Half-Tree: Halving the Cost of Tree Expansion in COT and DPF

Xiaojie Guo[*,†] Kang Yang[*] Xiao Wang[‡] Wenhao Zhang[‡]
Xiang Xie[§,¶] Jiang Zhang[*] Zheli Liu[†]

March 7, 2023

Abstract

GGM tree is widely used in the design of correlated oblivious transfer (COT), subfield vector oblivious linear evaluation (sVOLE), distributed point function (DPF), and distributed comparison function (DCF). Often, the cost associated with GGM tree dominates the computation and communication of these protocols. In this paper, we propose a suite of optimizations that can reduce this cost by half.

- **Halving the cost of COT and sVOLE.** Our COT protocol introduces extra correlation to each level of a GGM tree used by the state-of-the-art COT protocol. As a result, it reduces both the number of AES calls and the communication by half. Extending this idea to sVOLE, we are able to achieve similar improvement with either halved computation or halved communication.
- **Halving the cost of DPF and DCF.** We propose improved two-party protocols for the distributed generation of DPF/DCF keys. Our tree structures behind these protocols lead to more efficient full-domain evaluation and halve the communication and the round complexity of the state-of-the-art DPF/DCF protocols.

All protocols are provably secure in the random-permutation model and can be accelerated based on fixed-key AES-NI. We also improve the state-of-the-art schemes of puncturable pseudorandom function (PPRF), DPF, and DCF, which are of independent interest in dealer-available scenarios.

1 Introduction

The construction of Goldreich-Goldwasser-Micali (GGM) tree [GGM84] yields a pseudorandom function (PRF) family from any length-doubling pseudorandom generator (PRG). In this construction, a PRF key serves as a root and is expanded into a full binary tree, where each non-leaf node defines two child nodes from its PRG output. The PRF output for an input bit-string is defined as the leaf node labeled by this bit-string. GGM tree has been adapted widely for various cryptographic applications, especially in recent years.

A recent appealing application of GGM tree is to build efficient pseudorandom correlation generators (PCGs) [BCGI18, SGRR19, BCG+19a, YWL+20, BCG+20, WYKW21], e.g., correlated oblivious transfer (COT), subfield vector oblivious linear evaluation (sVOLE), etc. In this context, a GGM tree essentially serves as a puncturable pseudorandom function (PPRF). PCGs serve as

[*]State Key Laboratory of Cryptology. **Email:** `yangk@sklc.org`, `jiangzhang09@gmail.com`
[†]Nankai University. **Email:** `xiaojie.guo@mail.nankai.edu.cn`, `liuzheli@nankai.edu.cn`
[‡]Northwestern University. **Email:** `wangxiao@cs.northwestern.edu`, `wenhao.zhang@northwestern.edu`
[§]Shanghai Qi Zhi Institute. **Email:** `xiexiangiscas@gmail.com`
[¶]PADO Labs.

Protocol	Computation	Communication	# Rounds
COT (§ 4.1.1)	$2\times$	$2\times$	–
sVOLE (§ 4.1.2)	$2\times$	$1 \sim 2\times$	–
sVOLE (§ 4.2)	$1.33\times$	$2\times$	–
DPF (§ 5.2)	$1.33\times$	$3\times$	$2\times$
DCF (§ 5.3)	$1.6\times$	$2 \sim 3\times$	$2\times$

Table 1: **Improvements of our protocols in the random-permutation model.** Computation is measured as the number of fixed-key AES calls. In sVOLE, communication varies as per two field sizes $|\mathbb{F}|$ and $|\mathbb{K}|$. In DCF protocol, communication varies as per the range size $|\mathcal{R}|$ of comparison functions.

essential building blocks for secure multi-party computation (MPC) (e.g., [HK21, GMW87]), zero-knowledge proofs (e.g., [WYKW21, DIO21, BMRS21]), private set intersection (e.g., [GPR$^+$21, RS21]), etc. Another related application of GGM tree is to build function secret sharing (FSS). In an FSS scheme, a dealer produces two keys, each defining an additive secret sharing of the full-domain evaluation result of some function f without revealing the parameters of f. FSS is very useful even for a simple f, and the dealer can be emulated using an MPC protocol. A distributed point function (DPF) [GI14] is an FSS scheme for the family of point functions $f^{\bullet}_{\alpha,\beta}(x)$ that output β if $x = \alpha$ and 0 otherwise. DPF has found various applications, including RAM-based secure computation [Ds17], two-server PIR [GI14, BGI16], private heavy hitters [BBC$^+$21], oblivious linear evaluation (OLE) [BCG$^+$20], etc. One important variation of DPF is distributed comparison function (DCF), which is an FSS scheme for the family of comparison functions $f^{<}_{\alpha,\beta}(x)$ that output β if $x < \alpha$ and 0 otherwise. DCF has been applied to design mixed-mode MPC [BGI19, BCG$^+$21], secure machine-learning inference [GKCG22], etc.

In all applications above, the cost associated with GGM tree can often be significant. For example, in the most recent silent OT protocol [CRR21], distributing GGM-tree-related correlations takes more than 70% of the computation and essentially all communication. Similar bottlenecks have also been observed in DPF. For example, in the DPF-based secure RAM computation [Ds17], local expansion of DPF keys takes a majority of the time as well.

1.1 Our Contribution

We propose a suite of *half-trees* as tailored alternatives for several GGM-tree-based protocols, leading to halved computation/communication/round complexity (Table 1, detailed complexity is compared in the sections). Our constructions work in the random-permutation model (RPM) [RS08, BHKR13], which can be efficiently instantiated via, e.g., fixed-key AES-NI.

Correlated GGM trees for half-cost COT and sVOLE. We introduce correlated GGM (cGGM), a tree structure leading to both improved computation and communication in COT. It has an invariant that all same-level nodes sum up to the same global offset. We keep this invariant by setting a left child as the hash of its parent and the associated right child as the parent minus the left child. By plugging this tree into the state-of-the-art COT protocols [YWL$^+$20, CRR21], we can prove the security of the whole protocol in the random-permutation model by carefully choosing the hash function. Compared to the optimized GGM tree [GKWY20], this tree reduces the number of random-permutation calls and the communication by half.

Using cGGM tree, we can realize sVOLE for any large field and its subfield. This protocol reduces the computation of the prior protocols [BCG$^+$19a, WYKW21] by $2\times$ using a field-based random permutation. However, it only halves the communication when the subfield size is significantly smaller than the field size. Then, we modify our cGGM tree to obtain a pseudorandom correlated

	Assump.	Corr.	Computation	Communication (bits)	
				$P_0 \to P_1$	$P_1 \to P_0$
[BCG+22]	ROM	sVOLE	m RO calls	$2t(\log \frac{m}{t} - 1)\lambda + 3t \log \lvert\mathbb{K}\rvert$	$t \log \lvert\mathbb{F}\rvert$
	Ad-hoc[1]	sVOLE	m RP calls + $0.5m$ RO calls		
This work	RPM	COT	m RP calls	$t(\log \frac{m}{t} - 1)\lambda + \lambda$	–
		sVOLE	m RP calls	$t(\log \frac{m}{t} - 1) \log \lvert\mathbb{K}\rvert + \lambda$	$t(\log \frac{m}{t} + 1) \log \lvert\mathbb{F}\rvert$
		sVOLE	$1.5m$ RP calls	$t(\log \frac{m}{t} - 2)\lambda + 3t \log \lvert\mathbb{K}\rvert + \lambda$	$t \log \lvert\mathbb{F}\rvert$

[1] Security relies on the conjecture that the punctured result of the RPM-based UPF is unpredictable. This UPF uses GGM-style tree expansion $G(x) := \mathsf{H}_0(x) \parallel \mathsf{H}_1(x)$ for $\mathsf{H}_0(x) := \mathsf{H}(x) \oplus x$ and $\mathsf{H}_1(x) := \mathsf{H}(x) + x \bmod 2^\lambda$.

Table 2: **Comparison with the concurrent work.** "RO/ROM" (resp., "RP/RPM") is for random oracle (resp., permutation) and the model. P_0 is the sender with a global key, and P_1 is the receiver. Assume weight-t regular LPN noises in sVOLE extension with output length m, field $\mathbb{F}$, and extension field $\mathbb{K}$. Computation is measured by the amount of symmetric-key operations, and there is also LPN-related computation in practice. Communication is measured by assuming P_0 and P_1 have access to random precomputed tuples: (i) [BCG+22]: $t \log \frac{m}{t}$ COTs (+ t sVOLEs, for general sVOLE extension), (ii) our COT extension: $t \log \frac{m}{t}$ COTs, (iii) our first sVOLE extension: $t(\log \frac{m}{t} + 1)$ sVOLEs, and (iv) our second sVOLE extension: $t \log \frac{m}{t}$ COTs + t sVOLEs.

GGM (pcGGM) tree, which is similar to a cGGM tree but has pseudorandom leaves. In contrast, pcGGM tree leads to a $2\times$ saving in communication and a $1.33\times$ saving in computation.

Halved communication and round complexity in distributed key generation of DPF and DCF. We introduce another binary tree structure, which adapts our pcGGM tree into a secretly shared form. This tree leads to a new DPF scheme with an improved distributed key generation protocol. This DPF protocol reduces the computation, communication, and round complexity of the prior work roughly by $1.33\times$, $3\times$, and $2\times$, respectively. When the range of point functions is a general ring, this shared tree allows simpler secure computation than the prior works in terms of the last correction word.

We also use an extended version of this shared pcGGM tree to design a new DCF scheme also with an improved distributed key generation protocol. The tree expansion in our DCF is much simpler than the prior work [BCG+21], where each parent node has to quadruple in length to produce additional correction words. In our extended shared pcGGM tree, this expansion factor in length is two or three, and the resulting additional correction words are more 2PC-friendly. When used in our DCF protocol with typical parameters, this extended tree leads to about $1.6\times$, $2 \sim 3\times$, and $2\times$ savings in terms of computation, communication, and round complexity in contrast to the prior work.

1.2 Concurrent Work

Recently, Boyle et al. [BCG+22] propose two unpredictable punctured functions (UPFs) that can be converted to PPRF with additional $0.5N$ RO calls for N-sized domain. Their first UPF construction needs N RO calls and is provably secure while the second UPF construction needs N RP calls but relies on an ad-hoc conjecture. For m-sized sVOLE tuples, the sVOLE extension protocols based on their proposal either needs $1.5m$ RO calls, or needs m RP calls plus $0.5m$ RO calls. They also propose an sVOLE extension protocol that is based on a stronger variation of UPF and requires m RO calls in total.

In contrast, our protocol is secure in the random-permutation model without any conjecture. Our COT protocol, as a special case of sVOLE protocol, only requires m RP calls and can reduce communication by half; our two sVOLE protocols need m or $1.5m$ RP calls with different levels of communication reduction. More importantly, we also demonstrate how the idea can be applied to DPF/DCF protocols as well.

In Table 2, we compare the cost of sVOLE extension in the two works. The sVOLE extension in both works can be easily turned into the extension of random OTs via the standard transformation [IKNP03, Bea95, BCG+19a]. If we regard one (length-preserving) RO call as two RP calls according to the XOR-based construction of [BN18], our work also beats the concurrent one in terms of concrete efficiency.

2 Preliminaries

2.1 Notation

Let λ denote the computational security parameter. $n = n(\lambda)$ means that $n \in \mathbb{N}$ is polynomial in λ. Let $\mathsf{negl}(\cdot)$ denote an unspecified negligible function and $\log(\cdot)$ denote the logarithm in base 2. Let $x \leftarrow S$ denote sampling x uniformly at random from a finite set S. Let $[a, b) := \{a, \ldots, b-1\}$ and $[a, b] := \{a, \ldots, b\}$. Let $\mathbb{G}$ (resp., $\mathcal{R}$) denote finite group (resp., ring). We use bold lowercase letters (e.g., $\mathbf{a}$) for vectors. For $i \geq 0$, let $\mathbf{a}^{(i)}$ denote the i-th entry of vector $\mathbf{a}$. Let $\mathbf{unit}_{\mathbb{G}}(n, \alpha, \beta) \in \mathbb{G}^n$ denote the vector whose α-th entry is β and others are 0. For some field $\mathbb{F}$ and irreducible polynomial $f(X) \in \mathbb{F}[X]$, let $\mathbb{K} = \mathbb{F}[X]/f(X)$ denote an extension field. For some $n \in \mathbb{N}$, we interchangeably use $\mathbb{F}_{2^n}$, $\mathbb{F}_2^n$, and $\{0,1\}^n$, where $\oplus$ is for bitwise-XOR. For some bit-string $x \in \{0,1\}^n$, let $\mathsf{lsb}(x)$ denote its least significant bit (LSB), $\mathsf{hb}(x)$ denote its high $n-1$ bits, and x_i denote its i-th bit such that x_1 is the most significant one. We use $\|$ for bit-string concatenation and $\circ$ for function composition. Let $\mathsf{Convert}_{\mathbb{G}} : \{0,1\}^* \to \mathbb{G}$ denote a function that maps random strings to pseudorandom $\mathbb{G}$ elements (see Appendix F.1 for its implementation).

Binary trees. In an n-level tree, let X_i^j denote the j-th node on its i-level for $i \in [1, n]$ and $j \in [0, 2^i)$. We can write the superscript j into i-bit decomposition, i.e., $X_i^{j_1 \cdots j_i} := X_i^j$. When a node $X_i^j \in \{0,1\}^n$, we can decompose it into a seed $s_i^j := \mathsf{hb}(X_i^j) \in \{0,1\}^{n-1}$ and a control bit $t_i^j := \mathsf{lsb}(X_i^j) \in \{0,1\}$ such that $X_i^j = (s_i^j \,\|\, t_i^j)$. We usually omit the superscript j if it is the i-bit prefix of a path $\alpha \in \{0,1\}^n$ of particular interest in a given context. For completeness, let X_0 denote the root. For some $i \in [1, n]$ and $b \in \{0,1\}$, let K_i^b denote the sum of the 2^{i-1} b-side (i.e., left or right) nodes on the i-th level.

Secret sharings. For some additive Abelian group $\mathbb{G}$ and $x \in \mathbb{G}$, we use $\langle x \rangle^{\mathsf{A}}$ to mean that x is additively shared between two parties and call it a secret for short. For some secret $\langle x \rangle^{\mathsf{A}}$ for $x \in \mathbb{G}$ and party $b \in \{0,1\}$, let $\langle x \rangle_b^{\mathsf{A}} \in \mathbb{G}$ denote the secret share of the party b such that $x = \langle x \rangle_0^{\mathsf{A}} + \langle x \rangle_1^{\mathsf{A}}$. We abbreviate $\langle x \rangle^{\mathsf{A}}$ to $\langle x \rangle$ and $\langle x \rangle_b^{\mathsf{A}}$ to $\langle x \rangle_b$ if $\mathbb{G} = \{0,1\}^n$. For some secret $\langle x \rangle$ for $x \in \{0,1\}^n$ and efficiently computable (possibly non-linear) Boolean circuit $\mathsf{H} : \{0,1\}^n \to \{0,1\}^*$, let $\mathsf{H}(\langle x \rangle)$ denote such a *linear evaluation* that returns a secret $\langle y \rangle$ with share $\langle y \rangle_b := \mathsf{H}(\langle x \rangle_b)$ for each $b \in \{0,1\}$.

2.2 Security Model and Functionalities

We use the *universal composability* (UC) framework [Can01] to prove security in the presence of a semi-honest, static adversary. We say that a protocol Π *UC-realizes* an ideal functionality $\mathcal{F}$ if for any probabilistic polynomial-time (PPT) adversary $\mathcal{A}$, there exists a PPT adversary (simulator) $\mathcal{S}$ such that for any PPT environment $\mathcal{Z}$ with arbitrary auxiliary input z, the output distribution of

Functionality $\mathcal{F}_{\mathsf{sVOLE}}$

Parameters: Field $\mathbb{F}$ and its extension field $\mathbb{K}$.

Initialize: Upon receiving (init) from P_0 and P_1, sample $\Delta \leftarrow \mathbb{K}$ if P_0 is honest; otherwise, receive $\Delta \in \mathbb{K}$ from the adversary. Store Δ and send it to P_0. Ignore all subsequent (init) commands.

Extend: This functionality allows polynomially many (extend) commands. Upon receiving (extend, m) from P_0 and P_1:

1. If P_0 is honest, sample $\mathbf{v} \leftarrow \mathbb{K}^m$; otherwise, receive $\mathbf{v} \in \mathbb{K}^m$ from the adversary.
2. If P_1 is honest, sample $\mathbf{u} \leftarrow \mathbb{F}^m$, and compute $\mathbf{w} := \mathbf{v} + \mathbf{u} \cdot \Delta \in \mathbb{K}^m$; otherwise, receive $(\mathbf{u}, \mathbf{w}) \in \mathbb{F}^m \times \mathbb{K}^m$ from the adversary, and recompute $\mathbf{v} := \mathbf{w} - \mathbf{u} \cdot \Delta \in \mathbb{K}^m$.
3. Send $\mathbf{v}$ to P_0 and $(\mathbf{u}, \mathbf{w})$ to P_1.

Global-key queries: If P_1 is corrupted, upon receiving $(\mathsf{guess}, \Delta')$, where $\Delta' \in \mathbb{K}$, from the adversary, send $(\mathsf{success})$ to the adversary if $\Delta = \Delta'$; send (fail) to the adversary otherwise.

Figure 1: Functionality for subfield VOLE.

$\mathcal{Z}$ in the *real-world* execution where the parties interact with $\mathcal{A}$ and execute Π is computationally indistinguishable from the output distribution of $\mathcal{Z}$ in the *ideal-world* execution where the parties interact with $\mathcal{S}$ and $\mathcal{F}$.

Our protocols use the functionality $\mathcal{F}_{\mathsf{sVOLE}}$ (Figure 1) of subfield vector oblivious linear evaluation. If $\mathbb{K} = \mathbb{F}_{2^\lambda}$ and $\mathbb{F} = \mathbb{F}_2$, $\mathcal{F}_{\mathsf{sVOLE}}$ degenerates to the COT functionality $\mathcal{F}_{\mathsf{COT}}$ in [YWL+20]. If $\mathbb{K} = \mathbb{F}$, $\mathcal{F}_{\mathsf{sVOLE}}$ serves as the VOLE functionality in [BCGI18, SGRR19, RS21]. We omit the session IDs and sub-session IDs in the functionalities for simplicity. By convention, we can write sVOLE tuples as two-party *information-theoretic message authentication codes* (IT-MACs) [NNOB12, DPSZ12]. Let $\Delta_b \in \mathbb{K}$ denote the global key of one party P_b. P_b authenticates a value $x \in \mathbb{F}$ of the other party P_{1-b} by sampling a uniform one-time key $\mathsf{K}_b[x] \leftarrow \mathbb{K}$ and giving to P_{1-b} the MAC $\mathsf{M}_{1-b}[x] := \mathsf{K}_b[x] + x \cdot \Delta_b \in \mathbb{K}$. If identity $b \in \{0, 1\}$ is clear in a given context, we write Δ, $\mathsf{K}[x]$, and $\mathsf{M}[x]$ for Δ_b, $\mathsf{K}_b[x]$, and $\mathsf{M}_{1-b}[x]$, respectively.

2.3 Circular Correlation Robustness

Circular correlation robustness (CCR) [CKKZ12, GKWY20] is the security notion first introduced for the circuit garbling with Free-XOR optimization [KS08], where there exists a global key Δ offsetting the inputs and outputs of some function H. [GKWY20] showed that a CCR function H can be constructed from a fixed-key block cipher (e.g., AES) modeled as random permutation and a *linear orthomorphism*[1]. In this construction, it takes one block-cipher call to invoke a CCR function.

Definition 1 (Circular Correlation Robustness, [GKWY20])**.** *Let* $\mathsf{H} : \{0,1\}^\lambda \to \{0,1\}^\lambda$, χ *be a distribution on* $\{0,1\}^\lambda$, *and* $\mathcal{O}^{\mathsf{ccr}}_{\mathsf{H},\Delta}(x, b) := \mathsf{H}(x \oplus \Delta) \oplus b \cdot \Delta$ *be an oracle for* $x, \Delta \in \{0,1\}^\lambda$ *and* $b \in \{0,1\}$. H *is* (t, q, ρ, ϵ)*-CCR if, for any distinguisher* $\mathcal{D}$ *running in time at most* t *and making*

[1] A mapping $\sigma : \mathbb{G} \to \mathbb{G}$ for an additive Abelian group $\mathbb{G}$ is a linear orthomorphism if (i) σ is a permutation, (ii) $\sigma(x + y) = \sigma(x) + \sigma(y)$ for any $x, y \in \mathbb{G}$, and (iii) $\sigma'(x) := \sigma(x) - x$ is also a permutation. [GKWY20] presents two efficient instantiations of σ (with well-defined efficient σ^{-1}, σ', and σ'^{-1}): (i) if $\mathbb{G}$ is a field, $\sigma(x) := c \cdot x$ for some $c \neq 0, 1 \in \mathbb{G}$, and (ii) if $\mathbb{G} = \{0,1\}^n$, $\sigma(x) = \sigma(x_L \,\|\, x_R) := (x_L \oplus x_R) \,\|\, x_L$ where x_L and x_R are the left and right halves of x.

at most q queries to $\mathcal{O}^{\mathsf{ccr}}_{\mathsf{H},\Delta}(\cdot,\cdot)$, and any χ with min-entropy at least ρ, it holds that

$$\left| \Pr_{\Delta \leftarrow \chi}\left[\mathcal{D}^{\mathcal{O}^{\mathsf{ccr}}_{\mathsf{H},\Delta}(\cdot,\cdot)}(1^\lambda) = 1\right] - \Pr_{f \leftarrow \mathcal{F}_{\lambda+1,\lambda}}\left[\mathcal{D}^{f(\cdot,\cdot)}(1^\lambda) = 1\right] \right| \le \epsilon,$$

where $\mathcal{D}$ cannot query both $(x,0)$ and $(x,1)$ for any $x \in \{0,1\}^\lambda$.

In this work, $\mathcal{D}$ can only make CCR queries with *restricted* forms, which are reminiscent of those in the Half-Gate garbling scheme [ZRE15]. We defer the formal definition of these restricted queries to Appendix A.

2.4 Function Secret Sharing

A function secret sharing (FSS) is a secret sharing scheme where a dealer distributes the shares of a function f to multiple parties, and each party can use its share to *locally* compute the share of $f(x)$ for any *public* x in the domain of f. In this work, we focus on two-party FSS schemes.

Definition 2 (Function Secret Sharing, [BGI16, BCG+21])**.** *For a family $\mathcal{F}_{\mathcal{X},\mathbb{G}}$ of functions with domain $\mathcal{X}$ and range $\mathbb{G}$, where $\mathbb{G}$ is an Abelian group, a two-party FSS scheme with key space $\mathcal{K}_0 \times \mathcal{K}_1$ has the following syntax:*

- $(k_0, k_1) \leftarrow \mathsf{Gen}(1^\lambda, \hat{f})$. *On input 1^λ and the description $\hat{f} \in \{0,1\}^*$ of a function $f \in \mathcal{F}_{\mathcal{X},\mathbb{G}}$, output a key pair $(k_0, k_1) \in \mathcal{K}_0 \times \mathcal{K}_1$.*
- $f_b(x) \leftarrow \mathsf{Eval}(b, k_b, x)$. *On input the party identifier $b \in \{0,1\}$, the party's key $k_b \in \mathcal{K}_b$, and a point $x \in \mathcal{X}$, output the share $f_b(x) \in \mathbb{G}$.*

A two-party FSS scheme $(\mathsf{Gen}, \mathsf{Eval})$ is secure for the function family $\mathcal{F}_{\mathcal{X},\mathbb{G}}$ with leakage $\mathsf{Leak} : \{0,1\}^* \to \{0,1\}^*$ if the following properties hold.

- **Correctness**. *For any function $f \in \mathcal{F}_{\mathcal{X},\mathbb{G}}$ with description $\hat{f}$, and any $x \in \mathcal{X}$,*

$$\Pr\left[(k_0, k_1) \leftarrow \mathsf{Gen}(1^\lambda, \hat{f}) : \textstyle\sum_{b \in \{0,1\}} \mathsf{Eval}(b, k_b, x) = f(x)\right] = 1.$$

- **Security**. *There exists a PPT simulator Sim such that, for any function $f \in \mathcal{F}_{\mathcal{X},\mathbb{G}}$ with the description $\hat{f}$, any $b \in \{0,1\}$, and any PPT adversary $\mathcal{A}$,*

$$\Bigg| \Pr\left[(k_0, k_1) \leftarrow \mathsf{Gen}(1^\lambda, \hat{f}) : \mathcal{A}(1^\lambda, k_b) = 1\right] - \Pr\left[k_b \leftarrow \mathsf{Sim}(1^\lambda, b, \mathsf{Leak}(\hat{f})) : \mathcal{A}(1^\lambda, k_b) = 1\right] \Bigg| \le \mathsf{negl}(\lambda).$$

By default, the leakage $\mathsf{Leak}(\hat{f})$ only involves the domain and the range of f. The following two special FSS schemes have been proposed in [BGI16, BCG+21].

Distributed Point Functions (DPFs). A two-party *distributed point function* with domain $\mathcal{X}$ and range $\mathbb{G}$ is a two-party FSS scheme $(\mathsf{DPF.Gen}, \mathsf{DPF.Eval})$ for the function family $\mathcal{F}_{\mathcal{X},\mathbb{G}} = \{f^\bullet_{\alpha,\beta}\}_{\alpha \in \mathcal{X}, \beta \in \mathbb{G}}$ where $f^\bullet_{\alpha,\beta}$ is a *point function* such that $f^\bullet_{\alpha,\beta}(\alpha) = \beta$, and $f^\bullet_{\alpha,\beta}(x) = 0$ for $x \neq \alpha \in \mathcal{X}$.

Distributed Comparison Functions (DCFs). A two-party *distributed comparison function* with domain $\mathcal{X}$ and range $\mathbb{G}$ is a two-party FSS scheme $(\mathsf{DCF.Gen}, \mathsf{DCF.Eval})$ for the function family $\mathcal{F}_{\mathcal{X},\mathbb{G}} = \{f^<_{\alpha,\beta}\}_{\alpha \in \mathcal{X}, \beta \in \mathbb{G}}$ where $f^<_{\alpha,\beta}$ is a *comparison function* such that $f^<_{\alpha,\beta}(x) = \beta$ if $x < \alpha \in \mathcal{X}$, and $f^<_{\alpha,\beta}(x) = 0$ otherwise.

3 Technical Overview

3.1 Improved COT/sVOLE from Correlated GGM Trees

Since COT/sVOLE can be constructed from its "single-point" version using an appropriate LPN assumption, we focus on single-point COT/sVOLE, where the vector $\mathbf{u}$ in a COT/sVOLE tuple $\mathbf{w} = \mathbf{v} + \mathbf{u} \cdot \Delta$ has exactly one non-zero entry.

Correlated OT from correlated GGM. The core idea behind our single-point COT protocol is that, instead of using a GGM tree with pseudorandom nodes as the state-of-the-art works, our protocol uses a *correlated* GGM (cGGM) tree where *the sum of all same-level nodes equals a global offset* Δ. This invariant can be maintained by using a generalized Davies-Meyer construction with a hash function H: every parent x has left child $\mathsf{H}(x)$ and right child $x - \mathsf{H}(x)$. cGGM tree leads to two improvements: (i) no additional hash computation is needed for every right child so that the computation is halved, and (ii) if the global offset Δ (i.e., the difference between two first-level nodes) is set up by precomputed random COT tuples, the single-point COT protocol sends only λ bits per level, in contrast to 2λ bits from a standard OT per level in the state-of-the-art works.

To explain our second improvement in detail, we first recall the prior construction from the perspective of GGM tree. In this construction, the sender holds an n-level GGM tree, whose 2^n leaves in $\mathbb{F}_{2^\lambda}$ forms a vector $\mathbf{v} \in \mathbb{F}_{2^\lambda}^{2^n}$. The receiver with a punctured point $\alpha = \alpha_1 \dots \alpha_n \in \{0,1\}^n$ uses, for each $i \in [1, n]$, a standard OT to select the XOR of all $\overline{\alpha}_i$-side nodes on the i-th level. From these n XORs, the receiver recovers the n off-path GGM-tree nodes just leaving the path α and use these n nodes to recover all leaves except the α-th one, corresponding to a vector $\mathbf{w} \in \mathbb{F}_{2^\lambda}^{2^n}$ with the punctured entry $\mathbf{w}^{(\alpha)}$. The sender samples $\Delta \leftarrow \mathbb{F}_{2^\lambda}$, defines its output as $(\Delta, \mathbf{v})$, and sends $\psi := \Delta \oplus (\oplus_{j \in [0, 2^n)} \mathbf{v}^{(j)}) \in \mathbb{F}_{2^\lambda}$ to the receiver. The receiver patches $\mathbf{w}^{(\alpha)} := \psi \oplus (\oplus_{j \neq \alpha} \mathbf{w}^{(j)})$ and defines its output as $(\mathbf{u}, \mathbf{w})$ for $\mathbf{u} = \mathbf{unit}_{\mathbb{F}_2}(2^n, \alpha, 1)$. The computation is dominated by the full GGM-tree expansion while the communication is from n parallel standard OTs, which need n precomputed COT tuples via the standard technique [IKNP03, Bea95].

In contrast, our cGGM-tree single-point COT, where the global offset in a cGGM tree coincides with the global key in the n precomputed COT tuples, can directly use these tuples. For each level $i \in [1, n]$, let $\mathsf{M}[r_i] = \mathsf{K}[r_i] \oplus r_i \cdot \Delta$ be such a tuple where the sender has $(\Delta, \mathsf{K}[r_i]) \in \mathbb{F}_{2^\lambda} \times \mathbb{F}_{2^\lambda}$ and the receiver has $(r_i, \mathsf{M}[r_i]) \in \mathbb{F}_2 \times \mathbb{F}_{2^\lambda}$, and $K_i^b \in \mathbb{F}_{2^\lambda}$ be the XOR of all b-side nodes for $b \in \{0, 1\}$. To select $K_i^{\overline{\alpha}_i}$ as in the prior construction, the receiver sends $\overline{\alpha}_i \oplus r_i$ to the sender, receives back $c_i := K_i^0 \oplus \mathsf{K}[r_i] \oplus (\overline{\alpha}_i \oplus r_i) \cdot \Delta$, and computes

$$c_i \oplus \mathsf{M}[r_i] = K_i^0 \oplus \mathsf{K}[r_i] \oplus (\overline{\alpha}_i \oplus r_i) \cdot \Delta \oplus \mathsf{M}[r_i] = K_i^0 \oplus \overline{\alpha}_i \cdot \Delta = K_i^{\overline{\alpha}_i},$$

where the last equality holds since the cGGM tree uses Δ as global offset. For each level, the sender sends λ bits to the receiver, only a half of the 2λ bits in a standard OT. When the point α is random, the message $\overline{\alpha}_i \oplus r_i$ can be avoided as well. The single-point COT outputs are defined as in the prior construction, except that the receiver locally patches $\mathbf{w}^{(\alpha)} := \oplus_{j \neq \alpha} \mathbf{w}^{(j)}$.

The security against the semi-honest sender is straightforward. However, a subtle issue arises in proving the security against the semi-honest receiver. Note that the environment $\mathcal{Z}$ can observe the global key Δ from the honest sender's output and use it to distinguish the two worlds. Let $\{X_i^{\alpha_1 \dots \alpha_{i-1} \overline{\alpha}_i}\}_{i \in [1,n]}$ be the cGGM-tree off-path nodes recovered by the receiver. In the real world, these off-path nodes satisfy the consistency with Δ: for $j \in [2, n]$, $X_j^{\alpha_1 \dots \alpha_{j-1} \overline{\alpha}_j}$ equals

$$\mathsf{H}\left(\Delta \oplus \bigoplus\nolimits_{i \in [1, j-1]} X_i^{\alpha_1 \dots \alpha_{i-1} \overline{\alpha}_i}\right) \oplus \overline{\alpha}_j \cdot \left(\Delta \oplus \bigoplus\nolimits_{i \in [1, j-1]} X_i^{\alpha_1 \dots \alpha_{i-1} \overline{\alpha}_i}\right). \tag{1}$$

However, this consistency does not hold in the ideal world where $\{c_i\}_{i \in [1,n]}$ sent by the simulator are sampled at random so that the n off-path nodes will be independently uniform in the ideal world.

Thus, $\mathcal{Z}$ can trivially distinguish the two worlds by using the known Δ to check (1). Our security proof addresses this issue by carefully constructing H from a random permutation, allowing global-key queries in the single-point COT functionality, and programming the random permutation and its inverse to keep the consistency. The intuition is that, to distinguish the two worlds, $\mathcal{Z}$ must query the random permutation or its inverse with Δ-related transcripts. Thus, the simulator can observe these queries and extract every potential Δ from them. Using global-key queries, the simulator checks whether an extracted Δ matches that in the single-point COT functionality or not. If so, it immediately programs the two permutation oracles using this Δ so that they are consistent with the simulated $\{c_i\}_{i\in[1,n]}$. Similar proof technique in the random-oracle model have been used in TinyOT [NNOB12, HSS17].

Subfield VOLE from correlated GGM. We further propose a cGGM-based blueprint of single-point sVOLE for field $\mathbb{F}$ and its exponentially large extension $\mathbb{K}$. In this blueprint, we construct an n-level cGGM tree from a hash function $\mathsf{H} : \mathbb{K} \to \mathbb{K}$ so that all nodes are in $\mathbb{K}$, and extend the spirit of our single-point COT. The spirit is that the equality $\mathbf{w}^{(\alpha)} = \mathbf{v}^{(\alpha)} \oplus \Delta$ at the punctured point α automatically holds by embedding Δ into a cGGM tree. For single-point sVOLE, we want to likewise keep $\mathbf{w}^{(\alpha)} = \mathbf{v}^{(\alpha)} + \beta \cdot \Delta$ for some $\beta \in \mathbb{F}^*$ and $\Delta \in \mathbb{K}$ at the punctured point α. However, we cannot use $\beta \cdot \Delta$, which is unknown to the sender, as the cGGM-tree global offset. Instead, we can define this offset as the sender's additive share of $\beta \cdot \Delta$ so that the receiver can correct the automatically preserved result at the point α by using its additive share of $\beta \cdot \Delta$.

In detail, the two parties use a random sVOLE tuple $\mathsf{M}[\beta] = \mathsf{K}[\beta] + \beta \cdot \Delta$ for the $\beta \cdot \Delta$ term, where the sender has $(\Delta, \mathsf{K}[\beta]) \in \mathbb{K} \times \mathbb{K}$ and the receiver has $(\beta, \mathsf{M}[\beta]) \in \mathbb{F}^* \times \mathbb{K}$. The sender uses $\mathsf{K}[\beta]$ as the global offset of its cGGM tree, and the receiver selects, for each level i, the sum of all $\overline{\alpha}_i$-side nodes. For the i-th level, let $K_i^b \in \mathbb{K}$ be the sum of all b-side nodes for $b \in \{0, 1\}$, and let the two parties have access to a special sVOLE tuple[2] $\mathsf{M}[r_i] = \mathsf{K}[r_i] + r_i \cdot \mathsf{K}[\beta]$, where the sender has $\mathsf{K}[r_i] \in \mathbb{K}$ and the receiver has $(r_i, \mathsf{M}[r_i]) \in \mathbb{F}_2 \times \mathbb{K}$. The sender sends $c_i := \mathsf{K}[r_i] + K_i^0 \in \mathbb{K}$ to the receiver, who defines $\overline{\alpha}_i := r_i$ and can compute

$$(-1)^{r_i} \cdot (-\mathsf{M}[r_i] + c_i) = (-1)^{\overline{\alpha}_i} \cdot (K_i^0 - \overline{\alpha}_i \cdot \mathsf{K}[\beta]) = K_i^{\overline{\alpha}_i},$$

where the last equality holds due to the cGGM invariant. The n selected sums allow the receiver to recover, in a top-down manner, the n off-nodes with respect to α and the 2^n cGGM leaves except the α-th one. The sender defines $\mathbf{v} \in \mathbb{K}^{2^n}$ from its 2^n cGGM-tree leaves, while the receiver defines $\mathbf{w} \in \mathbb{K}^{2^n}$ from the α-exclusive $2^n - 1$ leaves and the locally patched punctured leaf $\mathbf{w}^{(\alpha)} := \mathsf{M}[\beta] - \sum_{j \neq \alpha} \mathbf{w}^{(j)} = \mathsf{M}[\beta] - (\sum_{j \neq \alpha} \mathbf{w}^{(j)} + \mathbf{v}^{(\alpha)}) + \mathbf{v}^{(\alpha)} = \mathbf{v}^{(\alpha)} + \beta \cdot \Delta$. If the sender defines its output as $(\Delta, \mathbf{v})$ and the receiver defines its output as $(\mathbf{u}, \mathbf{w})$ for $\mathbf{u} := \mathbf{unit}_{\mathbb{F}}(2^n, \alpha, \beta)$, the two parties share a single-point sVOLE correlation.

Our cGGM-based single-point sVOLE protocol also has the issue in proving the security against the semi-honest receiver as the environment $\mathcal{Z}$ sees Δ from the honest sender's output. $\mathcal{Z}$ can compute the cGGM offset $\mathsf{K}[\beta] = \mathsf{M}[\beta] - \beta \cdot \Delta$ and, to distinguish the two worlds, check if the consistency (1) holds for $\mathsf{K}[\beta]$ or not. As in our cGGM-based single-point COT, our simulator addresses this issue by extracting every possible $\mathsf{K}[\beta]$ and the associated $\Delta = \beta^{-1} \cdot (\mathsf{M}[\beta] - \mathsf{K}[\beta])$, querying the single-point sVOLE functionality with Δ, and programming the random permutation and its inverse if the global-key query succeeds.

Subfield VOLE from pseudorandom correlated GGM. There is another single-point sVOLE blueprint [BCG+19a, WYKW21] basing its security on the pseudorandomness of GGM-tree nodes: for some path $\alpha \in \{0, 1\}^n$, the n off-path nodes and the α-th leaf are pseudorandom. Our cGGM

[2] The special sVOLE tuples for selecting n sums can be obtained from n precomputed random sVOLE tuples by the receiver sending $n \cdot \log |\mathbb{F}|$ bits.

tree cannot be used in this blueprint since its same-level nodes are correlated under the global offset. However, we observe that a cGGM tree can be modified into a *pseudorandom* cGGM (pcGGM) tree with the required pseudorandomness.

In an n-level pcGGM tree, we preserve the cGGM invariant for the $\mathbb{F}_{2^\lambda}$ nodes on the first $n-1$ levels, i.e., using a hash function $\mathsf{H}' : \mathbb{F}_{2^\lambda} \to \mathbb{F}_{2^\lambda}$ and Davies-Meyer construction to keep that all same-level nodes are XORed to a global offset $\Delta \in \mathbb{F}_{2^\lambda}$. Nevertheless, we break the last-level correlation in the pcGGM tree: every parent $x \in \mathbb{F}_{2^\lambda}$ on the $(i-1)$-th level has left child $\mathsf{H}'(x)$ and right child $\mathsf{H}'(x \oplus 1)$. In sVOLE protocols for $\mathbb{K} \neq \mathbb{F}_{2^\lambda}$, the pcGGM leaves will be further converted by the function $\mathsf{Convert}_{\mathbb{K}} : \mathbb{F}_{2^\lambda} \to \mathbb{K}$.

Our core observation for arguing the pseudorandomness of the $n+1$ pcGGM nodes is that the inputs of the hash function H' are of CCR forms. More specifically, a global $\Delta \in \mathbb{F}_{2^\lambda}$ offsets the two first-level nodes of the pcGGM tree and induces the first $n-1$ off-path nodes $\{X_i^{\alpha_1 \ldots \alpha_{i-1} \overline{\alpha}_i}\}_{i \in [1,n-1]}$ according to (1) for H'. Meanwhile, the last off-path node $X_n^{\alpha_1 \ldots \alpha_{n-1} \overline{\alpha}_n} \in \mathbb{F}_{2^\lambda}$ and the α-th pcGGM leaf $X_n \in \mathbb{F}_{2^\lambda}$ come from two hash calls of the following form: for $b \in \{0,1\}$,

$$X_n^{\alpha_1 \ldots \alpha_{n-1} b} = \mathsf{H}' \Big(\Delta \oplus \big(\textstyle\bigoplus_{i \in [1,n-1]} X_i^{\alpha_1 \ldots \alpha_{i-1} \overline{\alpha}_i} \big) \oplus b \Big).$$

Intuitively, we can use a CCR hash function H' to argue the pseudorandomness of the n off-path nodes and the α-th leaf, which is sufficient for the single-point sVOLE blueprint. The challenge in this security reduction is to show that the CCR queries to H' are legal (i.e., no $(x,0)$ and $(x,1)$ for the same x) with overwhelming probability. We address this challenge by resorting to the observation that these inputs are *restricted* so that they are well-structured and are not arbitrarily chosen by the corrupted receiver (the only case where we need the pseudorandomness). Such restricted inputs are reminiscent of the "naturally derived keys" [ZRE15, GKWY20] in the Half-Gate garbling scheme so that we can bound the probability similarly. We defer the details to Appendix A. Note that even if one uses $\mathsf{Convert}_{\mathbb{K}}$ to map the leaves into $\mathbb{K}$, the pseudorandomness of these nodes still holds due to the pseudorandomness of $\mathsf{Convert}_{\mathbb{K}}$.

By plugging our pcGGM tree into the prior single-point sVOLE blueprint, we obtain a more efficient protocol. The improvement owes to the cGGM invariant in its first $n-1$ levels. In terms of communication, the receiver can use $n-1$ precomputed random COTs to select the XORs on these levels and recover the first $n-1$ levels of the sender's pcGGM tree; in contrast, the prior protocols use a standard OT per level due to the two pseudorandom XORs. For the last level in our protocol, the two parties also need a standard OT due to the broken correlation of the two sums. Given the random-permutation-based CCR hash functions in [GKWY20], our pcGGM-based single-point sVOLE protocol is secure in the random-permutation model. In particular, this protocol can implement the single-point sVOLE functionality *without* global-key queries since $\Delta \in \mathbb{F}_{2^\lambda}$ is only used in the pcGGM tree and is not included in the sender's output.

3.2 DPF/DCF from Shared Pseudorandom Correlated GGM Trees

DPF scheme and protocol. Using a pcGGM-like trick, we present a new DPF scheme, followed by a more efficient distributed protocol. Recall that, in the prior DPF scheme [BGI16], there are two parties sharing an n-level GGM-style tree where the n nodes on some path $\alpha \in \{0,1\}^n$ are pseudorandom with LSB one, and others are zero. Then, the two-party shares of the α-th leaf mask the DPF payload $\beta \in \mathbb{G}$. Our core observation is that we need the pseudorandom α-th leaf to hide β, but the internal pseudorandom on-path nodes are not mandatory. Instead, the two parties can share an n-level pcGGM-style tree (say, spcGGM tree) where (i) the root X_0 and the first $n-1$ on-path nodes equal a global offset $\Delta \in \mathbb{F}_{2^\lambda}$ with $\mathsf{lsb}(\Delta) = 1$, (ii) the last on-path node (i.e., the

α-th leaf) is pseudorandom with LSB one, and (iii) other nodes are zero. As in the prior scheme, the per-party share of this tree is compressed as a key including an XOR share of the root and $n+1$ public *pseudorandom* correction words.

We explain our construction of these correction words in detail. To keep the invariant (i), the spcGGM tree uses a correction procedure different from the prior one. For each level $i \in [1, n-1]$ with a public correction word $\mathsf{CW}_i \in \mathbb{F}_{2^\lambda}$, and $b \in \{0,1\}$, the b-side secret child of the $(i-1)$-th on-path secret node $\langle X_{i-1} \rangle = \langle s_{i-1} \,\|\, t_{i-1} \rangle$ is defined as follows:

$$\langle X_i^{\alpha_1 \ldots \alpha_{i-1} b} \rangle := \mathsf{H}'(\langle X_{i-1} \rangle) \oplus b \cdot \langle X_{i-1} \rangle \oplus \langle t_{i-1} \rangle \cdot \mathsf{CW}_i.$$

Solving this *linear* equation for the *public* CW_i under the constraint (i), we have

$$\mathsf{CW}_i = \mathsf{H}'(\langle X_{i-1} \rangle_0) \oplus \mathsf{H}'(\langle X_{i-1} \rangle_1) \oplus \overline{\alpha}_i \cdot \Delta.$$

As for (ii), we use a public correction word $\mathsf{CW}_n = (\mathsf{HCW}, \mathsf{LCW}^0, \mathsf{LCW}^1) \in \mathbb{F}_{2^{\lambda-1}} \times \mathbb{F}_2 \times \mathbb{F}_2$ to follow the same last-level correction as the prior work. For $b \in \{0,1\}$, define a function $\mathsf{H}'_b(\cdot) := \mathsf{H}'(\cdot \oplus b)$ and the b-side secret child of the $(n-1)$-th on-path secret node $\langle X_{n-1} \rangle = \langle s_{n-1} \,\|\, t_{n-1} \rangle$ as follows:

$$\langle X_n^{\alpha_1 \ldots \alpha_{n-1} b} \rangle := \mathsf{H}'_b(\langle X_{n-1} \rangle) \oplus \langle t_{n-1} \rangle \cdot (\mathsf{HCW} \,\|\, \mathsf{LCW}^b).$$

Solving this *linear* equation for the *public* CW_n under the constraint (i) and (iii),

$$\begin{aligned} \mathsf{HCW} &= \mathsf{hb}\Big(\mathsf{H}'_{\overline{\alpha}_n}(\langle X_{n-1} \rangle_0) \oplus \mathsf{H}'_{\overline{\alpha}_n}(\langle X_{n-1} \rangle_1)\Big), \\ \forall b \in \{0,1\} : \mathsf{LCW}^b &= \mathsf{lsb}\Big(\mathsf{H}'_b(\langle X_{n-1} \rangle_0) \oplus \mathsf{H}'_b(\langle X_{n-1} \rangle_1)\Big) \oplus \alpha_n \oplus \overline{b}. \end{aligned} \tag{2}$$

Note that the n off-path secret nodes $\{\langle X_i^{\alpha_1 \ldots \alpha_{i-1} \overline{\alpha}_i} \rangle\}_{i \in [1,n]}$ are zero secrets according to the above correction procedures. As a result, the two parties hold identical shares of these n off-path nodes and their subtrees, given that the share of a subtree is fully determined by the share of its root (i.e., an off-path node) and the public correction words. This implies the constraint (iii). Finally, the $(n+1)$-th public correction word is defined from the secret α-th leaf $\langle X_n \rangle = \langle s_n \,\|\, t_n \rangle$ and the function $\mathsf{Convert}_{\mathbb{G}} : \mathbb{F}_{2^{\lambda-1}} \to \mathbb{G}$ as follows:

$$\mathsf{CW}_{n+1} = (\langle t_n \rangle_0 - \langle t_n \rangle_1) \cdot \Big(\mathsf{Convert}_{\mathbb{G}}(\langle s_n \rangle_1) - \mathsf{Convert}_{\mathbb{G}}(\langle s_n \rangle_0) + \beta\Big) \in \mathbb{G},$$

where the DPF payload β is masked by the XOR shares of the α-th leaf.

The DPF security primarily follows from that the first n correction words are of CCR forms, i.e., for $i \in [0, n-1]$, $\langle X_i \rangle_0 \oplus \langle X_i \rangle_1 = X_i = \Delta$ according to the XOR secret sharing scheme and the invariant (i). The Δ-circular correlation in $\mathsf{CW}_1, \ldots, \mathsf{CW}_{n-1}$ is obvious for either corrupted party. In CW_n, the honest party's H' inputs also differ from the corrupted party's H' inputs by Δ. Intuitively, these n correction words use CCR responses as one-time pads, and the underlying CCR queries are as structured as those in the original pcGGM tree. By using a CCR H' and upper bounding the probability of illegal CCR queries, we can prove the pseudorandomness of the first n correction words and the high $\lambda - 1$ bits (i.e., s_n) of the α-th leaf. The pseudorandom $s_n = \langle s_n \rangle_0 \oplus \langle s_n \rangle_1$ and $\mathsf{Convert}_{\mathbb{G}}$ ensure the pseudorandom CW_{n+1} for either corrupted party.

Our DPF scheme enables a more efficient distributed key generation protocol due to the construction of the first $n-1$ correction words. The insight is that the two parties, who share $\langle \alpha \rangle$ and $\langle \beta \rangle^{\mathsf{A}}$, can use their precomputed COT tuples to set up a secret $\langle \Delta \rangle$ with $\mathsf{lsb}(\Delta) = 1$ and share $\{\langle \overline{\alpha}_i \cdot \Delta \rangle\}_{i \in [1,n]}$ in two rounds, and use the black-box evaluation technique in [Ds17] to locally share each secret $\mathsf{H}'(\langle X_{i-1} \rangle)$. This technique relies on the invariant (iii) so that, for each $i \in [1, n]$, summing the shares of the 2^i nodes on the i-th level returns the share of the i-th level on-path node. Given the two-party shares of $\langle \overline{\alpha}_i \cdot \Delta \rangle$ and $\mathsf{H}'(\langle X_{i-1} \rangle)$, the secure computation of each CW_i only needs one round for revealing $\langle \mathsf{CW}_i \rangle$, leading to $n-1$ rounds for the first $n-1$ correction words in total. In contrast, the prior protocol [Ds17] uses (2) for each correction word, and the i-th level HCW depends on $\overline{\alpha}_i$ and should be computed level-by-level. Thus, it securely computes the first $n-1$ correction words in $2(n-1)$ rounds: for each level, one round is to share $\langle \mathsf{CW}_i \rangle$ from standard OTs, and another round is to reveal this secret.

We remark that our CW_{n+1} construction uses $\langle t_n \rangle_0 - \langle t_n \rangle_1$ to replace the $(-1)^{\langle t_n \rangle_1}$ term in the prior construction. The correctness is unaffected due to the non-zero LSB (i.e., t_n) of the α-th leaf. However, when $\mathbb{G}$ is a ring, our CW_{n+1} allows the two parties to locally share $\langle t_n \rangle_0 - \langle t_n \rangle_1$ on the ring via the black-box evaluation technique [Ds17]. Thus, the secure computation of CW_{n+1} uses only one secure multiplication of two locally shared ring operands.

DCF scheme and protocol. We further show that our spcGGM tree can be extended to realize more efficient DCF scheme and its distributed protocol. Note that comparison function $f^{<}_{\alpha,\beta}(x)$ can be written as the sum of point function $f^{\bullet}_{\alpha,-\alpha_n \cdot \beta}(x)$ and a prefix function $V_{\alpha,\beta}(x)$, which returns $\alpha_{h+1} \cdot \beta \in \mathbb{G}$ such that $\alpha_1 \dots \alpha_h = x_1 \dots x_h$ is the longest common prefix of α and x (for completeness, $\alpha_{n+1} := \alpha_n$). We have shown how to realize the DPF scheme for point function $f^{\bullet}_{\alpha,-\alpha_n \cdot \beta}(x)$ from spcGGM tree. Then, we want to compute $V_{\alpha,\beta}(x)$ by reusing the prefix information with respect to α and x when traversing the spcGGM tree to evaluate the point function. Following the GGM-style DCF scheme [BCG+21], we do this by introducing more nodes to the spcGGM tree and an additional correction procedure to ensure that the sum of the introduced nodes along the path x equals $V_{\alpha,\beta}(x)$. However, our extended spcGGM tree can use less nodes and simpler correction words to compute $V_{\alpha,\beta}(x)$.

To give more details, we first recall how [BCG+21] works. It extends a shared GGM tree by replacing its length-doubling PRG with a length-quadrupling PRG so that each secret parent spawns two more secret children in $\mathbb{F}_{2^\lambda}$. For each level $i \in [1,n]$, let $\langle v_i^0 \rangle$ and $\langle v_i^1 \rangle$ denote such two secret children of the $(i-1)$-th on-path secret parent $\langle X_{i-1} \rangle = \langle s_{i-1} \,\|\, t_{i-1} \rangle$, and the two parties correct their additive shares for $V_{\alpha,\beta}(x)$ via the public correction word VCW_i:

$$\begin{aligned}
\mathsf{V}_{i-1} &:= \textstyle\sum_{b \in \{0,1\}} (-1)^{1-b} \cdot \Big(\mathsf{Convert}_{\mathbb{G}}(\langle v_{i-1}^{\overline{\alpha}_{i-1}} \rangle_b) - \mathsf{Convert}_{\mathbb{G}}(\langle v_{i-1}^{\alpha_{i-1}} \rangle_b) \Big) \in \mathbb{G}, \\
\mathsf{VCW}_i &:= (-1)^{\langle t_{i-1} \rangle_1} \cdot \Big(\big(\mathsf{Convert}_{\mathbb{G}}(\langle v_i^{\overline{\alpha}_i} \rangle_1) - \mathsf{Convert}_{\mathbb{G}}(\langle v_i^{\overline{\alpha}_i} \rangle_0)\big) - \mathsf{V}_{i-1} + (\alpha_i - \alpha_{i-1}) \cdot \beta \Big) \in \mathbb{G},
\end{aligned}$$

where $\mathsf{V}_0 := 0 \in \mathbb{G}$ and $\alpha_0 = 0$. The DCF key per party includes its DPF key for $f^{\bullet}_{\alpha,-\alpha_n \cdot \beta}(x)$ and $\{\mathsf{VCW}_i\}_{i \in [1,n]}$. The DCF security also requires the pseudorandomness of the n VCW_i's.

In contrast, our DCF scheme shows that it is overkill to introduce two more secret children to each secret parent for the DCF security. For each $i \in [1,n]$, one additional secret child $\langle v_i \rangle = \langle v_i^0 \rangle = \langle v_i^1 \rangle$ of the secret parent $\langle X_{i-1} \rangle$ suffices, and the pseudorandomness of VCW_i relies on a random $v_i = \langle v_i \rangle_0 \oplus \langle v_i \rangle_1 \in \mathbb{F}_{2^\lambda}$ as $\mathsf{Convert}_{\mathbb{G}}$ maps random strings to pseudorandom $\mathbb{G}$ elements. We can argue the pseudorandomness of v_i based on the CCR induced by $X_{i-1} = \Delta$, if we use $v_i := \mathsf{H}'(\langle X_{i-1} \rangle_0 \oplus 2) \oplus \mathsf{H}'(\langle X_{i-1} \rangle_1 \oplus 2)$. Collecting all H' inputs for the DPF part and v_i's, we find that these inputs are as structured as those in the original pcGGM tree. The DCF security can follow from a similar hybrid argument.

Our DCF protocol is extended from our DPF protocol with the additional secure computation of $\{\mathsf{VCW}_i\}_{i \in [1,n]}$. Compared with the prior work, our DCF protocol achieves better efficiency due to not only its optimized DPF part but also the structure of each VCW_i. This structure makes the $\mathsf{Convert}_{\mathbb{G}}$ difference term independent of $\overline{\alpha}_i$. This independence allows the two parties to locally share the $\mathsf{Convert}_{\mathbb{G}}$ difference via the black-box evaluation technique [Ds17], in contrast to the

Functionality $\mathcal{F}_{\mathsf{spsVOLE}}$

Parameters: Field $\mathbb{F}$ and its extension field $\mathbb{K}$.

Initialize: Upon receiving (init) from P_0 and P_1, sample $\Delta \leftarrow \mathbb{K}$ if P_0 is honest; otherwise, receive $\Delta \in \mathbb{K}$ from the adversary. Store Δ and send it to P_0. Ignore all subsequent (init) commands.

Extend: This functionality allows polynomially many (extend) commands. Upon receiving (extend, N) from P_0 and P_1:

1. If P_0 is honest, sample $\mathbf{v} \leftarrow \mathbb{K}^N$; otherwise, receive $\mathbf{v} \in \mathbb{K}^N$ from the adversary.
2. If P_1 is honest, sample $\mathbf{u} \leftarrow \mathbb{F}^N$ with exactly one nonzero entry, and compute $\mathbf{w} := \mathbf{v} + \mathbf{u} \cdot \Delta \in \mathbb{K}^N$; otherwise, receive $(\mathbf{u}, \mathbf{w}) \in \mathbb{F}^N \times \mathbb{K}^N$ from the adversary, where $\mathbf{u}$ has at most one nonzero entry, and recompute $\mathbf{v} := \mathbf{w} - \mathbf{u} \cdot \Delta \in \mathbb{K}^N$.
3. Send $\mathbf{v}$ to P_0 and $(\mathbf{u}, \mathbf{w})$ to P_1.

Global-key queries: If P_1 is corrupted, upon receiving $(\mathsf{guess}, \Delta')$, where $\Delta' \in \mathbb{K}$, from the adversary, send $(\mathsf{success})$ to the adversary if $\Delta = \Delta'$; send (fail) to the adversary otherwise.

Figure 2: Functionality for single-point subfield VOLE.

technique plus OT-based 2PC in the prior protocol. Since there is only one more secret child for each secret parent, the local computation for sharing this difference is halved as well. We can also replace the $(-1)^{\langle t_{i-1} \rangle_1}$ term in the prior VCW_i construction by a linear term $\langle t_{i-1} \rangle_0 - \langle t_{i-1} \rangle_1$, which can be locally shared via the same black-box evaluation technique if $\mathbb{G}$ is a ring. As a result, except the 2PC for sharing $\{\langle \alpha_i \cdot \beta \rangle^{\mathsf{A}}\}_{i \in [1,n]}$, the secure computation of $\{\mathsf{VCW}_i\}_{i \in [1,n]}$ requires n secure multiplications of two shared ring elements. These secure multiplications can run in parallel with that for CW_{n+1}.

In our DCF protocol, each $\langle \alpha_i \cdot \beta \rangle^{\mathsf{A}}$ is secretly shared by carefully reusing the two precomputed COT tuples, which were used to share $\langle \overline{\alpha}_i \cdot \Delta \rangle$, to run a COT-based multiplication between the XOR shared α_i and the additively shared β on the ring. This multiplication generalizes the binary case [ALSZ13, GKWY20] for an XOR shared bit and an XOR shared string by using the well-known arithmetic XOR on the ring: $\langle \alpha_i \rangle_0 \oplus \langle \alpha_i \rangle_1 = \langle \alpha_i \rangle_0 + \langle \alpha_i \rangle_1 - 2 \cdot \langle \alpha_i \rangle_0 \cdot \langle \alpha_i \rangle_1$.

4 Subfield VOLE Extension

Our sVOLE extension follows the blueprint of [BCG+19a, SGRR19, YWL+20, WYKW21], which uses LPN to locally convert t single-point sVOLE (spsVOLE) tuples output by functionality $\mathcal{F}_{\mathsf{spsVOLE}}$ (Figure 2) into an sVOLE tuple. We focus on the efficient spsVOLE protocol that UC-realizes $\mathcal{F}_{\mathsf{spsVOLE}}$. Note that the spsVOLE protocol dominates the computation and contributes all communication in sVOLE extension.

$\mathcal{F}_{\mathsf{spsVOLE}}$ is parameterized by a field $\mathbb{F}$ and its extension $\mathbb{K}$, and covers the single-point COT functionality $\mathcal{F}_{\mathsf{spCOT}}$ if $\mathbb{F} = \mathbb{F}_2$ and $\mathbb{K} = \mathbb{F}_{2^\lambda}$. This functionality is the same as that in [WYKW21], except that $\mathcal{F}_{\mathsf{spsVOLE}}$ will not abort for an incorrect global-key query. Allowing for global-key queries has been considered in [NNOB12, HSS17] and does not weaken the effective security. In the spsVOLE protocol based on pseudorandom correlated GGM, such global-key queries can be removed.

In essence, our spsVOLE protocols work as the PCG protocol [BCG+19b, BCG+19a, BCG+20, CRR21] of spsVOLE correlation, although we do not divide the correlation generation into two explicit PCG phases. In Appendix E.1, we show how to modify one of our spsVOLE protocols to

Parameters: Tree depth $n \in \mathbb{N}$. Field $\mathbb{K}$ with $|\mathbb{K}| \geq 2^\lambda$. Hash function $\mathsf{H} : \mathbb{K} \to \mathbb{K}$.

$\mathsf{cGGM.FullEval}(\Delta, k)$: Given $(\Delta, k) \in \mathbb{K}^2$,

$X_1^0 := k \in \mathbb{K}$, $X_1^1 := \Delta - k \in \mathbb{K}$.
for $i \in [2, n]$, $j \in [0, 2^{i-1})$ **do**
  $X_i^{2j} := \mathsf{H}(X_{i-1}^j) \in \mathbb{K}$, $X_i^{2j+1} := X_{i-1}^j - X_i^{2j} \in \mathbb{K}$.
$\mathbf{v} := (X_n^0, \ldots, X_n^{2^n-1}) \in \mathbb{K}^{2^n}$.
for $i \in [1, n]$ **do** $K_i^0 := \sum_{j \in [0, 2^{i-1})} X_i^{2j} \in \mathbb{K}$.
return $(\mathbf{v}, \{K_i^0\}_{i \in [1,n]})$

$\mathsf{cGGM.PuncFullEval}(\alpha, \{K_i^{\overline{\alpha}_i}\}_{i \in [1,n]})$: Given $(\alpha, \{K_i^{\overline{\alpha}_i}\}_{i \in [1,n]}) \in \{0,1\}^n \times \mathbb{K}^n$,

$X_1^{\overline{\alpha}_1} := K_1^{\overline{\alpha}_1} \in \mathbb{K}$.
for $i \in [2, n]$ **do**
  for $j \in [0, 2^{i-1})$, $j \neq \alpha_1 \ldots \alpha_{i-1}$ **do**
    $X_i^{2j} := \mathsf{H}(X_{i-1}^j) \in \mathbb{K}$, $X_i^{2j+1} := X_{i-1}^j - X_i^{2j} \in \mathbb{K}$.
  $X_i^{\alpha_1 \ldots \alpha_{i-1} \overline{\alpha}_i} := K_i^{\overline{\alpha}_i} - \sum_{j \in [0, 2^{i-1}), j \neq \alpha_1 \ldots \alpha_{i-1}} X_i^{2j + \overline{\alpha}_i} \in \mathbb{K}$.
$X_n^\alpha := -\sum_{j \in [0, 2^n), j \neq \alpha} X_n^j \in \mathbb{K}$, $\mathbf{w} := (X_n^0, \ldots, X_n^{2^n-1}) \in \mathbb{K}^{2^n}$.
return $\mathbf{w}$

Figure 3: Two full-evaluation algorithms for correlated GGM tree.

define such two phases, in order to satisfy the "silent property" that a long spsVOLE tuple can be stored as two sublinearly short correlated seeds.

4.1 Single-point COT and sVOLE from Correlated GGM

In Figure 3, we present the two evaluation algorithms for our correlated GGM tree, which is defined by two first-level nodes $(k, \Delta - k) \in \mathbb{K}^2$. For every non-leaf node $x \in \mathbb{K}$, its left child is defined as $\mathsf{H}(x) \in \mathbb{K}$ while its right child is defined as $x - \mathsf{H}(x) \in \mathbb{K}$. The following claim is straightforward from an induction.

Claim 1 (Leveled correlation)**.** *For any two first-level nodes $(k, \Delta - k) \in \mathbb{K}^2$ and any $i \in [1, n]$, the offset $\Delta \in \mathbb{K}$ equals the sum of all nodes on the i-th level of the correlated GGM tree expanded from $(k, \Delta - k)$ as per $\mathsf{cGGM.FullEval}$.*

Corollary 1. *For any $\alpha \in [0, 2^n)$, any $(k, \Delta - k) \in \mathbb{K}^2$, and*

$$(\mathbf{v}, \{K_i^0\}_{i \in [1,n]}) := \mathsf{cGGM.FullEval}(\Delta, k),$$
$$\mathbf{w} := \mathsf{cGGM.PuncFullEval}(\alpha, \{K_i^{\overline{\alpha}_i}\}_{i \in [1,n]}),$$

where $K_i^{\overline{\alpha}_i} := \overline{\alpha}_i \cdot \Delta + (-1)^{\overline{\alpha}_i} \cdot K_i^0$ for $i \in [1, n]$, we have $\mathbf{w}^{(\alpha)} - \mathbf{v}^{(\alpha)} = -\Delta$.

Proof. Claim 1 and the definition of $\mathsf{cGGM.FullEval}$ imply that $K_i^{\overline{\alpha}_i} \in \mathbb{K}$ in this corollary defines the sum of all $\overline{\alpha}_i$-side nodes on the i-th level of the correlated GGM tree. Then, it follows from the definition of $\mathsf{cGGM.PuncFullEval}$ that $\mathbf{v}^{(j)} = \mathbf{w}^{(j)}$ for any $j \neq \alpha \in [0, 2^n)$. Using Claim 1 for the last level, we have $\mathbf{w}^{(\alpha)} - \mathbf{v}^{(\alpha)} = -\sum_{j \in [0,2^n), j \neq \alpha} \mathbf{w}^{(j)} - \mathbf{v}^{(\alpha)} = -\sum_{j \in [0,2^n), j \neq \alpha} \mathbf{v}^{(j)} - \mathbf{v}^{(\alpha)} = -\Delta$. □

4.1.1 Single-point COT

Figure 4 describes our single-point COT protocol Π_{spCOT} that runs in the $\mathcal{F}_{\mathsf{COT}}$-hybrid model and uses the cGGM expansion in Figure 3.

Protocol Π_{spCOT}

Parameters: Field $\mathbb{F}_2$ and its extension field $\mathbb{F}_{2^\lambda}$.

Initialize: This procedure is executed only once.

1. P_0 and P_1 send (init) to $\mathcal{F}_{\mathsf{COT}}$, which returns $\Delta \in \mathbb{F}_{2^\lambda}$ to P_0. P_0 outputs Δ.

Extend: This procedure can be executed many times. P_0 and P_1 input $N = 2^n$ and use cGGM (c.f. Figure 3) for n and $\mathbb{F}_{2^\lambda}$.

2. P_0 and P_1 send (extend, n) to $\mathcal{F}_{\mathsf{COT}}$, which returns $(\mathsf{K}[r_1], \ldots, \mathsf{K}[r_n]) \in \mathbb{F}_{2^\lambda}^n$ to P_0 and $((r_1, \ldots, r_n), (\mathsf{M}[r_1], \ldots, \mathsf{M}[r_n])) \in \mathbb{F}_2^n \times \mathbb{F}_{2^\lambda}^n$ to P_1 such that $\mathsf{M}[r_i] = \mathsf{K}[r_i] \oplus r_i \cdot \Delta$ for $i \in [1, n]$.
3. P_0 samples $c_1 \leftarrow \mathbb{F}_{2^\lambda}$ and sets $k := \mathsf{K}[r_1] \oplus c_1$,
$$(\mathbf{v}, \{K_i^0\}_{i \in [1,n]}) := \mathsf{cGGM.FullEval}(\Delta, k),$$
and $c_i := \mathsf{K}[r_i] \oplus K_i^0$ for $i \in [2, n]$. P_0 sends $(c_1, \ldots, c_n)$ to P_1.
4. P_1 sets $\alpha = \alpha_1 \ldots \alpha_n := \overline{r}_1 \ldots \overline{r}_n \in [0, N)$, $K_i^{\overline{\alpha}_i} := \mathsf{M}[r_i] \oplus c_i$ for $i \in [1, n]$, and
$$\mathbf{u} := \mathbf{unit}_{\mathbb{F}_2}(N, \alpha, 1), \quad \mathbf{w} := \mathsf{cGGM.PuncFullEval}(\alpha, \{K_i^{\overline{\alpha}_i}\}_{i \in [1,n]}).$$
5. P_0 outputs $\mathbf{v}$ and P_1 outputs $(\mathbf{u}, \mathbf{w})$.

Figure 4: cGGM-based single-point COT protocol in the $\mathcal{F}_{\mathsf{COT}}$-hybrid model.

The same Δ in correlated GGM trees. Note that $\mathcal{F}_{\mathsf{spCOT}}$ produces single-point COT tuples with the same global key $\Delta \in \mathbb{F}_{2^\lambda}$ in a number of **Extend** executions. To realize $\mathcal{F}_{\mathsf{spCOT}}$, our protocol Π_{spCOT} proceeds as sketched in Section 3.1 but uses the same Δ for the cGGM trees of these executions, each of which samples a fresh $k \leftarrow \mathbb{F}_{2^\lambda}$ for $\mathsf{cGGM.FullEval}(\Delta, k)$. A merit of using the same Δ in several tree instances is that Π_{spCOT} only invokes one $\mathcal{F}_{\mathsf{COT}}$ instance, and the amortized cost per precomputed COT tuple can be small.

Security. We prove Theorem 1 by following the sketched intuition in Section 3.1 and defer the proof to Appendix B.1. Our proof considers polynomially many concurrent **Extend** executions (strictly speaking, sub-sessions with unique sub-session IDs) that uses the one-time initialized Δ.

Theorem 1. *Given random permutation $\pi : \mathbb{F}_{2^\lambda} \to \mathbb{F}_{2^\lambda}$, efficiently computable linear orthomorphism $\sigma : \mathbb{F}_{2^\lambda} \to \mathbb{F}_{2^\lambda}$ with efficiently computable σ^{-1}, $\sigma'(x) := \sigma(x) \oplus x$, and σ'^{-1} (Footnote 1), and hash function $\mathsf{H}(x) := \pi(\sigma(x)) \oplus \sigma(x)$, protocol Π_{spCOT} (Figure 4) UC-realizes functionality $\mathcal{F}_{\mathsf{spCOT}}$ (Figure 2) against any semi-honest adversary in the $\mathcal{F}_{\mathsf{COT}}$-hybrid model and the RPM.*

Communication optimization. For t concurrent **Extend** executions (e.g., in COT extension), the random c_1's in these executions can be compressed via a PRF $F : \mathbb{F}_{2^\lambda} \times \{0,1\}^* \to \mathbb{F}_{2^\lambda}$. Concretely, P_0 samples a PRF key $k_{\mathsf{prf}} \leftarrow \mathbb{F}_{2^\lambda}$ *after receiving its COT outputs in all executions* and sends this key to P_1. For each execution with sub-session ID ssid, the two parties locally defines the element $c_1 := F(k_{\mathsf{prf}}, \mathsf{ssid})$. This PRF key is only used for the t concurrent executions. The security of this optimization follows from the PRF security and the fact that, in the concurrent executions, the COT messages chosen by the corrupted receiver cannot depend on the PRF key to be sampled by the honest sender.

Complexity analysis. Consider the complexity per execution when the PRF-based optimization is used in t concurrent **Extend** executions. Π_{spCOT} needs n precomputed COT tuples. P_0 sends $(n-1) \cdot \lambda + \frac{\lambda}{t}$ bits, and P_1 sends nothing. The computation per party comes from the tree expansion with N RP calls.

In the $\mathcal{F}_{\mathsf{COT}}$-hybrid model, the prior single-point COT protocol [YWL+20] consumes n precomputed COT tuples. However, P_0 sends $2n \cdot \lambda$ bits. Each party performs about N length-doubling PRG calls, which in turn result in $2N$ RP calls. We can see that our protocol halves both the computation and communication in the prior work. When looking at the whole protocol, the improvement is still huge. For example, the micro benchmark in Silver [CRR21] reported that 70% of the time is spent on GGM-tree-related computation, and thus our protocol will lead to more than 30% of end-to-end computational improvement in COT.

4.1.2 Single-point sVOLE

We can also realize single-point sVOLE from our cGGM tree by using the high-level idea sketched in Section 3.1. This protocol extends Π_{spCOT} by using a cGGM tree whose nodes are in a general *exponentially large* extension field $\mathbb{K}$. The tree expansion therein uses a hash function constructed from a random permutation and a linear orthomorphism over $\mathbb{K}$. We defer the detailed protocol and its security proof to Appendix B.2.

4.2 Single-point sVOLE from Pseudorandom Correlated GGM

We can adapt our correlated GGM tree for a *pseudorandom* correlated one with the property that the leaf node at some punctured position α is pseudorandom. This pseudorandom correlated GGM tree pcGGM is defined in Figure 5, where the first $n-1$ levels preserve the correlation in Claim 1 but all last-level nodes are processed by H_S to break this correlation. The keyed hash function H_S uses some key $S \in \mathbb{F}_{2^\lambda}$, which can be sampled by the receiver in single-point sVOLE and, for simplicity, is assumed to have been sent to the sender before protocol execution. The implementation of H_S is given in Theorem 2. In fact, this pcGGM tree yields PPRF, which is proved in Appendix C.

The pseudorandomness only at the cost of the last-level correlation allows us to follow the single-point sVOLE blueprint in [BCG+19a, WYKW21] but also take advantage of the correlation in the first $n-1$ levels. The protocol is presented in Figure 6. In this protocol, the sender P_0 only sends λ bits to the receiver P_1 for each of the first $n-1$ levels, given a precomputed COT tuple. For the last level, the two parties use a COT tuple and the standard technique [IKNP03, Bea95] to emulate the string OT as in the prior protocols. To amortize the cost per precomputed COT tuple, the pcGGM trees in many **Extend** executions also use the same Δ set by $\mathcal{F}_{\mathsf{COT}}$.

Security. The security against the semi-honest P_0 resorts to the one-time pad s from $\mathcal{F}_{\mathsf{sVOLE}}$. Meanwhile, the security against the semi-honest P_1 relies on that (i) the pcGGM tree with a CCR structure has n pseudorandom off-path nodes and the punctured leaf, giving pseudorandom $c_1, \ldots, c_{n-1}$ and $(c_n^{r_n}, \psi)$, and (ii) the mask of the unselected message $c_n^{\bar{r}_n}$ in the emulated last-level OT is computed by applying $\mathsf{Convert}_{\mathbb{K}}$ to a CCR response, which is for a legal CCR query with overwhelming probability due to the uniform μ. The proof of Theorem 2 can be found in Appendix B.3, where we consider polynomially many concurrent **Extend** executions, which use the one-time initialized Δ.

Theorem 2. *Given CCR function* $\mathsf{H} : \mathbb{F}_{2^\lambda} \to \mathbb{F}_{2^\lambda}$, *function* $\mathsf{Convert}_{\mathbb{K}} : \mathbb{F}_{2^\lambda} \to \mathbb{K}$, *and keyed hash function* $\mathsf{H}_S(x) := \mathsf{H}(S \oplus x)$ *with some key* $S \leftarrow \mathbb{F}_{2^\lambda}$, *protocol* $\Pi_{\mathsf{spsVOLE-pcGGM}}$ *(Figure 6) UC-realizes functionality* $\mathcal{F}_{\mathsf{spsVOLE}}$ *(Figure 2) without global-key queries against any semi-honest adversary in the* $(\mathcal{F}_{\mathsf{COT}}, \mathcal{F}_{\mathsf{sVOLE}})$*-hybrid model.*

Communication optimization. $\Pi_{\mathsf{spsVOLE-pcGGM}}$ can be optimized as follows:

Parameters: Tree depth $n \in \mathbb{N}$. Field $\mathbb{K}$. Keyed hash function $\mathsf{H}_S : \mathbb{F}_{2^\lambda} \to \mathbb{F}_{2^\lambda}$. Function $\mathsf{Convert}_{\mathbb{K}} : \mathbb{F}_{2^\lambda} \to \mathbb{K}$.

$\mathsf{pcGGM.FullEval}(\Delta, k)$: Given $(\Delta, k) \in \mathbb{F}_{2^\lambda}^2$,

$X_1^0 := k \in \mathbb{F}_{2^\lambda}$, $X_1^1 := \Delta \oplus k \in \mathbb{F}_{2^\lambda}$.
for $i \in [2, n-1]$, $j \in [0, 2^{i-1})$ **do**
  $X_i^{2j} := \mathsf{H}_S(X_{i-1}^j) \in \mathbb{F}_{2^\lambda}$, $X_i^{2j+1} := X_{i-1}^j \oplus X_i^{2j} \in \mathbb{F}_{2^\lambda}$.
for $j \in [0, 2^{n-1})$, $b \in \{0,1\}$ **do** $X_n^{2j+b} := \mathsf{Convert}_{\mathbb{K}}(\mathsf{H}_S(X_{n-1}^j \oplus b)) \in \mathbb{K}$.
$\mathbf{v} := (X_n^0, \ldots, X_n^{2^n-1}) \in \mathbb{K}^{2^n}$.
for $i \in [1, n-1]$ **do** $K_i^0 := \oplus_{j \in [0, 2^{i-1})} X_i^{2j} \in \mathbb{F}_{2^\lambda}$.
$(K_n^0, K_n^1) := (\sum_{j \in [0, 2^{n-1})} X_n^{2j}, \sum_{j \in [0, 2^{n-1})} X_n^{2j+1}) \in \mathbb{K}^2$.
return $(\mathbf{v}, \{K_i^0\}_{i \in [1, n-1]}, (K_n^0, K_n^1))$

$\mathsf{pcGGM.PuncFullEval}(\alpha, \{K_i^{\overline{\alpha}_i}\}_{i \in [1,n]}, \gamma)$: Given $(\alpha, \{K_i^{\overline{\alpha}_i}\}_i, \gamma) \in \{0,1\}^n \times \mathbb{K}^n \times \mathbb{K}$,

$X_1^{\overline{\alpha}_1} := K_1^{\overline{\alpha}_1} \in \mathbb{F}_{2^\lambda}$.
for $i \in [2, n-1]$ **do**
  for $j \in [0, 2^{i-1})$, $j \neq \alpha_1 \ldots \alpha_{i-1}$ **do**
    $X_i^{2j} := \mathsf{H}_S(X_{i-1}^j) \in \mathbb{F}_{2^\lambda}$, $X_i^{2j+1} := X_{i-1}^j \oplus X_i^{2j} \in \mathbb{F}_{2^\lambda}$.
  $X_i^{\alpha_1 \ldots \alpha_{i-1} \overline{\alpha}_i} := K_i^{\overline{\alpha}_i} \oplus (\oplus_{j \in [0, 2^{i-1}), j \neq \alpha_1 \ldots \alpha_{i-1}} X_i^{2j + \overline{\alpha}_i}) \in \mathbb{F}_{2^\lambda}$.
for $j \in [0, 2^{n-1})$, $j \neq \alpha_1 \ldots \alpha_{n-1}$, $b \in \{0,1\}$ **do**
  $X_n^{2j+b} := \mathsf{Convert}_{\mathbb{K}}(\mathsf{H}_S(X_{n-1}^j \oplus b)) \in \mathbb{K}$.
$X_n^{\alpha_1 \ldots \alpha_{n-1} \overline{\alpha}_n} := K_n^{\overline{\alpha}_n} - \sum_{j \in [0, 2^{n-1}), j \neq \alpha_1 \ldots \alpha_{n-1}} X_n^{2j + \overline{\alpha}_n} \in \mathbb{K}$.
$X_n^\alpha := \gamma - \sum_{j \in [0, 2^n), j \neq \alpha} X_n^j \in \mathbb{K}$, $\mathbf{w} := (X_n^0, \ldots, X_n^{2^n-1}) \in \mathbb{K}^{2^n}$.
return $\mathbf{w}$

Figure 5: Two full-evaluation algorithms for pseudorandom correlated GGM tree.

- The two random (c_1, μ) to be sent by the sender in $\Pi_{\mathsf{spsVOLE-pcGGM}}$ can be compressed via the PRF technique for Π_{spCOT}. In t concurrent **Extend** executions, all such random messages can also be compressed in batch.

- The optimization for a large field $\mathbb{F}$ in $\Pi_{\mathsf{spsVOLE-cGGM}}$ also applies.

- If $\mathbb{F} = \mathbb{F}_2$, $\Pi_{\mathsf{spsVOLE-pcGGM}}$ degenerates to single-point COT and can do away with $\mathcal{F}_{\mathsf{sVOLE}}$ so that the receiver need not send a difference $d \in \mathbb{F}$. Instead, the sender locally samples $\Gamma \in \mathbb{K}$ and masks this value with the sum of all last-level nodes in a pcGGM tree. This optimization has been used in [BCG+19a].

Complexity analysis. Consider the complexity per execution when the PRF-based optimization is used in t concurrent **Extend** executions. $\Pi_{\mathsf{spsVOLE-pcGGM}}$ uses n precomputed COT tuples and one precomputed sVOLE tuple. P_0 sends $(n-2) \cdot \lambda + 3 \cdot \log|\mathbb{K}| + \frac{\lambda}{t}$ bits, and P_1 sends $\log|\mathbb{F}|$ bits. The computation is dominated by the tree expansion with $1.5N$ RP calls for each party. Compared with the prior works [BCG+19a, WYKW21], our protocol roughly halve the communication, and the reduction in computation is 25%. This computation cost includes no PRG call in $\mathsf{Convert}_{\mathbb{K}}$, which can be implemented from cheap modulo operations for the field size $|\mathbb{K}|$ considered in many sVOLE applications, e.g., [WYKW21, YSWW21, RS21, WYX+21].

Protocol $\Pi_{\mathsf{spsVOLE-pcGGM}}$

Parameters: Field $\mathbb{F}$ and its extension field $\mathbb{K}$.

Initialize: This procedure is executed only once.

1. P_0 and P_1 send (init) to $\mathcal{F}_{\mathsf{COT}}$, which returns $\Delta \in \mathbb{F}_{2^\lambda}$ to P_0.
2. P_0 and P_1 send (init) to $\mathcal{F}_{\mathsf{sVOLE}}$, which returns $\Gamma \in \mathbb{K}$ to P_0. P_0 outputs Γ.

Extend: This procedure can be executed many times. P_0 and P_1 input $N = 2^n$ and use pcGGM (c.f. Figure 5) for n, $\mathbb{K}$, keyed hash function $\mathsf{H}_S : \mathbb{F}_{2^\lambda} \to \mathbb{F}_{2^\lambda}$, and function $\mathsf{Convert}_{\mathbb{K}} : \mathbb{F}_{2^\lambda} \to \mathbb{K}$.

3. P_0 and P_1 send (extend, n) to $\mathcal{F}_{\mathsf{COT}}$, which returns $(\mathsf{K}[r_1], \ldots, \mathsf{K}[r_n]) \in \mathbb{F}_{2^\lambda}^n$ to P_0 and $((r_1, \ldots, r_n), (\mathsf{M}[r_1], \ldots, \mathsf{M}[r_n])) \in \mathbb{F}_2^n \times \mathbb{F}_{2^\lambda}^n$ to P_1 such that $\mathsf{M}[r_i] = \mathsf{K}[r_i] \oplus r_i \cdot \Delta$ for $i \in [1, n]$.
4. P_0 and P_1 send $(\mathsf{extend}, 1)$ to $\mathcal{F}_{\mathsf{sVOLE}}$, which returns $\mathsf{K}[s] \in \mathbb{K}$ to P_0 and $(s, \mathsf{M}[s]) \in \mathbb{F} \times \mathbb{K}$ to P_1 such that $\mathsf{M}[s] = \mathsf{K}[s] + s \cdot \Gamma$.
5. P_1 samples $\beta \leftarrow \mathbb{F}^*$, sets $\mathsf{M}[\beta] := \mathsf{M}[s]$, and sends $d := s - \beta \in \mathbb{F}$ to P_0.
   P_0 sets $\mathsf{K}[\beta] := \mathsf{K}[s] + d \cdot \Gamma$ such that $\mathsf{M}[\beta] = \mathsf{K}[\beta] + \beta \cdot \Gamma$.
6. P_0 samples $(c_1, \mu) \leftarrow \mathbb{F}_{2^\lambda}^2$ and sets $k := \mathsf{K}[r_1] \oplus c_1$,
   $$(\mathbf{v}, \{K_i^0\}_{i \in [1, n-1]}, (K_n^0, K_n^1)) := \mathsf{pcGGM.FullEval}(\Delta, k),$$
   $c_i := \mathsf{K}[r_i] \oplus K_i^0$ for $i \in [2, n-1]$, $c_n^b := \mathsf{Convert}_{\mathbb{K}}(\mathsf{H}_S(\mu \oplus \mathsf{K}[r_n] \oplus b \cdot \Delta)) + K_n^b$ for $b \in \{0, 1\}$, and $\psi := K_n^0 + K_n^1 - \mathsf{K}[\beta]$.
   P_0 sends $(c_1, \ldots, c_{n-1}, \mu, c_n^0, c_n^1, \psi)$ to P_1.
7. P_1 sets $\alpha = \alpha_1 \ldots \alpha_n := \overline{r}_1 \ldots \overline{r}_n \in [0, N)$, $K_i^{\overline{\alpha}_i} := \mathsf{M}[r_i] \oplus c_i$ for $i \in [1, n-1]$, $K_n^{\overline{\alpha}_n} := c_n^{r_n} - \mathsf{Convert}_{\mathbb{K}}(\mathsf{H}_S(\mu \oplus \mathsf{M}[r_n]))$, and
   $$\mathbf{u} := \mathbf{unit}_{\mathbb{F}}(N, \alpha, \beta), \quad \mathbf{w} := \mathsf{pcGGM.PuncFullEval}(\alpha, \{K_i^{\overline{\alpha}_i}\}_{i \in [1, n]}, \psi + \mathsf{M}[\beta]).$$
8. P_0 outputs $\mathbf{v}$ and P_1 outputs $(\mathbf{u}, \mathbf{w})$.

Figure 6: pcGGM-based single-point sVOLE protocol in the $(\mathcal{F}_{\mathsf{COT}}, \mathcal{F}_{\mathsf{sVOLE}})$-hybrid model.

Functionality $\mathcal{F}_{\mathsf{FSS}}$

Parameters: Ring $\mathcal{R}$. $\mathsf{FSS} \in \{\mathsf{DPF}, \mathsf{DCF}\}$ with domain $[0, N)$, where domain size $N = 2^n$ for $n \in \mathbb{N}$, and range $\mathcal{R}$.

Gen: This functionality allows polynomially many (gen) commands. Upon receiving $(\mathsf{gen}, \langle\alpha\rangle_b, \langle\beta\rangle_b^{\mathsf{A}})$ from P_b for each $b \in \{0, 1\}$, where $(\langle\alpha\rangle_b, \langle\beta\rangle_b^{\mathsf{A}}) \in [0, N) \times \mathcal{R}$:

1. Set $\alpha := \langle\alpha\rangle_0 \oplus \langle\alpha\rangle_1 \in [0, N)$, $\beta := \langle\beta\rangle_0^{\mathsf{A}} + \langle\beta\rangle_1^{\mathsf{A}} \in \mathcal{R}$, and $\mathbf{r} \in \mathcal{R}^N$ such that
   - If $\mathsf{FSS} = \mathsf{DPF}$, $\mathbf{r}^{(j)} = 0$ for $j \in [0, N)$, $j \neq \alpha$, and $\mathbf{r}^{(\alpha)} = \beta$.
   - If $\mathsf{FSS} = \mathsf{DCF}$, $\mathbf{r}^{(j)} = 0$ for $j \in [0, N)$, $j \geq \alpha$, and $\mathbf{r}^{(j)} = \beta$ otherwise.
2. If both parties are honest, sample $\langle\mathbf{r}\rangle_0^{\mathsf{A}}, \langle\mathbf{r}\rangle_1^{\mathsf{A}} \leftarrow \mathcal{R}^N$ such that $\langle\mathbf{r}\rangle_0^{\mathsf{A}} + \langle\mathbf{r}\rangle_1^{\mathsf{A}} = \mathbf{r}$; otherwise (i.e., P_b is corrupted), receive $\langle\mathbf{r}\rangle_b^{\mathsf{A}} \in \mathcal{R}^N$ from the adversary and recompute $\langle\mathbf{r}\rangle_{1-b}^{\mathsf{A}} := \mathbf{r} - \langle\mathbf{r}\rangle_b^{\mathsf{A}} \in \mathcal{R}^N$.
3. Send $\langle\mathbf{r}\rangle_0^{\mathsf{A}}$ to P_0 and $\langle\mathbf{r}\rangle_1^{\mathsf{A}}$ to P_1.

Figure 7: Functionality for DPF/DCF correlation generation.

Parameters: Domain size $N = 2^n$ for $n \in \mathbb{N}$. Group $\mathbb{G}$. Keyed hash function $\mathsf{H}_S : \mathbb{F}_{2^\lambda} \to \mathbb{F}_{2^\lambda}$. Function $\mathsf{Convert}_{\mathbb{G}} : \{0,1\}^* \to \mathbb{G}$.

$\mathsf{DPF.Gen}(1^\lambda, (\alpha, \beta, n, \mathbb{G}))$:

Parse $\alpha = \alpha_1 \dots \alpha_n \in \{0,1\}^n$ and $\beta \in \mathbb{G}$.
Sample $\Delta \leftarrow \{0,1\}^\lambda$ such that $\mathsf{lsb}(\Delta) = 1$.
Sample $\langle s_0 \,\|\, t_0 \rangle_0, \langle s_0 \,\|\, t_0 \rangle_1 \leftarrow \{0,1\}^\lambda$ such that $\langle s_0 \,\|\, t_0 \rangle_0 \oplus \langle s_0 \,\|\, t_0 \rangle_1 = \Delta$.
for $i \in [1, n-1]$ **do**
    $\mathsf{CW}_i := \mathsf{H}_S(\langle s_{i-1} \,\|\, t_{i-1} \rangle_0) \oplus \mathsf{H}_S(\langle s_{i-1} \,\|\, t_{i-1} \rangle_1) \oplus \overline{\alpha}_i \cdot \Delta$
    $\langle s_i \,\|\, t_i \rangle_0 := \mathsf{H}_S(\langle s_{i-1} \,\|\, t_{i-1} \rangle_0) \oplus \alpha_i \cdot \langle s_{i-1} \,\|\, t_{i-1} \rangle_0 \oplus \langle t_{i-1} \rangle_0 \cdot \mathsf{CW}_i$
    $\langle s_i \,\|\, t_i \rangle_1 := \mathsf{H}_S(\langle s_{i-1} \,\|\, t_{i-1} \rangle_1) \oplus \alpha_i \cdot \langle s_{i-1} \,\|\, t_{i-1} \rangle_1 \oplus \langle t_{i-1} \rangle_1 \cdot \mathsf{CW}_i$
$\langle \mathsf{high}^\sigma \,\|\, \mathsf{low}^\sigma \rangle_0 := \mathsf{H}_S(\langle s_{n-1} \,\|\, t_{n-1} \rangle_0 \oplus \sigma)$ for $\sigma \in \{0,1\}$
$\langle \mathsf{high}^\sigma \,\|\, \mathsf{low}^\sigma \rangle_1 := \mathsf{H}_S(\langle s_{n-1} \,\|\, t_{n-1} \rangle_1 \oplus \sigma)$ for $\sigma \in \{0,1\}$
$\mathsf{HCW} := \langle \mathsf{high}^{\overline{\alpha}_n} \rangle_0 \oplus \langle \mathsf{high}^{\overline{\alpha}_n} \rangle_1$
$\mathsf{LCW}^0 := \langle \mathsf{low}^0 \rangle_0 \oplus \langle \mathsf{low}^0 \rangle_1 \oplus \overline{\alpha}_n, \quad \mathsf{LCW}^1 := \langle \mathsf{low}^1 \rangle_0 \oplus \langle \mathsf{low}^1 \rangle_1 \oplus \alpha_n$
$\mathsf{CW}_n := (\mathsf{HCW} \,\|\, \mathsf{LCW}^0 \,\|\, \mathsf{LCW}^1)$
$\langle s_n \,\|\, t_n \rangle_0 := \langle \mathsf{high}^{\alpha_n} \,\|\, \mathsf{low}^{\alpha_n} \rangle_0 \oplus \langle t_{n-1} \rangle_0 \cdot (\mathsf{HCW} \,\|\, \mathsf{LCW}^{\alpha_n})$
$\langle s_n \,\|\, t_n \rangle_1 := \langle \mathsf{high}^{\alpha_n} \,\|\, \mathsf{low}^{\alpha_n} \rangle_1 \oplus \langle t_{n-1} \rangle_1 \cdot (\mathsf{HCW} \,\|\, \mathsf{LCW}^{\alpha_n})$
$\mathsf{CW}_{n+1} := (\langle t_n \rangle_0 - \langle t_n \rangle_1) \cdot (\mathsf{Convert}_{\mathbb{G}}(\langle s_n \rangle_1) - \mathsf{Convert}_{\mathbb{G}}(\langle s_n \rangle_0) + \beta)$
$k_b := (\langle s_0 \,\|\, t_0 \rangle_b, \{\mathsf{CW}_i\}_{i \in [1, n+1]})$ for $b \in \{0,1\}$
return (k_0, k_1)

$\mathsf{DPF.Eval}(b, k_b, x)$:

Parse $k_b = (\langle s_0^0 \,\|\, t_0^0 \rangle_b, \{\mathsf{CW}_i\}_{i \in [1, n+1]})$, $\mathsf{CW}_n = (\mathsf{HCW} \,\|\, \mathsf{LCW}^0 \,\|\, \mathsf{LCW}^1)$, and $x = x_1 \dots x_n \in \{0,1\}^n$.
for $i \in [1, n-1]$ **do**
    $\langle s_i^{x_1 \dots x_i} \,\|\, t_i^{x_1 \dots x_i} \rangle_b := \mathsf{H}_S(\langle s_{i-1}^{x_1 \dots x_{i-1}} \,\|\, t_{i-1}^{x_1 \dots x_{i-1}} \rangle_b) \oplus x_i \cdot \langle s_{i-1}^{x_1 \dots x_{i-1}} \,\|\, t_{i-1}^{x_1 \dots x_{i-1}} \rangle_b \oplus \langle t_{i-1}^{x_1 \dots x_{i-1}} \rangle_b \cdot \mathsf{CW}_i$
$\langle \mathsf{high} \,\|\, \mathsf{low} \rangle_b := \mathsf{H}_S(\langle s_{n-1}^{x_1 \dots x_{n-1}} \,\|\, t_{n-1}^{x_1 \dots x_{n-1}} \rangle_b \oplus x_n)$
$\langle s_n^x \,\|\, t_n^x \rangle_b := \langle \mathsf{high} \,\|\, \mathsf{low} \rangle_b \oplus \langle t_{n-1}^{x_1 \dots x_{n-1}} \rangle_b \cdot (\mathsf{HCW} \,\|\, \mathsf{LCW}^{x_n})$
return $y_b := (-1)^b \cdot (\mathsf{Convert}_{\mathbb{G}}(\langle s_n^x \rangle_b) + \langle t_n^x \rangle_b \cdot \mathsf{CW}_{n+1})$

Figure 8: Our DPF scheme with domain $[0, N)$ and range $\mathbb{G}$.

5 DPF and DCF Correlation Generation

We model DPF/DCF correlation generation in functionality $\mathcal{F}_{\mathsf{FSS}}$ (Figure 7), which includes distributed key generation and local full-domain evaluation. By putting both procedures in the same functionality, we are able to model FSS as an ideal functionality and avoid caveats in the proof. $\mathcal{F}_{\mathsf{FSS}}$ focuses on $N = 2^n$ for $n \in \mathbb{N}$, and we can define a similar functionality for a general $N \in \mathbb{N}$. Using padding, our protocols for $\mathcal{F}_{\mathsf{FSS}}$ also works in this general case.

One can view $\mathcal{F}_{\mathsf{FSS}}$ as an alternative to the FSS key generation functionality that outputs each FSS key in the key pair to the designated party, who locally uses its key to evaluate its shares of the evaluation results at several points. We note that the full-domain evaluation included in $\mathcal{F}_{\mathsf{FSS}}$ does not complicate its implementation in contrast to the known protocols [Ds17, BCG+21] of the FSS key generation functionality. The reason is that, using the black-box evaluation technique [Ds17], these protocols also perform full-domain evaluation. If FSS correlations are generated for immediate use without long-term storage (e.g., [Ds17]), $\mathcal{F}_{\mathsf{FSS}}$ can be a drop-in replacement of the FSS key generation functionality. However, we also show in Appendix E.2 that our protocols for $\mathcal{F}_{\mathsf{FSS}}$ can be adapted to realize this key generation functionality.

Parameters: Domain size $N = 2^n$ for $n \in \mathbb{N}$. Group $\mathbb{G}$. Keyed hash function $\mathsf{H}_S : \mathbb{F}_{2^\lambda} \to \mathbb{F}_{2^\lambda}$. Function $\mathsf{Convert}_{\mathbb{G}} : \{0,1\}^* \to \mathbb{G}$.

$\mathsf{DCF.Gen}(1^\lambda, (\alpha, \beta, n, \mathbb{G}))$:

Parse $\alpha = \alpha_1 \ldots \alpha_n \in \{0,1\}^n$ and $\beta \in \mathbb{G}$. Let $\alpha_0 := 0$.
Run $(k'_0, k'_1) \leftarrow \mathsf{DPF.Gen}(1^\lambda, (\alpha, -\alpha_n \cdot \beta, n, \mathbb{G}))$ and store its internal variables.
for $i \in [1, n]$ **do**
    $\langle v_i \rangle_0 := \mathsf{H}_S(\langle s_{i-1} \,\|\, t_{i-1} \rangle_0 \oplus 2)$
    $\langle v_i \rangle_1 := \mathsf{H}_S(\langle s_{i-1} \,\|\, t_{i-1} \rangle_1 \oplus 2)$
    $\mathsf{VCW}_i := (\langle t_{i-1} \rangle_0 - \langle t_{i-1} \rangle_1) \cdot (\mathsf{Convert}_{\mathbb{G}}(\langle v_i \rangle_1) - \mathsf{Convert}_{\mathbb{G}}(\langle v_i \rangle_0) + (\alpha_i - \alpha_{i-1}) \cdot \beta)$
$k_b := (k'_b, \{\mathsf{VCW}_i\}_{i \in [1,n]})$ for $b \in \{0,1\}$
return (k_0, k_1)

$\mathsf{DCF.Eval}(b, k_b, x)$:

Parse $k_b = (k'_b, \{\mathsf{VCW}_i\}_{i \in [1,n]})$. Let $V_b^0 := 0 \in \mathbb{G}$.
Run $y'_b := \mathsf{DPF.Eval}(b, k'_b, x)$ and store its internal variables.
for $i \in [1, n]$ **do**
    $\langle v_i^{x_1 \ldots x_{i-1}} \rangle_b := \mathsf{H}_S(\langle s_{i-1}^{x_1 \ldots x_{i-1}} \,\|\, t_{i-1}^{x_1 \ldots x_{i-1}} \rangle_b \oplus 2)$
    $V_b^i := V_b^{i-1} + (-1)^b \cdot (\mathsf{Convert}_{\mathbb{G}}(\langle v_i^{x_1 \ldots x_{i-1}} \rangle_b) + \langle t_{i-1}^{x_1 \ldots x_{i-1}} \rangle_b \cdot \mathsf{VCW}_i)$
return $y_b := y'_b + V_b^n$

Figure 9: Our DCF scheme with domain $[0, N)$ and range $\mathbb{G}$.

5.1 DPF and DCF Schemes

Note that DPF/DCF scheme may be used in not only distributed settings (e.g., [Ds17]) but also the scenarios where a trusted dealer is available (e.g., two-server PIR [GI14, BGI16]). It would be better for us to present the two schemes alone.

We present in Figure 8 (resp., Figure 9) our DPF (resp., DCF) scheme, which is implicitly constructed from a *shared* pseudorandom correlated GGM tree. For simplicity of exposition, we slightly abuse the function $\mathsf{Convert}_{\mathbb{G}} : \{0,1\}^* \to \mathbb{G}$ so that it can map random strings of either λ or $\lambda - 1$ bits to pseudorandom group elements in $\mathbb{G}$. Our DCF scheme makes *non-black-box* use of our DPF scheme.

Note that our DPF and DCF schemes use a keyed hash function H_S. When there is a trusted dealer, the key S can be uniformly sampled by the dealer. In our DPF and DCF protocols in the upcoming sections, it can be jointly sampled by two parties using one-time public coin-tossing. This hash key can be reused across polynomially many FSS key pairs.

Complexity analysis. Consider the group $\mathbb{G}$ (e.g., in [GI14, BGI16, Ds17, BGI19, BCG+21]) with the PRG-free implementation of $\mathsf{Convert}_{\mathbb{G}}$ (c.f. Appendix F.1).

Our DPF scheme has a full-domain evaluation that takes $1.5N$ RP calls, in contrast to the $2N$ RP calls in the state-of-the-art construction of [BGI16]. Its key generation algorithm uses about $2n + 2$ RP calls while this figure is about $4n$ in the prior work. In our scheme, the key size is $n \cdot \lambda + (\lambda + 1) + \log |\mathbb{G}|$ bits, and the evaluation algorithm takes about n RP calls, both remaining the same complexity as those in the prior work. In our DCF scheme, the full-domain evaluation requires $2.5N$ RP calls, in contrast to $4N$ RP calls in the state-of-the-art construction [BCG+21]. Its key generation needs about $4n + 2$ RP calls, in contrast to $8n$ RP calls in the prior work. The key size is $n \cdot \lambda + (\lambda + 1) + (n + 1) \cdot \log |\mathbb{G}|$ bits, and the evaluation requires about $2n$ RP calls, without any improvement.

Security. We prove the following theorems in Appendix D.2 and Appendix D.3. These theorems

Protocol Π_{DPF}

Parameters: Domain size $N = 2^n$ for $n \in \mathbb{N}$. Ring $\mathcal{R}$. Keyed hash function $\mathsf{H}_S : \mathbb{F}_{2^\lambda} \to \mathbb{F}_{2^\lambda}$. Function $\mathsf{Convert}_{\mathcal{R}} : \{0,1\}^* \to \mathcal{R}$. Let $\mathsf{H}' := \mathsf{hb} \circ \mathsf{H}_S$.

DPF Gen: This procedure can be executed many times. For each $b \in \{0,1\}$, P_b inputs $(\langle \alpha \rangle_b, \langle \beta \rangle_b^{\mathsf{A}}) \in [0, N) \times \mathcal{R}$ and proceeds as follows:

1. The two parties run sub-protocol Π_{PREP} (Figure 11), which, for each $b \in \{0,1\}$, returns $\langle \Delta \rangle_b$ and $\{(\mathsf{K}_b[\langle \alpha_i \rangle_{1-b}], \mathsf{M}_b[\langle \alpha_i \rangle_b])\}_{i \in [1,n]}$ to P_b such that $\mathsf{lsb}(\langle \Delta \rangle_0 \oplus \langle \Delta \rangle_1) = 1$, and $\mathsf{M}_b[\langle \alpha_i \rangle_b] = \mathsf{K}_{1-b}[\langle \alpha_i \rangle_b] \oplus \langle \alpha_i \rangle_b \cdot \langle \Delta \rangle_{1-b}$ for $i \in [1, n]$.
2. The two parties send $(\mathsf{sample}, \lambda)$ to $\mathcal{F}_{\mathsf{Rand}}$, which returns $W \in \{0,1\}^\lambda$ to them.
3. P_b computes $\langle s_0^0 \,\|\, t_0^0 \rangle_b := \langle \Delta \rangle_b \oplus W$. For $i \in [1, n-1]$, P_b sends to P_{1-b}
$$\langle \mathsf{CW}_i \rangle_b := (\oplus_{j \in [0, 2^{i-1})} \mathsf{H}_S(\langle s_{i-1}^j \,\|\, t_{i-1}^j \rangle_b)) \oplus \overline{\langle \alpha_i \rangle_b} \cdot \langle \Delta \rangle_b \oplus \mathsf{K}_b[\langle \alpha_i \rangle_{1-b}] \oplus \mathsf{M}_b[\langle \alpha_i \rangle_b],$$
receives $\langle \mathsf{CW}_i \rangle_{1-b}$ from P_{1-b}, and computes $\mathsf{CW}_i := \langle \mathsf{CW}_i \rangle_b \oplus \langle \mathsf{CW}_i \rangle_{1-b}$ and
$$\{\langle s_i^j \,\|\, t_i^j \rangle_b\}_{j \in [0, 2^i)} := \mathsf{DPF.NextLevel}(i, \{\langle s_{i-1}^j \,\|\, t_{i-1}^j \rangle_b\}_{j \in [0, 2^{i-1})}, \mathsf{CW}_i).$$
4. P_b samples $\mu_b \leftarrow \{0,1\}^\lambda$, computes
$$\langle \mathsf{Xhigh}^\sigma \,\|\, \mathsf{Xlow}^\sigma \rangle_b := \oplus_{j \in [0, 2^{n-1})} \mathsf{H}_S(\langle s_{n-1}^j \,\|\, t_{n-1}^j \rangle_b \oplus \sigma) \quad \text{for } \sigma \in \{0,1\},$$
$$d_b := \mathsf{H}'(\mu_b \oplus \mathsf{K}_b[\langle \alpha_n \rangle_{1-b}]) \oplus \mathsf{H}'(\mu_b \oplus \mathsf{K}_b[\langle \alpha_n \rangle_{1-b}] \oplus \langle \Delta \rangle_b) \oplus \langle \mathsf{Xhigh}^0 \oplus \mathsf{Xhigh}^1 \rangle_b,$$
sends (μ_b, d_b) to P_{1-b}, and receives (μ_{1-b}, d_{1-b}) from P_{1-b}. Then, P_b computes
$$\langle \mathsf{HCW} \rangle_b := \langle \mathsf{Xhigh}^{\overline{\langle \alpha_n \rangle_b}} \rangle_b \oplus \mathsf{H}'(\mu_b \oplus \mathsf{K}_b[\langle \alpha_n \rangle_{1-b}]) \oplus \mathsf{H}'(\mu_{1-b} \oplus \mathsf{M}_b[\langle \alpha_n \rangle_b]) \oplus \langle \alpha_n \rangle_b \cdot d_{1-b},$$
$$\langle \mathsf{LCW}^0 \rangle_b := \langle \mathsf{Xlow}^0 \rangle_b \oplus \langle \alpha_n \rangle_b \oplus b, \qquad \langle \mathsf{LCW}^1 \rangle_b := \langle \mathsf{Xlow}^1 \rangle_b \oplus \langle \alpha_n \rangle_b,$$
sends $\langle \mathsf{CW}_n \rangle_b := (\langle \mathsf{HCW} \rangle_b \,\|\, \langle \mathsf{LCW}^0 \rangle_b \,\|\, \langle \mathsf{LCW}^1 \rangle_b)$ to P_{1-b}, receives $\langle \mathsf{CW}_n \rangle_{1-b}$ from P_{1-b}, and computes $\mathsf{CW}_n := \langle \mathsf{CW}_n \rangle_b \oplus \langle \mathsf{CW}_n \rangle_{1-b}$ and
$$\{\langle s_n^j \,\|\, t_n^j \rangle_b\}_{j \in [0, N)} := \mathsf{DPF.NextLevel}(n, \{\langle s_{n-1}^j \,\|\, t_{n-1}^j \rangle_b\}_{j \in [0, 2^{n-1})}, \mathsf{CW}_n).$$
5. **(Binary field $\mathcal{R} = \mathbb{F}_{2^\ell}$, without $\mathcal{F}_{\mathsf{OLE}}$)**
P_b computes $\langle \mathsf{CW}_{n+1} \rangle_b^{\mathsf{A}} := (\sum_{j \in [0, N)} \mathsf{Convert}_{\mathcal{R}}(\langle s_n^j \rangle_b)) + \langle \beta \rangle_b^{\mathsf{A}}$.
(General ring $\mathcal{R}$, using $\mathcal{F}_{\mathsf{OLE}}$)
The two parties run sub-protocol Π_{MULT} (Figure 12), which, for each $b \in \{0,1\}$, takes as input
$$\langle A \rangle_b^{\mathsf{A}} := (-1)^b \cdot \textstyle\sum_{j \in [0, N)} \langle t_n^j \rangle_b \in \mathcal{R},$$
$$\langle B \rangle_b^{\mathsf{A}} := (-1)^{1-b} \cdot \textstyle\sum_{j \in [0, N)} \mathsf{Convert}_{\mathcal{R}}(\langle s_n^j \rangle_b) + \langle \beta \rangle_b^{\mathsf{A}} \in \mathcal{R},$$
and returns $\langle \mathsf{CW}_{n+1} \rangle_b^{\mathsf{A}}$ to P_b.
In either case, P_b sends $\langle \mathsf{CW}_{n+1} \rangle_b^{\mathsf{A}}$ to P_{1-b}, receives $\langle \mathsf{CW}_{n+1} \rangle_{1-b}^{\mathsf{A}}$ from P_{1-b}, and computes $\mathsf{CW}_{n+1} := \langle \mathsf{CW}_{n+1} \rangle_b^{\mathsf{A}} + \langle \mathsf{CW}_{n+1} \rangle_{1-b}^{\mathsf{A}}$.
6. P_b computes $k_b := (\langle \Delta \rangle_b \oplus W, \{\mathsf{CW}_i\}_{i \in [1, n+1]})$ and $\langle \mathbf{r}^{(j)} \rangle_b^{\mathsf{A}} := \mathsf{DPF.Eval}(b, k_b, j)$ for $j \in [0, N)$, and outputs $\langle \mathbf{r} \rangle_b^{\mathsf{A}} \in \mathcal{R}^N$.

Figure 10: DPF correlation generation in the $(\mathcal{F}_{\mathsf{COT}}, \mathcal{F}_{\mathsf{Rand}}, \mathcal{F}_{\mathsf{OLE}})$-hybrid model.

turn to the intuition that $\mathsf{CW}_1, \ldots, \mathsf{CW}_n$ are masked by pseudorandom CCR outputs (as the root and the first $n-1$ on-path shared nodes are Δ), and $\mathsf{CW}_{n+1}, \mathsf{VCW}_1, \ldots, \mathsf{VCW}_n$ are masked by some pseudorandom $\mathsf{Convert}_{\mathbb{G}}$ terms taking (pseudo)random CCR outputs as input.

Theorem 3. *Given CCR function* $\mathsf{H} : \mathbb{F}_{2^\lambda} \to \mathbb{F}_{2^\lambda}$*, function* $\mathsf{Convert}_{\mathbb{G}} : \mathbb{F}_{2^{\lambda-1}} \to \mathbb{G}$*, and keyed hash function* $\mathsf{H}_S(x) := \mathsf{H}(S \oplus x)$ *with some key* $S \leftarrow \mathbb{F}_{2^\lambda}$*, Figure 8 gives a DPF scheme with domain*

Protocol Π_{PREP}

Initialize: This procedure is executed only once for each $b \in \{0,1\}$. The two parties send (init) to $\mathcal{F}_{\mathsf{COT}}^b$ with identifier b, which returns $\Delta'_b \in \{0,1\}^\lambda$ to P_b. P_b sends $\mathsf{lsb}(\Delta'_b)$ to P_{1-b}, receives $\mathsf{lsb}(\Delta'_{1-b})$ from P_{1-b}, and sets $\langle\Delta\rangle_b := \Delta'_b \oplus (0^{\lambda-1} \,\|\, (\mathsf{lsb}(\Delta'_{1-b}) \oplus b))$ such that $\mathsf{lsb}(\langle\Delta\rangle_0 \oplus \langle\Delta\rangle_1) = 1$.

For each $b \in \{0,1\}$: P_b inputs $\langle\alpha\rangle_b \in \{0,1\}^n$ and proceeds as follows.

1-1. The two parties send (extend, n) to $\mathcal{F}_{\mathsf{COT}}^b$ with identifier b, which returns $\mathbf{k}_b \in \mathbb{F}_{2^\lambda}^n$ to P_b and $(\mathbf{r}_{1-b}, \mathbf{m}_{1-b}) \in \mathbb{F}_2^n \times \mathbb{F}_{2^\lambda}^n$ to P_{1-b} such that $\mathbf{m}_{1-b} = \mathbf{k}_b \oplus \mathbf{r}_{1-b} \cdot \Delta'_b$.

1-2. P_b sets $\mathbf{g}_b := \langle\alpha\rangle_b \oplus \mathbf{r}_b$, sends $\mathbf{g}_b$ to P_{1-b}, and receives $\mathbf{g}_{1-b}$ from P_{1-b}. For $i \in [1,n]$, P_b sets

$$\mathsf{K}_b[\langle\alpha_i\rangle_{1-b}] := \mathbf{k}_b^{(i)} \oplus \mathbf{g}_{1-b}^{(i)} \cdot \langle\Delta\rangle_b, \quad \mathsf{M}_b[\langle\alpha_i\rangle_b] := \mathbf{m}_b^{(i)} \oplus \mathbf{r}_b^{(i)} \cdot (0^{\lambda-1} \,\|\, (\mathsf{lsb}(\Delta'_b) \oplus (1-b))).$$

1-3. P_b outputs $\langle\Delta\rangle_b$ and $\{(\mathsf{K}_b[\langle\alpha_i\rangle_{1-b}], \mathsf{M}_b[\langle\alpha_i\rangle_b])\}_{i\in[1,n]}$.

Figure 11: Preprocessing sub-protocol for DPF/DCF correlation generation.

Protocol Π_{MULT}

For each $b \in \{0,1\}$: P_b inputs $(\langle A\rangle_b^{\mathsf{A}}, \langle B\rangle_b^{\mathsf{A}}) \in \mathcal{R}^2$ and proceeds as follows.

1. The two parties send $(\mathsf{extend}, 2)$ to $\mathcal{F}_{\mathsf{OLE}}$, which, for each $b \in \{0,1\}$, returns $(\mathbf{x}_b, \mathbf{z}_b) \in \mathcal{R}^2 \times \mathcal{R}^2$ to P_b such that $\mathbf{z}_0 + \mathbf{z}_1 = \mathbf{x}_0 \cdot \mathbf{x}_1$.
2. P_b computes $(\gamma_b, \zeta_b) := (\langle A\rangle_b^{\mathsf{A}}, \langle B\rangle_b^{\mathsf{A}}) + (\mathbf{x}_b^{(b)}, \mathbf{x}_b^{(1-b)})$, sends (γ_b, ζ_b) to P_{1-b}, receives $(\gamma_{1-b}, \zeta_{1-b})$ from P_{1-b}, and defines $\langle A \cdot B\rangle_b^{\mathsf{A}} := \langle A\rangle_b^{\mathsf{A}} \cdot \langle B\rangle_b^{\mathsf{A}} + \langle A\rangle_b^{\mathsf{A}} \cdot \zeta_{1-b} - \mathbf{x}_b^{(1-b)} \cdot \gamma_{1-b} + \mathbf{z}_b^{(0)} + \mathbf{z}_b^{(1)}$.
3. P_b outputs $\langle A \cdot B\rangle_b^{\mathsf{A}}$.

Figure 12: OLE-based multiplication sub-protocol.

$[0, N)$ *and range* $\mathbb{G}$.

Theorem 4. *Given CCR function* $\mathsf{H} : \mathbb{F}_{2^\lambda} \to \mathbb{F}_{2^\lambda}$*, function* $\mathsf{Convert}_{\mathbb{G}} : \mathbb{F}_{2^\ell} \to \mathbb{G}$ *with* $\ell \in \{\lambda-1, \lambda\}$*, and keyed hash function* $\mathsf{H}_S(x) := \mathsf{H}(S \oplus x)$ *with some key* $S \leftarrow \mathbb{F}_{2^\lambda}$*, Figure 9 gives a DCF scheme with domain* $[0, N)$ *and range* $\mathbb{G}$*.*

5.2 DPF Correlation Generation

We define a leveled evaluation algorithm $\mathsf{DPF.NextLevel}$ such that, on input a level index $i \in [1, n]$, all nodes on the $(i-1)$-th level of the share of a shared pseudorandom correlated GGM tree, and the public correction word CW_i for the i-th level, outputs all nodes one the i-th level.

In Figure 10, we present our DPF correlation generation protocol Π_{DPF}. This protocol operates in the $(\mathcal{F}_{\mathsf{COT}}, \mathcal{F}_{\mathsf{Rand}}, \mathcal{F}_{\mathsf{OLE}})$-hybrid model. $\mathcal{F}_{\mathsf{Rand}}$ is the standard coin-tossing functionality that outputs a uniform string to both parties. $\mathcal{F}_{\mathsf{OLE}}$ is the functionality for oblivious linear evaluation (OLE) on ring $\mathcal{R}$, where P_0 (resp., P_1) is given *random* $(\mathbf{x}_0, \mathbf{z}_0) \in \mathcal{R}^N \times \mathcal{R}^N$ (resp., $(\mathbf{x}_1, \mathbf{z}_1) \in \mathcal{R}^N \times \mathcal{R}^N$) such that $\mathbf{z}_0 + \mathbf{z}_1$ equals the component-wise multiplication $\mathbf{x}_0 \odot \mathbf{x}_1$. We refer readers to Appendix F.2 and Appendix F.3 for the definitions and instantiations of $\mathcal{F}_{\mathsf{Rand}}$ and $\mathcal{F}_{\mathsf{OLE}}$. If β is a bit-string, Π_{DPF} never uses $\mathcal{F}_{\mathsf{OLE}}$.

Π_{DPF} requires $\mathcal{F}_{\mathsf{Rand}}$ for the following reason. Note that Π_{DPF} uses the same global offset Δ as the roots of polynomially many shared trees, each of which defines a fresh DPF correlation. So, the two shares of this identical root should be "re-randomized" to avoid the identical per-party shares of the defined correlations. The two parties do this re-randomization by calling $\mathcal{F}_{\mathsf{Rand}}$ for a public randomness W and XORing this value to their shares of Δ, respectively.

In Π_{DPF}, the key S of the keyed hash function H_S can be produced by one $\mathcal{F}_{\mathsf{Rand}}$ invocation before protocol execution, and we omit this setup for simplicity.

Security. We prove Theorem 5 in Appendix D.4. This proof will consider polynomially many concurrent **Gen** executions that uses the one-time initialized Δ. Intuitively, the security primarily follows from the COT-based secure computation of correction words, where the COT tuples are related to the global offset Δ so that the transcripts are masked by CCR responses. In particular, the intermediate transcript d_b is masked by a CCR response coming from a legal CCR query with overwhelming probability due to the uniform μ_b.

Theorem 5. *Given CCR function* $\mathsf{H} : \mathbb{F}_{2^\lambda} \to \mathbb{F}_{2^\lambda}$*, function* $\mathsf{Convert}_{\mathcal{R}} : \mathbb{F}_{2^{\lambda-1}} \to \mathcal{R}$*, and keyed hash function* $\mathsf{H}_S(x) := \mathsf{H}(S \oplus x)$ *with some key* $S \leftarrow \mathbb{F}_{2^\lambda}$*, protocol* Π_{DPF} *(Figure 10) UC-realizes functionality* $\mathcal{F}_{\mathsf{DPF}}$ *(Figure 7) against any semi-honest adversary in the* $(\mathcal{F}_{\mathsf{COT}}, \mathcal{F}_{\mathsf{Rand}}, \mathcal{F}_{\mathsf{OLE}})$*-hybrid model. If* $\mathcal{R} = \mathbb{F}_{2^\ell}$ *for* $\ell \in \mathbb{N}$*, protocol* Π_{DPF} *never invokes* $\mathcal{F}_{\mathsf{OLE}}$*.*

Communication optimization. Π_{DPF} has the following two optimizations:

- For t concurrent **Gen** executions (e.g., in its applications to RAM-based computation [Ds17], FSS-based MPC [BCG+21], and OLE extension [BCG+20], etc), each P_b can compress all μ_b's in these executions via a PRF $F : \mathbb{F}_{2^\lambda} \times \{0,1\}^* \to \mathbb{F}_{2^\lambda}$ with a fresh key $k_{\mathsf{prf},b} \leftarrow \mathbb{F}_{2^\lambda}$ sampled *after receiving its COT outputs (from both* $\mathcal{F}_{\mathsf{COT}}^b$ *and* $\mathcal{F}_{\mathsf{COT}}^{1-b}$*) in all executions.* For each execution with sub-session ID ssid, the two parties define $\mu_b := F(k_{\mathsf{prf},b}, \mathsf{ssid})$.

- All invocations of $\mathcal{F}_{\mathsf{Rand}}$ can be compressed via another independent PRF key sampled *after the one-time initialization of* $\mathcal{F}_{\mathsf{COT}}^b$ *and* $\mathcal{F}_{\mathsf{COT}}^{1-b}$ so that the root of each P_b's tree is (pseudo)random.

- Another method to save the communication for random μ_b's is to replace H_S by a hash function that meets "CCR for naturally derived keys" [ZRE15, GKWY20], which can also be implemented in one RP call. Note that μ_b is introduced to prevent the replay attack, which results from the manipulation of COT outputs, against the hashing mask in d_b. The alternative hash function addresses this attack by adding non-repeating tweaks.

Complexity analysis (binary field). Consider the complexity per execution when the first PRF-based optimization is used in t concurrent **Gen** executions. The cost is symmetric. Π_{DPF} uses n COT tuples per party and one $\mathcal{F}_{\mathsf{Rand}}$ call. Each party sends $(n+1) + (n+1)\cdot\lambda + \frac{\lambda}{t} + \log|\mathcal{R}|$ bits. The computation per party is dominated by the tree expansion in n $\mathsf{DPF.NextLevel}$ calls, or $1.5N$ RP calls. Π_{DPF} runs in $n+3$ rounds (without counting the one-time setup).

In contrast, the binary-field protocol [Ds17] can be implemented from GMW-style 2PC and n string OTs each with $(\lambda-1)$-bit payloads. One can cast these string OTs into n precomputed COT tuples according to [IKNP03, Bea95]. Using these tuples, each party sends $n + n\cdot(3\lambda-1) + \log|\mathcal{R}|$ bits, and the computation per party is dominated by the $2N$ RP calls in GGM tree expansion. This protocol can proceed in $2n+2$ rounds: one for sending n masked choice bits, two for sharing and revealing each of the first n correction words, and one for revealing the $(n+1)$-th correction word. Our savings in computation, communication, and round complexity are about 25%, 66.6%, and 50%, respectively.

We implement Π_{PREP} and Π_{DPF} in `C++`, and perform benchmarks on a pair of Amazon EC2 `R5.xlarge` instances. We take binary fields $\mathcal{R} = \mathbb{F}_{2^{127}}$ and $\mathcal{R} = \mathbb{F}_2$ under computational security parameter $\lambda \approx 128$. The reported time include both distributed key generation and full-domain evaluation. We set 1Gbps bandwidth with no latency as our LAN setting, and 20Mbps bandwidth

		$n=20$	$n=22$	$n=24$	$n=26$	$n=28$
$\mathcal{R}=\mathbb{F}_{2^{127}}$	LAN	50	120	397	1501	5920
	WAN	2752	3020	3492	4786	9355
$\mathcal{R}=\mathbb{F}_2$	LAN	29	30	34	52	120
	WAN	2930	3132	3337	3554	3823

Table 3: **The efficiency of distributed correlation generation for our DPF scheme.** All numbers are in milliseconds (ms).

with $100ms$ latency as our WAN setting. The results are shown in Table 3. We can see that our protocol is practically efficient, especially for two-server PIR. Although all numbers are reported based on one thread, performing one correlation generation for 2^{28} 127-bit values takes about 6 seconds, which is about 30% to 40% faster than the performance from a prior implementation in the same threads [Ds17].

Complexity analysis (general ring). The two parties additionally need two precomputed OLE tuples for the secure multiplication. Overall, each party sends $(n+1)+(n+1)\cdot\lambda+\frac{\lambda}{t}+3\cdot\log|\mathcal{R}|$ bits, and the protocol runs in $n+4$ rounds.

In contrast, the binary-field protocol [Ds17] can be adapted for the general-ring CW_{n+1} in the DPF scheme [BGI16]. Securely computing this CW_{n+1} consumes two OLE tuples and needs the level-by-level 2PC, which leads to two additional bits in each OT payload per level, to share the last-level control bit $\langle t_n\rangle_1$. Each party sends at most $n+n\cdot(3\lambda+3)+3\cdot\log|\mathcal{R}|$ bits, and the protocol runs in $2n+3$ rounds. The improvement is the same as the binary-field case.

5.3 DCF Correlation Generation

Our DCF protocol Π_{DCF} in Figure 13 extends Π_{DPF} by also computing n value correction words and defining the evaluation result as per our DCF scheme. If β is a bit-string, the two parties can compute n value correction words without using precomputed OLE tuples. Otherwise, for a general ring element β, these correction words are obtained from OLE-based secure multiplication.

Security. We prove Theorem 6 in Appendix D.5, where polynomially many concurrent **Gen** executions are considered. The security is also based on the COT- and OLE-based secure computation of the n additional correction words of our DCF scheme. Note that the intermediate y_b^i's are pseudorandom due the masking CCR responses, which are for the legal CCR queries with overwhelming probability in the presence of uniform x_b^i's.

Theorem 6. *Given CCR function $\mathsf{H}:\mathbb{F}_{2^\lambda}\to\mathbb{F}_{2^\lambda}$, function $\mathsf{Convert}_{\mathcal{R}}:\mathbb{F}_{2^\ell}\to\mathcal{R}$ for $\ell\in\{\lambda-1,\lambda\}$, and keyed hash function $\mathsf{H}_S(x):=\mathsf{H}(S\oplus x)$ with some key $S\leftarrow\mathbb{F}_{2^\lambda}$, protocol Π_{DCF} (Figure 13) UC-realizes functionality $\mathcal{F}_{\mathsf{DCF}}$ (Figure 7) against any semi-honest adversary in the $(\mathcal{F}_{\mathsf{COT}},\mathcal{F}_{\mathsf{Rand}},\mathcal{F}_{\mathsf{OLE}})$-hybrid model. If $\mathcal{R}=\mathbb{F}_{2^\ell}$ for $\ell\in\mathbb{N}$, protocol Π_{DCF} never invokes $\mathcal{F}_{\mathsf{OLE}}$.*

Communication optimization. The optimizations in Section 5.2 also applies to the DCF protocol Π_{DCF}. Moreover, the random elements $\{x_b^i\}_{i\in[1,n]}$ in Π_{DCF} can also be compressed using the same technique for the random μ_b's.

Complexity analysis (binary field). Consider the complexity per execution when the first PRF-based optimization is used in t concurrent **Gen** executions. The cost is symmetric. Π_{DCF} consumes n COT tuples per party and one $\mathcal{F}_{\mathsf{Rand}}$ call. Each party sends $(n+1)+(n+1)\cdot\lambda+\frac{\lambda}{t}+(2n+1)\cdot\log|\mathcal{R}|$ bits, and the computation per party comes from the $2.5N$ RP calls in the tree expansion. Π_{DCF} has round complexity $n+3$, the same as Π_{DPF} in the binary-field case.

Protocol Π_{DCF}

Parameters: Domain size $N = 2^n$ for $n \in \mathbb{N}$. Ring $\mathcal{R}$. Keyed hash function $\mathsf{H}_S : \mathbb{F}_{2^\lambda} \to \mathbb{F}_{2^\lambda}$. Function $\mathsf{Convert}_{\mathcal{R}} : \{0,1\}^* \to \mathcal{R}$. Let $\mathsf{H}^* := \mathsf{Convert}_{\mathcal{R}} \circ \mathsf{H}_S$.

DCF Gen: This procedure can be executed many times. For each $b \in \{0,1\}$, P_b inputs $(\langle\alpha\rangle_b, \langle\beta\rangle_b^{\mathsf{A}}) \in [0,N) \times \mathcal{R}$ and proceeds as in Π_{DPF} (Figure 8), with the same Step 1, 2 and the following modifications to the subsequent steps:

3. Along with $\langle \mathsf{CW}_i\rangle_b$ for $i \in [1, n-1]$, P_b samples $x_b^i \leftarrow \{0,1\}^\lambda$, computes
$$y_b^i := \mathsf{H}^*(x_b^i \oplus \mathsf{K}_b[\langle\alpha_i\rangle_{1-b}]) - \mathsf{H}^*(x_b^i \oplus \mathsf{K}_b[\langle\alpha_i\rangle_{1-b}] \oplus \langle\Delta\rangle_b) + \langle\beta\rangle_b^{\mathsf{A}} - 2 \cdot \langle\alpha_i\rangle_b \cdot \langle\beta\rangle_b^{\mathsf{A}},$$
sends (x_b^i, y_b^i) to P_{1-b}, receive (x_{1-b}^i, y_{1-b}^i) from P_{1-b}, and computes
$$\langle\alpha_i \cdot \beta\rangle_b^{\mathsf{A}} := \langle\alpha_i\rangle_b \cdot \langle\beta\rangle_b^{\mathsf{A}} - \mathsf{H}^*(x_b^i \oplus \mathsf{K}_b[\langle\alpha_i\rangle_{1-b}]) + \mathsf{H}^*(x_{1-b}^i \oplus \mathsf{M}_b[\langle\alpha_i\rangle_b]) + \langle\alpha_i\rangle_b \cdot y_{1-b}^i.$$
4. Along with $\langle \mathsf{CW}_n\rangle_b$, P_b repeats Step 3 for $i = n$ and computes $\langle\alpha_n \cdot \beta\rangle_b^{\mathsf{A}}$.
5. For $i \in [1,n]$ and $j \in [0, 2^{i-1})$, P_b computes $\langle v_i^j\rangle_b := \mathsf{H}_S(\langle s_{i-1}^j \,\|\, t_{i-1}^j\rangle_b \oplus 2)$ and $\langle\alpha_0 \cdot \beta\rangle_b^{\mathsf{A}} := 0$. P_b computes $\langle \mathsf{CW}_{n+1}\rangle_b^{\mathsf{A}}$ by using $\langle\alpha_n \cdot \beta\rangle_b^{\mathsf{A}}$ instead of $\langle\beta\rangle_b^{\mathsf{A}}$, and:
   (Binary field $\mathcal{R} = \mathbb{F}_{2^\ell}$, without $\mathcal{F}_{\mathsf{OLE}}$) For $i \in [1,n]$ in parallel:
   P_b computes $\langle \mathsf{VCW}_i\rangle_b^{\mathsf{A}} := (\sum_{j\in[0,2^{i-1})} \mathsf{Convert}_{\mathcal{R}}(\langle v_i^j\rangle_b)) + \langle\alpha_i \cdot \beta\rangle_b^{\mathsf{A}} - \langle\alpha_{i-1} \cdot \beta\rangle_b^{\mathsf{A}}$.
   (General ring $\mathcal{R}$, using $\mathcal{F}_{\mathsf{OLE}}$) For $i \in [1,n]$ in parallel:
   The two parties run sub-protocol Π_{MULT} (Figure 12), which, for each $b \in \{0,1\}$, takes as input
$$\langle A_i\rangle_b^{\mathsf{A}} := (-1)^b \cdot \textstyle\sum_{j\in[0,2^{i-1})} \langle t_{i-1}^j\rangle_b \in \mathcal{R},$$
$$\langle B_i\rangle_b^{\mathsf{A}} := (-1)^{1-b} \cdot \textstyle\sum_{j\in[0,2^{i-1})} \mathsf{Convert}_{\mathcal{R}}(\langle v_i^j\rangle_b) + \langle\alpha_i \cdot \beta\rangle_b^{\mathsf{A}} - \langle\alpha_{i-1} \cdot \beta\rangle_b^{\mathsf{A}} \in \mathcal{R},$$
   and returns $\langle \mathsf{VCW}_i\rangle_b^{\mathsf{A}}$ to P_b.
   In either case, along with $\langle \mathsf{CW}_{n+1}\rangle_b^{\mathsf{A}}$, P_b sends $\langle \mathsf{VCW}_i\rangle_b^{\mathsf{A}}$ to P_{1-b}, receives $\langle \mathsf{VCW}_i\rangle_{1-b}^{\mathsf{A}}$ from P_{1-b}, and computes $\mathsf{VCW}_i := \langle \mathsf{VCW}_i\rangle_b^{\mathsf{A}} + \langle \mathsf{VCW}_i\rangle_{1-b}^{\mathsf{A}}$.
6. P_b computes $k_b := (\langle\Delta\rangle_b \oplus W, \{\mathsf{CW}_i\}_{i\in[1,n+1]}, \{\mathsf{VCW}_i\}_{i\in[1,n]})$ and $\langle \mathbf{r}^{(j)}\rangle_b^{\mathsf{A}} := \mathsf{DCF.Eval}(b, k_b, j)$ for $j \in [0,N)$, and outputs $\langle \mathbf{r}\rangle_b^{\mathsf{A}} \in \mathcal{R}^N$.

Figure 13: DCF correlation generation in the $(\mathcal{F}_{\mathsf{COT}}, \mathcal{F}_{\mathsf{Rand}}, \mathcal{F}_{\mathsf{OLE}})$-hybrid model.

In contrast, the state-of-the-art protocol of [BCG+21] requires n string OTs to run GMW-style 2PC. The string OTs consume n precomputed COT tuples and have payloads of $(\lambda-1)+2\cdot\log|\mathcal{R}|$ bits. Using n COT tuples, each party sends $n + n\cdot(3\lambda - 1 + 5\cdot\log|\mathcal{R}|) + \log|\mathcal{R}|$ bits, and the computation per party is dominated by the $4N$ RP calls in GGM tree expansion in $2n+2$ rounds. Our savings in computation and round complexity are 37.5% and 50%, respectively. For a typical ring $\mathcal{R}$ with size $|\mathcal{R}| \approx 2^\lambda$, the communication reduction is about 62.5%. When $\mathcal{R}$ is sufficiently small, this reduction can be 66.6%.

Complexity analysis (general ring). Π_{DCF} also works for general $\mathcal{R}$ at the cost of additionally using $2n+2$ precomputed OLE tuples. This general-ring version proceeds in $n+4$ rounds, and the overall outgoing communication per party is $(n+1) + (n+1)\cdot\lambda + \frac{\lambda}{t} + (4n+3)\cdot\log|\mathcal{R}|$ bits.

In contrast, the OT-based protocol [BCG+21] can run in $2n+3$ rounds. Each party sends at most $n + n\cdot(3\lambda+3+4\cdot\log|\mathcal{R}|) + (3n+3)\cdot\log|\mathcal{R}|$ bits and uses $2n+2$ OLE tuples. Our savings in communication and round complexity are about 50% $\sim$ 66.6% and 50%, respectively, for typical ring size $|\mathcal{R}| \leq 2^\lambda$.

Acknowledgements

Work of Kang Yang is supported by the National Key Research and Development Program of China (Grant No. 2022YFB2702000), and by the National Natural Science Foundation of China (Grant Nos. 62102037, 61932019). Work of Xiao Wang is supported by DARPA under Contract No. HR001120C0087, NSF award #2016240, and research awards from Meta and Google. The views, opinions, and/or findings expressed are those of the author(s) and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government. Work of Jiang Zhang is supported by the National Key Research and Development Program of China (Grant No. 2022YFB2702000), and by the National Natural Science Foundation of China (Grant Nos. 62022018, 61932019). Work of Zheli Liu is supported by the National Natural Science Foundation of China (Grant No. 62032012).

References

[ALSZ13] Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. More efficient oblivious transfer and extensions for faster secure computation. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013*, pages 535–548. ACM Press, November 2013.

[BBC+21] Dan Boneh, Elette Boyle, Henry Corrigan-Gibbs, Niv Gilboa, and Yuval Ishai. Lightweight techniques for private heavy hitters. In *2021 IEEE Symposium on Security and Privacy*, pages 762–776. IEEE Computer Society Press, May 2021.

[BCG+19a] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Peter Rindal, and Peter Scholl. Efficient two-round OT extension and silent non-interactive secure computation. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 291–308. ACM Press, November 2019.

[BCG+19b] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators: Silent OT extension and more. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 489–518. Springer, Heidelberg, August 2019.

[BCG+20] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators from ring-LPN. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 387–416. Springer, Heidelberg, August 2020.

[BCG+21] Elette Boyle, Nishanth Chandran, Niv Gilboa, Divya Gupta, Yuval Ishai, Nishant Kumar, and Mayank Rathee. Function secret sharing for mixed-mode and fixed-point secure computation. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part II*, volume 12697 of *LNCS*, pages 871–900. Springer, Heidelberg, October 2021.

[BCG+22] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Nicolas Resch, and Peter Scholl. Correlated pseudorandomness from expand-accumulate codes. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 603–633. Springer, Heidelberg, August 2022.

[BCGI18] Elette Boyle, Geoffroy Couteau, Niv Gilboa, and Yuval Ishai. Compressing vector OLE. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 896–912. ACM Press, October 2018.

[Bea95] Donald Beaver. Precomputing oblivious transfer. In Don Coppersmith, editor, *CRYPTO'95*, volume 963 of *LNCS*, pages 97–109. Springer, Heidelberg, August 1995.

[BGI16] Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing: Improvements and extensions. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 1292–1303. ACM Press, October 2016.

[BGI19] Elette Boyle, Niv Gilboa, and Yuval Ishai. Secure computation with preprocessing via function secret sharing. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part I*, volume 11891 of *LNCS*, pages 341–371. Springer, Heidelberg, December 2019.

[BHKR13] Mihir Bellare, Viet Tung Hoang, Sriram Keelveedhi, and Phillip Rogaway. Efficient garbling from a fixed-key blockcipher. In *2013 IEEE Symposium on Security and Privacy*, pages 478–492. IEEE Computer Society Press, May 2013.

[BMRS21] Carsten Baum, Alex J. Malozemoff, Marc B. Rosen, and Peter Scholl. Mac'n'cheese: Zero-knowledge proofs for boolean and arithmetic circuits with nested disjunctions. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part IV*, volume 12828 of *LNCS*, pages 92–122, Virtual Event, August 2021. Springer, Heidelberg.

[BN18] Srimanta Bhattacharya and Mridul Nandi. Full indifferentiable security of the xor of two or more random permutations using the χ^2 method. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 387–412. Springer, Heidelberg, April / May 2018.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001.

[CKKZ12] Seung Geol Choi, Jonathan Katz, Ranjit Kumaresan, and Hong-Sheng Zhou. On the security of the "free-XOR" technique. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 39–53. Springer, Heidelberg, March 2012.

[CRR21] Geoffroy Couteau, Peter Rindal, and Srinivasan Raghuraman. Silver: Silent VOLE and oblivious transfer from hardness of decoding structured LDPC codes. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part III*, volume 12827 of *LNCS*, pages 502–534, Virtual Event, August 2021. Springer, Heidelberg.

[CS14] Shan Chen and John P. Steinberger. Tight security bounds for key-alternating ciphers. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 327–350. Springer, Heidelberg, May 2014.

[DIO21] Samuel Dittmer, Yuval Ishai, and Rafail Ostrovsky. Line-point zero knowledge and its applications. In *2nd Conference on Information-Theoretic Cryptography*, 2021.

[DNNR17] Ivan Damgård, Jesper Buus Nielsen, Michael Nielsen, and Samuel Ranellucci. The TinyTable protocol for 2-party secure computation, or: Gate-scrambling revisited. In

Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 167–187. Springer, Heidelberg, August 2017.

[DPSZ12] Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 643–662. Springer, Heidelberg, August 2012.

[Ds17] Jack Doerner and abhi shelat. Scaling ORAM for secure computation. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 523–535. ACM Press, October / November 2017.

[GGM84] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions (extended abstract). In *25th FOCS*, pages 464–479. IEEE Computer Society Press, October 1984.

[GI14] Niv Gilboa and Yuval Ishai. Distributed point functions and their applications. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 640–658. Springer, Heidelberg, May 2014.

[GKCG22] Kanav Gupta, Deepak Kumaraswamy, Nishanth Chandran, and Divya Gupta. Llama: A low latency math library for secure inference. Privacy Enhancing Technologies Symposium (PETS 2022), 2022.

[GKWY20] Chun Guo, Jonathan Katz, Xiao Wang, and Yu Yu. Efficient and secure multiparty computation from fixed-key block ciphers. In *2020 IEEE Symposium on Security and Privacy*, pages 825–841. IEEE Computer Society Press, May 2020.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229. ACM Press, May 1987.

[GNN17] Satrajit Ghosh, Jesper Buus Nielsen, and Tobias Nilges. Maliciously secure oblivious linear function evaluation with constant overhead. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part I*, volume 10624 of *LNCS*, pages 629–659. Springer, Heidelberg, December 2017.

[GPR+21] Gayathri Garimella, Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. Oblivious key-value stores and amplification for private set intersection. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part II*, volume 12826 of *LNCS*, pages 395–425, Virtual Event, August 2021. Springer, Heidelberg.

[HK21] David Heath and Vladimir Kolesnikov. One hot garbling. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 574–593. ACM Press, November 2021.

[HL10] Carmit Hazay and Yehuda Lindell. *Efficient Secure Two-Party Protocols - Techniques and Constructions*. ISC. Springer, Heidelberg, 2010.

[HSS17] Carmit Hazay, Peter Scholl, and Eduardo Soria-Vazquez. Low cost constant round MPC combining BMR and oblivious transfer. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part I*, volume 10624 of *LNCS*, pages 598–628. Springer, Heidelberg, December 2017.

[IKNP03] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 145–161. Springer, Heidelberg, August 2003.

[KOS16] Marcel Keller, Emmanuela Orsini, and Peter Scholl. MASCOT: Faster malicious arithmetic secure computation with oblivious transfer. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 830–842. ACM Press, October 2016.

[KPR18] Marcel Keller, Valerio Pastro, and Dragos Rotaru. Overdrive: Making SPDZ great again. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 158–189. Springer, Heidelberg, April / May 2018.

[KS08] Vladimir Kolesnikov and Thomas Schneider. Improved garbled circuit: Free XOR gates and applications. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *ICALP 2008, Part II*, volume 5126 of *LNCS*, pages 486–498. Springer, Heidelberg, July 2008.

[NNOB12] Jesper Buus Nielsen, Peter Sebastian Nordholt, Claudio Orlandi, and Sai Sheshank Burra. A new approach to practical active-secure two-party computation. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 681–700. Springer, Heidelberg, August 2012.

[Pat09] Jacques Patarin. The "coefficients H" technique (invited talk). In Roberto Maria Avanzi, Liam Keliher, and Francesco Sica, editors, *SAC 2008*, volume 5381 of *LNCS*, pages 328–345. Springer, Heidelberg, August 2009.

[RS08] Phillip Rogaway and John P. Steinberger. Constructing cryptographic hash functions from fixed-key blockciphers. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 433–450. Springer, Heidelberg, August 2008.

[RS21] Peter Rindal and Phillipp Schoppmann. VOLE-PSI: Fast OPRF and circuit-PSI from vector-OLE. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part II*, volume 12697 of *LNCS*, pages 901–930. Springer, Heidelberg, October 2021.

[SGRR19] Phillipp Schoppmann, Adrià Gascón, Leonie Reichert, and Mariana Raykova. Distributed vector-OLE: Improved constructions and implementation. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 1055–1072. ACM Press, November 2019.

[WYKW21] Chenkai Weng, Kang Yang, Jonathan Katz, and Xiao Wang. Wolverine: Fast, scalable, and communication-efficient zero-knowledge proofs for boolean and arithmetic circuits. In *2021 IEEE Symposium on Security and Privacy*, pages 1074–1091. IEEE Computer Society Press, May 2021.

[WYX+21] Chenkai Weng, Kang Yang, Xiang Xie, Jonathan Katz, and Xiao Wang. Mystique: Efficient conversions for zero-knowledge proofs with applications to machine learning. In Michael Bailey and Rachel Greenstadt, editors, *USENIX Security 2021*, pages 501–518. USENIX Association, August 2021.

[YSWW21] Kang Yang, Pratik Sarkar, Chenkai Weng, and Xiao Wang. QuickSilver: Efficient and affordable zero-knowledge proofs for circuits and polynomials over any field. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 2986–3001. ACM Press, November 2021.

[YWL+20] Kang Yang, Chenkai Weng, Xiao Lan, Jiang Zhang, and Xiao Wang. Ferret: Fast extension for correlated OT with small communication. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 1607–1626. ACM Press, November 2020.

[ZRE15] Samee Zahur, Mike Rosulek, and David Evans. Two halves make a whole - reducing data transfer in garbled circuits using half gates. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 220–250. Springer, Heidelberg, April 2015.

A Circular Correlation Robustness for Restricted Queries

In some of our protocols, we use a *public* keyed hash function H_S and require that the responses from the CCR oracle $\mathcal{O}^{\mathsf{ccr}}_{\mathsf{H}_S,\Delta}(\cdot,\cdot)$ are pseudorandom from the view of the distinguisher unaware of the global key Δ. The distinguisher can only make *restricted queries* to this CCR oracle in the sense that all oracle queries are the type 5 operations for $\mathcal{O}(x_j, b_j) = \mathcal{O}^{\mathsf{ccr}}_{\mathsf{H}_S,\Delta}(x_j, b_j)$ in Definition 3.

The considered operations are reminiscent of those under the definition of "circular correlation robustness for naturally derived keys" [ZRE15, GKWY20] in the Half-Gate-based circuit garbling, except that (i) the hash function and the oracle are tweak-free, (ii) a *small* set of public values τ can be added to hash/oracle inputs, and (iii) two syntactically identical operations are not allowed.

Definition 3. *Let $\mathsf{H} : \{0,1\}^\lambda \to \{0,1\}^\lambda$ be a function, $\mathcal{O} : \{0,1\}^\lambda \times \{0,1\} \to \{0,1\}^\lambda$ be an oracle, and $\mathcal{T} \subseteq \{0,1\}^\lambda \setminus \{0^\lambda\}$ be a set of linearly independent strings. Q_i is a* ***natural and non-trivial operation*** *(NNO) if it defines a result $x_i \in \{0,1\}^\lambda$ such that (i) x_i is in one of the following types:*

1. *$x_i \leftarrow \{0,1\}^\lambda$.*
2. *$x_i := x_j \oplus x_k$, where x_j, x_k were defined by two distinct NNOs Q_j, Q_k.*
3. *$x_i := x_j \oplus \tau$, where x_j was defined by an NNO Q_j, and $\tau \in \mathcal{T}$.*
4. *$x_i := \mathsf{H}(x_j)$, where x_j was defined by an NNO Q_j.*
5. *$x_i := \mathcal{O}(x_j, b_j)$, where x_j was defined by an NNO Q_j, and $b_j \in \{0,1\}$.*

and (ii) x_i syntactically differs from any x_j that was defined by an NNO Q_j.

Let $\mathsf{Real}_{\mathsf{H}_S,\Delta}(\cdot)$ be a real-world oracle, which, on input a natural and non-trivial operation, sets $\mathsf{H}(\cdot) = \mathsf{H}_S(\cdot)$ and $\mathcal{O}(\cdot,\cdot) = \mathcal{O}^{\mathsf{ccr}}_{\mathsf{H}_S,\Delta}(\cdot,\cdot)$, executes this operation as per Definition 3, and returns the defined result. The ideal-world oracle $\mathsf{Ideal}_{\mathsf{H}_S}(\cdot)$ is identical to $\mathsf{Real}_{\mathsf{H}_S,\Delta}(\cdot)$, except that it sets $\mathcal{O}(\cdot,\cdot)$ to an oracle that returns a fresh uniform string upon every invocation. We have the following lemma.

Lemma 1 (Circular correlation robustness for restricted queries)**.** *Let $\mathsf{H} : \{0,1\}^\lambda \to \{0,1\}^\lambda$ be a (t, q, ρ, ϵ)-CCR function, and χ be a distribution on $\{0,1\}^\lambda$ with min-entropy at least ρ. Let $\mathsf{H}_S(x) := \mathsf{H}(S \oplus x)$ for $S, x \in \{0,1\}^\lambda$. There exists a polynomial $\mathsf{poly}(\cdot)$ such that, for any PPT distinguisher $\mathcal{D}$ running in time at most $t - \mathsf{poly}(\lambda)$ and querying the oracle $\mathsf{Real}_{\mathsf{H}_S,\Delta}(\cdot)$ with at most q natural and non-trivial operations, it holds that*

$$\left| \Pr_{\substack{\Delta\leftarrow\chi, S\leftarrow\chi,\\ \mathsf{coins}\leftarrow\mathcal{U}}} \left[\mathcal{D}^{\mathsf{Real}_{\mathsf{H}_S,\Delta}(\cdot)}(1^\lambda) = 1\right] - \Pr_{\substack{\mathsf{coins}'\leftarrow\mathcal{U}, S\leftarrow\chi,\\ \mathsf{coins}\leftarrow\mathcal{U}}} \left[\mathcal{D}^{\mathsf{Ideal}_{\mathsf{H}_S}(\cdot)}(1^\lambda) = 1\right] \right| \leq 2\epsilon + \frac{q^2 \cdot 2^{|\mathcal{T}|}}{2^{\lambda+1}},$$

where $\mathcal{U}$ denotes the uniform distribution on bit-strings of an unspecified polynomial length, coins *denotes the random coins used to define the results of type 1 operations, and* coins' *denotes the random coins used to define the results of type 5 operations in $\mathsf{Ideal}_{\mathsf{H}_S}(\cdot)$.*

Proof. We prove this lemma via the following two lemmas.

Lemma 2. *Let $\mathcal{U}$ and* coins *be defined as per Lemma 1. Let $\mathsf{coll}_{\mathsf{real}}$ denote the event that, during the interaction between $\mathcal{D}$ and $\mathsf{Real}_{\mathsf{H}_S,\Delta}(\cdot)$, there exists two defined results $x, x' \in \{0,1\}^\lambda$ such that $x = x'$. It holds that*

$$\Pr_{\substack{\Delta\leftarrow\chi, S\leftarrow\chi,\\ \mathsf{coins}\leftarrow\mathcal{U}}} [\mathsf{coll}_{\mathsf{real}}] \leq \epsilon + \frac{q^2 \cdot 2^{|\mathcal{T}|}}{2^{\lambda+1}}.$$

Proof. For some $f' \in \mathcal{F}_{\lambda,\lambda}$, define a hybrid oracle $\mathsf{Hyb}_{f',\Delta}(\cdot)$ that is identical to $\mathsf{Real}_{\mathsf{H}_S,\Delta}(\cdot)$ except that it executes a type 4 operation by defining $x_i := f'(x_j)$ and executes a type 5 operation by defining $x_i := f'(x_j \oplus \Delta) \oplus b_j \cdot \Delta$. In other words, $\mathsf{Hyb}_{f',\Delta}(\cdot)$ replaces the H_S calls in $\mathsf{Real}_{\mathsf{H}_S,\Delta}(\cdot)$ by f' calls. Let $\mathsf{coll}_{\mathsf{hyb}}$ denote the event that, during the interaction between $\mathcal{D}$ and $\mathsf{Hyb}_{f',\Delta}(\cdot)$, there exist two results $x, x' \in \{0,1\}^\lambda$ defined by two operations such that $x = x'$.

First, we proceed to prove that

$$\left| \Pr_{\substack{\Delta \leftarrow \chi, S \leftarrow \chi, \\ \mathsf{coins} \leftarrow \mathcal{U}}} [\mathsf{coll}_{\mathsf{real}}] - \Pr_{\substack{\Delta \leftarrow \chi, f' \leftarrow \mathcal{F}_{\lambda,\lambda}, \\ \mathsf{coins} \leftarrow \mathcal{U}}} \left[\mathsf{coll}_{\mathsf{hyb}}\right] \right| \le \epsilon. \tag{3}$$

Assume that, for the sake of contradiction, (3) does not hold for some deterministic polynomial-time $\mathcal{D}$. We show that the following adversary $\mathcal{A}$ can use $\mathcal{D}$ to break the CCR property of H:

1. $\mathcal{A}$ internally runs $\mathcal{D}$ and samples $\Delta \leftarrow \chi$. Upon receiving a natural and non-trivial operation Q_i from $\mathcal{D}$, $\mathcal{A}$ proceeds as follows:
   - If Q_i is a type 1/2/3 operation, execute it as required. In particular, the results of type 1 operations are defined by the random coins $\mathsf{coins} \leftarrow \mathcal{U}$.
   - If Q_i is a type 4 operation with input x_j, query the CCR oracle with $(x_j, 0)$, receive a response r, and define the result of Q_i to r.
   - If Q_i is a type 5 operation with input (x_j, b_j), query the CCR oracle with $(x_j \oplus \Delta, 0)$, receive a response r, and define the result of Q_i to $r \oplus b_j \cdot \Delta$.
2. $\mathcal{A}$ terminates when $\mathcal{D}$ terminates. It outputs 1 if there exist two defined results x, x' such that $x = x'$; otherwise it outputs 0.

$\mathcal{A}$ makes at most q queries to the CCR oracle since there are at most q natural and non-trivial operations. $\mathcal{A}$ is polynomial-time since $\mathcal{D}$ is polynomial-time.

If $\mathcal{A}$ is given $\mathcal{O}^{\mathsf{ccr}}_{\mathsf{H},S}(\cdot,\cdot)$ for some $S \leftarrow \chi$, then $\mathcal{D}$ interacts with an emulated oracle that perfectly functions as $\mathsf{Real}_{\mathsf{H}_S,\Delta}(\cdot)$. During the interaction between $\mathcal{D}$ and this emulated oracle, $\mathsf{coll}_{\mathsf{real}}$ occurs with the same probability as that in the case where $\mathcal{D}$ interacts with the real oracle $\mathsf{Real}_{\mathsf{H}_S,\Delta}(\cdot)$. Although $\mathcal{D}$ does not know the key S in the emulation (in contrast to the public S in $\mathsf{Real}_{\mathsf{H}_S,\Delta}(\cdot)$), this does not affect the probability of $\mathsf{coll}_{\mathsf{real}}$ since every operation only depends on the previous responses from the given oracle. One can consider the event $\mathsf{coll}_{\mathsf{real}}$ in the emulated and the real cases under the same probability space, i.e., the same literal values of Δ, (unknown) S, and coins in both two cases. Since $\mathcal{D}$ is deterministic, an induction from the first operation queried by $\mathcal{D}$ shows that the responses are the *same* in the two cases given the same Δ, S, and coins. Depending on these responses, the occurrence of $\mathsf{coll}_{\mathsf{real}}$ is the same. Thus,

$$\Pr_{S \leftarrow \chi} \left[\mathcal{A}^{\mathcal{O}^{\mathsf{ccr}}_{\mathsf{H},S}(\cdot,\cdot)}(1^\lambda) = 1\right] = \Pr_{\substack{\Delta \leftarrow \chi, S \leftarrow \chi, \\ \mathsf{coins} \leftarrow \mathcal{U}}} [\mathsf{coll}_{\mathsf{real}}].$$

If $\mathcal{A}$ is given $f^*(\cdot,\cdot)$ for some $f^* \leftarrow \mathcal{F}_{\lambda+1,\lambda}$, then $\mathcal{D}$ interacts with an emulated oracle that perfectly functions as $\mathsf{Hyb}_{f',\Delta}(\cdot)$ for the random function $f'(\cdot) := f^*(\cdot, 0)$. Likewise, we have

$$\Pr_{f^* \leftarrow \mathcal{F}_{\lambda+1,\lambda}} \left[\mathcal{A}^{f^*(\cdot,\cdot)}(1^\lambda) = 1\right] = \Pr_{\substack{\Delta \leftarrow \chi, f' \leftarrow \mathcal{F}_{\lambda,\lambda}, \\ \mathsf{coins} \leftarrow \mathcal{U}}} \left[\mathsf{coll}_{\mathsf{hyb}}\right].$$

By the contradiction assumption, $\mathcal{A}$ breaks the CCR property of H.

Second, we prove in the following induction that

$$\Pr_{\substack{\Delta \leftarrow \chi, f' \leftarrow \mathcal{F}_{\lambda,\lambda}, \\ \mathsf{coins} \leftarrow \mathcal{U}}} \left[\mathsf{coll}_{\mathsf{hyb}}\right] \leq \frac{q^2 \cdot 2^{|\mathcal{T}|}}{2^{\lambda+1}}. \tag{4}$$

As the base case, (4) trivially holds if there is only one operation. Then, assume that (4) holds for exact $1 \leq q' - 1 \leq q - 1$ operations and consider the q'-th operation that defines a result $x \in \{0,1\}^\lambda$. Let $\mathsf{coll}_{\mathsf{hyb}}(i)$ be the event that, during the interaction between $\mathcal{D}$ and $\mathsf{Hyb}_{f',\Delta}(\cdot)$, there exist two results $x, x' \in \{0,1\}^\lambda$ such that $x = x'$ in the first $i \in \mathbb{N}$ operations. $\mathsf{coll}_{\mathsf{hyb}} = \mathsf{coll}_{\mathsf{hyb}}(q)$ and

$$\begin{aligned}
&\Pr_{\substack{\Delta \leftarrow \chi, f' \leftarrow \mathcal{F}_{\lambda,\lambda}, \\ \mathsf{coins} \leftarrow \mathcal{U}}} \left[\mathsf{coll}_{\mathsf{hyb}}(q')\right] \\
= &\Pr_{\substack{\Delta \leftarrow \chi, f' \leftarrow \mathcal{F}_{\lambda,\lambda}, \\ \mathsf{coins} \leftarrow \mathcal{U}}} \left[\mathsf{coll}_{\mathsf{hyb}}(q') \mid \mathsf{coll}_{\mathsf{hyb}}(q'-1)\right] \cdot \Pr_{\substack{\Delta \leftarrow \chi, f' \leftarrow \mathcal{F}_{\lambda,\lambda}, \\ \mathsf{coins} \leftarrow \mathcal{U}}} \left[\mathsf{coll}_{\mathsf{hyb}}(q'-1)\right] \\
&\quad + \Pr_{\substack{\Delta \leftarrow \chi, f' \leftarrow \mathcal{F}_{\lambda,\lambda}, \\ \mathsf{coins} \leftarrow \mathcal{U}}} \left[\mathsf{coll}_{\mathsf{hyb}}(q') \mid \neg\mathsf{coll}_{\mathsf{hyb}}(q'-1)\right] \cdot \Pr_{\substack{\Delta \leftarrow \chi, f' \leftarrow \mathcal{F}_{\lambda,\lambda}, \\ \mathsf{coins} \leftarrow \mathcal{U}}} \left[\neg\mathsf{coll}_{\mathsf{hyb}}(q'-1)\right] \\
= &\Pr_{\substack{\Delta \leftarrow \chi, f' \leftarrow \mathcal{F}_{\lambda,\lambda}, \\ \mathsf{coins} \leftarrow \mathcal{U}}} \left[\mathsf{coll}_{\mathsf{hyb}}(q'-1)\right] \\
&\quad + \Pr_{\substack{\Delta \leftarrow \chi, f' \leftarrow \mathcal{F}_{\lambda,\lambda}, \\ \mathsf{coins} \leftarrow \mathcal{U}}} \left[\mathsf{coll}_{\mathsf{hyb}}(q') \mid \neg\mathsf{coll}_{\mathsf{hyb}}(q'-1)\right] \cdot \Pr_{\substack{\Delta \leftarrow \chi, f' \leftarrow \mathcal{F}_{\lambda,\lambda}, \\ \mathsf{coins} \leftarrow \mathcal{U}}} \left[\neg\mathsf{coll}_{\mathsf{hyb}}(q'-1)\right] \\
\leq &\Pr_{\substack{\Delta \leftarrow \chi, f' \leftarrow \mathcal{F}_{\lambda,\lambda}, \\ \mathsf{coins} \leftarrow \mathcal{U}}} \left[\mathsf{coll}_{\mathsf{hyb}}(q'-1)\right] + \Pr_{\substack{\Delta \leftarrow \chi, f' \leftarrow \mathcal{F}_{\lambda,\lambda}, \\ \mathsf{coins} \leftarrow \mathcal{U}}} \left[\mathsf{coll}_{\mathsf{hyb}}(q') \mid \neg\mathsf{coll}_{\mathsf{hyb}}(q'-1)\right]
\end{aligned}$$

For each result $x' \in \{0,1\}^\lambda$ defined in the previous $q'-1$ operations, it *syntactically* holds that

$$x \oplus x' = \left(\bigoplus_{i \in \mathcal{I}} x_i\right) \oplus \left(\bigoplus_{j \in \mathcal{J}} f'(x_j)\right) \oplus \left(\bigoplus_{k \in \mathcal{K}} f'(x_k \oplus \Delta)\right) \oplus b \cdot \Delta \oplus \bigoplus_{\tau \in \mathcal{T}} b_\tau \cdot \tau$$

since x, x' can be expressed as a linear combination of operations. Note that the non-triviality of operations ensures that the case "$\mathcal{I} = \mathcal{J} = \mathcal{K} = \varnothing$ and $b = 0$ and each $b_\tau = 0$" is impossible. Conditioned on $\neg\mathsf{coll}_{\mathsf{hyb}}(q'-1)$, there are two cases for such x and x':

- $\mathcal{I} = \mathcal{J} = \mathcal{K} = \varnothing$ and $b = 0$ and at least one $b_\tau \neq 0$. By the definition of $\mathcal{T}$, $x \oplus x' \neq 0$.
- At least one $\mathcal{I}, \mathcal{J}, \mathcal{K} \neq \varnothing$ or $b \neq 0$. This case is identical to that at least one $\mathcal{I}, \mathcal{J}, \mathcal{K} \neq \varnothing$ since $b \neq 0$ implies that $\mathcal{K}$ is not trivially empty, given that (i) $b \neq 0$ implies at least one syntactical term of the form $f'(x_k \oplus \Delta)$ for some $k \in \mathcal{K}$, and (ii) any x_k for $k \in \mathcal{K}$ is distinct conditioned on $\neg\mathsf{coll}_{\mathsf{hyb}}(q'-1)$ (i.e., the additive terms in $\mathcal{K}$ cannot be canceled due to previous collisions).

  The probability of this case equals that of the event that a random string (resulting from at least one nonempty $\mathcal{I}, \mathcal{J}, \mathcal{K}$) lies in the space spanned by linearly independent strings in $\mathcal{T}$ (for any Boolean coefficients $\{b_\tau\}_{\tau \in \mathcal{T}}$ specified by operations). This probability is bounded by $2^{|\mathcal{T}|}/2^\lambda$.

Combining the two cases, we can see that $x = x'$ occurs with probability at most $2^{|\mathcal{T}|}/2^\lambda$. Taking a union bound over all x''s in the previous $q'-1$ operations,

$$\Pr_{\substack{\Delta \leftarrow \chi, f' \leftarrow \mathcal{F}_{\lambda,\lambda}, \\ \mathsf{coins} \leftarrow \mathcal{U}}} \left[\mathsf{coll}_{\mathsf{hyb}}(q') \mid \neg\mathsf{coll}_{\mathsf{hyb}}(q'-1)\right] \leq \frac{(q'-1) \cdot 2^{|\mathcal{T}|}}{2^\lambda}.$$

Using the induction assumption, we have that

$$\begin{aligned}\Pr_{\substack{\Delta\leftarrow\chi, f'\leftarrow\mathcal{F}_{\lambda,\lambda},\\ \mathsf{coins}\leftarrow\mathcal{U}}}\left[\mathsf{coll}_{\mathsf{hyb}}(q')\right] &\leq \Pr_{\substack{\Delta\leftarrow\chi, f'\leftarrow\mathcal{F}_{\lambda,\lambda},\\ \mathsf{coins}\leftarrow\mathcal{U}}}\left[\mathsf{coll}_{\mathsf{hyb}}(q'-1)\right] \\ &\quad + \Pr_{\substack{\Delta\leftarrow\chi, f'\leftarrow\mathcal{F}_{\lambda,\lambda},\\ \mathsf{coins}\leftarrow\mathcal{U}}}\left[\mathsf{coll}_{\mathsf{hyb}}(q') \mid \neg\mathsf{coll}_{\mathsf{hyb}}(q'-1)\right] \\ &\leq \frac{(q'-1)^2\cdot 2^{|\mathcal{T}|}}{2^{\lambda+1}} + \frac{(q'-1)\cdot 2^{|\mathcal{T}|}}{2^{\lambda}} < \frac{q'^2\cdot 2^{|\mathcal{T}|}}{2^{\lambda+1}}\end{aligned}$$

The above induction shows that (4) holds. This lemma follows from (3) and (4). □

Lemma 3. *Let $\mathcal{U}$, coins, and coins' be defined as per Lemma 1, and $\mathsf{coll}_{\mathsf{real}}$ be the event defined as per Lemma 2. It holds that*

$$\left|\Pr_{\substack{\Delta\leftarrow\chi,S\leftarrow\chi,\\ \mathsf{coins}\leftarrow\mathcal{U}}}\left[\mathcal{D}^{\mathsf{Real}_{\mathsf{H}_S,\Delta}(\cdot)}(1^\lambda)=1\right] - \Pr_{\substack{\mathsf{coins}'\leftarrow\mathcal{U},S\leftarrow\chi,\\ \mathsf{coins}\leftarrow\mathcal{U}}}\left[\mathcal{D}^{\mathsf{Ideal}_{\mathsf{H}_S}(\cdot)}(1^\lambda)=1\right]\right| \leq \epsilon + \Pr_{\substack{\Delta\leftarrow\chi,S\leftarrow\chi,\\ \mathsf{coins}\leftarrow\mathcal{U}}}\left[\mathsf{coll}_{\mathsf{real}}\right]$$

Proof. Assume that this lemma does not hold for some deterministic polynomial-time $\mathcal{D}$. The following adversary $\mathcal{A}$ can use $\mathcal{D}$ to break the CCR property of H.

1. $\mathcal{A}$ internally runs $\mathcal{D}$ and samples $S\leftarrow\chi$. Upon receiving a natural and non-trivial operation Q_i from $\mathcal{D}$, $\mathcal{A}$ proceeds as follows:
   - If Q_i is a type 1/2/3/4 operation, execute it as required. In particular, the results of type 1 operations are defined by the random coins $\mathsf{coins}\leftarrow\mathcal{U}$.
   - If Q_i is a type 5 operation with input (x_j, b_j) and x_j was not included in previous operations, query the CCR oracle with $(x_j\oplus S, b_j)$, receive a response r, and define the result of Q_i to r. If Q_i is a type 5 operation with input (x_j, b_j) and x_j was included in some previous operation, sample $s\leftarrow\{0,1\}^\lambda$ and define the result of Q_i to s.
2. $\mathcal{A}$ terminates when $\mathcal{D}$ terminates and outputs whatever $\mathcal{D}$ outputs.

$\mathcal{A}$ makes at most q queries to the CCR oracle since there are at most q natural and non-trivial operations. $\mathcal{A}$ is polynomial-time since $\mathcal{D}$ is polynomial-time.

Consider the case where $\mathcal{A}$ is given $\mathcal{O}^{\mathsf{ccr}}_{\mathsf{H},\Delta}(\cdot,\cdot)$ for some $\Delta\leftarrow\chi$. We analyze the event $\mathsf{coll}_{\mathsf{real}}$ during the interaction between $\mathcal{D}$ and either the oracle emulated by $\mathcal{A}$ or the oracle $\mathsf{Real}_{\mathsf{H}_S,\Delta}(\cdot)$ under the same probability space. That is, the randomness Δ, S, and coins take the *same* values in both cases, respectively (note that, in the emulated case, $\mathsf{coll}_{\mathsf{real}}$ does not depend on the randomness s in the presence of any colliding $x_j\in\{0,1\}^\lambda$; otherwise, there will be a circularity).

For any given deterministic $\mathcal{D}$, the initial operation is the same regardless of the oracle given to it. Assume that the event $\mathsf{coll}_{\mathsf{real}}$ does not occur. By the assumption, an induction based on the same Δ, S, and coins shows that every defined result and every queried operation are the *same* in the two cases. Thus, for any fixed Δ, S, and coins, the event $(\mathcal{D}^{\mathsf{Real}_{\mathsf{H}_S,\Delta}(\cdot)}(1^\lambda)=1)\wedge\neg\mathsf{coll}_{\mathsf{real}}$ occurs if and only if the event $(\mathcal{D}^{\mathcal{A}(\cdot)}(1^\lambda)=1)\wedge\neg\mathsf{coll}_{\mathsf{real}}$ occurs. Since, in the emulation, $\mathcal{A}$ outputs 1 if and only if the distinguisher $\mathcal{D}$ outputs 1, we can see that the latter event occurs if and only if $(\mathcal{A}^{\mathcal{O}^{\mathsf{ccr}}_{\mathsf{H},\Delta}(\cdot,\cdot)}(1^\lambda)=1)\wedge\neg\mathsf{coll}_{\mathsf{real}}$ occurs. It follows from the well-known Difference Lemma that

$$\left|\Pr_{\Delta\leftarrow\chi}\left[\mathcal{A}^{\mathcal{O}^{\mathsf{ccr}}_{\mathsf{H},\Delta}(\cdot,\cdot)}(1^\lambda)=1\right] - \Pr_{\substack{\Delta\leftarrow\chi,S\leftarrow\chi,\\ \mathsf{coins}\leftarrow\mathcal{U}}}\left[\mathcal{D}^{\mathsf{Real}_{\mathsf{H}_S,\Delta}(\cdot)}(1^\lambda)=1\right]\right| \leq \Pr_{\substack{\Delta\leftarrow\chi,S\leftarrow\chi,\\ \mathsf{coins}\leftarrow\mathcal{U}}}\left[\mathsf{coll}_{\mathsf{real}}\right]. \tag{5}$$

If $\mathcal{A}$ is given $f^*(\cdot,\cdot)$ for some $f^* \leftarrow \mathcal{F}_{\lambda+1,\lambda}$, we have

$$\Pr_{f^* \leftarrow \mathcal{F}_{\lambda+1,\lambda}}\left[\mathcal{A}^{f^*(\cdot,\cdot)}(1^\lambda) = 1\right] = \Pr_{\substack{\mathsf{coins}' \leftarrow \mathcal{U}, S \leftarrow \chi, \\ \mathsf{coins} \leftarrow \mathcal{U}}}\left[\mathcal{D}^{\mathsf{Ideal}_{\mathsf{H}_S}(\cdot)}(1^\lambda) = 1\right] \tag{6}$$

for the following reasons. First, if there is no collision in the execution of type 5 operations, the uniqueness of x_j's ensures that the output of the random function f^* (when $\mathcal{D}$ is given an oracle emulated by $\mathcal{A}$) are as uniform and pairwise independent as those uniformly sampled (when $\mathcal{D}$ is given $\mathsf{Ideal}_{\mathsf{H}_S}(\cdot)$). Second, if there exists at least one collision, the result defined by the first operation with the conflicting x_j is still uniform as per f^*. For the subsequent results with respect to the same x_j, their uniformness and pairwise independence are ensured by the sampling of $\mathcal{A}$.

Using the contradiction assumption and (5), (6), we have

$$\begin{aligned}
&\left|\Pr_{\Delta \leftarrow \chi}\left[\mathcal{A}^{\mathcal{O}^{\mathsf{ccr}}_{\mathsf{H},\Delta}(\cdot,\cdot)}(1^\lambda) = 1\right] - \Pr_{f^* \leftarrow \mathcal{F}_{\lambda+1,\lambda}}\left[\mathcal{A}^{f^*(\cdot,\cdot)}(1^\lambda) = 1\right]\right| \\
= &\left|\left(\Pr_{\Delta \leftarrow \chi}\left[\mathcal{A}^{\mathcal{O}^{\mathsf{ccr}}_{\mathsf{H},\Delta}(\cdot,\cdot)}(1^\lambda) = 1\right] - \Pr_{\substack{\Delta \leftarrow \chi, S \leftarrow \chi, \\ \mathsf{coins} \leftarrow \mathcal{U}}}\left[\mathcal{D}^{\mathsf{Real}_{\mathsf{H}_S,\Delta}(\cdot)}(1^\lambda) = 1\right]\right)\right. \\
&\left. + \left(\Pr_{\substack{\Delta \leftarrow \chi, S \leftarrow \chi, \\ \mathsf{coins} \leftarrow \mathcal{U}}}\left[\mathcal{D}^{\mathsf{Real}_{\mathsf{H}_S,\Delta}(\cdot)}(1^\lambda) = 1\right] - \Pr_{\substack{\mathsf{coins}' \leftarrow \mathcal{U}, S \leftarrow \chi, \\ \mathsf{coins} \leftarrow \mathcal{U}}}\left[\mathcal{D}^{\mathsf{Ideal}_{\mathsf{H}_S}(\cdot)}(1^\lambda) = 1\right]\right)\right| \\
\geq &\left|\left|\Pr_{\substack{\Delta \leftarrow \chi, S \leftarrow \chi, \\ \mathsf{coins} \leftarrow \mathcal{U}}}\left[\mathcal{D}^{\mathsf{Real}_{\mathsf{H}_S,\Delta}(\cdot)}(1^\lambda) = 1\right] - \Pr_{\substack{\mathsf{coins}' \leftarrow \mathcal{U}, S \leftarrow \chi, \\ \mathsf{coins} \leftarrow \mathcal{U}}}\left[\mathcal{D}^{\mathsf{Ideal}_{\mathsf{H}_S}(\cdot)}(1^\lambda) = 1\right]\right|\right. \\
&\left. - \left|\Pr_{\Delta \leftarrow \chi}\left[\mathcal{A}^{\mathcal{O}^{\mathsf{ccr}}_{\mathsf{H},\Delta}(\cdot,\cdot)}(1^\lambda) = 1\right] - \Pr_{\substack{\Delta \leftarrow \chi, S \leftarrow \chi, \\ \mathsf{coins} \leftarrow \mathcal{U}}}\left[\mathcal{D}^{\mathsf{Real}_{\mathsf{H}_S,\Delta}(\cdot)}(1^\lambda) = 1\right]\right|\right| > \epsilon,
\end{aligned}$$

which contradicts the CCR property of H. ☐

This lemma follows from Lemma 2 and Lemma 3. ☐

B Security Proofs and Detailed Protocol in Section 4

B.1 Proof of Theorem 1

Theorem 1. *Given random permutation $\pi : \mathbb{F}_{2^\lambda} \to \mathbb{F}_{2^\lambda}$, efficiently computable linear orthomorphism $\sigma : \mathbb{F}_{2^\lambda} \to \mathbb{F}_{2^\lambda}$ with efficiently computable σ^{-1}, $\sigma'(x) := \sigma(x) \oplus x$, and σ'^{-1} (Footnote 1), and hash function $\mathsf{H}(x) := \pi(\sigma(x)) \oplus \sigma(x)$, protocol Π_{spCOT} (Figure 4) UC-realizes functionality $\mathcal{F}_{\mathsf{spCOT}}$ (Figure 2) against any semi-honest adversary in the $\mathcal{F}_{\mathsf{COT}}$-hybrid model and the RPM.*

Proof. We consider polynomially many concurrent **Extend** executions, each of which is implicitly assigned the same session ID but a unique sub-session ID. We focus on the setting where there is exact one corrupted party[3] and first analyze its correctness.

Correctness analysis. The correctness relies on the two cGGM full-evaluation algorithms. The outputs of $\mathsf{cGGM.PuncFullEval}$ and $\mathsf{cGGM.FullEval}$ are identical everywhere except the punctured point. Moreover, the correlation at the punctured point follows from Corollary 1.

Corrupted P_0. In the one-time **Initialize** execution:

1. Upon receiving the first (init) from $\mathcal{A}$ to $\mathcal{F}_{\mathsf{COT}}$, $\mathcal{S}$ waits for $\mathcal{A}$ to choose the global key Δ. Then, $\mathcal{S}$ sends (init) and Δ to $\mathcal{F}_{\mathsf{spCOT}}$.

Then, in each **Extend** execution:

2. Upon receiving (extend, n) from $\mathcal{A}$ to $\mathcal{F}_{\mathsf{COT}}$, $\mathcal{S}$ waits for $\mathcal{A}$ to choose the COT transcript $(\mathsf{K}[r_1], \dots, \mathsf{K}[r_n])$.

3. $\mathcal{S}$ receives $(c_1, \dots, c_n)$ from $\mathcal{A}$ and computes
$$(\mathbf{v}, \dots) := \mathsf{cGGM.FullEval}(\Delta, \mathsf{K}[r_1] \oplus c_1).$$
Then, $\mathcal{S}$ sends (extend, N) and $\mathbf{v}$ to $\mathcal{F}_{\mathsf{spCOT}}$.

The simulation is perfect. Without loss of generality, assume a deterministic environment $\mathcal{Z}$. The view of $\mathcal{A}$ only includes Δ in the one-time **Initialize** execution and $(\mathsf{K}[r_1], \dots, \mathsf{K}[r_n])$ in each concurrent **Extend** execution. These transcripts, which are essentially chosen by $\mathcal{Z}$, must take the same literal values in the two worlds since $\mathcal{Z}$ is deterministic, and the transcripts can depend on no transcript from the honest P_1.

In the concurrent executions, the output $(\mathbf{u}, \mathbf{w})$'s are identically distributed in the two worlds conditioned on some literal values of the one-time initialized Δ and all $(\mathsf{K}[r_1], \dots, \mathsf{K}[r_n])$'s. First, in each **Extend** execution, the real-world index α of the only non-zero entry of $\mathbf{u}$ is as execution-wise uniform as the ideal-world one due to the execution-wise uniform COT choice bits for the honest P_1. Thus, the $\mathbf{u}$'s in the concurrent executions have the same distribution in the two worlds. Second, in each **Extend** execution additionally conditioned on some literal values of $\mathbf{u}$, the vector $\mathbf{w}$ takes the same value $\mathbf{w} = \mathbf{v}^* \oplus \mathbf{u} \cdot \Delta$ in the two worlds. In the real world, the well-formed transcript $(c_1, \dots, c_n)$ of the semi-honest P_0 and the correctness of Π_{spCOT} guarantee that the literal value of $\mathbf{v}^*$ in this equality comes from $(\mathbf{v}^*, \dots) := \mathsf{cGGM.FullEval}(\Delta, \mathsf{K}[r_1] \oplus c_1)$. Note that the randomness

[3] We omit the case where both parties are honest. The reason is that, in most eventual applications (e.g., generic 2PC/MPC, zero-knowledge proofs, and private set intersection) of $\mathcal{F}_{\mathsf{spCOT}}$ (or $\mathcal{F}_{\mathsf{spsVOLE}}$), the semi-honest security in this case trivially holds if secure channel is available (see [HL10, Section 2.3]) and the consistency $\mathbf{w} = \mathbf{v} + \mathbf{u} \cdot \Delta$ holds. Besides this consistency, the distribution of $((\mathbf{v}, \Delta), (\mathbf{u}, \mathbf{w}))$ will not skew the joint output distribution in these applications. Hence, we need not consider the distribution of $((\mathbf{v}, \Delta), (\mathbf{u}, \mathbf{w}))$ when both parties are honest.

c_1 chosen by $\mathcal{Z}$ is identical in the two worlds due to the deterministic $\mathcal{Z}$. Thus, the ideal-world $\mathbf{v}$ is the same as the real-world $\mathbf{v}^*$ conditioned on the fixed transcripts, and so is the $\mathbf{w}$.

Corrupted P_1. In the ideal world, $\mathcal{S}$ emulates the random permutation π and its inverse π^{-1} throughout the lifespan of the distinguishing game. These two oracles are used for *all* concurrent **Extend** executions. During the distinguishing game in either world, let $\mathcal{Q}_\pi$ denote an *ordered* list of the adversary's query/answer pairs to/from the random permutation or its inverse (note that $(x, y) \in \mathcal{Q}_\pi$ means that the adversary learns $\pi(x) = y$, regardless of whether it queried $\pi(x)$ or $\pi^{-1}(y)$), and $\mathcal{Q}_\mathcal{O}$ denote an *ordered* list of the adversary's query/answer pairs to/from the "oracle" emulated by the non-corrupted machine(s) (i.e., the honest P_0 plus the subroutine $\mathcal{F}_{\mathsf{COT}}$ in the real world, or the simulator $\mathcal{S}$ in the ideal world). One can think that, for each **Extend** execution, $\mathcal{Q}_\mathcal{O}$ records the query $\{(r_i, \mathsf{M}[r_i])\}_{i\in[1,n]}$ (i.e., the COT output chosen by the corrupted P_1) with its answer $(c_1, \dots, c_n)$ (i.e., the transcript sent by the honest P_0).

$\mathcal{S}$ has access to $\mathcal{Q}_\pi$ and $\mathcal{Q}_\mathcal{O}$ in the ideal world, and the transcript viewed by $\mathcal{Z}$ in either world includes the spCOT global key Δ (the honest P_0's output in the one-time **Initialize** execution), the honest P_0's output $\mathbf{v}$ in each **Extend** execution, and the two ordered lists $\mathcal{Q}_\pi$ and $\mathcal{Q}_\mathcal{O}$. We first focus on the *ordered* transcript $\mathcal{Q} := (\Delta, \mathcal{Q}_\pi, \mathcal{Q}_\mathcal{O})$ (where the pairs of $\mathcal{Q}_\pi$ and $\mathcal{Q}_\mathcal{O}$ are interleaved; we will consider this order but use this notation for the simplicity of exposition) and discuss the output $\mathbf{v}$ in each **Extend** execution later. Let $p = p(\lambda) := |\mathcal{Q}_\pi|$ and $q = q(\lambda) := |\mathcal{Q}_\mathcal{O}|$ (note that q is the number of concurrent **Extend** executions).

$\mathcal{S}$ works as follows. In the one-time **Initialize** execution:

1. Upon receiving the first (init) from $\mathcal{A}$ to $\mathcal{F}_{\mathsf{COT}}$, $\mathcal{S}$ sends (init) to $\mathcal{F}_{\mathsf{spCOT}}$.

Then, in each **Extend** execution:

2. Upon receiving (extend, n) from $\mathcal{A}$ to $\mathcal{F}_{\mathsf{COT}}$, $\mathcal{S}$ waits for $\mathcal{A}$ to choose the COT transcript $((r_1, \dots, r_n), (\mathsf{M}[r_1], \dots, \mathsf{M}[r_n]))$.
3. $\mathcal{S}$ sends random $(c_1, \dots, c_n) \leftarrow \mathbb{F}_{2^\lambda}^n$ to $\mathcal{A}$ and computes

$$\mathbf{u} := \mathbf{unit}_{\mathbb{F}_2}(N, \overline{r}_1 \dots \overline{r}_n, 1),$$
$$\mathbf{w} := \mathsf{cGGM.PuncFullEval}(\overline{r}_1 \dots \overline{r}_n, \{\mathsf{M}[r_i] \oplus c_i\}_{i\in[1,n]}).$$

   Then, $\mathcal{S}$ sends (extend, N) and $(\mathbf{u}, \mathbf{w})$ to $\mathcal{F}_{\mathsf{spCOT}}$.

- **Global-key query.** $\mathcal{S}$ performs the global-key query in the following cases:
  - query_1: $\mathcal{Z}$ queries the random permutation π with x or its inverse π^{-1} with y. For every $(\{(r_i, \mathsf{M}[r_i])\}_{i\in[1,n]}, \{c_i\}_{i\in[1,n]}) \in \mathcal{Q}_\mathcal{O}$, $\mathcal{S}$ does:
    1. $\mathcal{S}$ computes

$$\begin{aligned} \{z_i\}_{i\in[1,n]} &:= \mathsf{cGGM.OffPath}(\overline{r}_1 \dots \overline{r}_n, \{\mathsf{M}[r_i] \oplus c_i\}_{i\in[1,n]}), \\ \forall j \in [2, n] : w_j &:= \oplus_{i\in[1,j-1]} z_i, \end{aligned} \tag{7}$$

    where $\mathsf{cGGM.OffPath}$ is a macro such that, on input a path α and n sums $\{K_i^{\overline{\alpha}_i}\}_{i\in[1,n]}$ used in $\mathsf{cGGM.PuncFullEval}$ to define an n-level correlated GGM tree except the n on-path nodes, it outputs the n off-path nodes for the path α. This macro ensures that, for $j \in [1, n]$,

$$z_j = (\mathsf{M}[r_j] \oplus c_j) \oplus \text{“other } r_j\text{-side nodes on the } j\text{-th level defined by the off-path nodes } \{z_i\}_{i\in[1,j-1]}\text{”}. \tag{8}$$

2. For every $j \in [2, n]$, $\mathcal{S}$ extracts $\Delta'_1 \in \mathbb{F}_{2^\lambda}$ and sends $(\mathsf{guess}, \Delta'_1)$ to $\mathcal{F}_{\mathsf{spCOT}}$ if $\mathcal{Z}$ queries π with x, or extracts $\Delta'_2 \in \mathbb{F}_{2^\lambda}$ and $(\mathsf{guess}, \Delta'_2)$ to $\mathcal{F}_{\mathsf{spCOT}}$ if $\mathcal{Z}$ queries π^{-1} with y, where

$$\Delta'_1 := \sigma^{-1}(x) \oplus w_j, \qquad \Delta'_2 := \begin{cases} \sigma^{-1}(y \oplus z_j) \oplus w_j & \text{, if } r_j = 0 \\ \sigma'^{-1}(y \oplus z_j) \oplus w_j & \text{, if } r_j = 1 \end{cases} \tag{9}$$

Note that $\sigma'(x) := \sigma(x) \oplus x$ is a permutation as $\sigma : \mathbb{F}_{2^\lambda} \to \mathbb{F}_{2^\lambda}$ is an orthomorphism, and its inverse σ'^{-1} should be well-defined.

– query_2: A new pair $(\{(r_i, \mathsf{M}[r_i])\}_{i\in[1,n]}, \{c_i\}_{i\in[1,n]})$ is added to $\mathcal{Q}_{\mathcal{O}}$. Using this pair, $\mathcal{S}$ computes $\{z_i\}_{i\in[1,n]}$ and $\{w_j\}_{j\in[2,n]}$ as per (7). Then, for every $(x, y) \in \mathcal{Q}_\pi$ and every $j \in [2, n]$, $\mathcal{S}$ uses (9) to extract Δ'_1 *and* Δ'_2, and sends $(\mathsf{guess}, \Delta'_1)$ *and* $(\mathsf{guess}, \Delta'_2)$ to $\mathcal{F}_{\mathsf{spCOT}}$.

In either case, if $\mathcal{S}$ receives $(\mathsf{success})$ from $\mathcal{F}_{\mathsf{spCOT}}$ for some guess Δ, $\mathcal{S}$ will program the random permutation π and its inverse π^{-1} such that, for the $\{z_i\}_{i\in[1,n]}$ and $\{w_j\}_{j\in[2,n]}$ computed from every pair in $\mathcal{Q}_{\mathcal{O}}$ up to this time and every $j \in [2, n]$,

$$\pi(\sigma(\Delta \oplus w_j)) = \sigma(\Delta \oplus w_j) \oplus r_j \cdot (\Delta \oplus w_j) \oplus z_j, \tag{10}$$

which must hold in the real world due to the construction of H and the definition of z_j and w_j. After the programming, $\mathcal{S}$ uses the global key Δ extracted from $\mathcal{F}_{\mathsf{spCOT}}$ to emulate $\mathcal{F}_{\mathsf{COT}}$ and $(c_1, \ldots, c_n)$ for any incomplete **Extend** execution by following the specification in Π_{spCOT}.

We use the H-coefficient technique [Pat09, CS14] to prove that this ideal world is computationally indistinguishable from the real one. Without loss of generality, we consider a deterministic environment $\mathcal{Z}$ and the transcript $\mathcal{Q}$, which is a part of the joint distribution of the view of $\mathcal{A}$ and the output of the honest P_0 (note that the remaining part of this joint distribution is the $\mathbf{v}$'s in all **Extend** executions, and its consistency with $\mathcal{Q}$ will be checked in the end).

We borrow the summary of the H-coefficient technique from [GKWY20]. Let $\mathcal{T}$ be the set of *attainable* transcripts for some information-theoretic distinguisher, and let $\Pr_{\mathsf{real}}[\cdot]$ and $\Pr_{\mathsf{ideal}}[\cdot]$ be the probabilities in the real and ideal worlds, respectively. The H-coefficient technique divides $\mathcal{T}$ into a "bad" subset $\mathcal{T}_{\mathsf{bad}}$ and a "good" subset $\mathcal{T}_{\mathsf{good}} := \mathcal{T} \setminus \mathcal{T}_{\mathsf{bad}}$, and shows that

$$\Pr_{\mathsf{ideal}}[\mathcal{Q} \in \mathcal{T}_{\mathsf{bad}}] \le \epsilon_1, \quad \forall \mathcal{Q} \in \mathcal{T}_{\mathsf{good}} : \frac{\Pr_{\mathsf{real}}[\mathcal{Q}]}{\Pr_{\mathsf{ideal}}[\mathcal{Q}]} \ge 1 - \epsilon_2.$$

Then, the advantage of the distinguisher is at most $\epsilon_1 + \epsilon_2$.

A transcript $\mathcal{Q} = (\Delta, \mathcal{Q}_\pi, \mathcal{Q}_{\mathcal{O}})$ is **bad** if it falls into one of the following cases:

- bad_1. There exist two distinct transcript pairs $((\{(r_i, \mathsf{M}[r_i])\}_{i\in[1,n]}, \{c_i\}_{i\in[1,n]}), j) \in \mathcal{Q}_{\mathcal{O}} \times [2, n]$ and $((\{(r'_i, \mathsf{M}[r'_i])\}_{i\in[1,n]}, \{c'_i\}_{i\in[1,n]}), j') \in \mathcal{Q}_{\mathcal{O}} \times [2, n]$ such that

$$\Big(w_j = w'_{j'}\Big) \vee \Big(\sigma(w_j) \oplus r_j \cdot (\Delta \oplus w_j) \oplus z_j = \sigma(w'_{j'}) \oplus r'_{j'} \cdot (\Delta \oplus w'_{j'}) \oplus z'_{j'}\Big),$$

  where z_j, w_j, $z'_{j'}$, and $w'_{j'}$ are defined as per (7). This case captures the collision between the queries to the random permutation or its inverse.

- bad_2. There exists a $((\{(r_i, \mathsf{M}[r_i])\}_{i\in[1,n]}, \{c_i\}_{i\in[1,n]}), j) \in \mathcal{Q}_{\mathcal{O}} \times [2, n]$ such that

$$\Big((\sigma(\Delta \oplus w_j), \ldots) \in \mathcal{Q}_\pi \vee (\ldots, \sigma(\Delta \oplus w_j) \oplus r_j \cdot (\Delta \oplus w_j) \oplus z_j) \in \mathcal{Q}_\pi\Big)$$
$$\wedge \Big(\pi(\sigma(\Delta \oplus w_j)) \ne \sigma(\Delta \oplus w_j) \oplus r_j \cdot (\Delta \oplus w_j) \oplus z_j\Big),$$

  where z_j and w_j are defined as per (7). This case captures the inconsistency in the already defined entries of the random permutation and its inverse.

Consider bad_1 in the ideal world. For any two distinct pairs in $\mathcal{Q}_{\mathcal{O}} \times [2, n]$,

$$\begin{aligned}
&\Pr_{\mathsf{ideal}}\left[\left(w_j = w'_{j'}\right) \vee \left(\sigma(w_j) \oplus r_j \cdot (\Delta \oplus w_j) \oplus z_j = \sigma(w'_{j'}) \oplus r'_{j'} \cdot (\Delta \oplus w'_{j'}) \oplus z'_{j'}\right)\right] \\
\leq\ &\Pr_{\mathsf{ideal}}\left[w_j = w'_{j'}\right] + \Pr_{\mathsf{ideal}}\left[\sigma(w_j) \oplus r_j \cdot (\Delta \oplus w_j) \oplus z_j = \sigma(w'_{j'}) \oplus r'_{j'} \cdot (\Delta \oplus w'_{j'}) \oplus z'_{j'}\right] \\
=\ &2^{-\lambda} + 2^{-\lambda} = 2 \cdot 2^{-\lambda},
\end{aligned}$$

where the equality follows from that (i) the c_i's are uniform and pairwise independent in the ideal world, and (ii) each z_i is as uniform as c_i due to (8). Taking a union bound over all distinct pairs in $\mathcal{Q}_{\mathcal{O}} \times [2, n]$, it holds that

$$\Pr_{\mathsf{ideal}}\left[\mathsf{bad}_1\right] \leq \frac{q^2(n-1)^2}{2^\lambda}. \tag{11}$$

As for bad_2, it is sufficient to bound $\Pr_{\mathsf{ideal}}\left[\mathsf{bad}_2 \mid \neg\mathsf{bad}_1\right]$ using the following three sub-cases. In the *first* sub-case where

$$\begin{aligned}
&\forall((\{(r_i, \mathsf{M}[r_i])\}_{i\in[1,n]}, \{c_i\}_{i\in[1,n]}), j) \in \mathcal{Q}_{\mathcal{O}} \times [2, n] : \\
&\quad (\sigma(\Delta \oplus w_j), \dots) \notin \mathcal{Q}_\pi \wedge (\dots, \sigma(\Delta \oplus w_j) \oplus r_j \cdot (\Delta \oplus w_j) \oplus z_j) \notin \mathcal{Q}_\pi,
\end{aligned}$$

it is clear that bad_2 will not happen.

Otherwise, in the following two complement sub-cases, $\mathcal{S}$ can extract the global key Δ (which is a part of the transcript $\mathcal{Q}$) by solving either $x = \sigma(\Delta \oplus w_j)$ or $y = \sigma(\Delta \oplus w_j) \oplus r_j \cdot (\Delta \oplus w_j) \oplus z_j$ as per (9). Then, $\mathcal{S}$ will use Δ to program π and π^{-1} to keep (10). If the programming succeeds, bad_2 will not happen since (10) holds for every pair in $\mathcal{Q}_{\mathcal{O}}$ and every $j \in [2, n]$. After the programming, the consistency trivially holds for any future pair in $\mathcal{Q}_{\mathcal{O}}$ and $j \in [2, n]$ since Δ is known.

In the *second* sub-case where a new query to π or π^{-1} triggers the programming (i.e., query_1 case), the programming must succeed conditioned on $\neg\mathsf{bad}_1$. The condition $\neg\mathsf{bad}_1$ ensures that, for every $((\{(r_i, \mathsf{M}[r_i])\}_{i\in[1,n]}, \{c_i\}_{i\in[1,n]}), j) \in \mathcal{Q}_{\mathcal{O}} \times [2, n]$, the pre-image $x := \sigma(\Delta \oplus w_j)$ and the image $y := \sigma(\Delta \oplus w_j) \oplus r_j \cdot (\Delta \oplus w_j) \oplus z_j$ to be programmed do not incur collision. More importantly, these two entries are not determined by the query/answer pairs in $\mathcal{Q}_\pi$ up to this time. Otherwise, $\exists(x, \dots) \in \mathcal{Q}_\pi$ or $\exists(\dots, y) \in \mathcal{Q}_\pi$ ahead of the current query such that the programming must have happened for the same $\mathcal{Q}_{\mathcal{O}}$ (i.e., contradicting that the programming is triggered by the current query). Hence, all pairs in $\mathcal{Q}_{\mathcal{O}}$ can be programmed as per (10). The answer to the current query is determined by the programmed result. The surely successful programming makes bad_2 impossible.

Consider the *third* sub-case where the programming arises from a new pair added to $\mathcal{Q}_{\mathcal{O}}$ (i.e., query_2 case). Assume that the global key Δ to be used in the programming is extracted from this pair, an existing $(x^*, \dots) \in \mathcal{Q}_\pi$ or $(\dots, y^*) \in \mathcal{Q}_\pi$, and $j^* \in [2, n]$. Note that either $(x^*, \dots) \in \mathcal{Q}_\pi$ or $(\dots, y^*) \in \mathcal{Q}_\pi$ has been given to $\mathcal{Z}$ and *cannot* be programmed. We bound the probability of this *undesired* sub-case. In the ideal world,

$$\begin{aligned}
&\Pr_{\mathsf{ideal}}\left[(x^* = \sigma(\Delta \oplus w_{j^*})) \vee (y^* = \sigma(\Delta \oplus w_{j^*}) \oplus r_{j^*} \cdot (\Delta \oplus w_{j^*}) \oplus z_{j^*})\right] \\
\leq\ &\Pr_{\mathsf{ideal}}\left[x^* = \sigma(\Delta \oplus w_{j^*})\right] + \Pr_{\mathsf{ideal}}\left[y^* = \sigma(\Delta \oplus w_{j^*}) \oplus r_{j^*} \cdot (\Delta \oplus w_{j^*}) \oplus z_{j^*}\right] \\
=\ &\Pr_{\mathsf{ideal}}\left[w_{j^*} = \sigma^{-1}(x^*) \oplus \Delta\right] + \Pr_{\mathsf{ideal}}\left[z_{j^*} = y^* \oplus \sigma(\Delta \oplus w_{j^*}) \oplus r_{j^*} \cdot (\Delta \oplus w_{j^*})\right] \\
=\ &2^{-\lambda} + 2^{-\lambda} = 2 \cdot 2^{-\lambda},
\end{aligned}$$

where z_{j^*} and w_{j^*} are computed from the newly added pair, and the probability is taken over the uniform c_{j^*}. Taking a union bound over $\mathcal{Q}_\pi \times \mathcal{Q}_{\mathcal{O}} \times [2, n]$, we can see that this sub-case happens with probability at most $2pq(n-1)/2^\lambda$.

Combining the above three sub-cases, we have that

$$\Pr_{\mathsf{ideal}}[\mathsf{bad}_2 \mid \neg\mathsf{bad}_1] \leq \frac{2pq(n-1)}{2^\lambda}. \tag{12}$$

Using (11) and (12), it holds that

$$\begin{aligned}
\Pr_{\mathsf{ideal}}[\mathcal{Q} \in \mathcal{T}_{\mathsf{bad}}] &= \Pr_{\mathsf{ideal}}[\mathsf{bad}_1 \vee \mathsf{bad}_2] \\
&= \Pr_{\mathsf{ideal}}[\mathsf{bad}_1] + \Pr_{\mathsf{ideal}}[\mathsf{bad}_2] - \Pr_{\mathsf{ideal}}[\mathsf{bad}_1 \wedge \mathsf{bad}_2] \\
&= \Pr_{\mathsf{ideal}}[\mathsf{bad}_1] + \Pr_{\mathsf{ideal}}[\mathsf{bad}_2 \mid \neg\mathsf{bad}_1] \cdot \Pr_{\mathsf{ideal}}[\neg\mathsf{bad}_1] \\
&\leq \Pr_{\mathsf{ideal}}[\mathsf{bad}_1] + \Pr_{\mathsf{ideal}}[\mathsf{bad}_2 \mid \neg\mathsf{bad}_1] \\
&\leq \frac{q^2(n-1)^2 + 2pq(n-1)}{2^\lambda}.
\end{aligned}$$

We proceed to bound the ratio of $\Pr_{\mathsf{real}}[\mathcal{Q}]$ to $\Pr_{\mathsf{ideal}}[\mathcal{Q}]$ for some fixed **good** transcript $\mathcal{Q} = (\Delta, \mathcal{Q}_\pi, \mathcal{Q}_\mathcal{O})$. Let $\mathcal{L} \subseteq \mathcal{Q}_\mathcal{O} \times [2, n]$ such that

$$\begin{aligned}
&\forall((\{(r_i, \mathsf{M}[r_i])\}_{i\in[1,n]}, \{c_i\}_{i\in[1,n]}), j) \in \mathcal{L} : \\
&\qquad (\sigma(\Delta \oplus w_j), \dots) \in \mathcal{Q}_\pi \vee (\dots, \sigma(\Delta \oplus w_j) \oplus r_j \cdot (\Delta \oplus w_j) \oplus z_j) \in \mathcal{Q}_\pi.
\end{aligned}$$

Conditioned on $\neg(\mathsf{bad}_1 \vee \mathsf{bad}_2)$ implicit in good transcripts, $\mathcal{L}$ induces a subset $\mathcal{Q}_\pi(\mathcal{L}) \subseteq \mathcal{Q}_\pi$. There exists a *bijective* correspondence between a $(x, y) \in \mathcal{Q}_\pi(\mathcal{L})$ and the (z_j, w_j) computed as per (7) from a pair in $\mathcal{L}$ such that $x = \sigma(\Delta \oplus w_j)$ and $y = \sigma(\Delta \oplus w_j) \oplus r_j \cdot (\Delta \oplus w_j) \oplus z_j$. It is clear that $|\mathcal{L}| = |\mathcal{Q}_\pi(\mathcal{L})| \leq \min(|\mathcal{Q}_\pi|, |\mathcal{Q}_\mathcal{O}| \cdot (n-1)) = \min(p, q(n-1))$.

For $\mathcal{X} \subseteq \mathcal{Q}_\pi$, let $\pi \sim \mathcal{X}$ be the event that the permutation π is consistent with the queries and answers in $\mathcal{X}$, i.e., $\forall (x, y) \in \mathcal{X} : \pi(x) = y$. Let $c(j) \sim \mathcal{Q}_\mathcal{O}$ be the event that the $\{c_i\}_{i\in[1,j]}$ in each **Extend** execution take the literal values in $\mathcal{Q}_\mathcal{O}$. For $\mathcal{Y} \subseteq \mathcal{Q}_\mathcal{O} \times [2, n]$, let $(\pi, \Delta) \sim \mathcal{Y}$ be the *real-world* event that π and the global key Δ are consistent with the pairs in $\mathcal{Y}$, i.e.,

$$\begin{aligned}
&\forall((\{(r_i, \mathsf{M}[r_i])\}_{i\in[1,n]}, \{c_i\}_{i\in[1,n]}), j) \in \mathcal{Y} : \\
&\qquad \pi(\sigma(\Delta \oplus w_j)) \oplus \sigma(\Delta \oplus w_j) \oplus r_j \cdot (\Delta \oplus w_j) = z_j
\end{aligned}$$

for z_j and w_j computed from (7). Let $(a)_b := a \cdots (a - b + 1)$ be falling factorial.

In the real world, π is a random permutation without programming, and

$$\begin{aligned}
\Pr_{\mathsf{real}}[\mathcal{Q}] &= \Pr_{\mathsf{real}}[c(1) \sim \mathcal{Q}_\mathcal{O} \wedge (\pi, \Delta) \sim \mathcal{Q}_\mathcal{O} \times [2, n] \mid \pi \sim \mathcal{Q}_\pi \wedge \Delta] \\
&\quad \cdot \Pr_{\mathsf{real}}[\pi \sim \mathcal{Q}_\pi \wedge \Delta] \\
&= \Pr_{\mathsf{real}}[c(1) \sim \mathcal{Q}_\mathcal{O} \wedge (\pi, \Delta) \sim \mathcal{Q}_\mathcal{O} \times [2, n] \mid \pi \sim \mathcal{Q}_\pi \wedge \Delta] \\
&\quad \cdot \Pr_{\mathsf{real}}[\pi \sim \mathcal{Q}_\pi] \cdot \Pr_{\mathsf{real}}[\Delta] \\
&= \Pr_{\mathsf{real}}\left[(\pi, \Delta) \sim \mathcal{Q}_\mathcal{O} \times [2, n] \;\middle|\; \begin{array}{c} \pi \sim \mathcal{Q}_\pi \wedge \Delta \\ \wedge\, c(1) \sim \mathcal{Q}_\mathcal{O} \end{array}\right] \\
&\quad \cdot \Pr_{\mathsf{real}}[c(1) \sim \mathcal{Q}_\mathcal{O} \mid \pi \sim \mathcal{Q}_\pi \wedge \Delta] \cdot \Pr_{\mathsf{real}}[\pi \sim \mathcal{Q}_\pi] \cdot \Pr_{\mathsf{real}}[\Delta] \\
&= \Pr_{\mathsf{real}}\left[\begin{array}{l} (\pi, \Delta) \sim \mathcal{L} \\ \quad \wedge\, (\pi, \Delta) \sim (\mathcal{Q}_\mathcal{O} \times [2, n]) \setminus \mathcal{L} \end{array} \;\middle|\; \begin{array}{c} \pi \sim \mathcal{Q}_\pi \wedge \Delta \\ \wedge\, c(1) \sim \mathcal{Q}_\mathcal{O} \end{array}\right] \\
&\quad \cdot \Pr_{\mathsf{real}}[c(1) \sim \mathcal{Q}_\mathcal{O}] \cdot \Pr_{\mathsf{real}}[\pi \sim \mathcal{Q}_\pi] \cdot \Pr_{\mathsf{real}}[\Delta] \\
&= \Pr_{\mathsf{real}}\left[(\pi, \Delta) \sim (\mathcal{Q}_\mathcal{O} \times [2, n]) \setminus \mathcal{L} \;\middle|\; \begin{array}{c} \pi \sim \mathcal{Q}_\pi \wedge \Delta \\ \wedge\, c(1) \sim \mathcal{Q}_\mathcal{O} \wedge (\pi, \Delta) \sim \mathcal{L} \end{array}\right] \\
&\quad \cdot \Pr_{\mathsf{real}}\left[(\pi, \Delta) \sim \mathcal{L} \;\middle|\; \begin{array}{c} \pi \sim \mathcal{Q}_\pi \wedge \Delta \\ \wedge\, c(1) \sim \mathcal{Q}_\mathcal{O} \end{array}\right] \cdot \frac{1}{(2^\lambda)^q} \cdot \frac{1}{(2^\lambda)_p} \cdot \frac{1}{2^\lambda}
\end{aligned}$$

Conditioned on $\neg\mathsf{bad}_2$ in the good transcript, the definition of $\mathcal{L}$ implies that

$$\Pr_{\mathsf{real}}\left[(\pi,\Delta)\sim\mathcal{L}\;\middle|\;\begin{array}{c}\pi\sim\mathcal{Q}_\pi\wedge\Delta\\ \wedge\, c(1)\sim\mathcal{Q}_\mathcal{O}\end{array}\right]=1.$$

Moreover, $\neg\mathsf{bad}_1$ ensures that, for each $((\{(r_i,\mathsf{M}[r_i])\}_{i\in[1,n]},\{c_i\}_{i\in[1,n]}),j)\in\mathcal{Q}_\mathcal{O}\times[2,n]$, the permutation entries $\pi(\sigma(\Delta\oplus w_j))$ and $\pi^{-1}(\sigma(\Delta\oplus w_j)\oplus r_j\cdot(\Delta\oplus w_j)\oplus z_j)$ are not determined by other pairs in $\mathcal{Q}_\mathcal{O}\times[2,n]$. For those pairs in $(\mathcal{Q}_\mathcal{O}\times[2,n])\setminus\mathcal{L}$, these two entries are even not determined by all queries in $\mathcal{Q}_\pi$ due to $\neg\mathsf{bad}_2$ and the definition of $\mathcal{L}$. By the randomness of π, we have

$$\Pr_{\mathsf{real}}\left[(\pi,\Delta)\sim(\mathcal{Q}_\mathcal{O}\times[2,n])\setminus\mathcal{L}\;\middle|\;\begin{array}{c}\pi\sim\mathcal{Q}_\pi\wedge\Delta\\ \wedge\, c(1)\sim\mathcal{Q}_\mathcal{O}\wedge(\pi,\Delta)\sim\mathcal{L}\end{array}\right]=\frac{1}{(2^\lambda-p)_{q(n-1)-|\mathcal{L}|}}.$$

Using these results, we have

$$\begin{aligned}\Pr_{\mathsf{real}}[\mathcal{Q}]&=\frac{1}{(2^\lambda-p)_{q(n-1)-|\mathcal{L}|}\cdot(2^\lambda)_p\cdot(2^\lambda)^{q+1}}\\&=\frac{1}{(2^\lambda)_{p+q(n-1)-|\mathcal{L}|}\cdot(2^\lambda)^{q+1}}.\end{aligned}\tag{13}$$

In the ideal world, it holds that

$$\begin{aligned}\Pr_{\mathsf{ideal}}[\mathcal{Q}]&=\Pr_{\mathsf{ideal}}[\pi\sim\mathcal{Q}_\pi\mid c(n)\sim\mathcal{Q}_\mathcal{O}\wedge\Delta]\cdot\Pr_{\mathsf{ideal}}[c(n)\sim\mathcal{Q}_\mathcal{O}\wedge\Delta]\\&=\Pr_{\mathsf{ideal}}\left[\begin{array}{c}\pi\sim\mathcal{Q}_\pi(\mathcal{L})\\ \wedge\,\pi\sim\mathcal{Q}_\pi\setminus\mathcal{Q}_\pi(\mathcal{L})\end{array}\;\middle|\;c(n)\sim\mathcal{Q}_\mathcal{O}\wedge\Delta\right]\\&\quad\cdot\Pr_{\mathsf{ideal}}[c(n)\sim\mathcal{Q}_\mathcal{O}]\cdot\Pr_{\mathsf{ideal}}[\Delta]\\&=\Pr_{\mathsf{ideal}}\left[\pi\sim\mathcal{Q}_\pi\setminus\mathcal{Q}_\pi(\mathcal{L})\;\middle|\;\begin{array}{c}c(n)\sim\mathcal{Q}_\mathcal{O}\wedge\Delta\\ \wedge\,\pi\sim\mathcal{Q}_\pi(\mathcal{L})\end{array}\right]\\&\quad\cdot\Pr_{\mathsf{ideal}}[\pi\sim\mathcal{Q}_\pi(\mathcal{L})\mid c(n)\sim\mathcal{Q}_\mathcal{O}\wedge\Delta]\cdot\frac{1}{(2^\lambda)^{qn}\cdot2^\lambda}.\end{aligned}$$

By the programmed π and π^{-1} conditioned on $\neg\mathsf{bad}_2$, the defined $\mathcal{Q}_\pi(\mathcal{L})$ implies that

$$\Pr_{\mathsf{ideal}}[\pi\sim\mathcal{Q}_\pi(\mathcal{L})\mid c(n)\sim\mathcal{Q}_\mathcal{O}\wedge\Delta]=1,$$

$$\Pr_{\mathsf{ideal}}\left[\pi\sim\mathcal{Q}_\pi\setminus\mathcal{Q}_\pi(\mathcal{L})\;\middle|\;\begin{array}{c}c(n)\sim\mathcal{Q}_\mathcal{O}\wedge\Delta\\ \wedge\,\pi\sim\mathcal{Q}_\pi(\mathcal{L})\end{array}\right]=\frac{1}{(2^\lambda)_{p-|\mathcal{Q}_\pi(\mathcal{L})|}}=\frac{1}{(2^\lambda)_{p-|\mathcal{L}|}}.$$

Using the above results and (13), we have

$$\begin{aligned}\Pr_{\mathsf{ideal}}[\mathcal{Q}]&=\frac{1}{(2^\lambda)_{p-|\mathcal{L}|}}\cdot\frac{1}{(2^\lambda)^{qn}\cdot2^\lambda}\\&\le\frac{1}{(2^\lambda-q(n-1))_{p-|\mathcal{L}|}}\cdot\frac{1}{(2^\lambda)_{q(n-1)}\cdot(2^\lambda)^{q+1}}\\&=\frac{1}{(2^\lambda)_{p+q(n-1)-|\mathcal{L}|}\cdot(2^\lambda)^{q+1}}=\Pr_{\mathsf{real}}[\mathcal{Q}].\end{aligned}$$

By the H-coefficient technique, $\mathcal{Z}$ can only distinguish the two worlds with advantage at most $(q^2(n-1)^2+2pq(n-1))/2^\lambda$ if it is only given the transcript $\mathcal{Q}=(\Delta,\mathcal{Q}_\pi,\mathcal{Q}_\mathcal{O})$. In fact, this is the advantage that $\mathcal{Z}$ can distinguish the two worlds since the output $\mathbf{v}$'s in all **Extend** executions must be consistent with $\mathcal{Q}$ (or rather, its resulting Δ and the $(\mathbf{u},\mathbf{w})$'s) in the real world (due to the correctness of Π_{spCOT}) and the ideal world. Thus, this theorem holds. $\square$

Protocol $\Pi_{\mathsf{spsVOLE-cGGM}}$

Parameters: Field $\mathbb{F}$ and its extension field $\mathbb{K}$ with $|\mathbb{K}| \geq 2^\lambda$.

Initialize: This procedure is executed only once.

1. P_0 and P_1 send (init) to $\mathcal{F}_{\mathsf{sVOLE}}$, which returns $\Delta \in \mathbb{K}$ to P_0. P_0 outputs Δ.

Extend: This procedure can be executed many times. P_0 and P_1 input $N = 2^n$ and use cGGM (c.f. Figure 3) for n and $\mathbb{K}$.

2. P_0 and P_1 send $(\mathsf{extend}, n+1)$ to $\mathcal{F}_{\mathsf{sVOLE}}$, which returns $(\mathsf{K}[s_0], \ldots, \mathsf{K}[s_n]) \in \mathbb{K}^{n+1}$ to P_0 and $((s_0, \ldots, s_n), (\mathsf{M}[s_0], \ldots, \mathsf{M}[s_n])) \in \mathbb{F}^{n+1} \times \mathbb{K}^{n+1}$ to P_1 such that $\mathsf{M}[s_i] = \mathsf{K}[s_i] + s_i \cdot \Delta$ for $i \in [0, n]$.
3. P_1 samples $\beta \leftarrow \mathbb{F}^*$ and $(r_1, \ldots, r_n) \leftarrow \mathbb{F}_2^n$, sets $\mathsf{M}[\beta] := \mathsf{M}[s_0]$ and $\mathsf{M}[r_i] := r_i \cdot \mathsf{M}[\beta] - \mathsf{M}[s_i]$ for $i \in [1, n]$, and sends $(d_0, d_1, \ldots, d_n) := (s_0, s_1, \ldots, s_n) - (1, r_1, \ldots, r_n) \cdot \beta \in \mathbb{F}^{n+1}$ to P_0.
   P_0 sets $\mathsf{K}[\beta] := \mathsf{K}[s_0] + d_0 \cdot \Delta$ and $\mathsf{K}[r_i] := -\mathsf{K}[s_i] - d_i \cdot \Delta$ for $i \in [1, n]$ such that $\mathsf{M}[\beta] = \mathsf{K}[\beta] + \beta \cdot \Delta$ and $\mathsf{M}[r_i] = \mathsf{K}[r_i] + r_i \cdot \mathsf{K}[\beta]$ for $i \in [1, n]$.
4. P_0 samples $c_1 \leftarrow \mathbb{K}$ and sets $k := -\mathsf{K}[r_1] + c_1$,
   $$(\mathbf{v}, \{K_i^0\}_{i \in [1,n]}) := \mathsf{cGGM.FullEval}(\mathsf{K}[\beta], k),$$
   and $c_i := \mathsf{K}[r_i] + K_i^0$ for $i \in [2, n]$. P_0 sends $(c_1, \ldots, c_n)$ to P_1.
5. P_1 sets $\alpha = \alpha_1 \ldots \alpha_n := \overline{r}_1 \ldots \overline{r}_n \in [0, N)$, $K_i^{\overline{\alpha}_i} := (-1)^{r_i} \cdot (-\mathsf{M}[r_i] + c_i)$ for $i \in [1, n]$, and
   $$\mathbf{u} := \mathbf{unit}_{\mathbb{F}}(N, \alpha, \beta),$$
   $$\mathbf{w} := \mathsf{cGGM.PuncFullEval}(\alpha, \{K_i^{\overline{\alpha}_i}\}_{i \in [1,n]}) + \mathbf{unit}_{\mathbb{F}}(N, \alpha, \mathsf{M}[\beta]).$$
6. P_0 outputs $\mathbf{v}$ and P_1 outputs $(\mathbf{u}, \mathbf{w})$.

Figure 14: cGGM-based single-point sVOLE protocol in the $\mathcal{F}_{\mathsf{sVOLE}}$-hybrid model.

B.2 Single-point sVOLE from Correlated GGM Tree

Our cGGM-based single-point sVOLE protocol is presented in Figure 14. The intuition behind this protocol has been summarized in Section 3.1. In this appendix, we will prove its security in Theorem 7 and present two communication optimizations, followed by the complexity analysis of the optimized protocol.

Theorem 7. *Given random permutation $\pi : \mathbb{K} \to \mathbb{K}$, efficiently computable linear orthomorphism $\sigma : \mathbb{K} \to \mathbb{K}$ with efficiently computable σ^{-1}, $\sigma'(x) := \sigma(x) - x$, and σ'^{-1} (Footnote 1), and hash function $\mathsf{H}(x) := \pi(\sigma(x)) + \sigma(x)$, protocol $\Pi_{\mathsf{spsVOLE-cGGM}}$ (Figure 14) UC-realizes functionality $\mathcal{F}_{\mathsf{spsVOLE}}$ (Figure 2) against any semi-honest adversary in the $\mathcal{F}_{\mathsf{sVOLE}}$-hybrid model and the RPM, for $|\mathbb{K}| \geq 2^\lambda$.*

Proof. This proof is similar to that in Appendix B.1. We consider polynomially many concurrent **Extend** executions, each of which is implicitly assigned the same session ID but a unique sub-session ID. We also focus on the setting where there is exact one corrupted party for the same reason in Appendix B.1.

Correctness analysis. The correctness resorts to the property of cGGM tree.

Corrupted P_0. In the one-time **Initialize** execution:

1. Upon receiving the first (init) from $\mathcal{A}$ to $\mathcal{F}_{\mathsf{sVOLE}}$, $\mathcal{S}$ waits for $\mathcal{A}$ to choose the global key Δ. Then, $\mathcal{S}$ sends (init) and Δ to $\mathcal{F}_{\mathsf{spsVOLE}}$.

Then, in each **Extend** execution:

2. Upon receiving $(\mathsf{extend}, n+1)$ from $\mathcal{A}$ to $\mathcal{F}_{\mathsf{sVOLE}}$, $\mathcal{S}$ waits for $\mathcal{A}$ to choose the sVOLE transcript $(\mathsf{K}[s_0], \dots, \mathsf{K}[s_n])$.

3. $\mathcal{S}$ sends random $(d_0, d_1, \dots, d_n) \leftarrow \mathbb{F}^{n+1}$ to $\mathcal{A}$.

4. $\mathcal{S}$ receives $(c_1, \dots, c_n)$ from $\mathcal{A}$ and computes

$$\mathsf{K}[\beta] := \mathsf{K}[s_0] + d_0 \cdot \Delta,$$
$$\mathsf{K}[r_1] := -\mathsf{K}[s_1] - d_1 \cdot \Delta,$$
$$(\mathbf{v}, \dots) := \mathsf{cGGM.FullEval}(\mathsf{K}[\beta], -\mathsf{K}[r_1] + c_1).$$

   Then, $\mathcal{S}$ sends (extend, N) and $\mathbf{v}$ to $\mathcal{F}_{\mathsf{spsVOLE}}$.

The simulation is perfect. Without loss of generality, assume a deterministic environment $\mathcal{Z}$. Note that the view of $\mathcal{A}$ includes Δ in the one-time **Initialize** execution, and $(\mathsf{K}[s_0], \dots, \mathsf{K}[s_n])$ and $(d_0, d_1, \dots, d_n)$ in each concurrent **Extend** execution. Δ is the first transcript chosen by $\mathcal{Z}$ so it is identical in the two worlds. The transcript $(d_0, d_1, \dots, d_n)$ in each concurrent execution is uniform and independent in the real world (due to the execution-wise uniform one-time pad) and the ideal world. The transcript $(\mathsf{K}[s_0], \dots, \mathsf{K}[s_n])$'s in all **Extend** executions are chosen by $\mathcal{Z}$ and can only depend on the $(d_0, d_1, \dots, d_n)$'s therein. Thus, they will take the same literal values conditioned on some fixed $(d_0, d_1, \dots, d_n)$'s in the concurrent executions.

In the concurrent executions, the output $(\mathbf{u}, \mathbf{w})$'s are identically distributed in the two worlds conditioned on some literal values of the one-time initialized Δ and all $(\mathsf{K}[s_0], \dots, \mathsf{K}[s_n])$'s' and $(d_0, d_1, \dots, d_n)$'s. First, in each **Extend** execution, the real-world index α of the only non-zero entry of $\mathbf{u}$ is as execution-wise uniform as the ideal-world one due to the execution-wise uniform COT choice bits for the honest P_1, and the value β is identically distributed in the two worlds. As a result, the $\mathbf{u}$'s in the concurrent executions have the same distribution in the two worlds. Second, in each **Extend** execution additionally conditioned on some literal values of $\mathbf{u}$, the vector $\mathbf{w}$ takes the same value $\mathbf{w} = \mathbf{v}^* \oplus \mathbf{u} \cdot \Delta$ in the two worlds. In the real world, the well-formed transcript $(c_1, \dots, c_n)$ of the semi-honest P_0 and the correctness of $\Pi_{\mathsf{spsVOLE-cGGM}}$ guarantee that the literal value of $\mathbf{v}^*$ in this equality comes from

$$(\mathbf{v}^*, \dots) := \mathsf{cGGM.FullEval}(\mathsf{K}[s_0] + d_0 \cdot \Delta, \mathsf{K}[s_1] + d_1 \cdot \Delta + c_1).$$

Note that the randomness c_1 chosen by $\mathcal{Z}$ is identical in the two worlds due to the deterministic $\mathcal{Z}$. Thus, the ideal-world $\mathbf{v}$ is the same as the real-world $\mathbf{v}^*$ conditioned on the fixed transcripts, yielding the same $\mathbf{w}$ in the two worlds.

Corrupted P_1. In the ideal world, $\mathcal{S}$ emulates the random permutation π and its inverse π^{-1} throughout the lifespan of the distinguishing game. These two oracles are used for *all* concurrent **Extend** executions. During the distinguishing game in either world, let $\mathcal{Q}_\pi$ denote an *ordered* list of the adversary's query/answer pairs to/from the random permutation or its inverse (note that $(x, y) \in \mathcal{Q}_\pi$ means that the adversary learns $\pi(x) = y$, regardless of whether it queried $\pi(x)$ or $\pi^{-1}(y)$), and $\mathcal{Q}_\mathcal{O}$ denote an *ordered* list of the adversary's query/answer pairs to/from the "oracle" emulated by the non-corrupted machine(s) (i.e., the honest P_0 plus the subroutine $\mathcal{F}_{\mathsf{sVOLE}}$ in the real world, or the simulator $\mathcal{S}$ in the ideal world). One can think that, for each **Extend** execution, $\mathcal{Q}_\mathcal{O}$ records the query $((\beta, \mathsf{M}[\beta]), \{(r_i, \mathsf{M}[r_i])\}_{i \in [1,n]})$ (i.e., the effective sVOLE output chosen by the corrupted P_1) with its answer $(c_1, \dots, c_n)$ (i.e., the transcript sent by the honest P_0). $\mathcal{S}$ has access to $\mathcal{Q}_\pi$ and $\mathcal{Q}_\mathcal{O}$ in the ideal world.

$\mathcal{S}$ works as follows. In the one-time **Initialize** execution:

1. Upon receiving the first (init) from $\mathcal{A}$ to $\mathcal{F}_{\mathsf{sVOLE}}$, $\mathcal{S}$ sends (init) to $\mathcal{F}_{\mathsf{spsVOLE}}$.

Then, in each **Extend** execution:

2. Upon receiving $(\mathsf{extend}, n+1)$ from $\mathcal{A}$ to $\mathcal{F}_{\mathsf{sVOLE}}$, $\mathcal{S}$ waits for $\mathcal{A}$ to choose the sVOLE transcript $((s_0, \ldots, s_n), (\mathsf{M}[s_0], \ldots, \mathsf{M}[s_n]))$.
3. $\mathcal{S}$ receives $(d_0, d_1, \ldots, d_n)$ from $\mathcal{A}$ and extracts $((\beta, \mathsf{M}[\beta]), \{(r_i, \mathsf{M}[r_i])\}_{i\in[1,n]})$.
4. $\mathcal{S}$ sends random $(c_1, \ldots, c_n) \leftarrow \mathbb{K}^n$ to $\mathcal{A}$ and computes

$$\mathbf{u} := \mathbf{unit}_{\mathbb{F}}(N, \overline{r}_1 \ldots \overline{r}_n, \beta),$$
$$\mathbf{w} := \mathsf{cGGM.PuncFullEval}(\overline{r}_1 \ldots \overline{r}_n, \{(-1)^{r_i} \cdot (-\mathsf{M}[r_i] + c_i)\}_{i\in[1,n]}) + \mathbf{unit}_{\mathbb{F}}(N, \overline{r}_1 \ldots \overline{r}_n, \mathsf{M}[\beta]).$$

Then, $\mathcal{S}$ sends (extend, N) and $(\mathbf{u}, \mathbf{w})$ to $\mathcal{F}_{\mathsf{spsVOLE}}$.

- **Global-key query.** $\mathcal{S}$ performs the global-key query in the following cases:
  - query_1: $\mathcal{Z}$ queries the random permutation π with x or its inverse π^{-1} with y. For every $((\beta, \mathsf{M}[\beta]), \{(r_i, \mathsf{M}[r_i])\}_{i\in[1,n]}, \{c_i\}_{i\in[1,n]}) \in \mathcal{Q}_{\mathcal{O}}$, $\mathcal{S}$ does:
    1. $\mathcal{S}$ computes

$$\{z_i\}_{i\in[1,n]} := \mathsf{cGGM.OffPath}(\overline{r}_1 \ldots \overline{r}_n, \{(-1)^{r_i} \cdot (-\mathsf{M}[r_i] + c_i)\}_{i\in[1,n]}),$$
$$\forall j \in [2, n] : w_j := \textstyle\sum_{i\in[1,j-1]} z_i,$$

where $\mathsf{cGGM.OffPath}$ is a macro such that, on input a path α and n sums $\{K_i^{\overline{\alpha}_i}\}_{i\in[1,n]}$ used in $\mathsf{cGGM.PuncFullEval}$ to define an n-level correlated GGM tree except the n on-path nodes, it outputs the n off-path nodes for the path α. This macro ensures that, for $j \in [1, n]$,

$$z_j = (-1)^{r_j} \cdot (-\mathsf{M}[r_j] + c_j) - \text{“other } r_j\text{-side nodes on the } j\text{-th level defined by the off-path nodes } \{z_i\}_{i\in[1,j-1]}\text{”}.$$

    2. For every $j \in [2, n]$, $\mathcal{S}$ extracts $\mathsf{K}[\beta]'_1 \in \mathbb{K}$ and sends $(\mathsf{guess}, \beta^{-1} \cdot (\mathsf{M}[\beta] - \mathsf{K}[\beta]'_1))$ to $\mathcal{F}_{\mathsf{spsVOLE}}$ if $\mathcal{Z}$ queries π with x, or extracts $\mathsf{K}[\beta]'_2 \in \mathbb{K}$ and $(\mathsf{guess}, \beta^{-1} \cdot (\mathsf{M}[\beta] - \mathsf{K}[\beta]'_2))$ to $\mathcal{F}_{\mathsf{spsVOLE}}$ if $\mathcal{Z}$ queries π^{-1} with y, where

$$\mathsf{K}[\beta]'_1 := \sigma^{-1}(x) + w_j, \qquad \mathsf{K}[\beta]'_2 := \begin{cases} \sigma^{-1}(-y + z_j) + w_j & \text{, if } r_j = 0 \\ \sigma'^{-1}(-y - z_j) + w_j & \text{, if } r_j = 1 \end{cases}$$

Note that $\sigma'(x) := \sigma(x) - x$ is a permutation as $\sigma : \mathbb{K} \to \mathbb{K}$ is an orthomorphism, and its inverse σ'^{-1} should be well-defined.
  - query_2: A new pair $((\beta, \mathsf{M}[\beta]), \{(r_i, \mathsf{M}[r_i])\}_{i\in[1,n]}, \{c_i\}_{i\in[1,n]})$ is added to $\mathcal{Q}_{\mathcal{O}}$. Using this pair, $\mathcal{S}$ computes $\{z_i\}_{i\in[1,n]}$ and $\{w_j\}_{j\in[2,n]}$ as per (7). Then, for every $(x, y) \in \mathcal{Q}_\pi$ and every $j \in [2, n]$, $\mathcal{S}$ extracts $\mathsf{K}[\beta]'_1$ *and* $\mathsf{K}[\beta]'_2$, and sends $(\mathsf{guess}, \beta^{-1} \cdot (\mathsf{M}[\beta] - \mathsf{K}[\beta]'_1))$ *and* $(\mathsf{guess}, \beta^{-1} \cdot (\mathsf{M}[\beta] - \mathsf{K}[\beta]'_2))$ to $\mathcal{F}_{\mathsf{spsVOLE}}$.

  In either case, if $\mathcal{S}$ receives $(\mathsf{success})$ from $\mathcal{F}_{\mathsf{spsVOLE}}$ for some guess Δ, $\mathcal{S}$ will program the random permutation π and its inverse π^{-1} such that, for the offset $\mathsf{K}[\beta] := \mathsf{M}[\beta] - \beta \cdot \Delta$, the $\{z_i\}_{i\in[1,n]}$ and $\{w_j\}_{j\in[2,n]}$ computed from every pair in $\mathcal{Q}_{\mathcal{O}}$ up to this time, and every $j \in [2, n]$,

$$\pi(\sigma(\mathsf{K}[\beta] - w_j)) = \sigma(\mathsf{K}[\beta] - w_j) + (-1)^{r_j} \cdot (-r_j \cdot (\mathsf{K}[\beta] - w_j) + z_j),$$

  which must hold in the real world due to the construction of H and the definition of z_j and w_j. After the programming, $\mathcal{S}$ uses the global key Δ extracted from $\mathcal{F}_{\mathsf{spsVOLE}}$ to emulate $\mathcal{F}_{\mathsf{sVOLE}}$ and $(c_1, \ldots, c_n)$ in any incomplete **Extend** execution by following the specification in $\Pi_{\mathsf{spsVOLE-cGGM}}$.

The ideal world is computationally indistinguishable from the real one. The advantage of a deterministic environment $\mathcal{Z}$ can be bounded via the H-coefficient technique as in Appendix B.1. A remarkable difference is that $\mathcal{S}$ query $\mathcal{F}_{\mathsf{spsVOLE}}$ with $\Delta \in \mathbb{K}$ by first extracting $\mathsf{K}[\beta]$ from the transcript $\mathcal{Q}$ and then guessing $\Delta := \beta^{-1} \cdot (\mathsf{M}[\beta] - \mathsf{K}[\beta])$. For any given $\beta \neq 0$ (due to the semi-honest P_1) and $\mathsf{M}[\beta]$, there is a bijective correspondence between $\mathsf{K}[\beta]$ and Δ. We omit the similar probability analysis and claim that $\mathcal{Z}$ can only distinguish the two worlds with advantage at most $(q^2(n-1)^2 + 2pq(n-1))/|\mathbb{K}|$, where $p = |\mathcal{Q}_\pi|$ and $q = |\mathcal{Q}_\mathcal{O}|$. □

Communication optimization. We have the following two optimizations:

- The PRF-based optimization of our single-point COT protocol Π_{spCOT} also applies to t concurrent **Extend** executions (e.g., in sVOLE extension).
- If $\mathbb{F}$ is a large field (i.e., $|\mathbb{F}| \geq 2^\rho$ for some statistical security parameter $\rho \in \mathbb{N}$), the two parties can directly use a random precomputed sVOLE tuple for $\mathsf{M}[\beta] = \mathsf{K}[\beta] + \beta \cdot \Delta$ as $\beta \in \mathbb{F}$ is nonzero with overwhelming probability. This optimization has been used in [WYKW21].

Complexity analysis. Consider the complexity per execution when the PRF-based optimization is used in t concurrent **Extend** executions. In the $\mathcal{F}_{\mathsf{sVOLE}}$-hybrid model, $\Pi_{\mathsf{spsVOLE-cGGM}}$ needs $n+1$ precomputed sVOLE tuples. P_0 sends $(n-1) \cdot \log|\mathbb{K}| + \frac{\lambda}{t}$ bits while P_1 sends $(n+1) \cdot \log|\mathbb{F}|$ bits. The computation per party comes from the tree expansion with N RP calls.

Casted into the $(\mathcal{F}_{\mathsf{COT}}, \mathcal{F}_{\mathsf{sVOLE}})$-hybrid model for comparison, the single-point sVOLE protocols [BCG+19a, WYKW21] use n precomputed COT tuples and one precomputed sVOLE tuple. These protocols use a COT tuple to emulate a string OT, where the sender P_0 sends 2λ bits to the receiver P_1. The outgoing communication of P_0 is $2(n-1) \cdot \lambda + 3 \cdot \log|\mathbb{K}|$ bits, and the outgoing communication of P_1 is $\log|\mathbb{F}|$ bits. Each party performs $2N$ RP calls. The overall communication of $\Pi_{\mathsf{spsVOLE-cGGM}}$ is roughly identical to that of the prior protocols if $|\mathbb{F}| \approx |\mathbb{K}|$. However, for $|\mathbb{F}| \ll |\mathbb{K}|$, $\Pi_{\mathsf{spsVOLE-cGGM}}$ still halves the communication of the prior protocols. The reduction in computation is 50%.

Note that our protocol $\Pi_{\mathsf{spsVOLE-cGGM}}$ only invokes $\mathcal{F}_{\mathsf{sVOLE}}$, which requires less setup cost than $\mathcal{F}_{\mathsf{COT}}$ and $\mathcal{F}_{\mathsf{sVOLE}}$ in the prior protocols.

B.3 Proof of Theorem 2

Theorem 2. *Given CCR function $\mathsf{H} : \mathbb{F}_{2^\lambda} \to \mathbb{F}_{2^\lambda}$, function $\mathsf{Convert}_{\mathbb{K}} : \mathbb{F}_{2^\lambda} \to \mathbb{K}$, and keyed hash function $\mathsf{H}_S(x) := \mathsf{H}(S \oplus x)$ with some key $S \leftarrow \mathbb{F}_{2^\lambda}$, protocol $\Pi_{\mathsf{spsVOLE-pcGGM}}$ (Figure 6) UC-realizes functionality $\mathcal{F}_{\mathsf{spsVOLE}}$ (Figure 2) without global-key queries against any semi-honest adversary in the $(\mathcal{F}_{\mathsf{COT}}, \mathcal{F}_{\mathsf{sVOLE}})$-hybrid model.*

Proof. We consider polynomially many concurrent **Extend** executions, each of which is implicitly assigned the same session ID but a unique sub-session ID.

Correctness analysis. By definition, the outputs of $\mathsf{pcGGM.PuncFullEval}$ and $\mathsf{pcGGM.FullEval}$ are identical at every non-punctured point. For the punctured point α, it holds that

$$\begin{aligned}
\mathbf{w}^{(\alpha)} &= \psi + \mathsf{M}[\beta] - \textstyle\sum_{j \in [0,2^n), j \neq \alpha} \mathbf{w}^{(j)} \\
&= \psi + \mathsf{M}[\beta] - \textstyle\sum_{j \in [0,2^n), j \neq \alpha} \mathbf{v}^{(j)} \\
&= \textstyle\sum_{j \in [0,2^n)} \mathbf{v}^{(j)} - \mathsf{K}[\beta] + \mathsf{M}[\beta] - \textstyle\sum_{j \in [0,2^n), j \neq \alpha} \mathbf{v}^{(j)} \\
&= \mathbf{v}^{(\alpha)} + \beta \cdot \Gamma
\end{aligned}$$

as required by single-point sVOLE.

Corrupted P_0. In the one-time **Initialize** execution:

1. Upon receiving the first (init) from $\mathcal{A}$ to $\mathcal{F}_{\mathsf{COT}}$, $\mathcal{S}$ waits for $\mathcal{A}$ to choose the COT global key Δ.
2. Upon receiving the first (init) from $\mathcal{A}$ to $\mathcal{F}_{\mathsf{sVOLE}}$, $\mathcal{S}$ waits for $\mathcal{A}$ to choose the sVOLE global key Γ. Then, $\mathcal{S}$ sends (init) and Γ to $\mathcal{F}_{\mathsf{spsVOLE}}$.

Then, in each **Extend** execution:

3. Upon receiving (extend, n) from $\mathcal{A}$ to $\mathcal{F}_{\mathsf{COT}}$, $\mathcal{S}$ waits for $\mathcal{A}$ to choose the COT transcript $(\mathsf{K}[r_1], \dots, \mathsf{K}[r_n])$.
4. Upon receiving $(\mathsf{extend}, 1)$ from $\mathcal{A}$ to $\mathcal{F}_{\mathsf{sVOLE}}$, $\mathcal{S}$ waits for $\mathcal{A}$ to choose the sVOLE transcript $\mathsf{K}[s]$.
5. $\mathcal{S}$ sends random $d \leftarrow \mathbb{F}$ to $\mathcal{A}$.
6. $\mathcal{S}$ receives $(c_1, \dots, c_{n-1}, \mu, c_n^0, c_n^1, \psi)$ from $\mathcal{A}$ and computes

$$(\mathbf{v}, \dots) := \mathsf{pcGGM.FullEval}(\Delta, \mathsf{K}[r_1] \oplus c_1).$$

   Then, $\mathcal{S}$ sends (extend, N) and $\mathbf{v}$ to $\mathcal{F}_{\mathsf{spsVOLE}}$.

The simulation is perfect. Without loss of generality, assume a deterministic environment $\mathcal{Z}$. The view of $\mathcal{A}$ consists of (Δ, Γ) in the one-time **Initialize** execution, and $(\mathsf{K}[r_1], \dots, \mathsf{K}[r_n])$, $\mathsf{K}[s]$, and d in each concurrent **Extend** execution. Note that (Δ, Γ) is the first transcript chosen by $\mathcal{Z}$ and is identical in the two worlds. The transcript d in each concurrent execution is uniform and independent in the real world (due to the execution-wise uniform one-time pad s) and the ideal world. The other transcripts are chosen by $\mathcal{Z}$ and, conditioned on some fixed d's in all concurrent executions, have the same literal values in the two worlds as they can only depend on the d's.

In the concurrent executions, the output $(\mathbf{u}, \mathbf{w})$'s are identically distributed in the two worlds conditioned on some literal values of the one-time initialized (Δ, Γ) and all $(\mathsf{K}[r_1], \dots, \mathsf{K}[r_n])$'s, $\mathsf{K}[s]$'s, and d's. First, in each **Extend** execution, the real-world index α of the only non-zero entry of $\mathbf{u}$ is as uniform and independent as the ideal-world one due to the execution-wise uniform COT choice bits in the presence of the corrupted P_0, and the value β is identically distributed in the two worlds. As a result, the real-world $\mathbf{u}$'s in the concurrent executions are identically distributed as the ideal-world ones. Second, in each **Extend** execution additionally conditioned on some literal value of $\mathbf{u}$, the vector $\mathbf{w}$ takes the same value, i.e., $\mathbf{w} = \mathbf{v}^* + \mathbf{u} \cdot \Gamma$, in the two worlds. In the real world, it follows from the well-formed transcript $(c_1, \dots, c_{n-1}, \mu, c_n^0, c_n^1, \psi)$ of the semi-honest P_0 and the correctness of $\Pi_{\mathsf{spsVOLE-pcGGM}}$ that the literal value of $\mathbf{v}^*$ in this equality comes from $(\mathbf{v}^*, \dots) := \mathsf{pcGGM.FullEval}(\Delta, \mathsf{K}[r_1] \oplus c_1)$. Note that the randomness c_1 chosen by $\mathcal{Z}$ is identical in the two worlds due to the deterministic $\mathcal{Z}$. Thus, the ideal-world $\mathbf{v}$ must be identical to the real-world $\mathbf{v}^*$ conditioned on the fixed transcripts, leading to the same $\mathbf{w}$ in the two worlds.

Corrupted P_1. In the one-time **Initialize** execution:

1. Upon receiving the first (init) from $\mathcal{A}$ to $\mathcal{F}_{\mathsf{COT}}$, $\mathcal{S}$ does nothing.
2. Upon receiving the first (init) from $\mathcal{A}$ to $\mathcal{F}_{\mathsf{sVOLE}}$, $\mathcal{S}$ sends (init) to $\mathcal{F}_{\mathsf{spsVOLE}}$.

Then, in each **Extend** execution:

3. Upon receiving (extend, n) from $\mathcal{A}$ to $\mathcal{F}_{\mathsf{COT}}$, $\mathcal{S}$ waits for $\mathcal{A}$ to choose the COT transcript $((r_1, \dots, r_n), (\mathsf{M}[r_1], \dots, \mathsf{M}[r_n]))$.

4. Upon receiving $(\mathsf{extend}, 1)$ from $\mathcal{A}$ to $\mathcal{F}_{\mathsf{sVOLE}}$, $\mathcal{S}$ waits for $\mathcal{A}$ to choose the sVOLE transcript $(s, \mathsf{M}[s])$.

5. $\mathcal{S}$ receives d from $\mathcal{A}$ and extracts $\beta := s - d$.

6. $\mathcal{S}$ sends random $(c_1, \ldots, c_{n-1}, \mu, c_n^0, c_n^1, \psi) \leftarrow \mathbb{F}_{2^\lambda}^n \times \mathbb{K}^3$ to $\mathcal{A}$ and computes

$$\mathbf{u} := \mathbf{unit}_{\mathbb{F}}(N, \overline{r}_1 \ldots \overline{r}_n, \beta),$$
$$\mathbf{w} := \mathsf{pcGGM.PuncFullEval}(\overline{r}_1 \ldots \overline{r}_n, \{K_i^{r_i}\}_{i \in [1,n]}, \psi + \mathsf{M}[\beta]).$$

Then, $\mathcal{S}$ sends (extend, N) and $(\mathbf{u}, \mathbf{w})$ to $\mathcal{F}_{\mathsf{spsVOLE}}$.

We use hybrid argument to prove that the two worlds are computationally indistinguishable.

- Hybrid_0. This is the real world.
- Hybrid_1. This hybrid is identical to the previous one, except that $\mathcal{S}$ emulates $\mathcal{F}_{\mathsf{COT}}$ and, in each **Extend** execution, the transcript $(c_1, \ldots, c_{n-1}, \mu, c_n^0, c_n^1, \psi)$ sent to $\mathcal{A}$ is computed as follows.

$\mathcal{S}$ has access to the real-world oracle $\mathsf{Real}_{\mathsf{H}_S,\Delta}(\cdot)$ (c.f. Lemma 1) for some $\Delta \leftarrow \{0,1\}^\lambda$ and uses it for all concurrent **Extend** executions. At the beginning of each concurrent execution: First, $\mathcal{S}$ queries $\mathsf{Real}_{\mathsf{H}_S,\Delta}(\cdot)$ with $Q_1 : X_1^{\overline{\alpha}_1} := \mathsf{xor}_1 \leftarrow \{0,1\}^\lambda$. Then, for $i \in [2, n-1]$, it does:

– Query $\mathsf{Real}_{\mathsf{H}_S,\Delta}(\cdot)$ with $Q_{i,1} : \mathsf{temp}_1 := \mathcal{O}(\mathsf{xor}_{i-1}, \overline{\alpha}_i)$.
– Query $\mathsf{Real}_{\mathsf{H}_S,\Delta}(\cdot)$ with $Q_{i,2} : \mathsf{temp}_2 := \mathsf{temp}_1 \oplus \mathsf{xor}_{i-1}$.
– If $\overline{\alpha}_i = 0$, regard temp_1 as $X_i^{\alpha_1 \ldots \alpha_{i-1}\overline{\alpha}_i}$ and temp_2 as xor_i (without new query).
– If $\overline{\alpha}_i = 1$, regard temp_1 as xor_i and temp_2 as $X_i^{\alpha_1 \ldots \alpha_{i-1}\overline{\alpha}_i}$ (without new query).

Finally, it queries $\mathsf{Real}_{\mathsf{H}_S,\Delta}(\cdot)$ with the following operations:

$$Q_{n,1} : R_0 := \mathcal{O}(\mathsf{xor}_{n-1}, 0),$$
$$Q_{n,2} : \mathsf{temp} := \mathsf{xor}_{n-1} \oplus 1,$$
$$Q_{n,3} : R_1 := \mathcal{O}(\mathsf{temp}, 0),$$
$$Q_{n,4} : \mathsf{rand} \leftarrow \{0,1\}^\lambda,$$
$$Q_{n,5} : \mathsf{pad} := \mathcal{O}(\mathsf{rand}, 0).$$

$\mathcal{S}$ defines $X_n^{\alpha_1 \ldots \alpha_{n-1}\overline{\alpha}_n} := \mathsf{Convert}_{\mathbb{K}}(R_{\overline{\alpha}_n}), X_n := \mathsf{Convert}_{\mathbb{K}}(R_{\alpha_n})$ and compute $\{K_i^{\overline{\alpha}_i}\}_{i \in [1,n]}$ and $K_n^{\alpha_n}$ from $\alpha := \overline{r}_1 \ldots \overline{r}_n$, $\{X_i^{\alpha_1 \ldots \alpha_{i-1}\overline{\alpha}_i}\}_{i \in [1,n]}$, and X_n.

In the rest of this execution, $\mathcal{S}$ uses these oracle responses for the transcript of the honest P_0:

$$c_i := K_i^{\overline{\alpha}_i} \oplus \mathsf{M}[r_i], \quad \forall i \in [1, n-1]$$
$$\mu := \mathsf{rand} \oplus \mathsf{M}[r_n],$$
$$c_n^{r_n} := K_n^{\overline{\alpha}_n} + \mathsf{Convert}_{\mathbb{K}}(\mathsf{H}_S(\mathsf{rand})),$$
$$c_n^{\overline{r}_n} := K_n^{\alpha_n} + \mathsf{Convert}_{\mathbb{K}}(\mathsf{pad}),$$
$$\psi := K_n^{\overline{\alpha}_n} + K_n^{\alpha_n} - \mathsf{K}[\beta].$$

The resulting transcript is equivalently defined as in the previous hybrid under the identically distributed global Δ and the identically distributed c_1 and μ in each **Extend** execution. Therefore, this hybrid is identically distributed as the previous one.

Now, $\mathcal{S}$ can directly use the path $\alpha := \overline{r}_1 \ldots \overline{r}_n$, the n off-path nodes, and the punctured leaf X_n to define all pcGGM-tree leaves, i.e., the output $\mathbf{v}$ of the honest P_0.

- Hybrid_2. This hybrid is identical to the previous one, except that $\mathcal{S}$ has access to the ideal-world oracle $\mathsf{Ideal}_{\mathsf{H}_S}(\cdot)$ instead of the real-world oracle $\mathsf{Real}_{\mathsf{H}_S,\Delta}(\cdot)$ in Lemma 1. It follows from the lemma that this hybrid is computationally indistinguishable from the previous one.

  In each **Extend** execution, the n off-path nodes are uniform due to $\mathsf{Ideal}_{\mathsf{H}_S}(\cdot)$ and serve as one-time pad for $\{K_i^{\overline{\alpha}_i}\}_{i\in[1,n-1]}$, leading to uniform $\{c_i\}_{i\in[1,n-1]}$. Meanwhile, the μ and pad in each concurrent execution are also uniform due to $\mathsf{Ideal}_{\mathsf{H}_S}(\cdot)$.

- Hybrid_3. This hybrid is identical to the previous one, except that, in each **Extend** execution, $\mathcal{S}$ replaces $c_n^{r_n}$, $c_n^{\overline{r}_n}$, and ψ by random values. This hybrid is computationally indistinguishable from the previous one since, from the construction of $K_n^{\overline{\alpha}_n}$ and $K_n^{\alpha_n}$, $X_n^{\alpha_1\ldots\alpha_{n-1}\overline{\alpha}_n}$, $\mathsf{Convert}_{\mathbb{K}}(\mathsf{pad})$, and X_n essentially serve as pseudorandom one-time pad for $c_n^{r_n}$, $c_n^{\overline{r}_n}$, and ψ, respectively.

- Hybrid_4. This hybrid is identical to the previous one, except that $\mathcal{S}$ emulates $\mathcal{F}_{\mathsf{sVOLE}}$ and, in each **Extend** execution, (i) follows protocol $\Pi_{\mathsf{spsVOLE-pcGGM}}$ to compute the output $(\mathbf{u}, \mathbf{w})$ of the corrupted P_1 by using its internal randomness extracted from the emulated $\mathcal{F}_{\mathsf{COT}}$ and $\mathcal{F}_{\mathsf{sVOLE}}$ or the transcript received from $\mathcal{A}$ (i.e., the extracted $\beta := s - d$), and the transcript sent by $\mathcal{S}$, and (ii) sends (extend, N) and $(\mathbf{u}, \mathbf{w})$ to $\mathcal{F}_{\mathsf{spsVOLE}}$.

  This hybrid is identically distributed as the previous one since, from the correctness of protocol $\Pi_{\mathsf{spsVOLE-pcGGM}}$, the only difference between the two hybrids is where and how the spsVOLE global key Γ is defined, and the real-world Γ is sampled in $\mathcal{F}_{\mathsf{sVOLE}}$ and identically distributed as the ideal-world one in $\mathcal{F}_{\mathsf{spsVOLE}}$.

  It is clear that this hybrid is the ideal world.

The above hybrid argument completes this proof. $\square$

C PPRF from Pseudorandom Correlated GGM

We first recall the following definition of PPRF.

Definition 4 (Puncturable Pseudorandom Function, [BCG+19b]). *A (single-point) PPRF scheme* PPRF *with key space $\mathcal{K}$, punctured key space $\mathcal{K}_P$, domain $\mathcal{X}$, and range $\mathbb{G}$, where $\mathbb{G}$ is an Abelian group, has the following syntax:*

- $k_{\mathsf{pprf}} \leftarrow \mathsf{PPRF.Gen}(1^\lambda)$. *On input 1^λ, output a key $k_{\mathsf{pprf}} \in \mathcal{K}$.*
- $k_{\mathsf{pprf}}\{\alpha\} \leftarrow \mathsf{PPRF.Punc}(k_{\mathsf{pprf}}, \alpha)$. *On input a key $k_{\mathsf{pprf}} \in \mathcal{K}$ and a punctured point $\alpha \in \mathcal{X}$, output a punctured key $k_{\mathsf{pprf}}\{\alpha\} \in \mathcal{K}_P$.*
- $y \leftarrow \mathsf{PPRF.Eval}(k_{\mathsf{pprf}}, x)$. *On input a key $k_{\mathsf{pprf}} \in \mathcal{K}$ and a point $x \in \mathcal{X}$, output the result $y \in \mathbb{G}$.*
- $\{y, \perp\} \leftarrow \mathsf{PPRF.PuncEval}(\alpha, k_{\mathsf{pprf}}\{\alpha\}, x)$. *On input a punctured key $k_{\mathsf{pprf}}\{\alpha\} \in \mathcal{K}_P$ and a point $x \in \mathcal{X}$, output the result $y \in \mathbb{G}$ if $x \neq \alpha$; otherwise output $\perp$.*

  A PPRF scheme PPRF *is secure if the following properties hold.*

- **Correctness**. *For any $x, \alpha \in \mathcal{X}$, it holds that*

$$\Pr\left[\begin{array}{l} k_{\mathsf{pprf}} \leftarrow \mathsf{PPRF.Gen}(1^\lambda), \\ k_{\mathsf{pprf}}\{\alpha\} \leftarrow \mathsf{PPRF.Punc}(k_{\mathsf{pprf}}, \alpha) \end{array} : \begin{array}{l} \mathsf{PPRF.Eval}(k_{\mathsf{pprf}}, x) \\ \quad = \mathsf{PPRF.PuncEval}(\alpha, k_{\mathsf{pprf}}\{\alpha\}, x) \end{array}\right] = 1.$$

- **Pseudorandomness**. *For any PPT adversary $\mathcal{A}$, and any $\alpha \in \mathcal{X}$ chosen by $\mathcal{A}$, it holds that*

$$\left|\Pr\left[\begin{array}{l} k_{\mathsf{pprf}} \leftarrow \mathsf{PPRF.Gen}(1^\lambda), \\ k_{\mathsf{pprf}}\{\alpha\} \leftarrow \mathsf{PPRF.Punc}(k_{\mathsf{pprf}}, \alpha), \\ y \leftarrow \mathsf{PPRF.Eval}(k_{\mathsf{pprf}}, \alpha) \end{array} : \mathcal{A}(1^\lambda, \alpha, k_{\mathsf{pprf}}\{\alpha\}, y) = 1\right] \right.$$
$$\left. - \Pr\left[\begin{array}{l} k_{\mathsf{pprf}} \leftarrow \mathsf{PPRF.Gen}(1^\lambda), \\ k_{\mathsf{pprf}}\{\alpha\} \leftarrow \mathsf{PPRF.Punc}(k_{\mathsf{pprf}}, \alpha), \\ y \leftarrow \mathbb{G} \end{array} : \mathcal{A}(1^\lambda, \alpha, k_{\mathsf{pprf}}\{\alpha\}, y) = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

In Figure 15, we present a pcGGM-based PPRF scheme, whose security is proved in Theorem 8.

Theorem 8. *Given CCR function $\mathsf{H} : \mathbb{F}_{2^\lambda} \to \mathbb{F}_{2^\lambda}$, function $\mathsf{Convert}_{\mathbb{K}} : \mathbb{F}_{2^\lambda} \to \mathbb{K}$, and keyed hash function $\mathsf{H}_S(x) := \mathsf{H}(S \oplus x)$ with some key $S \leftarrow \mathbb{F}_{2^\lambda}$, Figure 15 gives a PPRF scheme with domain $[0, N)$ and range $\mathbb{K}$.*

Proof. It is clear that this PPRF scheme is correct, and we focus on its pseudorandomness. For any $\alpha \in \{0,1\}^n$ chosen by $\mathcal{A}$, we write $\mathsf{Expt}_{\mathsf{real}}(1^\lambda, \alpha)$ (resp., $\mathsf{Expt}_{\mathsf{ideal}}(1^\lambda, \alpha)$) for the distribution of $(k_{\mathsf{pprf}}\{\alpha\}, y)$ resulting from the experiment where $k_{\mathsf{pprf}} \leftarrow \mathsf{PPRF.Gen}(1^\lambda)$, $k_{\mathsf{pprf}}\{\alpha\} \leftarrow \mathsf{PPRF.Punc}(k_{\mathsf{pprf}}, \alpha)$, and $y \leftarrow \mathsf{PPRF.Eval}(k_{\mathsf{pprf}}, \alpha)$ (resp., the one where $k_{\mathsf{pprf}} \leftarrow \mathsf{PPRF.Gen}(1^\lambda)$, $k_{\mathsf{pprf}}\{\alpha\} \leftarrow \mathsf{PPRF.Punc}(k_{\mathsf{pprf}}, \alpha)$, and $y \leftarrow \mathbb{K}$). Let $\mathsf{Expt}_{\mathsf{rand}}(1^\lambda, \alpha)$ be the distribution resulting from the hybrid experiment where $k_{\mathsf{pprf}}\{\alpha\} \leftarrow \mathbb{F}_{2^\lambda}^{n-1} \times \mathbb{K}$ and $y \leftarrow \mathbb{K}$. This theorem follows from the following two lemmas.

Lemma 4. *For any PPT adversary $\mathcal{A}$, and any $\alpha \in \mathcal{X}$ chosen by $\mathcal{A}$,*

$$\left|\Pr\left[\begin{array}{l} (k_{\mathsf{pprf}}\{\alpha\}, y) \\ \qquad \leftarrow \mathsf{Expt}_{\mathsf{real}}(1^\lambda, \alpha) \end{array} : \mathcal{A}(1^\lambda, \alpha, k_{\mathsf{pprf}}\{\alpha\}, y) = 1\right]\right.$$
$$\left. - \Pr\left[\begin{array}{l} (k_{\mathsf{pprf}}\{\alpha\}, y) \\ \qquad \leftarrow \mathsf{Expt}_{\mathsf{rand}}(1^\lambda, \alpha) \end{array} : \mathcal{A}(1^\lambda, \alpha, k_{\mathsf{pprf}}\{\alpha\}, y) = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

Parameters: Domain size $N = 2^n$ for $n \in \mathbb{N}$. Field $\mathbb{K}$. pcGGM (c.f. Figure 5) for n and $\mathbb{K}$.

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

1: **return** $k_{\mathsf{pprf}} := (\Delta, k) \leftarrow \mathbb{F}^2_{2^\lambda}$.

$\mathsf{PPRF.Punc}(k_{\mathsf{pprf}}, \alpha)$:

1: Parse $k_{\mathsf{pprf}} = (\Delta, k) \in \mathbb{F}^2_{2^\lambda}$ and $\alpha = \alpha_1 \ldots \alpha_n \in \{0,1\}^n$.
2: Run $(\mathbf{v}, \{K_i^0\}_{i\in[1,n-1]}, (K_n^0, K_n^1)) := \mathsf{pcGGM.FullEval}(\Delta, k)$.
3: **for** $i \in [1, n-1]$ **do** $K_i^{\overline{\alpha}_i} := \overline{\alpha}_i \cdot \Delta \oplus K_i^0$.
4: **return** $k_{\mathsf{pprf}}\{\alpha\} := \{K_i^{\overline{\alpha}_i}\}_{i\in[1,n]}$.

$\mathsf{PPRF.Eval}(k_{\mathsf{pprf}}, x)$:

1: Parse $k_{\mathsf{pprf}} = (\Delta, k) \in \mathbb{F}^2_{2^\lambda}$.
2: Run $(\mathbf{v}, \{K_i^0\}_{i\in[1,n-1]}, (K_n^0, K_n^1)) := \mathsf{pcGGM.FullEval}(\Delta, k)$.
3: **return** $\mathbf{v}^{(x)}$.

$\mathsf{PPRF.PuncEval}(\alpha, k_{\mathsf{pprf}}\{\alpha\}, x)$: // Output $\perp$ if $x = \alpha$

1: Parse $k_{\mathsf{pprf}}\{\alpha\} = \{K_i^{\overline{\alpha}_i}\}_{i\in[1,n]} \in \mathbb{F}^{n-1}_{2^\lambda} \times \mathbb{K}$.
2: Run $\mathbf{w} := \mathsf{pcGGM.PuncFullEval}(\alpha, \{K_i^{\overline{\alpha}_i}\}_{i\in[1,n]}, \perp)$.
3: **return** $\mathbf{w}^{(x)}$.

Figure 15: Puncturable pseudorandom function with domain $[0, N)$ and range $\mathbb{K}$.

Proof. Let $\mathsf{Expt}_{\mathsf{hyb}}(1^\lambda, \alpha)$ be identical to $\mathsf{Expt}_{\mathsf{rand}}(1^\lambda, \alpha)$, except that

$$X_n^{\alpha_1\ldots\alpha_{n-1}\overline{\alpha}_n} := \mathsf{Convert}_{\mathbb{K}}(R_{\overline{\alpha}_n}), \quad y := \mathsf{Convert}_{\mathbb{K}}(R_{\alpha_n})$$

for some $R_0, R_1 \leftarrow \mathbb{F}_{2^\lambda}$.

First, we prove that, for any PPT $\mathcal{A}$ and any $\alpha \in \{0,1\}^n$ chosen by $\mathcal{A}$,

$$\begin{aligned}\Bigg| \Pr\left[\begin{array}{l}(k_{\mathsf{pprf}}\{\alpha\}, y) \\ \quad \leftarrow \mathsf{Expt}_{\mathsf{real}}(1^\lambda, \alpha)\end{array} : \mathcal{A}(1^\lambda, \alpha, k_{\mathsf{pprf}}\{\alpha\}, y) = 1\right] \\ - \Pr\left[\begin{array}{l}(k_{\mathsf{pprf}}\{\alpha\}, y) \\ \quad \leftarrow \mathsf{Expt}_{\mathsf{hyb}}(1^\lambda, \alpha)\end{array} : \mathcal{A}(1^\lambda, \alpha, k_{\mathsf{pprf}}\{\alpha\}, y) = 1\right]\Bigg| \le \mathsf{negl}(\lambda).\end{aligned} \tag{14}$$

Assume that this does not hold for some $\mathcal{A}$. Then, there exists an adversary $\mathcal{B}$ can break Lemma 1 for the CCR hash function H and $\mathcal{T} = \{1\}$. $\mathcal{B}$ is given an oracle $\mathcal{O}' \in \{\mathsf{Real}_{\mathsf{H}_S,\Delta}, \mathsf{Ideal}_{\mathsf{H}_S}\}$ considered in Lemma 1, where $\Delta \leftarrow \mathbb{F}_{2^\lambda}$, and works as follows.

1. $\mathcal{B}$ internally runs $\mathcal{A}$ and receives $\alpha \in \{0,1\}^n$ from $\mathcal{A}$.
2. $\mathcal{B}$ queries $\mathcal{O}'$ with $Q_1 : X_1^{\overline{\alpha}_1} := \mathsf{xor}_1 \leftarrow \{0,1\}^\lambda$. Then, for $i \in [2, n-1]$, it does:
   - Query $\mathcal{O}'$ with $Q_{i,1} : \mathsf{temp}_1 := \mathcal{O}(\mathsf{xor}_{i-1}, \overline{\alpha}_i)$.
   - Query $\mathcal{O}'$ with $Q_{i,2} : \mathsf{temp}_2 := \mathsf{temp}_1 \oplus \mathsf{xor}_{i-1}$.
   - If $\overline{\alpha}_i = 0$, regard temp_1 as $X_i^{\alpha_1\ldots\alpha_{i-1}\overline{\alpha}_i}$ and temp_2 as xor_i (without new query).
   - If $\overline{\alpha}_i = 1$, regard temp_1 as xor_i and temp_2 as $X_i^{\alpha_1\ldots\alpha_{i-1}\overline{\alpha}_i}$ (without new query).

   Finally, it queries $\mathcal{O}'$ with the following operations:

$$\begin{aligned}Q_{n,1} &: R_0 := \mathcal{O}(\mathsf{xor}_{n-1}, 0), \\ Q_{n,2} &: \mathsf{temp} := \mathsf{xor}_{n-1} \oplus 1, \\ Q_{n,3} &: R_1 := \mathcal{O}(\mathsf{temp}, 0).\end{aligned}$$

$\mathcal{S}$ defines $X_n^{\alpha_1\ldots\alpha_{n-1}\overline{\alpha}_n} := \mathsf{Convert}_{\mathbb{K}}(R_{\overline{\alpha}_n})$ and $y := \mathsf{Convert}_{\mathbb{K}}(R_{\alpha_n})$.

3. $\mathcal{B}$ computes $k_{\mathsf{pprf}}\{\alpha\} := \{K_i^{\overline{\alpha}_i}\}_{i\in[1,n]}$ from α and the n off-path nodes, sends $(k_{\mathsf{pprf}}\{\alpha\}, y)$ to $\mathcal{A}$, and outputs whatever $\mathcal{A}$ outputs.

On the one hand, if $\mathcal{B}$ is given $\mathsf{Real}_{\mathsf{H}_S,\Delta}(\cdot)$ for some $\Delta \leftarrow \mathbb{F}_{2^\lambda}$, then

$$\forall i \in [2, n-1] : X_i^{\alpha_1\ldots\alpha_{i-1}\overline{\alpha}_i} = \mathsf{H}_S\left(\Delta \oplus \bigoplus\nolimits_{j\in[1,i-1]} X_j^{\alpha_1\ldots\alpha_{j-1}\overline{\alpha}_j}\right) \oplus \overline{\alpha}_i \cdot \left(\Delta \oplus \bigoplus\nolimits_{j\in[1,i-1]} X_j^{\alpha_1\ldots\alpha_{j-1}\overline{\alpha}_j}\right),$$
$$R_{\overline{\alpha}_n} = \mathsf{H}_S\left(\Delta \oplus (\bigoplus\nolimits_{j\in[1,n-1]} X_j^{\alpha_1\ldots\alpha_{j-1}\overline{\alpha}_j}) \oplus \overline{\alpha}_n\right),$$
$$R_{\alpha_n} = \mathsf{H}_S\left(\Delta \oplus (\bigoplus\nolimits_{j\in[1,n-1]} X_j^{\alpha_1\ldots\alpha_{j-1}\overline{\alpha}_j}) \oplus \alpha_n\right).$$

The resulting $(k_{\mathsf{pprf}}\{\alpha\}, y)$ is identically distributed as that from $\mathsf{Expt}_{\mathsf{real}}(1^\lambda, \alpha)$. On the other hand, if $\mathcal{B}$ is given $\mathsf{Ideal}_{\mathsf{H}_S}(\cdot)$, then we have uniformly sampled $\{X_i^{\alpha_1\ldots\alpha_{i-1}\overline{\alpha}_i}\}_{i\in[1,n-1]}$ and (R_0, R_1). The resulting $(k_{\mathsf{pprf}}\{\alpha\}, y)$ is identically distributed as that from $\mathsf{Expt}_{\mathsf{hyb}}(1^\lambda, \alpha)$. Hence, it follows from the contradiction assumption that $\mathcal{B}$ breaks Lemma 1 for the CCR hash function H and $\mathcal{T} = \{1\}$.

Second, it is easy to see that, due to the pseudorandomness of $\mathsf{Convert}_{\mathbb{K}}$,

$$\begin{aligned}\Bigg|\Pr\left[\begin{array}{l}(k_{\mathsf{pprf}}\{\alpha\}, y) \\ \quad \leftarrow \mathsf{Expt}_{\mathsf{hyb}}(1^\lambda, \alpha)\end{array} : \mathcal{A}(1^\lambda, \alpha, k_{\mathsf{pprf}}\{\alpha\}, y) = 1\right] \\ - \Pr\left[\begin{array}{l}(k_{\mathsf{pprf}}\{\alpha\}, y) \\ \quad \leftarrow \mathsf{Expt}_{\mathsf{rand}}(1^\lambda, \alpha)\end{array} : \mathcal{A}(1^\lambda, \alpha, k_{\mathsf{pprf}}\{\alpha\}, y) = 1\right]\Bigg| \le \mathsf{negl}(\lambda).\end{aligned} \tag{15}$$

Using (14) and (15), one can see that this lemma holds. □

Lemma 5. *For any PPT adversary $\mathcal{A}$, and any $\alpha \in \mathcal{X}$ chosen by $\mathcal{A}$,*

$$\begin{aligned}\Bigg|\Pr\left[\begin{array}{l}(k_{\mathsf{pprf}}\{\alpha\}, y) \\ \quad \leftarrow \mathsf{Expt}_{\mathsf{rand}}(1^\lambda, \alpha)\end{array} : \mathcal{A}(1^\lambda, \alpha, k_{\mathsf{pprf}}\{\alpha\}, y) = 1\right] \\ - \Pr\left[\begin{array}{l}(k_{\mathsf{pprf}}\{\alpha\}, y) \\ \quad \leftarrow \mathsf{Expt}_{\mathsf{ideal}}(1^\lambda, \alpha)\end{array} : \mathcal{A}(1^\lambda, \alpha, k_{\mathsf{pprf}}\{\alpha\}, y) = 1\right]\Bigg| \le \mathsf{negl}(\lambda).\end{aligned}$$

Proof. Let $\mathsf{Expt}_{\mathsf{hyb}}(1^\lambda, \alpha)$ be identical to $\mathsf{Expt}_{\mathsf{rand}}(1^\lambda, \alpha)$, except that

$$X_n^{\alpha_1\ldots\alpha_{n-1}\overline{\alpha}_n} := \mathsf{Convert}_{\mathbb{K}}(R_{\overline{\alpha}_n})$$

for some $R_{\overline{\alpha}_n} \leftarrow \mathbb{F}_{2^\lambda}$.

First, we prove that, for any PPT $\mathcal{A}$ and any $\alpha \in \{0,1\}^n$ chosen by $\mathcal{A}$,

$$\begin{aligned}\Bigg|\Pr\left[\begin{array}{l}(k_{\mathsf{pprf}}\{\alpha\}, y) \\ \quad \leftarrow \mathsf{Expt}_{\mathsf{ideal}}(1^\lambda, \alpha)\end{array} : \mathcal{A}(1^\lambda, \alpha, k_{\mathsf{pprf}}\{\alpha\}, y) = 1\right] \\ - \Pr\left[\begin{array}{l}(k_{\mathsf{pprf}}\{\alpha\}, y) \\ \quad \leftarrow \mathsf{Expt}_{\mathsf{hyb}}(1^\lambda, \alpha)\end{array} : \mathcal{A}(1^\lambda, \alpha, k_{\mathsf{pprf}}\{\alpha\}, y) = 1\right]\Bigg| \le \mathsf{negl}(\lambda).\end{aligned} \tag{16}$$

Assume that this does not hold for some $\mathcal{A}$. We use this adversary $\mathcal{A}$ to construct the adversary $\mathcal{B}$, which is given an oracle $\mathcal{O}' \in \{\mathsf{Real}_{\mathsf{H}_S,\Delta}, \mathsf{Ideal}\}$ for some $\Delta \leftarrow \mathbb{F}_{2^\lambda}$, against Lemma 1 for the CCR hash function H and $\mathcal{T} = \{1\}$.

1. $\mathcal{B}$ internally runs $\mathcal{A}$ and receives $\alpha \in \{0,1\}^n$ from $\mathcal{A}$.

2. $\mathcal{B}$ queries $\mathcal{O}'$ with $Q_1 : X_1^{\overline{\alpha}_1} := \mathsf{xor}_1 \leftarrow \{0,1\}^\lambda$. Then, for $i \in [2, n-1]$, it does:
   - Query $\mathcal{O}'$ with $Q_{i,1} : \mathsf{temp}_1 := \mathcal{O}(\mathsf{xor}_{i-1}, \overline{\alpha}_i)$.
   - Query $\mathcal{O}'$ with $Q_{i,2} : \mathsf{temp}_2 := \mathsf{temp}_1 \oplus \mathsf{xor}_{i-1}$.
   - If $\overline{\alpha}_i = 0$, regard temp_1 as $X_i^{\alpha_1 \ldots \alpha_{i-1} \overline{\alpha}_i}$ and temp_2 as xor_i (without new query).
   - If $\overline{\alpha}_i = 1$, regard temp_1 as xor_i and temp_2 as $X_i^{\alpha_1 \ldots \alpha_{i-1} \overline{\alpha}_i}$ (without new query).

   Finally, it proceeds as follows:
   - If $\overline{\alpha}_n = 0$, query $\mathcal{O}'$ with $Q_{n,1} : R_{\overline{\alpha}_n} := \mathcal{O}(\mathsf{xor}_{n-1}, 0)$.
   - If $\overline{\alpha}_n = 1$, query $\mathcal{O}'$ with:

   $$Q_{n,1} : \mathsf{temp} := \mathsf{xor}_{n-1} \oplus 1,$$
   $$Q_{n,2} : R_{\overline{\alpha}_n} := \mathcal{O}(\mathsf{temp}, 0).$$

   $\mathcal{B}$ defines $X_n^{\alpha_1 \ldots \alpha_{n-1} \overline{\alpha}_n} := \mathsf{Convert}_{\mathbb{K}}(R_{\overline{\alpha}_n})$ and samples $y \leftarrow \mathbb{K}$.

3. $\mathcal{B}$ computes $k_{\mathsf{pprf}}\{\alpha\} := \{K_i^{\overline{\alpha}_i}\}_{i \in [1,n]}$ from α and the n off-path nodes, sends $(k_{\mathsf{pprf}}\{\alpha\}, y)$ to $\mathcal{A}$, and outputs whatever $\mathcal{A}$ outputs.

On the one hand, if $\mathcal{B}$ is given $\mathsf{Real}_{\mathsf{H}_S, \Delta}(\cdot)$ for some $\Delta \leftarrow \mathbb{F}_{2^\lambda}$, then the resulting $(k_{\mathsf{pprf}}\{\alpha\}, y)$ is identically distributed as that from $\mathsf{Expt}_{\mathsf{ideal}}(1^\lambda, \alpha)$. On the other hand, if $\mathcal{B}$ is given $\mathsf{Ideal}_{\mathsf{H}_S}(\cdot)$, then $\{X_i^{\alpha_1 \ldots \alpha_{i-1} \overline{\alpha}_i}\}_{i \in [1, n-1]}$ and $R_{\overline{\alpha}_n}$ are uniformly sampled. The resulting $(k_{\mathsf{pprf}}\{\alpha\}, y)$ is identically distributed as that from $\mathsf{Expt}_{\mathsf{hyb}}(1^\lambda, \alpha)$. Therefore, it follows from the contradiction assumption that $\mathcal{B}$ breaks Lemma 1 for the CCR hash function H and $\mathcal{T} = \{1\}$.

Second, using the pseudorandomness of $\mathsf{Convert}_{\mathbb{K}}$, we have that

$$\begin{aligned} \Bigg| \Pr \left[\begin{array}{c} (k_{\mathsf{pprf}}\{\alpha\}, y) \\ \leftarrow \mathsf{Expt}_{\mathsf{hyb}}(1^\lambda, \alpha) \end{array} : \mathcal{A}(1^\lambda, \alpha, k_{\mathsf{pprf}}\{\alpha\}, y) = 1 \right] \\ - \Pr \left[\begin{array}{c} (k_{\mathsf{pprf}}\{\alpha\}, y) \\ \leftarrow \mathsf{Expt}_{\mathsf{rand}}(1^\lambda, \alpha) \end{array} : \mathcal{A}(1^\lambda, \alpha, k_{\mathsf{pprf}}\{\alpha\}, y) = 1 \right] \Bigg| \leq \mathsf{negl}(\lambda). \end{aligned} \tag{17}$$

This lemma is immediate from (16) and (17). □

Combining the above two lemmas, this theorem holds. □

D Security Proofs in Section 5

D.1 Proofs of Two Correctness Lemmas for DPF and DCF

Before introducing our two correctness lemmas, we have the following claim.

Claim 2. *For any $n \in \mathbb{N}$, $x, \alpha \in \{0,1\}^n$, $i \in [0,n]$, and $b \in \{0,1\}$, it holds in* $\mathsf{DPF.Eval}$ *that*

$$\langle s_i^{x_1\ldots x_i} \,\|\, t_i^{x_1\ldots x_i}\rangle_b = \langle s_i \,\|\, t_i\rangle_b,$$

where the right-hand term is in the scope of $\mathsf{DPF.Gen}$*, if $x_1 \ldots x_i = \alpha_1 \ldots \alpha_i$.*

Proof. This claim follows from that $\langle s_i^{x_1\ldots x_i} \,\|\, t_i^{x_1\ldots x_i}\rangle_b$ (resp., $\langle s_i \,\|\, t_i\rangle_b$) is a deterministic function of the *same* share $\langle s_0^0 \,\|\, t_0^0\rangle_b = \langle s_0 \,\|\, t_0\rangle_b$ of the root, the prefix $x_1 \ldots x_i$ (resp., $\alpha_1 \ldots \alpha_i$), and the *same* public correction words $\mathsf{CW}_1, \ldots, \mathsf{CW}_i$ used in the two algorithms. □

Lemma 6. *For any $n \in \mathbb{N}$, $x, \alpha \in \{0,1\}^n$, and $i \in [0,n]$, it holds in* $\mathsf{DPF.Eval}$ *that*

$$\langle s_i^{x_1\ldots x_i} \,\|\, t_i^{x_1\ldots x_i}\rangle_0 \oplus \langle s_i^{x_1\ldots x_i} \,\|\, t_i^{x_1\ldots x_i}\rangle_1 = \begin{cases} \Delta & \text{, if } x_1 \ldots x_i = \alpha_1 \ldots \alpha_i,\ i < n \\ (\mathsf{HCW} \oplus \mathsf{HCW}^*) \,\|\, 1 & \text{, if } x = \alpha \\ 0^\lambda & \text{, if } x_1 \ldots x_i \neq \alpha_1 \ldots \alpha_i \end{cases}$$

where $\mathsf{HCW}^* := \langle \mathsf{high}^{\alpha_n}\rangle_0 \oplus \langle \mathsf{high}^{\alpha_n}\rangle_1$ *is implicitly defined in* $\mathsf{DPF.Gen}$*.*

Proof. We prove Lemma 6 via induction. By construction, we can see that

$$\underbrace{\langle s_0^0 \,\|\, t_0^0\rangle_0 \oplus \langle s_0^0 \,\|\, t_0^0\rangle_1}_{\text{In } \mathsf{DPF.Eval}} = \underbrace{\langle s_0 \,\|\, t_0\rangle_0 \oplus \langle s_0 \,\|\, t_0\rangle_1}_{\text{In } \mathsf{DPF.Gen}} = \Delta,$$

which is exactly the **base case** $i = 0$ of our induction (note that $n \geq 1$). For the **induction step**, we assume that Lemma 6 holds for $i - 1 \geq 0$ and consider $i \leq n$. There are two cases:

- $i = n$. In this case, it holds that

$$\begin{aligned}
&\langle s_n^{x_1\ldots x_n} \,\|\, t_n^{x_1\ldots x_n}\rangle_0 \oplus \langle s_n^{x_1\ldots x_n} \,\|\, t_n^{x_1\ldots x_n}\rangle_1 \\
=\ &\langle \mathsf{high} \,\|\, \mathsf{low}\rangle_0 \oplus \langle t_{n-1}^{x_1\ldots x_{n-1}}\rangle_0 \cdot (\mathsf{HCW} \,\|\, \mathsf{LCW}^{x_n}) \\
&\quad \oplus \langle \mathsf{high} \,\|\, \mathsf{low}\rangle_1 \oplus \langle t_{n-1}^{x_1\ldots x_{n-1}}\rangle_1 \cdot (\mathsf{HCW} \,\|\, \mathsf{LCW}^{x_n}) \\
=\ &\langle \mathsf{high} \,\|\, \mathsf{low}\rangle_0 \oplus \langle \mathsf{high} \,\|\, \mathsf{low}\rangle_1 \\
&\quad \oplus (\langle t_{n-1}^{x_1\ldots x_{n-1}}\rangle_0 \oplus \langle t_{n-1}^{x_1\ldots x_{n-1}}\rangle_1) \cdot (\langle \mathsf{high}^{\overline{\alpha}_n}\rangle_0 \oplus \langle \mathsf{high}^{\overline{\alpha}_n}\rangle_1 \,\|\, \langle \mathsf{low}^{x_n}\rangle_0 \oplus \langle \mathsf{low}^{x_n}\rangle_1 \oplus \alpha_n \oplus \overline{x}_n)
\end{aligned}$$

  1. $x_1 \ldots x_{n-1} = \alpha_1 \ldots \alpha_{n-1}$ and $x_n = \alpha_n$ (i.e., $x = \alpha$). The induction assumption and Claim 2 for $x_1 \ldots x_{n-1} = \alpha_1 \ldots \alpha_{n-1}$ ensure $\langle \mathsf{high} \,\|\, \mathsf{low}\rangle_b = \langle \mathsf{high}^{x_n} \,\|\, \mathsf{low}^{x_n}\rangle_b$ for each $b \in \{0,1\}$, where the left-hand term comes from $\mathsf{DPF.Eval}$ and the right-hand one is in the scope of $\mathsf{DPF.Gen}$. The induction assumption also gives $\langle t_{n-1}^{x_1\ldots x_{n-1}}\rangle_0 \oplus \langle t_{n-1}^{x_1\ldots x_{n-1}}\rangle_1 = \mathsf{lsb}(\Delta) = 1$. Thus,

$$\langle s_n^{x_1\ldots x_n} \,\|\, t_n^{x_1\ldots x_n}\rangle_0 \oplus \langle s_n^{x_1\ldots x_n} \,\|\, t_n^{x_1\ldots x_n}\rangle_1 = (\mathsf{HCW} \oplus \mathsf{HCW}^*) \,\|\, (\alpha_n \oplus \overline{x}_n) = (\mathsf{HCW} \oplus \mathsf{HCW}^*) \,\|\, 1.$$

  2. $x_1 \ldots x_{n-1} = \alpha_1 \ldots \alpha_{n-1}$ but $x_n = \overline{\alpha}_n$ (i.e., $x_1 \ldots x_n \neq \alpha_1 \ldots \alpha_n$). It follows from the same analysis as the previous sub-case that

$$\langle s_n^{x_1\ldots x_n} \,\|\, t_n^{x_1\ldots x_n}\rangle_0 \oplus \langle s_n^{x_1\ldots x_n} \,\|\, t_n^{x_1\ldots x_n}\rangle_1 = 0^{\lambda-1} \,\|\, 0 = 0^\lambda.$$

3. $x_1 \ldots x_{n-1} \neq \alpha_1 \ldots \alpha_{n-1}$ (i.e., $x_1 \ldots x_n \neq \alpha_1 \ldots \alpha_n$). Using the induction assumption, we have $\langle s_{n-1}^{x_1\ldots x_{n-1}} \parallel t_{n-1}^{x_1\ldots x_{n-1}} \rangle_0 = \langle s_{n-1}^{x_1\ldots x_{n-1}} \parallel t_{n-1}^{x_1\ldots x_{n-1}} \rangle_1$ and $\langle \mathsf{high} \parallel \mathsf{low} \rangle_0 = \langle \mathsf{high} \parallel \mathsf{low} \rangle_1$. Thus,

$$\langle s_n^{x_1\ldots x_n} \parallel t_n^{x_1\ldots x_n} \rangle_0 \oplus \langle s_n^{x_1\ldots x_n} \parallel t_n^{x_1\ldots x_n} \rangle_1 = 0^{\lambda-1} \parallel 0 = 0^\lambda.$$

- $i < n$. In this case, it holds that

$$\begin{aligned}
&\langle s_i^{x_1\ldots x_i} \parallel t_i^{x_1\ldots x_i} \rangle_0 \oplus \langle s_i^{x_1\ldots x_i} \parallel t_i^{x_1\ldots x_i} \rangle_1 \\
= &\ \mathsf{H}_S(\langle s_{i-1}^{x_1\ldots x_{i-1}} \parallel t_{i-1}^{x_1\ldots x_{i-1}} \rangle_0) \oplus x_i \cdot \langle s_{i-1}^{x_1\ldots x_{i-1}} \parallel t_{i-1}^{x_1\ldots x_{i-1}} \rangle_0 \oplus \langle t_{i-1}^{x_1\ldots x_{i-1}} \rangle_0 \cdot \mathsf{CW}_i \\
&\quad \oplus \mathsf{H}_S(\langle s_{i-1}^{x_1\ldots x_{i-1}} \parallel t_{i-1}^{x_1\ldots x_{i-1}} \rangle_1) \oplus x_i \cdot \langle s_{i-1}^{x_1\ldots x_{i-1}} \parallel t_{i-1}^{x_1\ldots x_{i-1}} \rangle_1 \oplus \langle t_{i-1}^{x_1\ldots x_{i-1}} \rangle_1 \cdot \mathsf{CW}_i \\
= &\ \mathsf{H}_S(\langle s_{i-1}^{x_1\ldots x_{i-1}} \parallel t_{i-1}^{x_1\ldots x_{i-1}} \rangle_0) \oplus \mathsf{H}_S(\langle s_{i-1}^{x_1\ldots x_{i-1}} \parallel t_{i-1}^{x_1\ldots x_{i-1}} \rangle_1) \\
&\quad \oplus x_i \cdot (\langle s_{i-1}^{x_1\ldots x_{i-1}} \parallel t_{i-1}^{x_1\ldots x_{i-1}} \rangle_0 \oplus \langle s_{i-1}^{x_1\ldots x_{i-1}} \parallel t_{i-1}^{x_1\ldots x_{i-1}} \rangle_1) \\
&\quad \oplus (\langle t_{i-1}^{x_1\ldots x_{i-1}} \rangle_0 \oplus \langle t_{i-1}^{x_1\ldots x_{i-1}} \rangle_1) \cdot (\mathsf{H}_S(\langle s_{i-1} \parallel t_{i-1} \rangle_0) \oplus \mathsf{H}_S(\langle s_{i-1} \parallel t_{i-1} \rangle_1) \oplus \overline{\alpha}_i \cdot \Delta)
\end{aligned}$$

1. $x_1 \ldots x_{i-1} = \alpha_1 \ldots \alpha_{i-1}$ and $x_i = \alpha_i$ (i.e., $x_1 \ldots x_i = \alpha_1 \ldots \alpha_i$). By the induction assumption, $\langle s_{i-1}^{x_1\ldots x_{i-1}} \parallel t_{i-1}^{x_1\ldots x_{i-1}} \rangle_0 \oplus \langle s_{i-1}^{x_1\ldots x_{i-1}} \parallel t_{i-1}^{x_1\ldots x_{i-1}} \rangle_1 = \Delta$. Claim 2 for $x_1 \ldots x_{i-1} = \alpha_1 \ldots \alpha_{i-1}$ gives

$$\langle s_i^{x_1\ldots x_i} \parallel t_i^{x_1\ldots x_i} \rangle_0 \oplus \langle s_i^{x_1\ldots x_i} \parallel t_i^{x_1\ldots x_i} \rangle_1 = (x_i \oplus \overline{\alpha}_i) \cdot \Delta = \Delta.$$

2. $x_1 \ldots x_{i-1} = \alpha_1 \ldots \alpha_{i-1}$ but $x_i = \overline{\alpha}_i$ (i.e., $x_1 \ldots x_i \neq \alpha_1 \ldots \alpha_i$). It follows from the same analysis as the previous sub-case that

$$\langle s_i^{x_1\ldots x_i} \parallel t_i^{x_1\ldots x_i} \rangle_0 \oplus \langle s_i^{x_1\ldots x_i} \parallel t_i^{x_1\ldots x_i} \rangle_1 = 0 \cdot \Delta = 0^\lambda.$$

3. $x_1 \ldots x_{i-1} \neq \alpha_1 \ldots \alpha_{i-1}$ (i.e., $x_1 \ldots x_i \neq \alpha_1 \ldots \alpha_i$). It follows from the induction assumption that $\langle s_{i-1}^{x_1\ldots x_{i-1}} \parallel t_{i-1}^{x_1\ldots x_{i-1}} \rangle_0 = \langle s_{i-1}^{x_1\ldots x_{i-1}} \parallel t_{i-1}^{x_1\ldots x_{i-1}} \rangle_1$. Thus,

$$\langle s_i^{x_1\ldots x_i} \parallel t_i^{x_1\ldots x_i} \rangle_0 \oplus \langle s_i^{x_1\ldots x_i} \parallel t_i^{x_1\ldots x_i} \rangle_1 = 0^\lambda.$$

Combining the base case and the induction step completes this proof. □

Lemma 7. *For any $n \in \mathbb{N}$, $x, \alpha \in \{0,1\}^n$, let $p \in [0, n]$ such that $x_1 \ldots x_p = \alpha_1 \ldots \alpha_p$ is the longest common prefix of x and α. It holds in* $\mathsf{DCF.Eval}$ *that*

$$V_0^n + V_1^n = \alpha_{p+1} \cdot \beta,$$

where, if $p = n$, we define $\alpha_{p+1} := \alpha_n$ for completeness.

Proof. By the construction, it holds that

$$\begin{aligned}
V_0^n + V_1^n &= \textstyle\sum_{i\in[1,n]} (\mathsf{Convert}_{\mathbb{G}}(\langle v_i^{x_1\ldots x_{i-1}} \rangle_0) + \langle t_{i-1}^{x_1\ldots x_{i-1}} \rangle_0 \cdot \mathsf{VCW}_i) \\
&\quad - \textstyle\sum_{i\in[1,n]} (\mathsf{Convert}_{\mathbb{G}}(\langle v_i^{x_1\ldots x_{i-1}} \rangle_1) + \langle t_{i-1}^{x_1\ldots x_{i-1}} \rangle_1 \cdot \mathsf{VCW}_i) \\
&= \textstyle\sum_{i\in[1,n]} (\mathsf{Convert}_{\mathbb{G}}(\langle v_i^{x_1\ldots x_{i-1}} \rangle_0) - \mathsf{Convert}_{\mathbb{G}}(\langle v_i^{x_1\ldots x_{i-1}} \rangle_1)) \\
&\quad + \textstyle\sum_{i\in[1,n]} ((\langle t_{i-1}^{x_1\ldots x_{i-1}} \rangle_0 - \langle t_{i-1}^{x_1\ldots x_{i-1}} \rangle_1) \cdot \mathsf{VCW}_i) \\
&= \textstyle\sum_{i\in[1,n]} (\mathsf{Convert}_{\mathbb{G}}(\langle v_i^{x_1\ldots x_{i-1}} \rangle_0) - \mathsf{Convert}_{\mathbb{G}}(\langle v_i^{x_1\ldots x_{i-1}} \rangle_1)) \\
&\quad + \textstyle\sum_{i\in[1,n]} \Big((\langle t_{i-1}^{x_1\ldots x_{i-1}} \rangle_0 - \langle t_{i-1}^{x_1\ldots x_{i-1}} \rangle_1) \cdot (\langle t_{i-1} \rangle_0 - \langle t_{i-1} \rangle_1) \\
&\qquad\qquad \cdot (\mathsf{Convert}_{\mathbb{G}}(\langle v_i \rangle_1) - \mathsf{Convert}_{\mathbb{G}}(\langle v_i \rangle_0) + (\alpha_i - \alpha_{i-1}) \cdot \beta) \Big)
\end{aligned} \tag{18}$$

Consider the following two cases:

- $p = n$ (i.e., $x = \alpha$). For every $i \in [1, n]$, we have $x_1 \dots x_{i-1} = \alpha_1 \dots \alpha_{i-1}$ and, from Lemma 6 and Claim 2 for $x_1 \dots x_{i-1} = \alpha_1 \dots \alpha_{i-1}$,

$$\begin{aligned} \forall b \in \{0,1\} : \langle v_i^{x_1 \dots x_{i-1}} \rangle_b = \mathsf{H}_S(\langle s_{i-1}^{x_1 \dots x_{i-1}} \,\|\, t_{i-1}^{x_1 \dots x_{i-1}} \rangle_b \oplus 2) = \mathsf{H}_S(\langle s_{i-1} \,\|\, t_{i-1} \rangle_b \oplus 2) = \langle v_i \rangle_b, \\ (\langle t_{i-1}^{x_1 \dots x_{i-1}} \rangle_0 - \langle t_{i-1}^{x_1 \dots x_{i-1}} \rangle_1) \cdot (\langle t_{i-1} \rangle_0 - \langle t_{i-1} \rangle_1) = (\langle t_{i-1}^{x_1 \dots x_{i-1}} \rangle_0 - \langle t_{i-1}^{x_1 \dots x_{i-1}} \rangle_1)^2 = 1. \end{aligned} \tag{19}$$

  Using (18) and (19), we can see that

$$V_0^n + V_1^n = \sum_{i \in [1,n]} (\alpha_i - \alpha_{i-1}) \cdot \beta = \alpha_n \cdot \beta.$$

- $p \in [0, n-1]$ (i.e., $x \neq \alpha$). For every $i \in [1, p+1]$, we also have $x_1 \dots x_{i-1} = \alpha_1 \dots \alpha_{i-1}$ and (19) from Lemma 6 and Claim 2 for $x_1 \dots x_{i-1} = \alpha_1 \dots \alpha_{i-1}$. Moreover, for every $i \in [p+2, n]$, $x_1 \dots x_{i-1} \neq \alpha_1 \dots \alpha_{i-1}$ and thus

$$\begin{gathered} \langle s_{i-1}^{x_1 \dots x_{i-1}} \,\|\, t_{i-1}^{x_1 \dots x_{i-1}} \rangle_0 = \langle s_{i-1}^{x_1 \dots x_{i-1}} \,\|\, t_{i-1}^{x_1 \dots x_{i-1}} \rangle_1, \\ \langle v_i^{x_1 \dots x_{i-1}} \rangle_0 = \mathsf{H}_S(\langle s_{i-1}^{x_1 \dots x_{i-1}} \,\|\, t_{i-1}^{x_1 \dots x_{i-1}} \rangle_0 \oplus 2) = \mathsf{H}_S(\langle s_{i-1}^{x_1 \dots x_{i-1}} \,\|\, t_{i-1}^{x_1 \dots x_{i-1}} \rangle_1 \oplus 2) = \langle v_i^{x_1 \dots x_{i-1}} \rangle_1 \end{gathered} \tag{20}$$

  by Lemma 6. Using (18), (19), and (20), we have that

$$V_0^n + V_1^n = \sum_{i \in [1,p+1]} (\alpha_i - \alpha_{i-1}) \cdot \beta = \alpha_{p+1} \cdot \beta.$$

The above two cases conclude this proof. □

D.2 Proof of Theorem 3

Theorem 3. *Given CCR function $\mathsf{H} : \mathbb{F}_{2^\lambda} \to \mathbb{F}_{2^\lambda}$, function $\mathsf{Convert}_{\mathbb{G}} : \mathbb{F}_{2^{\lambda-1}} \to \mathbb{G}$, and keyed hash function $\mathsf{H}_S(x) := \mathsf{H}(S \oplus x)$ with some key $S \leftarrow \mathbb{F}_{2^\lambda}$, Figure 8 gives a DPF scheme with domain $[0, N)$ and range $\mathbb{G}$.*

Proof. **Correctness.** The correctness follows from the construction where

$$\begin{aligned} y_0 + y_1 &= (\mathsf{Convert}_{\mathbb{G}}(\langle s_n^x \rangle_0) + \langle t_n^x \rangle_0 \cdot \mathsf{CW}_{n+1}) - (\mathsf{Convert}_{\mathbb{G}}(\langle s_n^x \rangle_1) + \langle t_n^x \rangle_1 \cdot \mathsf{CW}_{n+1}) \\ &= (\mathsf{Convert}_{\mathbb{G}}(\langle s_n^x \rangle_0) - \mathsf{Convert}_{\mathbb{G}}(\langle s_n^x \rangle_1)) + (\langle t_n^x \rangle_0 - \langle t_n^x \rangle_1) \cdot \mathsf{CW}_{n+1} \\ &= (\mathsf{Convert}_{\mathbb{G}}(\langle s_n^x \rangle_0) - \mathsf{Convert}_{\mathbb{G}}(\langle s_n^x \rangle_1)) \\ &\quad + (\langle t_n^x \rangle_0 - \langle t_n^x \rangle_1) \cdot (\langle t_n \rangle_0 - \langle t_n \rangle_1) \cdot (\mathsf{Convert}_{\mathbb{G}}(\langle s_n \rangle_1) - \mathsf{Convert}_{\mathbb{G}}(\langle s_n \rangle_0) + \beta). \end{aligned} \tag{21}$$

If $x = \alpha \in \{0,1\}^n$, Lemma 6 and Claim 2 ensures that

$$\begin{gathered} \forall b \in \{0,1\} : \langle s_n^x \,\|\, t_n^x \rangle_b = \langle s_n \,\|\, t_n \rangle_b, \\ (\langle t_n^x \rangle_0 - \langle t_n^x \rangle_1) \cdot (\langle t_n \rangle_0 - \langle t_n \rangle_1) = (\langle t_n^x \rangle_0 - \langle t_n^x \rangle_1)^2 = 1. \end{gathered}$$

Thus, $y_0 + y_1 = \beta$. Otherwise, using Lemma 6 for $x \neq \alpha$ leads to $y_0 + y_1 = 0$.

Security. Consider the hybrids $\{\mathsf{Hyb}_{n,\mathbb{G},d}\}_{d \in [0,3]}$ in Figure 16. We have the following lemmas.

Lemma 8. *Let $\mathsf{H} : \{0,1\}^\lambda \to \{0,1\}^\lambda$ be a $(T, 4n, \lambda - 1, \epsilon)$-CCR function, χ be a distribution on $\{0,1\}^\lambda$ with min-entropy at least $\lambda - 1$, $S \leftarrow \chi$ be a public key, and $\mathsf{H}_S(x) := \mathsf{H}(S \oplus x)$ for*

$\mathsf{Hyb}_{n,\mathbb{G},d}(1^\lambda, b, \alpha, \beta)$:

Parse $\alpha = (\alpha_1, ..., \alpha_n) \in \{0,1\}^n$ and $\beta \in \mathbb{G}$.
Sample $\Delta \leftarrow \{0,1\}^{\lambda-1} \,\|\, 1$ and $\langle s_0 \,\|\, t_0 \rangle_b \leftarrow \{0,1\}^\lambda$.
for $i \in [1, n-1]$ **do**
  if $d \in \{1,2,3\}$ **then**
    Sample $\mathsf{CW}_i \leftarrow \{0,1\}^\lambda$
  else // $d = 0$
    $\mathsf{CW}_i := \mathsf{H}_S(\langle s_{i-1} \,\|\, t_{i-1} \rangle_b) \oplus \mathsf{H}_S(\langle s_{i-1} \,\|\, t_{i-1} \rangle_b \oplus \Delta) \oplus \overline{\alpha}_i \cdot \Delta$
  $\langle s_i \,\|\, t_i \rangle_b := \mathsf{H}_S(\langle s_{i-1} \,\|\, t_{i-1} \rangle_b) \oplus \alpha_i \cdot \langle s_{i-1} \,\|\, t_{i-1} \rangle_b \oplus \langle t_{i-1} \rangle_b \cdot \mathsf{CW}_i$
if $d \in \{1,2,3\}$ **then**
  Sample $\langle \mathsf{high}^0 \,\|\, \mathsf{low}^0 \rangle_{1-b}, \langle \mathsf{high}^1 \,\|\, \mathsf{low}^1 \rangle_{1-b} \leftarrow \{0,1\}^\lambda$
else // $d = 0$
  $\langle \mathsf{high}^0 \,\|\, \mathsf{low}^0 \rangle_{1-b} := \mathsf{H}_S(\langle s_{n-1} \,\|\, t_{n-1} \rangle_b \oplus \Delta)$
  $\langle \mathsf{high}^1 \,\|\, \mathsf{low}^1 \rangle_{1-b} := \mathsf{H}_S(\langle s_{n-1} \,\|\, t_{n-1} \rangle_b \oplus \Delta \oplus 1)$
if $d \in \{2,3\}$ **then**
  Sample $\mathsf{CW}_n \leftarrow \{0,1\}^{\lambda+1}$ and $\mathsf{HCW}^* \leftarrow \{0,1\}^{\lambda-1}$
else // $d \in \{0,1\}$
  $\langle \mathsf{high}^0 \,\|\, \mathsf{low}^0 \rangle_b := \mathsf{H}_S(\langle s_{n-1} \,\|\, t_{n-1} \rangle_b)$
  $\langle \mathsf{high}^1 \,\|\, \mathsf{low}^1 \rangle_b := \mathsf{H}_S(\langle s_{n-1} \,\|\, t_{n-1} \rangle_b \oplus 1)$
  $\mathsf{HCW} := \langle \mathsf{high}^{\overline{\alpha}_n} \rangle_b \oplus \langle \mathsf{high}^{\overline{\alpha}_n} \rangle_{1-b}$
  $\mathsf{LCW}^0 := \langle \mathsf{low}^0 \rangle_b \oplus \langle \mathsf{low}^0 \rangle_{1-b} \oplus \overline{\alpha}_n$
  $\mathsf{LCW}^1 := \langle \mathsf{low}^1 \rangle_b \oplus \langle \mathsf{low}^1 \rangle_{1-b} \oplus \alpha_n$
  $\mathsf{CW}_n := (\mathsf{HCW} \,\|\, \mathsf{LCW}^0 \,\|\, \mathsf{LCW}^1)$ and $\mathsf{HCW}^* := \langle \mathsf{high}^{\alpha_n} \rangle_b \oplus \langle \mathsf{high}^{\alpha_n} \rangle_{1-b}$
if $d = 3$ **then**
  Sample $\mathsf{CW}_{n+1} \leftarrow \mathbb{G}$
else // $d \in \{0,1,2\}$
  $\langle s_n \,\|\, t_n \rangle_b := \langle \mathsf{high}^{\alpha_n} \,\|\, \mathsf{low}^{\alpha_n} \rangle_b \oplus \langle t_{n-1} \rangle_b \cdot (\mathsf{HCW} \,\|\, \mathsf{LCW}^{\alpha_n})$
  $\mathsf{CW}_{n+1} := (-1)^b \cdot (\langle t_n \rangle_b - (\langle t_n \rangle_b \oplus 1))$
    $\cdot ((-1)^{1-b} \cdot (\mathsf{Convert}_{\mathbb{G}}(\langle s_n \rangle_b) - \mathsf{Convert}_{\mathbb{G}}(\langle s_n \rangle_b \oplus \mathsf{HCW} \oplus \mathsf{HCW}^*)) + \beta)$
return $k_b := (\langle s_0 \,\|\, t_0 \rangle_b, \{\mathsf{CW}_i\}_{i \in [1, n+1]})$

Figure 16: The hybrids for the DPF security.

$x \in \{0,1\}^\lambda$. *There exists such a polynomial* $\mathsf{poly}(\cdot)$ *that, for any* $b \in \{0,1\}$, $(\alpha, \beta) \in \{0,1\}^n \times \mathbb{G}$, *and any PPT adversary* $\mathcal{A}$ *running in time* $T' \le T - \mathsf{poly}(\lambda)$, *it holds that*

$$\Big| \Pr\Big[k_b \leftarrow \mathsf{Hyb}_{n,\mathbb{G},0}(1^\lambda, b, \alpha, \beta) : \mathcal{A}(1^\lambda, k_b) = 1\Big] - \Pr\Big[k_b \leftarrow \mathsf{Hyb}_{n,\mathbb{G},1}(1^\lambda, b, \alpha, \beta) : \mathcal{A}(1^\lambda, k_b) = 1\Big] \Big| \le 2\epsilon + \frac{32n^2}{2^{\lambda+1}}.$$

Proof. Without loss of generality, we fix $b \in \{0,1\}$ and $(\alpha, \beta) \in \{0,1\}^n \times \mathbb{G}$. We assume that, for the sake of contradiction, there exists such an adversary $\mathcal{A}$ that can distinguish the two hybrids with advantage more than ϵ within time T'. We construct the following adversary $\mathcal{B}$ that is given an oracle $\mathcal{O}' \in \{\mathsf{Real}_{\mathsf{H}_S,\Delta}, \mathsf{Ideal}_{\mathsf{H}_S}\}$ and can use $\mathcal{A}$ to break Lemma 1. Let $\mathcal{T} = \{1\}$.

1. $\mathcal{B}$ queries $\mathcal{O}'$ with $Q_1 : \langle s_0 \,\|\, t_0 \rangle_b \leftarrow \{0,1\}^\lambda$. Then, for $i \in [1, n-1]$, it does:
   - Query $\mathcal{O}'$ with $Q_{i,1} : \mathsf{temp}_1 := \mathsf{H}_S(\langle s_{i-1} \,\|\, t_{i-1} \rangle_b)$.
   - Query $\mathcal{O}'$ with $Q_{i,2} : \mathsf{temp}_2 := \mathcal{O}(\langle s_{i-1} \,\|\, t_{i-1} \rangle_b, \overline{\alpha}_i)$.
   - Query $\mathcal{O}'$ with $Q_{i,3} : \mathsf{CW}_i := \mathsf{temp}_1 \oplus \mathsf{temp}_2$.

- If $\alpha_i = 0$ and $\langle t_{i-1}\rangle_b = 0$, regard temp_1 as $\langle s_i \,\|\, t_i\rangle_b$ (without new query).
- If $\alpha_i = 0$ and $\langle t_{i-1}\rangle_b = 1$, regard temp_2 as $\langle s_i \,\|\, t_i\rangle_b$ (without new query).
- If $\alpha_i = 1$ and $\langle t_{i-1}\rangle_b = 0$, query $\mathcal{O}'$ with $Q_{i,4} : \langle s_i \,\|\, t_i\rangle_b := \mathsf{temp}_1 \oplus \langle s_{i-1} \,\|\, t_{i-1}\rangle_b$.
- If $\alpha_i = 1$ and $\langle t_{i-1}\rangle_b = 1$, query $\mathcal{O}'$ with $Q_{i,4} : \langle s_i \,\|\, t_i\rangle_b := \mathsf{temp}_2 \oplus \langle s_{i-1} \,\|\, t_{i-1}\rangle_b$.

Finally, it queries $\mathcal{O}'$ with the following operations:

$$Q_{n,1} : \langle \mathsf{high}^0 \,\|\, \mathsf{low}^0\rangle_{1-b} := \mathcal{O}(\langle s_{n-1} \,\|\, t_{n-1}\rangle_b, 0),$$
$$Q_{n,2} : \mathsf{temp} := \langle s_{n-1} \,\|\, t_{n-1}\rangle_b \oplus 1,$$
$$Q_{n,3} : \langle \mathsf{high}^1 \,\|\, \mathsf{low}^1\rangle_{1-b} := \mathcal{O}(\mathsf{temp}, 0).$$

2. $\mathcal{B}$ runs $k_b \leftarrow \mathsf{Hyb}_{n,\mathbb{G},0}(1^\lambda, b, \alpha, \beta)$ except that it uses the responses from $\mathcal{O}'$ to instantiate the variables under the same names in $\mathsf{Hyb}_{n,\mathbb{G},0}$, instead of evaluating these variables by itself.

3. $\mathcal{B}$ invokes $\mathcal{A}(1^\lambda, k_b)$ and outputs whatever $\mathcal{A}$ outputs.

Since the input length $n = n(\lambda)$, the runtime of the distinguisher $\mathcal{B}$ is bounded by $T' + \mathsf{poly}(\lambda) \leq T$ for some implicit polynomial $\mathsf{poly}(\cdot)$. There are at most $q = 4n$ operations, which are natural and non-trivial as per Definition 3.

Here, χ is a distribution on $\{0,1\}^\lambda$ such that (i) the LSB of any $\Delta \leftarrow \chi$ is 1, and (ii) the high $\lambda - 1$ bits of Δ are uniform. χ has min-entropy $\lambda - 1$. On the one hand, if $\mathcal{B}$ is given $\mathsf{Real}_{\mathsf{H}_S,\Delta}(\cdot)$ for some $\Delta \leftarrow \chi$, then

$$\begin{aligned}\forall i \in [1, n-1] : \mathsf{CW}_i &= \mathsf{H}_S(\langle s_{i-1} \,\|\, t_{i-1}\rangle_b) \oplus \mathsf{H}_S(\langle s_{i-1} \,\|\, t_{i-1}\rangle_b \oplus \Delta) \oplus \overline{\alpha}_i \cdot \Delta,\\ \langle \mathsf{high}^0 \,\|\, \mathsf{low}^0\rangle_{1-b} &= \mathsf{H}_S(\langle s_{n-1} \,\|\, t_{n-1}\rangle_b \oplus \Delta),\\ \langle \mathsf{high}^1 \,\|\, \mathsf{low}^1\rangle_{1-b} &= \mathsf{H}_S(\langle s_{n-1} \,\|\, t_{n-1}\rangle_b \oplus \Delta \oplus 1).\end{aligned}$$

The key k_b in this case is identically distributed as that in $\mathsf{Hyb}_{n,\mathbb{G},0}$. Thus,

$$\Pr_{\Delta \leftarrow \chi}\left[\mathcal{B}^{\mathsf{Real}_{\mathsf{H}_S,\Delta}(\cdot)}(1^\lambda) = 1\right] = \Pr\left[k_b \leftarrow \mathsf{Hyb}_{n,\mathbb{G},0}(1^\lambda, b, \alpha, \beta) : \mathcal{A}(1^\lambda, k_b) = 1\right]. \tag{22}$$

On the other hand, if $\mathcal{B}$ is given $\mathsf{Ideal}_{\mathsf{H}_S}(\cdot)$, then

$$\begin{aligned}\forall i \in [1, n-1] : \mathsf{CW}_i &= \mathsf{H}_S(\langle s_{i-1} \,\|\, t_{i-1}\rangle_b) \oplus \mathcal{O}(\langle s_{i-1} \,\|\, t_{i-1}\rangle_b, \overline{\alpha}_i),\\ \langle \mathsf{high}^0 \,\|\, \mathsf{low}^0\rangle_{1-b} &= \mathcal{O}(\langle s_{n-1} \,\|\, t_{n-1}\rangle_b, 0),\\ \langle \mathsf{high}^1 \,\|\, \mathsf{low}^1\rangle_{1-b} &= \mathcal{O}(\langle s_{n-1} \,\|\, t_{n-1}\rangle_b \oplus 1, 0)\end{aligned}$$

are uniform since $\mathcal{O}$ returns a uniform string upon every invocation. Thus, the key k_b in this case is identically distributed as that in $\mathsf{Hyb}_{n,\mathbb{G},1}$, and

$$\Pr\left[\mathcal{B}^{\mathsf{Ideal}_{\mathsf{H}_S}(\cdot)}(1^\lambda) = 1\right] = \Pr\left[k_b \leftarrow \mathsf{Hyb}_{n,\mathbb{G},1}(1^\lambda, b, \alpha, \beta) : \mathcal{A}(1^\lambda, k_b) = 1\right]. \tag{23}$$

Using the contradiction assumption and (22), (23), we can see that

$$\left|\Pr_{\Delta \leftarrow \chi}\left[\mathcal{B}^{\mathsf{Real}_{\mathsf{H}_S,\Delta}(\cdot)}(1^\lambda) = 1\right] - \Pr\left[\mathcal{B}^{\mathsf{Ideal}_{\mathsf{H}_S}(\cdot)}(1^\lambda) = 1\right]\right| > 2\epsilon + \frac{32n^2}{2^{\lambda+1}},$$

which contradicts with Lemma 1 for the $(T, 4n, \lambda - 1, \epsilon)$-CCR H and $\mathcal{T} = \{1\}$. $\square$

Lemma 9. *For any $b \in \{0,1\}$, and $(\alpha,\beta) \in \{0,1\}^n \times \mathbb{G}$, it holds that*

$$\{k_b \mid k_b \leftarrow \mathsf{Hyb}_{n,\mathbb{G},1}(1^\lambda, b, \alpha, \beta)\} \equiv \{k_b \mid k_b \leftarrow \mathsf{Hyb}_{n,\mathbb{G},2}(1^\lambda, b, \alpha, \beta)\}.$$

Proof. Lemma 9 follows from the fact that, in $\mathsf{Hyb}_{n,\mathbb{G},1}$, the string

$$\langle \mathsf{high}^{\overline{\alpha}_n} \,\|\, \mathsf{low}^0 \,\|\, \mathsf{low}^1 \,\|\, \mathsf{high}^{\alpha_n} \rangle_{1-b}$$

is uniform and serves as an one-time pad for the plaintext

$$\langle \mathsf{high}^{\overline{\alpha}_n} \,\|\, (\mathsf{low}^0 \oplus \overline{\alpha}_n) \,\|\, (\mathsf{low}^1 \oplus \alpha_n) \,\|\, \mathsf{high}^{\alpha_n} \rangle_b.$$

Therefore, the resulting ciphertext

$$(\mathsf{HCW} \,\|\, \mathsf{LCW}^0 \,\|\, \mathsf{LCW}^1 \,\|\, \mathsf{HCW}^*)$$

in $\mathsf{Hyb}_{n,\mathbb{G},1}$ is as uniform as its counterpart in $\mathsf{Hyb}_{n,\mathbb{G},2}$. ☐

Lemma 10. *There exists such a polynomial $\mathsf{poly}_{\mathsf{conv}}(\cdot)$ that, for any $(T_{\mathsf{conv}}, \epsilon_{\mathsf{conv}})$-pseudorandom $\mathsf{Convert}_{\mathbb{G}} : \{0,1\}^{\lambda-1} \to \mathbb{G}$, any $b \in \{0,1\}$, $(\alpha,\beta) \in \{0,1\}^n \times \mathbb{G}$, and any PPT adversary $\mathcal{A}$ running in time $T \leq T_{\mathsf{conv}} - \mathsf{poly}_{\mathsf{conv}}(\lambda)$, it holds that*

$$\begin{aligned}&\Big|\Pr\Big[k_b \leftarrow \mathsf{Hyb}_{n,\mathbb{G},2}(1^\lambda, b, \alpha, \beta) : \mathcal{A}(1^\lambda, k_b) = 1\Big] \\ &\quad - \Pr\Big[k_b \leftarrow \mathsf{Hyb}_{n,\mathbb{G},3}(1^\lambda, b, \alpha, \beta) : \mathcal{A}(1^\lambda, k_b) = 1\Big]\Big| \leq \epsilon_{\mathsf{conv}}.\end{aligned}$$

Proof. Without loss of generality, we fix $b \in \{0,1\}$ and $(\alpha,\beta) \in \{0,1\}^n \times \mathbb{G}$. We assume that, for the sake of contradiction, there exists such an adversary $\mathcal{A}$ that can distinguish the two hybrids with advantage more than ϵ_{conv} within time T. We construct the following adversary $\mathcal{B}$ that has black-box access to the adversary $\mathcal{A}$ and can break the $(T_{\mathsf{conv}}, \epsilon_{\mathsf{conv}})$-pseudorandomness of $\mathsf{Convert}_{\mathbb{G}}$.

1. $\mathcal{B}$ receives an element $r \in \mathbb{G}$ from the challenger.
2. $\mathcal{B}$ follows the steps of $\mathsf{Hyb}_{n,\mathbb{G},2}(1^\lambda, b, \alpha, \beta)$ except that it sets

$$\mathsf{CW}_{n+1} := (-1)^b \cdot (\langle t_n \rangle_b - (\langle t_n \rangle_b \oplus 1)) \cdot ((-1)^{1-b} \cdot (\mathsf{Convert}_{\mathbb{G}}(\langle s_n \rangle_b) - r) + \beta).$$

   In the end, $\mathcal{B}$ generates a key k_b.
3. $\mathcal{B}$ invokes $\mathcal{A}(1^\lambda, k_b)$ and outputs whatever $\mathcal{A}$ outputs.

The runtime of $\mathcal{B}$ is at most $T + \mathsf{poly}_{\mathsf{conv}}(\lambda) \leq T_{\mathsf{conv}}$ for some implicit $\mathsf{poly}_{\mathsf{conv}}(\cdot)$.

If $r := \mathsf{Convert}_{\mathbb{G}}(s)$ for some $s \leftarrow \{0,1\}^{\lambda-1}$, then k_b is identically distributed as that in $\mathsf{Hyb}_{n,\mathbb{G},2}$ since the uniform HCW^* in $\mathsf{Hyb}_{n,\mathbb{G},2}$ gives the uniform $\langle s_n \rangle_b \oplus \mathsf{HCW} \oplus \mathsf{HCW}^*$. Thus,

$$\begin{aligned}&\Pr\Big[s \leftarrow \{0,1\}^{\lambda-1}, r := \mathsf{Convert}_{\mathbb{G}}(s) : \mathcal{B}(1^\lambda, r) = 1\Big] \\ &= \Pr\Big[k_b \leftarrow \mathsf{Hyb}_{n,\mathbb{G},2}(1^\lambda, b, \alpha, \beta) : \mathcal{A}(1^\lambda, k_b) = 1\Big]\end{aligned} \tag{24}$$

Instead, if $r \leftarrow \mathbb{G}$, then the distribution of k_b is identical to that in $\mathsf{Hyb}_{n,\mathbb{G},3}$ since CW_{n+1} in this hybrid has the same uniform distribution as r. Therefore,

$$\Pr\Big[r \leftarrow \mathbb{G} : \mathcal{B}(1^\lambda, r) = 1\Big] = \Pr\Big[k_b \leftarrow \mathsf{Hyb}_{n,\mathbb{G},3}(1^\lambda, b, \alpha, \beta) : \mathcal{A}(1^\lambda, k_b) = 1\Big] \tag{25}$$

Using the contradiction assumption and (24), (25), we can see that

$$\Big|\Pr\Big[s \leftarrow \{0,1\}^{\lambda-1}, r := \mathsf{Convert}_{\mathbb{G}}(s) : \mathcal{B}(1^\lambda, r) = 1\Big] - \Pr\Big[r \leftarrow \mathbb{G} : \mathcal{B}(1^\lambda, r) = 1\Big]\Big| > \epsilon_{\mathsf{conv}},$$

which contradicts with the $(T_{\mathsf{conv}}, \epsilon_{\mathsf{conv}})$-pseudorandomness of $\mathsf{Convert}_{\mathbb{G}}$. □

Lemma 6 gives $\{k_b \mid k_b \leftarrow \mathsf{Hyb}_{n,\mathbb{G},0}(1^\lambda, b, \alpha, \beta)\} \equiv \{k_b \mid (k_0, k_1) \leftarrow \mathsf{DPF.Gen}(1^\lambda, (\alpha, \beta, n, \mathbb{G}))\}$ for any $b \in \{0,1\}$ and $(\alpha, \beta) \in \{0,1\}^n \times \mathbb{G}$. Meanwhile, $\mathsf{Hyb}_{n,\mathbb{G},3}(1^\lambda, b, \alpha, \beta)$ yields a simulator $\mathsf{Sim}(1^\lambda, b, \mathsf{Leak}(f^\bullet_{\alpha,\beta}))$ that outputs a key of $n \cdot \lambda + (\lambda+1) + \log|\mathbb{G}|$ random bits. This theorem immediately follows from Lemma 8, 9 and 10. □

D.3 Proof of Theorem 4

Theorem 4. *Given CCR function* $\mathsf{H} : \mathbb{F}_{2^\lambda} \to \mathbb{F}_{2^\lambda}$, *function* $\mathsf{Convert}_{\mathbb{G}} : \mathbb{F}_{2^\ell} \to \mathbb{G}$ *with* $\ell \in \{\lambda-1, \lambda\}$, *and keyed hash function* $\mathsf{H}_S(x) := \mathsf{H}(S \oplus x)$ *with some key* $S \leftarrow \mathbb{F}_{2^\lambda}$, *Figure 9 gives a DCF scheme with domain* $[0, N)$ *and range* $\mathbb{G}$.

Proof. **Correctness.** In $\mathsf{DCF.Eval}$, it holds that

$$y_0 + y_1 = (\mathsf{DPF.Eval}(0, k'_0, x) + V^n_0) + (\mathsf{DPF.Eval}(1, k'_1, x) + V^n_1)$$

If $x = \alpha \in \{0,1\}^n$, Lemma 7 and the DPF correctness ensure $y_0 + y_1 = (-\alpha_n \cdot \beta) + \alpha_n \cdot \beta = 0$. If $x < \alpha$, there exists such an index $p \in [0, n-1]$ that

$$x_1 \dots x_p = \alpha_1 \dots \alpha_p \ \wedge\ x_{p+1} = 0 \ \wedge\ \alpha_{p+1} = 1.$$

Using Lemma 7 and the DPF correctness, we have $y_0 + y_1 = 0 + \alpha_{p+1} \cdot \beta = \beta$. Similarly, if $x > \alpha$, we can also find an index $p \in [0, n-1]$ that results in

$$x_1 \dots x_p = \alpha_1 \dots \alpha_p \ \wedge\ x_{p+1} = 1 \ \wedge\ \alpha_{p+1} = 0,$$

and $y_0 + y_1 = 0 + \alpha_{p+1} \cdot \beta = 0$ holds according to Lemma 7 and the DPF correctness. The above analysis shows that our DCF scheme is correct.

Security. The following proof is similar to that for our DPF scheme (c.f. Appendix D.2), except that we deal with n VCW_i's in n additional hybrids $\mathsf{Hyb}_{n,\mathbb{G},3+1}, \dots, \mathsf{Hyb}_{n,\mathbb{G},3+n}$. We first consider the hybrids $\{\mathsf{Hyb}_{n,\mathbb{G},d}\}_{d \in [0,3]}$ in Figure 17 and prove the following lemmas.

Lemma 11. *Let* $\mathsf{H} : \{0,1\}^\lambda \to \{0,1\}^\lambda$ *be a* $(T, 6n, \lambda-1, \epsilon)$*-CCR function,* χ *be a distribution on* $\{0,1\}^\lambda$ *with min-entropy at least* $\lambda - 1$, $S \leftarrow \chi$ *be a public key, and* $\mathsf{H}_S(x) := \mathsf{H}(S \oplus x)$ *for* $x \in \{0,1\}^\lambda$. *There exists such a polynomial* $\mathsf{poly}(\cdot)$ *that, for any* $b \in \{0,1\}$, $(\alpha, \beta) \in \{0,1\}^n \times \mathbb{G}$, *and any PPT adversary* $\mathcal{A}$ *running in time* $T' \le T - \mathsf{poly}(\lambda)$, *it holds that*

$$\Big|\Pr\Big[k_b \leftarrow \mathsf{Hyb}_{n,\mathbb{G},0}(1^\lambda, b, \alpha, \beta) : \mathcal{A}(1^\lambda, k_b) = 1\Big] - \Pr\Big[k_b \leftarrow \mathsf{Hyb}_{n,\mathbb{G},1}(1^\lambda, b, \alpha, \beta) : \mathcal{A}(1^\lambda, k_b) = 1\Big]\Big| \le 2\epsilon + \frac{144n^2}{2^{\lambda+1}}.$$

Proof. Without loss of generality, we fix $b \in \{0,1\}$ and $(\alpha, \beta) \in \{0,1\}^n \times \mathbb{G}$. We assume that, for the sake of contradiction, there exists such an adversary $\mathcal{A}$ that can distinguish the two hybrids with advantage more than ϵ within time T'. We can construct the following adversary $\mathcal{B}$ that is given an oracle $\mathcal{O}' \in \{\mathsf{Real}_{\mathsf{H}_S,\Delta}, \mathsf{Ideal}_{\mathsf{H}_S}\}$ and can use $\mathcal{A}$ to break Lemma 1. Let $\mathcal{T} = \{1, 2\}$.

$\mathsf{Hyb}_{n,\mathbb{G},d}(1^\lambda, b, \alpha, \beta)$:

Parse $\alpha = (\alpha_1, ..., \alpha_n) \in \{0,1\}^n$ and $\beta \in \mathbb{G}$.

Sample $\Delta \leftarrow \{0,1\}^{\lambda-1} \,\|\, 1$ and $\langle s_0 \,\|\, t_0 \rangle_b \leftarrow \{0,1\}^\lambda$.

for $i \in [1, n-1]$ **do**

  if $d \in \{1,2,3\}$ **then**

    Sample $\mathsf{CW}_i \leftarrow \{0,1\}^\lambda$ and $\langle v_i \rangle_{1-b} \leftarrow \{0,1\}^\lambda$

  else // $d = 0$

    $\mathsf{CW}_i := \mathsf{H}_S(\langle s_{i-1} \,\|\, t_{i-1} \rangle_b) \oplus \mathsf{H}_S(\langle s_{i-1} \,\|\, t_{i-1} \rangle_b \oplus \Delta) \oplus \overline{\alpha}_i \cdot \Delta$

    $\langle v_i \rangle_{1-b} := \mathsf{H}_S(\langle s_{i-1} \,\|\, t_{i-1} \rangle_b \oplus \Delta \oplus 2)$

  $\langle v_i \rangle_b := \mathsf{H}_S(\langle s_{i-1} \,\|\, t_{i-1} \rangle_b \oplus 2)$

  $\mathsf{VCW}_i := (-1)^b \cdot (\langle t_{i-1} \rangle_b - (\langle t_{i-1} \rangle_b \oplus 1))$
  $\quad \cdot ((-1)^{1-b} \cdot (\mathsf{Convert}_{\mathbb{G}}(\langle v_i \rangle_b) - \mathsf{Convert}_{\mathbb{G}}(\langle v_i \rangle_{1-b})) + (\alpha_i - \alpha_{i-1}) \cdot \beta)$

  $\langle s_i \,\|\, t_i \rangle_b := \mathsf{H}_S(\langle s_{i-1} \,\|\, t_{i-1} \rangle_b) \oplus \alpha_i \cdot \langle s_{i-1} \,\|\, t_{i-1} \rangle_b \oplus \langle t_{i-1} \rangle_b \cdot \mathsf{CW}_i$

if $d \in \{1,2,3\}$ **then**

  Sample $\langle \mathsf{high}^0 \,\|\, \mathsf{low}^0 \rangle_{1-b}, \langle \mathsf{high}^1 \,\|\, \mathsf{low}^1 \rangle_{1-b}, \langle v_n \rangle_{1-b} \leftarrow \{0,1\}^\lambda$

else // $d = 0$

  $\langle \mathsf{high}^0 \,\|\, \mathsf{low}^0 \rangle_{1-b} := \mathsf{H}_S(\langle s_{n-1} \,\|\, t_{n-1} \rangle_b \oplus \Delta)$

  $\langle \mathsf{high}^1 \,\|\, \mathsf{low}^1 \rangle_{1-b} := \mathsf{H}_S(\langle s_{n-1} \,\|\, t_{n-1} \rangle_b \oplus \Delta \oplus 1)$

  $\langle v_n \rangle_{1-b} := \mathsf{H}_S(\langle s_{n-1} \,\|\, t_{n-1} \rangle_b \oplus \Delta \oplus 2)$

if $d \in \{2,3\}$ **then**

  Sample $\mathsf{CW}_n \leftarrow \{0,1\}^{\lambda+1}$ and $\mathsf{HCW}^* \leftarrow \{0,1\}^{\lambda-1}$

else // $d \in \{0,1\}$

  $\langle \mathsf{high}^0 \,\|\, \mathsf{low}^0 \rangle_b := \mathsf{H}_S(\langle s_{n-1} \,\|\, t_{n-1} \rangle_b)$

  $\langle \mathsf{high}^1 \,\|\, \mathsf{low}^1 \rangle_b := \mathsf{H}_S(\langle s_{n-1} \,\|\, t_{n-1} \rangle_b \oplus 1)$

  $\mathsf{HCW} := \langle \mathsf{high}^{\overline{\alpha}_n} \rangle_b \oplus \langle \mathsf{high}^{\overline{\alpha}_n} \rangle_{1-b}$

  $\mathsf{LCW}^0 := \langle \mathsf{low}^0 \rangle_b \oplus \langle \mathsf{low}^0 \rangle_{1-b} \oplus \overline{\alpha}_n$

  $\mathsf{LCW}^1 := \langle \mathsf{low}^1 \rangle_b \oplus \langle \mathsf{low}^1 \rangle_{1-b} \oplus \alpha_n$

  $\mathsf{CW}_n := (\mathsf{HCW} \,\|\, \mathsf{LCW}^0 \,\|\, \mathsf{LCW}^1)$ and $\mathsf{HCW}^* := \langle \mathsf{high}^{\alpha_n} \rangle_b \oplus \langle \mathsf{high}^{\alpha_n} \rangle_{1-b}$

$\langle v_n \rangle_b := \mathsf{H}_S(\langle s_{n-1} \,\|\, t_{n-1} \rangle_b \oplus 2)$

$\mathsf{VCW}_n := (-1)^b \cdot (\langle t_{n-1} \rangle_b - (\langle t_{n-1} \rangle_b \oplus 1))$
$\quad \cdot ((-1)^{1-b} \cdot (\mathsf{Convert}_{\mathbb{G}}(\langle v_n \rangle_b) - \mathsf{Convert}_{\mathbb{G}}(\langle v_n \rangle_{1-b})) + (\alpha_n - \alpha_{n-1}) \cdot \beta)$

if $d = 3$ **then**

  Sample $\mathsf{CW}_{n+1} \leftarrow \mathbb{G}$

else // $d \in \{0,1,2\}$

  $\langle s_n \,\|\, t_n \rangle_b := \langle \mathsf{high}^{\alpha_n} \,\|\, \mathsf{low}^{\alpha_n} \rangle_b \oplus \langle t_{n-1} \rangle_b \cdot (\mathsf{HCW} \,\|\, \mathsf{LCW}^{\alpha_n})$

  $\mathsf{CW}_{n+1} := (-1)^b \cdot (\langle t_n \rangle_b - (\langle t_n \rangle_b \oplus 1))$
  $\quad \cdot ((-1)^{1-b} \cdot (\mathsf{Convert}_{\mathbb{G}}(\langle s_n \rangle_b) - \mathsf{Convert}_{\mathbb{G}}(\langle s_n \rangle_b \oplus \mathsf{HCW} \oplus \mathsf{HCW}^*)) - \alpha_n \cdot \beta)$

return $k_b := (\langle s_0 \,\|\, t_0 \rangle_b, \{\mathsf{CW}_i\}_{i \in [1,n+1]}, \{\mathsf{VCW}_i\}_{i \in [1,n]})$

Figure 17: The hybrids for the DCF security.

1. $\mathcal{B}$ queries $\mathcal{O}'$ with $Q_1 : \langle s_0 \,\|\, t_0 \rangle_b \leftarrow \{0,1\}^\lambda$. Then, for $i \in [1, n-1]$, it does:
   - Query $\mathcal{O}'$ with $Q_{i,1} : \mathsf{temp}_1 := \mathsf{H}_S(\langle s_{i-1} \,\|\, t_{i-1} \rangle_b)$.
   - Query $\mathcal{O}'$ with $Q_{i,2} : \mathsf{temp}_2 := \mathcal{O}(\langle s_{i-1} \,\|\, t_{i-1} \rangle_b, \overline{\alpha}_i)$.
   - Query $\mathcal{O}'$ with $Q_{i,3} : \mathsf{CW}_i := \mathsf{temp}_1 \oplus \mathsf{temp}_2$.
   - Query $\mathcal{O}'$ with $Q_{i,4} : \mathsf{temp}_3 := \langle s_{i-1} \,\|\, t_{i-1} \rangle_b \oplus 2$.
   - Query $\mathcal{O}'$ with $Q_{i,5} : \langle v_i \rangle_{1-b} := \mathcal{O}(\mathsf{temp}_3, 0)$.
   - If $\alpha_i = 0$ and $\langle t_{i-1} \rangle_b = 0$, regard temp_1 as $\langle s_i \,\|\, t_i \rangle_b$ (without new query).

- If $\alpha_i = 0$ and $\langle t_{i-1} \rangle_b = 1$, regard temp_2 as $\langle s_i \,\|\, t_i \rangle_b$ (without new query).
- If $\alpha_i = 1$ and $\langle t_{i-1} \rangle_b = 0$, query $\mathcal{O}'$ with $Q_{i,6} : \langle s_i \,\|\, t_i \rangle_b := \mathsf{temp}_1 \oplus \langle s_{i-1} \,\|\, t_{i-1} \rangle_b$.
- If $\alpha_i = 1$ and $\langle t_{i-1} \rangle_b = 1$, query $\mathcal{O}'$ with $Q_{i,6} : \langle s_i \,\|\, t_i \rangle_b := \mathsf{temp}_2 \oplus \langle s_{i-1} \,\|\, t_{i-1} \rangle_b$.

Finally, it queries $\mathcal{O}'$ with the following operations:

$$\begin{aligned}
&Q_{n,1} : \langle \mathsf{high}^0 \,\|\, \mathsf{low}^0 \rangle_{1-b} := \mathcal{O}(\langle s_{n-1} \,\|\, t_{n-1} \rangle_b, 0),\\
&Q_{n,2} : \mathsf{temp} := \langle s_{n-1} \,\|\, t_{n-1} \rangle_b \oplus 1,\\
&Q_{n,3} : \langle \mathsf{high}^1 \,\|\, \mathsf{low}^1 \rangle_{1-b} := \mathcal{O}(\mathsf{temp}, 0),\\
&Q_{n,4} : \mathsf{temp}' := \langle s_{n-1} \,\|\, t_{n-1} \rangle_b \oplus 2,\\
&Q_{n,5} : \langle v_n \rangle_{1-b} := \mathcal{O}(\mathsf{temp}', 0).
\end{aligned}$$

2. $\mathcal{B}$ runs $k_b \leftarrow \mathsf{Hyb}_{n,\mathbb{G},0}(1^\lambda, b, \alpha, \beta)$ except that it uses the responses from $\mathcal{O}'$ to instantiate the variables under the same names in $\mathsf{Hyb}_{n,\mathbb{G},0}$, instead of evaluating these variables by itself.

3. $\mathcal{B}$ invokes $\mathcal{A}(1^\lambda, k_b)$ and outputs whatever $\mathcal{A}$ outputs.

Since the input length $n = n(\lambda)$, the runtime of the distinguisher $\mathcal{B}$ is bounded by $T' + \mathsf{poly}(\lambda) \leq T$ for some implicit polynomial $\mathsf{poly}(\cdot)$. There are at most $q = 6n$ operations, which are natural and non-trivial as per Definition 3.

Here, χ is a distribution on $\{0,1\}^\lambda$ such that (i) the LSB of any $\Delta \leftarrow \chi$ is 1, and (ii) the high $\lambda - 1$ bits of Δ are uniform. χ has min-entropy $\lambda - 1$. On the one hand, if $\mathcal{B}$ is given $\mathsf{Real}_{\mathsf{H}_S,\Delta}(\cdot)$ for some $\Delta \leftarrow \chi$, then

$$\begin{aligned}
\forall i \in [1, n-1] : \mathsf{CW}_i &= \mathsf{H}_S(\langle s_{i-1} \,\|\, t_{i-1} \rangle_b) \oplus \mathsf{H}_S(\langle s_{i-1} \,\|\, t_{i-1} \rangle_b \oplus \Delta) \oplus \overline{\alpha}_i \cdot \Delta,\\
\langle \mathsf{high}^0 \,\|\, \mathsf{low}^0 \rangle_{1-b} &= \mathsf{H}_S(\langle s_{n-1} \,\|\, t_{n-1} \rangle_b \oplus \Delta),\\
\langle \mathsf{high}^1 \,\|\, \mathsf{low}^1 \rangle_{1-b} &= \mathsf{H}_S(\langle s_{n-1} \,\|\, t_{n-1} \rangle_b \oplus \Delta \oplus 1),\\
\forall i \in [1, n] : \langle v_i \rangle_{1-b} &= \mathsf{H}_S(\langle s_{n-1} \,\|\, t_{n-1} \rangle_b \oplus \Delta \oplus 2).
\end{aligned}$$

The resulting key k_b has the same form as that in $\mathsf{Hyb}_{n,\mathbb{G},0}$. Therefore,

$$\Pr_{\Delta \leftarrow \chi}\left[\mathcal{B}^{\mathsf{Real}_{\mathsf{H}_S,\Delta}(\cdot)}(1^\lambda) = 1\right] = \Pr\left[k_b \leftarrow \mathsf{Hyb}_{n,\mathbb{G},0}(1^\lambda, b, \alpha, \beta) : \mathcal{A}(1^\lambda, k_b) = 1\right]. \tag{26}$$

On the other hand, if $\mathcal{B}$ is given $\mathsf{Ideal}_{\mathsf{H}_S}(\cdot)$, then

$$\begin{aligned}
\forall i \in [1, n-1] : \mathsf{CW}_i &= \mathsf{H}_S(\langle s_{i-1} \,\|\, t_{i-1} \rangle_b) \oplus \mathcal{O}(\langle s_{i-1} \,\|\, t_{i-1} \rangle_b, \overline{\alpha}_i),\\
\langle \mathsf{high}^0 \,\|\, \mathsf{low}^0 \rangle_{1-b} &= \mathcal{O}(\langle s_{n-1} \,\|\, t_{n-1} \rangle_b, 0),\\
\langle \mathsf{high}^1 \,\|\, \mathsf{low}^1 \rangle_{1-b} &= \mathcal{O}(\langle s_{n-1} \,\|\, t_{n-1} \rangle_b \oplus 1, 0),\\
\forall i \in [1, n] : \langle v_i \rangle_{1-b} &= \mathcal{O}(\langle s_{i-1} \,\|\, t_{i-1} \rangle_b \oplus 2, 0),
\end{aligned}$$

are uniform since $\mathcal{O}$ returns a uniform string upon every invocation. Thus, the key k_b in this case is identically distributed as that in $\mathsf{Hyb}_{n,\mathbb{G},1}$, and

$$\Pr\left[\mathcal{B}^{\mathsf{Ideal}_{\mathsf{H}_S}(\cdot)}(1^\lambda) = 1\right] = \Pr\left[k_b \leftarrow \mathsf{Hyb}_{n,\mathbb{G},1}(1^\lambda, b, \alpha, \beta) : \mathcal{A}(1^\lambda, k_b) = 1\right]. \tag{27}$$

Using the contradiction assumption and (26), (27), we can see that

$$\left| \Pr_{\Delta \leftarrow \chi}\left[\mathcal{B}^{\mathsf{Real}_{\mathsf{H}_S,\Delta}(\cdot)}(1^\lambda) = 1\right] - \Pr\left[\mathcal{B}^{\mathsf{Ideal}_{\mathsf{H}_S}(\cdot)}(1^\lambda) = 1\right] \right| > 2\epsilon + \frac{144n^2}{2^{\lambda+1}},$$

which contradicts with Lemma 1 for the $(T, 6n, \lambda - 1, \epsilon)$-CCR H and $\mathcal{T} = \{1, 2\}$. $\square$

Lemma 12. *For any $b \in \{0,1\}$, and $(\alpha, \beta) \in \{0,1\}^n \times \mathbb{G}$, it holds that*

$$\{k_b \mid k_b \leftarrow \mathsf{Hyb}_{n,\mathbb{G},1}(1^\lambda, b, \alpha, \beta)\} \equiv \{k_b \mid k_b \leftarrow \mathsf{Hyb}_{n,\mathbb{G},2}(1^\lambda, b, \alpha, \beta)\}.$$

sketch. Similar to that for Lemma 9 using one-time pad. □

Lemma 13. *There exists such a polynomial $\mathsf{poly}_{\mathsf{conv}}(\cdot)$ that, for any $(T_{\mathsf{conv}}, \epsilon_{\mathsf{conv}})$-pseudorandom $\mathsf{Convert}_{\mathbb{G}} : \{0,1\}^{\lambda-1} \to \mathbb{G}$, any $b \in \{0,1\}$, $(\alpha, \beta) \in \{0,1\}^n \times \mathbb{G}$, and any PPT adversary $\mathcal{A}$ running in time $T \leq T_{\mathsf{conv}} - \mathsf{poly}_{\mathsf{conv}}(\lambda)$, it holds that*

$$\begin{aligned}\Big|\Pr\Big[k_b \leftarrow \mathsf{Hyb}_{n,\mathbb{G},2}(1^\lambda, b, \alpha, \beta) : \mathcal{A}(1^\lambda, k_b) = 1\Big] \\ - \Pr\Big[k_b \leftarrow \mathsf{Hyb}_{n,\mathbb{G},3}(1^\lambda, b, \alpha, \beta) : \mathcal{A}(1^\lambda, k_b) = 1\Big]\Big| \leq \epsilon_{\mathsf{conv}}.\end{aligned}$$

sketch. Similar to that for Lemma 10 which exploits the uniformness of HCW^*, except that the adversary $\mathcal{B}$ sets

$$\mathsf{CW}_{n+1} := (-1)^b \cdot (\langle t_n \rangle_b - (\langle t_n \rangle_b \oplus 1)) \cdot ((-1)^{1-b} \cdot (\mathsf{Convert}_{\mathbb{G}}(\langle s_n \rangle_b) - r) - \alpha_n \cdot \beta)$$

for an element $r \in \mathbb{G}$ received from the challenger. □

Then, we move to consider the subsequent n hybrids $\mathsf{Hyb}_{n,\mathbb{G},d}$ for $d \in [4, n+3]$, where $\mathsf{Hyb}_{n,\mathbb{G},d}$ is identical to $\mathsf{Hyb}_{n,\mathbb{G},d-1}$ except that the correction word $\mathsf{VCW}_{d-3} \in \mathbb{G}$ in $\mathsf{Hyb}_{n,\mathbb{G},d}$ is replaced by a uniform element in $\mathbb{G}$. We prove the following lemma.

Lemma 14. *There exists such a polynomial $\mathsf{poly}'_{\mathsf{conv}}(\cdot)$ that, for any $(T_{\mathsf{conv}}, \epsilon_{\mathsf{conv}})$-pseudorandom $\mathsf{Convert}_{\mathbb{G}} : \{0,1\}^{\lambda} \to \mathbb{G}$, any $d \in [4, n+3]$, $b \in \{0,1\}$, $(\alpha, \beta) \in \{0,1\}^n \times \mathbb{G}$, and any PPT adversary $\mathcal{A}$ running in time $T \leq T_{\mathsf{conv}} - \mathsf{poly}'_{\mathsf{conv}}(\lambda)$, it holds that*

$$\begin{aligned}\Big|\Pr\Big[k_b \leftarrow \mathsf{Hyb}_{n,\mathbb{G},d-1}(1^\lambda, b, \alpha, \beta) : \mathcal{A}(1^\lambda, k_b) = 1\Big] \\ - \Pr\Big[k_b \leftarrow \mathsf{Hyb}_{n,\mathbb{G},d}(1^\lambda, b, \alpha, \beta) : \mathcal{A}(1^\lambda, k_b) = 1\Big]\Big| \leq \epsilon_{\mathsf{conv}}.\end{aligned}$$

Proof. Without loss of generality, we fix $d \in [4, n+3]$, $b \in \{0,1\}$ and $(\alpha, \beta) \in \{0,1\}^n \times \mathbb{G}$. We assume that, for the sake of contradiction, there exists such an adversary $\mathcal{A}$ that can distinguish the two hybrids with advantage more than ϵ_{conv} within time T. We construct the following adversary $\mathcal{B}$ that can use $\mathcal{A}$ to break the $(T_{\mathsf{conv}}, \epsilon_{\mathsf{conv}})$-pseudorandomness of $\mathsf{Convert}_{\mathbb{G}}$.

1. $\mathcal{B}$ receives an element $r \in \mathbb{G}$ from the challenger.
2. $\mathcal{B}$ follows the steps of $\mathsf{Hyb}_{n,\mathbb{G},d-1}(1^\lambda, b, \alpha, \beta)$ except that it sets

$$\begin{aligned}\mathsf{VCW}_{d-3} := (-1)^b \cdot (\langle t_{d-4} \rangle_b - (\langle t_{d-4} \rangle_b \oplus 1)) \\ \cdot ((-1)^{1-b} \cdot (\mathsf{Convert}_{\mathbb{G}}(\langle v_{d-3} \rangle_b) - r) + (\alpha_{d-3} - \alpha_{d-4}) \cdot \beta).\end{aligned}$$

   In the end, $\mathcal{B}$ generates a key k_b.
3. $\mathcal{B}$ invokes $\mathcal{A}(1^\lambda, k_b)$ and outputs whatever $\mathcal{A}$ outputs.

The runtime of $\mathcal{B}$ is at most $T + \mathsf{poly}'_{\mathsf{conv}}(\lambda) \leq T_{\mathsf{conv}}$ for some implicit $\mathsf{poly}'_{\mathsf{conv}}(\cdot)$.

If $r := \mathsf{Convert}_{\mathbb{G}}(s)$ for some $s \leftarrow \{0,1\}^\lambda$, then k_b is identically distributed as in $\mathsf{Hyb}_{n,\mathbb{G},d-1}$ since $\langle v_{d-3}\rangle_{1-b}$ in $\mathsf{Hyb}_{n,\mathbb{G},d-1}$ is as uniform as s, and they lead to the same distribution of VCW_{d-3}. Thus

$$\begin{aligned}&\Pr\left[s \leftarrow \{0,1\}^\lambda, r := \mathsf{Convert}_{\mathbb{G}}(s) : \mathcal{B}(1^\lambda, r) = 1\right] \\ &= \Pr\left[k_b \leftarrow \mathsf{Hyb}_{n,\mathbb{G},d-1}(1^\lambda, b, \alpha, \beta) : \mathcal{A}(1^\lambda, k_b) = 1\right]\end{aligned} \tag{28}$$

Instead, if $r \leftarrow \mathbb{G}$, then the distribution of k_b is identical to that in $\mathsf{Hyb}_{n,\mathbb{G},d}$ since VCW_{d-3} in this hybrid has the same uniform distribution as r. Therefore,

$$\Pr\left[r \leftarrow \mathbb{G} : \mathcal{B}(1^\lambda, r) = 1\right] = \Pr\left[k_b \leftarrow \mathsf{Hyb}_{n,\mathbb{G},d}(1^\lambda, b, \alpha, \beta) : \mathcal{A}(1^\lambda, k_b) = 1\right] \tag{29}$$

Using the contradiction assumption and (28), (29), we can see that

$$\begin{aligned}&\Big|\Pr\left[s \leftarrow \{0,1\}^\lambda, r := \mathsf{Convert}_{\mathbb{G}}(s) : \mathcal{B}(1^\lambda, r) = 1\right] \\ &\quad - \Pr\left[r \leftarrow \mathbb{G} : \mathcal{B}(1^\lambda, r) = 1\right]\Big| > \epsilon_{\mathsf{conv}},\end{aligned}$$

which contradicts with the $(T_{\mathsf{conv}}, \epsilon_{\mathsf{conv}})$-pseudorandomness of $\mathsf{Convert}_{\mathbb{G}}$. □

Lemma 6 gives $\{k_b \mid k_b \leftarrow \mathsf{Hyb}_{n,\mathbb{G},0}(1^\lambda, b, \alpha, \beta)\} \equiv \{k_b \mid (k_0, k_1) \leftarrow \mathsf{DCF.Gen}(1^\lambda, (\alpha, \beta, n, \mathbb{G}))\}$ for any $b \in \{0,1\}$ and $(\alpha, \beta) \in \{0,1\}^n \times \mathbb{G}$. Meanwhile, $\mathsf{Hyb}_{n,\mathbb{G},n+3}(1^\lambda, b, \alpha, \beta)$ yields a simulator $\mathsf{Sim}(1^\lambda, b, \mathsf{Leak}(f^<_{\alpha,\beta}))$ that outputs a key of $n \cdot \lambda + (\lambda+1) + (n+1)\cdot \log|\mathbb{G}|$ random bits. This theorem immediately follows from Lemma 11, 12, 13 and 14. □

D.4 Proof of Theorem 5

Theorem 5. *Given CCR function* $\mathsf{H} : \mathbb{F}_{2^\lambda} \to \mathbb{F}_{2^\lambda}$, *function* $\mathsf{Convert}_{\mathcal{R}} : \mathbb{F}_{2^{\lambda-1}} \to \mathcal{R}$, *and keyed hash function* $\mathsf{H}_S(x) := \mathsf{H}(S \oplus x)$ *with some key* $S \leftarrow \mathbb{F}_{2^\lambda}$, *protocol* Π_{DPF} *(Figure 10) UC-realizes functionality* $\mathcal{F}_{\mathsf{DPF}}$ *(Figure 7) against any semi-honest adversary in the* $(\mathcal{F}_{\mathsf{COT}}, \mathcal{F}_{\mathsf{Rand}}, \mathcal{F}_{\mathsf{OLE}})$*-hybrid model. If* $\mathcal{R} = \mathbb{F}_{2^\ell}$ *for* $\ell \in \mathbb{N}$, *protocol* Π_{DPF} *never invokes* $\mathcal{F}_{\mathsf{OLE}}$.

Proof. We consider polynomially many concurrent **Gen** executions, each of which is implicitly assigned the same session ID but a unique sub-session ID.

Correctness analysis. It would be helpful to show the correctness of protocol Π_{DPF} before this security proof. In the one-time **Initialize** execution of sub-protocol Π_{PREP}, we have

$$\Delta = \langle\Delta\rangle_0 \oplus \langle\Delta\rangle_1 := \Delta'_0 \oplus \Delta'_1 \oplus (0^{\lambda-1} \,\|\, (\mathsf{lsb}(\Delta'_0) \oplus \mathsf{lsb}(\Delta'_1) \oplus 1)) \text{ s.t. } \mathsf{lsb}(\Delta) = 1$$

and, for each **Gen** execution,

$$\begin{aligned}&(\mathsf{K}_{1-b}[\langle\alpha_1\rangle_b], \dots, \mathsf{K}_{1-b}[\langle\alpha_n\rangle_b]) \oplus (\langle\alpha_1\rangle_b, \dots, \langle\alpha_n\rangle_b) \cdot \langle\Delta\rangle_{1-b} \\ := {} & \mathbf{k}_{1-b} \oplus ((\langle\alpha_1\rangle_b, \dots, \langle\alpha_n\rangle_b) \oplus \mathbf{r}_b) \cdot \langle\Delta\rangle_{1-b} \oplus (\langle\alpha_1\rangle_b, \dots, \langle\alpha_n\rangle_b) \cdot \langle\Delta\rangle_{1-b} \\ = {} & \mathbf{k}_{1-b} \oplus \mathbf{r}_b \cdot \left(\Delta'_{1-b} \oplus \left(0^{\lambda-1} \,\|\, (\mathsf{lsb}(\Delta'_b) \oplus (1-b))\right)\right) \\ = {} & (\mathbf{k}_{1-b} \oplus \mathbf{r}_b \cdot \Delta'_{1-b}) \oplus \mathbf{r}_b \cdot \left(0^{\lambda-1} \,\|\, (\mathsf{lsb}(\Delta'_b) \oplus (1-b))\right) \\ = {} & (\mathsf{M}_b[\langle\alpha_1\rangle_b], \dots, \mathsf{M}_b[\langle\alpha_n\rangle_b]).\end{aligned}$$

In each **Gen** execution, we have $\langle s_0^0 \,\|\, t_0^0\rangle_0 \oplus \langle s_0^0 \,\|\, t_0^0\rangle_1 = \Delta$ regardless of W and want to show that each correction word securely computed in Π_{DPF} is well-defined as in $\mathsf{DPF.Gen}$ so that the correctness of our DPF scheme (c.f. Section D.2) applies. For $i \in [1, n-1]$,

$$\begin{aligned}
\mathsf{CW}_i &= \langle \mathsf{CW}_i\rangle_0 \oplus \langle \mathsf{CW}_i\rangle_1 \\
&:= (\oplus_{j\in[0,2^{i-1})}\mathsf{H}_S(\langle s_{i-1}^j \,\|\, t_{i-1}^j\rangle_0)) \oplus \overline{\langle \alpha_i\rangle}_0 \cdot \langle\Delta\rangle_0 \oplus \mathsf{K}_0[\langle\alpha_i\rangle_1] \oplus \mathsf{M}_0[\langle\alpha_i\rangle_0] \\
&\quad\; \oplus (\oplus_{j\in[0,2^{i-1})}\mathsf{H}_S(\langle s_{i-1}^j \,\|\, t_{i-1}^j\rangle_1)) \oplus \overline{\langle \alpha_i\rangle}_1 \cdot \langle\Delta\rangle_1 \oplus \mathsf{K}_1[\langle\alpha_i\rangle_0] \oplus \mathsf{M}_1[\langle\alpha_i\rangle_1] \\
&= \mathsf{H}_S(\langle s_{i-1}^{\alpha_1\ldots\alpha_{i-1}} \,\|\, t_{i-1}^{\alpha_1\ldots\alpha_{i-1}}\rangle_0) \oplus \mathsf{H}_S(\langle s_{i-1}^{\alpha_1\ldots\alpha_{i-1}} \,\|\, t_{i-1}^{\alpha_1\ldots\alpha_{i-1}}\rangle_1) \oplus \overline{\alpha}_i \cdot \Delta. \qquad \text{(By Lemma 6)}
\end{aligned}$$

For $\mathsf{CW}_n = (\mathsf{HCW} \,\|\, \mathsf{LCW}^0 \,\|\, \mathsf{LCW}^1) := \langle \mathsf{CW}_n\rangle_0 \oplus \langle \mathsf{CW}_n\rangle_1$, it holds that

$$\begin{aligned}
\mathsf{HCW} &:= \langle \mathsf{HCW}\rangle_0 \oplus \langle \mathsf{HCW}\rangle_1 \\
&= \langle \mathsf{Xhigh}^{\overline{\langle\alpha_n\rangle_0}}\rangle_0 \oplus \mathsf{H}'(\mu_0 \oplus \mathsf{K}_0[\langle\alpha_n\rangle_1]) \oplus \mathsf{H}'(\mu_1 \oplus \mathsf{M}_0[\langle\alpha_n\rangle_0]) \oplus \langle\alpha_n\rangle_0 \cdot d_1 \\
&\quad\; \oplus \langle \mathsf{Xhigh}^{\overline{\langle\alpha_n\rangle_1}}\rangle_1 \oplus \mathsf{H}'(\mu_1 \oplus \mathsf{K}_1[\langle\alpha_n\rangle_0]) \oplus \mathsf{H}'(\mu_0 \oplus \mathsf{M}_1[\langle\alpha_n\rangle_1]) \oplus \langle\alpha_n\rangle_1 \cdot d_0 \\
&= \langle \mathsf{Xhigh}^{\overline{\langle\alpha_n\rangle_0}}\rangle_0 \oplus \langle \mathsf{Xhigh}^{\overline{\langle\alpha_n\rangle_1}}\rangle_1 \\
&\quad\; \oplus \mathsf{H}'(\mu_1 \oplus \mathsf{K}_1[\langle\alpha_n\rangle_0]) \oplus \mathsf{H}'(\mu_1 \oplus \mathsf{M}_0[\langle\alpha_n\rangle_0]) \oplus \langle\alpha_n\rangle_0 \cdot d_1 \\
&\quad\; \oplus \mathsf{H}'(\mu_0 \oplus \mathsf{K}_0[\langle\alpha_n\rangle_1]) \oplus \mathsf{H}'(\mu_0 \oplus \mathsf{M}_1[\langle\alpha_n\rangle_1]) \oplus \langle\alpha_n\rangle_1 \cdot d_0 \\
&= \langle \mathsf{Xhigh}^{\overline{\langle\alpha_n\rangle_0}}\rangle_0 \oplus \langle \mathsf{Xhigh}^{\overline{\langle\alpha_n\rangle_1}}\rangle_1 \\
&\quad\; \oplus \langle\alpha_n\rangle_0 \cdot (\langle \mathsf{Xhigh}^0\rangle_1 \oplus \langle \mathsf{Xhigh}^1\rangle_1) \oplus \langle\alpha_n\rangle_1 \cdot (\langle \mathsf{Xhigh}^0\rangle_0 \oplus \langle \mathsf{Xhigh}^1\rangle_0) \\
&= \langle \mathsf{Xhigh}^{\overline{\alpha}_n}\rangle_0 \oplus \langle \mathsf{Xhigh}^{\overline{\alpha}_n}\rangle_1 = \langle \mathsf{high}^{\overline{\alpha}_n}\rangle_0 \oplus \langle \mathsf{high}^{\overline{\alpha}_n}\rangle_1, \qquad \text{(By Lemma 6)} \\
\mathsf{LCW}^0 &:= \langle \mathsf{Xlow}^0\rangle_0 \oplus \langle \mathsf{Xlow}^0\rangle_1 \oplus \overline{\alpha}_n = \langle \mathsf{low}^0\rangle_0 \oplus \langle \mathsf{low}^0\rangle_1 \oplus \overline{\alpha}_n, \qquad \text{(By Lemma 6)} \\
\mathsf{LCW}^1 &:= \langle \mathsf{Xlow}^1\rangle_0 \oplus \langle \mathsf{Xlow}^1\rangle_1 \oplus \alpha_n = \langle \mathsf{low}^1\rangle_0 \oplus \langle \mathsf{low}^1\rangle_1 \oplus \alpha_n. \qquad \text{(By Lemma 6)}
\end{aligned}$$

When $\mathcal{R}$ is a binary field (so that $+/-$ is essentially $\oplus$), we have

$$\begin{aligned}
\mathsf{CW}_{n+1} &= \langle \mathsf{CW}_{n+1}\rangle_0^{\mathsf{A}} + \langle \mathsf{CW}_{n+1}\rangle_1^{\mathsf{A}} \\
&:= \left(\sum_{j\in[0,N)} \mathsf{Convert}_{\mathcal{R}}(\langle s_n^j\rangle_0)\right) + \langle\beta\rangle_0^{\mathsf{A}} + \left(\sum_{j\in[0,N)} \mathsf{Convert}_{\mathcal{R}}(\langle s_n^j\rangle_1)\right) + \langle\beta\rangle_1^{\mathsf{A}} \\
&= (\langle t_n\rangle_0 - \langle t_n\rangle_1) \cdot (\mathsf{Convert}_{\mathcal{R}}(\langle s_n\rangle_1) - \mathsf{Convert}_{\mathcal{R}}(\langle s_n\rangle_0) + \beta). \qquad \text{(By Lemma 6)}
\end{aligned}$$

When $\mathcal{R}$ is general, the OLE-based multiplication ensures

$$\begin{aligned}
\mathsf{CW}_{n+1} &= \langle \mathsf{CW}_{n+1}\rangle_0^{\mathsf{A}} + \langle \mathsf{CW}_{n+1}\rangle_1^{\mathsf{A}} \\
&= (\langle A\rangle_0^{\mathsf{A}} + \langle A\rangle_1^{\mathsf{A}}) \cdot (\langle B\rangle_0^{\mathsf{A}} + \langle B\rangle_1^{\mathsf{A}}) \qquad \text{(By OLE-based multiplication)} \\
&= \left(\sum_{j\in[0,N)} \langle t_n^j\rangle_0 - \sum_{j\in[0,N)} \langle t_n^j\rangle_1\right) \cdot \left(\sum_{j\in[0,N)} \mathsf{Convert}_{\mathcal{R}}(\langle s_n^j\rangle_1) - \sum_{j\in[0,N)} \mathsf{Convert}_{\mathcal{R}}(\langle s_n^j\rangle_0) + \beta\right) \\
&= (\langle t_n\rangle_0 - \langle t_n\rangle_1) \cdot (\mathsf{Convert}_{\mathcal{R}}(\langle s_n\rangle_1) - \mathsf{Convert}_{\mathcal{R}}(\langle s_n\rangle_0) + \beta). \qquad \text{(By Lemma 6)}
\end{aligned}$$

The above correction words are those defined in $\mathsf{DPF.Gen}$.

It is clear that the two parties are symmetric in Π_{DPF}. Thus, without loss of generality, we fix $b \in \{0, 1\}$ and consider the case where P_b is corrupted.

Corrupted P_b. In the one-time **Initialize** execution of sub-protocol Π_{PREP},

- Upon receiving (init) from $\mathcal{A}$ to $\mathcal{F}^b_{\mathsf{COT}}$, $\mathcal{S}$ waits for $\mathcal{A}$ to choose Δ'_b.

  Upon receiving (init) from $\mathcal{A}$ to $\mathcal{F}^{1-b}_{\mathsf{COT}}$, $\mathcal{S}$ sends uniform $\mathsf{lsb}(\Delta'_{1-b}) \leftarrow \{0,1\}$ to $\mathcal{A}$.

Then, in each **Gen** execution,

1-1. Upon receiving (extend, n) from $\mathcal{A}$ to $\mathcal{F}^b_{\mathsf{COT}}$, $\mathcal{S}$ waits for $\mathcal{A}$ to choose $\mathbf{k}_b$.

Upon receiving (extend, n) from $\mathcal{A}$ to $\mathcal{F}^{1-b}_{\mathsf{COT}}$, $\mathcal{S}$ waits for $\mathcal{A}$ to choose $(\mathbf{r}_b, \mathbf{m}_b)$.

1-2. $\mathcal{S}$ sends uniform $\mathbf{g}_{1-b} \leftarrow \{0,1\}^n$ to $\mathcal{A}$.

2. Upon receiving $(\mathsf{sample}, \lambda)$ from $\mathcal{A}$ to $\mathcal{F}_{\mathsf{Rand}}$, $\mathcal{S}$ sends uniform $W \leftarrow \{0,1\}^\lambda$ to $\mathcal{A}$.

3. For $i \in [1, n-1]$, $\mathcal{S}$ sends uniform $\langle \mathsf{CW}_i \rangle_{1-b} \leftarrow \{0,1\}^\lambda$ to $\mathcal{A}$.

4. $\mathcal{S}$ sends uniform $(\mu_{1-b}, d_{1-b}) \leftarrow \{0,1\}^\lambda \times \{0,1\}^{\lambda-1}$ to $\mathcal{A}$.

   Then, $\mathcal{S}$ sends uniform $\langle \mathsf{CW}_n \rangle_{1-b} \leftarrow \{0,1\}^{\lambda+1}$ to $\mathcal{A}$.

5. **(Binary field $\mathcal{R} = \mathbb{F}_{2^\ell}$, without $\mathcal{F}_{\mathsf{OLE}}$)**

   $\mathcal{S}$ sends uniform $\langle \mathsf{CW}_{n+1} \rangle^{\mathsf{A}}_{1-b} \leftarrow \mathcal{R}$ to $\mathcal{A}$.

   (General ring $\mathcal{R}$, using $\mathcal{F}_{\mathsf{OLE}}$)

   In sub-protocol Π_{MULT}, $\mathcal{S}$ emulates $\mathcal{F}_{\mathsf{OLE}}$ and sends uniform transcripts to $\mathcal{A}$.

   Then, $\mathcal{S}$ sends uniform $\langle \mathsf{CW}_{n+1} \rangle^{\mathsf{A}}_{1-b} \leftarrow \mathcal{R}$ to $\mathcal{A}$.

6. $\mathcal{S}$ follows protocol Π_{DPF} to compute the output $\langle \mathbf{r} \rangle^{\mathsf{A}}_b$ of the corrupted P_b by using its input $(\langle \alpha \rangle_b, \langle \beta \rangle^{\mathsf{A}}_b)$, its internal randomness extracted from the emulated subroutine ideal functionalities or directly received from $\mathcal{A}$, and the uniform transcripts sent by $\mathcal{S}$.

   Then, $\mathcal{S}$ sends $(\mathsf{gen}, \langle \alpha \rangle_b, \langle \beta \rangle^{\mathsf{A}}_b)$ and $\langle \mathbf{r} \rangle^{\mathsf{A}}_b$ to $\mathcal{F}_{\mathsf{DPF}}$.

We use hybrid argument to prove that the two worlds are computationally indistinguishable.

- Hybrid_0. This is the real world.
- Hybrid_1. This hybrid is identical to the previous one, except that $\mathcal{S}$ emulates $\mathcal{F}^b_{\mathsf{COT}}$, $\mathcal{F}^{1-b}_{\mathsf{COT}}$, $\mathcal{F}_{\mathsf{Rand}}$, and $\mathcal{F}_{\mathsf{OLE}}$, and sends random $\mathsf{lsb}(\Delta'_{1-b})$ in the one-time **Initialize** execution of sub-protocol Π_{PREP}. Moreover, in each **Gen** execution, it is given the corrupted party's input $(\langle \alpha \rangle_b, \langle \beta \rangle^{\mathsf{A}}_b)$ and the honest party's input $(\langle \alpha \rangle_{1-b}, \langle \beta \rangle^{\mathsf{A}}_{1-b})$, and sends to $\mathcal{A}$ the transcripts computed as follows.

  $\mathcal{S}$ has access to the real-world oracle $\mathsf{Real}_{\mathsf{H}_S,\Delta}(\cdot)$ (c.f. Lemma 1) for some $\Delta \leftarrow \{0,1\}^\lambda$ such that $\mathsf{lsb}(\Delta) = 1$ and uses it for all concurrent **Gen** executions. At the beginning of each concurrent execution: First, $\mathcal{S}$ recovers $\alpha = \langle \alpha \rangle_b \oplus \langle \alpha \rangle_{1-b}$ and $\beta = \langle \beta \rangle^{\mathsf{A}}_b + \langle \beta \rangle^{\mathsf{A}}_{1-b}$, and queries $\mathsf{Real}_{\mathsf{H}_S,\Delta}(\cdot)$ with $Q_1 : \langle s_0 \,\|\, t_0 \rangle_b \leftarrow \{0,1\}^\lambda$. Then, for $i \in [1, n-1]$, it does:

  - Query $\mathsf{Real}_{\mathsf{H}_S,\Delta}(\cdot)$ with $Q_{i,1} : \mathsf{temp}_1 := \mathsf{H}_S(\langle s_{i-1} \,\|\, t_{i-1} \rangle_b)$.
  - Query $\mathsf{Real}_{\mathsf{H}_S,\Delta}(\cdot)$ with $Q_{i,2} : \mathsf{temp}_2 := \mathcal{O}(\langle s_{i-1} \,\|\, t_{i-1} \rangle_b, \overline{\alpha}_i)$.
  - Query $\mathsf{Real}_{\mathsf{H}_S,\Delta}(\cdot)$ with $Q_{i,3} : \mathsf{CW}_i := \mathsf{temp}_1 \oplus \mathsf{temp}_2$.
  - If $\alpha_i = 0$ and $\langle t_{i-1} \rangle_b = 0$, regard temp_1 as $\langle s_i \,\|\, t_i \rangle_b$ (without new query).
  - If $\alpha_i = 0$ and $\langle t_{i-1} \rangle_b = 1$, regard temp_2 as $\langle s_i \,\|\, t_i \rangle_b$ (without new query).
  - If $\alpha_i = 1$ and $\langle t_{i-1} \rangle_b = 0$, query $\mathsf{Real}_{\mathsf{H}_S,\Delta}(\cdot)$ with $Q_{i,4} : \langle s_i \,\|\, t_i \rangle_b := \mathsf{temp}_1 \oplus \langle s_{i-1} \,\|\, t_{i-1} \rangle_b$.

– If $\alpha_i = 1$ and $\langle t_{i-1}\rangle_b = 1$, query $\mathsf{Real}_{\mathsf{H}_S,\Delta}(\cdot)$ with $Q_{i,4} : \langle s_i \,\|\, t_i\rangle_b := \mathsf{temp}_2 \oplus \langle s_{i-1} \,\|\, t_{i-1}\rangle_b$.

Finally, it queries $\mathsf{Real}_{\mathsf{H}_S,\Delta}(\cdot)$ with the following operations:

$$\begin{aligned}
&Q_{n,1} : \langle \mathsf{high}^0 \,\|\, \mathsf{low}^0\rangle_{1-b} := \mathcal{O}(\langle s_{n-1} \,\|\, t_{n-1}\rangle_b, 0),\\
&Q_{n,2} : \mathsf{temp} := \langle s_{n-1} \,\|\, t_{n-1}\rangle_b \oplus 1,\\
&Q_{n,3} : \langle \mathsf{high}^1 \,\|\, \mathsf{low}^1\rangle_{1-b} := \mathcal{O}(\mathsf{temp}, 0),\\
&Q_{n,4} : \mathsf{rand} \leftarrow \{0,1\}^\lambda,\\
&Q_{n,5} : \mathsf{pad} := \mathcal{O}(\mathsf{rand}, 0).
\end{aligned}$$

In the rest of this execution, $\mathcal{S}$ uses these oracle responses for the transcript of the honest P_{1-b}. We stress that, *before* $\mathcal{S}$ is required to send some transcript, it can compute the corresponding symmetric transcript to be sent by $\mathcal{A}$ since $\mathcal{S}$ can follow Π_{DPF} to run a copy of the semi-honest P_b *on-the-fly* by using its input, its internal randomness extracted from the emulated subroutine ideal functionalities or directly received from $\mathcal{A}$, and the transcripts sent by $\mathcal{S}$ so far. In details, $\mathcal{S}$ proceeds as follows:

1. $\mathcal{S}$ sends uniform $\mathbf{g}_{1-b}$.
2. $\mathcal{S}$ emulates $\mathcal{F}_{\mathsf{Rand}}$ by sending $W := \langle\Delta\rangle_b \oplus \langle s_0 \,\|\, t_0\rangle_b$, where $\langle\Delta\rangle_b$ is computed by running P_b on-the-fly.
3. For $i \in [1, n-1]$, $\mathcal{S}$ sends $\langle \mathsf{CW}_i\rangle_{1-b} := \mathsf{CW}_i \oplus \langle \mathsf{CW}_i\rangle_b$, where $\langle \mathsf{CW}_i\rangle_b$ is computed by running P_b on-the-fly.
4. $\mathcal{S}$ runs P_b on-the-fly to compute $\mathsf{M}_b[\langle\alpha_n\rangle_b]$ and $\{\langle \mathsf{Xhigh}^\sigma \,\|\, \mathsf{Xlow}^\sigma\rangle_b\}_{\sigma\in\{0,1\}}$. Then, it defines

$$\begin{aligned}
\langle \mathsf{Xhigh}^0 \,\|\, \mathsf{Xlow}^0\rangle_{1-b} &:= \langle \mathsf{Xhigh}^0 \,\|\, \mathsf{Xlow}^0\rangle_b \oplus \mathsf{H}_S(\langle s_{n-1} \,\|\, t_{n-1}\rangle_b) \oplus \langle \mathsf{high}^0 \,\|\, \mathsf{low}^0\rangle_{1-b},\\
\langle \mathsf{Xhigh}^1 \,\|\, \mathsf{Xlow}^1\rangle_{1-b} &:= \langle \mathsf{Xhigh}^1 \,\|\, \mathsf{Xlow}^1\rangle_b \oplus \mathsf{H}_S(\langle s_{n-1} \,\|\, t_{n-1}\rangle_b \oplus 1) \oplus \langle \mathsf{high}^1 \,\|\, \mathsf{low}^1\rangle_{1-b},
\end{aligned}$$

   and sends

$$\begin{aligned}
\mu_{1-b} &:= \mathsf{rand} \oplus \mathsf{M}_b[\langle\alpha_n\rangle_b] \oplus \langle\Delta\rangle_b,\\
d_{1-b} &:= \mathsf{H}'(\mu_{1-b} \oplus \mathsf{M}_b[\langle\alpha_n\rangle_b]) \oplus \mathsf{hb}(\mathsf{pad}) \oplus \langle \mathsf{Xhigh}^0 \oplus \mathsf{Xhigh}^1\rangle_{1-b}.
\end{aligned}$$

5. $\mathcal{S}$ follows $\mathsf{DPF.Gen}$ (c.f. Figure 8) to compute CW_n and CW_{n+1}. Then, $\mathcal{S}$ runs P_b on-the-fly to compute $\langle \mathsf{CW}_n\rangle_b$ and sends $\langle \mathsf{CW}_n\rangle_{1-b} := \mathsf{CW}_n \oplus \langle \mathsf{CW}_n\rangle_b$. Finally, $\mathcal{S}$ sends uniform transcripts in sub-protocol Π_{MULT} if $\mathcal{R}$ is not binary, runs P_b on-the-fly to compute $\langle \mathsf{CW}_{n+1}\rangle_b^{\mathsf{A}}$, and sends $\langle \mathsf{CW}_{n+1}\rangle_{1-b}^{\mathsf{A}} := \mathsf{CW}_{n+1} - \langle \mathsf{CW}_{n+1}\rangle_b^{\mathsf{A}}$.

Note that $\mathsf{lsb}(\Delta'_{1-b})$ is uniform in the two hybrids. In each **Gen** execution, $\mathbf{g}_{1-b}$, W, μ_{1-b}, and the P_{1-b}'s transcripts in sub-protocol Π_{MULT} are also uniform in the two hybrids, and $\{\langle \mathsf{CW}_i\rangle_{1-b}\}_{i\in[1,n+1]}$ and d_{1-b} are equivalently defined as in the previous hybrid. The two hybrids are identically distributed since Δ and the per-execution transcripts have the same distribution.

Now, in each **Gen** execution, $\mathcal{S}$ can (i) follow protocol Π_{DPF} to compute the output $\langle\mathbf{r}\rangle_b^{\mathsf{A}}$ of the corrupted P_b by using its input, its internal randomness extracted from the emulated subroutine ideal functionalities or directly received from $\mathcal{A}$, and the transcripts sent by $\mathcal{S}$, and (ii) send $(\mathsf{gen}, \langle\alpha\rangle_b, \langle\beta\rangle_b^{\mathsf{A}})$ and $\langle\mathbf{r}\rangle_b^{\mathsf{A}}$ to $\mathcal{F}_{\mathsf{DPF}}$, whose output consistency also holds in the previous hybrid due to the well-formed transcript of the semi-honest P_b and the correctness of protocol Π_{DPF}.

- Hybrid_2. This hybrid is identical to the previous one, except that $\mathcal{S}$ has access to the ideal-world oracle $\mathsf{Ideal}_{\mathsf{H}_S}(\cdot)$ instead of the real-world oracle $\mathsf{Real}_{\mathsf{H}_S,\Delta}(\cdot)$ in Lemma 1. It follows from the lemma that this hybrid is computationally indistinguishable from the previous one.

  Now, the per-execution $\{\langle \mathsf{CW}_i \rangle_{1-b}\}_{i\in[1,n]}$ and d_{1-b} can be uniformly sampled instead due to the execution-wise random one-time pad from $\mathsf{Ideal}_{\mathsf{H}_S}(\cdot)$.

- Hybrid_3. This hybrid is identical to the previous one, except that, in each **Gen** execution, $\mathcal{S}$ sends random $\langle \mathsf{CW}_{n+1}\rangle^{\mathsf{A}}_{1-b}$. This hybrid is computationally indistinguishable from the previous one due to the execution-wise pseudorandom one-time pad CW_{n+1} from $\mathsf{Convert}_{\mathcal{R}}$.

  In this hybrid, all transcripts in each **Gen** execution are independent of $(\langle\alpha\rangle_{1-b}, \langle\beta\rangle^{\mathsf{A}}_{1-b})$, the input of the honest P_{1-b} in the execution. It is clear that this hybrid is the ideal world.

The above hybrid argument completes this proof. □

D.5 Proof of Theorem 6

Theorem 6. *Given CCR function* $\mathsf{H} : \mathbb{F}_{2^\lambda} \to \mathbb{F}_{2^\lambda}$, *function* $\mathsf{Convert}_{\mathcal{R}} : \mathbb{F}_{2^\ell} \to \mathcal{R}$ *for* $\ell \in \{\lambda-1, \lambda\}$, *and keyed hash function* $\mathsf{H}_S(x) := \mathsf{H}(S \oplus x)$ *with some key* $S \leftarrow \mathbb{F}_{2^\lambda}$, *protocol* Π_{DCF} *(Figure 13) UC-realizes functionality* $\mathcal{F}_{\mathsf{DCF}}$ *(Figure 7) against any semi-honest adversary in the* $(\mathcal{F}_{\mathsf{COT}}, \mathcal{F}_{\mathsf{Rand}}, \mathcal{F}_{\mathsf{OLE}})$*-hybrid model. If* $\mathcal{R} = \mathbb{F}_{2^\ell}$ *for* $\ell \in \mathbb{N}$, *protocol* Π_{DCF} *never invokes* $\mathcal{F}_{\mathsf{OLE}}$.

Proof. We consider polynomially many concurrent **Gen** executions, each of which is implicitly assigned the same session ID but a unique sub-session ID.

Correctness analysis. The correctness of protocol Π_{DCF} follows from the correctness of Π_{DPF} (c.f. Section D.4), except that we need to show that the correction words $\{\mathsf{VCW}_i\}_{i\in[1,n]}$ are also well-formed as in $\mathsf{DCF.Gen}$. The secure computation of these correction words is similar to that of CW_{n+1}, and the correctness can be checked likewise using Lemma 6, the fact that $\langle v_i^j\rangle_b$ depends on $\langle s^j_{i-1} \,\|\, t^j_{i-1}\rangle_b$, and the correctly shared $\{\alpha_i \cdot \beta\}_{i\in[1,n]}$. We omit the checking for $\{\mathsf{VCW}_i\}_{i\in[1,n]}$ and only check that each $\alpha_i \cdot \beta \in \mathcal{R}$ is correctly shared as follows:

$$
\begin{aligned}
&\langle \alpha_i \cdot \beta\rangle^{\mathsf{A}}_0 + \langle \alpha_i \cdot \beta\rangle^{\mathsf{A}}_1 \\
:=\ & \langle\alpha_i\rangle_0 \cdot \langle\beta\rangle^{\mathsf{A}}_0 - \mathsf{H}^*(x^i_0 \oplus \mathsf{K}_0[\langle\alpha_i\rangle_1]) + \mathsf{H}^*(x^i_1 \oplus \mathsf{M}_0[\langle\alpha_i\rangle_0]) + \langle\alpha_i\rangle_0 \cdot y^i_1 \\
&\quad + \langle\alpha_i\rangle_1 \cdot \langle\beta\rangle^{\mathsf{A}}_1 - \mathsf{H}^*(x^i_1 \oplus \mathsf{K}_1[\langle\alpha_i\rangle_0]) + \mathsf{H}^*(x^i_0 \oplus \mathsf{M}_1[\langle\alpha_i\rangle_1]) + \langle\alpha_i\rangle_1 \cdot y^i_0 \\
=\ & \langle\alpha_i\rangle_0 \cdot \langle\beta\rangle^{\mathsf{A}}_0 + \langle\alpha_i\rangle_1 \cdot \langle\beta\rangle^{\mathsf{A}}_1 \\
&\quad - \mathsf{H}^*(x^i_1 \oplus \mathsf{K}_1[\langle\alpha_i\rangle_0]) + \mathsf{H}^*(x^i_1 \oplus \mathsf{M}_0[\langle\alpha_i\rangle_0]) + \langle\alpha_i\rangle_0 \cdot y^i_1 \\
&\quad - \mathsf{H}^*(x^i_0 \oplus \mathsf{K}_0[\langle\alpha_i\rangle_1]) + \mathsf{H}^*(x^i_0 \oplus \mathsf{M}_1[\langle\alpha_i\rangle_1]) + \langle\alpha_i\rangle_1 \cdot y^i_0 \\
=\ & \langle\alpha_i\rangle_0 \cdot \langle\beta\rangle^{\mathsf{A}}_0 + \langle\alpha_i\rangle_1 \cdot \langle\beta\rangle^{\mathsf{A}}_1 \\
&\quad + \langle\alpha_i\rangle_0 \cdot (\langle\beta\rangle^{\mathsf{A}}_1 - 2 \cdot \langle\alpha_i\rangle_1 \cdot \langle\beta\rangle^{\mathsf{A}}_1) + \langle\alpha_i\rangle_1 \cdot (\langle\beta\rangle^{\mathsf{A}}_0 - 2 \cdot \langle\alpha_i\rangle_0 \cdot \langle\beta\rangle^{\mathsf{A}}_0) \\
=\ & (\langle\alpha_i\rangle_0 + \langle\alpha_i\rangle_1 - 2 \cdot \langle\alpha_i\rangle_0 \cdot \langle\alpha_i\rangle_1) \cdot (\langle\beta\rangle^{\mathsf{A}}_0 + \langle\beta\rangle^{\mathsf{A}}_1) = \alpha_i \cdot \beta.
\end{aligned}
$$

The security proof is similar to Appendix D.4. Assume that P_b is corrupted.

Corrupted P_b. In the one-time **Initialize** execution of sub-protocol Π_{PREP},

- Upon receiving (init) from $\mathcal{A}$ to $\mathcal{F}^b_{\mathsf{COT}}$, $\mathcal{S}$ waits for $\mathcal{A}$ to choose Δ'_b.

  Upon receiving (init) from $\mathcal{A}$ to $\mathcal{F}^{1-b}_{\mathsf{COT}}$, $\mathcal{S}$ sends uniform $\mathsf{lsb}(\Delta'_{1-b}) \leftarrow \{0,1\}$ to $\mathcal{A}$.

Then, in each **Gen** execution,

1-1. Upon receiving (extend, n) from $\mathcal{A}$ to $\mathcal{F}^b_{\mathsf{COT}}$, $\mathcal{S}$ waits for $\mathcal{A}$ to choose $\mathbf{k}_b$.

Upon receiving (extend, n) from $\mathcal{A}$ to $\mathcal{F}^{1-b}_{\mathsf{COT}}$, $\mathcal{S}$ waits for $\mathcal{A}$ to choose $(\mathbf{r}_b, \mathbf{m}_b)$.

1-2. $\mathcal{S}$ sends uniform $\mathbf{g}_{1-b} \leftarrow \{0,1\}^n$ to $\mathcal{A}$.

2. Upon receiving $(\mathsf{sample}, \lambda)$ from $\mathcal{A}$ to $\mathcal{F}_{\mathsf{Rand}}$, $\mathcal{S}$ sends uniform $W \leftarrow \{0,1\}^\lambda$ to $\mathcal{A}$.

3. For $i \in [1, n-1]$, $\mathcal{S}$ sends uniform $(\langle \mathsf{CW}_i \rangle_{1-b}, x^i_{1-b}, y^i_{1-b}) \leftarrow \{0,1\}^\lambda \times \{0,1\}^\lambda \times \mathcal{R}$ to $\mathcal{A}$.

4. $\mathcal{S}$ sends uniform $(\mu_{1-b}, d_{1-b}) \leftarrow \{0,1\}^\lambda \times \{0,1\}^{\lambda-1}$ to $\mathcal{A}$.

   Then, $\mathcal{S}$ sends uniform $(\langle \mathsf{CW}_n \rangle_{1-b}, x^n_{1-b}, y^n_{1-b}) \leftarrow \{0,1\}^{\lambda+1} \times \{0,1\}^\lambda \times \mathcal{R}$ to $\mathcal{A}$.

5. **(Binary field $\mathcal{R} = \mathbb{F}_{2^\ell}$, without $\mathcal{F}_{\mathsf{OLE}}$)**

   $\mathcal{S}$ sends uniform $(\langle \mathsf{CW}_{n+1} \rangle^{\mathsf{A}}_{1-b}, \{\langle \mathsf{VCW}_i \rangle^{\mathsf{A}}_{1-b}\}_{i \in [1,n]}) \leftarrow \mathcal{R}^{n+1}$ to $\mathcal{A}$.

   (General ring $\mathcal{R}$, using $\mathcal{F}_{\mathsf{OLE}}$)

   In sub-protocol Π_{MULT}, $\mathcal{S}$ emulates $\mathcal{F}_{\mathsf{OLE}}$ and sends uniform transcripts to $\mathcal{A}$.

   Then, $\mathcal{S}$ sends uniform $(\langle \mathsf{CW}_{n+1} \rangle^{\mathsf{A}}_{1-b}, \{\langle \mathsf{VCW}_i \rangle^{\mathsf{A}}_{1-b}\}_{i \in [1,n]}) \leftarrow \mathcal{R}^{n+1}$ to $\mathcal{A}$.

6. $\mathcal{S}$ follows protocol Π_{DCF} to compute the output $\langle \mathbf{r} \rangle^{\mathsf{A}}_b$ of the corrupted P_b by using its input $(\langle \alpha \rangle_b, \langle \beta \rangle^{\mathsf{A}}_b)$, its internal randomness extracted from the emulated subroutine ideal functionalities or directly received from $\mathcal{A}$, and the uniform transcripts sent by $\mathcal{S}$.

   Then, $\mathcal{S}$ sends $(\mathsf{gen}, \langle \alpha \rangle_b, \langle \beta \rangle^{\mathsf{A}}_b)$ and $\langle \mathbf{r} \rangle^{\mathsf{A}}_b$ to $\mathcal{F}_{\mathsf{DCF}}$.

We use hybrid argument to prove that the two worlds are computationally indistinguishable.

- Hybrid_0. This is the real world.
- Hybrid_1. This hybrid is identical to the previous one, except that $\mathcal{S}$ emulates $\mathcal{F}^b_{\mathsf{COT}}$, $\mathcal{F}^{1-b}_{\mathsf{COT}}$, $\mathcal{F}_{\mathsf{Rand}}$, and $\mathcal{F}_{\mathsf{OLE}}$, and sends random $\mathsf{lsb}(\Delta'_{1-b})$ in the one-time **Initialize** execution of sub-protocol Π_{PREP}. Moreover, in each **Gen** execution, it is given the corrupted party's input $(\langle \alpha \rangle_b, \langle \beta \rangle^{\mathsf{A}}_b)$ and the honest party's input $(\langle \alpha \rangle_{1-b}, \langle \beta \rangle^{\mathsf{A}}_{1-b})$, and sends to $\mathcal{A}$ the transcripts computed as follows.

  $\mathcal{S}$ has access to the real-world oracle $\mathsf{Real}_{\mathsf{H}_S,\Delta}(\cdot)$ (c.f. Lemma 1) for some $\Delta \leftarrow \{0,1\}^\lambda$ such that $\mathsf{lsb}(\Delta) = 1$ and uses it for all concurrent **Gen** executions. At the beginning of each concurrent execution: First, $\mathcal{S}$ recovers $\alpha = \langle \alpha \rangle_b \oplus \langle \alpha \rangle_{1-b}$ and $\beta = \langle \beta \rangle^{\mathsf{A}}_b + \langle \beta \rangle^{\mathsf{A}}_{1-b}$, and queries $\mathsf{Real}_{\mathsf{H}_S,\Delta}(\cdot)$ with $Q_1 : \langle s_0 \,\|\, t_0 \rangle_b \leftarrow \{0,1\}^\lambda$. Then, for $i \in [1, n-1]$, it does:

  – Query $\mathsf{Real}_{\mathsf{H}_S,\Delta}(\cdot)$ with $Q_{i,1} : \mathsf{temp}_1 := \mathsf{H}_S(\langle s_{i-1} \,\|\, t_{i-1} \rangle_b)$.
  – Query $\mathsf{Real}_{\mathsf{H}_S,\Delta}(\cdot)$ with $Q_{i,2} : \mathsf{temp}_2 := \mathcal{O}(\langle s_{i-1} \,\|\, t_{i-1} \rangle_b, \overline{\alpha}_i)$.
  – Query $\mathsf{Real}_{\mathsf{H}_S,\Delta}(\cdot)$ with $Q_{i,3} : \mathsf{CW}_i := \mathsf{temp}_1 \oplus \mathsf{temp}_2$.
  – Query $\mathsf{Real}_{\mathsf{H}_S,\Delta}(\cdot)$ with $Q_{i,4} : \mathsf{temp}_3 := \langle s_{i-1} \,\|\, t_{i-1} \rangle_b \oplus 2$.
  – Query $\mathsf{Real}_{\mathsf{H}_S,\Delta}(\cdot)$ with $Q_{i,5} : \langle v_i \rangle_{1-b} := \mathcal{O}(\mathsf{temp}_3, 0)$.
  – Query $\mathsf{Real}_{\mathsf{H}_S,\Delta}(\cdot)$ with $Q_{i,6} : \mathsf{rand}_i \leftarrow \{0,1\}^\lambda$.
  – Query $\mathsf{Real}_{\mathsf{H}_S,\Delta}(\cdot)$ with $Q_{i,7} : \mathsf{pad}_i := \mathcal{O}(\mathsf{rand}_i, 0)$.
  – If $\alpha_i = 0$ and $\langle t_{i-1} \rangle_b = 0$, regard temp_1 as $\langle s_i \,\|\, t_i \rangle_b$ (without new query).
  – If $\alpha_i = 0$ and $\langle t_{i-1} \rangle_b = 1$, regard temp_2 as $\langle s_i \,\|\, t_i \rangle_b$ (without new query).

– If $\alpha_i = 1$ and $\langle t_{i-1}\rangle_b = 0$, query $\mathsf{Real}_{\mathsf{H}_S,\Delta}(\cdot)$ with $Q_{i,8} : \langle s_i \,\|\, t_i\rangle_b := \mathsf{temp}_1 \oplus \langle s_{i-1} \,\|\, t_{i-1}\rangle_b$.
– If $\alpha_i = 1$ and $\langle t_{i-1}\rangle_b = 1$, query $\mathsf{Real}_{\mathsf{H}_S,\Delta}(\cdot)$ with $Q_{i,8} : \langle s_i \,\|\, t_i\rangle_b := \mathsf{temp}_2 \oplus \langle s_{i-1} \,\|\, t_{i-1}\rangle_b$.

Finally, it queries $\mathsf{Real}_{\mathsf{H}_S,\Delta}(\cdot)$ with the following operations:

$$\begin{aligned}
&Q_{n,1} : \langle \mathsf{high}^0 \,\|\, \mathsf{low}^0\rangle_{1-b} := \mathcal{O}(\langle s_{n-1} \,\|\, t_{n-1}\rangle_b, 0),\\
&Q_{n,2} : \mathsf{temp} := \langle s_{n-1} \,\|\, t_{n-1}\rangle_b \oplus 1,\\
&Q_{n,3} : \langle \mathsf{high}^1 \,\|\, \mathsf{low}^1\rangle_{1-b} := \mathcal{O}(\mathsf{temp}, 0),\\
&Q_{n,4} : \mathsf{temp}' := \langle s_{n-1} \,\|\, t_{n-1}\rangle_b \oplus 2,\\
&Q_{n,5} : \langle v_n\rangle_{1-b} := \mathcal{O}(\mathsf{temp}', 0),\\
&Q_{n,6} : \mathsf{rand} \leftarrow \{0,1\}^\lambda,\\
&Q_{n,7} : \mathsf{pad} := \mathcal{O}(\mathsf{rand}, 0),\\
&Q_{n,8} : \mathsf{rand}_n \leftarrow \{0,1\}^\lambda,\\
&Q_{n,9} : \mathsf{pad}_n := \mathcal{O}(\mathsf{rand}_n, 0).
\end{aligned}$$

In the rest of this execution, $\mathcal{S}$ uses these oracle responses for the transcript of the honest P_{1-b}. We stress that, *before* $\mathcal{S}$ is required to send some transcript, it can compute the corresponding symmetric transcript to be sent by $\mathcal{A}$ since $\mathcal{S}$ can follow Π_{DCF} to run a copy of the semi-honest P_b *on-the-fly* by using its input, its internal randomness extracted from the emulated subroutine ideal functionalities or directly received from $\mathcal{A}$, and the transcripts sent by $\mathcal{S}$ so far. In details, $\mathcal{S}$ proceeds as follows:

1. $\mathcal{S}$ sends uniform $\mathbf{g}_{1-b}$.
2. $\mathcal{S}$ emulates $\mathcal{F}_{\mathsf{Rand}}$ by sending $W := \langle\Delta\rangle_b \oplus \langle s_0 \,\|\, t_0\rangle_b$, where $\langle\Delta\rangle_b$ is computed by running P_b on-the-fly.
3. For $i \in [1, n-1]$, $\mathcal{S}$ runs P_b on-the-fly to compute $\mathsf{M}_b[\langle\alpha_i\rangle_b]$ and $\langle\mathsf{CW}_i\rangle_b$, and sends

$$\begin{aligned}
\langle\mathsf{CW}_i\rangle_{1-b} &:= \mathsf{CW}_i \oplus \langle\mathsf{CW}_i\rangle_b,\\
x^i_{1-b} &:= \mathsf{rand}_i \oplus \mathsf{M}_b[\langle\alpha_i\rangle_b] \oplus \langle\Delta\rangle_b,\\
y^i_{1-b} &:= (-1)^{\langle\alpha_i\rangle_b} \cdot \big(\mathsf{H}^*(x^i_{1-b} \oplus \mathsf{M}_b[\langle\alpha_i\rangle_b]) - \mathsf{Convert}_{\mathcal{R}}(\mathsf{pad}_i)\big)\\
&\qquad + \langle\beta\rangle^{\mathsf{A}}_{1-b} - 2\cdot\langle\alpha_i\rangle_{1-b}\cdot\langle\beta\rangle^{\mathsf{A}}_{1-b}.
\end{aligned}$$

4. $\mathcal{S}$ runs P_b on-the-fly to compute $\mathsf{M}_b[\langle\alpha_n\rangle_b]$ and $\{\langle\mathsf{Xhigh}^\sigma \,\|\, \mathsf{Xlow}^\sigma\rangle_b\}_{\sigma\in\{0,1\}}$. Then, it defines

$$\begin{aligned}
\langle\mathsf{Xhigh}^0 \,\|\, \mathsf{Xlow}^0\rangle_{1-b} &:= \langle\mathsf{Xhigh}^0 \,\|\, \mathsf{Xlow}^0\rangle_b \oplus \mathsf{H}_S(\langle s_{n-1} \,\|\, t_{n-1}\rangle_b) \oplus \langle\mathsf{high}^0 \,\|\, \mathsf{low}^0\rangle_{1-b},\\
\langle\mathsf{Xhigh}^1 \,\|\, \mathsf{Xlow}^1\rangle_{1-b} &:= \langle\mathsf{Xhigh}^1 \,\|\, \mathsf{Xlow}^1\rangle_b \oplus \mathsf{H}_S(\langle s_{n-1} \,\|\, t_{n-1}\rangle_b \oplus 1) \oplus \langle\mathsf{high}^1 \,\|\, \mathsf{low}^1\rangle_{1-b},
\end{aligned}$$

and sends

$$\begin{aligned}
\mu_{1-b} &:= \mathsf{rand} \oplus \mathsf{M}_b[\langle\alpha_n\rangle_b] \oplus \langle\Delta\rangle_b,\\
d_{1-b} &:= \mathsf{H}'(\mu_{1-b} \oplus \mathsf{M}_b[\langle\alpha_n\rangle_b]) \oplus \mathsf{hb}(\mathsf{pad}) \oplus \langle\mathsf{Xhigh}^0 \oplus \mathsf{Xhigh}^1\rangle_{1-b},\\
x^n_{1-b} &:= \mathsf{rand}_n \oplus \mathsf{M}_b[\langle\alpha_n\rangle_b] \oplus \langle\Delta\rangle_b,\\
y^n_{1-b} &:= (-1)^{\langle\alpha_n\rangle_b} \cdot \big(\mathsf{H}^*(x^n_{1-b} \oplus \mathsf{M}_b[\langle\alpha_n\rangle_b]) - \mathsf{Convert}_{\mathcal{R}}(\mathsf{pad}_n)\big)\\
&\qquad + \langle\beta\rangle^{\mathsf{A}}_{1-b} - 2\cdot\langle\alpha_n\rangle_{1-b}\cdot\langle\beta\rangle^{\mathsf{A}}_{1-b}.
\end{aligned}$$

5. $\mathcal{S}$ follows $\mathsf{DCF.Gen}$ (c.f. Figure 9) to compute CW_n, CW_{n+1}, and $\{\mathsf{VCW}_i\}_{i\in[1,n]}$. Then, $\mathcal{S}$ runs P_b on-the-fly to compute $\langle\mathsf{CW}_n\rangle_b$ and sends $\langle\mathsf{CW}_n\rangle_{1-b} := \mathsf{CW}_n \oplus \langle\mathsf{CW}_n\rangle_b$. Finally, $\mathcal{S}$ sends uniform transcripts in sub-protocol Π_{MULT} if $\mathcal{R}$ is not binary, runs P_b on-the-fly to compute $\langle\mathsf{CW}_{n+1}\rangle_b^{\mathsf{A}}$ and $\{\langle\mathsf{VCW}_i\rangle_b^{\mathsf{A}}\}_{i\in[1,n]}$, and sends

$$\langle\mathsf{CW}_{n+1}\rangle_{1-b}^{\mathsf{A}} := \mathsf{CW}_{n+1} - \langle\mathsf{CW}_{n+1}\rangle_b^{\mathsf{A}},$$
$$\forall i \in [1,n] : \langle\mathsf{VCW}_i\rangle_{1-b}^{\mathsf{A}} := \mathsf{VCW}_i - \langle\mathsf{VCW}_i\rangle_b^{\mathsf{A}}.$$

Note that $\mathsf{lsb}(\Delta'_{1-b})$ is uniform in the two hybrids. In each **Gen** execution, $\mathbf{g}_{1-b}$, W, $\{x^i_{1-b}\}_{i\in[1,n]}$, μ_{1-b}, and the P_{1-b}'s transcripts in sub-protocol Π_{MULT} are also uniform in the two hybrids, and $\{\langle\mathsf{CW}_i\rangle_{1-b}\}_{i\in[1,n+1]}$, $\{\langle\mathsf{VCW}_i\rangle_{1-b}^{\mathsf{A}}\}_{i\in[1,n]}$, $\{y^i_{1-b}\}_{i\in[1,n]}$, and d_{1-b} are equivalently defined as in the previous hybrid. The two hybrids are identically distributed since Δ and the per-execution transcripts have the same distribution.

Now, in each **Gen** execution, $\mathcal{S}$ can (i) follow protocol Π_{DCF} to compute the output $\langle\mathbf{r}\rangle_b^{\mathsf{A}}$ of the corrupted P_b by using its input, its internal randomness extracted from the emulated subroutine ideal functionalities or directly received from $\mathcal{A}$, and the transcripts sent by $\mathcal{S}$, and (ii) send $(\mathsf{gen}, \langle\alpha\rangle_b, \langle\beta\rangle_b^{\mathsf{A}})$ and $\langle\mathbf{r}\rangle_b^{\mathsf{A}}$ to $\mathcal{F}_{\mathsf{DCF}}$, whose output consistency also holds in the previous hybrid due to the well-formed transcript of the semi-honest P_b and the correctness of protocol Π_{DCF}.

- Hybrid_2. This hybrid is identical to the previous one, except that $\mathcal{S}$ has access to the ideal-world oracle $\mathsf{Ideal}_{\mathsf{H}_S}(\cdot)$ instead of the real-world oracle $\mathsf{Real}_{\mathsf{H}_S,\Delta}(\cdot)$ in Lemma 1. It follows from the lemma that this hybrid is computationally indistinguishable from the previous one.

  Now, the per-execution $\{\langle\mathsf{CW}_i\rangle_{1-b}\}_{i\in[1,n]}$, $\{\mathsf{pad}_i\}_{i\in[1,n]}$, $\{\langle v_i\rangle_{1-b}\}_{i\in[1,n]}$, and d_{1-b} can be uniformly sampled instead due to the execution-wise uniform one-time pad from $\mathsf{Ideal}_{\mathsf{H}_S}(\cdot)$.

- Hybrid_3. This hybrid is identical to the previous one, except that, in each **Gen** execution, $\mathcal{S}$ sends random $\langle\mathsf{CW}_{n+1}\rangle_{1-b}^{\mathsf{A}}$, $\{y^i_{1-b}\}_{i\in[1,n]}$, and $\{\langle\mathsf{VCW}_i\rangle_{1-b}^{\mathsf{A}}\}_{i\in[1,n]}$. This hybrid is computationally indistinguishable from the previous one due to the execution-wise pseudorandom one-time pad CW_{n+1}, $\{\mathsf{Convert}_{\mathcal{R}}(\mathsf{pad}_i)\}_{i\in[1,n]}$, and $\{\mathsf{VCW}_i\}_{i\in[1,n]}$ from $\mathsf{Convert}_{\mathcal{R}}$.

  In this hybrid, all transcripts in each **Gen** execution are independent of $(\langle\alpha\rangle_{1-b}, \langle\beta\rangle_{1-b}^{\mathsf{A}})$, the input of the honest P_{1-b} in the execution. It is clear that this hybrid is the ideal world.

The above hybrid argument completes this proof. □

E PCG and FSS Protocols

In the main body of this paper, we present distributed protocols to produce the fully expanded correlations of popular PCGs (i.e., COT and sVOLE) and FSSs (i.e., DPF and DCF). In this appendix, we discuss how to construct distributed protocols that only output *sublinearly short* forms of such correlations.

E.1 (Single-point) Subfield VOLE with Silent Preprocessing

In a PCG scheme [BCG+19b, BCG+19a, BCG+20], the correlation generation consists of a seed generation algorithm and a seed expansion algorithm. The generation algorithm produces two short PCG seeds, and the expansion algorithm expands each seed into a long correlation share. In the 2PC setting where no trusted dealer runs the generation algorithm, two parties run a PCG protocol that securely computes this generation algorithm so that each party obtains its PCG seed. The subsequent seed expansion can be locally done by each party.

The silent preprocessing feature in the PCG protocol requires that the (online) generation phase consumes sublinear communication and produces two sublinearly short PCG seeds, which are to be locally expanded on demand.

As our pcGGM tree yields PPRF (see Appendix C) under the CCR assumption, one can prove that this pcGGM tree also yields (single-point) sVOLE PCG scheme via a similar proof to that in [BCG+19a]. In our pcGGM-based spsVOLE PCG, the sender's PCG seed includes (Δ, k) and the global key Γ, and the receiver's PCG seed is $(\alpha, \{K_i^{\overline{\alpha}_i}\}_{i\in[1,n]}, \beta, \psi + \mathsf{M}[\beta])$, where the transcripts are defined as per our spsVOLE protocol $\Pi_{\mathsf{spsVOLE-pcGGM}}$ (Figure 6). The expansion algorithm per party is straightforward.

We can adapt $\Pi_{\mathsf{spsVOLE-pcGGM}}$ into a silent spsVOLE PCG protocol by simply outputting the PCG seeds without expansion, but there are some subtle issues to be explained. First, in the silent spsVOLE PCG protocol with the corrupted receiver, the honest sender's output includes the global offset Δ so that this offset is no longer hidden from the view of the environment $\mathcal{Z}$ as in $\Pi_{\mathsf{spsVOLE-pcGGM}}$. As a result, we cannot prove the pseudorandomness of protocol transcripts under the CCR assumption. However, this challenge is reminiscent of that confronted in our cGGM-based COT/sVOLE protocol. Intuitively, we can also address issue by resorting to the programmable random permutation and a relaxed PCG seed generation functionality with the global-key queries of Δ.

Second, the silent spsVOLE PCG protocol requires one additional $\mathcal{F}_{\mathsf{COT}}$ instance for the last-level transcript in $\Pi_{\mathsf{spsVOLE-pcGGM}}$. Observe that the last-level string OT with the payload (c_n^0, c_n^1) in $\Pi_{\mathsf{spsVOLE-pcGGM}}$ is emulated by a precomputed COT tuple under the global key Δ. When the receiver is corrupted, the simulator cannot learn $K_n^{\alpha_n}$ from the receiver's PCG seed in the ideal world. Thus, the simulator can only simulate $c_n^{\alpha_n}$ with a random value, which will be detected by $\mathcal{Z}$ checking its consistency with the Δ in the sender's PCG seed. A fix to this issue is to use another independent $\mathcal{F}_{\mathsf{COT}}$ instance with another independent global key Δ' and use Δ' to emulate the last-level OT. Since Δ' is hidden from $\mathcal{Z}$, the pseudorandomness of $K_n^{\alpha_n}$ follows from the correlation robustness of hash function.

By running t concurrent **Extend** executions in this silent spsVOLE PCG protocol and using LPN encoding, we obtain a silent sVOLE PCG protocol. We note that, different from the sVOLE PCG protocol in [BCG+19a], our protocol reuses the same global offset Δ across polynomially many PCG seeds.

E.2 FSS Key Generation

In contrast to the functionality $\mathcal{F}_{\mathsf{FSS}}$ for FSS correlation generation, the functionality for FSS key generation distributes a pair of FSS keys to the two parties. It is required for an FSS key generation protocol to realize this functionality with communication sublinear in the domain size of functions.

We want to adapt our two FSS correlation generation protocols, Π_{DPF} (Figure 10) and Π_{DCF} (Figure 13), into the FSS key generation protocols by simply outputting the FSS keys computed therein. However, we again confront the subtle issue that the global offset Δ is available to the environment $\mathcal{Z}$ since it can observe the outputs (i.e., FSS keys) of the two parties. This issue can be addressed by using the programmable random permutation and a relaxed FSS key generation functionality allowing the queries of Δ.

Moreover, the adaption of each FSS key generation protocol also requires one additional $\mathcal{F}_{\mathsf{COT}}$ instance per party. Consider the DPF key generation protocol $\Pi_{\mathsf{DPF-Gen}}$, which follows Π_{DPF} but only outputs the two parties's keys. $\Pi_{\mathsf{DPF-Gen}}$ should change the way to compute the intermediate transcript d_{1-b} for each $b \in \{0,1\}$. The reason is that, in the DPF key generation, the environment $\mathcal{Z}$ is given both parties' DPF keys. Now, the environment $\mathcal{Z}$ that corrupts P_b can compute $\langle\Delta\rangle_{1-b}$ (which underlies d_{1-b}) from the key k_{1-b} output by P_{1-b} and the public randomness W output by $\mathcal{F}_{\mathsf{Rand}}$. Unfortunately, the simulator with k_b and the transcripts of subroutine functionalities in this case cannot simulate the d_{1-b} consistent with the ideal-world $\langle\Delta\rangle_{1-b}$ since k_b reveals no information about $\langle\mathsf{high}^{\alpha_n}\rangle_{1-b}$ (or rather, $\langle\mathsf{Xhigh}^{\alpha_n}\rangle_{1-b}$).

To address this issue, we can use the same technique as in our single-point sVOLE PCG (c.f. Appendix E.1). That is, each party can invoke an independent $\mathcal{F}_{\mathsf{COT}}$ instance with a global key Δ_b^* and use the COT tuples precomputed from this instance to define d_{1-b}. Since $\mathcal{Z}$ cannot see Δ_b^*, the simulator can simulate d_{1-b} by a random value due to the correlation robustness of hash function.

Likewise, we can construct a DCF key generation protocol $\Pi_{\mathsf{DCF-Gen}}$, which follows Π_{DCF} except that each party P_b uses one additional $\mathcal{F}_{\mathsf{COT}}$ instance to define its intermediate transcripts d_b and $\{y_b^i\}_{i\in[1,n]}$. Each DCF key k_b to be output by $\Pi_{\mathsf{DCF-Gen}}$ is implicitly computed in Π_{DCF}.

F Supplementary Preliminaries

F.1 Pseudorandom Conversion

Implementation of $\mathsf{Convert}_{\mathbb{G}}$. This function maps ℓ-bit random strings into pseudorandom elements in $\mathbb{G}$ and can be implemented as follows. Let $\rho \in \mathbb{N}$ be the statistical security parameter. On input an ℓ-bit random string x: if $(2^\ell \bmod |\mathbb{G}|)/2^\ell \leq 2^{-\rho}$,[4] it computes the remainder $x \bmod |\mathbb{G}|$ and outputs this remainder as a group element in $\mathbb{G}$; otherwise, it computes $G_{\mathsf{ext}}(x) \bmod |\mathbb{G}|$ for some PRG $G_{\mathsf{ext}} : \{0,1\}^\ell \to \{0,1\}^{\ell_0}$, where $\ell_0 := \lceil \log |\mathbb{G}| \rceil + \rho$, and outputs this remainder as a group element in $\mathbb{G}$. In either case, the bias is bounded by $2^{-\rho}$.

We focus on $\ell = \lambda - 1$ by default in the following definition but slightly abuse this definition in our pcGGM and DCF to also convert a random string with $\ell = \lambda$ bits. We stress that this abuse does not affect the security since one can always discard the LSB of a λ-bit random string to fit this string for the following $\mathsf{Convert}_{\mathbb{G}}$. This abuse is just for the simplicity of exposition.

Definition 5 (Pseudorandomness of $\mathsf{Convert}_{\mathbb{G}}$)**.** $\mathsf{Convert}_{\mathbb{G}} : \{0,1\}^{\lambda-1} \to \mathbb{G}$ *is (t,ϵ)-pseudorandom if, for any distinguisher $\mathcal{D}$ running in time at most t, and any finite group $\mathbb{G}$, it holds that*

$$\Big|\Pr\Big[s \leftarrow \{0,1\}^{\lambda-1}, r := \mathsf{Convert}_{\mathbb{G}}(s) : \mathcal{D}(1^\lambda, r) = 1\Big] \\ - \Pr\Big[r \leftarrow \mathbb{G} : \mathcal{D}(1^\lambda, r) = 1\Big]\Big| \leq \epsilon.$$

Theorem 9. *Let $\rho \in \mathbb{N}$ be the statistical security parameter, $\mathbb{G}$ be an arbitrary finite group, $G_{\mathsf{ext}} : \{0,1\}^{\lambda-1} \to \{0,1\}^{\ell_0}$ be a (t,ϵ)-secure PRG where $\ell_0 := \lceil \log |\mathbb{G}| \rceil + \rho$. There exists such a polynomial $\mathsf{poly}(\cdot)$ that the implementation of $\mathsf{Convert}_{\mathbb{G}}$ in this appendix is (t',ϵ')-pseudorandom as per Definition 5, where*

$$(t',\epsilon') = \begin{cases} (\infty, 2^{-\rho}) & , \textit{if } \frac{2^{\lambda-1} \bmod |\mathbb{G}|}{2^{\lambda-1}} \leq 2^{-\rho} \\ (t - \mathsf{poly}(\lambda), \epsilon + 2^{-\rho}) & , \textit{otherwise} \end{cases}.$$

Proof. Define $M := 2^{\lambda-1} \bmod |\mathbb{G}|$. Consider two cases regarding $\mathbb{G}$:

- **Case (i):** $M/2^{\lambda-1} \leq 2^{-\rho}$**.** Let X (resp., Y) denote the uniform distribution of $s \leftarrow [0, 2^{\lambda-1})$ (resp., $s \leftarrow [0, 2^{\lambda-1} - M)$), where s can be viewed as a string in $\{0,1\}^{\lambda-1}$. The statistical distance

$$\begin{aligned} \mathsf{SD}(X,Y) &= \frac{1}{2} \cdot \sum_{s' \in \mathsf{supp}(X) \cup \mathsf{supp}(Y)} \left|\Pr\left[X = s'\right] - \Pr\left[Y = s'\right]\right| \\ &= \frac{1}{2} \cdot \sum_{s' \in [0, 2^{\lambda-1})} \left|\Pr\left[X = s'\right] - \Pr\left[Y = s'\right]\right| \\ &= \frac{1}{2} \cdot \left((2^{\lambda-1} - M) \cdot \left(\frac{1}{2^{\lambda-1} - M} - \frac{1}{2^{\lambda-1}}\right) + M \cdot \frac{1}{2^{\lambda-1}}\right) \\ &= \frac{M}{2^{\lambda-1}} \leq 2^{-\rho}. \end{aligned}$$

  Since $\mathsf{Convert}_{\mathbb{G}}$ is a deterministic function using modulo operation and a bijective mapping, we have $\mathsf{SD}(\mathsf{Convert}_{\mathbb{G}}(X), \mathsf{Convert}_{\mathbb{G}}(Y)) \leq \mathsf{SD}(X,Y) \leq 2^{-\rho}$. Observe that $r := \mathsf{Convert}_{\mathbb{G}}(Y)$ is equivalent to the uniform sampling $r \leftarrow \mathbb{G}$. We can see that the distributions of r in the two worlds have statistical distance at most $2^{-\rho}$, which is the upper bound of the advantage of any *computationally unbounded* distinguisher $\mathcal{D}$.

[4] A special case is that $|\mathbb{G}|$ is a power of two such that $|\mathbb{G}|$ divides 2^ℓ.

- **Case (ii):** $M/2^{\lambda-1} > 2^{-\rho}$. In this case, $\mathsf{Convert}_{\mathbb{G}}$ will invoke the PRG G_{ext}. We assume that, for the sake of contradiction, $\mathsf{Convert}_{\mathbb{G}}$ is not (t', ϵ')-pseudorandom for some distinguisher $\mathcal{D}$. Let $\omega : [0, |\mathbb{G}|) \to \mathbb{G}$ be the bijective mapping. We can construct a PRG adversary $\mathcal{A}$ that, given a challenge $c \in \{0,1\}^{\ell_0}$ from the PRG challenger, interprets $c \in [0, 2^{\ell_0})$, sends $r := \omega(c \bmod |\mathbb{G}|)$ to $\mathcal{D}$, and outputs whatever $\mathcal{D}$ outputs. The runtime of $\mathcal{A}$ is at most $t' + \mathsf{poly}(\lambda) \le t$ for some fixed polynomial $\mathsf{poly}(\cdot)$.

  Similar to the analysis in **Case (i)**, the following computational indistinguishability is bounded by the statistical distance of c:

$$\begin{aligned}
&\Big|\Pr\Big[r \leftarrow \mathbb{G} : \mathcal{D}(1^\lambda, r) = 1\Big] \\
&\qquad - \Pr\Big[c \leftarrow [0, 2^{\ell_0}), r := \omega(c \bmod |\mathbb{G}|) : \mathcal{D}(1^\lambda, r) = 1\Big]\Big| \\
= &\Big|\Pr\Big[c \leftarrow [0, 2^{\ell_0} - 2^{\ell_0} \bmod |\mathbb{G}|), r := \omega(c \bmod |\mathbb{G}|) : \mathcal{D}(1^\lambda, r) = 1\Big] \\
&\qquad - \Pr\Big[c \leftarrow [0, 2^{\ell_0}), r := \omega(c \bmod |\mathbb{G}|) : \mathcal{D}(1^\lambda, r) = 1\Big]\Big| \\
\le &\frac{2^{\ell_0} \bmod |\mathbb{G}|}{2^{\ell_0}} < \frac{|\mathbb{G}|}{2^{\ell_0}} \le 2^{-\rho}.
\end{aligned} \tag{30}$$

  Due to the construction of $\mathcal{A}$, we have that

$$\begin{aligned}
&\Pr\Big[c \leftarrow \{0,1\}^{\ell_0} : \mathcal{A}(1^\lambda, c) = 1\Big] \\
&\qquad = \Pr\Big[c \leftarrow [0, 2^{\ell_0}), r := \omega(c \bmod |\mathbb{G}|) : \mathcal{D}(1^\lambda, r) = 1\Big], \\
&\Pr\Big[s \leftarrow \{0,1\}^{\lambda-1}, c := G_{\mathsf{ext}}(s) : \mathcal{A}(1^\lambda, c) = 1\Big] \\
&\qquad = \Pr\Big[s \leftarrow \{0,1\}^{\lambda-1}, r := \mathsf{Convert}_{\mathbb{G}}(s) : \mathcal{D}(1^\lambda, r) = 1\Big].
\end{aligned} \tag{31}$$

  Using the contradiction assumption and (30), (31), we can see

$$\begin{aligned}
&\Big|\Pr\Big[s \leftarrow \{0,1\}^{\lambda-1}, c := G_{\mathsf{ext}}(s) : \mathcal{A}(1^\lambda, c) = 1\Big] \\
&\qquad - \Pr\Big[c \leftarrow \{0,1\}^{\ell_0} : \mathcal{A}(1^\lambda, c) = 1\Big]\Big| > \epsilon' - 2^\rho = \epsilon,
\end{aligned}$$

  contradicting the (t, ϵ)-pseudorandomness of G_{ext}.

This theorem follows from the above two cases. □

F.2 Coin-tossing

In Figure 18, we present the functionality $\mathcal{F}_{\mathsf{Rand}}$ for random coin-tossing, where two parties can obtain a uniformly sampled string $R \in \{0,1\}^\lambda$.

This functionality can be implemented in the $\mathcal{F}_{\mathsf{COT}}$-hybrid model. The high-level idea is that a precomputed COT tuple $\mathsf{M}_{1-b}[r_{1-b}] = \mathsf{K}_b[r_{1-b}] \oplus r_{1-b} \cdot \Delta_b$, where P_b has $(\Delta_b, \mathsf{K}_b[r_{1-b}]) \in \{0,1\}^\lambda \times \{0,1\}^\lambda$ and P_{1-b} has $(r_{1-b}, \mathsf{M}_{1-b}[r_{1-b}]) \in \{0,1\} \times \{0,1\}^\lambda$, is an *information-theoretic message authentication code* [NNOB12, DPSZ12]. From this perspective, r_{1-b} is a uniform bit authenticated under the global key Δ_b and the one-time key $\mathsf{K}_b[r_{1-b}]$, and the resulting MAC tag is $\mathsf{M}_{1-b}[r_{1-b}]$. This authenticated bit r_{1-b} (only known to P_{1-b}) can be "opened" (like a commitment) by P_{1-b} sending $(r_{1-b}, \mathsf{M}_{1-b}[r_{1-b}])$ to P_b, who checks $\mathsf{M}_{1-b}[r_{1-b}] = \mathsf{K}_b[r_{1-b}] \oplus r_{1-b} \cdot \Delta_b$. By each party P_b

Functionality $\mathcal{F}_{\mathsf{Rand}}$

Sample: Upon receiving (sample, ℓ) from P_0 and P_1, sample $R \leftarrow \{0,1\}^\ell$ and send R to P_0 and P_1.

Figure 18: Functionality for coin-tossing.

Functionality $\mathcal{F}_{\mathsf{OLE}}$

Parameters: Ring $\mathcal{R}$.

Extend: This functionality allows polynomially many (extend) commands. Upon receiving (extend, N) from P_0 and P_1:

1. If both parties are honest, sample $(\mathbf{x}_0, \mathbf{z}_0), (\mathbf{x}_1, \mathbf{z}_1) \leftarrow \mathcal{R}^N \times \mathcal{R}^N$ such that $\mathbf{z}_0 + \mathbf{z}_1 = \mathbf{x}_0 \odot \mathbf{x}_1$; otherwise (i.e., P_b is corrupted), receive $(\mathbf{x}_b, \mathbf{z}_b) \in \mathcal{R}^N \times \mathcal{R}^N$ from the adversary, sample $\mathbf{x}_{1-b} \leftarrow \mathcal{R}^N$, and recompute $\mathbf{z}_{1-b} := \mathbf{x}_b \odot \mathbf{x}_{1-b} - \mathbf{z}_b \in \mathcal{R}^N$.
2. Send $(\mathbf{x}_0, \mathbf{z}_0)$ to P_0 and $(\mathbf{x}_1, \mathbf{z}_1)$ to P_1.

Figure 19: Functionality for OLE.

opening a COT tuple for its r_b, the two parties can flip a uniform coin $r := r_0 \oplus r_1$. To produce a random λ-bit string R, λ precomputed COT tuple are required for each party.

The above opening can be batched using the technique [DNNR17] in the RPM. In this batched opening where the two parties agree on a compression function f, each P_b computes $C_b := f(\mathsf{M}_b[r_{b,1}], \ldots, \mathsf{M}_b[r_{b,\lambda}])$ and sends $((r_{b,1}, \ldots, r_{b,\lambda}), C_b)$ to P_{1-b}, who uses these values to check

$$C_b = f(\mathsf{K}_{1-b}[r_{b,1}] \oplus r_{b,1} \cdot \Delta_{1-b}, \ldots, \mathsf{K}_{1-b}[r_{b,\lambda}] \oplus r_{b,\lambda} \cdot \Delta_{1-b}).$$

The compression function f can be implemented in the way that $f(x_1, \ldots, x_n) := \oplus_{i \in [1,n]} \mathsf{H}(x_i)$ for some correlation-robust hash function $\mathsf{H} : \{0,1\}^\lambda \to \{0,1\}^\lambda$, which can be instantiated in the RPM [GKWY20]. The resulting coin-tossing protocol is one-round in the $\mathcal{F}_{\mathsf{COT}}$-hybrid model, and each party sends 2λ bits.

F.3 Oblivious Linear Evaluation

In Figure 19, we present the functionality $\mathcal{F}_{\mathsf{OLE}}$ for oblivious linear evaluation (OLE), where two parties obtain their additive shares of the component-wise multiplication result $\mathbf{x}_0 \odot \mathbf{x}_1 \in \mathcal{R}^N$. $\mathcal{F}_{\mathsf{OLE}}$ can be efficiently realized from, e.g., linearly homomorphic encryption [DPSZ12, KPR18], oblivious transfer [KOS16, GNN17], or DPF [BCG+20], under their respective assumptions.