Compressing Correlations via Secret Replication: PCFs from Symmetric Cryptography*

Yuval Ishai† Hugo Krawczyk‡ Tal Rabin§

July 1, 2026

Abstract

We revisit the question of securely compressing multiparty correlations using only symmetric cryptography. A *linear correlation* $\mathcal{C}$, defined by a linear subspace $C \subseteq \mathbb{F}^n$, samples a secret random $\mathbf{c} \in C$ and assigns to each party a fixed subset of the entries of $\mathbf{c}$. Gilboa and Ishai (Crypto 1999) and Cramer, Damgård and Ishai (TCC 2005) provide a general technique for securely compressing many independent samples from $\mathcal{C}$ by replicating independent keys of a pseudorandom function (PRF) among the parties. This implies a *pseudorandom correlation function* (PCF) for $\mathcal{C}$ from any PRF, where the PCF key size scales with the number of minimal-support codewords in C.

We observe that the above generalizes to other types of useful target correlations $\mathcal{C}_T$ by using a *secret* replication pattern obtained via a random secret assignment of parties in $\mathcal{C}$ to parties in $\mathcal{C}_T$.

We present several corollaries of this general blueprint. These include a re-derivation of two-party PCF constructions for VOLE and subfield-VOLE over small domains (Roy, Crypto 2022) as well as new multiparty PCFs for small-domain VOLE-style correlations, including scalar-vector multiplication triples and their authenticated variants. Finally, we discuss applications to secure computation.

1 Introduction

Never Underestimate the Power of Replication.

Jacob Jacobson

Secure multiparty computation (MPC) without an honest majority inherently requires public-key cryptography. However, this is no longer the case if the parties have access to secret, correlated randomness. That is, we imagine a trusted dealer who generates a sample $(r_1, \ldots, r_n)$ from a correlated randomness distribution $\mathcal{C}$, and sends each r_i to party P_i before the inputs are known. Many MPC protocols can benefit from vast amounts of

*This is a full version of [32].

†Technion and Amazon Web Services, yuval.ishai@gmail.com. Work done while at AWS.

‡Amazon Web Services, hugokraw@gmail.com.

§Amazon Web Services, rabintal@amazon.com.

such correlated randomness, generated during a preprocessing phase, to realize fast and "information-theoretic" online protocols [5, 6, 24].

To mitigate the communication costs and trust assumptions associated with distributing this correlated randomness directly by an external party, pseudorandom correlation generators (PCGs) and pseudorandom correlation functions (PCFs) enable a *secure compression* of correlated randomness. A PCG [12, 14] allows the parties to locally expand short, correlated keys into a polynomially larger source of correlated pseudorandomness, whereas a PCF [15] extends this to enable random access to a virtually unlimited source. PCGs and PCFs are the correlated randomness analogues of pseudorandom generators (PRGs) and pseudorandom functions (PRFs), respectively. Crucially, the correlated keys viewed by any subset of the parties should not reveal more information about the outputs of the remaining parties beyond what the ideal correlation allows. The correlated keys can be securely generated by an offline protocol, whose cost is amortized away in the online phase. While the combined offline/online protocol is no longer information-theoretic, it still inherits the simplicity and low online cost of a protocol that has an ideal access to $\mathcal{C}$.

The above paradigm motivates the design of PCGs and PCFs for useful multiparty correlations under the weakest possible assumptions and with the best concrete efficiency. While this is a very active area of research (see Section 1.4), current results still leave much room for improvement. Most constructions for useful correlations, such as random oblivious transfer, are based on different flavors of the Learning Parity with Noise (LPN) assumption [12, 14, 15, 19] or alternatively on public-key cryptography assumptions based on lattices [25, 14, 18] or groups [41, 20]. Since a PCG for nontrivial correlations implies a PRG, it is natural to ask when the converse is true:

Which correlations can be compressed using symmetric cryptography?

Here, the term "symmetric cryptography" can stand for a PRG or a PRF, which from a feasibility viewpoint are both equivalent to a one-way function [29, 31].[1]

For the case of two-party correlations, Boyle et al. [14] show that any PRG implies a PCG (in fact, a PCF) for a single long instance of a *unit-vector correlation*, an additively shared vector containing 1 in a random position and 0's elsewhere. Roy [42] obtained a similar result for a *vector oblivious linear evaluation* (VOLE) correlation over a small field $\mathbb{F}$, where a "sender" party gets random long vectors $\mathbf{a}, \mathbf{b} \in \mathbb{F}^N$ and a "receiver" party gets a random scalar $x \in \mathbb{F}$ along with $\mathbf{a}x + \mathbf{b}$. This was generalized to the case where only the scalar x comes from a small domain [38], and used as a basis for low-communication oblivious transfer [42], non-interactive zero-knowledge proofs and post-quantum digital signatures [4, 3], threshold signatures [38, 37], and arithmetic garbling [33]. These isolated positive results call for a broader study of the possibility of compressing useful correlations using only symmetric cryptography.

1.1 Our Contribution

We obtain feasibility results for compressing certain types of correlated randomness using only symmetric cryptography. These are motivated by several applications we discuss in Sec-

[1] A broader interpretation of this term allows for the use of a random oracle. Here we will only use assumptions equivalent to a one-way function.

tion 1.2 below. Our results include a conceptually simple explanation for the VOLE-related results from [42] discussed above, as well as solutions for other useful related correlations that were not covered by previous works.

The results are based on a new general framework that utilizes *secure information-theoretic reductions* between correlations. Our main observation is that several useful and inherently nonlinear target correlations $\mathcal{C}_T$ can be compressed via a perfectly secure non-interactive reduction to a *linear* source correlation $\mathcal{C}_S$, namely one obtained by drawing a random secret codeword $\mathbf{c}$ from a publicly known linear code $C_S \subseteq \mathbb{F}^n$ and assigning a coordinate of $\mathbf{c}$ to each of the n parties. More generally, a linear correlation can assign any set of coordinates to each party, as long as this set is fixed and public.

The core mechanism of this reduction is a random projection, which assigns to each party in $\mathcal{C}_T$ a *secretly chosen* subset of virtual parties from the source correlation $\mathcal{C}_S$. We combine such a projection reduction with an existing PCF for linear correlations (Lin-PCF) due to Gilboa and Ishai [26] and Cramer, Damgård and Ishai [23] (see Section 1.4), which is based on a public replication of secret PRF keys. Using this combination, we obtain PCFs for useful *nonlinear* correlations that use *secretly replicated* PRF keys.

Overview of results. We present several concrete instances of the above general blueprint, revisiting previous results and expanding the class of useful correlations that can be compressed using symmetric cryptography. Our positive results apply to different flavors of "VOLE-style" correlations, namely ones that support secure scalar-vector multiplication. While their time complexity is polylogarithmic in the field size, it is polynomial in the scalar domain. This restricts scalars to come from a feasible-size domain. Concretely, our main results are summarized as follows:

- **Two-party VOLE:** We re-derive the previous PCF from [42, 38] for the VOLE correlation, also allowing the scalar to be restricted to any subset $X \subseteq \mathbb{F}$. This relies on an n-party linear correlation $\mathcal{C}_S$ obtained from a Reed-Solomon code of length $n = |X|$ defined by the evaluations of lines $L_{a,b}(\alpha) = a\alpha + b$ on all $\alpha \in X$. The evaluation time of this PCF scales linearly with $|X|$. We extend this to the subfield variant of VOLE, where $\mathbf{a}$ is over a subfield $\mathbb{F}'$ [14, 42].

- **Multiparty SVMT:** We obtain the first PCFs for *multiparty* VOLE-style correlations. Concretely, a k-party *scalar-vector multiplication triple* (SVMT) additively shares between k parties a random vector $\mathbf{a} \in \mathbb{F}^N$, scalar x with shares in $X \subseteq \mathbb{F}$, and their product $\mathbf{a}x$. This is the scalar-vector analogue of a standard Beaver-triple correlation [5]. Here we use a linear correlation $\mathcal{C}_S$ defined by a "multi-line" code of length $n = (k-1) \cdot |X|$, consisting of evaluations of $k-1$ lines L_{a,b_i} with a common slope a on all $\alpha \in X$. Our PCF provides random access to any instance of the SVMT correlation using $O(k \cdot |X|^{k-1})$ PRF calls.

- **Authenticated correlations:** To support authenticated VOLE-style correlations, the vector $\mathbf{a} \in \mathbb{F}^N$ needs to be multiplied by a second scalar y whose additive shares are taken from $Y \subseteq \mathbb{F}$. Our SVMT construction extends to this case, replacing $|X|$ by $|X| \cdot |Y|$ in the cost analysis.

1.2 Applications

Our new PCF constructions naturally support MPC protocols that consume VOLE-style correlations. However, one should account for the limitations on the scalar domain size and the number of parties imposed by our constructions. While these limitations rule out some use cases, many others remain.

Our PCF for k-party SVMT is useful for MPC tasks in which the same secret scalar is multiplied by many secret values or a long secret vector. This is a very common scenario. For example, it applies to any computation that involves matrix-vector multiplications or repeated cryptographic operations with a fixed key. The $|X|^{k-1}$ computational overhead is tolerable when the number of parties k is small and scalars can be taken from a small domain X. For example, the latter applies to standard secret-sharing based MPC for Boolean circuits, where we have $|X| = |\mathbb{F}| = 2$.

The usefulness of small-domain VOLE goes beyond computations over a small domain. This was already demonstrated, for the case of standard 2-party VOLE, by applications to OT extension [42] and threshold cryptography [37], which increase the domain size exponentially via repetition or CRT representation.

Some other applications *natively* involve multiplications of vectors over a large field $\mathbb{F}$ with scalars from a small subset $X \subseteq \mathbb{F}$. For example, in a recent protocol for computing *encrypted matrix-vector product* (EMVP) [8], a client decrypts the product by multiplying a public matrix with a secret decryption vector. Each entry of this vector can potentially come from a small domain, as long as the entire vector has sufficient entropy. Our PCF for k-party SVMT can help distribute an EMVP client.

Finally, while our authenticated SVMT variant is even more limited because of the additional overhead depending on $|Y|$, a small Y suffices for achieving a notion of "covert security" that may provide sufficient deterrence against cheating [2, 38]. See Section 5 for additional discussion of applications.

1.3 Overview of Techniques

Our results use an information-theoretic *projection reduction* of a k-party VOLE-style target correlation $\mathcal{C}_T$ to N instances of an n-party *linear* source correlation $\mathcal{C}_S$, for some $n > k$. This projection reduction can assign each of the k parties in $\mathcal{C}_T$ to a secretly chosen subset of the n parties in $\mathcal{C}_S$, and have each of the k parties apply some local post-processing. Using the Lin-PCF for linear correlations [26, 23], we obtain a PCF for $\mathcal{C}_T$ that involves a *secret replication* of PRF keys.

Beyond the simple special case of generating additive shares of 0, which is used extensively in the context of secure federated learning [9], linear correlations were mostly used for threshold cryptography and general MPC in the *honest-majority* setting [23, 10]. In particular, linear correlations are trivial in the two-party setting. Our results give the first nontrivial applications of linear correlations in the two-party and dishonest-majority settings.

The efficiency of our constructions is closely linked to the specifics of the Lin-PCF, which we briefly explain here. Recall that an instance of an n-party linear correlation $\mathcal{C}_S$ is distributed uniformly over a linear code $C_S \subseteq \mathbb{F}^n$. The Lin-PCF can generate a virtually unbounded number of instances of $\mathcal{C}$ by replicating independently chosen PRF keys between

the parties. The replication pattern is determined by a public collection of sets $H \subseteq [n]$. The collection is defined by the *minimal-support sets* $W(C_S)$: For each minimal nonempty set $H \subseteq [n]$ such that there is a codeword $\mathbf{c} \in C_S$ for which H is the set of nonzero coordinates, the set H is in $W(C_S)$. The Lin-PCF distributes the PRF key r_H to each party in H. To generate an instance of $\mathcal{C}_S$, each party evaluates the PRF on its keys, obtaining a fresh field element for each key, and applies a linear combination determined by C_S to the resulting elements. The time required to generate an instance of $\mathcal{C}_S$ using the PCF thus scales linearly with $|W(C_S)|$.

See Figure 1 for a pictorial overview of the combination of the Lin-PCF for $\mathcal{C}_S$ with the secret assignment induced by our projection reduction from $\mathcal{C}_T$ to $\mathcal{C}_S$.

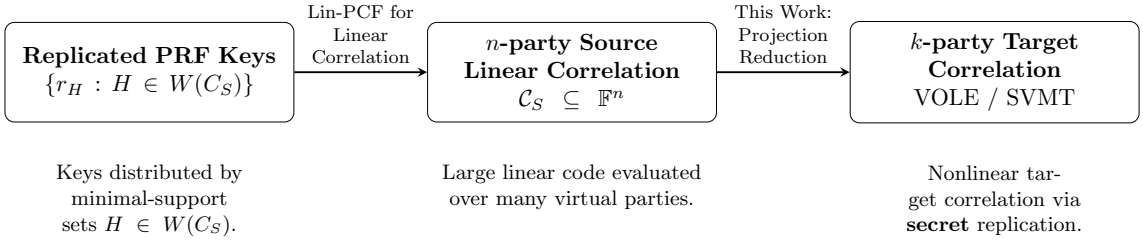


Figure 1: Overview of technique for PCF via secret replication.

We now describe the concrete projection reductions used for obtaining our PCF for vector oblivious linear evaluation (VOLE) and its multiparty generalization to scalar-vector multiplication triples (SVMT).

1.3.1 Small-Domain VOLE from Line Correlation

Recall that in a 2-party VOLE correlation over $\mathbb{F}$, the sender gets a random vector $\mathbf{a} \in \mathbb{F}^N$ and an intercept vector $\mathbf{b} \in \mathbb{F}^N$. The receiver gets a random scalar $x \in X$ (for scalar domain $X \subseteq \mathbb{F}$) and the evaluated vector $\mathbf{y} = \mathbf{a}x + \mathbf{b} \in \mathbb{F}^N$.

To compress this inherently nonlinear correlation, we use the linear source correlation $\mathcal{C}_S$ of a Reed-Solomon code C_S of length $n = |X|$ defined by degree-1 polynomials evaluated on X. The N instances of $\mathcal{C}_S$ can be viewed as the evaluations of N random lines $L(\alpha) = \mathbf{a}\alpha + \mathbf{b}$ at all points in X. The secret assignment then works as follows:

- **The Sender** is assigned the entire set of parties $[n] = X$. By observing the evaluations at all points in X, the sender can easily interpolate the N lines to recover the slope $\mathbf{a}$ and the intercept $\mathbf{b}$.
- **The Receiver** is assigned only a single, secret coordinate $\{x\}$. By reading the evaluation of the lines at this specific point, the receiver directly obtains $\mathbf{y} = \mathbf{a}x + \mathbf{b}$.

Because the receiver's coordinate x is completely hidden from the sender, this constitutes a perfectly secure reduction. Combined with the Lin-PCF for $\mathcal{C}_S$, the number of PRF keys distributed to each party corresponds to the number of minimal-support sets $H \in W(C_S)$, which is exactly $|X|$. Indeed, for each $x \in X$, the set $H = X \setminus \{x\}$ supports the nonzero lines L such that $L(x) = 0$, and no strict subset of H can support a nonzero line.

Theorem 1.1 (Informal: PCF for small-domain VOLE)**.** *Let $\mathbb{F}$ be a finite field and $X \subseteq \mathbb{F}$ be a scalar domain. There exists a 2-party PCF for VOLE with scalar $x \in X$ relying solely on the existence of PRFs. In this PCF, the sender holds $|X|$ PRF keys, the receiver holds $|X| - 1$ PRF keys. The PCF evaluation involves a single PRF evaluation for each key.*

As in [42], the PCF keys can be compressed via the use of a puncturable PRF [11, 35, 17], where the sender only holds the root of a GGM tree [29] and the receiver holds a punctured key. Moreover, to distribute the PCF key generation between the two parties, one can use concretely efficient protocols that make a black-box use of a PRG [13, 43, 42]. Similar optimizations and supporting protocols apply to other PCFs in this work, but we focus here on the core PCF results for simplicity.

Finally, similarly to [42], Theorem 1.1 can be extended to the case of *subfield* VOLE, in which $\mathbf{a} \in (\mathbb{F}')^N$ for a subfield $\mathbb{F}' \subset \mathbb{F}$, where typically $\mathbb{F}' = \mathbb{F}_2$. To minimize the PCF cost in this case, we utilize a generalization of the Lin-PCF construction to the case where each party can own multiple coordinates in a codeword. This generalized case was implicitly considered in [23] in the context of converting between secret-sharing schemes.

1.3.2 SVMT from Multi-Line Correlation

We now consider the more challenging case of a k-party scalar-vector multiplication triple (SVMT) correlation. Here, k parties hold additive secret shares of a random vector $\mathbf{a} \in \mathbb{F}^N$, scalar $x \in \mathbb{F}$, and their product $\mathbf{a}x$. As before, we will sometimes restrict each share of x to come from a smaller domain $X \subseteq \mathbb{F}$.

In the two-party case ($k = 2$), one can directly reduce SVMT to two instances of VOLE via a standard reduction (cf. [12]): Letting the sharing of $\mathbf{a}$ and x be $\mathbf{a} = \mathbf{a}_0 + \mathbf{a}_1$ and $x = x_0 + x_1$, we assign to each party i a random vector $\mathbf{a}_i$ and scalar x_i. To compute the additive shares of $\mathbf{a}x = \mathbf{a}_0x_0 + \mathbf{a}_0x_1 + \mathbf{a}_1x_0 + \mathbf{a}_1x_1$, each party i can locally compute $\mathbf{a}_ix_i$. To additively share the cross-terms $\mathbf{a}_0x_1$ and $\mathbf{a}_1x_0$, the parties jointly use two VOLE instances. Now each party i can locally add its shares of the 2 cross-terms given by the VOLE instances with its locally computed share, yielding an additive sharing of $\mathbf{a}x$.

Extending this to $k \geq 3$ parties seems challenging. A direct generalization would require $k \cdot (k-1)$ instances of VOLE to share the cross-terms $\mathbf{a}_ix_j$, but the values of the vector $\mathbf{a}_i$ used by a VOLE sender to multiply different receiver scalars x_j must be identical to ensure the cross-terms sum correctly to $\mathbf{a}x$. The repeated use of the same $\mathbf{a}_i$ in VOLE invocations with different parties, often referred to as "programmability" in the PCG context [15], is not supported by our PCF for VOLE.

A natural way around this would be for a sender party i to use a single source linear correlation representing one line, $L(\alpha) = \mathbf{a}_i\alpha + \mathbf{b}_i$, and have all other $k-1$ parties, acting as VOLE receivers, evaluate this exact same line at their respective scalar shares x_j. However, this approach critically fails to provide security against colluding parties. Because a degree-1 polynomial is uniquely determined by just two points, any coalition of two or more receiving parties could pool their evaluations, interpolate the line, and easily extract the shared slope $\mathbf{a}_i$, thereby entirely compromising party i's secret vector share.

An alternative approach would be to avoid the use of pairwise VOLE and attempt at a direct generalization using polynomials p of degree $< k$. For instance, an entry a of $\mathbf{a}$ can be encoded by a random p with a as its free coefficient, thereby hiding a from any strict

subset of parties. The projection reduction, given the scalar x, can then assign to the parties random distinct interpolation points x_i along with interpolation coefficients λ_i, subject to the constraint that for any polynomial p with free coefficient a, we have $\sum \lambda_i p(x_i) = ax$. Then, each party can output $\lambda_i \cdot p(x_i)$ as a correct additive share of ax. While this reduction achieves perfect correctness and perfectly hides a, it does not hide x.

To get around the above difficulties, we go back to a line-based solution, but use $k-1$ "fresh" lines with the same slope to encode each entry of $\mathbf{a}$. Concretely, define a linear source correlation $\mathcal{C}_S$ consisting of k independent blocks. Block i is assigned to party i acting as the "sender" for its vector share $\mathbf{a}_i$. To enforce consistency, block i consists of the evaluations of $k-1$ distinct lines that all share the *same* slope $\mathbf{a}_i$, but have independent intercepts $\mathbf{b}_{i\to j}$ for each receiver $j \neq i$. The secret assignment now proceeds as follows:

- **As a Sender:** Party i is assigned the entirety of block i. It interpolates the $k-1$ lines to extract its vector share $\mathbf{a}_i$ and the $k-1$ intercepts $\mathbf{b}_{i\to j}$.
- **As a Receiver:** In every other block $\ell \neq i$, party i is assigned a single coordinate corresponding to its secret scalar share x_i. It evaluates the line designated for it in block ℓ at the point x_i, yielding the cross-term $\mathbf{a}_\ell x_i + \mathbf{b}_{\ell\to i}$.

Applying the Lin-PCF to this specific block structure yields a combinatorial blow-up in the minimal-support sets due to the shared slope constraint across the $k-1$ lines. Counting these sets gives the exact PRF complexity for the multiparty protocol.

Theorem 1.2 (Informal: PCF for SVMT)**.** *Let $k \geq 3$ be the number of parties, and let $X \subseteq \mathbb{F}$ be the support of each scalar share x_i. There exists a k-party PCF for the SVMT correlation relying solely on PRFs. In this PCF, each party holds $k|X|^{k-1} - (k-1)|X|^{k-2} + 2k - 2$ PRF keys. The PCF evaluation involves a single PRF evaluation for each key.*

We finally note that a k-party SVMT correlation over $\mathbb{F}_2$ implies the kind of gadget used in the multi-party DPF construction from [16].[2] The gadget enables compressing k-party additive shares of either a long random vector r or a long 0-vector, such that no strict subset of parties can distinguish between the two cases, nor learn anything about r in the former case. Indeed, an SVMT correlation over $\mathbb{F}_2$ achieves exactly this by having the scalar act as a selector between the two cases.

1.4 Related Work

Pseudorandom secret sharing. A closely related primitive to PCFs for linear correlations is *pseudorandom secret sharing* (PRSS), introduced by Cramer et al. [23]. PRSS enables a set of parties to locally generate shares of pseudorandom secrets drawn from a given secret-sharing scheme. A PCF for linear correlations is a means for realizing linear PRSS. The general PRSS construction from [23] is closely related to the PCF implicit in [26], but uses the semantics of secret sharing and allows parties to hold multiple field

[2]The multiparty DPF from [16] was asymptotically improved in [27]. The improvement was obtained by first constructing a variant of the gadget that only requires polynomially many keys by settling for an inverse-polynomial security error, and then amplifying security at the level of the DPF application. It is not clear how to apply a similar security amplification step in the context of PCF for SVMT.

elements as shares (as opposed to a single element in [26]). Its analysis, based on canonical span programs, implies a *share conversion* transformation that locally converts replicated (aka CNF) shares of a secret s according to an access structure $\mathcal{A}$ to shares of the *same* secret s according to any other linear scheme that realizes $\mathcal{A}$. This PRSS technique was further extended by Benhamouda et al. [7], who introduced generalized PRSS for variants of Shamir's secret-sharing scheme that provide better efficiency for threshold PRSS in case there are gaps between the security threshold and the polynomial degree. This was primarily motivated by the case of packed secret sharing, where such gaps are inherent. These techniques can potentially be useful for improving our multi-party PCFs when settling for security against bounded collusion size.

Pseudorandom correlations. The quest to compress useful nonlinear correlations, such as VOLE and oblivious transfer, has led to the recent line of work of Boyle et al. on *pseudorandom correlation generators* (PCGs) [12, 14] and *pseudorandom correlation functions* (PCFs) [15]. Most PCG and PCF constructions in the literature target concrete efficiency and are based on different flavors of the LPN assumption. See [36, 30, 39, 19, 22] and references therein. Our PCG and PCF constructions rely on the weaker (and minimal) assumption that one-way functions exist. However, we are much more limited in the type of correlations we can support. In particular, even in the simple case of VOLE, we can only efficiently support instances in which the scalar comes from a feasible-size domain. The possibility of a PCG for general VOLE from symmetric cryptography is still open, and so is the possibility of basing PCG for oblivious transfer and other useful correlations on symmetric cryptography.

2 Preliminaries

We index n parties and entries of length-n vectors by $i = 0, 1, \ldots, n-1$ and let $[n] = \{0, 1, \ldots, n-1\}$. Similarly, for any finite field $\mathbb{F}$ of size q, we identify the elements of $\mathbb{F}$ with $x = 0, 1, \ldots q-1$.

2.1 Multiparty Correlations

An m-party correlation is a joint distribution $\mathcal{C} = (R_0, \ldots, R_{m-1})$ over $D_0 \times \cdots \times D_{m-1}$, where each D_i is a finite domain, typically a vector space over a finite field $\mathbb{F}$. We will use k to denote the number of parties in a *target correlation* that we want to securely generate and n to denote the number of parties in a *source correlation* that we use towards this end. We will use $\mathcal{C}^N$ to denote an m-party correlation obtained via N independent samples of $\mathcal{C}$.

2.1.1 Linear Correlations

A typical class of source correlations we will use in this work are n-party correlations defined by a linear code, namely a linear subspace $C \subseteq \mathbb{F}^n$. We will start with the simple notion of linear correlations from [26] in which each party receives exactly one coordinate and later generalize it to allow a general assignment of coordinates to parties.

Definition 2.1 (Linear correlation)**.** *Let $\mathbb{F}$ be a finite field and $C \subseteq \mathbb{F}^n$ be a linear subspace. The n-party (simple) linear correlation over $\mathbb{F}$ specified by C is the correlation $\mathcal{C}$ that provides the parties with a joint sample drawn uniformly at random from C. Namely, it samples a vector $(c_0, \ldots, c_{n-1}) \leftarrow C$ and outputs the i-th coordinate c_i to party i, for $i \in \{0, \ldots, n-1\}$.*

The above notion of a linear correlation can be generalized to allow each entry to contain multiple field elements. A useful instance of this is a correlation obtained by replicating independently chosen field elements. These correlations are useful because many (pseudo-)independent instances of such a correlations can be compressed using pseudorandomness.

Definition 2.2 (Replicated randomness correlation)**.** *A* replicated randomness correlation *over $\mathbb{F}$ of dimension d is an n-party correlation $\mathcal{C}$ in which each entry contains a fixed subset of the entries of a uniformly random vector $\mathbf{r} \in \mathbb{F}^d$.*

2.1.2 VOLE-Style Correlations

The main target correlations we consider in this work are "VOLE-style" correlations that support secure scalar-vector products. In such a correlation, we have a random scalar $x \in \mathbb{F}$ and long random vector $\mathbf{a} \in \mathbb{F}^N$, both of which may either be given in the clear to different parties or additively secret-shared between all parties. (We will sometimes have more than one scalar multiplying the same $\mathbf{a}$.)

Definition 2.3 (Standard and subfield VOLE)**.** *In a standard 2-party Vector Oblivious Linear Evaluation (VOLE) correlation over $\mathbb{F}$, the sender P_0 holds random vectors $\mathbf{a}, \mathbf{b} \in \mathbb{F}^N$, and the receiver P_1 holds a random scalar $x \in \mathbb{F}$. The correlation provides the receiver with the output vector $\mathbf{a}x + \mathbf{b} \in \mathbb{F}^N$. The definition requires that $\mathbf{a}$ and $\mathbf{b}$ remain hidden from the receiver, and x remain hidden from the sender. We also consider a variant where x is sampled from a subset $X \subseteq \mathbb{F}$, where $|X| \geq 2$. We denote this correlation by $\mathsf{VOLE}_{\mathbb{F},N,X}$, omitting X when $X = \mathbb{F}$. Finally, a* subfield VOLE *correlation $\mathsf{VOLE}_{\mathbb{F},\mathbb{F}',N,X}$ is defined similarly except that $\mathbf{a} \in (\mathbb{F}')^N$ for a subfield $\mathbb{F}' \subseteq \mathbb{F}$.*

The following correlation is analogous to a standard (Beaver-style [5]) multiplication triple, except for supporting a scalar-vector multiplication instead of a scalar-scalar multiplication. It differs from the above in that $\mathbf{a}$ and x are additively shared.

Definition 2.4 (Scalar-Vector Multiplication Triple (SVMT))**.** *A k-party Scalar-Vector Multiplication Triple (SVMT) correlation consists of a k-party additive secret sharing of a random scalar $x \in \mathbb{F}$ and a random vector $\mathbf{a} \in \mathbb{F}^N$, along with a k-party additive secret sharing of their scalar-vector product $\mathbf{a}x \in \mathbb{F}^N$.*

More generally, we may restrict the scalar shares x_i to be sampled uniformly from $X \subseteq \mathbb{F}^k$ where $|X| \geq 2$ (in which case the shared scalar x is not necessarily uniform and may not be perfectly hidden). We denote this by $\mathsf{SVMT}_{k,\mathbb{F},N,X}$, omitting X when $X = \mathbb{F}^k$.

The following extension of SVMT enables multiplying the same vector $\mathbf{a}$ by another scalar y to achieve an authenticated variant of scalar-vector multiplication. Indeed, keeping the ratio between x and y secret enables the type of consistency checks applied in "SPDZ-style" MPC protocols [6, 24].

Definition 2.5 (Authenticated SVMT)**.** *A k-party* authenticated *scalar-vector multiplication triple (ASVMT) extends SVMT by introducing an additional random scalar $y \in \mathbb{F}$, possibly restricting its shares to $Y \subseteq \mathbb{F}^k$. The correlation provides the parties with k-party additive shares of the scalars $x, y \in \mathbb{F}$, the vector $\mathbf{a} \in \mathbb{F}^N$, and the two scaled vectors $\mathbf{a}x \in \mathbb{F}^N$ and $\mathbf{a}y \in \mathbb{F}^N$. We denote this by $\mathsf{ASVMT}_{k,\mathbb{F},N,X,Y}$.*

Supporting random access. To support fast random access to a correlation instance when N is big (e.g., when $[N]$ is the input domain of a PRF), it will be convenient to view a k-party VOLE-style correlation $\mathcal{C}$ as a k-tuple of *correlated random functions*. Concretely, we view $\mathcal{C}$ as a joint distribution over k functions $(F_0, \ldots, F_{k-1})$, where $F_i : [N] \to D_i$, such that for an instance identifier $\mathsf{id} \in [N]$, the output $F_i(\mathsf{id})$ includes entry id of each vector share, along with the scalar shares which are independent of id.

2.2 Secure Non-Interactive Reductions

A central tool we use in this work is a perfectly secure and non-interactive *reduction* from a source correlation $\mathcal{C}_S$ to a target correlation $\mathcal{C}_T$, which provides a distributed way of converting a sample from $\mathcal{C}_S$ into a sample from $\mathcal{C}_T$. Here we consider a reduction that only involves local computation and no interaction. Such a reduction is a simple special case of the general notion of secure reductions, as formalized in [21, 28], where here a reduction only involves local computation and no interaction. The power of such (statistically secure) reductions in the two-party setting was studied in a recent line of work, originating from [34, 1]. We give an explicit definition below, focusing on the perfect case.

Definition 2.6 (Secure Non-Interactive Reduction (SNIR) [26, 34, 1])**.** *Let $k \geq 2$ be an integer. Let $\mathcal{C}_S$ be a k-party source correlation defined over the domain $\mathcal{S}_0 \times \cdots \times \mathcal{S}_{k-1}$, and let $\mathcal{C}_T$ be a k-party target correlation defined over the domain $\mathcal{T}_0 \times \cdots \times \mathcal{T}_{k-1}$.*

A (perfectly) secure non-interactive reduction *(SNIR) from the target correlation $\mathcal{C}_T$ to the source correlation $\mathcal{C}_S$ is a tuple of deterministic local functions $\mathcal{F} = (f_0, \ldots, f_{k-1})$, where $f_i \colon \mathcal{S}_i \to \mathcal{T}_i$, satisfying the following condition:*

For any strict subset of corrupted parties $A \subsetneq \{0, \ldots, k-1\}$ (with complement $\bar{A} = \{0, \ldots, k-1\} \setminus A$), there exists a randomized simulator Sim such that the following real distribution $\mathsf{Real}_{\mathcal{F},A}$ and the ideal distribution $\mathsf{Ideal}_{\mathsf{Sim},A}$ are identical. The distributions are defined by the following experiments:

- ***The Real Execution*** *($\mathsf{Real}_{\mathcal{F},A}$):*
  1. *Sample the source correlation: $\mathbf{s} \leftarrow \mathcal{C}_S$.*
  2. *Define the adversary's view of the source correlation: $\mathbf{s}_A = (s_i)_{i \in A}$.*
  3. *For all honest parties $j \in \bar{A}$, locally compute the target shares: $t_j = f_j(s_j)$. Let $\mathbf{t}_{\bar{A}} = (t_j)_{j \in \bar{A}}$.*
  4. *Output the joint distribution: $(\mathbf{s}_A, \mathbf{t}_{\bar{A}})$.*
- ***The Ideal Execution*** *($\mathsf{Ideal}_{\mathsf{Sim},A}$):*
  1. *Sample the target correlation: $\mathbf{t} \leftarrow \mathcal{C}_T$.*

2. *Define the target shares for the corrupted and honest parties:* $\mathbf{t}_A = (t_i)_{i\in A}$ *and* $\mathbf{t}_{\bar{A}} = (t_j)_{j\in\bar{A}}$.
3. *Run the simulator on the corrupted target shares to generate the simulated source shares:* $\mathbf{s}_A \leftarrow \mathsf{Sim}(\mathbf{t}_A)$.
4. *Output the joint distribution:* $(\mathbf{s}_A, \mathbf{t}_{\bar{A}})$.

Remark 2.1 (Correctness)**.** *The above simulation-based security definition implies the natural correctness requirement, namely that the local functions f_i map the source correlation to the target correlation. Indeed, when $A = \emptyset$, the adversary's view $\mathbf{s}_\emptyset$ is empty. The requirement that $\mathsf{Real}_{\mathcal{F},\emptyset} \equiv \mathsf{Ideal}_{\mathsf{Sim},\emptyset}$ dictates that the locally computed shares $\mathbf{t}_{\bar{A}} = (f_0(s_0), \ldots, f_{k-1}(s_{k-1}))$ generated from a fresh $\mathcal{C}_S$ sample are distributed identically to a fresh sample $\mathbf{t}$ drawn directly from $\mathcal{C}_T$.*

Remark 2.2 (Canonical consistent simulator)**.** *Definition 2.6 does not explicitly require the simulator's output $\mathbf{s}_A$ to satisfy $f_A(\mathbf{s}_A) = \mathbf{t}_A$. Indeed, a simulator may output source views that are not locally consistent with its input target shares, as long as the resulting joint distribution $(\mathbf{s}_A, \mathbf{t}_{\bar{A}})$ is correct. However, for a perfect SNIR with deterministic local maps, one may use without loss of generality a* canonical consistent simulator *that samples random $\mathbf{s}_A$ conditioned on $f_A(\mathbf{s_A}) = \mathbf{t}_A$.*

Remark 2.3 (Efficient simulation for linear SNIR)**.** *The above SNIR definition is purely information-theoretic and does not require the simulator* Sim *to be computationally efficient. However, in all of the SNIR instances considered in this work, the simulator is computationally efficient. In particular, this always holds for a* linear *SNIR, where both $\mathcal{C}_S$ and $\mathcal{C}_T$ are linear (in the sense of Definition 2.1), and the local mappings f_i are linear functions (see Appendix A.1). This condition can be generalized using a suitable efficient reverse-sampling requirement that depends not only on the correlations but also on the f_i (see Remark A.1). The general condition will be met for all instances of SNIR considered in this work.*

Remark 2.4 (Corruptible SNIR)**.** *Finally, we note that any perfect SNIR is also perfectly secure in the* corruptible correlation *setting [14, 40]. A corruptible variant $\mathcal{C}'$ of a k-party correlation $\mathcal{C}$ enables the adversary to fix its own entries (subject to being in the support of the distribution) and samples the honest parties' entries conditioned on this choice. Access to $\mathcal{C}'$ can be viewed as a weaker resource than access to $\mathcal{C}$ since $\mathcal{C}'$ gives the adversary partial control over the outputs. However, a perfect SNIR from $\mathcal{C}_T$ to $\mathcal{C}_S$ is also a perfect* corruptible *SNIR, namely SNIR from $\mathcal{C}'_T$ to $\mathcal{C}'_S$. See Appendix A.2.*

2.3 Pseudorandom Correlations

In this section, we review the concepts of Pseudorandom Correlation Generators (PCG) and Pseudorandom Correlation Functions (PCF), which allow for securely compressing long cryptographic correlations via short, correlated seeds. We only outline the definitions and refer the reader to [14, 15] for the formal versions.

Definition 2.7 (Pseudorandom Correlation Generator (PCG) [12, 14], informal)**.** *A 2-party PCG for a target correlation $\mathcal{C}$ is a pair of algorithms* $(\mathsf{Gen}, \mathsf{Expand})$. *The randomized* Gen *algorithm outputs a pair of short, correlated seeds (s_0, s_1). The deterministic* Expand

algorithm takes as input a party index $\sigma \in \{0,1\}$ and a seed s_σ, and outputs a long pseudorandom string R_σ. The security requirement guarantees that the expanded outputs (R_0, R_1) are computationally indistinguishable from $\mathcal{C}$, even to an adversary that corrupts one party and learns their seed. This is formalized by requiring that Expand *be a* computationally secure *SNIR from the* corruptible *variant $\mathcal{C}'$ of $\mathcal{C}$ (see Remark 2.4) to the seed correlation generated by* Gen*. This naturally generalizes to a k-party PCG.*

The use of a corruptible variant of $\mathcal{C}$ is necessary, since without it the above PCG definition cannot be realized [14]. In the context of applications, it means that the application should be secure even when relying on $\mathcal{C}'$. Fortunately, this is the case for natural applications.

A PCF can be viewed as a PCG for an exponentially big correlation, but where both the honest parties and the adversary have random access to the correlation. Here we will consider VOLE-style correlations with vector length N, where a correlation instance is indexed by an identifier $\mathsf{id} \in [N]$ that serves as the PCF input and gives random access to entry id of each vector.

Definition 2.8 (Pseudorandom Correlation Function (PCF) [15], informal)**.** *A PCF generalizes a PCG by providing oracle access to correlation instances indexed by $\mathsf{id} \in [N]$. It consists of a seed generation algorithm that outputs correlated keys (s_0, s_1) and a keyed evaluation function $F_s(\mathsf{id})$. For a suitable joint key distribution, the evaluated outputs $(F_{s_0}(\mathsf{id}), F_{s_1}(\mathsf{id}))$ are computationally indistinguishable from instances of the (corruptible) target correlation even when allowing an adaptively chosen sequence of indices id. Crucially, this indistinguishability holds against insiders, remaining secure even when the adversary is given one of the two keys s_σ. This naturally generalizes to a k-party PCF.*

Remark 2.5 (PCGs and PCFs via SNIR)**.** *A corollary of Remark 2.3 and Remark 2.4 is that for reverse-samplable distributions $\mathcal{C}$ and $\mathcal{C}_T$, a PCF (resp., PCG) for $\mathcal{C}$ together with a perfect SNIR from $\mathcal{C}_T$ to $\mathcal{C}$ imply a PCF (resp., PCG) for $\mathcal{C}_T$.*

From here on, we will formulate all of our results for the stronger PCF primitive, based on a PRF, with the understanding that a similar PCG analogue can be based on a PRG. The PCG variant is often sufficient for applications and can yield better concrete efficiency in some cases.

2.4 PCF for Linear Correlations

We now recall the replication-based approach from [26] for compressing linear correlations, formulated as a PCF for linear correlations based on any PRF.

Definition 2.9 (Minimal-support codeword)**.** *Let $\mathbb{F}$ be a finite field and $C \subseteq \mathbb{F}^n$ be a linear subspace. The support of a vector $v \in \mathbb{F}^n$, denoted $supp(v)$, is the set of indices $i \in \{0, \ldots, n-1\}$ for which $v_i \neq 0$. A non-zero codeword $w \in C$ is a minimal-support codeword if there is no non-zero $w' \in C$ such that $supp(w') \subsetneq supp(w)$. We denote by $W(C)$ the set of all minimal-support codewords in C, normalized by selecting exactly one representative for each support set (e.g., the one whose first nonzero entry is 1).*

The main technical result from [26] is a perfectly secure reduction from a linear correlation $\mathcal{C}$ to a more redundant linear correlation $\widehat{\mathcal{C}}$ which consists of replicated, independent field elements, with replication pattern determined by $W(C)$.

Definition 2.10 (Replicated minimal-support correlation $\widehat{\mathcal{C}}$)**.** *Let $\mathbb{F}$ be a finite field and $C \subseteq \mathbb{F}^n$ be a linear subspace, defining a linear correlation $\mathcal{C}$. Let $W(C)$ be the set of (normalized) minimal-support codewords of C. The* minimal-support correlation $\widehat{\mathcal{C}}$ *for C is a replicated n-party correlation containing $|W(C)|$ independent field elements r_w which are replicated as follows. For each minimal-support codeword $w \in W(C)$ and party $i \in \{0, \ldots, n-1\}$, party i is given r_w if and only if $i \in supp(w)$.*

The main technical result from [26] is captured by the following theorem.

Theorem 2.1 (Linear correlations from replicated randomness [26])**.** *For any linear correlation $\mathcal{C}$, there is a perfect SNIR from $\mathcal{C}$ to $\widehat{\mathcal{C}}$.*

The above reduction is motivated by the possibility of generating many (pseudo-) independent copies of each r_w using a dedicated PRF key s_w. This can be formalized as a simple direct construction of a PCF for N independent instances of $\widehat{\mathcal{C}}$ using any PRF, by combining Theorem 2.1 and Remark 2.5. This implies:

Corollary 2.1 (Compressing linear correlations [26, 7])**.** *Let $C \subseteq \mathbb{F}^n$ be a linear subspace defining a linear correlation $\mathcal{C}$. Let $W(C)$ be the set of (normalized) minimal-support codewords of C. There exists a PCF for $\mathcal{C}^N$ that requires $|W(C)|$ independent keys s_w for a PRF with input domain $[N]$ and output domain $\mathbb{F}$. Concretely, each s_w corresponds to $w \in W(C)$ and is received by each party $i \in supp(w)$. The PCF evaluation function (per instance) involves a single PRF evaluation per party per key.*

3 Secret Projection Reductions

The main idea that we introduce to obtain new PCF constructions is to use secretly chosen random *projections*, assigning to each party of a target correlation $\mathcal{C}_T$ one or more parties in a source correlation $\mathcal{C}_S$, and providing each party in $\mathcal{C}_T$ with all the data available to the parties in $\mathcal{C}_S$ it emulates. When applying this to the previous PCFs for linear correlation, we get PCFs whose keys consist of secretly replicated PRF keys.

We start by formalizing the notion of projection reductions.

Definition 3.1 (Projection reduction)**.** *Let Q be a distribution over $(2^{[n]})^k$, namely k-tuples of subsets of $[n]$. For an n-party correlation $\mathcal{C}_S$, the Q-induced correlation $Q(\mathcal{C}_S)$ is a k-party correlation defined via the following process:*

- *Sample $(Q_0, Q_1, \ldots, Q_{k-1}) \leftarrow Q$;*
- *Sample $\mathbf{s} = (s_0, s_1, \ldots, s_{n-1}) \leftarrow \mathcal{C}_S$;*
- *Output $\mathbf{t} = (t_0, t_1, \ldots, t_{k-1})$ where $t_i = (Q_i, \mathbf{s}|_{Q_i})$.*

A projection reduction *from a k-party $\mathcal{C}_T$ to an n-party $\mathcal{C}_S$ is a pair (Q, R), where R is a SNIR from $\mathcal{C}_T$ to $Q(\mathcal{C}_S)$.*

One could generalize the above notion to allow the reduction to provide additional partial information to each party beyond the set Q_i. However, the above notion suffices for our purposes.

The key observation is that a projection reduction can be used to directly convert a PCF for an n-party correlation $\mathcal{C}_S$ to a similar PCF for a k-party $\mathcal{C}_T$. Indeed, the PCF for $\mathcal{C}_T$ is obtained by simply replicating the n keys of a PCF for $\mathcal{C}_S$ according to the secret pattern induced by Q. (We assume that all distributions satisfy the reverse-sampling conditions required for efficient simulation.) This is captured by the following theorem, whose formal statement and proof are deferred to Appendix A.3.

Theorem 3.1 (PCF via projection, informal)**.** *Let $\mathcal{C}_S$ be an n-party correlation and $\mathcal{C}_T$ be a k-party correlation. Let Q be a distribution over $(2^{[n]})^k$. If there is a projection reduction (Q, R) from $\mathcal{C}_T$ to $\mathcal{C}_S$ and a PCF for $\mathcal{C}_S$, then the natural composition, where the key generator samples Q and distributes the PCF keys according to Q_i, is a PCF for $\mathcal{C}_T$.*

In the case of linear correlations, combining the PCF from Corollary 2.1 with a projection reduction leads to PCF whose keys include a *secret replication* of independent PRF keys.

4 From Linear to VOLE-Style Correlations

In this section we use linear n-party source correlations $\mathcal{C}_S$ to securely derive useful VOLE-style k-party target correlations $\mathcal{C}_T$ via projection reductions.

4.1 Small-Domain VOLE from Line Correlation

We start by reproducing the PCF for standard VOLE of Roy [42]. Here we take as the target correlation $\mathcal{C}_T = \mathsf{VOLE}_{\mathbb{F},N}$, namely a standard 2-party VOLE over $\mathbb{F}$ of length N. To obtain a PCF for VOLE, we let the source correlation $\mathcal{C}_S$ be an n-party correlation, for $n = |\mathbb{F}|$, corresponding to a length-n degree-1 Reed-Solomon code. Concretely, we use the code C_{Line}, which consists of evaluations of lines $L(\alpha) = a\alpha + b$ on the entire field. This can be generalized to a subset X of evaluation points, resulting in a code of length $n = |X|$, and to the case where $a \in \mathbb{F}'$ for a subfield $\mathbb{F}' \subseteq \mathbb{F}$. In the latter case, which serves as the basis for the SoftSpokenOT protocol [42], we get a linear correlation over $\mathbb{F}'$.

Theorem 4.1 (VOLE from line correlation)**.** *There is a projection reduction from $\mathsf{VOLE}_{\mathbb{F},N}$ to $\mathcal{C}^N_{\mathsf{Line},\mathbb{F}}$, where $\mathcal{C}_{\mathsf{Line},\mathbb{F}}$ is the $\mathbb{F}$-linear correlation defined by*

$$C_{\mathsf{Line},\mathbb{F}} = \{(L(\alpha))_{\alpha\in\mathbb{F}} \,:\, L(\alpha) = a\alpha + b,\, a, b \in \mathbb{F}\}.$$

More generally, there is a projection reduction from $\mathsf{VOLE}_{\mathbb{F},N,X}$ to $\mathcal{C}^N_{\mathsf{Line},\mathbb{F},\mathsf{X}}$, for the linear correlation $C_{\mathsf{Line},\mathbb{F},X} = \{(L(\alpha))_{\alpha\in X} \,:\, L(\alpha) = a\alpha + b,\, a, b \in \mathbb{F}\}$.

Proof. We prove the generalized statement, which directly implies the base case by setting $X = \mathbb{F}$. We must construct a projection reduction (Q, R) from the target correlation $\mathcal{C}_T = \mathsf{VOLE}_{\mathbb{F},N,X}$ to the source correlation $\mathcal{C}_S = \mathcal{C}^N_{\mathsf{Line},\mathbb{F},X}$. The target $\mathcal{C}_T$ has $k = 2$ parties, where party 0 is the sender and party 1 is the receiver.

We define the projection distribution Q over $(2^X)^2$ as follows:

- Q_0 is deterministically the entire set X.
- Q_1 is a singleton set $\{x\}$, where x is sampled uniformly at random from X.

Next, we define the non-interactive reduction $R = (f_0, f_1)$. In the source correlation, each coordinate $\alpha \in X$ contains the evaluation of N independent lines: $\mathbf{s}_\alpha = \mathbf{a}\alpha + \mathbf{b} \in \mathbb{F}^N$, where $\mathbf{a}, \mathbf{b} \in \mathbb{F}^N$.

- **Sender (f_0):** The sender receives $Q_0 = X$ and the evaluations $(\mathbf{s}_\alpha)_{\alpha \in X}$. Assuming $|X| \geq 2$, the sender efficiently interpolates these lines to recover the unique slopes $\mathbf{a} \in \mathbb{F}^N$ and intercepts $\mathbf{b} \in \mathbb{F}^N$. It outputs the target shares $t_0 = (\mathbf{a}, \mathbf{b})$.
- **Receiver (f_1):** The receiver receives $Q_1 = \{x\}$ and the single evaluation $\mathbf{s}_x \in \mathbb{F}^N$. It directly outputs the target shares $t_1 = (x, \mathbf{s}_x)$.

To prove that this (Q, R) forms a perfect SNIR, we construct a simulator Sim for any corrupted subset of parties $A \subsetneq \{0, 1\}$.

Case 1: Corrupted Sender ($A = \{0\}$). In the ideal execution, the simulator receives the sender's target shares $t_0 = (\mathbf{a}, \mathbf{b}) \leftarrow \mathcal{C}_T$. It must simulate the sender's view of the projected source correlation $(Q_0, \mathbf{s}_{Q_0})$. The simulator sets $Q_0 = X$ and computes $\mathbf{s}_\alpha = \mathbf{a}\alpha + \mathbf{b}$ for all $\alpha \in X$. It outputs $(X, (\mathbf{s}_\alpha)_{\alpha \in X})$. This perfectly matches the real execution, where $\mathbf{a}$ and $\mathbf{b}$ are sampled uniformly, and the honest receiver's output t_1 is exactly the evaluation at a uniformly random $x \in X$. Thus, $\mathsf{Real}_{\mathcal{F},\{0\}} \equiv \mathsf{Ideal}_{\mathsf{Sim},\{0\}}$.

Case 2: Corrupted Receiver ($A = \{1\}$). In the ideal execution, the simulator receives the receiver's target shares $t_1 = (x, \mathbf{y}) \leftarrow \mathcal{C}_T$, where $\mathbf{y} = \mathbf{a}x + \mathbf{b}$. It must simulate the receiver's view of the projected source correlation $(Q_1, \mathbf{s}_{Q_1})$. The simulator sets $Q_1 = \{x\}$ and $\mathbf{s}_x = \mathbf{y}$. It outputs $(\{x\}, \mathbf{y})$. In the real execution, the receiver's assigned coordinate is $\{x\}$ for a uniformly random $x \leftarrow X$, and its source share is exactly the evaluation at x, which is $\mathbf{a}x + \mathbf{b}$. Conditioned on $\mathbf{a}x + \mathbf{b} = \mathbf{y}$, the honest sender's output $(\mathbf{a}, \mathbf{b})$ in the real execution is perfectly uniform, matching the ideal execution. Thus, $\mathsf{Real}_{\mathcal{F},\{1\}} \equiv \mathsf{Ideal}_{\mathsf{Sim},\{1\}}$.

Since perfect simulation holds for all strict subsets, (Q, R) is a secure projection reduction. □

Combining the above reduction with Theorem 3.1 and the PCF for line correlation implied by Corollary 2.1, we get a PCF for VOLE.

Corollary 4.1 (PCF for VOLE, implicit in [42])**.** *Let $\mathbb{F}$ be a finite field and N be the domain size of a PRF. There exists a PCF for $\mathsf{VOLE}_{\mathbb{F},N}$, where the sender holds $|\mathbb{F}|$ PRF keys and the receiver holds all keys except the one indexed by x. More generally, there is a PCF for $\mathsf{VOLE}_{\mathbb{F},N,X}$, where the sender holds $|X|$ PRF keys and the receiver holds all keys except one. Evaluating a PCF instance requires each party to perform a single PRF evaluation per key.*

We conclude this section with a few remarks on extensions and optimizations.

Remark 4.1 (Subfield VOLE)**.** *The above PCF can be naturally extended to support subfield-VOLE $\mathsf{VOLE}_{\mathbb{F},\mathbb{F}',N,X}$. This uses a projection reduction from $\mathsf{VOLE}_{\mathbb{F},\mathbb{F}',N,X}$ to $\mathcal{C}^N_{\mathsf{Line},\mathbb{F},\mathbb{F}',X}$, for the $\mathbb{F}'$-linear correlation $C_{\mathsf{Line},\mathbb{F},\mathbb{F}',X} = \{(L(\alpha))_{\alpha \in X} : L(\alpha) = a\alpha + b, a \in \mathbb{F}', b \in \mathbb{F}\}$. Here each party owns multiple entries of the correlation over $\mathbb{F}'$. To match the cost of the above VOLE solution, we use a generalization of the GI construction that can assign multiple codeword coordinates to each party. See Appendix B.*

Remark 4.2 (Smooth VOLE)**.** *The above construction is not efficient for super-polynomial size $\mathbb{F}$, unless the set of evaluation points X is of polynomial size. However, by concatenating independent copies of $\mathsf{VOLE}_{\mathbb{F}_{p_i},N}$ for small primes p_i and using the Chinese Remainder Theorem, one can get a PCF for VOLE over an exponentially large smooth integer modulus $M = \prod p_i$. This type of PCF serves as the basis for Kondi's threshold ECDSA protocol [37].*

Remark 4.3 (Key compression and distributed setup)**.** *The PCF for VOLE obtained directly via Corollary 4.1 requires the VOLE sender to hold $|\mathbb{F}|$ PRF keys and the receiver to hold $|\mathbb{F}|-1$ keys. The keys can be compressed via the use of a puncturable PRF [11, 35, 17], where the sender only holds the root of a GGM tree [29] and the receiver holds a punctured key including a logarithmic number of nodes along the path to a leaf. Moreover, to distribute the key generation, one can use concretely efficient protocols that make a black-box use of a PRG [13, 43, 42].*

4.2 Scalar-Vector MT from Multi-Line Correlation

When generalizing from 2-party VOLE to a k-party Scalar-Vector Multiplication Triple (SVMT), a natural approach to generate the cross-terms $x_i\mathbf{a}_j$ would be for a sender party j to use a single source linear correlation representing one line, $L(\alpha) = \mathbf{a}_j\alpha + \mathbf{b}_j$, and have all other $k-1$ parties evaluate this exact same line at their respective scalar shares x_i. However, this approach critically fails to provide security against colluding parties. Because a degree-1 polynomial is uniquely determined by just two points, any coalition of two or more receiving parties could pool their evaluations, interpolate the line, and easily extract the shared slope $\mathbf{a}_j$, thereby entirely compromising party j's secret vector share.

To securely generate these cross-terms without leaking $\mathbf{a}_j$ to colluding subsets, we execute k symmetric instances of a random projection utilizing multiple lines with independent intercepts. For each party $j \in \{1, \ldots, k\}$ acting as the "sender", we define a source linear correlation C_S of length $M = (k-1)|\mathbb{F}|$. The code C_S consists of the evaluations of $k-1$ independent lines, denoted $L_{j\to i}(\alpha) = \mathbf{a}_j\alpha + \mathbf{b}_{j\to i}$ for each $i \neq j$, over all points $\alpha \in \mathbb{F}$. Crucially, all $k-1$ lines share the identical slope $\mathbf{a}_j$ (representing party j's vector share) but feature independent, uniformly random intercepts $\mathbf{b}_{j\to i}$.

In the random projection from C_S to C_T, party j is assigned all entries of C_S, thereby holding the keys that generate the intercepts $\mathbf{b}_{j\to i}$. Each receiving party $i \neq j$ is assigned the single entry corresponding to the evaluation of their specific line $L_{j\to i}$ at the secret point $\alpha = x_i$. Consequently, party i non-interactively evaluates the PRFs to obtain $\mathbf{c}_{i,j} = \mathbf{a}_j x_i + \mathbf{b}_{j\to i} \in \mathbb{F}^N$.

The values $\mathbf{c}_{i,j}$ held by party i and $-\mathbf{b}_{j\to i}$ held by party j directly constitute a 2-party additive sharing of the cross-term $x_i\mathbf{a}_j$. Summing these local shares across all pairs yields the final MT. This approach guarantees full k-party privacy: a coalition of $k-1$ corrupted parties aiming to learn the honest party j's vector $\mathbf{a}_j$ will collectively observe exactly one evaluated point on each of the $k-1$ lines. Due to the independent random intercepts, these evaluations are uniformly distributed and perfectly hide the shared slope.

Theorem 4.2 (SVMT from multi-line correlations)**.** *Let $S \subseteq \mathbb{F}$ be the union of the marginal supports of $X \subseteq \mathbb{F}^k$ (i.e., the set of all possible scalar shares x_i). There is a projection reduction from the scalar-vector multiplication triple correlation $\mathsf{SVMT}_{k,\mathbb{F},N,X}$ to the source*

correlation $\mathcal{C}^N_{\mathsf{SVMT-Line},\mathbb{F},\mathsf{S}}$*. The source correlation* $\mathcal{C}_{\mathsf{SVMT-Line},\mathbb{F},\mathsf{S}}$ *is defined over* k *independent blocks, where the* j*-th block is an* $\mathbb{F}$*-linear correlation consisting of* $k-1$ *lines over* S *with a common slope:*

$$C^{(j)}_{\mathsf{Line},\mathbb{F},\mathsf{S}} = \{(L_{j\to i}(\alpha))_{\alpha\in S, i\neq j} : L_{j\to i}(\alpha) = a_j\alpha + b_{j\to i},\, a_j, b_{j\to i} \in \mathbb{F}\}$$

Proof. Let $\mathcal{C}_T = \mathsf{SVMT}_{k,\mathbb{F},N,X}$ be the target correlation. This correlation can be equivalently defined by providing each party $\ell \in \{0,\ldots,k-1\}$ with a scalar share x_ℓ (where the joint vector $(x_0,\ldots,x_{k-1})$ is drawn from X), picking a uniformly random vector share $\mathbf{a}_\ell \in \mathbb{F}^N$, and additively sharing the cross-terms $\mathbf{a}_j x_i$ for all $i \neq j$. Specifically, party ℓ acts as a sender of its vector share to all $i \neq \ell$ by holding uniformly random intercepts $\mathbf{b}_{\ell\to i} \in \mathbb{F}^N$, and acts as a receiver from all $j \neq \ell$ by holding the evaluated cross-terms $\mathbf{c}_{\ell,j} = \mathbf{a}_j x_\ell + \mathbf{b}_{j\to\ell} \in \mathbb{F}^N$. Party ℓ's final share of the SVMT product is computed locally as $\mathbf{a}_\ell x_\ell + \sum_{j\neq\ell}\mathbf{c}_{\ell,j} - \sum_{i\neq\ell}\mathbf{b}_{\ell\to i}$.

We construct a projection reduction (Q,R) to the source correlation $\mathcal{C}_S = \mathcal{C}^N_{\mathsf{SVMT-Line},\mathbb{F},\mathsf{S}}$. The domain of $\mathcal{C}_S$ consists of k independent blocks. Each block $j \in \{0,\ldots,k-1\}$ comprises $k-1$ disjoint sub-domains $S_{j\to i}$ (each a copy of S) for all $i \neq j$.

We define the projection distribution Q over the subsets of this domain. For each party ℓ:

- **In its own block** ℓ**:** $Q^{(\ell)}_\ell$ is the entire domain of block ℓ, namely $\bigcup_{i\neq\ell} S_{\ell\to i}$.
- **In all other blocks** $j \neq \ell$**:** $Q^{(j)}_\ell$ consists of a single point in the sub-domain $S_{j\to\ell}$, specifically $Q^{(j)}_\ell = \{x_\ell\}$. The scalar share x_ℓ is consistent across all $j \neq \ell$.

We define the non-interactive reduction $R = (f_0,\ldots,f_{k-1})$. Given the projected source evaluations, each party ℓ locally computes its target shares $t_\ell \in \mathcal{C}_T$ as follows:

- By interpolating the $k-1$ lines in its full block ℓ, party ℓ recovers the common slope $\mathbf{a}_\ell \in \mathbb{F}^N$ and the $k-1$ independent intercepts $\mathbf{b}_{\ell\to i} \in \mathbb{F}^N$.
- From its singleton evaluations in the blocks $j \neq \ell$, party ℓ directly reads the vectors $\mathbf{c}_{\ell,j} \in \mathbb{F}^N$.
- Party ℓ computes
$$\mathbf{z}_\ell = \mathbf{a}_\ell x_\ell + \sum_{j\neq\ell}\mathbf{c}_{\ell,j} - \sum_{i\neq\ell}\mathbf{b}_{\ell\to i} \in \mathbb{F}^N$$
and outputs
$$t_\ell = (x_\ell, \mathbf{a}_\ell, \mathbf{z}_\ell).$$

To argue correctness, note that:

$$\sum_{\ell=0}^{k-1}\mathbf{z}_\ell = \sum_\ell \mathbf{a}_\ell x_\ell + \sum_\ell\sum_{j\neq\ell}(\mathbf{a}_j x_\ell + \mathbf{b}_{j\to\ell}) - \sum_\ell\sum_{i\neq\ell}\mathbf{b}_{\ell\to i} = \left(\sum_j \mathbf{a}_j\right)\left(\sum_\ell x_\ell\right) = \mathbf{a}x.$$

Moreover, conditioned on the shares $(x_\ell,\mathbf{a}_\ell)_\ell$, the vector $(\mathbf{z}_\ell)_\ell$ is uniform subject to $\sum_\ell \mathbf{z}_\ell = \mathbf{a}x$, owing to the refreshing induced by the independent directed-edge masks $\mathbf{b}_{j\to i}$.

To prove that (Q, R) is a perfect SNIR, we define a simulator Sim for any strict subset of corrupted parties $A \subsetneq \{0, \ldots, k-1\}$.

Fix a strict corrupted coalition $A \subsetneq [k]$. The simulator receives

$$(x_i, \mathbf{a}_i, \mathbf{z}_i)_{i \in A}.$$

It constructs the projected source view as follows. For every corrupted sender block $j \in A$, it samples all intercepts $\mathbf{b}_{j \to i} \leftarrow \mathbb{F}^N$ for $i \neq j$ uniformly, and outputs the full line evaluations

$$L_{j \to i}(\alpha) = \mathbf{a}_j \alpha + \mathbf{b}_{j \to i} \qquad \alpha \in S.$$

For corrupted receivers $i \in A$ and corrupted sender blocks $j \in A, j \neq i$, this determines

$$\mathbf{c}_{i,j} = \mathbf{a}_j x_i + \mathbf{b}_{j \to i}.$$

It remains to simulate the values $\mathbf{c}_{i,j}$ seen by corrupted receivers $i \in A$ in honest sender blocks $j \notin A$. For each fixed $i \in A$, define

$$\mathbf{K}_i = \mathbf{a}_i x_i + \sum_{j \in A,\, j \neq i} \mathbf{c}_{i,j} - \sum_{h \neq i} \mathbf{b}_{i \to h}.$$

Since A is strict, choose one honest index $h^\star \notin A$. For every $j \notin A, j \neq h^\star$, sample $\mathbf{c}_{i,j} \leftarrow \mathbb{F}^N$ uniformly, and set

$$\mathbf{c}_{i,h^\star} = \mathbf{z}_i - \mathbf{K}_i - \sum_{j \notin A,\, j \neq h^\star} \mathbf{c}_{i,j}.$$

The simulator places $\mathbf{c}_{i,j}$ at the projected coordinate x_i in the line $j \to i$.

In the real execution, for honest sender blocks $j \notin A$, the values $\mathbf{c}_{i,j} = \mathbf{a}_j x_i + \mathbf{b}_{j \to i}$ seen by corrupted receivers are independent uniform vectors because the intercepts $\mathbf{b}_{j \to i}$ are independent uniform pads. The only constraints involving these values and the corrupted target shares are the linear equations defining $\mathbf{z}_i$ above. The simulator samples exactly the uniform affine distribution subject to these equations.

Finally, we argue that the joint distribution of the simulated view with the honest target shares is distributed as it should. Conditioned on the corrupted target shares and on the simulated corrupted source view, the masks not appearing in the corrupted source view induce a uniform additive sharing of the remaining honest product share subject only to the global constraint

$$\sum_{\ell \in \bar{A}} \mathbf{z}_\ell = \left(\sum_j \mathbf{a}_j \right) \left(\sum_i x_i \right) - \sum_{\ell \in A} \mathbf{z}_\ell.$$

This follows from the fact that the pairwise masks unknown to A form a complete graph over the honest parties. Hence the honest target shares have exactly their ideal conditional distribution. This establishes a perfect SNIR simulation. □

Corollary 4.2 (PCF for SVMT)**.** *Let $\mathbb{F}$ be a finite field, N be the domain size of a PRF, and $X \subseteq \mathbb{F}^k$ be the support of the scalar shares. Let $S \subseteq \mathbb{F}$ be the union of the marginal supports of X. There exists a PCF for the k-party scalar-vector multiplication triple correlation $\mathsf{SVMT}_{k,\mathbb{F},N,X}$, where each party holds $k|S|^{k-1} - (k-1)|S|^{k-2} + 2k - 2$ PRF keys. Evaluating a PCF instance requires each party to perform a single PRF evaluation per key.*

Proof. The PCF is obtained by applying the Lin-PCF from [26] to the source linear correlation C_S, which consists of k independent blocks. By the properties of this construction, a PCF for a linear code requires exactly $|W(C)|$ PRF keys, where $W(C)$ is the set of minimal-support codewords up to scalar multiplication. A party holds the key corresponding to $w \in W(C)$ if and only if its assigned evaluation coordinate lies in the support of w.

Each block C_{block} is an $\mathbb{F}$-linear code of length $(k-1)|S|$ corresponding to the evaluation of $k-1$ lines over S sharing a common slope. The dimension of C_{block} is k. We can classify its minimal-support codewords $W(C_{block})$ into two mutually exclusive categories:

1. **Zero slope:** Codewords supported entirely on a single line's domain. There are exactly $k-1$ such codewords (one for each line).

2. **Non-zero slope:** Codewords that evaluate to zero at exactly one point per line. Because there are $|S|$ choices for the zero on each of the $k-1$ lines, there are exactly $|S|^{k-1}$ such codewords.

Thus, the total number of minimal-support codewords for a single block is $|W(C_{block})| = |S|^{k-1} + k - 1$.

In the projection reduction, each of the k parties acts as a sender for exactly one block and as a receiver for the remaining $k-1$ blocks.

- **As a sender:** The party is assigned the entire domain of its designated block. Since its coordinates cover the entire block, it lies in the support of all minimal-support codewords. Therefore, it holds all $|S|^{k-1} + k - 1$ keys for this block.

- **As a receiver:** In each of the other $k-1$ blocks, the party is assigned a single coordinate $x_i \in S$ on its designated line. It holds the key for a zero-slope codeword only if the word is supported on its line (1 key). It holds the key for a non-zero-slope codeword only if the word is non-zero at x_i. Since exactly $|S|^{k-2}$ of the non-zero-slope words evaluate to zero at x_i, the point lies in the support of the remaining $|S|^{k-1} - |S|^{k-2}$ words. Thus, the party holds $1 + |S|^{k-1} - |S|^{k-2}$ keys per receiver block.

Summing the keys across the single sender block and the $k-1$ receiver blocks yields the total PRF key count per party:

$$(|S|^{k-1} + k - 1) + (k-1)(|S|^{k-1} - |S|^{k-2} + 1) = k|S|^{k-1} - (k-1)|S|^{k-2} + 2k - 2$$

completing the proof. □

4.3 Authenticated Scalar-Vector MT

Building on the k-party scalar-vector MT, we now extend the construction to an authenticated scalar-vector MT. In this setting, the parties hold a k-party additive sharing of a random vector $\mathbf{v} = \sum_{i=1}^{k} \mathbf{a}_i$ alongside two scalars $x = \sum_{i=1}^{k} x_i$ and $y = \sum_{i=1}^{k} y_i$. The goal is to non-interactively generate additive sharings of both products, $x\mathbf{v}$ and $y\mathbf{v}$. The scalar y may serve as an information-theoretic MAC key for authenticating the shared vector and

subsequent operations on it. (Given our constraints on the scalar domains, this can only provide weak authentication; see Section 5 for discussion of how ASVMT may be used in applications.)

To achieve the above without compromising the secrecy of the sender's vector share $\mathbf{a}_j$, we naturally generalize the SVMT construction by expanding the source linear correlation C_S. For each party $j \in \{1, \ldots, k\}$, the correlation C_S is now defined over a length of $M = 2(k-1)|\mathbb{F}|$. It consists of the evaluations of $2(k-1)$ independent lines over $\mathbb{F}$, all of which share the exact same slope $\mathbf{a}_j$. Specifically, for each receiving party $i \neq j$, we define two lines: $L_{x,j\to i}(\alpha) = \mathbf{a}_j\alpha + \mathbf{b}_{x,j\to i}$ and $L_{y,j\to i}(\alpha) = \mathbf{a}_j\alpha + \mathbf{b}_{y,j\to i}$. The intercepts $\mathbf{b}_{x,j\to i}$ and $\mathbf{b}_{y,j\to i}$ are chosen independently and uniformly at random.

Corollary 4.3 (PCF for ASVMT)**.** *Let $\mathbb{F}$ be a finite field, N be the domain size of a PRF and $X, Y \subseteq \mathbb{F}^k$ be the supports of the scalar shares. Let $S_X, S_Y \subseteq \mathbb{F}$ be the unions of the marginal supports of X, Y, respectively. There exists a PCF for the k-party authenticated scalar-vector multiplication triple correlation $\mathsf{ASVMT}_{k,\mathbb{F},N,X,Y}$, where each party holds $k(|S_X||S_Y|)^{k-1} - (k-1)(|S_X||S_Y|)^{k-2} + 4(k-1)$ PRF keys. Evaluating a PCF instance requires each party to perform a single PRF evaluation per key.*

The construction, security proof and key count closely follow the SVMT case, applied simultaneously to the two independent families of cross-terms for $\mathbf{a}x$ and $\mathbf{a}y$. The shared slope $\mathbf{a}_j$ is hidden by the independent intercepts $\mathbf{b}_{x,j\to i}, \mathbf{b}_{y,j\to i}$.

5 Applications

Our new PCF constructions have broad applications for different flavors of secure multiparty computation (MPC), particularly in settings where multiplication triples (a.k.a. Beaver triples) or Vector Oblivious Linear Evaluations (VOLE) are employed. Our results are especially relevant for functionalities that involve repeated use of the same secret value. This includes matrix-vector multiplications (where the same scalar multiplies the coordinates of an entire vector), evaluating parallel SIMD-style arithmetic circuits, circuits that compute repeatedly with a fixed value (e.g., multiple encryptions/decryptions under a fixed key), and similar scenarios.

A caveat that should be kept in mind is that the computational cost of our PCFs typically scales with $|X|^{k-1}$, where $|X|$ is the scalar domain size and k is the number of parties. (In contrast, the PCF key size can be compressed using puncturable PRFs.) This requires the number of parties to be small and the repeated scalar values to be represented by elements from a small subset of the (potentially very large) field. Still, even with this limitation, there are quite a few applications that can benefit from the new PCF constructions. We discuss some concrete use cases below.

5.1 Secret-Sharing Based MPC

A central hub for applications is a secure scalar-vector multiplication primitive, where a vector $\mathbf{v} \in \mathbb{F}^N$ and scalar $w \in \mathbb{F}$ are either owned by different parties or additively shared between k parties. The output $w\mathbf{v}$ can either be revealed in the clear or secret-shared between the parties to be further processed by the protocol. Our PCG for SVMT can be

used to support all such scenarios by having the parties generate a big instance of $\mathsf{SVMT}_{\mathbb{F},N}$ from short PCG seeds. Concretely, by revealing the masked vector $\mathbf{v}' = \mathbf{v} + \mathbf{a}$ and masked scalar $w' = w + x$ (for $\mathbf{a}, x$ from SVMT), the parties can locally compute additive shares of $w\mathbf{v}$.

Such a secret-shared scalar-vector multiplication can serve as a general-purpose tool for circuit-based MPC, since it applies whenever the same value feeds into many multiplication gates. For instance, this is the case in the common scenario of circuits that involve matrix-vector multiplications. Our PCF for SVMT is particularly appealing when both k and $\mathbb{F}$ are small, e.g., in the case of 3-party evaluation of Boolean circuits. Moreover, the authenticated variant can be useful to provide lightweight (SPDZ-style) malicious security, though this use case is limited by the need to use MAC keys taken from a small domain.

5.2 Encrypted Matrix-Vector Product

A more concrete use case comes from a recent work on Encrypted Matrix-Vector Product (EMVP) [8]. The EMVP primitive has an offline phase and an online phase. In the offline phase, a client uses a secret key sk to encrypt a big matrix $\mathbf{M} \in \mathbb{F}^{N \times \ell}$ and store the encrypted matrix $\widehat{\mathbf{M}}$ on a server. In each invocation of the online phase, the client uses sk to encrypt a query vector $\mathbf{v} \in \mathbb{F}^{\ell}$ and send the encrypted vector $\widehat{\mathbf{v}}$ to the server. During the encryption of $\mathbf{v}$, the client also generates a short decryption key $\mathbf{d} \in \mathbb{F}^{s}$, where $s \ll \ell$. The server uses $\widehat{\mathbf{M}}$ and $\widehat{\mathbf{v}}$ to compress $\widehat{\mathbf{M}}$ into a much smaller encrypted matrix $\widehat{\mathbf{M}}' \in \mathbb{F}^{N \times s}$, which is sent back to the client. The client can recover $\mathbf{Mv}$ by computing the matrix-vector product $\widehat{\mathbf{M}}' \cdot \mathbf{d}$. Applications of EMVP include privacy-preserving linear algebra, similarity search, and machine-learning inference.

To ensure that the server's answer does not reveal anything beyond the output $\mathbf{Mv}$, or alternatively produce additive shares of $\mathbf{Mv}$ for further processing, the EMVP protocol from [8] suggests using linearly homomorphic encryption (LHE) to have the server multiply the compressed encrypted matrix $\widehat{\mathbf{M}}'$ by an encrypted version of the decryption key $\mathbf{d}$. An alternative approach proposed in [8] is to use a PCG for VOLE. This typically involves higher communication cost than the LHE-based approach (comparable to Ns), but can have faster computational cost and easier distributed encryption and decryption.

Our PRG-based PCG for SVMT (Section 4.2) can be used as an alternative implementation of the VOLE-based approach. In particular, it can be used for efficiently distributing the EMVP client among k parties along the lines of the generic MPC application discussed above, using only symmetric cryptography. More concretely, using s PCG instances for length-N SVMT, each corresponding to a different entry in $\mathbf{d}$, the distributed client can send to the server a masked version of $\mathbf{d}$ and the server send back a masked version of $\mathbf{M}'\mathbf{d}$ that enables locally computing additive shares of $\mathbf{M}'\mathbf{d} = \mathbf{Mv}$ using the s SVMT correlations.

The small-domain restriction of our PCGs is often not a problem for this use case. First, for some applications a small field $\mathbb{F}$ suffices. Most importantly, the PIR application from [8] (which has the lowest server circuit complexity among all known PIR protocols) specifically uses $\mathbb{F} = \mathbb{F}_4$. Second, even for EMVP over large fields $\mathbb{F}$, security can plausibly hold even if the entries of the decryption key $\mathbf{d}$ come from a small domain $X \subset \mathbb{F}$. In this case, our PCG for $\mathsf{SVMT}_{\mathbb{F},N,X}$ will have good concrete efficiency when k is not too large.

6 Open Questions

Similarly to previous constructions from the literature, the computation and storage costs of our PCF constructions for VOLE-style correlations scale exponentially both with the bit-length of the scalar and with the number of parties. Whether this is inherent is an interesting open question.

A natural first step is to study this question within the framework proposed in this work, namely information-theoretic projection reductions to linear correlations. For VOLE, if we consider the strict notion of projection where the receiver only gets a single coordinate in the codeword, then restricting to a small domain is clearly necessary because computation scales with the number of coordinates. But for more general projection reductions that allow the receiver to get multiple codeword coordinates and apply local postprocessing, the answer seems less obvious.

Use of generative AI tools. The authors made use of generative AI tools to write initial versions of some of the security proofs.

References

[1] Pratyush Agarwal, Varun Narayanan, Shreya Pathak, Manoj Prabhakaran, Vinod M. Prabhakaran, and Mohammad Ali Rehan. Secure non-interactive reduction and spectral analysis of correlations. In *Advances in Cryptology – EUROCRYPT 2022, Part III*, volume 13277 of *Lecture Notes in Computer Science*, pages 276–305. Springer, 2022. 10

[2] Yonatan Aumann and Yehuda Lindell. Security against covert adversaries: Efficient protocols for realistic adversaries. *J. Cryptol.*, 23(2):281–343, 2010. 4

[3] Carsten Baum, Ward Beullens, Cyprien de Saint Guilhem, Shibam Mukherjee, Emmanuela Orsini, Sebastian Ramacher, Christian Rechberger, Lawrence Roy, and Peter Scholl. One tree to rule them all: Optimizing GGM trees and OWFs for post-quantum signatures. *Lecture Notes in Computer Science*, pages 463–493, 2024. 2

[4] Carsten Baum, Lennart Braun, Cyprien Delpech de Saint Guilhem, Michael Klooß, Emmanuela Orsini, Lawrence David Roy, and Peter Scholl. Publicly verifiable zero-knowledge and post-quantum signatures from vole-in-the-head. *Lecture Notes in Computer Science*, pages 581–615, 2023. 2

[5] Donald Beaver. Efficient multiparty protocols using circuit randomization. In *Advances in Cryptology – CRYPTO '91*, pages 420–432. Springer, 1991. 2, 3, 9

[6] Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. Semi-homomorphic encryption and multiparty computation. In Kenneth G. Paterson, editor, *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, volume 6632 of *Lecture Notes in Computer Science*, pages 169–188. Springer, 2011. 2, 9

[7] Fabrice Benhamouda, Elette Boyle, Niv Gilboa, Shai Halevi, Yuval Ishai, and Ariel Nof. Generalized pseudorandom secret sharing and efficient straggler-resilient secure computation. In *Theory of Cryptography – 19th International Conference, TCC 2021, Proceedings, Part II*, volume 13043 of *Lecture Notes in Computer Science*, pages 129–161. Springer, 2021. 8, 13

[8] Fabrice Benhamouda, Caicai Chen, Shai Halevi, Yuval Ishai, Hugo Krawczyk, Tamer Mour, Tal Rabin, and Alon Rosen. Encrypted matrix-vector products from secret dual codes. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security, CCS 2025, Taipei, Taiwan, October 13-17, 2025*, pages 394–408. ACM, 2025. 4, 21

[9] Kallista A. Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In Bhavani Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 1175–1191. ACM, 2017. 4

[10] Dan Boneh, Elette Boyle, Henry Corrigan-Gibbs, Niv Gilboa, and Yuval Ishai. Zero-knowledge proofs on secret-shared data via fully linear pcps. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 67–97. Springer, 2019. 4

[11] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In *Advances in Cryptology – ASIACRYPT 2013*, pages 280–300. Springer, 2013. 6, 16

[12] Elette Boyle, Geoffroy Couteau, Niv Gilboa, and Yuval Ishai. Compressing vector OLE. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 896–912. ACM, 2018. 2, 6, 8, 11

[13] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Peter Rindal, and Peter Scholl. Efficient two-round OT extension and silent non-interactive secure computation. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 291–308. ACM, 2019. 6, 16

[14] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators: Silent OT extension and more. In *Annual International Cryptology Conference (CRYPTO)*, pages 489–518. Springer, 2019. 2, 3, 8, 11, 12

[15] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Correlated pseudorandom functions from variable-density LPN. In *61st Annual IEEE*

Symposium on Foundations of Computer Science (FOCS), pages 1069–1080. IEEE, 2020. 2, 6, 8, 11, 12

[16] Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing. In *Advances in Cryptology – EUROCRYPT 2015*, pages 337–367. Springer, 2015. 7

[17] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In *Public-Key Cryptography – PKC 2014*, pages 501–519. Springer, 2014. 6, 16

[18] Elette Boyle, Lisa Kohl, and Peter Scholl. Homomorphic secret sharing from lattices without FHE. In *EUROCRYPT 2019*, volume 11477 of *LNCS*, pages 3–33, 2019. 2

[19] Lennart Braun, Geoffroy Couteau, Kelsey Melissaris, Mahshid Riahinia, and Elahe Sadeghi. Fast pseudorandom correlation functions from sparse LPN. In *Advances in Cryptology – ASIACRYPT 2025*. Springer, 2025. 2, 8

[20] Dung Bui, Geoffroy Couteau, Pierre Meyer, Alain Passelègue, and Mahshid Riahinia. Fast public-key silent OT and more from constrained Naor-Reingold. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part VI*, volume 14656 of *Lecture Notes in Computer Science*, pages 88–118. Springer, 2024. 2

[21] Ran Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000. 10

[22] Geoffroy Couteau. *An Introduction to Silent Secure Computation*. Springer, 2026. 8

[23] Ronald Cramer, Ivan Damgård, and Yuval Ishai. Share conversion, pseudorandom secret-sharing and applications to secure computation. In *Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005, Proceedings*, volume 3378 of *Lecture Notes in Computer Science*, pages 342–362. Springer, 2005. 3, 4, 6, 7

[24] Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In *Advances in Cryptology – CRYPTO 2012*, pages 643–662. Springer, 2012. 2, 9

[25] Yevgeniy Dodis, Shai Halevi, Ron D. Rothblum, and Daniel Wichs. Spooky encryption and its applications. In *CRYPTO 2016*, volume 9816 of *LNCS*, pages 93–120. Springer, 2016. 2

[26] Niv Gilboa and Yuval Ishai. Compressing cryptographic resources. In *Advances in Cryptology – CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 591–608. Springer, 1999. 3, 4, 7, 8, 10, 12, 13, 19, 31

[27] Aarushi Goel, Mingyuan Wang, and Zhiheng Wang. Multiparty distributed point functions. In *Advances in Cryptology – CRYPTO 2025*, 2025. 7

[28] Oded Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, 2004. 10

[29] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM (JACM)*, 33(4):792–807, 1986. 2, 6, 16

[30] Sebastian Hasler and Pascal Reisert. Pseudorandom correlation functions for multiparty beaver triples from sparse LPN. Cryptology ePrint Archive, Paper 2025/2002, 2025. 8

[31] Johan Håstad, Russell Impagliazzo, Leonid A Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999. 2

[32] Yuval Ishai, Hugo Krawczyk, and Tal Rabin. Compressing correlations via secret replication: PCFs from symmetric cryptography. Information Theoretic Cryptography Conference (ITC 2026), 2026. 1

[33] Nakul Khambhati, Anwesh Bhattacharya, and David Heath. Duty-free bits: Projectivizing garbling schemes. Cryptology ePrint Archive, Paper 2026/476, 2026. 2

[34] Hamidreza Amini Khorasgani, Hemanta K. Maji, and Hai H. Nguyen. Secure noninteractive simulation from arbitrary joint distributions. In *Advances in Cryptology – EUROCRYPT 2022, Part III*, volume 13277 of *Lecture Notes in Computer Science*, pages 306–335. Springer, 2022. 10

[35] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 669–684. ACM, 2013. 6, 16

[36] Vladimir Kolesnikov, Stanislav Peceny, Srinivasan Raghuraman, and Peter Rindal. Stationary syndrome decoding for improved pcgs. In Yael Tauman Kalai and Seny F. Kamara, editors, *Advances in Cryptology - CRYPTO 2025 - 45th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2025, Proceedings, Part I*, volume 16000 of *Lecture Notes in Computer Science*, pages 284–317. Springer, 2025. 8

[37] Yashvanth Kondi. Two-party ECDSA signing at constant communication overhead. *IACR Cryptol. ePrint Arch.*, page 1813, 2025. 2, 4, 16

[38] Yashvanth Kondi, Claudio Orlandi, and Lawrence Roy. Two-round stateless deterministic two-party schnorr signatures from pseudorandom correlation functions. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part I*, Lecture Notes in Computer Science, pages 646–677. Springer, 2023. 2, 3, 4

[39] Zhe Li, Chaoping Xing, Yizhou Yao, and Chen Yuan. Efficient pseudorandom correlation generators over Z/p^k. In Yael Tauman Kalai and Seny F. Kamara, editors, *Advances in Cryptology - CRYPTO 2025 - 45th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2025, Proceedings, Part IV*, volume 16003 of *Lecture Notes in Computer Science*, pages 207–240. Springer, 2025. 8

[40] Daniel Masny and Peter Rindal. Endemic oblivious transfer. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*, pages 309–326. ACM, 2019. 11

[41] Claudio Orlandi, Peter Scholl, and Sophia Yakoubov. The rise of paillier: Homomorphic secret sharing and public-key silent OT. In *Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17-21, 2021, Proceedings, Part I*, volume 12696 of *Lecture Notes in Computer Science*, pages 678–708. Springer, 2021. 2

[42] Lawrence Roy. SoftSpokenOT: Communication–computation tradeoffs in OT extension. In *Annual International Cryptology Conference (CRYPTO)*, pages 255–285. Springer, 2022. 2, 3, 4, 6, 14, 15, 16, 31

[43] Phillipp Schoppmann, Adrià Gascón, Leonie Reichert, and Mariana Raykova. Distributed vector-OLE: Improved constructions and implementation. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 1055–1072. ACM, 2019. 6, 16

A Deferred Proofs

A.1 Efficient Simulation of Linear SNIR

Lemma A.1 (Efficient simulation of linear SNIR)**.** *Let $\mathcal{C}_S$ and $\mathcal{C}_T$ be k-party linear correlations over a finite field $\mathbb{F}$, defined by uniform distributions over linear subspaces C_S and C_T, respectively. If $\mathcal{F} = (f_0, \ldots, f_{k-1})$ is a perfect SNIR from $\mathcal{C}_T$ to $\mathcal{C}_S$ such that every local function f_i is a linear map, then there exists a computationally efficient simulator* Sim *that runs in time polynomial in the dimensions of the correlations and achieves perfect simulation.*

Proof. Let $A \subsetneq \{0, \ldots, k-1\}$ be the subset of corrupted parties. Because $\mathcal{C}_S$ is a uniform distribution over a linear subspace C_S, the marginal distribution of the adversary's shares $\mathbf{s}_A$ in the real execution is uniform over the projected subspace $C_S|_A$.

In the real execution, the target shares are deterministically computed as $\mathbf{t}_A = f_A(\mathbf{s}_A)$, where f_A represents the concatenated linear mappings of the corrupted parties. Thus, conditioned on a specific $\mathbf{t}_A$, the true distribution of $\mathbf{s}_A$ is uniform over the set of preimages:

$$V(\mathbf{t}_A) = \{\mathbf{s}_A \in C_S|_A \mid f_A(\mathbf{s}_A) = \mathbf{t}_A\}$$

Because the subspace $C_S|_A$ and the mapping f_A are entirely linear, $V(\mathbf{t}_A)$ forms an affine subspace over $\mathbb{F}$. To perfectly simulate the adversary's view, we can use a canonical simulator Sim (see Remark 2.2) that samples uniformly from this affine subspace given the input $\mathbf{t}_A$. The simulator proceeds via the following efficient algorithm:

1. **Set up the linear system:** The condition $\mathbf{s}_A \in C_S|_A$ can be expressed as $H_A\mathbf{s}_A = \mathbf{0}$, where H_A is a parity-check matrix for the projected code $C_S|_A$. The condition $f_A(\mathbf{s}_A) = \mathbf{t}_A$ can be written as $F_A\mathbf{s}_A = \mathbf{t}_A$, where F_A is the matrix representation of the linear map f_A.

2. **Combine constraints:** The simulator combines these into a single system of linear equations:
$$\begin{pmatrix} H_A \\ F_A \end{pmatrix} \mathbf{s}_A = \begin{pmatrix} \mathbf{0} \\ \mathbf{t}_A \end{pmatrix}$$

3. **Find a particular solution:** Using standard Gaussian elimination, the simulator finds a particular solution $\mathbf{s}_A^*$ to this system. Because $\mathcal{F}$ is a perfect SNIR, the existence of such a solution is unconditionally guaranteed for any $\mathbf{t}_A$ generated by the ideal functionality.

4. **Sample from the kernel:** The simulator computes a basis for the kernel (null space) of the combined matrix $\begin{pmatrix} H_A \\ F_A \end{pmatrix}$. It then samples a vector $\mathbf{k}$ uniformly at random from this kernel.

5. **Output:** The simulator outputs $\mathbf{s}_A = \mathbf{s}_A^* + \mathbf{k}$.

By the properties of linear algebra, the output $\mathbf{s}_A$ is distributed perfectly uniformly over the affine subspace $V(\mathbf{t}_A)$. Finding the parity-check matrix, performing Gaussian elimination, extracting the kernel basis, and sampling the linear combination all require at most $O(n^3)$ field operations, where n is the length of $\mathbf{s}_A$. Therefore, Sim is computationally efficient and achieves perfect simulation. □

Remark A.1 (A general sufficient condition for efficient simulation)**.** *It is natural to ask whether the existence of an efficient simulator for an SNIR is implied by the correlations themselves being "easy to sample." In particular, one might wonder if a reverse-samplable target correlation $\mathcal{C}_T$, where the conditional distribution of honest shares given corrupted shares can be efficiently sampled, guarantees efficient simulation. This is not the case. In the SNIR ideal execution, the target correlation $\mathcal{C}_T$ is sampled entirely by the ideal functionality, and the simulator is only given $\mathbf{t}_A$. The simulator never needs to reconstruct or sample the honest parties' target shares $\mathbf{t}_{\bar{A}}$.*

Instead, the computational bottleneck for the simulator lies only in the conditional inversion of the local reduction mappings. Specifically, the canonical simulator must efficiently sample from the conditional distribution of $\mathbf{s}_A \leftarrow \mathcal{C}_S|_A$ subject to the constraint that $f_A(\mathbf{s}_A) = \mathbf{t}_A$. In our linear SNIR instances (Lemma A.1), this is efficient because the preimage of a linear map over a linear subspace forms an easily solvable affine space. However,

for a general non-linear SNIR, even if both $\mathcal{C}_S$ and $\mathcal{C}_T$ are trivial to sample and reverse-sample, finding an $\mathbf{s}_A$ that satisfies $f_A(\mathbf{s}_A) = \mathbf{t}_A$ could be computationally intractable (e.g., if f_A acts as a one-way function). Thus, efficient simulation depends on the local functions f_i and their interaction with the source support, rather than only with the samplability of the source and target correlations.

A.2 Corruptible SNIR

Lemma A.2 (Perfect SNIR is corruptible)**.** *Let $\mathcal{C}_S$ be a k-party source correlation and $\mathcal{C}_T$ be a k-party target correlation. If $\mathcal{F} = (f_0, \ldots, f_{k-1})$ is a perfect secure non-interactive reduction (SNIR) from $\mathcal{C}_T$ to $\mathcal{C}_S$, then $\mathcal{F}$ is also a perfect corruptible SNIR from the corruptible target correlation $\mathcal{C}'_T$ to the corruptible source correlation $\mathcal{C}'_S$.*

Proof. Let $\mathcal{F} = (f_0, \ldots, f_{k-1})$ be a perfect SNIR from $\mathcal{C}_T$ to $\mathcal{C}_S$. We want to construct a corruptible simulator Sim_{corr} such that the real corruptible experiment $\mathsf{Real}^{corr}_{\mathcal{F},A}$ is perfectly indistinguishable from the ideal corruptible experiment $\mathsf{Ideal}^{corr}_{\mathsf{Sim}_{corr},A}$.

In the corruptible setting, the adversary actively chooses its source shares $\mathbf{s}^*_A$ from the support of $\mathcal{C}_S|_A$. We define the corruptible simulator Sim_{corr} to simply evaluate the local reduction functions on the adversary's chosen shares to extract the target shares:

$$\mathbf{t}^*_A = \mathsf{Sim}_{corr}(\mathbf{s}^*_A) := f_A(\mathbf{s}^*_A)$$

We must prove that the distribution of the honest parties' outputs $\mathbf{t}_{\bar{A}}$ is identical in both worlds. That is, for any fixed $\mathbf{s}^*_A$ in the support of $\mathcal{C}_S|_A$, we must show:

$$\Pr_{\mathbf{s}\leftarrow\mathcal{C}_S}[f_{\bar{A}}(\mathbf{s}_{\bar{A}}) = \mathbf{t}_{\bar{A}} \mid \mathbf{s}_A = \mathbf{s}^*_A] = \Pr_{\mathbf{t}\leftarrow\mathcal{C}_T}[\mathbf{t}_{\bar{A}} = \mathbf{t}_{\bar{A}} \mid \mathbf{t}_A = f_A(\mathbf{s}^*_A)]$$

By the definition of a standard perfect SNIR, there exists a simulator Sim such that the joint distributions of the real and ideal executions are identical. For any specific values $(\mathbf{s}^*_A, \mathbf{t}_{\bar{A}})$, we have:

$$\Pr[\mathsf{Real}_{\mathcal{F},A} = (\mathbf{s}^*_A, \mathbf{t}_{\bar{A}})] = \Pr[\mathsf{Ideal}_{\mathsf{Sim},A} = (\mathbf{s}^*_A, \mathbf{t}_{\bar{A}})]$$

In the real execution, $\mathbf{t}_A$ is deterministically equal to $f_A(\mathbf{s}_A)$. Using the canonical simulator from Remark 2.2, the ideal execution perfectly matches the real execution, and the simulator $\mathsf{Sim}(\mathbf{t}_A)$ outputs $\mathbf{s}^*_A$ with non-zero probability *only if* $\mathbf{t}_A = f_A(\mathbf{s}^*_A)$. Therefore, we can rewrite the probability of the ideal execution by evaluating it strictly at the matching target share $\mathbf{t}_A = f_A(\mathbf{s}^*_A)$:

$$\Pr[\mathsf{Real}_{\mathcal{F},A} = (\mathbf{s}^*_A, \mathbf{t}_{\bar{A}})] = \Pr_{\mathbf{t}\leftarrow\mathcal{C}_T}[\mathbf{t}_A = f_A(\mathbf{s}^*_A) \wedge \mathbf{t}_{\bar{A}} = \mathbf{t}_{\bar{A}}] \cdot \Pr[\mathsf{Sim}(f_A(\mathbf{s}^*_A)) = \mathbf{s}^*_A]$$

To find the marginal probability of the adversary's view $\mathbf{s}^*_A$, we sum the above equation over all possible values of $\mathbf{t}_{\bar{A}}$:

$$\Pr_{\mathbf{s}\leftarrow\mathcal{C}_S}[\mathbf{s}_A = \mathbf{s}^*_A] = \Pr_{\mathbf{t}\leftarrow\mathcal{C}_T}[\mathbf{t}_A = f_A(\mathbf{s}^*_A)] \cdot \Pr[\mathsf{Sim}(f_A(\mathbf{s}^*_A)) = \mathbf{s}^*_A]$$

We can now determine the conditional distribution of the honest parties' output in the real corruptible experiment. By dividing the joint probability by the marginal probability, we get:

$$\Pr_{\mathbf{s}\leftarrow\mathcal{C}_S}[f_{\bar{A}}(\mathbf{s}_{\bar{A}}) = \mathbf{t}_{\bar{A}} \mid \mathbf{s}_A = \mathbf{s}^*_A] = \frac{\Pr_{\mathbf{t}\leftarrow\mathcal{C}_T}[\mathbf{t}_A = f_A(\mathbf{s}^*_A) \wedge \mathbf{t}_{\bar{A}} = \mathbf{t}_{\bar{A}}] \cdot \Pr[\mathsf{Sim}(f_A(\mathbf{s}^*_A)) = \mathbf{s}^*_A]}{\Pr_{\mathbf{t}\leftarrow\mathcal{C}_T}[\mathbf{t}_A = f_A(\mathbf{s}^*_A)] \cdot \Pr[\mathsf{Sim}(f_A(\mathbf{s}^*_A)) = \mathbf{s}^*_A]}$$

Because $\mathbf{s}^*_A$ is in the support of $\mathcal{C}_S|_A$, its marginal probability is non-zero, meaning the term $\Pr[\mathsf{Sim}(f_A(\mathbf{s}^*_A)) = \mathbf{s}^*_A]$ is strictly greater than 0. We cancel this term from the numerator and denominator, yielding:

$$\Pr_{\mathbf{s}\leftarrow\mathcal{C}_S}[f_{\bar{A}}(\mathbf{s}_{\bar{A}}) = \mathbf{t}_{\bar{A}} \mid \mathbf{s}_A = \mathbf{s}^*_A] = \frac{\Pr_{\mathbf{t}\leftarrow\mathcal{C}_T}[\mathbf{t}_A = f_A(\mathbf{s}^*_A) \wedge \mathbf{t}_{\bar{A}} = \mathbf{t}_{\bar{A}}]}{\Pr_{\mathbf{t}\leftarrow\mathcal{C}_T}[\mathbf{t}_A = f_A(\mathbf{s}^*_A)]}$$

This is exactly the definition of the conditional probability $\Pr_{\mathbf{t}\leftarrow\mathcal{C}_T}[\mathbf{t}_{\bar{A}} = \mathbf{t}_{\bar{A}} \mid \mathbf{t}_A = f_A(\mathbf{s}^*_A)]$, which is the distribution sampled by the ideal functionality $\mathcal{C}'_T$ given the corruptible simulator's extracted query $\mathbf{t}^*_A$. Thus, $\mathsf{Real}^{corr}_{\mathcal{F},A} \equiv \mathsf{Ideal}^{corr}_{\mathsf{Sim}_{corr},A}$, concluding the proof. □

A.3 PCG/PCF via Projection

The following theorem formalizes the step of compressing the source correlation of a projection reduction via a PCG or PCF.

Theorem A.1 (PCG/PCF composition via projection)**.** *Let $\mathcal{C}_S$ be an n-party correlation and $\mathcal{C}_T$ a k-party correlation. Let $\mathcal{Q}$ be a distribution over $(2^{[n]})^k$. Suppose we have a projection reduction $(\mathcal{Q}, R)$ from $\mathcal{C}_T$ to $\mathcal{C}_S$ and a PCG/PCF for $\mathcal{C}_S$ with exact correctness, in the sense that every expanded output is in the support of $\mathcal{C}_S$. Then the natural composition, where the seed generator samples $Q = (Q_0, \ldots, Q_{k-1}) \leftarrow \mathcal{Q}$ and gives target party i the source PCG/PCF seeds corresponding to the virtual parties in Q_i, is a PCG/PCF for $\mathcal{C}_T$.*

Proof. We prove the PCG case. The PCF case is similar, except that the same argument is applied to the adaptively queried identifiers.

Let $(\mathsf{Gen}_S, \mathsf{Expand}_S)$ be the source PCG for $\mathcal{C}_S$. The composed generator samples $Q = (Q_0, \ldots, Q_{k-1}) \leftarrow \mathcal{Q}$ and source-PCG seeds $K = (K_0, \ldots, K_{n-1}) \leftarrow \mathsf{Gen}_S$. Target party i receives (Q_i, K_{Q_i}). To expand, party i computes $s_h = \mathsf{Expand}_{S,h}(K_h)$ for every $h \in Q_i$, and outputs $R_i(Q_i, s_{Q_i})$.

Fix a strict corrupted target coalition $A \subsetneq [k]$. For a sampled projection Q, define the corresponding virtual source coalition

$$Z = Z(Q, A) = \bigcup_{i\in A} Q_i \subseteq [n].$$

The corrupted target parties receive exactly (Q_A, K_Z) for $Q_A = (Q_i)_{i\in A}$.

We use the source PCG in its corruptible-output formulation: for a virtual coalition Z, the source-PCG simulator outputs the corrupted seed view K_Z, and the corrupted source output fixed in $\mathcal{C}'_S$ is the value $s_Z = \mathsf{Expand}_{S,Z}(K_Z) := (\mathsf{Expand}_{S,h}(K_h))_{h\in Z}$.

We now construct the composed corruptible simulator. It samples Q_A according to its marginal distribution induced by $\mathcal{Q}$ and sets $Z = \bigcup_{i\in A} Q_i$. If $Z \subsetneq [n]$, it runs the corruptible

simulator of the source PCG for coalition Z to obtain a simulated corrupted seed view K_Z. It then defines $s_Z = \mathsf{Expand}_{S,Z}(K_Z)$. If $Z = [n]$, it simply samples $K \leftarrow \mathsf{Gen}_S$ honestly, sets $K_Z = K$, and defines $s_Z = \mathsf{Expand}_{S,[n]}(K)$. By exact correctness, this s_Z lies in the support of $\mathcal{C}_S|_Z$.

The simulator forms the corrupted projected-source view $u_A = \big(Q_i, s_{Q_i}\big)_{i\in A}$ and fixes the corrupted target output in the corruptible target correlation $\mathcal{C}'_T$ to $t^*_A = R_A(u_A)$. It outputs the corrupted composed seed view (Q_A, K_Z).

We prove security via the following hybrids.

Hybrid 0. This is the real composed PCG execution. We sample $Q \leftarrow \mathcal{Q}$ and $K \leftarrow \mathsf{Gen}_S$. The adversary receives (Q_A, K_Z), where $Z = \bigcup_{i\in A} Q_i$. The honest target parties output

$$R_{\bar{A}}(Q_{\bar{A}}, s_{Q_{\bar{A}}}), \qquad s = \mathsf{Expand}_S(K).$$

Hybrid 1. Sample $Q \leftarrow \mathcal{Q}$ and set $Z = \bigcup_{i\in A} Q_i$. If $Z \subsetneq [n]$, replace the real source-PCG execution by the ideal corruptible source-PCG execution for coalition Z: the adversary receives simulated corrupted seeds K_Z, the corrupted source output is fixed to $s_Z = \mathsf{Expand}_{S,Z}(K_Z)$, and the honest virtual source outputs $s_{\bar{Z}}$ are sampled from $\mathcal{C}_S$ conditioned on s_Z. If $Z = [n]$, define Hybrid 1 to be the same as Hybrid 0. The honest target outputs are again computed as $R_{\bar{A}}(Q_{\bar{A}}, s_{Q_{\bar{A}}})$.

Hybrids 0 and 1 are computationally indistinguishable by source-PCG security when $Z \subsetneq [n]$. When $Z = [n]$, they are identical.

Hybrid 2. Keep the corrupted seed view (Q_A, K_Z) and the induced corrupted projected-source view

$$u_A = \big(Q_i, s_{Q_i}\big)_{i\in A}.$$

Instead of computing honest target outputs from the projected source sample, sample them from the corruptible target correlation $\mathcal{C}'_T$ conditioned on the corrupted target value

$$t^*_A = R_A(u_A).$$

In Hybrid 1, the seed view K_Z influences the honest target outputs only through the expanded source values s_Z, which are already contained in u_A. Thus, conditioned on u_A, the honest projected-source view has exactly the distribution of the honest part of the corruptible source correlation $\mathcal{Q}(\mathcal{C}_S)'$ conditioned on u_A. By Lemma A.2, the perfect SNIR R is also a perfect corruptible SNIR from $\mathcal{C}'_T$ to $\mathcal{Q}(\mathcal{C}_S)'$. Hence the distribution of

$$R_{\bar{A}}(Q_{\bar{A}}, s_{Q_{\bar{A}}})$$

in Hybrid 1 is identical to the honest target distribution in Hybrid 2 conditioned on t^*_A.

Hybrid 2 is exactly the ideal corruptible execution for the composed PCG: the simulator outputs the corrupted composed seed view (Q_A, K_Z) and fixes the corrupted target output to t^*_A. Hence the real composed PCG execution is computationally indistinguishable from the ideal corruptible execution.

For PCFs, the same proof is applied with oracle access to identifiers. The projection Q and the keys are sampled once. For each queried identifier id, the simulator computes

the corrupted source output induced by the corrupted keys at id, fixes the corresponding corrupted target output, and uses source-PCF security for the hybrid replacing hidden source outputs by ideal conditional samples. This proves the theorem. □

B Block-Linear Correlations

In this section we present a generalization of the PCF for linear correlations of Gilboa and Ishai [26] that considers the case where each of the n parties can be assigned an arbitrary subset (or "block") of n' codeword coordinates. While this can be reduced to the n'-party PCF construction from [26] by assigning each of the n parties the corresponding PCF keys, here we give a direct generalization that can offer better efficiency. This generalization is useful for matching the efficiency of Roy's PCF for subfield-VOLE [42].

Definition B.1 (Block-minimal codewords)**.** *Let $\mathbb{F}$ be a finite field, let $C \subseteq \mathbb{F}^{n'}$ be a linear code, and let $\mathcal{T} = (T_0, \ldots, T_{n-1})$ be subsets of $[n']$. We assume, without loss of generality, that $\bigcup_{i\in[n]} T_i = [n']$.*

- ***Block-support:*** *For a codeword $c \in C$, its block-support with respect to $\mathcal{T}$ is*

$$\mathrm{supp}_{\mathcal{T}}(c) = \{i \in [n] : c|_{T_i} \neq 0\}.$$

- ***Block-minimal codeword:*** *A nonzero codeword $c \in C$ is $\mathcal{T}$-minimal if there does not exist a nonzero $c' \in C$ such that*

$$\mathrm{supp}_{\mathcal{T}}(c') \subsetneq \mathrm{supp}_{\mathcal{T}}(c).$$

We define $W_{\mathcal{T}}(C)$ as the set of all $\mathcal{T}$-minimal codewords in C, taking exactly one representative for each one-dimensional subspace (e.g., the one whose first nonzero coordinate is 1).

Definition B.2 (Block-based replicated correlation)**.** *Using the notation above, the target and source correlations are defined as follows.*

- ***Target correlation $\mathcal{C}_C$:*** *The n-party correlation $\mathcal{C}_C$ samples a codeword $c \leftarrow C$ uniformly at random. Party $i \in [n]$ receives $c|_{T_i} \in \mathbb{F}^{T_i}$. If a coordinate belongs to several sets T_i, the corresponding entry of c is given consistently to all such parties.*

- ***Source correlation $\widehat{\mathcal{C}}_W$:*** *The replicated correlation $\widehat{\mathcal{C}}_W$ samples independent uniformly random scalars $r = (r_w)_{w\in W_{\mathcal{T}}(C)} \leftarrow \mathbb{F}^{|W_{\mathcal{T}}(C)|}$. Party $i \in [n]$ receives the subset of scalars $\widehat{c}_i = (r_w)_{w\in W_i}$, where $W_i = \{w \in W_{\mathcal{T}}(C) : i \in \mathrm{supp}_{\mathcal{T}}(w)\}$.*

To securely generate a block-based target correlation from the above replicated source correlation, we will need an extra spanning condition. This condition does not appear in the special case from [26] since it is guaranteed to hold in this case.

Theorem B.1 (Generalized GI99 SNIR for block-based correlations)**.** *Let $\mathbb{F}$ be a finite field, let $C \subseteq \mathbb{F}^{n'}$ be a linear code, and let $\mathcal{T} = (T_0, \ldots, T_{n-1})$ be (not necessarily disjoint) subsets of $[n']$. For every $A \subseteq [n]$, define*

$$C_A = \{c \in C : c|_{T_i} = 0 \text{ for every } i \in A\} = \{c \in C : \mathrm{supp}_{\mathcal{T}}(c) \cap A = \emptyset\}$$

and

$$W_{\bar{A}} = \{w \in W_{\mathcal{T}}(C) : \mathrm{supp}_{\mathcal{T}}(w) \cap A = \emptyset\}.$$

Suppose that for every strict subset $A \subsetneq [n]$,

$$\mathbf{Span}_{\mathbb{F}}(W_{\bar{A}}) = C_A. \qquad (\star)$$

Then there exists a perfectly secure non-interactive reduction (SNIR) from the block-based target correlation $\mathcal{C}_C$ to the replicated source correlation $\widehat{\mathcal{C}}_W$.

Proof. We define the non-interactive reduction $R = (f_0, \ldots, f_{n-1})$ as follows. Given its share of the replicated correlation $\widehat{c}_i = (r_w)_{w \in W_i}$, party i locally computes

$$f_i(\widehat{c}_i) = \sum_{w \in W_i} r_w \, w|_{T_i}.$$

For every $w \notin W_i$, we have $i \notin \mathrm{supp}_{\mathcal{T}}(w)$, and hence $w|_{T_i} = 0$. Therefore the local computation indeed depends only on the local view:

$$f_i(\widehat{c}_i) = \left(\sum_{w \in W_{\mathcal{T}}(C)} r_w w \right) \Big|_{T_i}.$$

We now prove perfect security against a corrupted coalition $A \subset [n]$ (correctness follows from the case $A = \emptyset$). Let $B = [n] \setminus A$. Define the coalition view map

$$\pi_A : C \to \prod_{i \in A} \mathbb{F}^{T_i}, \qquad \pi_A(c) = (c|_{T_i})_{i \in A}.$$

Thus $C_A = \ker(\pi_A)$.

Let $W_A = W_{\mathcal{T}}(C) \setminus W_{\bar{A}} = \{w \in W_{\mathcal{T}}(C) : \mathrm{supp}_{\mathcal{T}}(w) \cap A \neq \emptyset\}$, capturing the replicated keys visible to A. Define the visible-key map $\phi_A : \mathbb{F}^{W_A} \to \pi_A(C)$ by

$$\phi_A((r_w)_{w \in W_A}) = \pi_A \left(\sum_{w \in W_A} r_w w \right).$$

The map ϕ_A is surjective. Indeed, since $W_T(C)$ spans C, every $c \in C$ can be written as

$$c = \sum_{w \in W_T(C)} r_w w.$$

Moreover, every $w \in W_{\bar{A}}$ satisfies $\pi_A(w) = 0$. Hence

$$\pi_A(c) = \pi_A \left(\sum_{w \in W_A} r_w w \right),$$

so the images of the visible codewords span $\pi_A(C)$.

The simulator. We are now ready to explicitly describe the simulator. It receives the corrupted parties' target shares $y_A = \pi_A(c)$ for a uniformly sampled $c \leftarrow C$. It then samples $(r_w)_{w \in W_A}$ uniformly at random from the affine preimage $\phi_A^{-1}(y_A)$, and outputs the corresponding replicated-key views to the corrupted parties.

We show that this is a perfect SNIR simulator. For $U \subseteq W_T(C)$, define $\rho_U : \mathbb{F}^U \to C$:

$$\rho_U((r_w)_{w \in U}) = \sum_{w \in U} r_w w.$$

For the real execution, let

$$c_V = \rho_{W_A}((r_w)_{w \in W_A}), \qquad c_H = \rho_{W_{\bar{A}}}((r_w)_{w \in W_{\bar{A}}}).$$

Then the generated codeword is $c = c_V + c_H$. Since every $w \in W_{\bar{A}}$ satisfies $\pi_A(w) = 0$, we have

$$\pi_A(c) = \pi_A(c_V) = \phi_A((r_w)_{w \in W_A}).$$

Therefore, conditioned on $\pi_A(c) = y_A$, the visible keys $(r_w)_{w \in W_A}$ are uniformly distributed over $\phi_A^{-1}(y_A)$, exactly as in the ideal experiment with the above simulator.

It remains to compare the distribution of the honest parties' target shares. By the spanning assumption ($\star$), when the hidden coefficients $(r_w)_{w \in W_{\bar{A}}}$ are sampled uniformly, the hidden codeword c_H is uniform over C_A.

Fix any visible-key vector $(r_w)_{w \in W_A} \in \phi_A^{-1}(y_A)$. The honest parties' target shares in the real execution are distributed as $\pi_B(c_V + c_H)$, where $c_H \leftarrow C_A$ and $\pi_B(c) = (c|_{T_i})_{i \in B}$. Thus they are uniform over the affine space $\pi_B(c_V + C_A)$.

This affine set depends only on y_A, not on the particular visible-key preimage. Indeed, if c_V and c_V' are two visible codewords satisfying $\pi_A(c_V) = \pi_A(c_V') = y_A$, then $\pi_A(c_V - c_V') = 0$, and so $c_V - c_V' \in C_A$. It follows that $c_V + C_A = c_V' + C_A$.

In the ideal execution, a uniform codeword $c \leftarrow C$ conditioned on $\pi_A(c) = y_A$ is uniform over an affine coset $c_0 + C_A$ for any $c_0 \in C$ satisfying $\pi_A(c_0) = y_A$. Taking $c_0 = c_V$, the honest parties' target shares are uniform over $\pi_B(c_V + C_A)$, which is exactly the same distribution as in the real execution.

Thus, conditioned on the corrupted target shares y_A, the simulator produces the same joint distribution of corrupted replicated keys and honest target shares as the real execution. Since this holds for every strict coalition $A \subsetneq [n]$, the reduction is a perfect SNIR. □

Corollary B.1 (PCF for binary subfield VOLE)**.** *Let $E = \mathbb{F}_{2^m}$ and let $X = E$. Assuming a PRF with input domain $[N]$ and output domain $\mathbb{F}_2$, there exists a PCF for $\mathsf{VOLE}_{E,\mathbb{F}_2,N,E}$ in which the sender holds 2^m PRF keys and the receiver holds $2^m - 1$ PRF keys. Evaluating one VOLE coordinate requires one PRF evaluation per held key.*

Proof. Consider the $\mathbb{F}_2$-linear code $C = \{(a\alpha + b)_{\alpha \in E} : a \in \mathbb{F}_2,\ b \in E\}$, where each symbol in E is viewed as an m-dimensional vector over $\mathbb{F}_2$. Let $\mathcal{T} = (T_\alpha)_{\alpha \in E}$ be the partition of the $m|E|$ binary coordinates into blocks corresponding to the evaluation points $\alpha \in E$. The $\mathcal{T}$-minimal codewords are exactly

$$w_\beta = (\alpha + \beta)_{\alpha \in E}, \qquad \beta \in E.$$

Indeed, a nonzero constant word has full block-support, whereas w_β vanishes exactly on the block T_β and is nonzero on every other block; no nonzero degree-one word can vanish on more than one evaluation point. Thus $|W_{\mathcal{T}}(C)| = |E| = 2^m$.

We verify the hypothesis of Theorem B.1. For $A \subseteq E$, let C_A and $W_{\bar{A}}$ be as in Theorem B.1. If $A = \emptyset$, then the words w_β span C: $w_0 = (\alpha)_{\alpha \in E}$, and $w_\beta + w_0 = (\beta)_{\alpha \in E}$ gives every constant word. If $A = \{\gamma\}$, then $C_A = \mathbf{Span}_{\mathbb{F}_2}\{w_\gamma\}$, and the only minimal word avoiding T_γ is w_γ. If $|A| \geq 2$, then $C_A = \{0\}$, since a nonzero word $a\alpha + b$ with $a \in \mathbb{F}_2$ cannot vanish at two distinct points; also $W_{\bar{A}} = \emptyset$. Hence, for every strict coalition A, $\mathbf{Span}_{\mathbb{F}_2}(W_{\bar{A}}) = C_A$. By Theorem B.1, the replicated correlation indexed by $\{w_\beta : \beta \in E\}$ perfectly securely generates the above block line correlation.

Replacing each replicated random bit by a PRF output gives the PCF. The key counts are $|E| = 2^m$ for the sender and $|E| - 1 = 2^m - 1$ for the receiver. □