Efficient Pseudorandom Correlation Generators: Silent OT Extension and More

Elette Boyle[1], Geoffroy Couteau[2], Niv Gilboa[3], Yuval Ishai[4], Lisa Kohl[2], and Peter Scholl[5]

[1] IDC Herzliya
[2] Karlsruhe Institute of Technology
[3] Ben-Gurion University of the Negev
[4] Technion
[5] Aarhus University

Abstract. Secure multiparty computation (MPC) often relies on correlated randomness for better efficiency and simplicity. This is particularly useful for MPC with no honest majority, where input-independent correlated randomness enables a lightweight "non-cryptographic" online phase once the inputs are known. However, since the amount of randomness typically scales with the circuit size of the function being computed, securely generating correlated randomness forms an efficiency bottleneck, involving a large amount of communication and storage.

A natural tool for addressing the above limitations is a *pseudorandom correlation generator* (PCG). A PCG allows two or more parties to securely generate long sources of useful correlated randomness via a local expansion of correlated short seeds and no interaction. PCGs enable MPC with *silent preprocessing*, where a small amount of interaction used for securely sampling the seeds is followed by silent local generation of correlated pseudorandomness.

A concretely efficient PCG for Vector-OLE correlations was recently obtained by Boyle et al. (CCS 2018) based on variants of the learning parity with noise (LPN) assumption over large fields. In this work, we initiate a systematic study of PCGs and present concretely efficient constructions for several types of useful MPC correlations. We obtain the following main contributions:

- **PCG foundations.** We give a general security definition for PCGs. Our definition suffices for any MPC protocol satisfying a stronger security requirement that is met by existing protocols. We prove that a stronger security requirement is indeed necessary, and justify our PCG definition by ruling out a stronger and more natural definition.
- **Silent OT extension.** We present the first concretely efficient PCG for oblivious transfer correlations. Its security is based on a variant of the binary LPN assumption and any correlation-robust hash function. We expect it to provide a faster alternative to the IKNP OT extension protocol (Crypto 2003) when communication is the bottleneck. We present several applications, including protocols for non-interactive zero-knowledge with bounded-reusable preprocessing from binary LPN, and concretely efficient related-key oblivious pseudorandom functions.

- **PCGs for simple 2-party correlations.** We obtain PCGs for several other types of useful 2-party correlations, including (authenticated) one-time truth-tables and Beaver triples. While the latter PCGs are slower than our PCG for OT, they are still practically feasible. These PCGs are based on a host of assumptions and techniques, including specialized homomorphic secret sharing schemes and pseudorandom generators tailored to their structure.
- **Multiparty correlations.** We obtain PCGs for multiparty correlations that can be used to make the (input-dependent) online communication of MPC protocols scale *linearly* with the number of parties, instead of quadratically.

1 Introduction

Correlated secret randomness is a valuable resource for secure multi-party computation (MPC). A simple example is a common random key that is given to two parties, who can later use it as a one-time pad for secure message transmission. In the context of MPC, a more useful example is a random *oblivious transfer* (OT) correlation, in which one party is given a pair of random bits (more generally, strings) (s_0, s_1) and the other party is given the pair (r, s_r) for a random bit r. The OT correlation can serve as a basis for general MPC protocols with no honest majority [40, 54, 49]. Other kinds of two-party correlations that are useful for MPC include *oblivious linear-function evaluation* (OLE) correlations [58, 50, 3], *multiplication triples* (also known as "Beaver triples") [8, 10, 29], and *one-time truth tables* [47, 28, 30].

The above types of correlated randomness are commonly used to implement efficient MPC protocols in the *preprocessing model*. Such protocols consist of an offline, input-independent *preprocessing phase*, where many independent instances of the correlated randomness are generated, followed by a fast *online phase* that consumes this correlated randomness for the purpose of securely evaluate a given function of the inputs. In many cases, the online phase is "information-theoretic"[6] and its computational complexity is only a small-constant times higher than that of an insecure function evaluation. Most importantly for the present work, the online phase of such protocols typically outperforms all competing approaches in terms of concrete efficiency.

A major challenge in implementing such offline-online protocols is that the preprocessing phase needs to *securely* generate and store a large amount of correlated randomness. This is typically done by using a special-purpose interactive MPC protocol, which involves a significant amount of communication and computation for each gate of a circuit that should be evaluated in the online phase. A dream goal would be to replace this source of correlated randomness with *short* correlated seeds, which can be "silently" expanded *without any interaction* to produce a large amount of *pseudorandom* correlated randomness. This

[6] This can be formalized by requiring that the joint states of the parties in the end of the offline phase can be swapped by *computationally indistinguishable* states, given which the online protocol is secure against computationally unbounded parties.

process should emulate an ideal process for generating the target distribution not only from the point of view of outsiders, but also from the point of view of *insiders* who can observe the correlated seeds. We refer to such an object as a *pseudorandom correlation generator*, or PCG for short.

A bit more precisely, a two-party PCG is defined as follows. Let (R_0, R_1) be a target correlation, defined by some efficient sampling algorithm $\mathcal{C}$ that on input 1^λ outputs a pair of correlated strings (r_0, r_1). For instance, $\mathcal{C}(1^\lambda)$ may output $n = \lambda^3$ independent instances of an OT correlation. A PCG is a pair of efficient algorithms $(\mathsf{Gen}, \mathsf{Expand})$ such that:

- Gen samples a pair of short correlated seeds $(\mathsf{k}_0, \mathsf{k}_1) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$,
- Expand is a *local* deterministic seed expansion algorithm mapping k_i to $r_i \leftarrow \mathsf{Expand}(i, \mathsf{k}_i)$, where $|r_i| > |\mathsf{k}_i|$.

We would like the outputs (r_0, r_1) resulting from this process to be "indistinguishable" from an ideal sample (R_0, R_1) generated by $\mathcal{C}(1^\lambda)$, even to a party who receives one of the seeds k_b.

A useful special case of PCG was recently considered by Boyle et al. [14], who constructed (under variants of the Learning Parity with Noise assumption [12]) a concretely efficient PCG for the *vector OLE* (VOLE) correlation. The VOLE correlation over a field $\mathbb{F}$ samples a random scalar $x \in \mathbb{F}$ and vectors $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{F}^n$, and outputs $r_0 = (\boldsymbol{u}, \boldsymbol{v})$ to one party (the "sender") and $r_1 = (x, \boldsymbol{w} = \boldsymbol{u}x + \boldsymbol{v})$ to the other party (the "receiver"). The VOLE correlation is useful for secure computation of functions that employ scalar-vector products over large fields, such as ones arising in the context of linear algebra and keyword search [3].

Designing efficient PCGs for a wider class of correlations is strongly motivated by the goal of improving the efficiency of *general* MPC in the preprocessing model, where the preprocessing phase is used to securely generate the PCG seeds. We refer to this as *MPC with silent preprocessing*. More concretely, such a protocol consists of three phases: (1) an interactive *setup phase* for securely distributing the seed generation algorithm Gen; in the end of this phase, which involves a small amount of communication, only the short seeds are stored for later use; (2) a silent *seed expansion* phase, where the seeds are expanded into long correlated randomness via a local computation of Expand and without any interaction; (3) a final *online phase* where the correlated randomness is consumed to evaluate a given function of the inputs. One could employ Phase 1 when deciding that an MPC interaction *might* take place in the future, Phase 2 when interaction seems likely to take place in the near future, and Phase 3 to carry out the MPC interaction once the inputs are available. The low communication footprint of silent preprocessing can eliminate traffic analysis attacks that aim to anticipate future MPC plans. Finally, another benefit of the PCG-based approach is that it can help reduce the cost of protecting MPC protocols against malicious parties. Indeed, since Phase 2 does not involve any interaction, it suffices to protect Phase 1 and Phase 3 against malicious parties, which is typically much cheaper.

Several different kinds of PCG constructions are implicit in the MPC literature. These include PCGs for simple multi-party linear correlations from any pseudorandom generator [38, 26], for general correlations from indistinguishability obfuscation [45, 42], for so-called "bilinear" correlations from homomorphic secret sharing [16], for restricted variants of OT correlations from key-homomorphic pseudorandom functions [60] and, most recently, for VOLE correlation from LPN [14]. With the exception of linear multi-party correlations [38, 26] and VOLE correlations [14], none of these prior constructions seem appealing from a practical point of view. In particular, there was no prior approach (even a heuristic one) for constructing a concretely efficient PCG for OT correlations.

1.1 Our Contributions

In this work, we initiate a more systematic study of pseudorandom correlation generators. Our contributions are on both the foundational side, where we present new definitions, impossibility results and connections with other primitives, and the applied side, with concretely efficient constructions for commonly used MPC correlations, including OT correlations and others. Our most practical PCG constructions handle restricted (yet still useful) classes of correlations, while our more general constructions can handle much larger classes of correlations, at the expense of a bigger seed size and higher computational costs (and, for some of them, public-key-style assumptions such as lattice-based or pairing-based cryptography).

We now give a more detailed account of our contributions. Unless noted otherwise, we refer to MPC with computational security against semi-honest (i.e., passive) and static (i.e., non-adaptive) adversaries who may corrupt an arbitrary subset of parties.

Foundations of Pseudorandom Correlation Generators. Our first goal is to present a general security definition for the intuitive notion of PCG described above. As pointed out in [38], this is not quite as straightforward as one might imagine, and previous works side-stepped the problem by taking an ad-hoc approach. To motivate our general definition, we start by discussing the most natural alternative.

Ruling Out a Simulation-Based Definition. Recall that the ultimate desire would be that in any protocol, one can securely replace an ideal correlated randomness functionality $\mathcal{C}$ with pseudo-randomness obtained from expanding the correlated seeds of a PCG for $\mathcal{C}$. This would indeed follow from a natural simulation-based security definition for PCG as a computationally secure, dealer-assisted protocol for computing the randomized functionality defined by $\mathcal{C}$. Concretely, in the two-party case, the simulation-based definition requires the existence of a simulator S such that the *real* distribution $(\mathsf{k}_b, \mathsf{Expand}(\mathsf{k}_{1-b}))$, where $(\mathsf{k}_0, \mathsf{k}_1)$ are generated by Gen (capturing the view of a corrupted party b jointly with the output of the uncorrupted party $1-b$) is computationally indistinguishable from the *ideal* distribution $(S(r_b), r_{1-b})$, where (r_0, r_1) are sampled

by $\mathcal{C}$. Unfortunately, we show (building on [45], and extending an informal argument from [38]) that such a definition is impossible to realize even for simple correlations. Intuitively, the impossibility follows from the fact that in the real distribution k_b "explains" the output of the honest party in an efficiently verifiable way, whereas such an explanation of r_{1-b} cannot be generated from r_b in the ideal distribution.

A General PCG Definition. To get around the above impossibility, we present a relaxed indistinguishability-based definition of PCG security, generalizing the specialized security definition for the VOLE correlation from [14]. Our definition requires that given its PCG key k_b, corrupted party b cannot distinguish the *true* expanded output of the honest party $r_{1-b} = \mathsf{Expand}(1-b, \mathsf{k}_{1-b})$ from a *random* output r_{1-b} consistent with the correlation $\mathcal{C}$ and its own expanded output $r_b = \mathsf{Expand}(b, \mathsf{k}_b)$. In other words, we replace the ideal distribution in the above simulation-based definition by $(\mathsf{k}_b, [r_{1-b} \mid R_b = \mathsf{Expand}(\mathsf{k}_b)])$. Note that the latter distribution involves reverse-sampling from R_{1-b} conditioned on a fixed value for R_b, which may not be well-defined. However, in this work we only consider *additive* correlations, where (R_0, R_1) are additive secret shares (over a finite Abelian group) of a sample from some core distribution. For such additive correlations, the reverse-sampling is well-defined and is computationally efficient. More broadly, our general PCG definition is meaningful when this reverse-sampling is efficient.

Limitations. Our PCG definition is not good enough for generating correlated randomness in *all* applications. Indeed, the impossibility of the simulation-based definition discussed above implies such simple counterexamples for randomized functionalities. Concretely, for any $\mathcal{C}$ to which the impossibility result applies, there is a trivial MPC protocol for $\mathcal{C}$ given correlated randomness from $\mathcal{C}$ in which each party outputs its correlated randomness. However, the impossibility result shows that using *any* PCG for $\mathcal{C}$ would render this simple protocol insecure. We show, under standard cryptographic assumptions, that a similar impossibility holds even if one restricts attention to MPC for *deterministic* functionalities. Concretely, we show a protocol which uses correlated randomness $\mathcal{C}$ to realize a deterministic functionality with statistical security against malicious parties, but which becomes completely *insecure* (even against semi-honest parties) when $\mathcal{C}$ is replaced by a specific PCG for $\mathcal{C}$ that meets our indistinguishability-based definition.

A Plug-and-Play Use of PCG. We complement the above negative results by a positive result, showing that our PCG definition does suffice to imply our "ultimate desire" in the context of most applications. Concretely, we put forward a slightly stronger security requirement for MPC with preprocessing, such that in any protocol satisfying this requirement, a PCG can be used as a drop-in replacement for correlated randomness. The stronger security requirement asserts that security still hold even if the ideal correlation functionality (R_0, R_1) is replaced by a *corruptible* functionality that allows corrupted party b to pick its own

randomness r_b^*, and then delivers to the uncorrupted party a sample r_{1-b} from the conditional distribution $[r_{1-b} \mid R_b = r_b^*]$. It fortunately turns out that natural MPC protocols in the preprocessing model already satisfy this stronger security requirement. This allows for a plug-and-play use of PCGs in many application scenarios.

Relation with Homomorphic Secret Sharing. A (two-party) homomorphic secret sharing (HSS) scheme [18, 21] for a function class $\mathcal{F}$ splits a secret x into two shares (x_0, x_1), such that given any $f \in \mathcal{F}$ one can efficiently evaluate *additive* shares of $f(x)$ via local computation on the shares. We show a two-way relation between PCG and HSS. First, we show that a PCG for any *additive* correlation (as defined above) can be reduced to HSS for a related function class $\mathcal{F}$, generalizing and formalizing a previous observation from [16]. In particular, HSS for general circuits implies PCG for all additive correlations, which include most of the useful MPC correlations as special cases. (This is only a feasibility result, which does not directly imply concretely efficient constructions.) Second, we show that some converse is also true: a PCG for the degree-d "tensoring" correlation, obtained by picking a random vector $X \in \mathcal{R}^n$ and outputting additive shares of all products of at most d entries of X, implies HSS for the class $\mathcal{F}$ of degree-d (n-variate) polynomials over $\mathcal{R}^n$, where the share size grows linearly with n and the homomorphic evaluation time grows linearly with n^d.

Silent OT Extension. A central contribution of this work is the first *concretely efficient* construction of PCG for the oblivious transfer (OT) correlation. From an asymptotic point of view, our PCG can achieve an arbitrary polynomial stretch, assuming: (1) The *binary* Learning Parity with Noise (LPN) assumption [12] with a conservative choice of parameters, and (2) A correlation-robust hash function [46]. The hash function primitive, which is only used in a black-box way, can be instantiated in practice by a general-purpose hash function or block cipher. Assuming LPN with a linear number of samples and inverse-polynomial noise rate holds for the dual of a near-linear time encodable code (such as the codes proposed in [44, 34, 1, 3]), which is still a conservative assumption, the *computational* complexity of Expand is nearly linear in the output length.[7]

In a nutshell, our efficient PCG for OT applies the PCG for VOLE from [14] over a large extension field $\mathbb{F}_{2^\lambda}$, except for restricting the sender's output $\boldsymbol{u}$ to be over the base field. This yields n correlated instances of random OT that can be converted into standard OT by using a correlation-robust hash function, as in [46]. See Section 2 for more details.

By applying a secure two-party protocol for distributing Gen, we obtain a *silent OT extension* protocol that generates n pseudo-random OT instances using a small number of OTs, with a total of $O(n^\epsilon)$ bits of communication for any $\epsilon > 0$. This should be compared with existing OT extension protocols [9, 46] that

[7] In Section 1.1 below we describe an alternative LPN-based approach to constructing PCG for OT that dispenses with assumption (2), but requires at least quadratic computation in the output length n.

do not require the LPN assumption but where the communication complexity is bigger than n.

Concrete Efficiency. Our LPN-based PCG for OT is very attractive in terms of concrete efficiency, and we expect it to outperform state-of-the-art OT extension protocols [46, 7, 55] in settings where communication is the bottleneck. To give a few data points, our PCG can expand a pair of seeds of length 10KB into a million instances of random 128-bit string-OT, of total size 16MB (receiver) and 32MB (sender), in an estimated[8] time of around a second on a single core of a modern CPU. Alternatively, seeds of length 7KB can be expanded into 65 thousand OTs at roughly half the amortized computational cost. Factoring in the cost of securely distributing Gen (with semi-honest security, building on [32]), the amortized communication complexity of our silent OT extension protocol is 0–3 bits *for each random 128-bit string-OT*. To put that into context, state-of-the-art OT extension protocols [46, 7] require 128 bits of communication per random 128-bit string-OT and can generate around 10 million OTs per second [41] over a fast network, so the price we pay for the (much) lower communication complexity seems quite modest. Even for the easier case of random *bit*-OT, the best previous OT extension protocol [55] required roughly 80 bits of communication per OT.

Other PCG Constructions. We present an assortment of practically feasible PCGs for other useful two-party correlations, based on a variety of underlying tools and assumptions.

- *PCG for Constant-Degree Polynomials from LPN.* We show that a generalization of the LPN-based VOLE generator from [14] can be used to obtain a PCG for any constant-degree additive correlation, namely a correlation that additively secret-shares a vector of degree-d polynomials of a random $X \in \mathbb{F}^n$ for some constant $d \geq 2$. This PCG relies on LPN over $\mathbb{F}$ in a similar noise regime as the PCG for OT from Section 1.1. In fact, by increasing the computation time (but still keeping it polynomial), one can use the LPN assumption in a parameter regime that is not known to imply public-key encryption [2], let alone OT. The main caveat is that even for generating simple degree-d correlations, such as $\Omega(n)$ Beaver triples ($d = 2$), the *computational* complexity of Expand is bigger than n^d. While much slower than our PCG for OT, this construction may still be practically feasible for $d = 2$ even with reasonably large n. We leave the question of obtaining more efficient variants of this construction to future work.
  As discussed in Section 1.1, this PCG construction implies (2-party) HSS schemes for constant-degree polynomials from LPN. By additionally assum-

[8] We caution that we have not implemented our constructions. Our estimates are based on counting basic operations and estimating their cost; the actual running times may vary due to other costs we neglected such as cache misses. We leave the task of optimizing and implementing our constructions to future work.

ing a standard OT protocol, it implies secure two-party computation protocols for constant-degree polynomials in which the communication complexity is nearly linear in the input size. Using the techniques from [18, 25], it also implies an "almost-sublinear" general secure computation protocol: for any constant $c > 1$ and layered boolean circuit of size s (and assuming binary LPN and OT), there is a secure two-party computation protocol with polynomial computation and total communication bounded by s/c. We stress again that these are mainly feasibility results because of the high computational cost of this PCG construction.

- *PCG for One-Time Truth Tables from any PRG.* One-time truth tables (OTTT) are a type of correlation that allow secure evaluation of a public lookup table in MPC, on a secret-shared input [47, 28, 30], and are well-suited to computations such as the S-box of AES. For MPC with active security, the correlation outputs need to be authenticated with information-theoretic MACs, as in the recent TinyTable protocol [28]. We present a very simple PCG for authenticated OTTT using only a distributed point function (DPF) [39, 19], which in turn can be efficiently constructed from any pseudorandom generator (PRG). This PCG follows naturally from a building block of the silent OT extension construction (as we explain in Section 2). It compresses the storage cost of an authenticated OTTT from $O(\lambda n)$ bits down to $O(\lambda \log n)$ bits, for a table of size n, giving a reduction in size of over 20x for a length-256 table such as the AES S-box. There is a concretely efficient protocol to distribute the seed generator Gen with semi-honest security by using the distributed DPF key generation protocol from [32]. While a similar protocol with malicious security is considerably more expensive, even a naive approach based on general-purpose secure computation (e.g., using recent protocols such as [51]) is feasible in practice, enabling the compressed storage benefit of the PCG-based approach.
- *PCGs from Homomorphic Secret Sharing.* We give practically feasible PCG constructions for OLE and (authenticated) Beaver triple correlations, which are useful for arithmetic MPC protocols such as SPDZ [29]. For these constructions we use HSS based on ring-LWE [23, 31, 22] and the BGN (pairing-based) cryptosystem [13, 18, 16]. To expand the seeds, we rely on a multivariate quadratic (MQ) assumption based PRG, which limits the stretch to sub-quadratic, but allows for reasonable computational efficiency. For example, with our ring-LWE-based PCG we estimate that one should be able to expand a pair of 3GB seeds into 17GB of authenticated Beaver triples in a 128-bit field, at a rate of around 6 thousand triples per second; various tradeoffs are possible between seed size and computation time, and we also explore an iterative variant which produces triples in small batches. Securely computing Gen to distribute the seeds is relatively cheap compared to the expansion phase, and the overall performance should be comparable to recent work on actively secure triple generation with much more interaction [53]. With BGN, we estimate around 200ms for computing an OLE correlation over $\mathbb{Z}_N$ for small N (say, $N < 10$). Although much more expensive than our silent OT extension, an advantage of the ring-LWE-based constructions,

beyond the richer class of correlations, is that they can be extended to the *multi-party* setting, as we discuss next.

PCGs for Multi-Party Correlations. Finally, we present a general transformation for extending certain classes of PCGs from the 2-party to the multi-party setting. This can be applied to PCGs for simple bilinear correlations, including VOLE and Beaver triples, giving the first non-trivial, efficient PCG constructions in the multi-party setting. The transformation applies to most of our 2-party PCGs, including the LPN-based PCG for constant-degree correlations.

On top of the silent preprocessing feature, an appealing application of our multi-party PCGs is in obtaining secure M-party computation protocols with *total* communication complexity $O(Ms + M^2 \cdot s^\epsilon)$ (for circuit size s and constant $0 < \epsilon < 1$). The $O(Ms)$ term is the cost of the (information-theoretic) online phase, and the $O(M^2 \cdot s^\epsilon)$ term is the cost of distributing the PCG seed generation, which is the only part of the protocol requiring pairwise communication. This should be contrasted with OT-based MPC protocols, which have total communication complexity $\Omega(M^2 s)$ [40, 43] . Protocols with such communication complexity (without the silent preprocessing feature) could previously be based on different flavors of somewhat homomorphic encryption [35, 27, 29]. We get the first such protocol that only relies on LPN and OT, and the first practically feasible protocol that has sublinear-communication offline phase and information-theoretic online phase.

Table 1. Summary of the New PCG Constructions. Costs are estimated based on one core of a modern laptop.

PCG	Section 5	[15]	[15]	[15]	[15]	[15]
Assumption	LPN	PRG*	LPN	deg-d HSS + MQ/LPN	SXDH + LPN	LWE + MQ
Correlations	OT*	OTTT*	deg-d	deg-$d/2$	deg-2	deg-d
Efficiency	1M OT/s†	-	-	-	5 OLE/s‡	6000/s**
Multiparty (bilinear corr.)	✗	✗	✔	✔	✗	✔

* PRG stands for an arbitrary pseudorandom generator, OT for random oblivious transfer, and OTTT for authenticated one-time truth-table correlation.

† With average communication of 0.2 bits/OT.

‡ For OLE correlation over a small (constant size) ring.

** For generating authenticated Beaver triples over a 128-bit prime field.

Additional Applications. From our silent OT extension protocol, we obtain the following additional results:

- *Oblivious Pseudorandom Functions (OPRFs).* An OPRF [36] is a two-party protocol for securely evaluating a pseudorandom function, whose key is known by one party, on a secret input known by the second party. OPRFs serve as the main building blocks in recent protocols for private set intersection [56]. Our silent OT construction can be used to obtain a form of batch OPRF with cost as little as 1 bit of communication per OPRF evaluation on a random input, leading to around a factor two reduction in communication for these protocols.
- *Reusable-Preprocessing NIZK.* Consider the following setting for non-interactive zero knowledge (NIZK) with reusable interactive setup: In an offline setup phase, before the statements to be proved are known, the prover and the verifier interact to securely generate correlated random seeds. The seeds can then be used to prove any polynomial number of statements by having the prover send a single message to the verifier for each statement. Such a notion was recently constructed in [14], building on [24], using their PCG for VOLE. Our silent OT extension can be used to obtain an improved reusable-preprocessing NIZK system for NP, under the standard LPN assumption over $\mathbb{F}_2$. As compared to the reusable NIZK of [14], our NIZK relies on a more standard assumption (LPN over $\mathbb{F}_2$ versus large $\mathbb{F}$), and the setup cost is independent of both the number of statements and their size (whereas in [14], the setup cost was independent of the number of statements, but grows linearly with a bound on their size). On the down side, our OT-based NIZK protocols do not have the computational complexity advantages of the VOLE-based constructions from [14].
- *Efficient Secure Matrix Multiplication.* As a stepping stone towards silent OT extension, we construct a PCG for a generalization of VOLE called *subfield VOLE*. This can be seen as a form of batch VOLE where the $\boldsymbol{u}$ value is reused across several instances, and can be applied to compute secret-shared tensor products and matrix multiplication more efficiently. Compared with naively using a PCG for standard VOLE, we reduce the seed size by at least a $O(\log n)$ factor.

Finally, our PCG for OTTT yields the following application.

- *Improved 2-PC with Sublinear Online Communication.* Standard approaches to secure computation with preprocessing (e.g., SPDZ) still require online communication that is linear in the circuit size. Recently, Couteau [25] demonstrated asymptotic feasibility of information-theoretic secure 2-party computation (2-PC) in the preprocessing model for a natural class of circuits (namely, "layered" circuits), with sublinear online communication, $O(s/\log\log s)$ for circuit size s. However, this comes at the cost of generating and storing $O(s^2)$ bits of correlated randomness.
  Our compressed one-time truth-table (OTTT) construction allows one to match the asymptotic complexity of [25], while reducing the amount of cor-

related randomness from quadratic to quasilinear in the circuit size, in exchange for settling for computational security and assuming the existence of one-way functions.

1.2 Paper Organization

In this extended abstract we only present our main techniques and results. For the full details of constructions and other results, we refer to the full version of this work [15]. We begin in Section 2 with an overview of our techniques, followed by preliminaries in Section 3. In Section 4, we present our PCG definition and foundational results. Section 5 contains our PCGs for subfield-VOLE and OT, leading to our silent OT extension construction.

2 Technical Overview of Constructions

In this section we give a high-level overview of the techniques that underly our different PCG constructions.

2.1 Background

Our PCG constructions rely on different types of *homomorphic secret sharing* (HSS) and *function secret sharing* (FSS) schemes. Informally, HSS is a form of secret sharing that allows a secret x to be split up into shares $\mathsf{k}_0, \mathsf{k}_1$, such that a party holding k_i can *locally* obtain an additive secret share of $f(x)$, for some function f. FSS is the dual notion: starting with a function f, and splitting into shares f_0, f_1 such that each share f_i hides f, but can be used to obtain an additive sharing of $f(x)$ for some public input x.[9] FSS for a class of point functions (i.e., functions f which evaluate to 0 on all but a single input) is called a *distributed point function* [39], and can be constructed very efficiently based on a pseudorandom generator (PRG) [19]. There are HSS constructions for branching programs based on DDH [18] or lattices [22], or general circuits from strong forms of fully homomorphic encryption [31].

2.2 Overall methodology

At a high level, our constructions can all be seen as examples of the following blueprint: construct an HSS scheme that can homomorphically evaluate the composition of a pseudorandom generator (PRG) with a function f that uses the expanded randomness to compute the desired correlation. This can be used to obtain PCGs for any *additive* correlation; i.e., that outputs random additive shares of some distribution. Of course, the main challenge lies in instantiating this *efficiently*, since plugging in even a low-degree PRG to an off-the-shelf HSS

[9] FSS is actually equivalent to HSS for a related class of functions, but we differentiate between the two for convenience, depending on the applications.

scheme is typically not practical. We instead use specialized HSS constructions that pair well with our carefully chosen PRGs.

As a stepping stone, our constructions implicitly construct a *compressible* form of HSS, which allows the sharing of inputs from some distribution $\mathcal{D}$, such that the share size is *smaller than* an uncompressed output of $\mathcal{D}$, and we can still compute some useful function f on the expanded inputs. We typically choose $\mathcal{D}$ to be a sparse distribution on vectors, or another similarly compressible distribution. We then convert these long $\mathcal{D}$-vectors to slightly shorter (but still long) *random*-looking vectors, by homomorphically multiplying by a compressive linear map. Under a suitable LPN-type assumption, this combination of expanding the compressed $\mathcal{D}$-vector followed by linear compression acts as a PRG in the above blueprint, and we can proceed to homomorphically compute the desired correlation.

For example, when $\mathcal{D}$ samples a sparse, low-weight vector $\boldsymbol{e}$ over $\mathbb{F}_2$, and the linear map is a random matrix H, then distinguishing $(H, \boldsymbol{e} \cdot H)$ from random is as hard as the problem of decoding a random binary linear code, which corresponds to the standard LPN assumption [12, 2]. Another example is when $\mathcal{D}$ outputs a *tensor product* of two short, uniform vectors. Recovering the short vectors given only $(\boldsymbol{x} \otimes \boldsymbol{y}) \cdot H$ is the problem of solving a random system of multivariate quadratic equations (MQ problem), which is believed to be hard for a suitable choice of parameters [57, 63, 11, 4]. In particular, the decision version of MQ is polynomially reducible to its search version [11].

We remark that the resulting PRGs do not necessarily conform to standard metrics of simplicity, such as low degree or low locality, and in isolation may appear somewhat unnatural. This exemplifies an interesting observation that "HSS-friendliness" may indeed be a new type of metric that does not directly align with those previously studied.

2.3 Silent OT Extension

As a building block for silent OT extension, we start by constructing a PCG for a two-party correlation we call *subfield vector oblivious linear evaluation* (subfield VOLE). This correlation works over a field $\mathbb{F}_q$, and a subfield $\mathbb{F}_p$, where $q = p^r$. It first samples a random $x \in \mathbb{F}_q$, $\boldsymbol{u} \in \mathbb{F}_p^n, \boldsymbol{v} \in \mathbb{F}_q^n$, then outputs $(\boldsymbol{u}, \boldsymbol{v})$ to the sender and $(x, \boldsymbol{w} = \boldsymbol{u}x + \boldsymbol{v})$ to the receiver.[10] Our construction is a generalization of the vector-OLE construction from [14]: when $p = q$ the correlation is exactly vector-OLE, but using $q > p$ opens up additional applications. For example, viewing $x \in \mathbb{F}_q$ as a vector $\boldsymbol{x} \in \mathbb{F}_p^r$, subfield VOLE can be seen as computing additive shares of the $r \times n$ tensor product $\boldsymbol{x} \otimes \boldsymbol{u}$, which can be useful for secure two-party matrix multiplication, and other linear algebra tasks. Compared with using r copies of VOLE [14] to achieve the same task, we reduce the seed size by a $O(\log n)$ factor and obtain more efficient computation.

[10] We view elements of $\mathbb{F}_p$ embedded into $\mathbb{F}_q$ throughout, so that the multiplication $\boldsymbol{u} \cdot x$ happens over $\mathbb{F}_q$.

To build a PCG for subfield VOLE, we consider a compressible distribution $\mathcal{D}$ that outputs random sparse vectors of weight t and length n'. First, notice that we can compress a secret-sharing of the j-th unit vector $\boldsymbol{e}_j \in \{0,1\}^{n'}$, using a distributed point function (DPF) for the point $(j, 1)$: evaluating a DPF key on input i produces a random share of 0 on all inputs except $i = j$, where it outputs a share of 1. Hence, performing all n' evaluations results in shares of the entire vector $\boldsymbol{e}_j$. This easily extends to weight t vectors, by naively using t DPFs and summing up the shares of the t unit vectors (this step can be optimized with a multi-point DPF as described in [14]).

Although it may appear that this only allows us to compress sparse vectors, and not perform any useful HSS computations afterwards, we observe that with a small tweak we can use this to build HSS for the family of randomized functions

$$\mathcal{F} = \{f_H : \mathbb{F}_q \to \mathbb{F}_q^n, x \mapsto x \cdot \boldsymbol{e} \cdot H \mid \boldsymbol{e} \xleftarrow{\$} \mathcal{HW}_t, H \in \mathbb{F}_p^{n' \times n}\} \tag{1}$$

where $\mathcal{HW}_t$ is the distribution that outputs a random weight-t vector over $\mathbb{F}_p^{n'}$ (with each entry either 0 or uniform). We remark that a naive description of the class $\mathcal{F}$ gives functions with very high degree, which could *not* be evaluated using simple HSS schemes, which highlights the importance of tailoring a specific solution.

To upgrade the above sketch to get HSS for $\mathcal{F}$, we make one small modification: using t DPFs that output shares over $\mathbb{F}_q$, we specify the i-th DPF by the point $(j_i, y_i \cdot x)$ for some random index j_i and $y_i \in \mathbb{F}_p^*$, instead of $(j_i, 1)$ as before. When evaluating the DPFs, the parties now obtain additive shares of $\boldsymbol{e} \cdot x$, where $\boldsymbol{e}$ contains all t y_i's in random positions. Since additive secret sharing is linear, any linear map H can then be locally applied on the shares.

If $H \in \mathbb{F}_p^{n' \times n}$ is a compressive linear map with $n < n'$, the vector $\boldsymbol{u} = \boldsymbol{e} \cdot H$ is pseudorandom under a suitable form of the LPN (or syndrome decoding) assumption. Concretely, we require that a t-noisy random codeword in the code whose *parity check* matrix is H is pseudorandom. This immediately yields a subfield VOLE generator, where each party's seed contains a set of DPF seeds, and the sender additionally gets the points (j_i, y_i), and the receiver gets x, since additive shares of $x \cdot \boldsymbol{u}$ can be locally converted to the $(\boldsymbol{v}, \boldsymbol{w})$ components of a VOLE correlation.

Our next observation, inspired by the OT extension protocol of Ishai et al. [46], is that subfield VOLE already gives as a restricted form of oblivious transfer, known as correlated OT or Δ-OT. If we run subfield VOLE over $\mathbb{F}_2$, embedded in $\mathbb{F}_{2^r}$, then the VOLE sender obtains a set of pairs $u_i \xleftarrow{\$} \mathbb{F}_2, v_i \xleftarrow{\$} \mathbb{F}_{2^r}$, while the VOLE receiver gets $x \xleftarrow{\$} \mathbb{F}_{2^r}$ and $w_i = x \cdot u_i + v_i$, for $i = 1, \ldots, n$. Now switch the roles of sender and receiver, so the VOLE sender becomes an OT receiver with choice bit u_i and string v_i. If $u_i = 0$ then $v_i = w_i$, whilst if $u_i = 1$ then $v_i = w_i - x$, hence, this is exactly a 1-out-of-2 OT where the OT sender's (formerly VOLE receiver's) messages are all of the form $(w_i, w_i - x)$.

On its own, this type of Δ-OT is already useful for many applications such as garbled circuits and secure computation with information-theoretic MACs [61, 59]. However, most importantly, following [46], the parties can locally convert

such a correlated OT into an OT on random strings, using a hash function that is pseudorandom under correlated inputs. This gives us a PCG for random oblivious transfer, where the seed size is essentially that of t distributed point functions, or $O(t\lambda \log n)$ bits. Combining this with an efficient secure protocol for setting up a pair of DPF keys [32], we obtain our silent OT extension protocol, which produces n pseudorandom string-OTs with $o(n)$ bits of communication.

2.4 One-Time Truth Tables

We next show how to adapt the above approach to produce authenticated, one-time truth table correlations, which can be used to efficiently perform table lookups in MPC [47, 28, 30]. This construction is straightforward given the above description of our subfield-VOLE generator, so we informally explain it here and defer the complete description to the full version.

The correlation we want to produce, for a lookup table $T : [n] \to \{0,1\}^m$, is an additive secret-sharing of

$$\big(\alpha, \{y_i, \gamma_i\}_{i\in[n]}\big) \quad \text{where} \quad y_i = T(s + i \bmod n), \gamma_i = y_i \cdot \alpha \in \mathbb{F}_{2^\lambda} \tag{2}$$

for $\alpha \xleftarrow{\$} \mathbb{F}_{2^\lambda}, s \xleftarrow{\$} [n]$. Here, the y_i's are equal to T shifted by a random offset s, while the γ_i's are information-theoretic MACs on y_i under the key α, used to obtain active security in the MPC protocol.

Our starting point is the observation from [52] that the y_i's can be generated locally, given secret-shares of a random unit vector. This is because, if $\boldsymbol{e}_s \in \{0,1\}^n$ is the s-th unit vector, then we have

$$T(s + i \bmod n) = \sum_{j=1}^{n} \boldsymbol{e}_s[j] \cdot T(i + j \bmod n)$$

which is linear in $\boldsymbol{e}_s$. We can further obtain the γ_i's (namely, the authenticated $\gamma_i = y_i \cdot \alpha$) if we additionally have secret-shares of the corresponding scaled vector $\alpha \cdot \boldsymbol{e}_s$.

The core observation is that a DPF gives precisely a *compressed* secret sharing of such a secret vector $(1||\alpha) \cdot \boldsymbol{e}_s \in (\{0,1\}^{1+\lambda})^n$: requiring only $O(\lambda \log n)$ bits in the place of $O(\lambda n)$.

More concretely, this leads to the following, simple approach for a PCG to generate shares of (2): use the previous DPF-based construction of HSS for the family in (1) over $\mathbb{F}_2$, with $t = 1$, $x = (1\|\alpha)$ for $\alpha \xleftarrow{\$} \mathbb{F}_{2^\lambda}$, and H the linear map induced by T in the equation above. The resulting PCG has seed size essentially the same as one DPF, which is $O(\lambda \log n)$ bits. This gives a large compression over the previous, practical approach from [28], which required $O(\lambda n)$ bits per table. Expanding the PCG is relatively cheap in practice, since in 2-PC applications only a single entry of each table is ever used, and this can be computed on-the-fly with $O(n)$ PRG evaluations.

A downside of this construction is that it seems difficult to produce the necessary PCG seeds with good concrete efficiency in the malicious setting, since the only known approach in this setting requires evaluating a PRG inside 2-PC [32]. However, our result is still interesting for a preprocessing phase with semi-honest security, or when a trusted dealer is present. Alternatively, if one can afford the cost of distributing Gen with malicious security via general-purpose 2-PC, the resulting correlated seeds only require a small amount of storage, and their local expansion is (automatically) secure against malicious parties.

2.5 PCGs for Constant-Degree Polynomials from LPN

We construct PCGs for constant-degree polynomials, using again function secret sharing for multi-point functions together with LPN. At a high level, the construction builds upon the fact that given two sparse vectors $\boldsymbol{a}, \boldsymbol{b}$, their tensor product $\boldsymbol{a} \otimes \boldsymbol{b}$ is sparse as well, hence shares of $\boldsymbol{a} \otimes \boldsymbol{b}$ can be compressed using an FSS, as for vector-OLE generators and silent OT extension. Then, a compressive mapping can be applied to obtain $\boldsymbol{x} \otimes \boldsymbol{y}$ from $\boldsymbol{a} \otimes \boldsymbol{b}$, where $\boldsymbol{x} = (\boldsymbol{a} \cdot H)$ and $\boldsymbol{y} = (\boldsymbol{b} \cdot H)$ are *pseudorandom* under the LPN assumption, thanks to the bilinearity of the tensor product (and linearity of H). This immediately leads to a PCG for bilinear functions, which can be easily generalized to a PCG for constant-degree polynomials. However, the share size grows as $O(t^d)$, where t is the number of noisy coordinates in the LPN instance, and d is the degree of the polynomial. The computation cost grows as $O(n^{2d})$, where n is the input size.

2.6 PCGs from Ring-LWE and BGN-based HSS

We construct PCGs for more general two-party correlations, building upon the specific structure of homomorphic encryption-based HSS schemes [31, 22] and group-based HSS schemes [18, 20, 16]. Our key observation is that in both HSS schemes encodings of large pseudorandom strings can be compressed efficiently using an "HSS-friendly PRG" as described in Section 2.2. For the ring-LWE based construction, we obtain compression with a PRG based on the multivariate quadratic equations problem, and present several ways of optimizing this with batching techniques for homomorphic encryption, which lead to different tradeoffs for seed size and computational cost.

The group-based approach requires more involved techniques: The underlying HSS scheme uses two types of encodings, where so-called level-1 encodings are ElGamal ciphertexts, and level-2 encodings are shares of $\mathsf{sk} \cdot \boldsymbol{x}$ for a vector $\boldsymbol{x}$, where sk is the secret key of the homomorphic encryption scheme. Then, a special HSS operation allows to compute level-2 encodings of bilinear functions applied to a level-1 encoding and a level-2 encoding. Using two parallel instances of the PCG for vector-OLE of [14] allows us to efficiently compress shares of $\boldsymbol{y}$ and $\mathsf{sk} \cdot \boldsymbol{y}$, where $\boldsymbol{y}$ is a pseudorandom vector and sk is a shared value, to only $O(\lambda t \log n)$ bits, under the LPN assumption with t noisy coordinates. Furthermore, encrypting short random sparse vectors suffices for homomorphically evaluating a specific LPN-based PRG directly on the level-1 encodings, as long

as they support evaluation of degree-2 functions. This can be ensured by using BGN-style pairing-based encryption for the group-based HSS. Since the HSS comes with an inverse-polynomial error probability, we further develop a new method to efficiently remove the faulty outputs, building upon our silent OT extension protocol.

For both schemes, we discuss various optimizations and provide detailed efficiency estimations.

2.7 Multi-Party PCGs

As our final contribution, we construct *multi-party* PCGs for a useful class of bilinear correlations. Concretely, for a given bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, we consider M-party correlations of the form $\{(a_i, b_i, c_i)\}_{i\in[M]}$, consisting of additive secret shares of random elements $a \in \mathbb{G}_1$, $b \in \mathbb{G}_2$, and their image $c = e(a, b) \in \mathbb{G}_T$. For appropriate choice of groups and bilinear operation, this captures M-party OT, M-party vector OLE, M-party Beaver triples, and more.

Our construction approach provides a semi-generic transformation from any PCG for a corresponding *2-party* correlation $\{(a, c_1), (b, c_2)\}$ for random a, b, and $c_1 + c_2 = e(a, b)$, if the PCG satisfies an additional programmability property. Roughly, this property requires a way of "reusing" the inputs a and b across instances without compromising security.

The M-party construction leverages this structure by executing $M(M-1)$ pairwise instances of the underlying 2-party PCG, for all the "cross-terms." Namely, we think of each a_i and b_i from the final M-party correlation as playing the role of a or b in the 2-party correlation, with all possible partners. The desired M-party additive shares c_i can then be derived by combining $c_{ii} = a_i b_i$ (computable locally) together with $\{c_{ij}, c_{ji}\}_{j\in[M]\setminus\{i\}}$ resulting from the 2-party correlations for pairs (a_i, b_j) and (a_j, b_i). The resulting M-party PCG keys consist of $M(M-1)$ keys from the 2-party PCG, together with short expandable shares of 0 for rerandomization.

We observe that the necessary programmability property is satisfied by our subfield VOLE construction and the 2-party VOLE PCG from [14], as well as the 2-party bilinear PCGs constructed in this work (including OT and Beaver triples) from group-based and lattice-based HSS and from LPN (in the full version [15]). As a corollary, we obtain M-party variants of these correlations with quadratic blowup in computation and share size. Interestingly, our silent OT extension construction does *not* seem to support the necessary programmability, since the resulting sender message pairs are implicitly defined as a function of the receiver's bit selections.

3 Preliminaries

We say that a function $\mathsf{negl}\colon \mathbb{N} \to \mathbb{R}^+$ is *negligible* if it vanishes faster than every inverse polynomial. For two families of distributions $X = \{X_\lambda\}$ and $Y = \{Y_\lambda\}$

indexed by a security parameter $\lambda \in \mathbb{N}$, we write $X \overset{c}{\approx} Y$ if X and Y are *computationally indistinguishable* (namely, any family of circuits of size $\mathsf{poly}(\lambda)$ has a negligible distinguishing advantage), $X \overset{s}{\approx} Y$ if they are *statistically indistinguishable* (namely, the above holds for arbitrary distinguishers), and $X \equiv Y$ if the two families are identically distributed.

Notation. We usually denote matrices with capital letters (A, B, C) and vectors with bold lowercase $(\boldsymbol{x}, \boldsymbol{y})$. By default, vectors are assumed to be row vectors. We write $A|_{i,j}$ to denote the entry (i, j) of a matrix A. Given a vector $\boldsymbol{x}$ of length $|\boldsymbol{x}| = n$, the notation $\mathsf{HW}(x)$ denotes the Hamming weight $\boldsymbol{x}$, *i.e.*, the number of its nonzero entries. Given a distribution $\mathcal{D}$, we denote by $\mathsf{Im}(\mathcal{D})$ the image of $\mathcal{D}$ (i.e., its support set).

3.1 Function Secret Sharing

Informally, an FSS scheme for a class of functions $\mathcal{C}$ is a pair of algorithms $\mathsf{FSS} = (\mathsf{FSS.Gen}, \mathsf{FSS.Eval})$ such that:

- $\mathsf{FSS.Gen}$ given a function $f \in \mathcal{C}$ outputs a pair of keys (K_0, K_1);,
- $\mathsf{FSS.Eval}$, given K_b and input x, outputs y_b such that y_0 and y_1 form additive shares of $f(x)$.

The security requirement is that each key K_b computationally hide f, except for revealing the input and output domains of f. For a formal definition, see e.g. [19].

Some applications of FSS require applying the evaluation algorithm on *all inputs*. Following [19, 14], given an FSS scheme $(\mathsf{FSS.Gen}, \mathsf{FSS.Eval})$, we denote by $\mathsf{FSS.FullEval}$ an algorithm which, on input a bit b, and an evaluation key K_b (which defines the input domain I), outputs a list of $|I|$ elements of $\mathbb{G}$ corresponding to the evaluation of $\mathsf{FSS.Eval}(b, K_b, \cdot)$ on every input $x \in I$ (in some predetermined order). While $\mathsf{FSS.FullEval}$ can always be realized with $|I|$ invocations of $\mathsf{FSS.Eval}$, it is typically possible to obtain a more efficient construction. Below, we recall some results from [19] on FSS schemes for useful classes of functions.

Distributed Point Functions. A distributed point function (DPF) [39] is an FSS scheme for the class of point functions $f_{\alpha,\beta} : \{0,1\}^\ell \to \mathbb{G}$ which satisfy $f_{\alpha,\beta}(\alpha) = \beta$, and $f_{\alpha,\beta}(x) = 0$ for any $x \neq \alpha$. A sequence of works [39, 17, 19] has led to highly efficient constructions of DPF schemes from any pseudorandom generator (PRG), which can be implemented in practice using block ciphers such as AES.

Theorem 1 (PRG-based DPF [19], Theorems 3.3 and 3.4). *Given a PRG $G : \{0,1\}^\lambda \to \{0,1\}^{2\lambda+2}$, there exists a DPF for point functions $f_{\alpha,\beta} : \{0,1\}^\ell \to \mathbb{G}$ with key size $\ell \cdot (\lambda + 2) + \lambda + \lceil \log_2 |\mathbb{G}| \rceil$ bits. For $m = \lceil \frac{\log |\mathbb{G}|}{\lambda+2} \rceil$, the key generation algorithm Gen invokes G at most $2(\ell + m)$ times, the evaluation*

algorithm Eval *invokes* G *at most* $\ell + m$ *times, and the full evaluation algorithm* $\mathsf{FullEval}$ *invokes* G *at most* $2^\ell(1+m)$ *times.*

Note that a naive construction of $\mathsf{FullEval}$ from Eval would require $2^\ell(\ell+m)$ invocations of G.

FSS for Multi-Point Functions. Similarly to [14], we use FSS for *multi-point functions*. A k-point function evaluates to 0 everywhere, except on k specified points. When specifying multi-point functions we often view the domain of the function as $[n]$ for $n = 2^\ell$ instead of $\{0,1\}^\ell$.

Definition 2 (Multi-Point Function [14]). *An* (n,t)*-multi-point function over an abelian group* $(\mathbb{G},+)$ *is a function* $f_{S,\boldsymbol{y}} : [n] \to \mathbb{G}$*, where* $S = (s_1, \cdots, s_t)$ *is an ordered subset of* $[n]$ *of size* t *and* $\boldsymbol{y} = (y_1, \cdots, y_t) \in \mathbb{G}^t$*, defined by* $f_{S,\boldsymbol{y}}(s_i) = y_i$ *for any* $i \in [t]$*, and* $f_{S,y}(x) = 0$ *for any* $x \in [n] \setminus S$*.*

We assume that the description of S includes the input domain $[n]$ so that $f_{S,\boldsymbol{y}}$ is fully specified.

A *Multi-Point Function Secret Sharing* (MPFSS) is an FSS scheme for the class of multi-point functions, where a point function $f_{S,\boldsymbol{y}}$ is represented in a natural way. We assume that an MPFSS scheme leaks not only the input and output domains but also the number of points t that the multi-point function specifies. An MPFSS can be easily obtained by adding t instances of DPF; optimized constructions of MPFSS, using batch codes [48] to speed up the full domain evaluation algorithm, were presented in [14].

3.2 Learning Parity with Noise

Our constructions rely on variants of the Learning Parity with Noise (LPN) assumption [12] over either $\mathbb{F}_2$ or a large finite field $\mathbb{F}$. Unlike the LWE assumption, in LPN over $\mathbb{F}$ the noise is assumed to have a small Hamming weight. Concretely, the noise is a random field element in a small fraction of the coordinates and 0 elsewhere. Similar assumptions have been previously used in the context of secure arithmetic computation [58, 50, 3, 33, 37]. Unlike most of these works, the flavors of LPN on which we rely do not require the underlying code to have an algebraic structure and are thus not susceptible to algebraic (list-) decoding attacks.

Definition 3 (LPN). *Let* $\mathcal{D}(\mathcal{R}) = \{\mathcal{D}_{k,q}(\mathcal{R})\}_{k,q\in\mathbb{N}}$ *denote a family of distributions over a ring* $\mathcal{R}$*, such that for any* $k, q \in \mathbb{N}$*,* $\mathsf{Im}(\mathcal{D}_{k,q}(\mathcal{R})) \subseteq \mathcal{R}^q$*. Let* $\mathbf{C}$ *be a probabilistic code generation algorithm such that* $\mathbf{C}(k,q,\mathcal{R})$ *outputs a matrix* $A \in \mathcal{R}^{k\times q}$*. For dimension* $k = k(\lambda)$*, number of samples (or block length)* $q = q(\lambda)$*, and ring* $\mathcal{R} = \mathcal{R}(\lambda)$*, the* $(\mathcal{D}, \mathbf{C}, \mathcal{R})$-$\mathsf{LPN}(k,q)$ *assumption states that*

$$\{(A,\boldsymbol{b}) \mid A \xleftarrow{\$} \mathbf{C}(k,q,\mathcal{R}), \boldsymbol{e} \xleftarrow{\$} \mathcal{D}_{k,q}(\mathcal{R}), \boldsymbol{s} \xleftarrow{\$} \mathbb{F}^k, \boldsymbol{b} \leftarrow \boldsymbol{s}\cdot A + \boldsymbol{e}\}$$
$$\overset{c}{\approx} \{(A,\boldsymbol{b}) \mid A \xleftarrow{\$} \mathbf{C}(k,q,\mathcal{R}), \boldsymbol{b} \xleftarrow{\$} \mathcal{R}^q\}$$

Here and in the following, all parameters are functions of the security parameter λ and computational indistinguishability is defined with respect to λ.

When $\mathcal{R} = \mathbb{F}_2$ and $\mathcal{D}$ is the Bernoulli distribution over $\mathbb{F}_2^q$, where each coordinate is 1 with probability r and 0 otherwise, this corresponds to the standard binary LPN assumption.

Note that the search LPN problem, of finding the vector can be reduced to the decisional LPN assumption as defined above above when the code generator $\mathbf{C}$ outputs a uniform matrix A [12, 5]. However, this is less relevant for us as we are mainly interested in efficient variants with more structured codes. See [34] for further discussion of search-to-decision reductions in the general case.

Example: LPN with Fixed Weight Noise. For a finite field $\mathbb{F}$, we denote by $\mathcal{HW}_r(\mathbb{F})$ the distribution of uniform, weight r vectors over $\mathbb{F}$; that is, a sample from $\mathcal{HW}_r(\mathbb{F})$ is a uniformly random nonzero field element in r random positions, and zero elsewhere. The $(\mathsf{Ber}_r(\mathbb{F})^q, \mathbf{C}, \mathbb{F}) - \mathsf{LPN}(k, q)$ assumption corresponds to the standard (non-binary, fixed-weight) LPN assumption over a field $\mathbb{F}$ with code generator $\mathbf{C}$, dimension k, number of samples (or block length) q, and noise rate r.

When the block length q and noise rate r are such that k random coordinates will be all noiseless with non-negligible probability (e.g., when r is constant and $q = \Omega(k^2)$), LPN can be broken via Gaussian elimination (cf. [6]). This attack does not apply to our constructions, which typically have $q = O(k)$.

Definition 4 (dual LPN). *Let $\mathcal{D}(\mathcal{R})$ and $\mathbf{C}$ be as in Definition 3, $n, n' \in \mathbb{N}$ with $n' > n$, and define $\mathbf{C}^{\perp}(n', n, \mathcal{R}) = \{B \in \mathcal{R}^{n' \times n} : A \cdot B = 0, A \in \mathbf{C}(n' - n, n', \mathcal{R}), \mathsf{rank}(B) = n\}$.*

For $n = n(\lambda), n' = n'(\lambda)$ and $\mathcal{R} = \mathcal{R}(\lambda)$, the $(\mathcal{D}, \mathbf{C}, \mathcal{R})$-$\mathsf{dual\text{-}LPN}(n', n)$ assumption states that

$$\{(H, \boldsymbol{b}) \mid H \xleftarrow{\$} \mathbf{C}^{\perp}(n', n, \mathcal{R}), \boldsymbol{e} \xleftarrow{\$} \mathcal{D}(\mathcal{R}), \boldsymbol{b} \leftarrow \boldsymbol{e} \cdot H\}$$
$$\overset{\mathrm{c}}{\approx} \{(H, \boldsymbol{b}) \mid H \xleftarrow{\$} \mathbf{C}^{\perp}(n', n, \mathcal{R}), \boldsymbol{b} \xleftarrow{\$} \mathcal{R}^n\}$$

The search version of the dual LPN problem is also known as syndrome decoding. The decision version defined above is equivalent to primal variant of LPN from Definition 3 with dimension $k = n' - n$ and number of samples $q = n'$. This follows from the simple fact that $(\boldsymbol{s} \cdot A + \boldsymbol{e}) \cdot H = \boldsymbol{s} \cdot A \cdot H + \boldsymbol{e} \cdot H = \boldsymbol{e} \cdot H$, when H is the parity-check matrix of A.

Remark 5. For any code generation algorithm $\mathbf{C}$ where $\mathsf{dual\text{-}LPN}$ is hard, it must hold that for $H \xleftarrow{\$} \mathbf{C}^{\perp}(n', n', \mathcal{R})$, H is full rank with overwhelming probability. If that was not the case, then we could easily distinguish $\boldsymbol{e} \cdot H$ from uniform due to a linear relation between some of its outputs.

Remark 6. As a concrete example of the actual flavor of the dual-LPN assumption we will use, our construction of silent OT from Section 5 relies on the dual-LPN assumption of Definition 3 with respect to a random linear code over

the field $\mathbb{F}_2$. For deriving our concrete parameters, we choose a *regular* error distribution of weight t, where a length-n' error vector has t non-zero coordinates spread across weight-1 blocks of length n'/t. This is known as the regular-LPN or *regular syndrome decoding* problem. When $n \geq 2^{16}$ and $n' = 4n$, a fixed-weight noise of $t \approx 32$ suffices to achieve 80-bit security against the best known attacks on this flavor of LPN, which all take time exponential in $(n'/n) \cdot t$. We will also consider alternative choices of linear codes (such as LDPC codes or quasi-cyclic codes) to improve the concrete computational efficiency in our estimates; such codes still lead to plausible variants of LPN and do not significantly improve known attacks compared with random codes.

4 Pseudorandom Correlation Generators

In this section we put forward a general notion of pseudorandom correlation generator (PCG) and study some of its limitations, capabilities, and relation with other primitives. We start with our formal definition of PCG in Section 4.1. We then discuss in Section 4.2 a simpler and more natural simulation-based definition of PCG, that would suffice for *all* applications, but is not realizable. As a second-best alternative, we show in Section 4.3 that PCGs can be used as a drop-in replacement for correlated randomness in every protocol that meets a slightly stronger security requirement, which is indeed met by natural MPC protocols in the correlated randomness model. In the full version [15], we also show a two-way relation between PCGs for a useful class of "low-degree correlations" and homomorphic secret sharing for low-degree polynomials.

4.1 Defining Pseudorandom Correlation Generators

At a high level, a pseudorandom correlation generator (PCG) for some relation takes as input a pair of short, correlated seeds and outputs long correlated pseudorandom strings, where the expansion procedure is deterministic and can be applied locally.

For correctness we require that the expanded output of a PCG is indistinguishable from truly random correlated strings.

For security it would be natural and straightforward to require that we can securely replace long correlated strings by short correlated seeds in any secure protocol execution. Unfortunately, as shown in the following section, this security requirement would be impossible to meet. Therefore, we will introduce (and subsequently prove useful) an indistinguishability based security notion. Namely, we require that an adversary given access to one of the short seeds k_σ, cannot distinguish the pseudorandom string $R_{1-\sigma}$ from a pseudorandom string that is chosen at random conditioned on (R_0, R_1) being correlated (where $R_\sigma = \mathsf{PCG}(\mathsf{k}_\sigma)$). In other words, an adversary given access to a short seed cannot learn more about the other party's pseudorandom string than what is obvious given access to its own pseudorandom string.

In order to formally define pseudorandom correlations, we first introduce the concept of a *correlation generator* as a PPT algorithm outputting correlated elements.

Definition 7 (Correlation Generator). *A PPT algorithm $\mathcal{C}$ is called a* correlation generator*, if $\mathcal{C}$ on input 1^λ outputs a pair of elements in $\{0,1\}^n \times \{0,1\}^n$ for $n \in \mathsf{poly}(\lambda)$.*

In order to define security, we require the notion of a reverse-sampleable correlation generator introduced in the following.

Definition 8 (Reverse-sampleable Correlation Generator). *Let $\mathcal{C}$ be a correlation generator. We say $\mathcal{C}$ is* reverse sampleable *if there exists a PPT algorithm* $\mathsf{RSample}$ *such that for $\sigma \in \{0,1\}$ the correlation obtained via:*

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

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

The following definition of pseudorandom correlation generators can be viewed as a generalization of the definition of the pseudorandom VOLE generator in [14]. Note though that we do not enforce perfect correctness.

Definition 9 (Pseudorandom Correlation Generator (PCG)). *Let $\mathcal{C}$ be a reverse-sampleable correlation generator. A* pseudorandom correlation generator (PCG) for $\mathcal{C}$ *is a pair of algorithms* $(\mathsf{PCG.Gen}, \mathsf{PCG.Expand})$ *with the following syntax:*

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

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

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

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

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

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

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

Note that the above definition is trivial to achieve in general: We can let $\mathsf{PCG.Gen}$ on input 1^λ return $(R_0, R_1) \leftarrow \mathcal{C}(1^\lambda)$, and simply define Expand to be the identity. Typically, we will be interested in non-trivial constructions of PCGs, in which the seed size is significantly shorter than the output size. A pseudorandom generator with image in $\{0,1\}^n$ is a simple example for an expanding PCG for the equality correlation $\{(R, R) \mid R \in \{0,1\}^n\}$. In the following we will be interested in constructing PCGs for a much broader class of correlations, like OT correlations, OLE correlations and (authenticated) Beaver triples.

4.2 Impossibility of a Simulation-Based Definition

A natural and useful alternative to the security definition we gave in Section 4, is the following: In any secure protocol (say against semi-honest adversaries), one can replace sampling a pair of strings from the correlation $\mathcal{C}$ by generating a pair of seeds (which are later expanded) using a PCG for $\mathcal{C}$ without compromising security. Unfortunately, as sketched in [38], a non-trivial PCG construction cannot satisfy such a simulation-based definition. Consider the simple protocol, where P_0 samples a pair $(R_0, R_1) \leftarrow \mathcal{C}(1^\lambda)$ and sends R_1 to P_1, who simply outputs R_1. This protocol obviously realizes the protocol dictated by $\mathcal{C}$, with one-sided security against P_1. But, if P_0 instead generates $(\mathsf{k}_0, \mathsf{k}_1)$ according to the seed generation algorithm of the PCG and sends k_1 to P_1, a possible simulator runs into the following problem. Simulating the above protocol given only the output R_1 corresponds to finding a short seed k_1 that can be (deterministically) expanded to R_1. If the entropy in the second output of $\mathcal{C}$ exceeds the seed-length $|\mathsf{k}_1|$, such a compression violates correctness, as it could be used to distinguish R_1 from a string that is indeed chosen via $\mathcal{C}$.

In the full version, we present a formal and more general version of the above argument for ruling out a simulation-based definition for non-trivial correlations, based on a lower bound of Hubáček and Wichs [45].

4.3 Applying PCGs in Protocols with Correlated Randomness

In this section we show that one *can* use PCGs in a "plug-and-play" fashion in protocols consuming correlated randomness sampled by a given functionality. More precisely, we show that PCGs can be directly applied to any protocol using a weaker form of correlated randomness, where corrupted parties can influence their outputs.

A simple example is random OT, where the weaker functionality we can realize allows a corrupt sender/receiver to choose its outputs, then the other party's outputs are sampled at random correspondingly. When using OT in an MPC protocol, the OT is typically implemented from random OT by masking the actual OT inputs with fresh random OT outputs. Allowing a corrupt party to choose its own OT outputs does not affect the security of these protocols, since (intuitively) this can only weaken security for the corrupt party and not for honest parties. More generally, it turns out that many practical MPC protocols, including those based on preprocessed multiplication triples for arithmetic

circuits [10, 29] and binary circuits [59, 61, 62], use this kind of corruptible, correlated randomness, since it is often easier to design a protocol that realizes this.

More formally, the randomness is modelled by the functionality $\mathcal{F}^{\mathcal{C}}_{\mathsf{corr}*}$ (Fig. 1), where a corrupted party may first *choose* its own output, and then the honest party's output is computed with the reverse sampling algorithm for $\mathcal{C}$. As we show in the following, PCGs can be used to securely realize $\mathcal{F}^{\mathcal{C}}_{\mathsf{corr}*}$, opening up many important applications at no extra cost.

To realize $\mathcal{F}^{\mathcal{C}}_{\mathsf{corr}*}$, we use a simple protocol, $\Pi^{\mathcal{C}}_{\mathsf{corr}*}$, that calls $\mathcal{F}^{\mathsf{PCG.Gen}}_{\mathsf{corr}}$ so that each party obtains a seed k_σ, which is then expanded to get the output $\mathsf{PCG.Expand}(\sigma, \mathsf{k}_\sigma)$.

Functionality $\mathcal{F}^{\mathcal{C}}_{\mathsf{corr}*}$

On input 1^λ, the functionality does as follows:

- If no parties are corrupt, sample $(R_0, R_1) \xleftarrow{\$} \mathcal{C}(1^\lambda)$.
- Otherwise, if P_σ is corrupt, wait to receive $R_\sigma \in \{0,1\}^{n_\sigma}$ from $\mathcal{A}$, then sample $R_{1-\sigma} \xleftarrow{\$} \mathsf{RSample}(\sigma, R_\sigma)$.

The functionality outputs R_0 to P_0 and R_1 to P_1, and then halts.

Fig. 1. Corruptible correlated randomness functionality for a reverse-sampleable correlation generator, $\mathcal{C}$

Theorem 10. *Let* $\mathsf{PCG} = (\mathsf{PCG.Gen}, \mathsf{PCG.Expand})$ *be a secure PCG for a reverse-sampleable correlation generator,* $\mathcal{C}$*. Then the protocol* $\Pi_{\mathsf{corr}*}$ *securely realizes the* $\mathcal{F}^{\mathcal{C}}_{\mathsf{corr}*}$ *functionality against a static, malicious adversary.*

5 Silent Oblivious Transfer Extension From LPN

In this section we present a protocol for silent OT extension, which allows to generate n instances of random OT with sublinear communication complexity. To this end, we first show how to tweak the construction of Boyle et al. [14] to give correlated OT. Combining this observation with the OT extension technique of Ishai et al. [46] we obtain a PCG for random OT. Finally, we show how to use the protocol of Doerner and shelat [32] for secure computation of the seed, giving sublinear OT extension.

5.1 Subfield Vector-OLE

Here, we introduce the notion of subfield vector oblivious linear evaluation (sVOLE), and show that sVOLE for $\mathbb{F}_q$ over subfield $\mathbb{F}_p \subset \mathbb{F}_q$ gives 1-out-of-p correlated OT. More precisely, a single big instance of sVOLE will give many

1-out-of-p OTs at once. Our construction of sVOLE comes with two additional advantages: It enjoys lower computational costs, because matrix multiplications are performed with a matrix over $\mathbb{F}_p$, and for $p = 2$ we can reduce security to the better-studied binary LPN problem, instead of its arithmetic variant over larger fields.

Subfield VOLE is a form of vector oblivious linear evaluation (VOLE) over $\mathbb{F}_q$, which computes $\boldsymbol{w} = \boldsymbol{u}x + \boldsymbol{v}$, where the vector $\boldsymbol{u}$ is restricted to lie over a subfield $\mathbb{F}_p \subset \mathbb{F}_q$, for $q = p^r$ (and we multiply $\boldsymbol{u}$ with $x \in \mathbb{F}_q$ component-wise, by viewing x as a vector over $\mathbb{F}_p$). It outputs $(\boldsymbol{u}, \boldsymbol{v})$ to the sender and $(x, \boldsymbol{w})$ to the receiver.

The construction in Fig. 2 uses the function $\mathsf{spread}_n(S, \boldsymbol{y})$, which expands a set $S = (s_1, \ldots, s_{|S|}) \subset [n]$ and a vector $\boldsymbol{y} \in \mathbb{F}_p^{|S|}$ into the vector $\boldsymbol{\mu} \in \mathbb{F}_p^n$, where $\mu_{s_i} = y_i$ for $i = 1, \ldots, |S|$, and $\mu_j = 0$ for $j \in [n] \setminus S$. It is a generalization of the VOLE generator from [14], which follows from the case $p = q$.

Construction G_{sVOLE}

Parameters:

- Security parameter 1^λ, integers $n' > n$, $q = p^r$, and noise weight t.
- A code generation algorithm $\mathbf{C}$ and $H_{n',n} \xleftarrow{\$} \mathbf{C}(n', n, \mathbb{F}_p)$.
- A multi-point FSS scheme $(\mathsf{MPFSS.Gen}, \mathsf{MPFSS.FullEval})$.

Correlation: Output $(\boldsymbol{u}, \boldsymbol{v})$ and $(x, \boldsymbol{w})$, where $x \leftarrow \mathbb{F}_q$, $\boldsymbol{u} \xleftarrow{\$} \mathbb{F}_p^n$, $\boldsymbol{v} \xleftarrow{\$} \mathbb{F}_q^n$ and $\boldsymbol{w} = \boldsymbol{u}x + \boldsymbol{v}$.

Gen: On input 1^λ:

1. Pick a random size-t subset S of $[n']$, sorted in increasing order.
2. Pick a random vector $\boldsymbol{y} \in (\mathbb{F}_p^*)^t$ and $x \xleftarrow{\$} \mathbb{F}_q$.
3. Compute $(K_0^{\mathsf{fss}}, K_1^{\mathsf{fss}}) \xleftarrow{\$} \mathsf{MPFSS.Gen}(1^\lambda, f_{S,x\cdot\boldsymbol{y}})$.
4. Let $\mathsf{k}_0 \leftarrow (m, n, K_0^{\mathsf{fss}}, S, \boldsymbol{y})$ and $\mathsf{k}_1 \leftarrow (m, n, K_1^{\mathsf{fss}}, x)$.
5. Output $(\mathsf{k}_0, \mathsf{k}_1)$.

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

1. If $\sigma = 0$: parse k_0 as $(m, n, K_0^{\mathsf{fss}}, S, \boldsymbol{y})$. Set $\boldsymbol{\mu} \leftarrow \mathsf{spread}_{n'}(S, \boldsymbol{y})$ in $\mathbb{F}_p^{n'}$. Compute $\boldsymbol{v}_0 \leftarrow \mathsf{MPFSS.FullEval}(0, K_0^{\mathsf{fss}})$ in $\mathbb{F}_q^{n'}$. Output $(\boldsymbol{u}, \boldsymbol{v}) \leftarrow (\boldsymbol{\mu} \cdot H_{n',n}, -\boldsymbol{v}_0 \cdot H_{n',n})$.
2. If $\sigma = 1$: parse k_1 as $(m, n, K_1^{\mathsf{fss}}, x)$. Compute $\boldsymbol{v}_1 \leftarrow \mathsf{MPFSS.FullEval}(1, K_1^{\mathsf{fss}})$ in $\mathbb{F}_q^{n'}$, and output $(x, \boldsymbol{w} \leftarrow \boldsymbol{v}_1 \cdot H_{n',n})$.

Fig. 2. PCG for subfield vector-OLE

Theorem 11. *Suppose the* $(\mathcal{HW}_t, \mathbf{C}, \mathbb{F}_p)$-dual-LPN$(n', n)$ *assumption holds, and that* MPFSS *is a secure multi-point FSS scheme. Then the construction* G_{sVOLE} *(Fig. 2) is a secure PCG for the subfield vector-OLE correlation.*

Application to Correlated OT. Subfield VOLE immediately gives a PCG for *correlated OT* (or Δ-OT). This is a batch of 1-out-of-2 OTs where the sender's strings are of the form $(w_i, w_i \oplus \Delta)$ for some fixed string Δ, and is the main building block in practical MPC protocols such as TinyOT [59] and authenticated garbling [61, 62].

To obtain correlated OT, we run subfield VOLE with $p = 2$ and $q = 2^r$, so the VOLE sender obtains $u_i \in \mathbb{F}_2, v_i \in \mathbb{F}_{2^r}$, while the VOLE receiver gets $x \in \mathbb{F}_{2^r}$ and $w_i = x \cdot u_i + v_i$, for $i = 1, \ldots, n$. Now switching the roles of sender and receiver, the VOLE sender can be seen as an OT receiver with choice bit u_i and string v_i. This gives us a correlated OT, since the OT sender (formerly VOLE receiver) can compute the strings $(w_i, w_i + x)$, and we have $v_i = w_i$ if $u_i = 0$ and $v_i = w_i + x$ if $u_i = 1$.

Application to Matrix Multiplication. Our construction for subfield VOLE can alternatively be seen as a PCG for *tensor product*: writing $x \in \mathbb{F}_q$ as $\boldsymbol{x} = (x_1, \ldots, x_r) \in \mathbb{F}_p^r$, and $\boldsymbol{u} = (u_1, \ldots, u_n) \in \mathbb{F}_p^n$, sVOLE computes secret shares of $\boldsymbol{x} \otimes \boldsymbol{u}$, that is, $x_i \cdot u_j$ for every $(i, j) \in [r] \times [n]$. This allows evaluation of secret-shared tensor products in 2-PC, which can in turn be used for matrix multiplication.

The seed size scales linearly in r, but this still improves upon the naive way of using r PCGs for VOLE over $\mathbb{F}_p$; the latter approach (with the VOLE from [14]) has seed size $O(rt \cdot (\lambda \log n + \log p))$ bits, whereas we reduce this to $O(t \cdot (\lambda \log n + r \log p))$ bits, saving at least a $\log n$ factor when $\log p = O(\lambda)$.

5.2 PCG for Random Oblivious Transfer

In the full version [15], we give the formal construction of a PCG for the random oblivious transfer correlation, based on G_{sVOLE}. Given the above observation that subfield VOLE implies correlated OT, this is straightforward, as we can apply the OT extension technique of Ishai et al. [46], which converts correlated OTs into random OTs using a suitable hash function. We extend this in a natural way to generate 1-out-of-p random OTs using subfield VOLE over $\mathbb{F}_p$. Note that for security when applying the hash function, we now need $q = \lambda^{\omega(1)}$.

We use a generalization of a correlation robust function, called $\mathbb{F}_p$*-correlation robustness* (defined in the full version). As recently shown in [41], this can be instantiated with fixed-key AES modeled as a random permutation when $p = 2$.

Theorem 12. *Suppose that* H *is an* $\mathbb{F}_p$*-correlation robust hash function and* G_{sVOLE} *is a secure PCG. Then the silent OT construction (in the full version) is a secure PCG for the random 1-out-of-p OT correlation.*

5.3 From a PCG to Silent OT Extension

To construct an OT extension protocol, we can use 2-PC to securely compute the Gen algorithm of G_{OT}, and then have each party locally expand its output using $G_{\mathsf{OT}}.\mathsf{Expand}$. Applying Theorem 10 from Section 4.3, this realizes a *corruptible* form of the ideal functionality for random oblivious transfer, where corrupt parties may influence their random outputs.

To do this efficiently with semi-honest security, we use the black-box protocol of Doerner and shelat [32] (also used in [14]) for setting up distributed point function keys. For a single point function of domain size n, this requires $O(\log n)$ OTs on $O(\lambda)$-bit strings, giving $O(t \log n)$ OTs for a multi-bit point function. Implementing each OT with (non-silent) OT extension [46] costs $O(\lambda)$ bits of communication, plus a setup phase of λ base OTs. Putting this together, we obtain the following.

Theorem 13. *Suppose the* $(\mathcal{HW}_t, \mathbf{C}, \mathbb{F}_p)$-$\mathsf{dual\text{-}LPN}(n', n)$ *assumption holds, and an* $\mathbb{F}_p$*-correlation robust hash function exists. Then there is a protocol that uses* $O(\lambda)$ *1-out-of-2 OTs to realize* n *instances of random 1-out-of-*p *OT with semi-honest security, using* $O(t\lambda \log n) + \mathsf{poly}(\lambda)$ *bits of communication.*

We remark that this gives OT with *sublinear communication* when $t = o(n/(\lambda \log n))$, which translates to an instance of LPN with noise rate $1/\omega(\lambda \log n)$. If the matrix $H_{n',n}$ in G_{sVOLE} is uniformly random, the computational complexity is dominated by $O(n' \cdot n)$ arithmetic operations; using more structured matrices based on LDPC codes or quasi-cyclic codes, we get respective costs of $O(n')$ or $\tilde{O}(n')$ arithmetic and PRG operations.

Concrete Efficiency. In the full version, we analyze these costs more concretely and give a breakdown of the communication complexity, as well as some approximate runtime estimates based on the cost of the main operations. For example, for $n \leq 2^{22}$ OTs, the PCG seed size is under 10kB and requires less than 30kB of communication to create with the distributed setup procedure. After setup, we estimate that these seeds can be expanded into 16MB of OTs on 128-bit strings at a rate of around 1 million per second, or 2 million per second when expanding to 1MB, using a single core of a CPU on a modern laptop. When including the distributed setup procedure, in these two cases we get an amortized communication complexity of just 0.2 and 2.6 bits per OT, respectively.

Acknowledgements. We would like to thank Peter Rindal and Melissa Rossi for helpful discussions and pointers, and the anonymous Crypto 2019 reviewers for their comments.

E. Boyle, N. Gilboa, and Y. Ishai supported by ERC Project NTSC (742754). E. Boyle additionally supported by ISF grant 1861/16 and AFOSR Award FA9550-17-1-0069. G. Couteau supported by ERC Project PREP-CRYPTO (724307).

N. Gilboa additionally supported by ISF grant 1638/15 and a grant by the BGU Cyber Center. Y. Ishai additionally supported by ISF grant 1709/14, NSF-BSF grant 2015782, and a grant from the Ministry of Science and Technology, Israel and Department of Science and Technology, Government of India. L. Kohl supported by ERC Project PREP-CRYPTO (724307), by DFG grant HO 4534/2-2 and by a DAAD scholarship. This work was done in part while visiting the FACT Center at IDC Herzliya, Israel. P. Scholl supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No 731583 (SODA), and the Danish Independent Research Council under Grant-ID DFF-6108-00169 (FoCC).

References

1. Aguilar, C., Blazy, O., Deneuville, J.C., Gaborit, P., Zémor, G.: Efficient encryption from random quasi-cyclic codes. Cryptology ePrint Archive, Report 2016/1194 (2016), http://eprint.iacr.org/2016/1194
2. Alekhnovich, M.: More on average case vs approximation complexity. In: 44th FOCS. IEEE Computer Society Press (Oct 2003)
3. Applebaum, B., Damgård, I., Ishai, Y., Nielsen, M., Zichron, L.: Secure arithmetic computation with constant computational overhead. In: CRYPTO 2017, Part I. LNCS, Springer, Heidelberg (Aug 2017)
4. Applebaum, B., Haramaty, N., Ishai, Y., Kushilevitz, E., Vaikuntanathan, V.: Low-complexity cryptographic hash functions. In: ITCS 2017. LIPIcs (Jan 2017)
5. Applebaum, B., Ishai, Y., Kushilevitz, E.: Cryptography with constant input locality. Journal of Cryptology (4) (Oct 2009)
6. Arora, S., Ge, R.: New algorithms for learning in presence of errors. In: ICALP 2011, Part I. LNCS, Springer, Heidelberg (Jul 2011)
7. Asharov, G., Lindell, Y., Schneider, T., Zohner, M.: More efficient oblivious transfer and extensions for faster secure computation. In: ACM CCS 2013. ACM Press (Nov 2013)
8. Beaver, D.: Efficient multiparty protocols using circuit randomization. In: Feigenbaum, J. (ed.) Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings. Lecture Notes in Computer Science, vol. 576, pp. 420–432. Springer (1991), https://doi.org/10.1007/3-540-46766-1_34
9. Beaver, D.: Correlated pseudorandomness and the complexity of private computations. In: Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996. pp. 479–488 (1996), https://doi.org/10.1145/237814.237996
10. Bendlin, R., Damgård, I., Orlandi, C., Zakarias, S.: Semi-homomorphic encryption and multiparty computation. In: EUROCRYPT 2011. LNCS, Springer, Heidelberg (May 2011)
11. Berbain, C., Gilbert, H., Patarin, J.: QUAD: A practical stream cipher with provable security. In: EUROCRYPT 2006. LNCS, Springer, Heidelberg (May / Jun 2006)
12. Blum, A., Furst, M.L., Kearns, M.J., Lipton, R.J.: Cryptographic primitives based on hard learning problems. In: Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology Conference, Santa Barbara, California, USA, August 22-26, 1993, Proceedings. pp. 278–291 (1993), https://doi.org/10.1007/3-540-48329-2_24

13. Boneh, D., Goh, E.J., Nissim, K.: Evaluating 2-DNF formulas on ciphertexts. In: TCC 2005. LNCS, Springer, Heidelberg (Feb 2005)
14. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y.: Compressing vector OLE. In: ACM CCS 2018. ACM Press (Oct 2018)
15. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y., Kohl, L., Scholl, P.: Efficient pseudorandom correlation generators: Silent ot extension and more. Cryptology ePrint Archive, Report 2019/448 (2019), https://eprint.iacr.org/2019/448
16. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y., Orrù, M.: Homomorphic secret sharing: Optimizations and applications. In: ACM CCS 2017. ACM Press (Oct / Nov 2017)
17. Boyle, E., Gilboa, N., Ishai, Y.: Function secret sharing. In: EUROCRYPT 2015, Part II. LNCS, Springer, Heidelberg (Apr 2015)
18. Boyle, E., Gilboa, N., Ishai, Y.: Breaking the circuit size barrier for secure computation under DDH. In: CRYPTO 2016, Part I. LNCS, Springer, Heidelberg (Aug 2016)
19. Boyle, E., Gilboa, N., Ishai, Y.: Function secret sharing: Improvements and extensions. In: ACM CCS 2016. ACM Press (Oct 2016)
20. Boyle, E., Gilboa, N., Ishai, Y.: Group-based secure computation: Optimizing rounds, communication, and computation. In: EUROCRYPT 2017, Part II. LNCS, Springer, Heidelberg (Apr / May 2017)
21. Boyle, E., Gilboa, N., Ishai, Y., Lin, H., Tessaro, S.: Foundations of homomorphic secret sharing. In: 9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA. pp. 21:1–21:21 (2018), https://doi.org/10.4230/LIPIcs.ITCS.2018.21
22. Boyle, E., Kohl, L., Scholl, P.: Homomorphic secret sharing from lattices without fhe. In: EUROCRYPT '19 (2019), https://eprint.iacr.org/2019/129
23. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. In: ITCS 2012. ACM (Jan 2012)
24. Chase, M., Dodis, Y., Ishai, Y., Kraschewski, D., Liu, T., Ostrovsky, R., Vaikuntanathan, V.: Reusable non-interactive secure computation. IACR Cryptology ePrint Archive 2018, 940 (2018), https://eprint.iacr.org/2018/940
25. Couteau, G.: A note on the communication complexity of multiparty computation in the correlated randomness model. In: Advances in Cryptology - EUROCRYPT. Springer (2019)
26. Cramer, R., Damgård, I., Ishai, Y.: Share conversion, pseudorandom secret-sharing and applications to secure computation. In: TCC 2005. LNCS, Springer, Heidelberg (Feb 2005)
27. Cramer, R., Damgård, I., Nielsen, J.B.: Multiparty computation from threshold homomorphic encryption. In: Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001, Proceeding. pp. 280–299 (2001), https://doi.org/10.1007/3-540-44987-6_18
28. Damgård, I., Nielsen, J.B., Nielsen, M., Ranellucci, S.: The TinyTable protocol for 2-party secure computation, or: Gate-scrambling revisited. In: CRYPTO 2017, Part I. LNCS, Springer, Heidelberg (Aug 2017)
29. Damgård, I., Pastro, V., Smart, N.P., Zakarias, S.: Multiparty computation from somewhat homomorphic encryption. In: CRYPTO 2012. LNCS, Springer, Heidelberg (Aug 2012)
30. Dessouky, G., Koushanfar, F., Sadeghi, A.R., Schneider, T., Zeitouni, S., Zohner, M.: Pushing the communication barrier in secure computation using lookup tables. In: NDSS 2017. The Internet Society (Feb / Mar 2017)

31. Dodis, Y., Halevi, S., Rothblum, R.D., Wichs, D.: Spooky encryption and its applications. In: CRYPTO 2016, Part III. LNCS, Springer, Heidelberg (Aug 2016)
32. Doerner, J., shelat, a.: Scaling ORAM for secure computation. In: ACM CCS 2017. ACM Press (Oct / Nov 2017)
33. Döttling, N., Ghosh, S., Nielsen, J.B., Nilges, T., Trifiletti, R.: TinyOLE: Efficient actively secure two-party computation from oblivious linear function evaluation. In: ACM CCS 2017. ACM Press (Oct / Nov 2017)
34. Druk, E., Ishai, Y.: Linear-time encodable codes meeting the gilbert-varshamov bound and their cryptographic applications. In: ITCS 2014. ACM (Jan 2014)
35. Franklin, M.K., Haber, S.: Joint encryption and message-efficient secure computation. J. Cryptology 9(4), 217–232 (1996), https://doi.org/10.1007/BF00189261
36. Freedman, M.J., Ishai, Y., Pinkas, B., Reingold, O.: Keyword search and oblivious pseudorandom functions. In: TCC 2005. LNCS, Springer, Heidelberg (Feb 2005)
37. Ghosh, S., Nielsen, J.B., Nilges, T.: Maliciously secure oblivious linear function evaluation with constant overhead. In: ASIACRYPT 2017, Part I. LNCS, Springer, Heidelberg (Dec 2017)
38. Gilboa, N., Ishai, Y.: Compressing cryptographic resources. In: CRYPTO'99. LNCS, Springer, Heidelberg (Aug 1999)
39. Gilboa, N., Ishai, Y.: Distributed point functions and their applications. In: EUROCRYPT 2014. LNCS, Springer, Heidelberg (May 2014)
40. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or A completeness theorem for protocols with honest majority. In: 19th ACM STOC. ACM Press (May 1987)
41. Guo, C., Katz, J., Wang, X., Yu, Y.: Efficient and secure multiparty computation from fixed-key block ciphers. Cryptology ePrint Archive, Report 2019/074 (2019), https://eprint.iacr.org/2019/074
42. Halevi, S., Ishai, Y., Jain, A., Kushilevitz, E., Rabin, T.: Secure multiparty computation with general interaction patterns. In: ITCS 2016. ACM (Jan 2016)
43. Hazay, C., Orsini, E., Scholl, P., Soria-Vazquez, E.: TinyKeys: A new approach to efficient multi-party computation. In: CRYPTO 2018, Part III. LNCS, Springer, Heidelberg (Aug 2018)
44. Heyse, S., Kiltz, E., Lyubashevsky, V., Paar, C., Pietrzak, K.: Lapin: An efficient authentication protocol based on ring-LPN. In: FSE 2012. pp. 346–365 (2012)
45. Hubacek, P., Wichs, D.: On the communication complexity of secure function evaluation with long output. In: ITCS 2015. ACM (Jan 2015)
46. Ishai, Y., Kilian, J., Nissim, K., Petrank, E.: Extending oblivious transfers efficiently. In: CRYPTO 2003. LNCS, Springer, Heidelberg (Aug 2003)
47. Ishai, Y., Kushilevitz, E., Meldgaard, S., Orlandi, C., Paskin-Cherniavsky, A.: On the power of correlated randomness in secure computation. In: TCC 2013. LNCS, Springer, Heidelberg (Mar 2013)
48. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Batch codes and their applications. In: 36th ACM STOC. ACM Press (Jun 2004)
49. Ishai, Y., Prabhakaran, M., Sahai, A.: Founding cryptography on oblivious transfer - efficiently. In: Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings. pp. 572–591 (2008), https://doi.org/10.1007/978-3-540-85174-5_32
50. Ishai, Y., Prabhakaran, M., Sahai, A.: Secure arithmetic computation with no honest majority. In: TCC 2009. LNCS, Springer, Heidelberg (Mar 2009)

51. Katz, J., Ranellucci, S., Rosulek, M., Wang, X.: Optimizing authenticated garbling for faster secure two-party computation. In: Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part III. pp. 365–391 (2018), https://doi.org/10.1007/978-3-319-96878-0_13
52. 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: ACNS 17. LNCS, Springer, Heidelberg (Jul 2017)
53. Keller, M., Pastro, V., Rotaru, D.: Overdrive: Making SPDZ great again. In: EUROCRYPT 2018, Part III. LNCS, Springer, Heidelberg (Apr / May 2018)
54. Kilian, J.: Founding cryptography on oblivious transfer. In: Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA. pp. 20–31 (1988), https://doi.org/10.1145/62212.62215
55. Kolesnikov, V., Kumaresan, R.: Improved OT extension for transferring short secrets. In: CRYPTO 2013, Part II. LNCS, Springer, Heidelberg (Aug 2013)
56. Kolesnikov, V., Kumaresan, R., Rosulek, M., Trieu, N.: Efficient batched oblivious PRF with applications to private set intersection. In: ACM CCS 2016. ACM Press (Oct 2016)
57. Matsumoto, T., Imai, H.: Public quadratic polynominal-tuples for efficient signature-verification and message-encryption. In: EUROCRYPT'88. LNCS, Springer, Heidelberg (May 1988)
58. Naor, M., Pinkas, B.: Oblivious polynomial evaluation. SIAM J. Comput. 35(5), 1254–1281 (2006)
59. Nielsen, J.B., Nordholt, P.S., Orlandi, C., Burra, S.S.: A new approach to practical active-secure two-party computation. In: CRYPTO 2012. LNCS, Springer, Heidelberg (Aug 2012)
60. Scholl, P.: Extending oblivious transfer with low communication via key-homomorphic PRFs. In: PKC 2018, Part I. LNCS, Springer, Heidelberg (Mar 2018)
61. Wang, X., Ranellucci, S., Katz, J.: Authenticated garbling and efficient maliciously secure two-party computation. In: ACM CCS 2017. ACM Press (Oct / Nov 2017)
62. Wang, X., Ranellucci, S., Katz, J.: Global-scale secure multiparty computation. In: ACM CCS 2017. ACM Press (Oct / Nov 2017)
63. Wolf, C.: Multivariate quadratic polynomials in public key cryptography. Cryptology ePrint Archive, Report 2005/393 (2005), http://eprint.iacr.org/2005/393