k^e$.*

4.2.3 Old and New Candidates for Non-Binary Local PRG

We turn to describe candidates for *direct* constructions of local PRG over non-binary alphabets, motivated by the possibility to get below the $d \geq 5$ locality bound of the binary construction. We list here our conjectures and defer detailed description and the reasoning behind our choices to the full version.

Local PRG from Non-Abelian Groups. We start with an elegant candidate construction based on simple non-Abelian groups, proposed by Barak et al. [7].

Conjecture 2 (d-local PRG candidate based on non-Abelian groups [7]). Let $d \geq 3$ be a locality parameter and $\Sigma = \Sigma(\lambda)$ be a finite, simple non-Abelian group $\mathbb{G}$ (by default: an alternating group $\mathbb{G} = A_s$) with group operation $*$ and size $|\mathbb{G}| \leq \mathsf{poly}(\lambda)$. Consider the d-local function $G_{\Pi,P}$ over Σ (Definition 6) where

- Π is a sequence of randomly chosen d-tuples from $[k]^d$;
- P consists of n copies of the predicate

$$p : (x_1, \ldots, x_d) \mapsto x_1 * \ldots * x_d.$$

Then, for every $\epsilon > 0$, $G_{\Pi,P}$ is an d-local (k, n) PRG over Σ for $k = \lambda$ and $n = k^{d/2-\epsilon}$ (except for $\lambda^{-\Omega(1)}$ failure probability over the choice of Π).

New Candidate: "Generalized Sparse LWR" PRG. Our next candidate can be viewed as a conservative *sparse* variant of the Learning with Rounding (LWR) assumption [6] and is motivated by the goal of speeding up the asymptotic and concrete time required for evaluating the sparse PRGs (and the PCG constructions that use them) via standard FFT algorithms. This exploits the fact that Abelian group addition in the dense domain corresponds to standard convolution in the sparse domain.

Conjecture 3 (d-local "Generalized Sparse LWR" PRG candidate). Let $d \geq 3$ be a locality parameter, $c \geq 2$ be an information loss parameter ($c = 2$ by default), $B = B(\lambda)$ and $B_{\mathsf{in}} = c \cdot B$. We define the cyclic groups $\Sigma_{\mathsf{in}} = \mathbb{Z}_{B_{\mathsf{in}}}$ and $\Sigma = \mathbb{Z}_B$. Consider the d-local function $G_{\Pi,P}$ over input alphabet Σ_{in} and output alphabet Σ (Definition 6) where

- Π is a sequence of randomly chosen d-tuples from $[k]^d$;
- P is a sequence of predicates $p_1, \ldots, p_n$ defined by

$$p_i : (x_1, \ldots, x_d) \mapsto L_i\Big(\mathsf{Perm}_{i,1}(x_1) + \ldots + \mathsf{Perm}_{i,d}(x_d)\Big),$$

  where for all $i \in [n]$ and $j \in [d]$, $\mathsf{Perm}_{i,j} : \Sigma_{\mathsf{in}} \to \Sigma_{\mathsf{in}}$ is an independently chosen (public) random permutation over Σ_{in}, and the sum is taken over $\Sigma_{\mathsf{in}} = \mathbb{Z}_{B_{\mathsf{in}}}$. Furthermore, for all $i \in [n]$, $L_i : \Sigma_{\mathsf{in}} \to \Sigma$ is lossy c-to-1 "rounding" function $L_i : \mathbb{Z}_{B_{\mathsf{in}}} \to \mathbb{Z}_B$ mapping y to $\lfloor y/c \rfloor$.

Suppose $B(\lambda) = \omega(\log \lambda)$. Then, for every $\epsilon > 0$, $G_{\Pi,P}$ is a d-local (k, n) PRG over input alphabet $\Sigma_{\mathsf{in}} = \mathbb{Z}_{B_{\mathsf{in}}}$ and output alphabet $\Sigma = \mathbb{Z}_B$ for $k = \lambda$ and $n = k^{d/2-\epsilon}$ (except for $\lambda^{-\Omega(1)}$ failure probability over the choice of Π).

Note that this candidate is intuitively more conservative than a sparse varaint of standard LWR in the following sense. In LWR, we have a strong "algebraic" structure of scalar multiplications instead of our random permutations. Here we eliminate this structure. We also note that a 3-local variant of sparse (binary) LPN has been previously used in the literature (cf. [17]), where the locality does not count the LPN noise. Here the "noise" is obtained via the rounding.

Finally, we note that a simpler variant of our candidate that avoids the final rounding step (with inputs and outputs in $\mathbb{Z}_B$) can be broken when $n > kB$. Indeed, suppose $B = 2^b$. Now, viewing each permutation as a mapping from $\mathbb{Z}_2^b$ to $\mathbb{Z}_2^b$, this mapping involves B different monomials, with a total of kB monomials over all permutations. The key observation is that the least significant bit of additions in $\mathbb{Z}_B$ can be viewed as (noiseless) addition in $\mathbb{Z}_2$, so each of the n outputs gives a new $\mathbb{Z}_2$ linear equation in the kB monomials. The rounding step eliminates this kind of attacks by adding noise to the linear equations.

New Candidate: 2-Local PRG with Slightly Superlinear Stretch. We turn to our final (and most speculative) candidate, which is a variant of "random balanced 2-local function" constructions proposed and analyzed in [7,66].

Conjecture 4 (2-local PRG candidate). Let Σ_{out} be an output alphabet of size $B = B(\lambda)$ and Σ_{in} be an input alphabet of size $m \cdot B$ for $m = m(\lambda) > 1$. Consider the 2-local function $G_{\Pi,P}$ over Σ (Definition 6) with the following parameters.

- Π is a sequence of n pairs that form a graph with girth at least $g = g(k)$ over the node set $[k]$;
- P is a sequence of n random 1-resilient predicates $p_i : \Sigma_{\text{in}} \times \Sigma_{\text{in}} \to \Sigma_{\text{out}}$. A predicate $p(x_1, x_2)$ is 1-resilient if its output is uncorrelated with each input; namely, for each (y, x_1) there are exactly $m = |\Sigma_{\text{in}}|/|\Sigma_{\text{out}}|$ values of z such that $p(x_1, z) = y$, and similarly for each (y, x_2).

Suppose B, m, k, n, g are functions of λ such that: (1) $mB \leq \lambda^{O(1)}$ (for efficiency of the sparse PRG); (2) $(mB)^k \geq \lambda^{\omega(1)}$ (to avoid brute-force attack on the seed); (3) $n \leq k \log k \cdot B$ (to avoid attacks on 2-local PRGs from [7,67]); (4) $m^g \geq \lambda^{\omega(1)}$ (to avoid cycle-based attacks); (5) $n \log B \geq \omega(k \log(mB))$ (for superlinear stretch). Then, $G_{\Pi,P}$ is a 2-local (k, n) PRG over input alphabet Σ_{in} of size mB and output alphabet Σ_{out} of size B.

In particular, the above conditions can be all met by setting $k = \lambda$, $B = k$, $m = \sqrt{k}$, $n = k \cdot \log^2 k$ (or even $n = k \cdot 2^{\sqrt{\log k}}$), and where Π defines a graph with k nodes, $n(k)$ edges, and girth $g(k) = \omega(1)$.

Because of the poor asymptotic stretch of this candidate, it is mostly of theoretical interest. Determining whether it can lead to concretely efficient constructions will require a refined security analysis.

5 PCG for Unit-Vector Correlations

In this section, we show how to use the sparse PRGs from the previous section to obtain *pseudorandom correlation generators* for unit vector and related useful correlations, by making use of homomorphic secret sharing (HSS) [25]. Here we discuss the resulting PCG constructions.

5.1 General Blueprint from HSS

In this section, we demonstrate a simple framework for obtaining PCGs for unit vector and related correlations from sparse PRGs by use of *homomorphic secret sharing* (HSS). Recall that HSS is a form of secret sharing that supports homomorphic evaluation on individual shares, such that the resulting evaluated outputs additively recombine to the desired computation output. Below we give the theorem for the PCG framework and defer the proof to the full version.

PCG for Unit Vector Correlations using HSS
$(\mathsf{SPRG.Gen}, \mathsf{SPRG.Exp})$ is a $(k, n, B, \mathcal{R})$ sparse PRG; $(\mathsf{HSS.Gen}, \mathsf{HSS.Eval})$ is N-party HSS over $\mathcal{R}$.

- $\mathsf{PCG.Gen}(1^\lambda)$: Sample seed $x \leftarrow \mathsf{SPRG.Gen}(1^\lambda)$. Output the shares $(s_1, \ldots, s_N) \leftarrow \mathsf{HSS.Share}(1^\lambda, x)$.
- $\mathsf{PCG.Expand}(i, s_i)$: Output the homomorphic evaluation $y_i = \mathsf{HSS.Eval}(i, s_i, \mathsf{SPRG.Exp}(\cdot))$ of the degree-d computation $\mathsf{SPRG.Exp}$ applied to the HSS share s_i.

Fig. 2. PCG for unit vector correlations $\mathcal{C}_{\mathsf{UV}}(B, \mathcal{R})^n$ from sparse PRG and HSS.

Theorem 6 (Framework: PCG for Unit Vector Correlations). *Let* $\mathsf{SPRG} = (\mathsf{SPRG.Gen}, \mathsf{SPRG.Exp})$ *be a degree-d sparse PRG with parameters* $(k, n, B, \mathcal{R})$ *(Definition 5). Let* $(\mathsf{HSS.Gen}, \mathsf{HSS.Eval})$ *be an N-party, t-secure HSS scheme for degree-d polynomials over* $\mathcal{R}$*, with per-party share size s for inputs in* $\mathcal{R}^k$*, and pseudorandom outputs. Then the scheme* $(\mathsf{PCG.Gen}, \mathsf{PCG.Expand})$ *from Fig. 2 is a t-secure PCG with seed size s for correlation* $\mathcal{C}_{\mathsf{UV}}(B, \mathcal{R})^n$ *as in Definition 4.*

We next turn to producing a PCG for the *authenticated* unit vector correlation, $\mathcal{C}_{\mathsf{AUV}}(B, \mathcal{R}, n)$. The high-level idea is to HSS secret share both a randomly sampled sparse PRG seed $x \in \mathcal{R}^k$ together with $rx \in \mathcal{R}^k$ *scaled* by the desired scalar element $r \in \mathcal{R}$. The goal will be to use this to homomorphically evaluate the sparse PRG output $\mathsf{SPRG.Exp}(x)$ as before, as well as r times the sparse PRG output, $r \cdot \mathsf{SPRG.Exp}(x)$. To do so, recall that $\mathsf{SPRG.Exp}$ is expressible as a degree-d polynomial. Then $r \cdot \mathsf{SPRG.Exp}(x)$ can be expressed as the corresponding modified polynomial, where in each term one copy of an x_i is replaced by its scaled counterpart rx_i (see f in Fig. 3). The resulting PCG construction has twice the key size, and twice the expansion time, due to the additional value rx to be HSS secret shared. The proof of the following proposition holds directly from that of Thoerem 6.

Proposition 1 (PCG for authenticated unit vector correlations). *Let* $(\mathsf{SPRG.Gen}, \mathsf{SPRG.Exp})$ *and* $(\mathsf{HSS.Gen}, \mathsf{HSS.Eval})$ *be as in Theorem 6. Then the scheme* $(\mathsf{PCG.Gen}, \mathsf{PCG.Expand})$ *from Fig. 3 is a secure PCG with seed size* $2s$ *for correlation* $\mathcal{C}_{\mathsf{AUV}}(B, \mathcal{R}, n)$ *as in Definition 4.*

PCG for Authenticated Unit Vector Correlations using HSS
$(\mathsf{SPRG.Gen}, \mathsf{SPRG.Exp})$ is a $(k, n, B, \mathcal{R})$ sparse PRG; $(\mathsf{HSS.Gen}, \mathsf{HSS.Eval})$ is N-party HSS over $\mathcal{R}$.

- $\mathsf{PCG.Gen}(1^\lambda)$: Sample seed $x \leftarrow \mathsf{SPRG.Gen}(1^\lambda)$ (where $x \in \mathcal{R}^k$), and random $r \leftarrow \mathcal{R}$.
  Sample shares $(s_1, \ldots, s_N) \leftarrow \mathsf{HSS.Share}(1^\lambda, (x, rx))$, where $(x, rx) \in \mathcal{R}^{2k}$.
  Output $(s_1, \ldots, s_N)$.
- $\mathsf{PCG.Expand}(i, s_i)$: Output $y_i = \mathsf{HSS.Eval}(i, x_i, \mathsf{DoubleExpand}(\cdot))$, where $\mathsf{DoubleExpand} : \mathcal{R}^k \times \mathcal{R}^k \to \mathcal{R}^{nB}$ is the degree-d $\mathsf{DoubleExpand}(x, z) = (\mathsf{SPRG.Exp}(x), f(x, z))$, for f defined as follows. Given $\mathsf{SPRG.Exp}(x) =$

$$\sum_{i_1 \le \cdots \le i_d \le k} \alpha_{i_1,\ldots,i_k} \left(\Pi_{j=1}^{d} x_{i_j}\right), \text{ then } f(x, z) := \sum_{i_1 \le \cdots \le i_d \le k} \alpha_{i_1,\ldots,i_k} \left(z_{i_1} \Pi_{j=2}^{d} x_{i_j}\right).$$

Fig. 3. PCG for auth. UV correlations $\mathcal{C}_{\mathsf{AUV}}(B, \mathcal{R}, n)$ from sparse PRG and HSS.

5.2 HSS for Low-Degree Polynomials

We next discuss relevant existing HSS constructions from the literature, as well as a new DPF-based construction that is tailored to the case of sparse inputs.

5.2.1 Honest-Majority HSS (Information Theoretic & OWF)

Theorem 7 (Information-Theoretic HSS [9,33,41]). *For any $N, d, t, n \in \mathbb{N}$ for which $N > dt$, and any ring $\mathcal{R}$, there exists information-theoretic N-party t-secure HSS for degree-d polynomials over $\mathcal{R}$, in which the share of each party for inputs in $\mathcal{R}^k$ consists of at most $\mathsf{poly}(N) \cdot k$ elements of $\mathcal{R}$.*

In particular, for $\mathcal{R} = \mathbb{F}$ finite field of size $|\mathbb{F}| \ge N$, this is achievable with each share in $\mathcal{R}$; for $\mathcal{R} = \mathbb{Z}_2$, it is achievable with each share in $\mathcal{R}^{\lceil \log N \rceil}$ (both via Shamir secret sharing).

Note that we will typically consider the number of parties N as constant, in which case the $\mathsf{poly}(N)$ term in the share size expression above is itself constant.

Remark 10 (Compressing IT-HSS shares using OWF). In the information theoretically secure HSS scheme above, t of the parties' shares are random ring elements. By making use of pseudorandom generators, these random shares can be compressed, decreasing the corresponding share size [11,40].

"Sparse HSS" via Multiplicative DPF. Recall from the discussion in the Introduction that we can use a *multiplicative* form of Distributed Point Functions (DPF) [26,55] to obtain a notion of degree-d HSS for *sparse* inputs that is sufficient for our PCG construction framework. In effect, this will enable us to significantly compress the HSS share size (corresponding later to PCG seed size), while still maintaining low-degree homomorphism, based on just PRGs.

More concretely, we provide a new construction of N-party, 1-secure multiplicative DPFs where the parties locally compute CNF shares of the corresponding outputs. This further enables non-interactive conversion to an assortment of

other linear secret sharing schemes, including Shamir shares. Our construction improves on the (3,1) CNF-DPF of [32], both by extending to $(N, 1)$ for $N > 3$, and improving the seed size (and complexity of share generation and evaluation) for $N = 3$. We prove the following theorem in the full version.

Theorem 8 ($(N, 1)$-multiplicative DPF). *Assuming a black-box access to a PRG with seed length λ, there is an $(N, 1)$-CNF DPF for point functions $f_{\alpha,\beta} : \{0,1\}^{\ell} \to \mathcal{R}$ with per-party key size $O(N \cdot (\lambda\ell + \log|\mathcal{R}|))$.*

A simple corollary of Theorem 8 is a PRG-based N-party HSS for multivariate polynomials of degree $d = (N-1)$ over $\mathcal{R}$, with B-regular sparse inputs in $\mathcal{R}^{kB}$, with per-party share size $O(Nk \cdot (\lambda \log B + \log|\mathcal{R}|))$.

5.2.2 Dishonest Majority

We next present HSS results from the literature within the setting of dishonest majority; particularly, 2-party HSS.

Theorem 9 (2-party additive HSS). *The below efficiency costs correspond to HSS sharing of an input in $\mathcal{R}^k$ for the listed ring $\mathcal{R}$, and for homomorphic evaluation of m multiplication gates wherein each gate involves an $\mathcal{R}$-element from the original input.*

- *(QR). [29,48] Assume the Quadratic Residuosity assumption holds for Z_M^* for $M \in \mathbb{N}$. Then there exists 2-party HSS for degree-2 polynomials over $\mathcal{R} = \mathbb{Z}_2$, with share size $O(k)$ Z_M^*-elements, and evaluation cost dominated by $O(m)$ exponentiations in Z_M^*.*
- *(DCR). [69,74] Assume the Decisional Composite Residuosity assumption holds for $Z_{M^2}^*$ for $M \in \mathbb{N}$. Then there exists 2-party HSS for degree-d polynomials over $\mathcal{R} = \mathbb{Z}_M$, with share size $O(k)$ Z_M-elements, and evaluation cost dominated by $O(md)$ exponentiations in $Z_{M^2}^*$.*

Definition 7 (Multiplicative HSS). Let d be a positive integer, and $\mathbb{G}$ a cyclic group of order q with generator g. An N-party, t secure degree-d multiplicative HSS scheme over $\mathbb{G}$ is a pair of PPT algorithms $(\mathsf{MHSS.Gen}, \mathsf{MHSS.Eval})$ with the same syntax and t-security of additive HSS over $\mathbb{Z}_q$, but with the following modified correctness property.

- **Multiplicative correctness:** For every $\mathsf{poly}(\lambda)$ there exists a negligible $\mathsf{negl}(\lambda)$ such that for every λ, input $x \in \mathcal{R}^k$ (where $\mathcal{R} = \mathcal{R}(\lambda)$), and degree-$d$ arithmetic circuit P of size $\mathsf{poly}(\lambda)$ we have: $\Pr\left[\Pi_{i\in[N]} y_i \neq g^{P(x)}\right] \leq \mathsf{negl}(\lambda)$, where probability is taken over $(s_1, \ldots, s_N) \leftarrow \mathsf{MHSS.Share}(1^\lambda, x)$ and for $i \in [N]$, $y_i = \mathsf{MHSS.Eval}(i, s_i, P)$.

Theorem 10 (2-party Multiplicative HSS). *The below efficiency costs correspond to HSS sharing of an input in $\mathcal{R}^k$ for the listed ring $\mathcal{R}$, and for homomorphic evaluation of m multiplication gates wherein each gate involves an $\mathcal{R}$-element from the original input.*

- *(DDH). [25] Let $\mathbb{G}$ be a cyclic group of order p for which the Decisional Diffie-Hellman assumption holds. There exists 2-party multiplicative HSS for degree-2 polynomials over $\mathcal{R} = \mathbb{Z}_p$ with share size $O(k)$ $\mathbb{G}$-elements, and evaluation cost dominated by $O(m)$ exponentiations in $\mathbb{G}$.*
- *(Subgroup decision). [19] Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be cyclic groups of prime order p, such that $\mathbb{G}_1$, $\mathbb{G}_2$ are elliptic-curve groups of, there exists a non-degenerate bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_t$, and the subgroup decision assumption holds on $\mathbb{G}_1 \times \mathbb{G}_2$. There exists 2-party multiplicative HSS for degree-3 polynomials over $\mathcal{R} = \mathbb{Z}_p$ with share size $O(k)$ $\mathbb{G}_1$ and $\mathbb{G}_2$ elements and evaluation cost dominated by $O(m)$ pairings and $O(m)$ exponentiations in $\mathbb{G}_t$.*

5.3 Putting Things Together

We now combine the general framework from Theorem 6, together with various sparse PRG construction candidates and the HSS schemes from the full version. We begin in Sect. 5.3.1 by showing how to extend the basic framework from additive HSS to also yield PCG-type constructions from multiplicative HSS. Then in Sect. 5.3.2 we present the overall collection of implied PCG constructions.

5.3.1 Generating Correlations from Multiplicative HSS

We next show how to obtain useful correlations from HSS with *multiplicative* reconstruction. We begin by directly constructing a PCG for the random weight-one vector correlation $\mathcal{C}_{\mathsf{wt1}}(B, \mathcal{R})^n$; namely, secret shares of many independently random weight-1 vectors over $\mathcal{R}$. This correlation has useful applications, such as for compressing the seed size of Distributed Point Functions (see the full version). We then further use this PCG as a stepping stone toward generating UV correlations from multiplicative HSS, with a small amount of communication between parties.

PCG for Weight-1 Vector Correlations. The PCG scheme for $\mathcal{C}_{\mathsf{wt1}}(B, \mathcal{R})^n$ is achieved by two simple steps: First, we observe that directly applying the PCG construction for authenticated unit vector correlations from Proposition 1 yields correlated vectors over $\mathbb{G}$ which differ by $\mathcal{C}_{\mathsf{AUV}}(B, \mathbb{Z}_q, n)$ in the *exponent space* $\mathbb{Z}_q$ of $\mathbb{G}$. In particular, considering just the second part of the correlation, the disagreeing elements each differ by a fixed multiple g^r (here, $r \in \mathbb{Z}_q$ is the authentication scale factor in $\mathcal{C}_{\mathsf{AUV}}$). From here, we make use of correlation-robust hashing from $\mathbb{G}$ into $\mathcal{R}$ in order to destroy this structure, resulting in secret shares over the desired output space $\mathcal{R}$ that differ by 0 in most locations, and by *independent* random values in the others.

PCG Protocol for Unit Vector Correlations. In order to obtain the standard unit vector correlation $\mathcal{C}_{\mathsf{UV}}(B, \mathcal{R})^n$ (which we achieve for $\mathcal{R} = \mathbb{F}$ a finite

field), with additive shares of *1* in the nonzero positions instead of random offsets, our process requires a small amount of communication between parties. As such, it does not constitute a strict PCG, but does achieve a relaxed notion of *PCG protocol*, as considered in [18]. Here, the PCG-type nontriviality is because the communication grows sublinearly in the output correlation size. More specifically, to generate n length-B unit vector instances over $(\mathbb{F}^B)^n$, our protocol will require $O(n \log |\mathbb{F}|)$ bits of communication (sublinear for $B \in \omega(1)$).

Our PCG protocol for $\mathcal{C}_{UV}(B, \mathbb{F})^n$ begins by generating the weight-1 correlation correlation over $\mathbb{F}$ from our PCG above. The goal will be for the parties to identify the random $\mathbb{F}$-payload in each (length-B) weight-1 vector instance, and to divide each element of the vector out by this value (thus the need for field $\mathbb{F}$). This payload can be identified without revealing information on its *location* within the vector, by each party sending the sum of its B corresponding shares. In doing so, this vector instance can be scaled down to shares of a *unit* vector over $\mathbb{F}$, i.e. nonzero payload 1.

However, with probability $1/|\mathbb{F}|$, the random payload of a given "weight-1" vector from the original correlation will be the 0 element of $\mathbb{F}$, in which case this process cannot be executed. For this reason, we boost slightly the original weight-1 vector correlation to give us instances over $\mathbb{F}^\ell$ instead of $\mathbb{F}$, for some amplification parameter ℓ. The parties will parse the resulting $\mathcal{C}_{\mathsf{wt1}}(B, \mathbb{F}^\ell)^n$ as ℓ independent samples of $\mathcal{C}_{\mathsf{wt1}}(B, \mathbb{F})^n$, and perform the above-described procedure on the *first* such sample. Then, for each instance $i \in [n]$ resulting in a 0 sum (i.e., forming a secret sharing of the all-0 vector in $\mathbb{F}^B$), the parties will throw out this ith instance from the present sample, and move to the corresponding ith instance in the next sample $2 \in [\ell]$. This process is repeated until each instance i reaches a satisfying outcome, or all ℓ samples of the ith instance are exhausted (in which case we produce slightly fewer than n instances).

Note a slightly simpler variant of the procedure described above would be to instead begin with a correlation $\mathcal{C}_{\mathsf{wt1}}(B, \mathbb{F})^{n'}$ of $n' > n$ instances over $\mathbb{F}$ (vs $\mathbb{F}^\ell$), and to simply throw out the 0-sum instances, leaving behind n. This process will also work, but will be slightly more expensive. It requires greater stretch from the sparse PRG in order to generate these additional independent instances, whereas the $\mathbb{F}^\ell$ amplification approach simply requires a correlation-robust hash with a larger output space. (Indeed, note that all ℓ samples in a given instance location $i \in [n]$ share the same possibly-nonzero index within $[B]$.)

We conclude with a final remark for why the correlation-robust hashing step is necessary for this PCG protocol transformation. Namely, why must we convert to $\mathcal{C}_{\mathsf{wt1}}(B, \mathbb{F}^\ell)^n$ first, instead of remaining at the correlation step which differs by $\mathcal{C}_{\mathsf{UV}}(B, \mathbb{Z}_q)$ in the exponent space, and, say, performing the "are we 0" interactive checking process to the LSB of the corresponding $\mathbb{G}$ elements (resulting in a correlation $\mathcal{C}_{\mathsf{UV}}(B, \mathbb{Z}_2)^n$ over bits). Although correctness of the resulting output correlation would hold, the issue is that its *security* will not. Unlike in the weight-1 random case, where the other parties' disagreeing value can be an arbitrary

element in the field, here there is little entropy on what is held by the other party. Consider an instance $i \in [n]$, and for simplicity consider the 2-party setting. Then for instance i, a party holds (multiplicative) shares $(y_1, \ldots, y_B) \in \mathbb{G}^B$, such that in each entry $j \in [N]$, the other party holds either the same element y_j, or the element $y_j \cdot g$, offset by one multiplicative copy of the group generator. However, when considering the *additive* structure of this group (as done when comparing (in)equality of bits in group elements), then for some roughly balanced fraction of j it will hold that $y_j \cdot g$ agrees in the least-significant bit representation with y_j. Thus, by identifying a "good" instance (where the sum of our bits is 1, not 0), this in fact rules out half of the positions $j \in [B]$ in which our differing value can lie, *leaking* undesirable information about the resulting unit vector correlation.

The Constructions. We now proceed to the formal constructions. Our constructions make use of the following form of *correlation-robust* hashing. Correlation-robust hash functions were first defined in [59] in the context of achieving concretely efficient OT extension. We modify the standard definition slightly to accommodate multiplicative offsets. Correlation-robust hashing is a clean and relatively standard symmeteric-key cryptographic primitive that can be instantiated by a random oracle.

Definition 8 (Correlation-Robust Hash Function). Let $\mathbb{G}$ be a cyclic group of order $q \in \lambda^{\omega(1)}$ with generator g, and $\mathcal{R}$ a ring. An efficiently computable function $\mathcal{H} : \{0,1\}^\lambda \times \mathbb{G} \to \mathcal{R}$ is *correlation robust* if for any $m = \mathsf{poly}(\lambda)$, the distribution

$$\left\{ \left(t_1, \ldots, t_m, \;\; \mathcal{H}(1, t_1 \cdot g^r), \ldots, \mathcal{H}(m, t_m \cdot g^r) \right) : t_1, \ldots, t_m \leftarrow \mathbb{G}; r \leftarrow \mathbb{Z}_p \right\}$$

is computationally indistinguishable from uniform on $\mathbb{G}^m \times \mathcal{R}^m$.

As mentioned above, our construction in this section for $\mathcal{C}_{\mathsf{UV}}(B, \mathcal{R})^n$ is a *PCG protocol*, a relaxation of PCGs that allows communication between parties that is sublinear in the output correlation size. This is formalized via a standard simulation-based security definition, where the corresponding ideal functionality allows corrupted parties to choose their part of the correlation, and then reverse-samples the outputs of honest parties as per the output correlation. This functionality is a standard relaxation of the most natural ideal functionality candidate, which would simply output an honest sample from the correlation to all parties, and unfortunately is impossible to securely realize in general (see discussion in [19]). The weaker functionality is both realizable and suffices for typical use-cases of incorporating PCGs and PCG protocols within a larger secure protocol. See [19] for a formal definition of the notion of a PCG protocol.

We are now ready to state our main result of this section: constructing PCG for $\mathcal{C}_{\mathsf{wt1}}(B, \mathcal{R})^n$, and PCG protocols for $\mathcal{C}_{\mathsf{UV}}(B, \mathcal{R})^n$, from sparse PRGs and multiplicative HSS.

PCG for Weight-1 Correlations using Multiplicative HSS

$\mathbb{G}$ cyclic group of order q. $(\mathsf{SPRG.Gen}, \mathsf{SPRG.Exp})$ is a $(k, n, B, \mathbb{Z}_q)$ sparse PRG; $(\mathsf{MHSS.Gen}, \mathsf{MHSS.Eval})$ is N-party multiplicative HSS over $\mathbb{Z}_q$; $\mathcal{H} : \{0,1\}^\lambda \times \mathbb{G} \to \mathcal{R}$ is correlation-robust hash function.

- $\mathsf{PCG.Gen}'(1^\lambda)$: Sample seed $x \leftarrow \mathsf{SPRG.Gen}(1^\lambda)$ where $x \in \mathbb{Z}_q^k$, and random $r \leftarrow \mathbb{Z}_q$.
  Sample shares $(s_1, \ldots, s_N) \leftarrow \mathsf{MHSS.Share}(1^\lambda, (x, rx))$, where $(x, rx) \in \mathbb{Z}_q^{2k}$.
  Output $(s_1, \ldots, s_N)$.
- $\mathsf{PCG.Expand}'(i, s_i)$: Compute $y_i = \mathsf{MHSS.Eval}(i, x_i, f(\cdot))$, where $f : \mathbb{Z}_q^k \times Z_q^k \to \mathbb{Z}_q^n$ is the degree-d computation defined as follows. Given $\mathsf{SPRG.Exp}(x) =$

$$\sum_{i_1 \le \cdots \le i_d \le k} \alpha_{i_1,\ldots,i_k} \left(\Pi_{j=1}^d x_{i_j} \right), \text{ then } f(x, r) := \sum_{i_1 \le \cdots \le i_d \le k} \alpha_{i_1,\ldots,i_k} \left(r_{i_1} \Pi_{j=2}^d x_{i_j} \right).$$

  Parse $y_i = (y_{i,1}, \ldots, y_{i,nB}) \in \mathbb{G}^{nB}$. Output $\big(\mathcal{H}(1, y_{i,1}), \ldots, \mathcal{H}(nB, y_{i,nB})\big) \in \mathcal{R}^{nB}$.

Fig. 4. PCG for weight-1 vector correlations $\mathcal{C}_{\mathsf{wt1}}(B, \mathcal{R})^n$ based on sparse PRG, *multiplicative* HSS, and correlation-robust hashing.

Theorem 11 (PCG from Multiplicative HSS). *Let $\mathbb{G}$ be a cyclic group of order q. Assume the existence of:* $(\mathsf{SPRG.Gen}, \mathsf{SPRG.Exp})$ *a degree-d sparse PRG with parameters $(k, n, B, \mathbb{Z}_q)$ (Definition 5);* $(\mathsf{MHSS.Gen}, \mathsf{MHSS.Eval})$ *an N-party, t-secure* multiplicative *HSS scheme for degree-d polynomials over $\mathbb{Z}_q$, with per-party share size s for inputs in $\mathbb{Z}_q^k$, and pseudorandom outputs. Further, assume existence of a correlation-robust hash function $\mathcal{H} : \{0,1\}^\lambda \times \mathbb{G} \to \mathcal{R}$ (Definition 8).*

- *The scheme $\mathsf{PCG}' = (\mathsf{PCG.Gen}', \mathsf{PCG.Expand}')$ from Fig. 4 is a secure PCG for correlation $\mathcal{C}_{\mathsf{wt1}}(B, \mathcal{R})^n$ over $\mathcal{R}$ with seed size $2s$.*
- *Assume $R = \mathbb{F}$ is a finite field, and $\ell \in \mathbb{N}$ be polynomial in λ. The protocol Π_{uv} from Fig. 6 is a secure PCG protocol for correlation $\mathcal{C}_{\mathsf{UV}}(B, \mathbb{F})^n$, with failure probability $|\mathbb{F}|^{-\ell}$ (negligible for $\ell = \omega(\log \lambda)$), and expected $O(n \log |\mathbb{F}|)$ bits of communication.*

Here $\mathcal{C}_{\mathsf{wt1}}(B, \mathcal{R}), \mathcal{C}_{\mathsf{UV}}(B, \mathcal{R})$ are as in Definition 4.

We present our collective resulting PCG constructions in Fig. 5. We include only new asymptotic statements, omitting corollaries from lattice-based assumptions which can be attained through existing methods. We remark that while the PCG *feasibility* results from Decisional Composite Residuosity (DCR) follow from prior work, the asymptotic complexity of PCG expansion in our approach is superior. We additionally note that our constructions from Decisional Diffie-Hellman (DDH), and bilinear group BGN assumptions constitute new feasibility results; indeed, while additive HSS for higher-degree computations (NC^1) exists from these assumptions, the inverse-polynomial correctness error of the corresponding schemes prevents any direct construction.

Local PRG Assumption	Correlation (HSS assump)	Instances	Seed size	Expand time
Binary d-local PRG [56], $d \geq 5$ (for $d = 5, n = k^{1.5-\epsilon}$)	PCG for $\mathcal{C}_{\mathsf{UV}}(B,\mathcal{R})^n$:			
	(IT): $N > dt$	$n = k^{\Omega(d)}$	kB	$\tilde{O}(nB)$
	(OWF): $N = d+1, t = 1$	$n = k^{\Omega(d)}$	$k \log B$	$\tilde{O}(nB)$
	(DCR): $N = 2, \mathcal{R} = \mathbb{Z}_M$	$n = k^{\Omega(d)}$	kB	$\tilde{O}(nB)$
Non-Abelian candidate [7], $d \geq 3$ /or/ Sparse LWR (Conj 3), $d \geq 3$ (for $B = \omega(\log \lambda)$)	PCG for $\mathcal{C}_{\mathsf{UV}}(B,\mathcal{R})^n$:			
	(IT): $N > dt$	$n = k^{d/2-\epsilon}$	kB	$nB^2/\tilde{O}(nB)$
	(OWF): $N = d+1, t = 1$	$n = k^{d/2-\epsilon}$	$k \log B$	$nB^2/\tilde{O}(nB)$
	(DCR): $N = 2, \mathcal{R} = \mathbb{Z}_M$	$n = k^{d/2-\epsilon}$	kB	$nB^2/\tilde{O}(nB)$
	PCG for $\mathcal{C}_{\mathsf{wt1}}(B,\mathbb{Z}_q)^n$:			
	(BGN,CRF): $N = 2$	$n = k^{d/2-\epsilon}$	kB	$nB^2/\tilde{O}(nB)$
New 2-local candidate (Conj 4)	PCG for $\mathcal{C}_{\mathsf{UV}}(B,\mathcal{R})^n$:			
	(IT): $N > 2t$	$n = k^{1+o(1)}$	kB	$\tilde{O}(nB)$
	(OWF): $N = 3, t = 1$	$n = k^{1+o(1)}$	$k \log B$	$\tilde{O}(nB)$
	(DCR): $N = 2, \mathcal{R} = \mathbb{Z}_M$	$n = k^{1+o(1)}$	kB	$\tilde{O}(nB)$
	(QR): $N = 2, \mathcal{R} = \mathbb{Z}_2$	$n = k^{1+o(1)}$	kB	$\tilde{O}(nB)$
	PCG for $\mathcal{C}_{\mathsf{wt1}}(B,\mathbb{Z}_q)^n$:			
	(DDH,CRF): $N = 2$	$n = k^{1+o(1)}$	kB	$\tilde{O}(nB)$

Fig. 5. PCG corollaries given local PRGs and HSS. Here, k denotes PRG seed length, d the PRG locality, N denotes the number of parties, t denotes the corruption threshold. PRG seed sizes are based on existing and new conjectures (Sects. 4.2.2 and 4.2.3). Factors of $\mathsf{poly}(\lambda)$ are ignored in quoted asymptotics, omitting O notation. CRF denotes correlation-robust functions. Wherever $\mathcal{C}_{\mathsf{UV}}(B,\mathcal{R})^n$ is obtained, we can also generate the authenticated correlation $\mathcal{C}_{\mathsf{AUV}}(B,\mathcal{R},n)$ (and $\mathcal{C}_{\mathsf{wt1}}(B,\mathcal{R})^n$, additionally assuming CRF). Wherever $\mathcal{C}_{\mathsf{wt1}}(B,\mathcal{R})^n$ is obtained, we also achieve an interactive PCG protocol for $\mathcal{C}_{\mathsf{UV}}(B,\mathcal{R})^n$.

PCG Protocol Π_{uv} for Unit Vector Correlations

N parties $P_1, \ldots, P_N$ have no inputs. Each P_i outputs $r^{(i)} \in (\mathbb{F}^B)^n$ as part of target correlation $\mathcal{C}_{UV}(B,\mathbb{F})^n$. Let $\ell \in \mathbb{N}$, $(\mathsf{PCG.Gen}, \mathsf{PCG.Expand})$ is N-party PCG for correlation $\mathcal{C}_{\mathsf{wt1}}(B,\mathbb{F}^\ell)^n$.

1. The parties jointly execute a secure N-party computation protocol for sampling $(\mathsf{k}_1, \ldots, \mathsf{k}_N) \leftarrow \mathsf{PCG.Gen}(1^\lambda)$.
2. Each party $i \in [N]$ locally expands $R^{(i)} \leftarrow \mathsf{PCG.Expand}(i, \mathsf{k}_i)$, where we will index $\mathbb{F}$-elements in $R^{(i)} \in ((\mathbb{F}^\ell)^B)^n$ as $R^{(i)}_{j,k}[\mathsf{ctr}]$ for $j \in [n], k \in [B], \mathsf{ctr} \in [\ell]$.
3. The parties initialize $\mathsf{done} = \emptyset \subset [n]$ and execute the following for $\mathsf{ctr} = 1$ to ℓ: For each $j \in ([n] \setminus \mathsf{done})$:
   (a) Each party P_i sends the sum $\sigma^{(i)}_j[\mathsf{ctr}] = \sum_{k=1}^{B} R^{(i)}_{j,k}[\mathsf{ctr}] \in \mathbb{F}$.
   (b) Each party P_i computes the sum $\sigma_j[\mathsf{ctr}] = \sum_{i=1}^{N} \sigma^{(i)}_j[\mathsf{ctr}]$ of all parties' values.
   (c) If $y_j[\mathsf{ctr}] \neq 0 \in \mathbb{F}$, then: (i) set $\mathsf{done} \leftarrow \mathsf{done} \cup \{j\}$,
   (ii) $\forall k \in [B]$, set $r^{(i)}_{j,k} = R^{(i)}_{j,k}[\mathsf{ctr}] \cdot (\sigma_j[\mathsf{ctr}])^{-1}$).
4. Each party P_i outputs $(r^{(i)}_{j,k})_{j \in [n], k \in [B]} \in (\{0,1\}^B)^n$.

Fig. 6. Interactive PCG protocol for UV correlations given weight-1 vector correlations.

Acknowledgements. We thank Benny Applebaum, Andrej Bogdanov, Geoffroy Couteau, Itai Dinur, Aayush Jain and Vinod Vaikuntanathan for helpful discussions and pointers. E. Boyle was supported in part by AFOSR Award FA9550-21-1-0046 and ERC Project HSS (852952). Y. Ishai was supported by ERC grant NTSC (742754), BSF grant 2022370, ISF grant 2774/20, and ISF-NSFC grant 3127/23. Y. Ma was supported by a Microsoft Research PhD Fellowship.

References

1. Abram, D., Scholl, P.: Low-communication multiparty triple generation for SPDZ from ring-LPN. In: The International Conference on Practice and Theory in Public Key Cryptography (PKC) (2022)
2. Addanki, S., Garbe, K., Jaffe, E., Ostrovsky, R., Polychroniadou, A.: Prio+: privacy preserving aggregate statistics via boolean shares. In: Galdi, C., Jarecki, S. (eds.) SCN 2022. LNCS, vol. 13409, pp. 516–539. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-14791-3_23
3. Applebaum, B.: Cryptographic hardness of random local functions - survey. Comput. Complex. **25**, 667–722 (2016)
4. Applebaum, B., Ishai, Y., Kushilevitz, E.: On pseudorandom generators with linear stretch in nc^0. Comput. Complex. **17**(1), 38–69 (2008). https://doi.org/10.1007/S00037-007-0237-6
5. Applebaum, B., Lovett, S.: Algebraic attacks against random local functions and their countermeasures. In: Proceedings of the ACM Symposium on Theory of Computing (STOC) (2016)
6. Banerjee, A., Peikert, C., Rosen, A.: Pseudorandom functions and lattices. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 719–737. Springer, Cham (2012). https://doi.org/10.1007/978-3-642-29011-4_42
7. Barak, B., Brakerski, Z., Komargodski, I., Kothari, P.K.: Limits on low-degree pseudorandom generators (or: Sum-of-squares meets program obfuscation). In: Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT) (2017)
8. Beaver, D., Feigenbaum, J., Kilian, J., Rogaway, P.: Security with low communication overhead. In: Proceedings of the International Cryptology Conference (CRYPTO) (1991)
9. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: Proceedings of the ACM Symposium on Theory of Computing (STOC) (1988)
10. Bendlin, R., Damgård, I., Orlandi, C., Zakarias, S.: Semi-homomorphic encryption and multiparty computation. In: Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT) (2011)
11. Benhamouda, F., Boyle, E., Gilboa, N., Halevi, S., Ishai, Y., Nof, A.: Generalized pseudorandom secret sharing and efficient straggler-resilient secure computation. In: Proceedings of the Theory of Cryptography Conference (TCC) (2021)
12. Bombar, M., Bui, D., Couteau, G., Couvreur, A., Ducros, C., Servan-Schreiber, S.: FOLEAGE: OLE-based multi-party computation for boolean circuits. IACR Cryptol. ePrint Arch. 429 (2024). https://eprint.iacr.org/2024/429
13. Bombar, M., Couteau, G., Couvreur, A., Ducros, C.: Correlated pseudorandomness from the hardness of quasi-abelian decoding. In: Proceedings of the International Cryptology Conference (CRYPTO) (2023)

14. Boneh, D., Gentry, C., Halevi, S., Wang, F., Wu, D.J.: Private database queries using somewhat homomorphic encryption. In: Jacobson, M., Locasto, M., Mohassel, P., Safavi-Naini, R. (eds.) ACNS 2013. LNCS, vol. 7954, pp. 102–118. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38980-1_7
15. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y.: Compressing vector OLE. In: Proceedings of the ACM Conference on Computer and Communications Security (CCS) (2018)
16. Boyle, E., et al.: Correlated pseudorandomness from expand-accumulate codes. In: Proceedings of the International Cryptology Conference (CRYPTO) (2022)
17. Boyle, E., et al.: Oblivious transfer with constant computational overhead. In: Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT) (2023)
18. Boyle, E., et al.: Efficient two-round OT extension and silent non-interactive secure computation. In: Proceedings of the ACM Conference on Computer and Communications Security (CCS) (2019)
19. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y., Kohl, L., Scholl, P.: Efficient pseudorandom correlation generators: silent OT extension and more. In: Proceedings of the International Cryptology Conference (CRYPTO) (2019)
20. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y., Kohl, L., Scholl, P.: Correlated pseudorandom functions from variable-density LPN. In: Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS) (2020)
21. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y., Orrù, M.: Homomorphic secret sharing: optimizations and applications. In: Proceedings of the ACM Conference on Computer and Communications Security (CCS) (2017)
22. Boyle, E., Couteau, G., Meyer, P.: Sublinear secure computation from new assumptions. In: Proceedings of the Theory of Cryptography Conference (TCC) (2022)
23. Boyle, E., Couteau, G., Meyer, P.: Sublinear-communication secure multiparty computation does not require FHE. In: Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT) (2023)
24. Boyle, E., Gilboa, N., Ishai, Y.: Function secret sharing. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 337–367. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46803-6_12
25. Boyle, E., Gilboa, N., Ishai, Y.: Breaking the circuit size barrier for secure computation under DDH. In: Proceedings of the International Cryptology Conference (CRYPTO) (2016)
26. Boyle, E., Gilboa, N., Ishai, Y.: Function secret sharing: improvements and extensions. In: Proceedings of the ACM Conference on Computer and Communications Security (CCS) (2016)
27. Boyle, E., Gilboa, N., Ishai, Y.: Secure computation with preprocessing via function secret sharing. In: Proceedings of the Theory of Cryptography Conference (TCC) (2019)
28. Boyle, E., Kohl, L., Scholl, P.: Homomorphic secret sharing from lattices without FHE. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019. LNCS, vol. 11477, pp. 3–33. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17656-3_1
29. Boyle, E., LaVigne, R.: Personal communication (2023)
30. Brüggemann, A., Hundt, R., Schneider, T., Suresh, A., Yalame, H.: FLUTE: fast and secure lookup table evaluations. In: In Proceedings of the IEEE Symposium on Security and Privacy (S&P) (2023)

31. Bui, D., Couteau, G., Meyer, P., Passelègue, A., Riahinia, M.: Fast public-key silent OT and more from constrained naor-reingold. In: Joye, M., Leander, G. (eds.) EUROCRYPT 2024, Part VI. LNCS, vol. 14656, pp. 88–118. Springer, Cham (2024). https://doi.org/10.1007/978-3-031-58751-1_4
32. Bunn, P., Kushilevitz, E., Ostrovsky, R.: CNF-FSS and its applications. In: IACR International Conference on Public-Key Cryptography, pp. 283–314 (2022)
33. Chaum, D., Crépeau, C., Damgård, I.: Multiparty unconditionally secure protocols (extended abstract). In: Proceedings of the ACM Symposium on Theory of Computing (STOC) (1988)
34. Chor, B., Goldreich, O., Kushilevitz, E., Sudan, M.: Private information retrieval. In: Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS) (1995)
35. Corrigan-Gibbs, H., Boneh, D.: Prio: private, robust, and scalable computation of aggregate statistics. In: 14th USENIX symposium on networked systems design and implementation (NSDI 2017), pp. 259–282 (2017)
36. Couteau, G.: A note on the communication complexity of multiparty computation in the correlated randomness model. In: Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT) (2019)
37. Couteau, G., Ducros, C.: Pseudorandom correlation functions from variable-density LPN, revisited. In: Boldyreva, A., Kolesnikov, V. (eds.) PKC 2023, Part II. LNCS, vol. 13941, pp. 221–250. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-31371-4_8
38. Couteau, G., Dupin, A., Méaux, P., Rossi, M., Rotella, Y.: On the concrete security of Goldreich's pseudorandom generator. In: International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT) (2018)
39. Couteau, G., Meyer, P.: Breaking the circuit size barrier for secure computation under quasi-polynomial LPN. In: Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT) (2021)
40. Cramer, R., Damgård, I., Ishai, Y.: Share conversion, pseudorandom secret-sharing and applications to secure computation. In: Proceedings of the Theory of Cryptography Conference (TCC) (2005)
41. Cramer, R., Fehr, S., Ishai, Y., Kushilevitz, E.: Efficient multi-party computation over rings. In: Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT) (2003)
42. Damgård, I., Nielsen, J.B., Nielsen, M., Ranellucci, S.: The tinytable protocol for 2-party secure computation, or: gate-scrambling revisited. In: Proceedings of the International Cryptology Conference (CRYPTO) (2017)
43. Damgård, I., Pastro, V., Smart, N.P., Zakarias, S.: Multiparty computation from somewhat homomorphic encryption. In: Proceedings of the International Cryptology Conference (CRYPTO) (2012)
44. Dao, Q., Ishai, Y., Jain, A., Lin, H.: Multi-party homomorphic secret sharing and sublinear MPC from sparse LPN. In: Proceedings of the International Cryptology Conference (CRYPTO) (2023)
45. Dessouky, G., Koushanfar, F., Sadeghi, A.R., Schneider, T., Zeitouni, S., Zohner, M.: Pushing the communication barrier in secure computation using lookup tables. In: Proceedings of the Network and Distributed System Security Symposium (NDSS) (2017)

46. Devadas, L., Quach, W., Vaikuntanathan, V., Wee, H., Wichs, D.: Succinct LWE sampling, random polynomials, and obfuscation. In: Nissim, K., Waters, B. (eds.) TCC 2021, Part II. LNCS, vol. 13043, pp. 256–287. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-90453-1_9
47. Diaconis, P., Rockmore, D.: Efficient computation of the fourier transform on finite groups. J. Am. Math. Soc. **3**(2), 297–332 (1990)
48. Döttling, N., Garg, S., Malavolta, G.: Personal communication (2023)
49. Dwork, C., Kenthapadi, K., McSherry, F., Mironov, I., Naor, M.: Our data, ourselves: privacy via distributed noise generation. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 486–503. Springer, Heidelberg (2006). https://doi.org/10.1007/11761679_29
50. Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating noise to sensitivity in private data analysis. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 265–284. Springer, Heidelberg (2006). https://doi.org/10.1007/11681878_14
51. Eriguchi, R., Ichikawa, A., Kunihiro, N., Nuida, K.: Efficient noise generation to achieve differential privacy with applications to secure multiparty computation. In: Borisov, N., Diaz, C. (eds.) FC 2021. LNCS, vol. 12674, pp. 271–290. Springer, Cham (2021). https://doi.org/10.1007/978-3-662-64322-8_13
52. Escudero, D., Ghosh, S., Keller, M., Rachuri, R., Scholl, P.: Improved primitives for MPC over mixed arithmetic-binary circuits. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part II. LNCS, vol. 12171, pp. 823–852. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-56880-1_29
53. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of the ACM Symposium on Theory of Computing (STOC) (2009)
54. Geoffroy, C., Rindal, P., Raghuraman, S.: Silver: silent VOLE and oblivious transfer from hardness of decoding structured LDPC codes. In: Proceedings of the International Cryptology Conference (CRYPTO) (2021)
55. Gilboa, N., Ishai, Y.: Distributed point functions and their applications. In: Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT) (2014)
56. Goldreich, O.: Candidate one-way functions based on expander graphs. In: Studies in Complexity and Cryptography (2011)
57. Gueron, S., Kounavis, M.E.: Intel carry-less multiplication instruction and its usage for computing the GCM mode. https://www.intel.com/content/dam/develop/external/us/en/documents/clmul-wp-rev-2-02-2014-04-20.pdf
58. Ishai, Y., Kelkar, M., Narayanan, V., Zafar, L.: One-message secure reductions: on the cost of converting correlations. In: Proceedings of the International Cryptology Conference (CRYPTO) (2023)
59. Ishai, Y., Kilian, J., Nissim, K., Petrank, E.: Extending oblivious transfers efficiently. In: Proceedings of the International Cryptology Conference (CRYPTO) (2003)
60. Ishai, Y., Kushilevitz, E., Meldgaard, S., Orlandi, C., Paskin-Cherniavsky, A.: On the power of correlated randomness in secure computation. In: Proceedings of the Theory of Cryptography Conference (TCC) (2013)
61. Ishai, Y., Lai, R.W.F., Malavolta, G.: A geometric approach to homomorphic secret sharing. In: The International Conference on Practice and Theory in Public Key Cryptography (PKC) (2021)
62. Ito, M., Saito, A., Nishizeki, T.: Secret sharing schemes realizing general access structure. In: IEEE Global Telecommunication Conference (1987)

63. Jain, A., Lin, H., Sahai, A.: Indistinguishability obfuscation from well-founded assumptions. In: Proceedings of the ACM Symposium on Theory of Computing (STOC) (2021)
64. Keller, M., Orsini, E., Rotaru, D., Scholl, P., Soria-Vazquez, E., Vivek, S.: Faster secure multi-party computation of AES and DES using lookup tables. In: Proceedings of the International Conference on Applied Cryptography and Network Security (ACNS) (2017)
65. Lai, R.W.F., Malavolta, G., Schröder, D.: Homomorphic secret sharing for low degree polynomials. In: International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT) (2018)
66. Lin, H., Tessaro, S.: Indistinguishability obfuscation from trilinear maps and block-wise local PRGs. In: Proceedings of the International Cryptology Conference (CRYPTO) (2017)
67. Lombardi, A., Vaikuntanathan, V.: Limits on the locality of pseudorandom generators and applications to indistinguishability obfuscation. In: Proceedings of the Theory of Cryptography Conference (TCC) (2017)
68. Mossel, E., Shpilka, A., Trevisan, L.: On epsilon-biased generators in NC^0. In: Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS) (2003)
69. Orlandi, C., Scholl, P., Yakoubov, S.: The rise of paillier: homomorphic secret sharing and public-key silent OT. In: Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT) (2021)
70. Ostrovsky, R., Shoup, V.: Private information storage (extended abstract). In: STOC 1997, pp. 294–303 (1997)
71. Patra, A., Schneider, T., Suresh, A., Yalame, H.: ABY2.0: improved mixed-protocol secure two-party computation. In: Proceedings of the USENIX Security Symposium (2021)
72. Raghuraman, S., Rindal, P., Tanguy, T.: Expand-convolute codes for pseudorandom correlation generators from LPN. In: Proceedings of the International Cryptology Conference (CRYPTO) (2023)
73. Rotaru, D., Wood, T.: Marbled circuits: mixing arithmetic and boolean circuits with active security. In: Hao, F., Ruj, S., Sen Gupta, S. (eds.) INDOCRYPT 2019. LNCS, vol. 11898, pp. 227–249. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-35423-7_12
74. Roy, L., Singh, J.: Large message homomorphic secret sharing from DCR and applications. In: Proceedings of the International Cryptology Conference (CRYPTO) (2021)
75. Schoppmann, P., Gascón, A., Reichert, L., Raykova, M.: Distributed vector-ole: improved constructions and implementation. In: Proceedings of the ACM Conference on Computer and Communications Security (CCS) (2019)
76. Unal, A.: New baselines for local pseudorandom number generators by field extensions (2023). https://eprint.iacr.org/2023/550
77. Ünal, A.: Worst-case subexponential attacks on PRGs of constant degree or constant locality. In: Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT) (2023)

78. Yang, J., Guo, Q., Johansson, T., Lentmaier, M.: Revisiting the concrete security of Goldreich's pseudorandom generator. IEEE Trans. Inf. Theory **68**(2), 1329–1354 (2021)
79. Yang, K., Weng, C., Lan, X., Zhang, J., Wang, X.: Ferret: fast extension for correlated OT with small communication. In: Proceedings of the ACM Conference on Computer and Communications Security (CCS) (2020)
80. Zichron, L.: Locally computable arithmetic pseudorandom generators (2017)