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†

October 20, 2022

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 [GGM86] 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 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,
We propose a suite of half-trees as tailored alternatives for many GGM-tree-based protocols, resulting in halved computation/communication/round complexity (Table 1). 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 maintain 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 structure 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 trees [GKWY20], this structure 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× using a field-based random permutation. However, it only halves the communication when the subfield size is significantly

---

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Improvements</th>
<th>Computation</th>
<th>Communication</th>
<th># Rounds</th>
</tr>
</thead>
<tbody>
<tr>
<td>COT (§ 4.1.1)</td>
<td>2×</td>
<td>2×</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>sVOLE (§ 4.1.2)</td>
<td>2×</td>
<td>1 ~ 2×</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>sVOLE (§ 4.2)</td>
<td>1.33×</td>
<td>2×</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>DPF (§ 5.2)</td>
<td>1.33×</td>
<td>3×</td>
<td>2×</td>
<td></td>
</tr>
<tr>
<td>DCF (§ 5.3)</td>
<td>1.6×</td>
<td>2 ~ 3×</td>
<td>2×</td>
<td></td>
</tr>
</tbody>
</table>

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 |F| and |K|. In DCF protocol, communication varies as per the range size |R| of comparison functions. “–” indicates no improvement.

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.
Security relies on the conjecture that the adversary cannot evaluate the punctured result in their RPM-based UPF, where the GGM-style tree expansion uses $G(x) := H_0(x) \| H_1(x)$ for $H_0(x) := H(x) \oplus x$ and $H_1(x) := H(x) + x \mod 2^\lambda$.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[BCG+22]</td>
<td>ROM sVOLE</td>
<td>$m$ RO calls</td>
<td>$2t(\log \frac{m}{T} - 1)\lambda + 3t \log</td>
</tr>
<tr>
<td>Ad-hoc¹</td>
<td>sVOLE</td>
<td>$m$ RP calls + $0.5m$ RO calls</td>
<td>$t \log</td>
</tr>
<tr>
<td>This work</td>
<td>RPM</td>
<td>COT</td>
<td>$t(\log \frac{m}{T} - 1)\lambda + \lambda$</td>
</tr>
<tr>
<td>RPM</td>
<td>sVOLE</td>
<td>$m$ RP calls</td>
<td>$t(\log \frac{m}{T} - 1) \log</td>
</tr>
<tr>
<td>RPM</td>
<td>sVOLE</td>
<td>$1.5m$ RP calls</td>
<td>$t(\log \frac{m}{T} - 2)\lambda + 3t \log</td>
</tr>
<tr>
<td>RPM</td>
<td>sVOLE</td>
<td>$1.5m$ RP calls</td>
<td>$t \log</td>
</tr>
</tbody>
</table>

¹ Security relies on the conjecture that the adversary cannot evaluate the punctured result in their RPM-based UPF, where the GGM-style tree expansion uses $G(x) := H_0(x) \| H_1(x)$ for $H_0(x) := H(x) \oplus x$ and $H_1(x) := H(x) + x \mod 2^\lambda$.

Table 2: Comparison with the concurrent work. “RO/ROM” (resp., “RP/RPM”) is short for random oracle (resp., permutation) and the model. $m$ denotes the length of sVOLE correlations. Computation is measured by the amount of symmetric-key operations. In practice, there is also some LPN-related computation cost. Assume weight-$t$ regular LPN noises in sVOLE extension with field $\mathbb{F}$ and extension field $\mathbb{K}$.

Smaller than the field size. Then, we modify our cGGM tree to obtain a pseudorandom correlated GGM (pcGGM) tree, which is similar to a cGGM tree but has pseudorandom leaves. pcGGM tree reduces the communication of the prior works by $2\times$ and achieves a $1.33\times$ saving in terms of computation as well.

**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 correlations, the sVOLE extension protocols based on their proposal either needs 1.5$m$ 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, BCGI18]. 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

We use \( \lambda \) for the computational security parameter. We write \( \text{negl}(\cdot) \) to denote an unspecified negligible function and log to denote the logarithm in base 2. We write \( n = n(\lambda) \) to indicate that \( n \in \mathbb{N} \) is polynomial in \( \lambda \). We use \( x \leftarrow S \) to denote sampling \( x \) uniformly at random from a finite set \( S \). We define \( [a,b) = \{a, \ldots, b-1\} \) and \( [a,b] = \{a, \ldots, b\} \). For some \( i \geq 0 \), we write \( a[i] \) to denote the \( i \)-th entry of a vector \( a \). We use \( \text{Unit}_G(n, \alpha, \beta) \) for a vector in \( \mathbb{G}^n \) such that its \( \alpha \)-th entry is \( \beta \) and other entries are 0. For some bit-string \( x \in \{0,1\}^n \), we use \( \text{lsb}(x) \) for its least significant bit (LSB), \( \text{bb}(x) \) for its highest \( n-1 \) bits, and \( x_i \) for the \( i \)-th bit of \( x \) with the highest bit \( x_1 \). We will interchangeably use \( \mathbb{F}_{2^\lambda} \), \( \mathbb{F}_2 \), and \( \{0,1\}^\lambda \), where \( \oplus \) denotes mod-2 addition. We use \( \mathbb{F} \) for a field and \( \mathbb{K} = \mathbb{F}[X]/f(X) \) for its extension field under some irreducible polynomial \( f(X) \in \mathbb{F}[X] \). For some secret \( x \) and party \( b \in \{0,1\} \), we use \( \langle x \rangle_b \) for its bitwise mod-2 share of the bit-string of \( x \), and \( \langle x \rangle_b^A \) for its additive share on some finite group. We use \( || \) for bit-string concatenation and \( \circ \) for function composition.

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 \( \Pi \) 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 \( \mathcal{Z} \) in the real-world execution where the parties interact with \( \mathcal{A} \) and execute \( \Pi \) is computationally indistinguishable from the output distribution of \( \mathcal{Z} \) in the ideal-world execution where the parties interact with \( \mathcal{S} \) and \( \mathcal{F} \). Without loss of generality, we assume a dummy adversary \( \mathcal{A} \) that interacts freely with the environment \( \mathcal{Z} \) and sends its view to \( \mathcal{Z} \).

Our protocols use the functionality \( \mathcal{F}_{\text{sVOLE}} \) (Figure 1) of subfield vector oblivious linear evaluation. This functionality is a semi-honest version of that in [WYKW21]. If \( \mathcal{K} = \mathbb{F}_{2^\lambda} \) and \( \mathbb{F} = \mathbb{F}_2 \), \( \mathcal{F}_{\text{sVOLE}} \) degenerates to the COT functionality in [YWL+20], and we write it as \( \mathcal{F}_{\text{COT}} \). If \( \mathcal{K} = \mathbb{F} \), \( \mathcal{F}_{\text{sVOLE}} \) becomes the VOLE functionality in [BCGI18, SGRR19, RS21]. We omit the session identifiers in the functionalities for simplicity.

2.3 Circular Correlation Robustness

Circular correlation robustness (CCR) [CKKZ12, GKY20] is the security notion first introduced for the circuit garbling with Free-XOR optimization [KS08], where there exists a global key \( \Delta \).
Functionality \( F_{\text{VOLE}} \)

**Parameters:** Field \( \mathbb{F} \) and its extension field \( K \).

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

**Extend:** Upon receiving \( \text{extend}, m \) from \( P_0 \) and \( P_1 \):

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

Figure 1: Functionality for subfield VOLE.

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. In this construction, it takes one block-cipher call to invoke a CCR function.

**Definition 1** (Circular Correlation Robustness, [GKWY20]). Let \( H : \{0,1\}^\lambda \rightarrow \{0,1\}^\lambda \), let \( \chi \) be a distribution on \( \{0,1\}^\lambda \), and let \( O_{H,\Delta}^\chi(x,b) := H(x \oplus \Delta) \oplus b \cdot \Delta \) be an oracle for \( x, \Delta \in \{0,1\}^\lambda \) and \( b \in \{0,1\} \). \( H \) is \((t,q,\rho,\epsilon)\)-CCR if, for any distinguisher \( \mathcal{D} \) running in time at most \( t \) and making at most \( q \) queries to \( O_{H,\Delta}^{\chi(\cdot,\cdot)} \), and any \( \chi \) with min-entropy at least \( \rho \), it holds that

\[
\left| \Pr_{\Delta \leftarrow \chi} \left[ \mathcal{D}_{O_{H,\Delta}^{\chi(\cdot,\cdot)}}^\chi(1^\lambda) = 1 \right] - \Pr_{f \leftarrow F_{\lambda+1,\lambda}} \left[ \mathcal{D}_{O_{H,\cdot}^{\chi(\cdot,\cdot)}}^f(1^\lambda) = 1 \right] \right| \leq \epsilon,
\]

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

In this work, we focus on the CCR distinguisher \( \mathcal{D} \) that can only make a set of restricted queries. Such queries are similar to those made by the adversary in the Half-Gate optimization [ZRE15] for circuit garbling. We postpone 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 \( F_{X,G} \) of functions with domain \( X \) and range \( G \), where \( G \) is an Abelian group, a two-party FSS scheme with key space \( K_0 \times K_1 \) has the following syntax:

- \((k_0, k_1) \leftarrow \text{Gen}(1^\lambda, \hat{f})\). On input \( 1^\lambda \) and the description \( \hat{f} \in \{0,1\}^* \) of a function \( f \in F_{X,G} \), output a key pair \((k_0, k_1) \in K_0 \times K_1 \).
- \(f_b(x) \leftarrow \text{Eval}(b, k_b, x)\). On input the party identifier \( b \in \{0,1\} \), the party’s key \( k_b \in K_b \), and a point \( x \in X \), output the share \( f_b(x) \in G \).

A two-party FSS scheme \((\text{Gen}, \text{Eval})\) is secure for the function family \( F_{X,G} \) with leakage \( \text{Leak} : \{0,1\}^* \rightarrow \{0,1\}^* \) if the following properties hold.
• **Correctness.** For any function $f \in \mathcal{F}_{\mathcal{X},G}$ with description $\hat{f}$, and any $x \in \mathcal{X}$,

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

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

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

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

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

**Distributed Comparison Functions (DCF s).** A two-party distributed comparison function with domain $\mathcal{X}$ and range $\mathcal{G}$ is a two-party FSS scheme $(\text{DCF.Gen}, \text{DCF.Eval})$ for the function family $\mathcal{F}_{\mathcal{X},G} = \{ f_{\alpha,\beta} \}_{\alpha,\beta \in \mathcal{G}}$ 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 built from its “single-point” version by turning to the LPN assumption, we focus on single-point COT/sVOLE, where the vector $u$ in a COT/sVOLE correlation $w = v + u \cdot \Delta$ has a Hamming weight of 1.

**Correlated OT from correlated GGM.** The state-of-the-art single-point COT uses a GGM tree as a PPRF, and we recall this protocol from the perspective of GGM tree. In this protocol, the sender holds an $n$-level GGM tree, whose $2^n$ leaves in $\mathbb{F}_2^\lambda$ forms a vector $v \in \mathbb{F}_2^{2^n}$. The receiver with a punctured point $\alpha = \alpha_1 \ldots \alpha_n \in \{0,1\}^n$ uses, for each $i \in [1,n]$, a standard OT to select the XOR of all $\alpha_i$-side (i.e., left or right) nodes on the $i$-th level. Using these $n$ XOR values, the receiver can recover the $n$ off-path nodes just leaving the path labeled by $\alpha$ (say, $\alpha$-path) in the GGM tree and use these $n$ nodes to recover all leaves except the $\alpha$-th one, corresponding to a vector $w \in \mathbb{F}_2^{2^n}$ with the punctured entry $w[\alpha]$. The sender samples a global $\Delta \leftarrow \mathbb{F}_2^\lambda$, defines its output as $(\Delta, v)$, and sends $\psi := \Delta \oplus (\oplus_{j \in [0,2^n]} v[j]) \in \mathbb{F}_2^\lambda$ to the receiver. The receiver patches $w[\alpha] := \psi \oplus (\oplus_{j \neq \alpha} w[j])$ and defines its output as $(u, w)$ for $u = \text{Unit}_{\mathbb{F}_2^n}(2^n, \alpha, 1)$. Here, the computation is dominated by the full GGM-tree expansion while the communication comes from $n$ parallel standard OTs.

We show that by enforcing additional correlation on a GGM tree, thus correlated GGM tree ($c\text{GGM}$), we can improve both computation and communication. An $n$-level $c\text{GGM}$ tree is a full binary tree with the invariant that the sum of all same-level nodes equals to a global offset $\Delta$. We use a hash function $H$ and the generalized Davies-Meyer construction to keep this invariant: for each parent node $x$, we set its left child as $H(x)$ and its right child as $x - H(x)$. A $c\text{GGM}$ tree is defined by its two first-level nodes $(k, \Delta - k)$, and its root can be discarded.
In our cGGM-tree single-point COT, an important observation is that, if a $\Delta \in \mathbb{F}_{2\lambda}$ meanwhile serves as the cGGM-tree offset and the global key $\Delta$ in precomputed COT correlations, the single-point COT protocol does not need standard OT anymore. Assume that there is a precomputed COT correlation where the sender has $(\Delta, K[r_i]) \in \mathbb{F}_{2\lambda} \times \mathbb{F}_{2\lambda}$ and the receiver has $(r_i, M[r_i]) \in \mathbb{F}_2 \times \mathbb{F}_{2\lambda}$ for each $i \in [1, n]$. In order to learn the XOR $K[r_i] \in \mathbb{F}_{2\lambda}$ of all $\alpha$-side nodes on the $i$-th level, the receiver sends $\pi_i \oplus r_i$ to the sender, receives back $c_i := K_0^i \oplus K[r_i] \oplus (\pi_i \oplus r_i) \cdot \Delta$, and computes

$$c_i \oplus M[r_i] = K_0^i \oplus K[r_i] \oplus (\pi_i \oplus r_i) \cdot \Delta \oplus M[r_i] = K_0^i \oplus \pi_i \cdot \Delta = K_1^i,$$  

where the last equality uses $K_0^i \oplus K_1^i = \Delta$ for the two XORs $K_0^i, K_1^i \in \mathbb{F}_{2\lambda}$ in a cGGM tree with global offset $\Delta$. For each level, only one message is sent from the sender to the receiver, in contrast to two messages in a standard OT. Directly basing the protocol on COT halves the communication since a standard OT also uses a COT correlation as per the standard technique [IKNP03, Bea95]. When the point $\alpha$ is random, the message from the receiver can be avoided as well.

As for the single-point COT output, the sender (resp., the receiver) can use all $2^n$ leaves (resp., all $2^n$ ones except the $\alpha$-th leaf) in the cGGM tree to form its output vector $v \in \mathbb{F}_{2n}$ (resp., $w \in \mathbb{F}_{2\lambda}^n$ except $w[\alpha]$). Then, the receiver locally patches $w[\alpha] := \oplus_{j \neq \alpha} w[j]$ and still defines $u = \text{Unit}_{\mathbb{F}_2}(2^n, \alpha, 1)$. The single-point COT correlation $w = v \oplus u \cdot \Delta$ holds due to the cGGM invariant.

Note that the full cGGM-tree expansion requires $2^n$ H calls rather than $2^n$ length-doubling PRG calls in a GGM tree, where the latter can be implemented with $2 \times 2^n$ RP calls as suggested by [GKWY20]. In contrast, our cGGM-tree single-point COT is secure if $H$ is implemented from a random permutation and a linear orthomorphism\(^1\). This means that the full cGGM-tree expansion roughly costs $2^n$ RP calls, only a half of that in the prior work.

In our protocol, the security against the corrupted receiver turns to the programmability of the random permutation. Note that the environment $Z$ can see the global key $\Delta$ from the sender’s output and use it to distinguish two executions. In the real execution, the $n$ XOR values given to the receiver are masked by $n$ off-path nodes $\{s^i_\alpha\}_{i \in [1, n]}$, which depend on the $\Delta$ identical in precomputed COT correlations and single-point COT outputs, according to:

$$\forall j \in [2, n] : s^\Delta_\pi_j = H \left( \Delta \oplus \bigoplus_{i=1}^{j-1} s^\Delta_\pi_i \right) \oplus \pi_j \cdot \left( \Delta \oplus \bigoplus_{i=1}^{j-1} s^\Delta_\pi_i \right).$$  

However, these $n$ off-path nodes are uniform in the simulation so that (2) does not holds with overwhelming probability. $Z$ can easily detect this inconsistency by querying the random permutation and its inverse with the $\Delta$-related transcripts. To maintain the consistency in the simulation, the simulator can extract every possible $\Delta$ from $Z$’s queries to the random permutation and its inverse, and ask the single-point COT functionality whether the ideal execution uses this $\Delta$ or not. If so, the simulator uses the $\Delta$ to program the random permutation and its inverse so that they are consistent with the $n$ simulated off-path nodes. Similar proof technique in the ROM have been used in TinyOT [NNOB12, HSS17].

**sVOLE from pseudorandom correlated GGM.** The application of GGM-tree PPRF to single-point COT can be generalized to obtain the blueprint of single-point sVOLE [BCG+19a, WYKW21] for field $\mathbb{F}$ and its extension field $\mathbb{K}$, with the following modifications: (i) each leaf in $\mathbb{F}_{2\lambda}$ is turned into an entry of $v$ or $w$ in $\mathbb{K}$ by additionally applying a function $\text{Convert}_\mathbb{K} : \mathbb{F}_{2\lambda} \rightarrow \mathbb{K}$, which maps random strings to pseudorandom elements in $\mathbb{K}$, (ii) the last-level standard OT is instead

\(^1\)A mapping $\sigma : G \rightarrow G$ for an additive Abelian group $G$ is a linear orthomorphism if (i) $\sigma$ is a permutation, (ii) $\sigma'(x) := \sigma(x) - x$ is also a permutation, and (iii) $\sigma(x + y) = \sigma(x) + \sigma(y)$ for any $x, y \in G$. Compared with a random permutation, the computational cost of a linear orthomorphism is relatively cheap; see [GKWY20].
specialized for two elements in \( K \), (iii) a precomputed sVOLE correlation \( M[\beta] = K[\beta] + \beta \cdot \Gamma \) is required, where the sender has \( (\Gamma, K[\beta]) \in K \times K \) and the receiver has \( (\beta, M[\beta]) \in F^* \times K \), and (iv) the sender instead defines its output as \( (\Gamma, v) \) and sends \( \psi := \sum_{j \in [0, 2^n]} v[j] - K[\beta] \), and the receiver defines \( u := \text{Unif}_F(2^n, \alpha, \beta) \) and patches \( w[\alpha] := \psi + M[\beta] - \sum_{j \neq \alpha} w[j] \). This protocol costs \( 2 \times 2^n \) RP calls per party, and its communication bottleneck is the \( n \) parallel standard OTs.

Although our cGGM tree does not yield a PPRF due to its leveled correlation, we show that this tree can be modified to obtain a leaf-pseudorandom tree, thus pseudorandom cGGM (pcGGM) tree, which essentially yields a PPRF and can be used in the single-point sVOLE blueprint. An \( n \)-level pcGGM tree is identical to an \( n \)-level cGGM tree expanded from two first-level nodes in \( F_{2^\lambda} \) and a hash function \( H' : F_{2^\lambda} \rightarrow F_{2^\lambda} \), except that it defines the leaves as follows: for each node \( x \in F_{2^\lambda} \) on the \( (n - 1) \)-th level, it defines the left child as \( \text{Convert}_K(H'(x)) \) and the right child as \( \text{Convert}_K(H'(x \oplus 1)) \). This definition aims to break the cGGM-tree correlation in the last level to obtain pseudorandom leaves in \( K \).

To use pcGGM tree in the single-point sVOLE blueprint, we require that the \( n \) off-path nodes for the \( \alpha \)-path and the leaf at the punctured point \( \alpha \) are pseudorandom. This pseudorandomness relies on the observation that these nodes are formed from the responses to restricted CCR queries (see Appendix A for formal treatment). In other words, there exists a global \( \Delta \in F_{2^\lambda} \) that offsets two first-level pcGGM-tree nodes, defines the considered nodes, and has sufficient entropy from the view of \( Z \) (note that such a \( \Delta \) does not exist in our cGGM tree, where \( \Delta \) is eventually included in the sender’s output and given to \( Z \)). More specifically, \( \Delta \) defines the first \( n - 1 \) off-path nodes \( \{s_i\}_{i \in [1, n-1]} \) as per the CCR queries in (2) for \( H := H' \). Meanwhile, the last off-path node \( s_\mathbf{o}_n \) and the punctured leaf \( s_\mathbf{a}_n \) are of the form

\[
\forall b \in \{0, 1\} : s_b^n = \text{Convert}_K \left( H' \left( \Delta \oplus \bigoplus_{i \in [1, n-1]} s_i^n \oplus b \right) \right),
\]

where \( \text{Convert}_K \) takes CCR responses as input. To ensure the pseudorandomness of such CCR responses, we can construct \( H' \) from a CCR hash function, which can be instantiated in the RPM [GKWy20] and takes one RP call per call.

Our pcGGM tree leads to a more efficient single-point sVOLE protocol. The efficiency gains come from the leveled correlation in the first \( n - 1 \) levels. As in our cGGM-tree single-point COT, the receiver in this single-point sVOLE can use precomputed COT correlations to compute the \( \pi_i \)-side XOR as per (1) for each of these levels. Thus, the sender sends only one message for each of the first \( n - 1 \) levels. For the last level, the two parties still use a standard OT as in the prior works due to the last-level pseudorandomness. In this way, the two parties can define their pcGGM-tree leaves by following the same blueprint of the prior works. As \( n \) grows, the communication saving of our protocol is almost \( 2 \times \) while it reduces the number of RP calls from \( 2 \times 2^n \) to \( 1.5 \times 2^n \).

**sVOLE from correlated GGM.** Using cGGM tree, we also propose another flavor of single-point sVOLE protocol for any field \( F \) and its exponentially large extension \( K \). The cGGM tree uses a hash function \( H : K \rightarrow K \), which is implemented with a random permutation and a linear orthomorphism over \( K \). The global offset in the cGGM tree is \( K[\beta] \in K \), coming from a precomputed sVOLE correlation \( M[\beta] = K[\beta] + \beta \cdot \Delta \) where the sender has \( (\Delta, K[\beta]) \in K \times K \) and the receiver has \( (\beta, M[\beta]) \in F^* \times K \). For the \( i \)-th level, the two parties also precomputes a special sVOLE correlation \( M[r_i] = K[r_i] + r_i \cdot K[\beta] \), where the sender has \( K[r_i] \in K \) and the receiver has \( (r_i, M[r_i]) \in F_2 \times K \).

Our cGGM-tree single-point sVOLE protocol works as follows. The sender produces an \( n \)-level cGGM tree with global offset \( K[\beta] \in K \) and forms its vector \( v \in K^{2^n} \) from the \( 2^n \) leaves. For the \( i \)-th level, the receiver consumes a special sVOLE correlation to compute the sum \( K_{\mathbf{r}_i}^i \in K \) of all \( \pi_i \)-side nodes. More specifically, after receiving \( c_i := K[r_i] + K_i^0 \in K \) from the sender, the receiver
defines $\pi_i = r_i$ and computes

$$K_{\pi_i}^i := (-1)^{r_i} \cdot (-M[r_i] + c_i) = (-1)^{r_i} \cdot (K_0^i - r_i \cdot K[\beta]) = (-1)^{r_i} \cdot (K_0^i - \pi_i \cdot K[\beta])$$

according to the cGGM-tree invariant. Recall that the sums $\{K_{\pi_i}^i\}_{i \in [1,n]}$ allows the receiver to recover, in a top-down manner, the $n$ off-nodes for the $\alpha$-path and all $2^n$ leaves except the $\alpha$-th one. The receiver defines its vector $w \in \mathbb{K}^{2n}$ by using the $\alpha$-exclusive $2^n - 1$ leaves and the patched punctured leaf $w[\alpha] := M[\beta] - \sum_{j \neq \alpha} w[j] = M[\beta] - (\sum_{j \neq \alpha} w[j] + v[\alpha]) + v[\alpha] = v[\alpha] + \beta \cdot \Delta$. If the sender defines its output as $\langle \Delta, v \rangle$ and the receiver defines its output as $\langle u, w \rangle$ for $u := \text{Unit}_F(2^n, \alpha, \beta)$, the two parties share a single-point sVOLE correlation.

Similar to our cGGM-tree single-point COT, this protocol also faces the issue that $Z$ can observe $\Delta \in \mathbb{K}$ from the sender’s output and compute the cGGM-tree offset $K[\beta] = M[\beta] - \beta \cdot \Delta$ to distinguish the two executions. However, to perform this attack, $Z$ must query the random permutation or its inverse with the $K[\beta]$-related transcripts. Thus, the simulator can extract every possible $K[\beta]$ from the random permutation and its inverse, and interact with the single-point sVOLE functionality to guess the associated $\Delta := \beta^{-1} \cdot (M[\beta] - K[\beta])$. Given the $\Delta$ (or rather, its associated $K[\beta]$) used in the ideal execution, the simulator programs the random permutation and its inverse to maintain the consistency for the $n$ simulated off-path nodes.

Compared to our pcGGM-tree single-point sVOLE, this cGGM-tree version only requires $2^n$ RP calls. However, to set up the $n$ special sVOLE correlations from random ones, the receiver should send additional $n \cdot \log |F|$ bits. When $|F| \ll |\mathbb{K}|$ so that this additional communication is far less than the $c_i$ messages sent by the sender, this protocol can also halve the communication in the prior works [BCG+19a, WYKW21] as our pcGGM-tree version.

### 3.2 DPF/DCF from Shared Pseudorandom Correlated GGM Trees

#### DPF and its protocol from shared pseudorandom correlated GGM. The prior DPF scheme [BGI16] for $(\alpha, \beta) \in \{0,1\}^n \times \mathbb{G}$ uses XOR secret sharing $\langle \cdot \rangle$ to share an $n$-level GGM-style tree between two parties. In a shared GGM tree, each node is a $\lambda$-bit string, where the high $\lambda - 1$ bits serve as the seed $s$ of a PRG $G : \{0,1\}^{\lambda - 1} \rightarrow \{0,1\}^{2\lambda}$ and the LSB is called control bit $t$. This tree uses $G$ for its expansion and maintains the invariant that a node is pseudorandom in $\{0,1\}^{\lambda - 1} \parallel 1$ if it is on the $\alpha$-path, or $0^\lambda$ otherwise. To share this tree, the two parties first share a random root in $\{0,1\}^{\lambda - 1} \parallel 1$, meeting the invariant initially.

Then, the two parties maintain the invariant by using the observation that, for a shared seed $\langle s \rangle$ split into $\langle s \rangle = s_0 \oplus s_1$, $G(s_0) \oplus G(s_1) = 0^2 \lambda$ if $\langle s \rangle = 0^\lambda$; $G(s_0) \oplus G(s_1)$ is pseudorandom if $\langle s \rangle$ is random. Thus, $G(\langle s \rangle) := G(s_0) \oplus G(s_1)$ is a shared zero-or-pseudorandom secret according to $\langle s \rangle$. For a shared node $\langle s^{i-1} \parallel t^{i-1} \rangle$ on the $(i - 1)$-th level, its $\sigma$-side ($\sigma \in \{0,1\}$) child is shared as:

$$\langle s^{i-1} \parallel t^{i-1} \rangle := G_\sigma(\langle s^{i-1} \rangle) \oplus \langle t^{i-1} \rangle \cdot (\text{HCW}_i \parallel \text{LCW}_i[\sigma]),$$

where $G_0(s) \parallel G_1(s) := G(s)$, and the $i$-th public correction word $\text{HCW}_i$ contains the $(\lambda - 1)$-bit $\text{HCW}_i$ and two $\text{LCW}_i[\sigma]$ bits. For induction, assume that the invariant holds for the $(i - 1)$-th level. $\text{CW}_i$ is determined by solving the linear equation (3) under the invariant constraint for each $\sigma \in \{0,1\}$ and the induction assumption for the shared node $\langle s^{i-1} \parallel t^{i-1} \rangle$ is on the $\alpha$-path, i.e.,

$$\text{HCW}_i := \text{hs}(\langle s^{i-1} \rangle), \quad \text{LCW}_i[\sigma] := \text{lsb}(G_\sigma(\langle s^{i-1} \rangle)) \oplus \alpha_i \oplus \overline{\sigma}.$$
computed from the shared leaf $\langle s^n \parallel t^n \rangle = (s_0^n \parallel t_0^n) \oplus (s_1^n \parallel t_1^n)$ on the $\alpha$-path.

In this construction, the DPF key of each party includes its share of the root and the $n + 1$ pseudorandom public correction words. The DPF protocol of [Ds17] can distribute such a DPF key pair to the two parties who already share the parameters ($\alpha$, $\beta$). This protocol uses the black-box PRG evaluation, where, as a corollary of the maintained invariant, the share of the $i$-th node on the $\alpha$-path can be computed from the share of the sum of all $2^i$ nodes on the $i$-th level. This technique results in about $2 \times 2^n$ RP calls per party to perform full tree expansion. Moreover, the correction words are computed from generic 2PC, which can be instantiated from GMW protocol [GMW87] with standard OTs.

We present a new DPF scheme constructed from a shared pseudorandom correlated GGM (spcGGM) tree, which facilitates more 2PC-friendly expansion in the associated DPF protocol. Our first observation is that we only require pseudorandom leaves to hide $\beta$ in $\text{CW}_{n+1}$, and the intermediate pseudorandom nodes in a GGM tree are not mandatory for DPF. Our spcGGM tree introduces a global offset $\Delta \in \{0, 1\}^\lambda$ such that $\text{lsb}(\Delta) = 1$ and uses a hash function $H' : \{0, 1\}^\lambda \rightarrow \{0, 1\}^\lambda$ to maintain a new invariant: the first $n - 1$ nodes on the $\alpha$-path are corrected to $\Delta$, rather than pseudorandom values as in the prior construction. We also introduce a simpler tree expansion than (3): for $i \in [1, n - 1]$,

$$\langle s^i \parallel t^i \rangle := H'((s^{i-1} \parallel t^{i-1})) \oplus \sigma \cdot (s^{i-1} \parallel t^{i-1}) \oplus (t^{i-1}) \cdot \text{CW}_i,$$

where the shared spcGGM-tree root is initially $\Delta$, and $H'((s \parallel t))$ is defined as $G((s))$ similarly. Solving this linear equation for the public correction word $\text{CW}_i$ under the invariant constraint and the on-path node $\langle s^{i-1} \parallel t^{i-1} \rangle$, we have

$$\text{CW}_i := H'((s^{i-1} \parallel t^{i-1})) \oplus \sigma \cdot \Delta.$$

To obtain a shared leaf $\langle s^n \parallel t^n \rangle$, our spcGGM tree also uses (3) but replaces the half PRG output $G_\sigma((s^{n-1}))$ by $H'((s^{n-1} \parallel t^{n-1}) \oplus \sigma)$. To ensure the pseudorandomness of the first $n - 1$ public correction words and the leaves, we require $H'$ to be CCR for a set of restricted queries (corresponding to the level-by-level spcGGM tree expansion) described in Appendix A.

In our DPF protocol where $\alpha$ and $\beta$ are shared, each party sets up an $\mathcal{F}_{\text{COT}}$ instance and exchanges the LSB of its global key with the other party to obtain a share of $\Delta$. This share not only serves as the share of the spcGGM-tree root but also works with the share of $\alpha$ and the $\mathcal{F}_{\text{COT}}$ outputs to share all $\sigma_i \cdot \Delta$ in one round (by sending a few difference bits). For $i \in [1, n - 1]$, the two parties can locally share $H'((s^{i-1} \parallel t^{i-1}))$ by using the black-box evaluation technique in [Ds17].

Given these shares and the shares of $\sigma_i \cdot \Delta$, the computation of the first $n - 1$ correction words only needs $n - 1$ rounds for revealing $\text{CW}_i$. In contrast, [Ds17] requires $2(n - 1)$ rounds since, for each level, one round is to compute the share of $\text{CW}_i$ from standard OTs and another round is to reveal it.

Our second observation is that, when $\mathcal{G}$ is not a group of bit-strings, the prior $\text{CW}_{n+1}$ in (4) also needs 2PC to share $t_0^n$, the LSB of the $\alpha$-th leaf share among the $2^n$ ones only known to one party. If this 2PC is parallelized with that of correction words, each party sends $4n$ additional bits, without increasing computation and round complexity. When $\mathcal{G}$ is a ring, this $\text{CW}_{n+1}$ can be computed from the shared $t_0^n$ and $\beta$ via secure multiplication on the ring in two rounds.

Instead, our spcGGM tree introduces a conceptually simpler $\text{CW}_{n+1}$:

$$\text{CW}_{n+1} := (t_0^n - t_1^n) \cdot (\text{Convert}_\mathcal{G}(s_0^n) - \text{Convert}_\mathcal{G}(s_1^n) + \beta),$$

which is equivalent to (4) since $t_0^n \oplus t_1^n = 1$ at the $\alpha$-th leaf of the spcGGM tree. However, the two multiplication operands can be locally shared on $\mathcal{R}$ via the black-box evaluation technique. As a
result, two rounds are required to compute and reveal the multiplication result, i.e., $\text{CW}_{n+1}$. The $4n$-bit communication and the computation per party for the shared $t_i^0$ can be avoided.

Putting all things together, our DPF protocol runs in about a half of the rounds required by the state-of-the-art protocol as $n$ grows. The communication saving is about $3 \times$ since the two parties only exchange one revealing message for each of the first $n - 1$ correction words (and a few bits to set up all $\pi_i \cdot \Delta$ shares), in contrast to the two OT messages plus one revealing message.

DCF and its protocol from shared pseudorandom correlated GGM. The prior DCF scheme [BCG+21] shares an extended version of the GGM tree in the DPF scheme [BGI16]. In the extended tree, each $\lambda$-bit non-root node is also associated with a value in $\mathbb{G}$. To obtain such values, the DCF uses a PRG $G' : \{0, 1\}^{\lambda-1} \rightarrow \{0, 1\}^{2(\lambda-1)}$ such that $G_0(s) \parallel G_0^Y(s) \parallel G_1(s) \parallel G_1^Y(s) := G'(s)$.

We interpret this DCF scheme as a non-black-box construction based on the DPF scheme for point function $f_{\alpha, \beta}(x)$. The two $\lambda$-bit $G_{\alpha}(\cdot)$ strings maintain such a DPF scheme according to the computation in (3). Meanwhile, the other two $(\lambda - 1)$-bit $G_{\alpha}^Y(\cdot)$ strings are used to keep a DCF invariant that, for any $x \in \{0, 1\}^n$, the sum $V(x)$ of the $n$ values along the $x$-path equals $\alpha_{k+1} \cdot \beta \in \mathbb{G}$, where $\alpha_{k+1}$ is the highest different bit between $x$ and $\alpha$, and $\alpha_{k+1} := \alpha_n$ if $x = \alpha$. To obtain $V(x)$, any value on the $i$-th level will be corrected by a public value correction word $\text{VCW}_i$, which is computed from the $(i - 1)$-th node $\langle s^{-1}_{i-1} \parallel t^{-1}_{i-1} \rangle$ shared as per the DPF invariant on the $\alpha$-path:

$$\text{VCW}_i := (-1)^{t_{i-1}} \cdot \left( \left(\text{Convert}_G(v_i^{i, \pi_1}) - \text{Convert}_G(v_{i-1}^{i, \pi_1}) \right) - V_{i-1} + (\alpha_i - \alpha_{i-1}) \cdot \beta \right),$$

where we define $v_b^{i, \sigma} := G_{\sigma}^G(s_b^{-1}), V_0 := 0$, and, for $i \in [2, n]$,

$$V_{i-1} := \sum_{b \in \{0, 1\}} (-1)^{1-b} \cdot \left( \text{Convert}_G(v_b^{i-1, \pi_{i-1}}) - \text{Convert}_G(v_b^{i-1, \pi_{i-1}}) \right).$$

Note that the comparison function $f_{\alpha, \beta}^<(x) = f_{\alpha, \beta}^<(x) + V(x)$. The DCF key of each party includes its associated DPF key and the $n$ value correction words.

This interpretation of the DCF scheme inspires us to extend our spcGGM tree to obtain a new DCF scheme and its distributed protocol. This extended spcGGM tree keeps the invariant and the efficiency features of the original DPF part. In addition, it introduces a simpler DCF invariant.

Our first observation is that, similar to $\text{CW}_{n+1}$, the $(-1)^{t_{i-1}}$ term in the prior $\text{VCW}_i$ construction can be replaced by a locally shared term $t_{i-1}^0 - t_{i-1}^1 \in \mathcal{R}$. As a result, the secure computation of $(-1)^{t_{i-1}}$ can be avoided in our DCF protocol.

More importantly, our second observation is that, to ensure pseudorandom $\text{VCW}_i$, it is sufficient to use only one random $v_b^i = v_b^{i,0} = v_b^{i,1}$ so that each $V_{i-1}$ is syntactically zero. In our extended spcGGM-tree expansion, we define this $v_b^i := H'((s_b^{-1}_{i-1} \parallel t_{i-1}) \oplus 2)$. The resulting $\text{VCW}_i$ not only allows us to save one $G_{\sigma}^Y(\cdot)$ call per non-leaf node in the tree expansion but also turn the Convert$_G(\cdot)$ difference in $\text{VCW}_i$, which non-linearly depends on $\pi_i$, into an $\pi_i$-independent one such that it can be locally shared using the black-box evaluation technique.

In our DCF protocol, the secure computation of $\text{VCW}_i$ only includes the sharing of all $\alpha_i \cdot \beta$ and the secure multiplication on $\mathcal{R}$, given locally shared $t_{i-1}^0 - t_{i-1}^1$ and Convert$_G(\cdot)$ differences. Since $\alpha$ and $\beta$ are already shared, the two parties can share each $\alpha_i \cdot \beta$ along with the spcGGM-tree correction word $\text{CW}_i$ by securely reusing the share (or rather, COT correlation) of $\alpha_i \cdot \Delta$ for the COT-based multiplication between a shared bit and a shared ring element. This multiplication somewhat generalizes the binary case in [ALSZ13, GKWy20] since we share $\alpha_i \cdot \beta = (\langle \alpha_i \rangle_0 \oplus \langle \alpha_i \rangle_1) \cdot \beta \in \mathcal{R}$ on the ring by using the well-known arithmetic XOR: $\langle \alpha_i \rangle_0 \oplus \langle \alpha_i \rangle_1 = \langle \alpha_i \rangle_0 + \langle \alpha_i \rangle_1 - 2 \cdot \langle \alpha_i \rangle_0 \cdot \langle \alpha_i \rangle_1$.

Given the two shared multiplication operands, $\text{VCW}_i$ can be securely computed along with the correction word $\text{CW}_{n+1}$ due to their similar construction.
**Functionality $F_{\mathrm{spsVOLE}}$**

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

**Initialize:** Upon receiving $(\text{init})$ 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 $\Delta$ and send it to $P_0$. Ignore all subsequent $(\text{init})$ commands.

**Extend:** Upon receiving $(\text{extend}, N)$ from $P_0$ and $P_1$:

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

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

Figure 2: Functionality for single-point subfield VOLE.

Overall, our DCF protocol has the same round complexity as our DPF one. Since the prior DCF protocol of [BCG+21] extends [Ds17] by computing $VCW_i$’s along with $CW_i$’s, its round complexity can also be halved by our protocol.

### 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) correlations output by the functionality $F_{\mathrm{spsVOLE}}$ (Figure 2) into an sVOLE correlation. We focus on constructing efficient spsVOLE protocols that UC-realize $F_{\mathrm{spsVOLE}}$. Note that the spsVOLE protocol dominates the computation and contributes essentially all communication in sVOLE extension.

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

Our spsVOLE protocols essentially work as the pseudorandom correlation generator (PCG) protocol [BCG+19b, BCG+19a, BCG+20] of spsVOLE correlations, although we do not explicitly divide the correlation generation into two PCG phases. In Appendix C.1, we show how to adapt one of our spsVOLE protocols for such two phases, in order to satisfy the “silent property” [BCG+19b, BCG+19a, BCG+20, CRR21] that a long correlation 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 full evaluation and its punctured version defined by our correlated GGM tree cGGM. Unlike the original GGM tree, our tree is expanded from 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 $H(x) \in \mathbb{K}$ while its right child is defined as $x - H(x) \in \mathbb{K}$. The following property is straightforward from an induction.
We prove Theorem 1 in Appendix B.1.

Parameter definitions:

\text{cGGM}.FullEval(\Delta, k): \text{ Given } (\Delta, k) \in \mathbb{K}^2, \begin{align*}
1: s^0_i &:= k \in \mathbb{K}, s^1_i := \Delta - k \in \mathbb{K}, \\
2: \text{ for } i \in [2, n], j \in [0, 2^{i-1}) &\text{ do} \\
3: s^{2j}_i &:= H(s^{2j-1}_i) \in \mathbb{K}, s^{2j+1}_i := s^{2j}_i - s^{2j}_i \in \mathbb{K}, \\
4: v &:= (s^0_n, \ldots, s^{2^n-1}_n) \in \mathbb{K}^2^n, \\
5: \text{ for } i \in [1, n] &\text{ do } K^0_i := \sum_{j \in [0, 2^{i-1})} s^{2j}_i \in \mathbb{K}, \\
6: \text{return } (v, \{K^i_j\}_{i \in [1, n]}) &\text{ as per } \text{cGGM}.FullEval.
\end{align*}

\text{cGGM}.PuncFullEval(\alpha, \{K^i_j\}_{i \in [1, n]}): \text{ Given } (\alpha, \{K^i_j\}_{i \in [1, n]}) \in \{0, 1\}^n \times \mathbb{K}^n, \begin{align*}
1: s^{\alpha_i}_1 &:= K^i_1 \in \mathbb{K}, \\
2: \text{ for } i \in [2, n] &\text{ do} \\
3: \text{ for } j \in [0, 2^{i-1}), j \neq \alpha_1 \ldots \alpha_{i-1} &\text{ do} \\
4: s^{2j}_i &:= H(s^{2j-1}_i) \in \mathbb{K}, s^{2j+1}_i := s^{2j}_i - s^{2j}_i \in \mathbb{K}, \\
5: s^{2\alpha_1 \ldots \alpha_{i-1} \cdot \pi_i}_i &:= \sum_{j \in [0, 2^{i-1}), j \neq \alpha_1 \ldots \alpha_{i-1}} s^{2j+1}_i \in \mathbb{K}, \\
6: s^\alpha_n &:= -\sum_{j \in [0, 2^n), j \neq \alpha} s^j_n \in \mathbb{K}, w := (s^0_n, \ldots, s^{2^n-1}_n) \in \mathbb{K}^2^n, \\
7: \text{return } w &\text{ as per } \text{cGGM}.PuncFullEval.
\end{align*}

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 \text{cGGM}.FullEval.

Corollary 1. For any \(\alpha \in [0, 2^n)\), any \((k, \Delta - k) \in \mathbb{K}^2\), and
\[
(v, \{K^i_j\}_{i \in [1, n]}) := \text{cGGM}.FullEval(\Delta, k), \quad w := \text{cGGM}.PuncFullEval(\alpha, \{K^i_j\}_{i \in [1, n]}),
\]
where \(K^i_j := \pi_i \cdot \Delta + (-1)^{\pi_i} \cdot K^0_j\) for \(i \in [1, n]\), we have \(w[\alpha] - v[\alpha] = -\Delta\).

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

4.1.1 Single-point COT

In Figure 4, we present our single-point COT protocol \(\Pi_{\text{spCOT}}\) that operates in the \(\mathcal{F}_{\text{cOT}}\)-hybrid model and uses the expansion of our correlated GGM tree in Figure 3. This protocol proceeds as described in Section 3.1 except the following highlighted difference.

The same \(\Delta\) in correlated GGM trees. Note that \(\mathcal{F}_{\text{spCOT}}\) generates single-point COT correlations with the same global key \(\Delta \in \mathbb{F}_{2^n}\). To realize \(\mathcal{F}_{\text{spCOT}}\), our protocol \(\Pi_{\text{spCOT}}\) uses the same \(\Delta\) across polynomially many correlated GGM trees, each of which requires a fresh uniform \(k \leftarrow \mathbb{F}_{2^n}\) in \text{cGGM}.FullEval(\(\Delta, k\)). A merit of using the same \(\Delta\) in several tree instances is that, when run in an iterative manner, \(\Pi_{\text{spCOT}}\) only invokes one \(\mathcal{F}_{\text{cOT}}\) instance, and the amortized cost per precomputed COT correlation can be sufficiently small.

Security. We prove Theorem 1 in Appendix B.1.
Protocol $\Pi_{\text{spCOT}}$

**Parameters:** Field $F_2$ and its extension field $F_{2^\lambda}$.

**Initialize:** This procedure is executed only once.
1. $P_0$ and $P_1$ send (init) to $F_{\text{COT}}$, which returns $\Delta \in F_{2^\lambda}$ to $P_0$. $P_0$ outputs $\Delta$.

**Extend:** $P_0$ and $P_1$ input $N = 2^n$ and use $cGGM$ (c.f. Figure 3) for $n$ and $F_{2^\lambda}$.
2. $P_0$ and $P_1$ send (extend, $n$) to $F_{\text{COT}}$, which returns $(K[r_1], \ldots, K[r_n]) \in F_{2^\lambda}^n$ to $P_0$ and $((r_1, \ldots, r_n), (M[r_1], \ldots, M[r_n])) \in F_{2^\lambda}^n \times F_{2^\lambda}^n$ to $P_1$ such that $M[r_i] = K[r_i] \oplus r_i \cdot \Delta$ for $i \in [1,n]$.
3. $P_0$ samples $c_1 \leftarrow F_{2^\lambda}$, and sets $k := K[r_1] \oplus c_1$, $(v, \{K_0^i\}_{i \in [1,n]}) := cGGM.\text{FullEval}(\Delta, k)$, and $c_i := K[r_i] \oplus K_0^i$ for $i \in [2,n]$. $P_0$ sends $(c_1, \ldots, c_n)$ to $P_1$.
4. $P_1$ sets $\alpha = \alpha_1 \ldots \alpha_n := \tau_1 \ldots \tau_n \in [0,N)$, $K_0^\alpha := M[r_1] \oplus c_1$ for $i \in [1,n]$, and $u := \text{Unit}_{\lambda}(N, \alpha, 1)$, $w := cGGM.\text{PuncFullEval}(\alpha, \{K_0^i\}_{i \in [1,n]})$.
5. $P_0$ outputs $v$ and $P_1$ outputs $(u, w)$.

Figure 4: $cGGM$-based single-point COT protocol in the $F_{\text{COT}}$-hybrid model.

**Theorem 1.** Given random permutation $\pi : F_{2^\lambda} \to F_{2^\lambda}$, linear orthomorphism $\sigma : F_{2^\lambda} \to F_{2^\lambda}$, and the correlated GGM tree (Figure 3) for field $F_{2^\lambda}$ and hash function $H(x) := \pi(\sigma(x)) \oplus \sigma(x)$, protocol $\Pi_{\text{spCOT}}$ (Figure 4) UC-realizes functionality $\mathcal{F}_{\text{spCOT}}$ (Figure 2) against any semi-honest adversary in the $F_{\text{COT}}$-hybrid model and the RPM.

**Communication optimization.** For $t$ parallel executions of $\Pi_{\text{spCOT}}$ (e.g., in COT extension), the random element $c_1$ in each execution can be compressed via a PRF $F : F_{2^\lambda} \times \{0,1\}^* \to F_{2^\lambda}$. Concretely, $P_0$ samples a PRF key $k_{\text{prf}} \leftarrow F_{2^\lambda}$ after receiving its COT outputs in all executions and sends this key to $P_1$. For each execution with identifier $id$, the two parties locally defines the element $c_1 := F(k_{\text{prf}}, id)$. This PRF key is only used in the current parallel executions. The security of this optimization follows from the PRF security and the fact that, in the $t$ parallel 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 $t$ parallel executions. $\Pi_{\text{spCOT}}$ requires $n$ COT correlations. $P_0$ sends $(n - 1) \cdot \lambda + \frac{1}{2}$ bits, and $P_1$ sends nothing. The computation per party comes from the tree expansion with $N$ RP calls.

In the $F_{\text{COT}}$-hybrid model, the prior single-point COT protocol [YWL+20] consumes $n$ COT correlations. 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 at least 50% of end-to-end computational improvement in COT.
4.1.2 Single-point sVOLE

In the protocol $\Pi_{\text{spsVOLE}-\text{cGGM}}$ presented in Figure 13 and summarized in Section 3.1, we show that the idea behind our single-point COT protocol $\Pi_{\text{spCOT}}$ can be somewhat generalized to single-point sVOLE. This generalization requires a random permutation and a linear orthomorphism over an exponentially large field $\mathbb{K}$.

**Security.** We prove Theorem 2 in Appendix B.2.

**Theorem 2.** Given random permutation $\pi : \mathbb{K} \rightarrow \mathbb{K}$, linear orthomorphism $\sigma : \mathbb{K} \rightarrow \mathbb{K}$, and the correlated GGM tree (Figure 3) for field $\mathbb{K}$ and hash function $H(x) := \pi(\sigma(x)) + \sigma(x)$, protocol $\Pi_{\text{spsVOLE}-\text{cGGM}}$ (Figure 13) UC-realizes functionality $\mathcal{F}_{\text{spsVOLE}}$ (Figure 2) against any semi-honest adversary in the $\mathcal{F}_{\text{sVOLE}}$-hybrid model and the RPM, for $|\mathbb{K}| \geq 2^\lambda$.

**Communication optimization.** We have the following two optimizations:

- The PRF-based optimization of our single-point COT protocol $\Pi_{\text{spCOT}}$ also applies to $t$ parallel executions of $\Pi_{\text{spsVOLE}-\text{cGGM}}$ (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 sVOLE correlation for $M[\beta] = \mathbb{K}[\beta] + \beta \cdot \Delta$ since $\beta \in \mathbb{F}$ 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 $t$ parallel executions. In the $\mathcal{F}_{\text{sVOLE}}$-hybrid model, $\Pi_{\text{spsVOLE}-\text{cGGM}}$ needs $n + 1$ sVOLE correlations. $P_0$ sends $(n - 1) \cdot \log |\mathbb{K}| + \frac{n}{2}$ 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 ($\mathcal{F}_{\text{COT}}, \mathcal{F}_{\text{sVOLE}}$)-hybrid model for comparison, the single-point sVOLE protocols [BCG+19a, WYKW21] consume $n$ COT correlations and one sVOLE correlation. These protocols use each COT correlation to emulate a string OT, where the sender $P_0$ sends $2\lambda$ 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 $\Pi_{\text{spsVOLE}-\text{cGGM}}$ is roughly identical to that of the prior protocols if $|\mathbb{F}| \approx |\mathbb{K}|$. However, for $|\mathbb{F}| \ll |\mathbb{K}|$, $\Pi_{\text{spsVOLE}-\text{cGGM}}$ still halves the communication of the prior protocols. The reduction in computation is 50%.

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

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 $\alpha$ is pseudorandom. This pseudorandom correlated GGM tree $\text{pcGGM}$ 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 $H_S$ depends on the key $S$, 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 3. In fact, this pcGGM yields PPRF, which is described in Appendix C.2.

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 resulting protocol is presented in Figure 6. In this protocol, the sender $P_0$ sends one message to the receiver $P_1$ for each of the first $n - 1$ levels, given a precomputed
The two random elements $\Pi$ and the pseudorandom correlated GGM tree (Figure 5) for field $K$. We prove Theorem 3 in Appendix B.3.

### Parameters:
- Tree depth $n \in \mathbb{N}$.
- Field $K$.
- Keyed hash function $H_S : \mathbb{F}_{2^\lambda} \rightarrow \mathbb{F}_{2^\lambda}$.
- Function $\text{Convert}_k : \mathbb{F}_{2^\lambda} \rightarrow K$ (c.f. Appendix F.1).

#### $\text{pcGGM.FullEval}(\Delta, k)$:
1. $s_1^0 := k \in \mathbb{F}_{2^\lambda}$, $s_1^1 := \Delta \oplus k \in \mathbb{F}_{2^\lambda}$.
2. for $i \in [2, n-1]$, $j \in [0, 2^{i-1})$ do
   3. $s_i^j := H_S(s_{i-1}^j) \in \mathbb{F}_{2^\lambda}$, $s_i^{j+1} := s_{i-1}^j \oplus s_i^j \in \mathbb{F}_{2^\lambda}$.
4. for $j \in [0, 2^{n-1})$, $\sigma \in \{0, 1\}$ do $s_n^{2j+\sigma} := \text{Convert}_k(H_S(s_n^{2j} \oplus k)) \in K$.
5. $v := (s_0^n, \ldots, s_n^{2n-1}) \in K^{2n}$.
6. for $i \in [1, n-1]$ do $K_i^0 := \oplus_{j \in [0, 2^{i-1})} s_i^j \in \mathbb{F}_{2^\lambda}$.
7. $(K_0^n, K_n^n) := (\sum_{j \in [0, 2^{n-1})} s_n^{2j}, \sum_{j \in [0, 2^{n-1})} s_n^{2j+1}) \in K^2$.
8. return $(v, \{K_i^0\}_{i \in [1, n-1]}, (K_0^n, K_n^n))$.

#### $\text{pcGGM.PuncFullEval}(\alpha, \{K_i^0\}_{i \in [1, n]}, \gamma)$:
1. $s_1^0 := K_1^0 \in \mathbb{F}_{2^\lambda}$.
2. for $i \in [2, n-1]$ do
   3. for $j \in [0, 2^{i-1})$, $\not= \alpha_1 \ldots \alpha_{i-1}$ do
      4. $s_i^{j+1} := \text{Convert}_k(H_S(s_i^{j} \oplus k)) \in K$.
   5. $s_i^{\alpha_1 \ldots \alpha_{i-1}} := K_i^0 + (\oplus_{j \in [0, 2^{i-1}) \not= \alpha_1 \ldots \alpha_{i-1}} s_i^{j+1}) \in \mathbb{F}_{2^\lambda}$.
6. for $j \in [0, 2^{n-1})$, $\not= \alpha_1 \ldots \alpha_{n-1}$, $\sigma \in \{0, 1\}$ do
   7. $s_n^{2j+\sigma} := \text{Convert}_k(H_S(s_n^{2j} \oplus k)) \in K$.
   8. $s_n^{\alpha_1 \ldots \alpha_{n-1}} := K_n^0 - \sum_{j \in [0, 2^{n-1}) \not= \alpha_1 \ldots \alpha_{n-1}} s_n^{2j+\alpha} \in K$.
   9. $s_n^\alpha := \gamma - \sum_{j \in [0, 2^n) \not= \alpha} s_n^j \in K$, $w := (s_0^n, \ldots, s_n^{2n-1}) \in K^{2n}$.
10. return $w$

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

COT correlation. For the last level, the two parties use a COT correlation and the standard technique [IKNP03, Bea95] to emulate the string OT as in the prior protocols. To amortize the cost per COT correlation, the pcGGM trees underlying multiple protocol executions also use the same offset $\Delta$ set by $F_{\text{COT}}$.

**Security.** We prove Theorem 3 in Appendix B.3.

**Theorem 3.** Given CCR function $H : \mathbb{F}_{2^\lambda} \rightarrow \mathbb{F}_{2^\lambda}$, function $\text{Convert}_k : \mathbb{F}_{2^\lambda} \rightarrow K$ (Appendix F.1), and the pseudorandom correlated GGM tree (Figure 5) for field $K$, keyed hash function $H_S(x) := H(S \oplus x)$ with key $S \leftarrow \mathbb{F}_{2^\lambda}$, and function $\text{Convert}_k$, protocol $\Pi_{\text{spsVOLE} - \text{pcGGM}}$ (Figure 6) UC-realizes functionality $F_{\text{spsVOLE}}$ (Figure 2) without global-key queries against any semi-honest adversary in the $(F_{\text{COT}}, F_{\text{svoLE}})$-hybrid model.

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

- The two random elements $(c_1, \mu)$ to be sent by the sender in $\Pi_{\text{spsVOLE} - \text{pcGGM}}$ can be compressed via the PRF technique for $\Pi_{\text{spsCOT}}$. If there are $t$ parallel executions, all such random messages can also be compressed in batch.

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

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

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

**Extend:** $P_0$ and $P_1$ input $N = 2^n$ and use pcGGM (c.f. Figure 5) for $n$, $\mathbb{K}$, keyed hash function $H_S: \mathbb{F}_{2^\lambda} \rightarrow \mathbb{F}_{2^\lambda}$, and function $\text{Convert}_K: \mathbb{F}_{2^\lambda} \rightarrow \mathbb{K}$.

1. $P_0$ and $P_1$ send (extend, $n$) to $\mathcal{F}_{\text{COT}}$, which returns $(K_1, \ldots, K_n) \in \mathbb{F}_{2^\lambda}^n$ to $P_0$ and $(r_1, \ldots, r_n), (M[r_1], \ldots, M[r_n]) \in \mathbb{F}_2 \times \mathbb{F}_{2^\lambda}$ to $P_1$ such that $M[r_i] = K[r_i] \oplus r_i \cdot \Delta$ for $i \in [1, n]$.
2. $P_0$ and $P_1$ send (extend, 1) to $\mathcal{F}_{\text{sVOLE}}$, which returns $K[s] \in \mathbb{K}$ to $P_0$ and $(s, M[s]) \in \mathbb{F} \times \mathbb{K}$ to $P_1$ such that $M[s] = K[s] + s \cdot \Gamma$.
3. $P_0$ samples $\beta \leftarrow \mathbb{F}^*$, sets $M[\beta] := M[s]$, and sends $d := s - \beta \in \mathbb{F}$ to $P_0$.
   $P_0$ sets $K[\beta] := K[s] + d \cdot \Gamma$ such that $M[\beta] = K[\beta] + \beta \cdot \Gamma$.
4. $P_0$ samples $(c_1, \mu) \leftarrow \mathbb{F}_{2^\lambda}^2$ and sets $K[r_1] \oplus c_1$, $\psi := \text{Convert}_K(H_S(\mu \oplus M[r_1] \oplus \sigma \cdot \Delta)) + K^n_\sigma$ for $\sigma \in \{0, 1\}$, and $\delta := K^n_0 + K^n_1 - K[\beta]$.
   $P_0$ sends $(c_1, \ldots, c_{n-1}, \mu, c_n, \psi)$ to $P_1$.
5. $P_1$ sets $\alpha := \alpha_1 \ldots \alpha_n := r_1 \ldots r_n \in [0, N)$, $K^{\alpha_i}_n := M[r_i] \oplus c_i$ for $i \in [1, n-1]$, $K^{\alpha_n}_n := c^{\alpha_n}_n - \text{Convert}_K(H_S(\mu \oplus M[r_n]))$, and $\psi := \text{Unit}_\psi(N, \alpha, \beta)$, $\psi := \text{pcGGM.PuncFullEval}(\alpha, \{K^{\alpha_x}_n\}_{x \in [1, n]}, \psi + M[\beta])$.
6. $P_0$ outputs $\nu$ and $P_1$ outputs $(u, \nu)$.

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

**Complexity analysis.** Consider the complexity per execution when the PRF-based optimization is used in $t$ parallel executions. $\Pi_{\text{sVOLE–pcGGM}}$ uses $n$ COT correlations and one sVOLE correlation. $P_0$ sends $(n - 2) \cdot \lambda + 3 \cdot \log |\mathbb{K}| + \frac{\lambda}{2}$ bits, and $P_1$ sends $\log |\mathbb{F}|$ bits. The computation per party is dominated by the tree expansion with $1.5N$ RP calls. Compared with [BCG+19a, WYKW21] (c.f. Section 4.1.2), our protocol roughly halve the communication, and the reduction in computation is 25%. Note that this computation cost does not include any PRG invocation in $\text{Convert}_K$, which is simply implemented from cheap modulo operations for the field size $|\mathbb{K}|$ considered in many sVOLE applications, e.g., [WYKW21, YSWW21, RS21, WXY+21].

5 DPF and DCF Correlation Generation

We model DPF/DCF correlation generation in functionality $\mathcal{F}_{\text{FSS}}$ (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. $\mathcal{F}_{\text{FSS}}$ 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 $\mathcal{F}_{\text{FSS}}$ also works in this general case.
the same complexity as those in the prior work. Its key generation algorithm uses about \( n \) \( 2^{\lambda} \) \( RP \) calls in the state-of-the-art construction of \([BGI16]\). This hash key can be reused across polynomially many \( FSS \) key pairs.

Note that \( DPF \) and \( DCF \) schemes 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 and highlight our complexity improvement.

### 5.1 DPF and DCF Schemes

Note that \( DPF \) and \( DCF \) schemes 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 and highlight our complexity improvement.

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

We consider the group \( \mathcal{G} \) (e.g., in \([GI14, BGI16, Ds17, BGI19, BCG+21]\)) whose size allows the PRG-free implementation of \( \text{Convert}_{\mathcal{G}} \) (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 \( n \cdot \lambda + (\lambda + 1) + \log |\mathcal{G}| \) bits, and the evaluation algorithm takes about \( n \) \( RP \) calls, both remaining the same complexity as those in the prior work.

---

### Functionality \( F_{\text{FSS}} \)

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

**Correlation generation:** Upon receiving \((\text{gen}, (\alpha)_b, (\beta)_b)\) from \( P_b \) for each \( b \in \{0, 1\} \), where \((\alpha)_b, (\beta)_b \in [0, N] \times R\):

1. Set \( \alpha := (\alpha)_0 \oplus (\alpha)_1 \in [0, N] \), \( \beta := (\beta)_0 + (\beta)_1 \in R \), and \( r \in R^N \) such that
   - If \( FSS = DPF \), \( r[j] = 0 \) for \( j \in [0, N], j \neq \alpha \), and \( r[\alpha] = \beta \).
   - If \( FSS = DCF \), \( r[j] = 0 \) for \( j \in [0, N], j \geq \alpha \), and \( r[j] = \beta \) otherwise.

2. If both parties are honest, sample \((r)_0^\lambda, (r)_1^\lambda \leftarrow R^N \) such that \((r)_0^\lambda + (r)_1^\lambda = r \); otherwise (i.e., \( P_b \) is corrupted), receive \((r)_b^\lambda \in R^N \) from the adversary and recompute \((r)_{1-b}^\lambda := r - (r)_b^\lambda \in R^N \).

3. Send \((r)_0^\lambda \) to \( P_0 \) and \((r)_1^\lambda \) to \( P_1 \).

Figure 7: Functionality for \( DPF/DCF \) correlation generation.
Parameters: Domain size $N = 2^n$ for $n \in \mathbb{N}$. Group $G$. Keyed hash function $H_S : F_{2^\lambda} \rightarrow F_{2^\lambda}$. Function $\text{Convert}_G : \{0,1\}^* \rightarrow G$ (cf. Appendix F.1).

**DPF.Gen**$(1^\lambda, (\alpha, \beta, n, G))$:
1. Parse $\alpha = \alpha_1 \ldots \alpha_n \in \{0,1\}^n$ and $\beta \in G$.
2. Sample $\Delta \leftarrow \{0,1\}^\lambda$ such that $\text{lsb}(\Delta) = 1$.
3. Sample $(s_0^\alpha || t_0^\alpha), (s_0^\beta || t_0^\beta) \leftarrow \{0,1\}^\lambda$ such that $(s_0^\alpha || t_0^\alpha) + (s_0^\beta || t_0^\beta) = \Delta$.
4. for $i \in [1, n-1]$ do
5. \hspace{1em} $\text{CW}_i := H_S(s_0^\alpha \oplus t_0^\alpha, s_0^\beta \oplus t_0^\beta) \oplus \alpha_i \cdot \Delta$
6. \hspace{1em} $(s_0^\alpha \oplus t_0^\alpha) := H_S(s_0^\alpha \oplus t_0^\alpha, \alpha_i \cdot (s_0^\beta \oplus t_0^\beta) \oplus \text{CW}_i$
7. \hspace{1em} $(s_0^\beta \oplus t_0^\beta) := H_S(s_0^\beta \oplus t_0^\beta, \alpha_i \cdot (s_0^\alpha \oplus t_0^\alpha) \oplus \text{CW}_i$
8. \hspace{1em} $(\text{high}_0[\sigma] || \text{low}_0[\sigma]) := H_S((s_0^\alpha \oplus t_0^\alpha, \sigma) \oplus \text{high}_0[\sigma])$ for $\sigma \in \{0,1\}$
9. \hspace{1em} $(\text{high}_1[\sigma] || \text{low}_1[\sigma]) := H_S((s_0^\beta \oplus t_0^\beta, \sigma) \oplus \text{low}_1[\sigma])$ for $\sigma \in \{0,1\}$
10. \hspace{2em} $\text{HCW} := \text{high}_0[\alpha_n] \oplus \text{high}_1[\alpha_n]$
11. \hspace{2em} $\text{LCW}[0] := \text{low}_0[0] \oplus \text{low}_1[0] \oplus \alpha_n$, $\text{LCW}[1] := \text{low}_0[1] \oplus \text{low}_1[1] \oplus \alpha_n$
12. \hspace{2em} $\text{CW}_n := (\text{HCW} || \text{LCW}[0] || \text{LCW}[1])$
13. \hspace{2em} $(s_b^\alpha || t_b^\alpha) := ((\text{high}_0[\alpha_n]) || \text{low}_0[\alpha_n]) \oplus t_n^{\alpha - 1} \cdot (\text{HCW} || \text{LCW}[\alpha_n])$
14. \hspace{2em} $(s_b^\beta || t_b^\beta) := ((\text{high}_1[\alpha_n]) || \text{low}_1[\alpha_n]) \oplus t_n^{\beta - 1} \cdot (\text{HCW} || \text{LCW}[\alpha_n])$
15. \hspace{2em} $\text{CW}_{n+1} := (t_n^0 - t_n^1) \cdot (\text{Convert}_G(s_0^\alpha) - \text{Convert}_G(s_0^\beta) + \beta)$
16. \hspace{2em} $k_b := ((s_b^\alpha || t_b^\alpha), \{\text{CW}_i \}_{i \in [1,n+1]})$ for $b \in \{0,1\}$
17. return $(k_0, k_1)$

**DPF.Eval**$(b, k_b, x)$:
1. Parse $k_b = (s_b^\alpha || t_b^\alpha), \{\text{CW}_i \}_{i \in [1,n+1]}$, $\text{CW}_{n} = (\text{HCW} || \text{LCW}[0] || \text{LCW}[1])$, and $x = x_1 \ldots x_n \in \{0,1\}^n$.
2. for $i \in [1, n-1]$ do
3. \hspace{1em} $(s_b^\alpha || t_b^\alpha) := H_S(s_b^\alpha \oplus t_b^\alpha, x_i) \cdot (s_b^\alpha || t_b^\alpha) \cdot t_b^\alpha$ \cdot \text{CW}_i
4. \hspace{1em} $(\text{high}[,x_n] || \text{low}[x_n]) := H_S((s_n^{\alpha - 1} \oplus t_n^{\alpha - 1}) \oplus x_n$
5. \hspace{1em} $(s_b^\alpha || t_b^\alpha) := (\text{high}[,x_n] || \text{low}[x_n]) \oplus t_n^{\alpha - 1} \cdot (\text{HCW} || \text{LCW}[x_n])$
6. return $y_b := (-1)^b \cdot (\text{Convert}_G(s_b^\alpha) + t_b^\alpha \cdot \text{CW}_{n+1})$

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

In our DCF scheme, the full-domain evaluation algorithm 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 $n \cdot \lambda + (\lambda + 1) + (n + 1) \cdot \log |G|$ 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.

**Theorem 4.** Given CCR function $H : \{0,1\}^\lambda \rightarrow \{0,1\}^\lambda$, function $\text{Convert}_G : \{0,1\}^{\lambda - 1} \rightarrow G$ (Appendix F.1), and the keyed hash function $H_S(x) := H(S \oplus x)$ with key $S \leftarrow \{0,1\}^\lambda$, (DPF.Gen, DPF.Eval) (Figure 8) is a DPF scheme with domain $[0, N)$ and range $G$.

**Theorem 5.** Given CCR function $H : \{0,1\}^\lambda \rightarrow \{0,1\}^\lambda$, function $\text{Convert}_G : \{0,1\}^\ell \rightarrow G$ (Appendix F.1) with $\ell \in (\lambda - 1, \lambda)$, and the keyed hash function $H_S(x) := H(S \oplus x)$ with key $S \leftarrow \{0,1\}^\lambda$, (DCF.Gen, DCF.Eval) (Figure 9) is a DCF scheme with domain $[0, N)$ and range $G$.

### 5.2 DPF Correlation Generation

We define a leveled evaluation algorithm **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 $\text{CW}_i$ for the $i$-th level, outputs all nodes one the $i$-th level.
Parameters: Domain size $N = 2^n$ for $n \in \mathbb{N}$. Group $\mathbb{G}$. Keyed hash function $H_S : \mathbb{F}_{2^\lambda} \rightarrow \mathbb{F}_{2^\lambda}$, Function $\text{Convert}_G : \{0,1\}^* \rightarrow \mathbb{G}$ (c.f. Appendix F.1).

DCF.Gen$(1^\lambda, (\alpha, \beta, n, \mathbb{G}))$:
1. Parse $\alpha = \alpha_1 \ldots \alpha_n \in \{0,1\}^n$ and $\beta \in \mathbb{G}$. Let $\alpha_0 := 0$.
2. Run $(k_0', k_1')$ := DPF.Gen$(1^\lambda, (\alpha, -\alpha_n \cdot \beta, n, \mathbb{G}))$ and store its internal variables.
3. for $i \in [1, n]$ do
   4. $v_0^i := H_S((s_0^{i-1} \parallel t_0^{i-1}) \oplus 2)$
   5. $v_1^i := H_S((s_1^{i-1} \parallel t_1^{i-1}) \oplus 2)$
   6. $\text{VCW}_i := (t_0^{i-1} - t_1^{i-1}) \cdot (\text{Convert}_G(v_1^i) - \text{Convert}_G(v_0^i) + (\alpha_i - \alpha_{i-1}) \cdot \beta)$
7. $k_b := (k_b', \{\text{VCW}_i\}_{i \in [1, n]})$ for $b \in \{0,1\}$
8. return $(k_0, k_1)$

DCF.Eval$(b, k_b, x)$:
1. Parse $k_b = (k_b', \{\text{VCW}_i\}_{i \in [1, n]})$. Let $V_b^0 := 0 \in \mathbb{G}$.
2. Run $y_b^i := \text{DPF.Eval}(b, k_b, x)$ and store its internal variables.
3. for $i \in [1, n]$ do
   4. $v_0^i := H_S((s_0^{i-1} \parallel t_0^{i-1}) \oplus 2)$
   5. $V_b^i := V_b^{i-1} + (-1)^{t_b^i} \cdot (\text{Convert}_G(v_b^i) + t_b^{i-1} \cdot \text{VCW}_i)$
6. return $y_b := y_b^i + V_b^n$

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

In Figure 10, we present our DPF correlation generation protocol $\Pi_{\text{DPF}}$. This protocol operates in the $(\mathcal{F}_{\text{COT}}, \mathcal{F}_{\text{Rand}}, \mathcal{F}_{\text{OLE}})$-hybrid model. $\mathcal{F}_{\text{Rand}}$ is the standard coin-tossing functionality that outputs a uniform string to both parties. $\mathcal{F}_{\text{OLE}}$ is the functionality for oblivious linear evaluation (OLE) on ring $\mathbb{R}$, where $P_0$ (resp., $P_1$) is given random $(x_0, z_0) \in \mathbb{R}^N \times \mathbb{R}^N$ (resp., $(x_1, z_1) \in \mathbb{R}^N \times \mathbb{R}^N$) such that $z_0 + z_1 = x_0 \odot x_1$, and $\odot$ denotes component-wise multiplication. We refer readers to Appendix F.2 and Appendix F.3 for the definitions and instantiations of $\mathcal{F}_{\text{Rand}}$ and $\mathcal{F}_{\text{OLE}}$. If $\beta$ is a bit-string, no $\mathcal{F}_{\text{OLE}}$ call is required in $\Pi_{\text{DPF}}$.

$\Pi_{\text{DPF}}$ requires $\mathcal{F}_{\text{Rand}}$ for the following reason. Observe that $\Pi_{\text{DPF}}$ reuses the same global offset $\Delta$ as the roots of polynomially many shared trees, each of which defines a fresh DPF correlation. Thus, the two shares of this identical root should be “re-randomized” to avoid the identical per-party shares of many defined correlations. The two parties achieve this re-randomization by calling $\mathcal{F}_{\text{Rand}}$ for a public randomness $W \leftarrow \{0,1\}^\lambda$ and XORing this value to their shares of $\Delta$, respectively.

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

Security. We analyze correctness and prove Theorem 6 in Appendix D.4.

Theorem 6. Given CCR function $H : \{0,1\}^\lambda \rightarrow \{0,1\}^\lambda$, function $\text{Convert}_R : \{0,1\}^{\lambda-1} \rightarrow \mathbb{R}$ (Appendix F.1), and the keyed hash function $H_S(x) := H(S \oplus x)$ with key $S \leftarrow \{0,1\}^\lambda$, protocol $\Pi_{\text{DPF}}$ (Figure 10) UC-realizes functionality $\mathcal{F}_{\text{DPF}}$ (Figure 7) against any semi-honest adversary in the $(\mathcal{F}_{\text{COT}}, \mathcal{F}_{\text{Rand}}, \mathcal{F}_{\text{OLE}})$-hybrid model. If $\mathbb{R} = \mathbb{F}_{2^\mu}$ for out $\in \mathbb{N}$, protocol $\Pi_{\text{DPF}}$ never invokes functionality $\mathcal{F}_{\text{OLE}}$.

Communication optimization. $\Pi_{\text{DPF}}$ has the following two optimizations:

- For $t$ parallel executions of $\Pi_{\text{DPF}}$ (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 $\mu_b$’s in these executions via a PRF $F : \mathbb{F}_{2^\lambda} \times \{0,1\}^* \rightarrow \mathbb{F}_{2^\lambda}$ with a fresh key $k_{\text{prf},b} \leftarrow \mathbb{F}_{2^\lambda}$ sampled after
Protocol \( \Pi_{DPF} \)

**Parameters:** Domain size \( N = 2^n \) for \( n \in \mathbb{N} \). Ring \( \mathcal{R} \). Keyed hash function \( H_S : \mathbb{F}_{2^n} \rightarrow \mathbb{F}_{2^n} \). Function \( \text{Convert}_\mathcal{R} : \{0,1\}^* \rightarrow \mathcal{R} \). Let \( H' := h_b \circ H_S \).

**DPF correlation generation:** For each \( b \in \{0,1\} \), \( P_b \) inputs \( (\langle \alpha \rangle_b, \langle \beta \rangle_b) \in [0,N] \times \mathcal{R} \) and runs with \( P_{1-b} \) symmetrically.

1. The two parties run preprocessing sub-protocol \( \Pi_{PREP} \) (Figure 11), which, for each \( b \in \{0,1\} \), returns \( (\langle \Delta_b \rangle_b, \{K[\langle \alpha_i \rangle_{1-b}], M[\langle \alpha_i \rangle_b]\}_i \in [1,n]) \) to \( P_b \) such that \( h_b(\langle \Delta \rangle_0 \oplus \langle \Delta \rangle_1) = 1 \), and \( M[\langle \alpha_i \rangle_b] = K[\langle \alpha_i \rangle_b] + \langle \alpha_i \rangle_b \cdot \langle \Delta \rangle_{1-b} \in \{0,1\}^\lambda \) for \( i \in [1,n] \) and \( b \in \{0,1\} \).

2. The two parties send \( \text{(sample, }\lambda) \) to \( \mathcal{F}_{\text{Rand}} \), which returns \( W \in \{0,1\}^\lambda \) to them.

3. \( P_b \) computes \( (s_b^0,0, s_b^0,0) := \langle \Delta \rangle_b \oplus W \). For \( i \in [1,n-1] \), \( P_b \) sends to \( P_{1-b} \)

\[
\langle CW_i \rangle_b := (\oplus_j \in [0,2^{n-1}]) H_S(s_b^{i-1,j} \parallel t_b^{i-1,j}) \oplus (\langle \alpha_i \rangle_b) \cdot \langle \Delta \rangle_b \oplus K[\langle \alpha_i \rangle_{1-b}] \oplus M[\langle \alpha_i \rangle_b],
\]
receives \( \langle CW_i \rangle_{1-b} \) from \( P_{1-b} \), and computes \( CW_i := \langle CW_i \rangle_b \oplus \langle CW_i \rangle_{1-b} \) and

\[
\{(s_b^{i,j} \parallel t_b^{i,j})\}_{j \in [0,2^{n-1}]} := \text{DPF.NextLevel}(i, \{(s_b^{i-1,j} \parallel t_b^{i-1,j})\}_{j \in [0,2^{n-1}]), CW_i}\).
\]

4. \( P_b \) samples \( \mu_b \leftarrow \{0,1\}^\lambda \), computes

\[
(XH_b[\sigma] \parallel XL_b[\sigma]) := \oplus_j \in [0,2^{n-1}] H_S((s_b^{n-1,j} \parallel t_b^{n-1,j}) \oplus \sigma) \quad \text{for } \sigma \in \{0,1\},
\]

\[
d_b := H'(\mu_b \oplus K[\langle \alpha_n \rangle_{1-b}]) \oplus H'(\mu_b \oplus K[\langle \alpha_n \rangle_{1-b}], \langle \Delta \rangle_b) \oplus (XH_b[0] \oplus XL_b[1]),
\]
sends \( (\mu_b, d_b) \) to \( P_{1-b} \), and receives \( (\mu_{1-b}, d_{1-b}) \) from \( P_{1-b} \). Then, \( P_b \) computes

\[
\langle HCW \rangle_b := XH_b[\langle \sigma \rangle_b] \oplus H'(\mu_b + K[\langle \alpha_n \rangle_{1-b}]) \oplus H'(\mu_{1-b} + M[\langle \alpha_n \rangle_b]),
\]

\[
\langle LCW \rangle_b := XL_b[0] \oplus \langle \alpha_n \rangle_b, \quad \langle LCW \rangle_{1-b} := XL_b[0] \oplus \langle \alpha_n \rangle_b,
\]
sends \( \langle CW_n \rangle_b := (\langle HCW \rangle_b \parallel \langle LCW \rangle_b) \parallel \langle LCW \rangle_{1-b} \) to \( P_{1-b} \), receives \( \langle CW_n \rangle_{1-b} \) from \( P_{1-b} \), and computes

\[
\langle CW_n \rangle := \langle CW_n \rangle_b \oplus \langle CW_n \rangle_{1-b} \quad \text{and}
\]

\[
\{(s_b^{n-1,j} \parallel t_b^{n-1,j})\}_{j \in [0,2^{n-1}]} := \text{DPF.NextLevel}(n, \{(s_b^{n-1,j} \parallel t_b^{n-1,j})\}_{j \in [0,2^{n-1}]), CW_n}\).
\]

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

\( P_b \) sends \( \langle CW_{n+1} \rangle_b^A := (\sum_{j \in [0,N]} \text{Convert}_{\mathcal{R}}(s_b^{n,j})) \parallel \langle \beta \rangle_b \) to \( P_{1-b} \), receives \( \langle CW_{n+1} \rangle_{1-b} \) from \( P_{1-b} \), and computes \( \langle CW_{n+1} \rangle := \langle CW_{n+1} \rangle_b^A + \langle CW_{n+1} \rangle_{1-b}^A \).

**((General ring \( \mathcal{R} \), using \( \mathcal{F}_{\text{OLE}} \)**)

The two parties send \( \text{(extend, }2) \) to \( \mathcal{F}_{\text{OLE}} \), which, for each \( b \in \{0,1\} \), returns \( (x_b, z_b, x_b) \in \mathbb{R}^2 \times \mathbb{R}^2 \) to \( P_b \). \( P_b \) computes \( t_b := (-1)^{b} \cdot \sum_{j \in [0,N]} t_b^{n,j}, s_b := (-1)^{1-b} \cdot \sum_{j \in [0,N]} \text{Convert}_{\mathcal{R}}(s_b^{n,j}) \parallel \langle \beta \rangle_b^A \) and \( (\gamma_b, \zeta_b) := (t_b, s_b) + (x_b[1-b], x_b[1-b]) \). \( P_b \) sends \( (\gamma_b, \zeta_b) \) to \( P_{1-b} \), and receives \( \langle \gamma_{1-b}, \zeta_{1-b} \rangle \) from \( P_{1-b} \). Then, \( P_b \) sends \( \langle CW_{n+1} \rangle_b^A := t_b \cdot s_b + t_b \cdot \zeta_{1-b} - x_b[1-b] \cdot \gamma_{1-b} + z_b[0] + z_b[1] \) to \( P_{1-b} \), receives \( \langle CW_{n+1} \rangle_{1-b} \) from \( P_{1-b} \), and computes \( \langle CW_{n+1} \rangle := \langle CW_{n+1} \rangle_b^A + \langle CW_{n+1} \rangle_{1-b}^A \).

6. \( P_b \) computes a vector \( \langle r \rangle_b^A \in \mathcal{R}^N \) such that, for \( j \in [0,N] \), \( \langle r[j] \rangle_b^A := (-1)^{b} \cdot (\text{Convert}_{\mathcal{R}}(s_b^{n,j}) + t_b^{n,j} \cdot CW_{n+1}) \) and outputs \( \langle r \rangle_b^A \).

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

receiving its COT outputs (from both \( \mathcal{F}_{\text{COT}} \) and \( \mathcal{F}_{\text{COT}}^{1-b} \)) in all executions. For each execution with identifier id, the two parties define \( \mu_b := F(k_{prf,b}, \text{id}) \).
and 50% 25% word. Our savings in computation, communication, and round complexity are about $n$ and revealing each of the first $m$ bits, and the computation per party is dominated by the tree expansion in $n$ COT correlations per party and one PRF-based optimization is used in $t$ parallel executions. The cost is symmetric. $\Pi_{\text{PF}}$ uses $n$ COT correlations per party and one $F_{\text{Rand}}$ call. Each party sends $(n+1) + (n+1) \cdot \lambda + \frac{3}{2} + \log |\mathcal{R}|$ bits. The computation per party is dominated by the tree expansion in $n$ DPF calls, or $1.5N$ RP calls. $\Pi_{\text{PF}}$ 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 $n$ COT correlations according to [IKNP03, Bea95]. Using COT correlations, 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.

<table>
<thead>
<tr>
<th>$R = F_{2^{127}}$</th>
<th>$n = 20$</th>
<th>$n = 22$</th>
<th>$n = 24$</th>
<th>$n = 26$</th>
<th>$n = 28$</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAN</td>
<td>50</td>
<td>120</td>
<td>397</td>
<td>1501</td>
<td>5920</td>
</tr>
<tr>
<td>WAN</td>
<td>2752</td>
<td>3020</td>
<td>3492</td>
<td>4786</td>
<td>9355</td>
</tr>
<tr>
<td>$R = F_2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LAN</td>
<td>29</td>
<td>30</td>
<td>34</td>
<td>52</td>
<td>120</td>
</tr>
<tr>
<td>WAN</td>
<td>2930</td>
<td>3132</td>
<td>3337</td>
<td>3554</td>
<td>3823</td>
</tr>
</tbody>
</table>

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

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

- Another method to save the communication for random $\mu_b$’s is to replace $H_2$ by a hash function that meets “CCR for naturally derived keys” [ZRE15, GKWY20], which can also be implemented in one RP call. Note that $\mu_b$ 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 protocol execution when the first PRF-based optimization is used in $t$ parallel executions. The cost is symmetric. $\Pi_{\text{PF}}$ uses $n$ COT correlations per party and one $F_{\text{Rand}}$ call. Each party sends $(n+1) + (n+1) \cdot \lambda + \frac{3}{2} + \log |\mathcal{R}|$ bits. The computation per party is dominated by the tree expansion in $n$ DPF calls, or $1.5N$ RP calls. $\Pi_{\text{PF}}$ 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 $n$ COT correlations according to [IKNP03, Bea95]. Using COT correlations, 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.

22
Protocol $\Pi_{DCF}$

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

**DCF correlation generation:** For each $b \in \{0,1\}$, $P_b$ inputs $(\langle \alpha \rangle_b, \langle \beta \rangle_b) \in [0,N] \times \mathcal{R}$ and runs $\Pi_{DFF}$ (Figure 8) with $P_{1-b}$ symmetrically, with the following modifications to the corresponding steps (with the same Step 1 and 2 in $\Pi_{DFF}$):

3. Along with $\langle \text{CW}_i \rangle_b$, for $i \in [1, n-1]$, $P_b$ samples $x^i_b \leftarrow \{0,1\}^\lambda$, computes

\[ y^i_b := H^*(x^i_b \oplus K[\langle \alpha \rangle_{1-b}]) - H^*(x^i_b \oplus K[\langle \alpha \rangle_{1-b}] \oplus \langle \Delta \rangle_b) + \langle \beta \rangle_b^A - 2 \cdot \langle \alpha \rangle_b \cdot \langle \beta \rangle_b^A, \]

sends $(x^i_b, y^i_b)$ to $P_{1-b}$, receives $(x^i_{1-b}, y^i_{1-b})$ from $P_{1-b}$, and computes

\[ \langle \alpha^* \cdot \beta \rangle_b^A := \langle \alpha \rangle_b \cdot \langle \beta \rangle_b^A - H^*(x^i_b \oplus K[\langle \alpha \rangle_{1-b}]) + H^*(x^i_{1-b} \oplus M[\langle \alpha \rangle_b]) + \langle \alpha \rangle_b \cdot y^i_{1-b}. \]

4. Along with $\langle \text{CW}_n \rangle_b$, $P_b$ repeats Step 3 for $i = n$ and computes $\langle \alpha_n \cdot \beta \rangle_b^A$.

5. For $i \in [1, n]$ and $j \in [0, 2^{i-1}]$, $P_b$ computes $v^{i,j}_b := H_S((s^{i-1,j}_b \| t^{i-1,j}_b) \oplus 2)$ and $\langle \alpha_0 \cdot \beta \rangle_b^A := 0$. $P_b$ computes $\langle \text{CW}_{n+1} \rangle_b^A$ by using $\langle \alpha_n \cdot \beta \rangle_b^A$ instead of $\langle \beta \rangle_b^A$, and:

**Binary field $\mathcal{R} = \mathbb{F}_{2^m}$, without $\mathcal{F}_{\text{OLE}}$**

Along with $\langle \text{CW}_{n+1} \rangle_b^A$, $P_b$ sends $\langle \langle \text{VCW} \rangle_b^A \rangle_i \in [1,n]$ to $P_{1-b}$, where

\[ \langle \text{VCW}_i \rangle_b^A := \Bigl( \sum_{j \in [0,2^{i-1}]} \text{Convert}_\mathcal{R}(v^{i,j}_b) \Bigr) + \langle \alpha^* \cdot \beta \rangle_b^A - \langle \alpha_{i-1} \cdot \beta \rangle_b^A \]

for $i \in [1, n]$, receives $\{ \langle \text{VCW}_i \rangle_b^A \}_{i \in [1, n]}$ from $P_{1-b}$, and computes $\text{VCW}_i := \langle \text{VCW}_i \rangle_b^A + \langle \text{VCW}_i \rangle_{1-b}^A$ for $i \in [1, n]$.

**General ring $\mathcal{R}$, using $\mathcal{F}_{\text{OLE}}$** For $i \in [1, n]$ in parallel:

The two parties send $\langle \text{extend}, 2 \rangle$ to $\mathcal{F}_{\text{OLE}}$, which, for each $b \in \{0,1\}$, returns $(x^i_{b,i}, x^i_{b,i}) \in \mathcal{R}^2 \times \mathcal{R}^2$ to $P_b$. Along with $\langle \gamma_6, \zeta \rangle$, $P_b$ computes

\[ t^i_{b,i} := (-1)^b \cdot \sum_{j \in [0,2^{i-1}]} t^{i-1,j}_b, \]

\[ v^i_{b,i} := (-1)^{1-b} \cdot \sum_{j \in [0,2^{i-1}]} \text{Convert}_\mathcal{R}(v^{i,j}_b) + \langle \alpha \cdot \beta \rangle_b^A - \langle \alpha_{i-1} \cdot \beta \rangle_b^A, \]

\[ \langle \gamma^i_{b,i}, \zeta^i_{b,i} \rangle := (t^i_{b,i}, v^i_{b,i}) + (x^i_{b,i}, x^i_{b,i}[1-b]), \]

sends $\langle \gamma^i_{b,i}, \zeta^i_{b,i} \rangle$ to $P_{1-b}$, and receives $\langle \gamma^i_{1-b,i}, \zeta^i_{1-b,i} \rangle$ from $P_{1-b}$. Then, along with $\langle \text{CW}_{n+1} \rangle_b^A$, $P_b$ sends

\[ \langle \text{VCW}_i \rangle_b^A := t^i_{b,i} \cdot v^i_{b,i} + t^i_{b,i} \cdot \zeta^i_{1-b,i} - x^i_{b,i}[1-b] \cdot \gamma^i_{1-b,i} + \zeta^i_{b,i}[0] + \zeta^i_{b,i}[1] \]

to $P_{1-b}$, receives $\langle \text{VCW}_i \rangle_{1-b}^A$ from $P_{1-b}$, and computes $\text{VCW}_i := \langle \text{VCW}_i \rangle_b^A + \langle \text{VCW}_i \rangle_{1-b}^A$ for $i \in [1, n]$.

6. Instead, $P_b$ computes a vector $\langle r \rangle_b^A \in \mathcal{R}^N$ such that, for $j \in [0, N)$, $\langle r[j] \rangle_b^A := (-1)^b \cdot \langle \text{Convert}_\mathcal{R}(s^{i,j} \oplus \langle \Delta \rangle_b) \rangle_b$, where

\[ V_b[j] := \sum_{i \in [1,n], j := j_{1} \cdots j_{i-1}} \left( \text{Convert}_\mathcal{R}(v^i_{b,i}) + t^{i-1,j}_b \cdot \langle \text{VCW}_i \rangle_b^A \right). \]

Figure 12: DCF correlation generation in the ($\mathcal{F}_{\text{COT}}, \mathcal{F}_{\text{Rand}}, \mathcal{F}_{\text{OLE}}$)-hybrid model.
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 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).** In this case, the two parties additionally require two OLE correlations, which are used to securely multiply two locally shared elements. Overall, each party sends \((n + 1) + (n + 1) \cdot \lambda + \frac{3}{1} + 3 \cdot \log |\mathcal{R}| \) 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 [BG16]. This correction word requires the level-by-level 2PC to share the last-level control bit \( t^\ell_1 \), leading to two additional bits in each OT payload per level. This share is used with two OLE correlations to compute CW\(_{n+1}\). 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. Our general-ring protocol leads to almost the same improvement as the binary-field case.

### 5.3 DCF Correlation Generation

Our DCF protocol Π_{DCF} in Figure 12 extends Π_{DPF} by also computing \( n \) value correction words and defining the evaluation result as per our DCF scheme. If \( \beta \) is a bit-string, the two parties can compute \( n \) value correction words without OLE correlations. Otherwise, for a general ring element \( \beta \), these correction words are obtained from OLE-based secure multiplication.

**Security.** We analyze correctness and prove Theorem 7 in Appendix D.5.

**Theorem 7.** Given CCR function \( H : \{0, 1\}^\lambda \rightarrow \{0, 1\}^\lambda \), function \( \text{Convert}_{\mathcal{R}} : \{0, 1\}^\ell \rightarrow \mathcal{R} \) for \( \ell \in \{\lambda - 1, \lambda\} \) (Appendix F.1), and the keyed hash function \( H_S(x) := H(S \oplus x) \) with key \( S \in \{0, 1\}^\lambda \), protocol Π_{DCF} (Figure 12) UC-realizes functionality \( \mathcal{F}_{DCF} \) (Figure 7) against any semi-honest adversary in the \((\mathcal{F}_{COT}, \mathcal{F}_{Rand}, \mathcal{F}_{OLE})\)-hybrid model. If \( \mathcal{R} = \mathbb{F}_{2^{2n}} \) for out \( n \in \mathbb{N} \), protocol Π_{DCF} never invokes functionality \( \mathcal{F}_{OLE} \).

**Communication optimization.** The optimizations (c.f. Section 5.2) for our DPF protocol Π_{DPF} also applies to the DCF protocol Π_{DCF}. Moreover, the random elements \( \{s_i^k\}_{i \in [1, n]} \) in Π_{DCF} can also be compressed using the same technique for the random \( \mu_i^k \)’s.

**Complexity analysis (binary field).** Consider the complexity per protocol execution when the PRF-based optimization is used in \( t \) parallel executions. The cost is symmetric. Π_{DCF} consumes \( n \) COT correlations per party and one \( \mathcal{F}_{Rand} \) call. Each party sends \((n + 1) + (n + 1) \cdot \lambda + \frac{3}{1} + (2n + 1) \cdot \log |\mathcal{R}| \) 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.

In contrast, the state-of-the-art protocol of [BCG+21] requires \( n \) string OTs to run GMW-style 2PC. The string OTs consume \( n \) COT correlations and have payloads of \((\lambda - 1) + 2 \cdot \log |\mathcal{R}|\) bits. Using \( n \) COT correlations, 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 \(|\mathcal{R}| \approx 2^\lambda \), the communication reduction is about 62.5%. When \( \mathcal{R} \) is sufficiently small, this reduction can be 66.6%.
Complexity analysis (general ring). \( \Pi_{DCF} \) also works for general \( \mathcal{R} \) at the cost of additionally using \( 2n + 2 \) OLE correlations. This general-ring version proceeds in \( n + 4 \) rounds, and the overall outgoing communication per party is \( (n + 1) + (n + 1) \cdot \lambda + \frac{\lambda}{2} + (4n + 3) \cdot \log |\mathcal{R}| \) 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 correlations. Our savings in communication and round complexity are about 50% ~ 66.6% and 50%, respectively, for typical ring size \( |\mathcal{R}| \leq 2^\lambda \).

Acknowledgements

Work of Kang Yang is supported in part by the National Key Research and Development Program of China (Grant No. 2022YFB2702000), and in part by the National Natural Science Foundation of China (Grant Nos. 62102037, 61932019). Work of Xiao Wang is supported in part 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 in part by the National Key Research and Development Program of China (Grant No. 2022YFB2702000), and in part by the National Natural Science Foundation of China (Grant Nos. 62022018, 61932019). Work of Zheli Liu is supported in part by the National Natural Science Foundation of China (Grant No. 62032012).

References


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 $O^\text{CCR}_{H_S,\Delta}(\cdot, \cdot)$ are pseudorandom from the view of the distinguisher unaware of the global key $\Delta$. The distinguisher can only make restricted queries to this CCR oracle in the sense that all oracle queries are the type 5 operations for $O(x_j, b_j) = O^\text{CCR}_{H_S,\Delta}(x_j, b_j)$ 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 $\tau$ can be added to hash/oracle inputs, and (iii) two syntactically identical operations are not allowed.

**Definition 3.** Let $H : \{0, 1\}^\lambda \rightarrow \{0, 1\}^\lambda$ be a function, let $O : \{0, 1\}^\lambda \times \{0, 1\} \rightarrow \{0, 1\}^\lambda$ be an oracle, and let $T \subseteq \{0, 1\}^\lambda \setminus \{0^\lambda\}$ 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 $\tau \in T$.
4. $x_i := H(x_j)$, where $x_j$ was defined by an NNO $Q_j$.
5. $x_i := O(x_j, b_j)$, 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 $\text{Real}_{H_S,\Delta}(\cdot)$ be a real-world oracle, which, on input a natural and non-trivial operation, sets $H(\cdot) = H_S(\cdot)$ and $O(\cdot, \cdot) = O^\text{CCR}_{H_S,\Delta}(\cdot, \cdot)$, executes this operation as per Definition 3, and returns the defined result. The ideal-world oracle $\text{Ideal}_{H_S}(\cdot)$ is identical to $\text{Real}_{H_S,\Delta}(\cdot)$, except that it sets $O(\cdot, \cdot)$ 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 $H : \{0, 1\}^\lambda \rightarrow \{0, 1\}^\lambda$ be a $(t, q, \rho, \epsilon)$-CCR function, let $\chi$ be a distribution on $\{0, 1\}^\lambda$ with min-entropy at least $\rho$, and let $H_S(x) := H(S \oplus x)$ for $S, x \in \{0, 1\}^\lambda$. There exists a polynomial $\text{poly}(\cdot)$ such that, for any PPT distinguisher $D$ running in time at most $t - \text{poly}(\lambda)$ and querying the oracle $\text{Real}_{H_S,\Delta}(\cdot)$ with at most $q$ natural and non-trivial operations, it holds that

$$
\Pr_{\Delta \leftarrow \chi, S \leftarrow \chi, \text{coins} \leftarrow \mathcal{U}} \left[ D^\text{Real}_{H_S,\Delta}(\cdot)(1^\lambda) = 1 \right] - \Pr_{\text{coins} \leftarrow \chi} \left[ D^\text{Ideal}_{H_S}(\cdot)(1^\lambda) = 1 \right] \leq 2\epsilon + \frac{q^2 \cdot 2^{|T|}}{2\lambda + 1},
$$

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

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

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

$$
\Pr_{\Delta \leftarrow \chi, S \leftarrow \chi, \text{coins} \leftarrow \mathcal{U}} \left[ \text{coll}_{\text{real}} \right] \leq \epsilon + \frac{q^2 \cdot 2^{|T|}}{2\lambda + 1}.
$$
Proof. For some \( f' \in \mathcal{F}_{\lambda,\lambda} \), define a hybrid oracle \( \text{Hyb}_{f',\Delta}(\cdot) \) that is identical to \( \text{Real}_{H_S,\Delta}(\cdot) \) except that it executes a type 4 operation by defining \( x_i := f'(x_j) \) and executes a type 5 operation by defining \( x_i := f'(x_j + \Delta) \oplus b_j \cdot \Delta \). In other words, \( \text{Hyb}_{f',\Delta}(\cdot) \) replaces the \( H_S \) calls in \( \text{Real}_{H_S,\Delta}(\cdot) \) by \( f' \) calls. Let \( \text{coll}_{\text{hyb}} \) denote the event that, during the interaction between \( D \) and \( \text{Hyb}_{f',\Delta}(\cdot) \), there exist two results \( x, x' \in \{0, 1\}^\lambda \) defined by two operations such that \( x = x' \).

First, we proceed to prove that

\[
\left| \Pr_{\Delta \leftarrow \chi, S \leftarrow \chi, \text{coins} \leftarrow U} [\text{coll}_{\text{real}}] - \Pr_{\Delta \leftarrow \chi, f' \leftarrow \mathcal{F}_{\lambda,\lambda}, \text{coins} \leftarrow U} [\text{coll}_{\text{hyb}}] \right| \leq \epsilon. \tag{5}
\]

Assume that, for the sake of contradiction, \( (5) \) does not hold for some deterministic polynomial-time \( D \). We show that the following adversary \( A \) can use \( D \) to break the CCR property of \( H_S \):

1. \( A \) internally runs \( D \) and samples \( \Delta \leftarrow \chi \). Upon receiving a natural and non-trivial operation \( Q_i \) from \( D \), \( A \) 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 \( \text{coins} \leftarrow U \).
   - 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 \( (x_j + \Delta, 0) \), receive a response \( r \), and define the result of \( Q_i \) to \( r \oplus b_j \cdot \Delta \).

2. \( A \) terminates when \( D \) terminates. It outputs 1 if there exist two defined results \( x, x' \) such that \( x = x' \); otherwise it outputs 0.

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

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

\[
\Pr_{S \leftarrow \chi} \left[ A^{\text{ccr}}_{\mathcal{F}_{\lambda,\lambda}}(1^\lambda) = 1 \right] = \Pr_{\Delta \leftarrow \chi, \text{coins} \leftarrow U} [\text{coll}_{\text{real}}].
\]

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

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

By the contradiction assumption, \( A \) breaks the CCR property of \( H \).
Second, we prove in the following induction that

\[
\Pr_{\Delta \leftarrow \chi, f' \leftarrow \mathcal{F}_{\lambda, \lambda}, \text{coins} \leftarrow \mathcal{U}} \left[ \text{coll}_{\text{hyb}}(q') \right] \leq \frac{q^2 \cdot 2^{|T|}}{2^{\lambda + 1}}. \tag{6}
\]

As the base case, (6) trivially holds if there is only one operation. Then, assume that (6) 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 \(\text{coll}_{\text{hyb}}(i)\) be the event that, during the interaction between \(D\) and \(\text{Hyb}_{f', \Delta}()\), there exist two results \(x, x' \in \{0, 1\}^\lambda\) such that \(x = x'\) in the first \(i \in \mathbb{N}\) operations. \(\text{coll}_{\text{hyb}} = \text{coll}_{\text{hyb}}(q)\) and

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

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

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

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

- \(\mathcal{I} = \mathcal{J} = \mathcal{K} = \emptyset\) and \(b = 0\) and at least one \(b_\tau \neq 0\). By the definition of \(\mathcal{T}\), \(x \otimes x' \neq 0\).

- At least one \(\mathcal{I}, \mathcal{J}, \mathcal{K} \neq \emptyset\) or \(b \neq 0\). This case is identical to that at least one \(\mathcal{I}, \mathcal{J}, \mathcal{K} \neq \emptyset\) 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 \(\neg \text{coll}_{\text{hyb}}(q - 1)\) (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,

\[
\Pr_{\Delta \leftarrow \chi, f' \leftarrow \mathcal{F}_{\lambda, \lambda}, \text{coins} \leftarrow \mathcal{U}} \left[ \text{coll}_{\text{hyb}}(q') \mid \neg \text{coll}_{\text{hyb}}(q' - 1) \right] \leq \frac{(q' - 1) \cdot 2^{|\mathcal{T}|}}{2^\lambda}.
\]
Using the induction assumption, we have that
\[
\Pr_{\Delta \leftarrow \chi, f', \mathcal{F}_{\lambda}, \text{coins} \leftarrow \mathcal{U}} \left[ \text{coll}_{\text{hyb}}(q') \right] \leq \Pr_{\Delta \leftarrow \chi, f', \mathcal{F}_{\lambda}, \text{coins} \leftarrow \mathcal{U}} \left[ \text{coll}_{\text{hyb}}(q' - 1) \right] + \Pr_{\Delta \leftarrow \chi, f', \mathcal{F}_{\lambda}, \text{coins} \leftarrow \mathcal{U}} \left[ \text{coll}_{\text{hyb}}(q') | \neg \text{coll}_{\text{hyb}}(q' - 1) \right] \leq \frac{(q' - 1) \cdot 2^{\lceil T \rceil}}{2^{\lambda + 1}} + \frac{(q' - 1) \cdot 2^{\lceil T \rceil}}{2^{\lambda}} < \frac{q^2 \cdot 2^{\lceil T \rceil}}{2^{\lambda + 1}}.
\]

The above induction shows that (6) holds. This lemma follows from (5) and (6).

**Lemma 3.** Let \( \mathcal{U} \), coins, and coins' be defined as per Lemma 1, and \( \text{coll}_{\text{real}} \) be the event defined as per Lemma 2. It holds that
\[
\left| \Pr_{\Delta \leftarrow \chi, S \leftarrow \chi, \text{coins} \leftarrow \mathcal{U}} \left[ \mathcal{D}_{\text{Real}_{\mathcal{H}S, \Delta}^{(\cdot)}(1^\lambda)} = 1 \right] - \Pr_{\text{coins} \leftarrow \mathcal{U}} \left[ \mathcal{D}_{\text{Ideal}_{\mathcal{H}S, \Delta}^{(\cdot)}(1^\lambda)} = 1 \right] \right| \leq \epsilon + \Pr_{\Delta \leftarrow \chi, S \leftarrow \chi, \text{coins} \leftarrow \mathcal{U}} \left[ \text{coll}_{\text{real}} \right].
\]

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

1. \( A \) internally runs \( \mathcal{D} \) and samples \( S \leftarrow \chi \). Upon receiving a natural and non-trivial operation \( Q_i \) from \( \mathcal{D} \), \( A \) 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 coins \( \leftarrow \mathcal{U} \).
   - If \( Q_i \) is a type 5 operation with input \((x_i, b_i)\) and \( x_i \) was not included in previous operations, query the CCR oracle with \((x_i \oplus S, b_i)\), receive a response \( r \), and define the result of \( Q_i \) to \( r \).

2. \( A \) terminates when \( \mathcal{D} \) terminates and outputs whatever \( \mathcal{D} \) outputs.

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

Consider the case where \( A \) is given \( \mathcal{O}_{\text{CCR}_{\mathcal{H}, \Delta}^{(\cdot)}}(\cdot) \) for some \( \Delta \leftarrow \chi \). We analyze the event \( \text{coll}_{\text{real}} \) during the interaction between \( \mathcal{D} \) and either the oracle emulated by \( A \) or the oracle \( \text{Real}_{\mathcal{H}S, \Delta}^{(\cdot)} \) under the same probability space. That is, the randomness \( \Delta \), \( S \), and coins take the same values in both cases, respectively (note that, in the emulated case, \( \text{coll}_{\text{real}} \) 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 \( \text{coll}_{\text{real}} \) does not occur. By the assumption, an induction based on the same \( \Delta \), \( S \), and coins shows that every defined result and every queried operation are the same in the two cases. Thus, for any fixed \( \Delta \), \( S \), and coins, the event \((\mathcal{D}_{\text{Real}_{\mathcal{H}S, \Delta}^{(\cdot)}(1^\lambda)} = 1) \land \neg \text{coll}_{\text{real}} \) occurs if and only if the event \((\mathcal{A}_{\mathcal{H}S, \Delta}^{(\cdot)}(1^\lambda)} = 1) \land \neg \text{coll}_{\text{real}} \) occurs. Since, in the emulation, \( A \) outputs 1 if and only if the distinguisher \( \mathcal{D} \) outputs 1, we can see that the latter event occurs if and only if \((\mathcal{A}_{\mathcal{H}S, \Delta}^{(\cdot)}(1^\lambda)} = 1) \land \neg \text{coll}_{\text{real}} \) occurs. It follows from the well-known Difference Lemma that
\[
\left| \Pr_{\Delta \leftarrow \chi, S \leftarrow \chi, \text{coins} \leftarrow \mathcal{U}} \left[ \mathcal{A}_{\mathcal{H}S, \Delta}^{(\cdot)}(1^\lambda)} = 1 \right] - \Pr_{\Delta \leftarrow \chi, S \leftarrow \chi, \text{coins} \leftarrow \mathcal{U}} \left[ \mathcal{D}_{\text{Real}_{\mathcal{H}S, \Delta}^{(\cdot)}(1^\lambda)} = 1 \right] \right| \leq \Pr_{\Delta \leftarrow \chi, S \leftarrow \chi, \text{coins} \leftarrow \mathcal{U}} \left[ \text{coll}_{\text{real}} \right].
\]
If $\mathcal{A}$ is given $f^*(\cdot, \cdot)$ for some $f^* \leftarrow \mathcal{F}_{\lambda+1, \lambda}$, we have

$$
\Pr_{f^* \leftarrow \mathcal{F}_{\lambda+1, \lambda}} \left[ \mathcal{A}^{f^*(\cdot)}(1^\lambda) = 1 \right] = \Pr_{\text{coins} \leftarrow \mathcal{U}, \mathcal{S} \leftarrow \chi} \left[ \mathcal{D}^{\text{Ideal}_{H_S}(\cdot)}(1^\lambda) = 1 \right]
$$

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 $\text{Ideal}_{H_S}(\cdot)$). 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 (7), (8), we have

$$
\left| \Pr_{\Delta \leftarrow \chi} \left[ \mathcal{A}^{\text{O}_{\text{H}}, \Delta}(\cdot)(1^\lambda) = 1 \right] - \Pr_{f^* \leftarrow \mathcal{F}_{\lambda+1, \lambda}} \left[ \mathcal{A}^{f^*(\cdot)}(1^\lambda) = 1 \right] \right|
$$

$$
= \left| \Pr_{\Delta \leftarrow \chi} \left[ \mathcal{A}^{\text{O}_{\text{H}}, \Delta}(\cdot)(1^\lambda) = 1 \right] - \Pr_{\Delta \leftarrow \chi, \mathcal{S} \leftarrow \chi} \left[ \mathcal{D}^{\text{Real}_{H_S}, \Delta}(\cdot)(1^\lambda) = 1 \right] \right|

+ \left| \Pr_{\Delta \leftarrow \chi, \mathcal{S} \leftarrow \chi} \left[ \mathcal{D}^{\text{Real}_{H_S}, \Delta}(\cdot)(1^\lambda) = 1 \right] - \Pr_{\text{coins} \leftarrow \mathcal{U}, \mathcal{S} \leftarrow \chi} \left[ \mathcal{D}^{\text{Ideal}_{H_S}(\cdot)}(1^\lambda) = 1 \right] \right|
$$

$$
\geq \left| \Pr_{\Delta \leftarrow \chi, \mathcal{S} \leftarrow \chi} \left[ \mathcal{D}^{\text{Real}_{H_S}, \Delta}(\cdot)(1^\lambda) = 1 \right] - \Pr_{\text{coins} \leftarrow \mathcal{U}, \mathcal{S} \leftarrow \chi} \left[ \mathcal{D}^{\text{Ideal}_{H_S}(\cdot)}(1^\lambda) = 1 \right] \right|

- \left| \Pr_{\Delta \leftarrow \chi} \left[ \mathcal{A}^{\text{O}_{\text{H}}, \Delta}(\cdot)(1^\lambda) = 1 \right] - \Pr_{\Delta \leftarrow \chi, \mathcal{S} \leftarrow \chi} \left[ \mathcal{D}^{\text{Real}_{H_S}, \Delta}(\cdot)(1^\lambda) = 1 \right] \right| > \epsilon,
$$

which contradicts the CCR property of $H$.

This lemma follows from Lemma 2 and Lemma 3.

\section*{B Security Proofs in Section 4}

\subsection*{B.1 Proof of Theorem 1}

\textbf{Theorem 1.} Given random permutation $\pi : \mathbb{F}_{2^\lambda} \rightarrow \mathbb{F}_{2^\lambda}$, linear orthomorphism $\sigma : \mathbb{F}_{2^\lambda} \rightarrow \mathbb{F}_{2^\lambda}$, and the correlated GGM tree (Figure 3) for field $\mathbb{F}_{2^\lambda}$ and hash function $H(x) := \pi(\sigma(x)) \oplus \sigma(x)$, protocol $\Pi_{\text{spCOT}}$ (Figure 4) UC-realizes functionality $\mathcal{F}_{\text{spCOT}}$ (Figure 2) against any semi-honest adversary in the $\mathcal{F}_{\text{cot}}$-hybrid model and the RPM.

\textit{Proof.} We consider $t \geq 1$ parallel $\Pi_{\text{spCOT}}$ executions, which have access to the same subroutine $\mathcal{F}_{\text{cot}}$ instance. For simplicity of exposition, we present the simulator $\mathcal{S}$ for a single execution, and the simulator for the $t$ parallel executions interacts with $\mathcal{F}_{\text{spCOT}}$ and simply runs $\mathcal{S}$ for each execution. $\mathcal{S}$ internally runs the real-world adversary $\mathcal{A}$ and relays messages between $\mathcal{A}$ and the environment $\mathcal{E}$. We focus on the setting where there is exact one corrupted party\footnote{We omit the case where both parties are honest. The reason is that, in most ultimate applications (e.g., generic 2PC/MPC, zero-knowledge proofs, and private set intersection) of $\mathcal{F}_{\text{spCOT}}$ (or $\mathcal{F}_{\text{spVOLE}}$), the semi-honest security in} and first analyze its correctness.
Correctness analysis. The correctness of $\Pi_{sp\text{COT}}$ follows from the two $c\text{GGM}$ full-evaluation algorithms. The output of $c\text{GGM}.\text{PuncFullEval}$ is identical to the output of $c\text{GGM}.\text{FullEval}$ everywhere except the punctured point. The correlation at the punctured point follows from Corollary 1.

Corrected $P_0$. In Initialize phase:

1. Upon receiving the first $(\text{init})$ from $A$ to $F_{\text{COT}}$, $S$ waits for $A$ to choose $\Delta$ and records this value.
   Then, $S$ sends $(\text{init})$ and $\Delta$ to $F_{sp\text{COT}}$.

In Extend phase:

2. Upon receiving $(\text{extend}, n)$ from $A$ to $F_{\text{COT}}$, $S$ waits for $A$ to choose its output $(K[r_1], \ldots, K[r_n])$.

3. $S$ receives $(c_1, \ldots, c_n)$ from $A$, sets $k := K[r_1] \oplus c_1$, and computes $(v, \ldots) := c\text{GGM}.\text{FullEval}(\Delta, k)$.
   Then, $S$ sends $(\text{extend}, N)$ and $v$ to $F_{sp\text{COT}}$.

The simulation is perfect. Without loss of generality, assume a deterministic environment $Z$. It is clear that the view of $A$ contains $\Delta$ in Initialize phase and $(K[r_1], \ldots, K[r_n])$ in the current Extend phase. These transcripts, which are essentially chosen by $Z$ itself, are identically distributed in the two executions since (i) they depend on the previous view of $A$ (which was forwarded to $Z$), and (ii) this previous view is identically distributed in the two executions (this holds for the initial iteration and can be regarded as the induction assumption for subsequent iterations).

Conditioned on some literal values of $\Delta$ and $(K[r_1], \ldots, K[r_n])$, the output $(u, w)$ of the honest $P_1$ is identically distributed in the two executions. In the real execution, $\alpha \in [0, N)$ comes from the COT choice bits $(r_1, \ldots, r_n)$, which are uniform in the presence of the corrupted $P_0$ according to $F_{\text{COT}}$. Thus, the index $\alpha$ of the only nonzero entry in the real $u \in \mathbb{F}_2^N$ is as uniform as the ideal one. As for $w \in \mathbb{F}_2^N$, in the real execution, the transcript $(c_1, \ldots, c_n)$ generated by the semi-honest $A$ in the real execution ensures that

$$K_{\alpha}^1 := M[r_1] \oplus c_1 = (K[r_1] \oplus r_1 \cdot \Delta) \oplus c_1 = k \oplus r_1 \cdot \Delta = K_0^1 \oplus \overline{\alpha}_1 \cdot \Delta,$$

$$K_{\alpha}^i := M[r_i] \oplus c_i = (K[r_i] \oplus r_i \cdot \Delta) \oplus (K[r_i] \oplus K_0^i) = K_0^i \oplus \overline{\alpha}_i \cdot \Delta, \quad \forall i \in [2, n].$$

$(c_1, \ldots, c_n)$ depends on the view of $A$ and must be identical in the two executions under the conditioned view. Additionally conditioned on $\alpha$ (or rather, $u$), Claim 1 ensures that $\{K_{\alpha}^i\}_{i \in [1, n]}$ are the sums required by $c\text{GGM}.\text{PuncFullEval}$. According to Corollary 1, the real $w$ meets the consistency $w = v \oplus u \cdot \Delta \in \mathbb{F}_2^n$ as in the ideal execution, where $v$ is determined by the view of $A$.

Corrected $P_1$. $S$ maintains two sets $Q_{\pi}, Q_{\sigma}$ of transcripts throughout its execution. $Q_{\pi}$ records the distinguisher’s queries/answers to/from the random permutation $\pi$ or its inverse $\pi^{-1}$ (where a transcript $(x, y) \in Q_{\pi}$ means that the distinguisher learns $\pi(x) = y$, regardless of whether it queried $\pi(x)$ or $\pi^{-1}(y)$). $Q_{\sigma}$ records the query $\{(r_i, M[r_i])\}_{i \in [1, n]}$ (i.e., the COT output chosen by the corrupted $P_1$) with its answer $(c_1, \ldots, c_n)$. In Initialize phase:

1. Upon receiving the first $(\text{init})$ from $A$ to $F_{\text{COT}}$, $S$ sends $(\text{init})$ to $F_{sp\text{COT}}$.

In Extend phase:

this case trivially holds if secure channel is available (see [HL10, Section 2.3]) and the consistency $w = v + u \cdot \Delta$ holds. Besides this consistency, the distribution of $((v, \Delta), (u, w))$ does not skew the joint output distribution in these applications in this case. Therefore, we need not to consider the distribution of $((v, \Delta), (u, w))$ when both parties are honest.
2. Upon receiving \((\text{extend}, n)\) from \(A\) to \(F_{cot}\), \(S\) waits for \(A\) to choose its COT transcript \(((r_1, \ldots, r_n), (M[r_1], \ldots, M[r_n]))\).

3. \(S\) samples \((c_1, \ldots, c_n) \leftarrow \mathbb{F}_2^n\), records \(\{(r_i, M[r_i])\}_{i \in [1,n]}, \{c_i\}_{i \in [1,n]}\) in \(Q_O\), sends \((c_1, \ldots, c_n)\) to \(A\), and computes \(u := \text{Unit}_{\pi_2}(N, \pi_1 \ldots \pi_n, 1)\) and

\[
\begin{align*}
  w := c_{GGM}.\text{PuncFullEval}(\pi_1 \ldots \pi_n, \{M[r_i] \oplus c_i\}_{i \in [1,n]}).
\end{align*}
\]

Then, \(S\) sends \((\text{extend}, N)\) and \((u, w)\) to \(F_{sp\text{COT}}\).

- **Global-key query.** \(S\) performs the global-key query in the following cases:

  - **query\(_1\):** \(Z\) queries the random permutation \(\pi\) with \(x\) or its inverse \(\pi^{-1}\) with \(y\). For every \(\{(r_i, M[r_i])\}_{i \in [1,n]}, \{c_i\}_{i \in [1,n]}\) \(\in Q_O\), \(S\) proceeds as follows:

    1. \(S\) defines

    \[
    \{z_i\}_{i \in [1,n]} := c_{GGM}.\text{OffPath}(\pi_1 \ldots \pi_n, \{M[r_i] \oplus c_i\}_{i \in [1,n]}),
    \]

    \[
    w_j := \oplus_{i \in [1,j-1]} z_i, \quad \forall j \in [2, n],
    \]

    where \(c_{GGM}.\text{OffPath}\) is a macro such that, on input a path labeled by \(\alpha\) and \(n\) sums \(\{K^n_{\pi i}\}_{i \in [1,n]}\) used in \(c_{GGM}.\text{PuncFullEval}\) to define an \(n\)-level correlated GGM tree except the \(n\) on-path nodes, it outputs the siblings of these on-path nodes, i.e., the \(n\) off-path nodes just leaving the \(\alpha\)-path. This macro ensures that, for \(j \in [1, n]\),

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

    2. For \(j \in [2, n]\), \(S\) extracts \(\Delta'_1\) and sends \((\text{guess}, \Delta'_1)\) to \(F_{sp\text{COT}}\) if \(Z\) queries \(\pi\) with \(x\), or extracts \(\Delta'_2\) and \((\text{guess}, \Delta'_2)\) to \(F_{sp\text{COT}}\) if \(Z\) queries \(\pi^{-1}\) with \(y\), where

    \[
    \begin{align*}
    \Delta'_1 & := \sigma^{-1}(x) \oplus w_j, \\
    \Delta'_2 & := \begin{cases} 
    \sigma^{-1}(y \oplus z_j) \oplus w_j & , \text{if } r_j = 0 \\
    \sigma^{-1}(y \oplus z_j) \oplus w_j & , \text{if } r_j = 1
    \end{cases}
    \end{align*}
    \]

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

  - **query\(_2\):** A new query-response pair \(\{(r_i, M[r_i])\}_{i \in [1,n]}, \{c_i\}_{i \in [1,n]}\) is added to \(Q_O\). \(S\) sets \(\{z_i\}_{i \in [1,n]}\) and \(\{w_j\}_{j \in [2,n]}\) as per (9) in the query\(_1\) case. Then, for \((x, y) \in Q_\pi\) and \(j \in [2, n]\), \(S\) extracts both \(\Delta'_1\) and \(\Delta'_2\) as per (11), and sends \((\text{guess}, \Delta'_1)\) and \((\text{guess}, \Delta'_2)\) to \(F_{sp\text{COT}}\).

In either case, if \(S\) receives \((\text{success})\) from \(F_{sp\text{COT}}\) for some guess \(\Delta\), \(S\) will program the random permutation \(\pi\) and its inverse \(\pi^{-1}\) such that, for each \(\{(r_i, M[r_i])\}_{i \in [1,n]}, \{c_i\}_{i \in [1,n]}\) \(\in Q_O\) up to this time and \(j \in [2, n]\),

\[
\pi(\sigma(\Delta \oplus w_j)) = \sigma(\Delta \oplus w_j) \oplus r_j \cdot (\Delta \oplus w_j) \oplus z_j.
\]

After programming, \(S\) uses the global key \(\Delta\) extracted from \(F_{sp\text{COT}}\) to emulate \(F_{cot}\) and \((c_1, \ldots, c_n)\) in any future **Extend** iteration by following the specification in \(\Pi_{sp\text{COT}}\).
We use the H-coefficient technique [Pat09, CS14] to prove that this ideal execution is computationally indistinguishable from the real one. Without loss of generality, we consider a deterministic environment \( Z \) and the transcript \( Q = (Q_\pi, Q_\sigma, \Delta) \), which is a part of the joint distribution of the view of \( A \) and the output of the honest \( P_l \) (note that the remaining part of this joint distribution is \( \nu \), and its consistency with \( Q \) will be checked in the end).

In the two executions, \( Q_\pi \) records the adversary’s queries/answers to/from the random permutation \( \pi \) or its inverse \( \pi^{-1} \). In the real execution, \( \Delta \leftarrow F_{2\lambda} \) according to \( F_{\text{COT}} \), and \( Q_\sigma \) records \((\{(r_i,M[r_i])\}_{i\in[1,n]}, \{c_{ij}\}_{i\in[1,n]} )\), the adversary’s queries/answers to/from the “oracle” jointly emulated by the non-adversarial machines (i.e., the honest \( P_l \) and the ideal functionality \( F_{\text{COT}} \)) in \( \Pi_{\text{spCOT}} \), where the consistency (12) always holds due to the construction of \( H \) and the definition of \( z_j \) and \( w_j \). In the ideal execution, \( \Delta \leftarrow F_{2\lambda} \) is sampled inside \( F_{\text{spCOT}} \), and \( Q_\sigma \) records the queries/answers with the same syntax where \( \{c_{ij}\}_{i\in[1,n]} \) are uniformly sampled by \( S \) instead. We define \( |Q_\pi| := p = p(\lambda) \) and \( |Q_\sigma| := q = q(\lambda) \).

We borrow the summary of the H-coefficient technique from [GKWy20]. Let \( T \) be the set of all possible transcripts, and let \( \Pr_{\text{real}}[\cdot] \) and \( \Pr_{\text{ideal}}[\cdot] \) be the probabilities of events in the real and ideal executions, respectively. The H-coefficient technique involves dividing \( T \) into a “bad” subset \( T_{\text{bad}} \) and a “good” subset \( T_{\text{good}} := T \setminus T_{\text{bad}} \) and showing that

\[
\Pr_{\text{ideal}}[Q \in T_{\text{bad}}] \leq \epsilon_1, \quad \forall Q \in T_{\text{good}} : \frac{\Pr_{\text{real}}[Q]}{\Pr_{\text{ideal}}[Q]} \geq 1 - \epsilon_2.
\]

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

A transcript \( Q = (Q_\pi, Q_\sigma, \Delta) \) is \textbf{bad} if it falls into one of the following cases:

- **bad1.** There exist two distinct transcript pairs \((\{(r_i,M[r_i])\}_{i\in[1,n]}, \{c_{ij}\}_{i\in[1,n]}),j \in Q_\sigma \times [2,n] \) and \((\{(r'_i,M[r'_i])\}_{i\in[1,n]}, \{c'_{ij}\}_{i\in[1,n]}),j' \in Q_\sigma \times [2,n] \) such that

\[
(w_j = w'_{j'}) \lor (\sigma(w_j) \oplus r_j \cdot (\Delta \oplus w_j) \oplus z_j = \sigma(w'_{j'}) \oplus r'_{j'} \cdot (\Delta \oplus w'_{j'}) \oplus z'_{j'})
\]

where \( z_j, w_j, z'_{j'}, \) and \( w'_{j'} \) are defined as per (9). This case captures the collision between the queries to the random permutation or its inverse.

- **bad2.** There exists a \((\{(r_i,M[r_i])\}_{i\in[1,n]}, \{c_{ij}\}_{i\in[1,n]}),j \in Q_\sigma \times [2,n] \) such that

\[
(\sigma(\Delta \oplus w_j), \ldots ) \in Q_\pi \lor (\ldots , \sigma(\Delta \oplus w_j) \oplus r_j \cdot (\Delta \oplus w_j) \oplus z_j) \in Q_\pi \]

\[
\land (\pi(\sigma(\Delta \oplus w_j)) \neq \sigma(\Delta \oplus w_j) \oplus r_j \cdot (\Delta \oplus w_j) \oplus z_j)
\]

where \( z_j \) and \( w_j \) are defined as per (9). This case captures the inconsistency in the already defined entries of the random permutation and its inverse.

Consider \textbf{bad1} in the ideal execution. For two distinct pairs in \( Q_\sigma \times [2,n], \)

\[
\Pr_{\text{ideal}}[w_j = w'_{j'}] \lor (\sigma(w_j) \oplus r_j \cdot (\Delta \oplus w_j) \oplus z_j = \sigma(w'_{j'}) \oplus r'_{j'} \cdot (\Delta \oplus w'_{j'}) \oplus z'_{j'})
\]

\[
\leq \Pr_{\text{ideal}}[w_j = w'_{j'}] + \Pr_{\text{ideal}}[\sigma(w_j) \oplus r_j \cdot (\Delta \oplus w_j) \oplus z_j = \sigma(w'_{j'}) \oplus r'_{j'} \cdot (\Delta \oplus w'_{j'}) \oplus z'_{j'}]
\]

\[
= 2^{-\lambda} + 2^{-\lambda} = 2 \cdot 2^{-\lambda},
\]

where the equality follows from that (i) the \( c_i \)'s are uniform and pairwise independent in the ideal execution, and (ii) each \( z_i \) is uniform due to the uniform \( c_i \) in (10). Taking a union bound over all distinct pairs in \( Q_\sigma \times [2,n], \) it holds that

\[
\Pr_{\text{ideal}}[\text{bad1}] \leq \frac{q^2(n-1)^2}{2\lambda}.
\]

(13)
As for \( \text{bad}_2 \), it is sufficient to bound \( \Pr_{\text{ideal}}[\text{bad}_2 \mid \neg \text{bad}_1] \) using the following three sub-cases. In the first sub-case where
\[
\forall ((\{(r_i, M[r_i])\}_{i \in [1,n]}, \{c_i\}_{i \in [1,n]}), j) \in Q_\pi \times [2,n] : \\
(\sigma(\Delta \oplus w_j), \ldots) \notin Q_\pi \land (\ldots, \sigma(\Delta \oplus w_j) \oplus r_j \cdot (\Delta \oplus w_j) \oplus z_j) \notin Q_\pi,
\]
it is clear that \( \text{bad}_2 \) will not happen.

Otherwise, in the following two complement sub-cases, \( S \) must successfully extract the global key \( \Delta \) (which is a part of the transcript \( Q \)) 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 (11). \( S \) will use \( \Delta \) to program \( \pi \) and \( \pi^{-1} \) as per (12). If the programming succeeds, \( \text{bad}_2 \) will not happen since the programming ensures the consistency (12) for every transcript in \( Q_\pi \) and every \( j \in [2,n] \). After programming, the consistency trivially holds for any future transcript in \( Q_\pi \) and \( j \in [2,n] \) since \( \Delta \) is known.

In the second sub-case where the programming comes from a new query to \( \pi \) or \( \pi^{-1} \) (i.e., \( \text{query}_1 \) case), the programming must succeed conditioned on \( \neg \text{bad}_1 \). The condition \( \neg \text{bad}_1 \) ensures that, for every \( ((\{(r_i, M[r_i])\}_{i \in [1,n]}, \{c_i\}_{i \in [1,n]}), j) \in Q_\pi \times [2,n] \), the pre-image entry \( x := \sigma(\Delta \oplus w_j) \) and the image entry \( y := \sigma(\Delta \oplus w_j) \oplus r_j \cdot (\Delta \oplus w_j) \oplus z_j \) to be programmed in \( \pi \) do not incur collision. More importantly, these two entries are not determined by the queries/answers in \( Q_\pi \) up to this time. Otherwise, \( \exists (x, \ldots) \in Q_\pi \) or \( \exists (\ldots, y) \in Q_\pi \) ahead of the current query such that the programming must have happened for the same \( Q_\pi \) (i.e., contradicting that the programming is triggered by the current query). Therefore, all transcripts in \( Q_\pi \) can be programmed as per (12). The answer to the current query is determined by the programmed result. The surely successful programming makes \( \text{bad}_2 \) impossible.

Consider the third sub-case where the programming arises from a new transcript added to \( Q_\pi \) (i.e., \( \text{query}_2 \) case). Assume that the global key \( \Delta \) to be used in the programming is extracted from this transcript, an existing \( (x^*, \ldots) \in Q_\pi \) or \( (\ldots, y^*) \in Q_\pi \), and \( j^* \in [2,n] \). Note that either \( (x^* \ldots) \in Q_\pi \) or \( (\ldots, y^*) \in Q_\pi \) has been determined and cannot be programmed. We bound the probability of this undesired sub-case. In the ideal execution,
\[
\Pr_{\text{ideal}}[\{x^* = \sigma(\Delta \oplus w_{j^*})\} \lor \{y^* = \sigma(\Delta \oplus w_{j^*}) \oplus r_{j^*} \cdot (\Delta \oplus w_{j^*}) \oplus z_{j^*}\}] \\
\leq \Pr_{\text{ideal}}[x^* = \sigma(\Delta \oplus w_{j^*})] + \Pr_{\text{ideal}}[y^* = \sigma(\Delta \oplus w_{j^*}) \oplus r_{j^*} \cdot (\Delta \oplus w_{j^*}) \oplus z_{j^*}] \\
= \Pr_{\text{ideal}}[w_{j^*} = \sigma^{-1}(x^*) \oplus \Delta] + \Pr_{\text{ideal}}[z_{j^*} = y^* \oplus \sigma(\Delta \oplus w_{j^*}) \oplus r_{j^*} \cdot (\Delta \oplus w_{j^*})] \\
= 2^{-\lambda} + 2^{-\lambda} = 2 \cdot 2^{-\lambda},
\]
where \( z_{j^*} \) and \( w_{j^*} \) are defined by the newly added \( Q_\pi \) transcript, and the probability is taken over the uniform \( c_{j^*} \). Taking a union bound over \( Q_\pi \times Q_\pi \times [2,n] \), 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
\[
\Pr_{\text{ideal}}[\text{bad}_2 \mid \neg \text{bad}_1] \leq \frac{2pq(n-1)}{2^\lambda}.
\] (14)

Using (13) and (14), it holds that
\[
\Pr_{\text{ideal}}[Q \in T_{\text{bad}}] = \Pr_{\text{ideal}}[\text{bad}_1 \lor \text{bad}_2] \\
= \Pr_{\text{ideal}}[\text{bad}_1] + \Pr_{\text{ideal}}[\text{bad}_2] - \Pr_{\text{ideal}}[\text{bad}_1 \land \text{bad}_2] \\
= \Pr_{\text{ideal}}[\text{bad}_1] + \Pr_{\text{ideal}}[\text{bad}_2 \mid \neg \text{bad}_1] \cdot \Pr_{\text{ideal}}[\neg \text{bad}_1] \\
\leq \Pr_{\text{ideal}}[\text{bad}_1] + \Pr_{\text{ideal}}[\text{bad}_2 \mid \neg \text{bad}_1] \\
\leq \frac{q^2(n-1)^2 + 2pq(n-1)}{2^\lambda}.
\]
We proceed to bound the ratio of $\Pr_{\text{real}}[\mathcal{Q}]$ to $\Pr_{\text{ideal}}[\mathcal{Q}]$ for some fixed good transcript $\mathcal{Q} = (\mathcal{Q}_\pi, \mathcal{Q}_\sigma, \Delta)$. Let $\mathcal{L} \subseteq \mathcal{Q}_\sigma \times [2, n]$ such that

$$\forall((\{r_i, M[r_i]\}_{i \in [1, n]}, \{c_i\}_{i \in [1, n]}), j) \in \mathcal{L}:
\begin{align*}
(\sigma(\Delta \oplus w_j), \ldots) & \in \mathcal{Q}_\pi \land (\ldots, \sigma(\Delta \oplus w_j) \oplus r_j \cdot (\Delta \oplus w_j) \oplus z_j) \in \mathcal{Q}_\pi.
\end{align*}$$

Conditioned on $-\text{bad}_1 \lor \text{bad}_2$ implicit in good transcripts, $\mathcal{L}$ induces a subset $\mathcal{Q}_\pi(\mathcal{L}) \subseteq \mathcal{Q}_\pi$. There exists a bijective correspondence between $(x, y) \in \mathcal{Q}_\pi(\mathcal{L})$ and $((\{r_i, M[r_i]\}_{i \in [1, n]}, \{c_i\}_{i \in [1, n]}), j) \in \mathcal{L}$ such that $x = \sigma(\Delta \oplus w_j)$ and $y = \sigma(\Delta \oplus w_j) \oplus r_j \cdot (\Delta \oplus w_j) \oplus z_j$, where $z_j$ and $w_j$ are defined in (9). It is clear that $|\mathcal{L}| = |\mathcal{Q}_\pi(\mathcal{L})| \leq \min(|\mathcal{Q}_\pi|, |\mathcal{Q}_\sigma| \cdot (n - 1)) = \min(p, q(n - 1))$.

For $\mathcal{X} \subseteq \mathcal{Q}_\pi$, let $\pi \sim \mathcal{X}$ be the event that the permutation $\pi$ is consistent with the queries/answers in $\mathcal{X}$. By the randomness of $\sigma(\Delta \oplus w_j)$ and $\sigma(\Delta \oplus w_j) \oplus r_j \cdot (\Delta \oplus w_j) \oplus z_j$ from an execution take the literal values in $\mathcal{Q}_\pi$. For $\mathcal{Y} \subseteq \mathcal{Q}_\sigma \times [2, n]$, let $(\pi, \Delta) \sim \mathcal{Y}$ be the real-world event that the permutation $\pi$ and the key $\Delta$ are consistent with the queries/answers in $\mathcal{Y}$, i.e.,

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

for $z_j$ and $w_j$ defined in (9). Let $(a)_b := a \cdot \ldots \cdot (a - b + 1)$ be falling factorial.

In the real execution, $\pi$ is a random permutation without programming, and

$$\Pr_{\text{real}}[\mathcal{Q}] = \Pr_{\text{real}}[c(1) \sim \mathcal{Q}_\sigma \land (\pi, \Delta) \sim \mathcal{Q}_\sigma \times [2, n] \land \pi \sim \mathcal{Q}_\pi \land \Delta]$$

$$= \Pr_{\text{real}}[c(1) \sim \mathcal{Q}_\sigma \land (\pi, \Delta) \sim \mathcal{Q}_\sigma \times [2, n] \land \pi \sim \mathcal{Q}_\pi \land \Delta] \cdot \Pr_{\text{real}}[\mathcal{L}]$$

$$= \Pr_{\text{real}}[(\pi, \Delta) \sim \mathcal{Q}_\sigma \times [2, n] \land \pi \sim \mathcal{Q}_\pi \land \Delta] \cdot \Pr_{\text{real}}[c(1) \sim \mathcal{Q}_\sigma \land \Delta]$$

$$= \Pr_{\text{real}}[(\pi, \Delta) \sim \mathcal{Q}_\sigma \times [2, n] \land c(1) \sim \mathcal{Q}_\sigma \land \Delta] \cdot \Pr_{\text{real}}[\mathcal{L}]$$

$$= \Pr_{\text{real}}[(\pi, \Delta) \sim \mathcal{Q}_\sigma \times [2, n] \land c(1) \sim \mathcal{Q}_\sigma \land \Delta] \cdot \Pr_{\text{real}}[\mathcal{L}]$$

$$= \Pr_{\text{real}}[(\pi, \Delta) \sim \mathcal{Q}_\sigma \times [2, n] \land \Delta] \cdot \Pr_{\text{real}}[\mathcal{L}]$$

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

$$-\text{bad}_2$$

ensures that, for each $((\{r_i, M[r_i]\}_{i \in [1, n]}, \{c_i\}_{i \in [1, n]}), j) \in \mathcal{Q}_\sigma \times [2, n]$, 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 $\mathcal{Q}_\sigma \times [2, n]$. For those pairs in $(\mathcal{Q}_\sigma \times [2, n]) \setminus \mathcal{L}$, these two entries are even not determined by all queries in $\mathcal{Q}_\pi$ due to $-\text{bad}_1$ and the definition of $\mathcal{L}$. By the randomness of $\pi$, we have

$$\Pr_{\text{real}}[(\pi, \Delta) \sim \mathcal{Q}_\sigma \times [2, n] \land \Delta] = \frac{1}{(2^\Delta - p)(q(n - 1))}.$$
Using these results, we have

\[
Pr_{\text{real}}[Q] = \frac{1}{(2^\lambda - p)q(n-1) - |\mathcal{L}| \cdot (2^\lambda)_p \cdot (2^\lambda)^q + 1}.
\]  

(15)

In the ideal execution, it holds that

\[
Pr_{\text{ideal}}[Q] = Pr_{\text{ideal}}[\pi \sim Q_\pi \mid c(n) \sim Q_O \land \Delta] \cdot Pr_{\text{ideal}}[c(n) \sim Q_O \land \Delta]
\]

\[
= Pr_{\text{ideal}}[\pi \sim Q_\pi \mid c(n) \sim Q_O \land \Delta] \cdot Pr_{\text{ideal}}[\pi \sim Q_\pi \mid c(n) \sim Q_O \land \Delta] 
\]

\[
\cdot Pr_{\text{ideal}}[c(n) \sim Q_O] \cdot Pr_{\text{ideal}}[\Delta]
\]

\[
= Pr_{\text{ideal}}[\pi \sim Q_\pi \mid c(n) \sim Q_O \land \Delta] \cdot \frac{1}{(2^\lambda)^q \cdot 2^\lambda}.
\]

By the randomness of \( \pi \) conditioned on \( \neg \text{bad}_2 \), the defined \( Q_\pi(\mathcal{L}) \) implies that

\[
Pr_{\text{ideal}}[\pi \sim Q_\pi \mid c(n) \sim Q_O \land \Delta] = 1,
\]

\[
Pr_{\text{ideal}}[\pi \sim Q_\pi \mid c(n) \sim Q_O \land \Delta] = \frac{1}{(2^\lambda - |Q_\pi(\mathcal{L})|)p - |Q_\pi(\mathcal{L})|}
\]

\[
= \frac{1}{(2^\lambda - |\mathcal{L}|)p - |\mathcal{L}|}.
\]

Using the above results and (15), we have

\[
Pr_{\text{ideal}}[Q] = \frac{1}{(2^\lambda - |\mathcal{L}|)p - |\mathcal{L}|} \cdot \frac{1}{(2^\lambda)^q \cdot 2^\lambda}
\]

\[
\leq \frac{1}{(2^\lambda - q(n-1))p - |\mathcal{L}|} \cdot \frac{1}{(2^\lambda)^q(n-1) \cdot (2^\lambda)^q + 1}
\]

\[
= \frac{1}{(2^\lambda)^{p+q(n-1)-|\mathcal{L}|} \cdot (2^\lambda)^q + 1} = Pr_{\text{real}}[Q].
\]

By the H-coefficient technique, \( Z \) can only distinguish the transcript \( Q \) in the two executions with advantage at most \((q^2(n-1)^2 + 2pq(n-1))/2^\lambda \). In fact, this is the advantage that \( Z \) can distinguish the real and the ideal executions since the output \( v \) in the joint distribution must be consistent with \( Q \) (or rather, its defined \( \Delta \) and \((u, w)) \) in either execution due to the correctness of protocol or \( F_{\text{spCOT}} \). Thus, this theorem follows.

B.2 Proof of Theorem 2

Theorem 2. Given random permutation \( \pi : \mathbb{K} \to \mathbb{K} \), linear orthomorphism \( \sigma : \mathbb{K} \to \mathbb{K} \), and the correlated GGM tree (Figure 3) for field \( \mathbb{K} \) and hash function \( H(x) := \pi(\sigma(x)) + \sigma(x) \), protocol \( \Pi_{\text{spVOLE-cGGM}} \) (Figure 13) UC-realizes functionality \( F_{\text{spVOLE}} \) (Figure 2) against any semi-honest adversary in the \( F_{\text{spVOLE}} \)-hybrid model and the RPM, for \( |\mathbb{K}| \geq 2^\lambda \).
Protocol $\Pi_{\text{spVOLE-}c\text{GGM}}$

**Parameters:** Field $\mathbb{F}$ and its extension field $\mathbb{K}$ with $|\mathbb{K}| \geq 2^4$.

**Initialize:** This procedure is executed only once.
1. $P_0$ and $P_1$ send (init) to $\mathcal{F}_\text{VOLE}$, which returns $\Delta \in \mathbb{K}$ to $P_0$. $P_0$ outputs $\Delta$.

**Extend:** $P_0$ and $P_1$ input $N = 2^n$ and use $c\text{GGM}$ (c.f. Figure 3) for $n$ and $\mathbb{K}$.
2. $P_0$ and $P_1$ send $(\text{extend},n + 1)$ to $\mathcal{F}_\text{VOLE}$, which returns $(K[s_0], \ldots, K[s_n]) \in \mathbb{K}^{n+1}$ to $P_0$ and $((s_0, \ldots, s_n), (M[s_0], \ldots, M[s_n])) \in \mathbb{K}^{n+1} \times \mathbb{K}^{n+1}$ to $P_1$ such that $M[s_i] = K[s_i] + s_i \cdot \Delta$ 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 $M[\beta] := M[s_0]$ and $M[r_i] := r_i \cdot M[\beta] - M[s_i]$ 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 $K[\beta] := K[s_0] + d_0 \cdot \Delta$ and $K[r_i] := -K[s_i] - d_i \cdot \Delta$ for $i \in [1, n]$ such that $M[\beta] = K[\beta] + \Delta$ and $M[r_i] = K[r_i] + r_i \cdot K[\beta]$ for $i \in [1, n]$.
4. $P_0$ samples $c_1 \leftarrow \mathbb{K}$ and sets $k := -K[r_1] + c_1$.
   $$(v, \{K^i_{\alpha}\}_{i \in [1, n]}) := c\text{GGM}.\text{FullEval}(K[\beta], k),$$
   and $c_i := K[r_i] + K^i_{\alpha}$ for $i \in [2, n]$. $P_0$ sends $(c_1, \ldots, c_n)$ to $P_1$.
5. $P_1$ sets $\alpha = \alpha_1 \ldots \alpha_n := \tau_1 \ldots \tau_n \in [0, N]$, $K^i_{\alpha} := (-1)^{\tau_i} \cdot (-M[r_i] + c_i)$ for $i \in [1, n]$, $u := \text{Unit}_\mathbb{F}(N, \alpha, M[\beta])$, and
   $w := c\text{GGM}.\text{PunctFullEval}(\alpha, \{K^i_{\alpha}\}_{i \in [1, n]}) + \text{Unit}_\mathbb{F}(N, \alpha, M[\beta])$.
6. $P_0$ outputs $v$ and $P_1$ outputs $(u, w)$.

Figure 13: $c\text{GGM}$-based single-point sVOLE protocol in the $\mathcal{F}_\text{VOLE}$-hybrid model.

**Proof.** This proof is similar to that in Appendix B.1, and we highlight its differences. We consider $t \geq 1$ parallel $\Pi_{\text{spVOLE-}c\text{GGM}}$ executions, which have access to the same subroutine $\mathcal{F}_\text{VOLE}$ instance. For simplicity of exposition, we present the simulator $\mathcal{S}$ for a single execution, and the simulator for the $t$ parallel executions interacts with $\mathcal{F}_{\text{spVOLE}}$ and simply runs $\mathcal{S}$ for each execution. $\mathcal{S}$ internally runs the real-world adversary $A$ and relays messages between $A$ and the environment $Z$. We also focus on the setting where there is exactly one corrupted party for the same reason in Appendix B.1.

**Correctness analysis.** Similar to our protocol $\Pi_{\text{spCOT}}$ in Appendix B.1.

**Corrupted $P_0$.** In **Initialize** phase:
1. Upon receiving the first (init) from $A$ to $\mathcal{F}_\text{VOLE}$, $\mathcal{S}$ waits for $A$ to choose $\Delta$ and records this value. Then, $\mathcal{S}$ sends (init) and $\Delta$ to $\mathcal{F}_{\text{spVOLE}}$.

In **Extend** phase:
2. Upon receiving $(\text{extend},n + 1)$ from $A$ to $\mathcal{F}_\text{VOLE}$, $\mathcal{S}$ waits for $A$ to choose its sVOLE transcript $(K[s_0], \ldots, K[s_n])$.
3. $\mathcal{S}$ sends random $(d_0, d_1, \ldots, d_n)$ to $A$ and receives $(c_1, \ldots, c_n)$ from $A$. $\mathcal{S}$ behaves as an honest $P_0$ to compute $(K[\beta], K[r_1], \ldots, K[r_n])$, $k := -K[r_1] + c_1$, and $(v, \ldots) := c\text{GGM}.\text{FullEval}(K[\beta], k)$. Then, $\mathcal{S}$ sends $(\text{extend}, N)$ and $v$ to $\mathcal{F}_\text{spVOLE}$.

The simulation is perfect. Without loss of generality, assume a deterministic environment $Z$. It is clear that the view of $A$ includes $\Delta$ in **Initialize** phase and $(K[s_0], \ldots, K[s_n]), (d_0, d_1, \ldots, d_n)$
in the current **Extend** phase. Note that \((d_0, d_1, \ldots, d_n)\) is uniform in both the real execution (due to the uniform mask \((s_0, s_1, \ldots, s_n)\)) and the ideal execution. The other two COT transcripts, which are essentially chosen by \(Z\) itself, are identically distributed in the two executions since (i) they depend on the previous view of \(A\) (which was forwarded to \(Z\)), and (ii) this previous view is identically distributed in the two executions (this holds for the initial iteration and can be regarded as the induction assumption for subsequent iterations).

The output \((u, w)\) of the honest \(P_1\) is identically distributed in the two executions conditioned on some fixed \(\Delta\), \((K[s_0], \ldots, K[s_n])\), and \((d_0, d_1, \ldots, d_n)\). It is clear that \(u \in \mathbb{F}^N\) is identically distributed in the two executions. For some fixed \(u\) with the index of its nonzero entry, it holds in the real execution that

\[
K^1_{\pi_i} := (-1)^{r_i} \cdot (M[r_i] + c_i) = (-1)^{r_i} \cdot (-K[r_i] - r_i \cdot K[\beta] + c_i) = (-1)^{r_i} \cdot (K_0 - r_i \cdot K[\beta]),
\]

\[
K^2_{\pi_i} := (-1)^{r_i} \cdot (-M[r_i] + c_i) = (-1)^{r_i} \cdot (-K[r_i] - r_i \cdot K[\beta] + K[r_i] + K_0) = (-1)^{r_i} \cdot (K_0 - r_i \cdot K[\beta]), \quad \forall i \in [2, n].
\]

Note that \((c_1, \ldots, c_n)\) depends on the view of \(A\) and must be identical in the two executions under the conditioned view. For the fixed \(u\) with \(\alpha\), Claim 1 ensures that \(\{K^1_{\pi_i}\}_{i \in [1, n]}\) are the sums required by \texttt{cGGM.PuncFullEval}. It follows from Corollary 1 that the consistency \(w = v + u \cdot \Delta \in \mathbb{F}\) holds as in the ideal execution. In particular, we have \(w[\alpha] = v[\alpha] - K[\beta] + M[\beta] = v[\alpha] + \beta \cdot \Delta\).

**Corrupted** \(P_1\). \(S\) maintains two sets \(Q_\pi, Q_O\) of transcripts throughout its execution. \(Q_\pi\) records the distinguisher’s queries/answers to/from the random permutation \(\pi\) or its inverse \(\pi^{-1}\) (where a transcript \((x, y)\) \(\in Q_\pi\) means that the distinguisher learns \(\pi(x) = y\), regardless of whether it queried \(\pi(x)\) or \(\pi^{-1}(y)\)). \(Q_O\) records the query \(((\beta, M[\beta]), \{(r_i, M[r_i])\}_{i \in [1, n]}\) (which is chosen by the corrupted \(P_1\)) with its answer \((c_1, \ldots, c_n)\). In **Initialize** phase:

1. Upon receiving the first (init) from \(A\) to \(F_{sVOLE}\), \(S\) sends (init) to \(F_{sVOLE}\).

In **Extend** phase:

2. Upon receiving (extend, \(n + 1\)) from \(A\) to \(F_{sVOLE}\), \(S\) waits for \(A\) to choose its sVOLE transcript \(((s_0, \ldots, s_n), (M[s_0], \ldots, M[s_n]))\).

3. \(S\) receives \((d_0, d_1, \ldots, d_n)\) from \(A\), samples \((c_1, \ldots, c_n) \leftarrow \mathbb{F}^n\), behaves as an honest \(P_1\) to compute and record \(((\beta, M[\beta]), \{(r_i, M[r_i])\}_{i \in [1, n]}, \{c_i\}_{i \in [1, n]}\) in \(Q_O\), sends \((c_1, \ldots, c_n)\) to \(A\), and computes \(u := \texttt{Unit}_F(N, \tau_1 \ldots \tau_n, \beta)\) and

\[
w := \texttt{cGGM.PuncFullEval}(\tau_1 \ldots \tau_n, \{(-1)^{r_i} \cdot (M[r_i] + c_i)\}_{i \in [1, n]}) + \texttt{Unit}_F(N, \tau_1 \ldots \tau_n, M[\beta]).
\]

Then, \(S\) sends (extend, \(N\)) and \((u, w)\) to \(F_{sVOLE}\).

- **Global-key query.** \(S\) performs the global-key query in the following cases:

  - query\(_1\): \(Z\) queries the random permutation \(\pi\) with \(x\) or its inverse \(\pi^{-1}\) with \(y\). For every \(((\beta, M[\beta]), \{(r_i, M[r_i])\}_{i \in [1, n]}, \{c_i\}_{i \in [1, n]}\) \(\in Q_O\), \(S\) does:

    1. \(S\) defines

       \[
       \{z_i\}_{i \in [1, n]} := \texttt{cGGM.OffPath}(\tau_1 \ldots \tau_n, \{(-1)^{r_i} \cdot (M[r_i] + c_i)\}_{i \in [1, n]}),
       \]

       \[
w_j := \sum_{i \in [1, j-1]} z_i, \quad \forall j \in [2, n],
\]

41
where cGGM.OffPath is a macro such that, on input a path labeled by $\alpha$ and $n$ sums \( \{K^\alpha_i\}_{i \in [1,n]} \) used in cGGM.PuncFullEval to define an $n$-level correlated GGM tree except the $n$ on-path nodes, it outputs the siblings of these on-path nodes, i.e., the $n$ off-path nodes just leaving the $\alpha$-path. This macro ensures that, for $j \in [1,n]$,

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

2. For $j \in [2,n]$, $S$ extracts $K[\beta]^j_1$ and sends $(\text{guess}, \beta^{-1} \cdot (M[\beta] - K[\beta]^j_1))$ to $\mathcal{F}_{\text{spsVOLE}}$ if $Z$ queries $\pi$ with $x$, or extracts $K[\beta]^j_2$ and $(\text{guess}, \beta^{-1} \cdot (M[\beta] - K[\beta]^j_2))$ to $\mathcal{F}_{\text{spsVOLE}}$ if $Z$ queries $\pi^{-1}$ with $y$, where

\[
K[\beta]^j_1 := \sigma^{-1}(x) + w_j, \quad K[\beta]^j_2 := \begin{cases} \sigma^{-1}(-y + z_j) + w_j, & \text{if } r_j = 0 \\ \sigma'^{-1}(-y - z_j) + w_j, & \text{if } r_j = 1 \end{cases}
\]

Note that $\sigma'(x) := \sigma(x) - x$ is a permutation since $\sigma : \mathbb{K} \rightarrow \mathbb{K}$ is an orthomorphism, and its inverse $\sigma'^{-1}$ is well-defined.

- query2: A new query-response pair $((\beta, M[\beta]), \{(r_i, M[r_i])\}_{i \in [1,n]}, \{c_i\}_{i \in [1,n]})$ is added to $Q_\pi$. $S$ sets $\{z_i\}_{i \in [1,n]}$ and $\{w_j\}_{j \in [2,n]}$ as in the query1 case. Then, for $(x,y) \in Q_\pi$ and $j \in [2,n]$, $S$ extracts both $K[\beta]^j_1$ and $K[\beta]^j_2$, and sends $(\text{guess}, \beta^{-1} \cdot (M[\beta] - K[\beta]^j_1))$ and $(\text{guess}, \beta^{-1} \cdot (M[\beta] - K[\beta]^j_2))$ to $\mathcal{F}_{\text{spsVOLE}}$.

In either case, if $S$ receives (success) from $\mathcal{F}_{\text{spsVOLE}}$ for some guess $\Delta$, $S$ will program the random permutation $\pi$ and its inverse $\pi^{-1}$ such that, for each $((\beta, M[\beta]), \{(r_i, M[r_i])\}_{i \in [1,n]}, \{c_i\}_{i \in [1,n]}) \in Q_\pi$ up to this time and its associated $K[\beta] := M[\beta] - \beta \cdot \Delta$, and $j \in [2,n]$,

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

After programming, $S$ uses the global key $\Delta$ extracted from $\mathcal{F}_{\text{spsVOLE}}$ to emulate $\mathcal{F}_{\text{VOLE}}$ and $(c_1, \ldots, c_n)$ in any future $\text{Extend}$ iteration by following the specification in $\Pi_{\text{spsVOLE-cGGM}}$.

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

B.3 Proof of Theorem 3

**Theorem 3.** Given CCR function $H : \mathbb{F}_{2^\lambda} \rightarrow \mathbb{F}_{2^\lambda}$, function $\text{Convert}_{\mathbb{K}} : \mathbb{F}_{2^\lambda} \rightarrow \mathbb{K}$ (Appendix F.1), and the pseudorandom correlated GGM tree (Figure 5) for field $\mathbb{K}$, keyed hash function $H_S(x) := H(S \circ x)$ with key $S \leftarrow \mathbb{F}_{2^\lambda}$, and function $\text{Convert}_{\mathbb{K}}$, protocol $\Pi_{\text{spsVOLE-pcGGM}}$ (Figure 6) UC-realizes functionality $\mathcal{F}_{\text{spsVOLE}}$ (Figure 2) without global-key queries against any semi-honest adversary in the $(\mathcal{F}_{\text{COT}}, \mathcal{F}_{\text{VOLE}})$-hybrid model.

**Proof.** We consider $t \geq 1$ parallel $\Pi_{\text{spsVOLE-pcGGM}}$ executions, which have access to the same subroutine $\mathcal{F}_{\text{COT}}$ and $\mathcal{F}_{\text{VOLE}}$ instances. For simplicity of exposition, we present the simulator $S$ for a
single execution, and the simulator for the $t$ parallel executions interacts with $F_{\text{spvVOLE}}$ and simply runs $S$ for each execution. $S$ internally runs the real-world adversary $A$ and relays messages between $A$ and the environment $Z$.

**Correctness analysis.** By definition, the outputs of $\text{pcGGM.PuncFullEval}$ and $\text{pcGGM.FullEval}$ are identical at every non-punctured point. Thus, for the punctured point $\alpha$, it holds that

$$w[\alpha] = \psi + M[\beta] - \sum_{j \in [0, 2^n), j \neq \alpha} w[j]$$
$$= \psi + M[\beta] - \sum_{j \in [0, 2^n), j \neq \alpha} v[j]$$
$$= \sum_{j \in [0, 2^n)} v[j] - K[\beta] + M[\beta] - \sum_{j \in [0, 2^n), j \neq \alpha} v[j]$$
$$= v[\alpha] + \beta \cdot \Gamma$$

as required by a single-point sVOLE correlation.

**Corrupted $P_0$.** In **Initialize** phase:

1. Upon receiving the first (init) from $A$ to $F_{\text{COT}}$, $S$ waits for $A$ to choose $\Delta$ and records this value.
2. Upon receiving the first (init) from $A$ to $F_{\text{spvVOLE}}$, $S$ waits for $A$ to choose $\Gamma$ and records this value. Then, $S$ sends (init) and $\Gamma$ to $F_{\text{spvVOLE}}$.

In **Extend** phase:

3. Upon receiving (extend, $n$) from $A$ to $F_{\text{COT}}$, $S$ waits for $A$ to choose its output $(K[r_1], \ldots, K[r_n])$.
4. Upon receiving (extend, 1) from $A$ to $F_{\text{spvVOLE}}$, $S$ waits for $A$ to choose $K[s]$.
5. $S$ sends random $d \leftarrow \mathbb{F}$ to $A$.
6. $S$ receives $(c_1, \ldots, c_{n-1}, \mu, c_n^0, c_n^1, \psi)$ from $A$, sets $k := K[r_1] \oplus c_1$, and computes $(v, \ldots) := \text{pcGGM.FullEval}(\Delta, k)$. Then, $S$ sends (extend, $N$) and $v$ to $F_{\text{spvVOLE}}$.

The simulation is perfect. Without loss of generality, assume a deterministic environment $Z$. The view of $A$ consists of $(\Delta, \Gamma)$ in **Initialize** phase, and $(K[r_1], \ldots, K[r_n]), K[s]$, and $d$ in the current **Extend** phase. $d$ is uniform in both the real execution (due to the uniform mask $s$) and the ideal execution. The other transcripts, which are essentially chosen by $Z$ itself, are identically distributed in the two executions since (i) they depend on the previous view of $A$ (which was forwarded to $Z$), and (ii) this previous view is identically distributed in the two executions (this holds for the initial iteration and can be regarded as the induction assumption for subsequent iterations).

The output $(u, w)$ is identically distributed in the two executions conditioned on some literal values of $(\Delta, \Gamma), (K[r_1], \ldots, K[r_n]), K[s]$, and $d$. In the real execution, the index $\alpha$ of the only nonzero entry of $u$ is uniform since the COT choice bits are uniform in the presence of the corrupted $P_0$. Plus the identical distribution of $\beta$ in the two executions, $u$ is identically distributed. Note that, for some fixed $u$ with the nonzero entry $u[\alpha] = \beta$ in the real execution,

$$K_{\alpha}^1 := M[r_1] \oplus c_1 = (K[r_1] \oplus r_1 \cdot \Delta) \oplus c_1 = k \oplus r_1 \cdot \Delta = K_0^1 \oplus \overline{c}_1 \cdot \Delta,$$
$$K_{\alpha}^2 := M[r_1] \oplus c_i = (K[r_1] \oplus r_1 \cdot \Delta) \oplus (K[r_1] \oplus K_0^i) = K_0^i \oplus \overline{c}_i \cdot \Delta, \quad \forall i \in [2, n - 1],$$
$$K_{\alpha}^n := c_n^r - \text{Convert}_K(H_S(\mu \oplus M[r_n]))$$
$$= \text{Convert}_K(H_S(\mu \oplus K[r_n] \oplus r_n \cdot \Delta)) - \text{Convert}_K(H_S(\mu \oplus M[r_n])) + K_{\alpha}^n,$$
The transcript \((c_1, \ldots, c_{n-1}, \mu, c_n^0, c_n^1, \psi)\) is identical in the two executions conditioned on the same literal value of the view of \(A\). Additionally conditioned on \(u\), running \(\text{pcGGM}.\text{PuncFullEval}\) on the above \(\{K_{i}^{j}, i \in [1,n]\}\) and \(\psi + M[\beta]\) results in the ideal consistency \(w = v + u \cdot \Gamma \in \mathbb{K}\) in the real execution.

**Corrupted \(P_1\).** In **Initialize** phase:

1. Upon receiving the first (\text{init}) from \(A\) to \(\mathcal{F}_{\text{COT}}\), \(S\) does nothing.

2. Upon receiving the first (\text{init}) from \(A\) to \(\mathcal{F}_{\text{sVOLE}}\), \(S\) sends (\text{init}) to \(\mathcal{F}_{\text{spVOLE}}\).

In **Extend** phase:

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

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

5. \(S\) receives \(d\) from \(A\) and extracts \(\beta := s - d\).

6. \(S\) sends random \((c_1, \ldots, c_{n-1}, \mu, c_n^0, c_n^1, \psi) \leftarrow \mathbb{F}_2^n \times \mathbb{K}^3\) to \(A\) and behaves as an honest \(P_1\) to compute \(u := \text{Unit}_\mathcal{T}(N, \tau_1 \ldots \tau_n, \beta)\) and

\[
w := \text{pcGGM}.\text{PuncFullEval}(\tau_1 \ldots \tau_n, \{K_{i}^{j}, i \in [1,n]\}, \psi + M[\beta])\]

Then, \(S\) sends (\text{extend}, \(N\)) and \((u, w)\) to \(\mathcal{F}_{\text{spVOLE}}\).

We prove that this ideal execution is computationally indistinguishable from the real one via the following hybrid argument.

- **Hybrid\(_0\).** This is the real execution.

- **Hybrid\(_1\).** This hybrid is identical to the previous one, except that \(S\) emulates \(\mathcal{F}_{\text{COT}}\) and \(\mathcal{F}_{\text{sVOLE}}\), uses the *honestly* generated \((c_1, \ldots, c_{n-1}, \mu, c_n^0, c_n^1, \psi)\) and the other transcripts held by the corrupted \(P_1\) to compute its output \((u, w)\), and sends (\text{extend}, \(N\)) and \((u, w)\) to \(\mathcal{F}_{\text{spVOLE}}\). This hybrid is identical to the previous one since \(\Pi_{\text{spVOLE-\text{pcGGM}}}\) is correct, and \(S\) just learns the corrupted party’s output consistent with its view and inputs this output to \(\mathcal{F}_{\text{spVOLE}}\) to define the honest party’s output accordingly.

- **Hybrid\(_2\).** This hybrid is identical to the previous one, except that \(S\) changes the way to compute \((c_1, \ldots, c_{n-1}, \mu, c_n^0, c_n^1, \psi)\). More specifically, \(S\) uses the real-world oracle \(\text{Real}_{H_{\Delta}}(\cdot)\) in Lemma 1 for some \(\Delta \leftarrow \{0, 1\}^\lambda\) as follows. First, \(S\) queries \(\text{Real}_{H_{\Delta}}(\cdot)\) with \(Q_1 : s_{\overline{\tau}_i} := \text{xor}[1 : 1] \leftarrow \{0, 1\}^\lambda\). Then, for \(i \in [2, n - 1]\), it sequentially proceeds as follow:

  - Query \(\text{Real}_{H_{\Delta}}(\cdot)\) with \(Q_{i,1} : \text{temp}_1 := \mathcal{O}(\text{xor}[1, i - 1], \overline{\tau}_i)\).
  - Query \(\text{Real}_{H_{\Delta}}(\cdot)\) with \(Q_{i,2} : \text{temp}_2 := \text{temp}_1 \oplus \text{xor}[1 : i - 1]\).
  - If \(\overline{\tau}_i = 0\), regard \(\text{temp}_1\) as \(s_{\overline{\tau}_i}^i\) and \(\text{temp}_2\) as \(\text{xor}[1 : i]\) (without new query).
  - If \(\overline{\tau}_i = 1\), regard \(\text{temp}_1\) as \(\text{xor}[1 : i]\) and \(\text{temp}_2\) as \(s_{\overline{\tau}_i}^i\) (without new query).
Finally, it queries $\text{Real}_{H_S, \Delta}()$ with the following operations:

\[ Q_{n,1} : \text{temp} := O(xor[1 : n - 1], 0), \]
\[ Q_{n,2} : \text{temp'} := xor[1 : n - 1] \oplus 1, \]
\[ Q_{n,3} : \text{temp''} := O(\text{temp'}, 0), \]
\[ Q_{n,4} : \text{rand} \leftarrow \{0, 1\}^\lambda, \]
\[ Q_{n,5} : \text{pad} := O(\text{rand}, 0). \]

If $\pi_n = 0$, $B$ defines $R_1 := \text{temp}$ and $R_2 := \text{temp''}$; otherwise, it defines $R_1$ and $R_2$ reversely. Then, $S$ defines $s^n_{\pi_n} := \text{Convert}_{K}(R_1)$ and $s^n_{\alpha_n} := \text{Convert}_{K}(R_2)$, and computes $\{K^i_{\pi_i}\}_{i \in [1,n]}$ and $K^n_{\alpha_n}$ from $\alpha := r_1 \ldots r_n$, $\{s^i_{\pi_i}\}_{i \in [1,n]}$, and $s^n_{\alpha_n}$ such that

\[ \forall i \in [1, n - 1] : K^i_{\pi_i} = s^i_{\pi_i} \oplus \text{“other } \pi_i\text{-side nodes on the } i\text{-th level defined by the off-path nodes } \{s^j_{\pi_j}\}_{j \in [1,i-1]}\text{”}, \]
\[ \forall \sigma \in \{0, 1\} : K^0_{\sigma} = s^n_{\sigma} + \text{“other } \sigma\text{-side nodes on the } n\text{-th level defined by the off-path nodes } \{s^j_{\pi_j}\}_{j \in [1,n-1]}\text{”}. \]

$S$ sets the following transcripts of the honest $P_0$:

\[ c_i := K^i_{\pi_i} \oplus M[r_i], \quad \forall i \in [1, n - 1] \]
\[ \mu := \text{rand} \oplus M[r_n], \]
\[ c^n_{\pi_n} := K^n_{\pi_n} + \text{Convert}_{K}(H_S(\text{rand})), \]
\[ c^n_{\alpha_n} := K^n_{\alpha_n} + \text{Convert}_{K}(\text{pad}), \]
\[ \psi := K^n_{\pi_n} + K^n_{\alpha_n} - K[\beta]. \]

One can check that these transcripts are equivalently defined as in the previous hybrid under the identically distributed $\Delta$, $c_1$, and $\mu$. Therefore, this hybrid is identically distributed as the previous one.

- **Hybrid$_3$.** This hybrid is identical to the previous one, except that $S$ replaces $\text{Real}_{H_S, \Delta}()$ by the ideal-world oracle $\text{Ideal}_{H_S}(\cdot)$. Lemma 1 ensures that this hybrid is computationally indistinguishable from the previous one. Note that $\{c_i\}_{i \in [1,n-1]}$, $\mu$, and $\text{pad}$ can be uniformly sampled since $\{s^i_{\pi_i}\}_{i \in [1,n-1]}$ are uniform according to $\text{Ideal}_{H_S}(\cdot)$ and serve as one-time pad for $\{K^i_{\pi_i}\}_{i \in [1,n-1]}$.

- **Hybrid$_4$.** This hybrid is identical to the previous one, except that $S$ replaces $s^n_{\pi_n}$, $s^n_{\alpha_n}$, and $\text{Convert}_{K}(\text{pad})$ by random values. This hybrid is computationally indistinguishable from the previous one due to the pseudorandom $\text{Convert}_{K}$.

- **Hybrid$_5$.** This hybrid is identical to the previous one, except that $S$ replaces $c^n_{\pi_n}$, $c^n_{\alpha_n}$, and $\psi$ by random values. This hybrid is identically distributed as the previous one since, from the construction of $K^n_{\pi_n}$ and $K^n_{\alpha_n}$, we can see that $s^n_{\pi_n}$, $\text{Convert}_{K}(\text{pad})$, and $s^n_{\alpha_n}$ essentially serve as uniform one-time pad for $c^n_{\pi_n}$, $c^n_{\alpha_n}$, and $\psi$, respectively.

It is clear that this hybrid is the ideal execution.

The above hybrid argument completes this proof. \qed
C More Results from Pseudorandom Correlated GGM

C.1 (Single-point) sVOLE with Silent Preprocessing

In the PCG framework [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 framework requires that the (online) generation phase consumes sublinear communication and produces two sublinearly short PCG seeds, which are to be locally expanded on demand.

We can adapt our pcGGM-based spsVOLE protocol $\Pi_{spsVOLE-pcGGM}$ (Figure 6) for the silent spsVOLE PCG protocol at the cost of using one additional $\mathcal{F}_{COT}$ instance. This PCG protocol outputs the two parties’ PCG seeds implicit in $\Pi_{spsVOLE-pcGGM}$ instead of the expanded shares of correlations. The sender’s PCG seed includes $(\Delta, k)$ and the global key $\Gamma$, and the receiver’s PCG seed is $(\alpha, \{K_i^\alpha\}_{i \in [1,n]}, \beta, \psi + M[\beta])$. The expansion algorithm per party is straightforward from the two full-evaluation algorithms in $\Pi_{spsVOLE-pcGGM}$.

We proceed to explain why we need the additional $\mathcal{F}_{COT}$ instance. Observe that the last-level string OT with the payload $(c_0^n, c_1^n)$ in $\Pi_{spsVOLE-pcGGM}$ is emulated by a COT correlation under the global key $\Delta$. However, $\Delta$ is included in the sender’s PCG seed so that the environment $Z$ can see this value from the sender’s output in both the real and the ideal executions of the PCG protocol. When the receiver is corrupted, the simulator cannot learn $K_\alpha^n$ from the receiver’s PCG seed in the ideal execution. Thus, the simulator can only simulate $c_\alpha^n$ with a random value, which will be detected by $Z$ checking its consistency with the sender’s PCG seed. A fix to this issue is to use another independent $\mathcal{F}_{COT}$ instance with another independent global key $\Delta'$ and use $\Delta'$ to emulate the last-level OT. Since $\Delta'$ is hidden from $Z$, the pseudorandomness of $K_\alpha^n$ follows from the correlation robustness of hash function.

By running $t > 1$ instances of this silent spsVOLE PCG protocol and using LPN encoding, one can obtain the silent sVOLE PCG protocol. We note that, different from the sVOLE PCG protocol in [BCG+19a], our protocol reuses the same global offset $\Delta$ across polynomially many PCG seeds. As implied by the security of $\Pi_{spsVOLE-pcGGM}$, reusing the same $\Delta$ only increases the advantage of the adversary by a polynomial factor and will not significantly undermine the pseudorandomness of expanded correlations.

C.2 Puncturable Pseudorandom Function

We first recall the following definition of PPRF.

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

- $k_{\text{prf}} \leftarrow \text{PRF.Gen}(1^\lambda)$. On input $1^\lambda$, output a key $k_{\text{prf}} \in \mathcal{K}$.

- $k_{\text{prf}}\{\alpha\} \leftarrow \text{PRF.Punc}(k_{\text{prf}}, \alpha)$. On input a key $k_{\text{prf}} \in \mathcal{K}$ and a punctured point $\alpha \in \mathcal{X}$, output a punctured key $k_{\text{prf}}\{\alpha\} \in \mathcal{K}_p$.

- $y \leftarrow \text{PRF.Eval}(k_{\text{prf}}, x)$. On input a key $k_{\text{prf}} \in \mathcal{K}$ and a point $x \in \mathcal{X}$, output the result $y \in \mathcal{G}$. 

46
**Parameters:** Domain size $N = 2^n$ for $n \in \mathbb{N}$. Field $\mathbb{K}$.pcGGM (c.f. Figure 5) for $n$ and $\mathbb{K}$.

PPRF.Gen$(1^\lambda)$:
1. return $k_{\text{pprf}} := (\Delta, k) \leftarrow \mathbb{F}_2^{\lambda}$.

PPRF.Punc$(k_{\text{pprf}}, \alpha)$:
1. Parse $k_{\text{pprf}} = (\Delta, k) \in \mathbb{F}_2^{\lambda}$, and $\alpha = \alpha_1 \ldots \alpha_n \in \{0, 1\}^n$.
2. Run $(v, (K_0^i)_{i \in [n, n-1]}, (K_1^n, K_1^i)) := \text{pcGGM.FullEval}(\Delta, k)$.
3. for $i \in [1, n-1]$ do $K_\alpha^i := \overline{\alpha}_i \cdot \Delta \oplus K_0^n$.
4. return $k_{\text{pprf}}(\alpha) := \{K_\alpha^i\}_{i \in [1, n]}$.

PPRF.Eval$(k_{\text{pprf}}, x)$:
1. Parse $k_{\text{pprf}} = (\Delta, k) \in \mathbb{F}_2^{\lambda}$.
2. Run $(v, (K_0^i)_{i \in [1, n-1]}, (K_1^n, K_1^i)) := \text{pcGGM.FullEval}(\Delta, k)$.
3. return $v[x]$.

PPRF.PuncEval$(\alpha, k_{\text{pprf}}(\alpha), x)$: // Output ⊥ if $x = \alpha$
1. Parse $k_{\text{pprf}}(\alpha) = \{K_\alpha^i\}_{i \in [1, n]} \in \mathbb{F}_2^{\lambda} \times \mathbb{K}$.
2. Run $w := \text{pcGGM.PuncFullEval}(\alpha, \{K_\alpha^i\}_{i \in [1, n]}, \perp)$.
3. return $w[x]$.

Figure 14: Puncturable pseudorandom function with domain $[0, N]$ and range $\mathbb{K}$.

- $\{y, \perp\} \leftarrow \text{PPRF.PuncEval}(\alpha, k_{\text{pprf}}(\alpha), x)$. On input a punctured key $k_{\text{pprf}}(\alpha) \in \mathcal{K}_P$ and a point $x \in \mathcal{X}$, output the result $y \in \mathbb{G}$ if $x \neq \alpha$; otherwise output $\perp$.

A PPRF scheme $\text{PPRF}$ is secure if the following properties hold.

- **Correctness.** For any $x, \alpha \in \mathcal{X}$, it holds that
  $$\Pr \left[ k_{\text{pprf}} \leftarrow \text{PPRF.Gen}(1^{\lambda}), k_{\text{pprf}}(\alpha) \leftarrow \text{PPRF.Punc}(k_{\text{pprf}}, \alpha) : \text{PPRF.Eval}(k_{\text{pprf}}, x) = \text{PPRF.PuncEval}(\alpha, k_{\text{pprf}}(\alpha), x) \right] = 1.$$

- **Pseudorandomness.** For any PPT adversary $\mathcal{A}$, and any $\alpha \in \mathcal{X}$ chosen by $\mathcal{A}$, it holds that
  $$\left| \Pr \left[ k_{\text{pprf}} \leftarrow \text{PPRF.Gen}(1^{\lambda}), k_{\text{pprf}}(\alpha) \leftarrow \text{PPRF.Punc}(k_{\text{pprf}}, \alpha), y \leftarrow \text{PPRF.Eval}(k_{\text{pprf}}, \alpha) : \mathcal{A}(1^{\lambda}, \alpha, k_{\text{pprf}}(\alpha), y) = 1 \right] - \Pr \left[ k_{\text{pprf}} \leftarrow \text{PPRF.Gen}(1^{\lambda}), k_{\text{pprf}}(\alpha) \leftarrow \text{PPRF.Punc}(k_{\text{pprf}}, \alpha), y \leftarrow \mathcal{G}^\perp : \mathcal{A}(1^{\lambda}, \alpha, k_{\text{pprf}}(\alpha), y) = 1 \right] \right| \leq \text{negl}(\lambda).$$

In Figure 14, we present our pcGGM-based PPRF scheme, whose security is proved in Theorem 8.

**Theorem 8.** Given CCR function $H : \mathbb{F}_2^{\lambda} \rightarrow \mathbb{F}_2^{\lambda}$, function $\text{Convert}_S : \mathbb{F}_2^{\lambda} \rightarrow \mathbb{K}$ (Appendix F.1), and the pseudorandom correlated GGM tree (Figure 5) for field $\mathbb{K}$, keyed hash function $H_S(x) := H(S \oplus x)$ with key $S \leftarrow \mathbb{F}_2^{\lambda}$, and function $\text{Convert}_S$, Figure 14 defines 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 $\alpha$ chosen by the PPT adversary $\mathcal{A}$, we write $\text{Expt}_{\text{real}}(1^{\lambda}, \alpha)$ (resp., $\text{Expt}_{\text{ideal}}(1^{\lambda}, \alpha)$) for the distribution of $(k_{\text{pprf}}(\alpha), y)$ resulting from the experiment where $k_{\text{pprf}} \leftarrow \text{PPRF.Gen}(1^{\lambda})$, $k_{\text{pprf}}(\alpha) \leftarrow \ldots$
PPRF.Punc($k_{pprf}$, $\alpha$), and $y \leftarrow$ PPRF.Eval($k_{pprf}$, $\alpha$) (resp., the one where $k_{pprf} \leftarrow$ PPRF.Gen($1^\lambda$), $k_{pprf}\{\alpha\} \leftarrow$ PPRF.Punc($k_{pprf}$, $\alpha$), and $y \leftarrow \mathbb{K}$). We define a hybrid distribution Expt$_{\text{rand}}(1^\lambda, \alpha)$ resulting from the experiment where $k_{pprf}\{\alpha\} \leftarrow \mathbb{F}_{2^\lambda} \times \mathbb{K}$ and $y \leftarrow \mathbb{K}$. This theorem follows from the following two lemmas.

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

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

Proof. We define a distribution Expt$_{\text{hyb}}(1^\lambda, \alpha)$ that is identical to Expt$_{\text{rand}}(1^\lambda, \alpha)$, except that $s^\alpha_{\bar{\alpha}_n} := \text{Convert}_{\mathcal{K}}(R_1)$ and $y := \text{Convert}_{\mathcal{K}}(R_2)$ for some $R_1, R_2 \leftarrow \mathbb{F}_{2^\lambda}$.

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

$$\Pr \left[ \begin{array}{c} (k_{pprf}\{\alpha\}, y) \\ \leftarrow \text{Expt}_{\text{real}}(1^\lambda, \alpha) \end{array} : A(1^\lambda, \alpha, k_{pprf}\{\alpha\}, y) = 1 \right] - \Pr \left[ \begin{array}{c} (k_{pprf}\{\alpha\}, y) \\ \leftarrow \text{Expt}_{\text{hyb}}(1^\lambda, \alpha) \end{array} : A(1^\lambda, \alpha, k_{pprf}\{\alpha\}, y) = 1 \right] \leq \text{negl}(\lambda).$$

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

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

2. $B$ queries $O'$ with the operation $Q_1 : s^1_{\bar{\alpha}_1} := \text{xor}[1 : 1] \leftarrow \{0, 1\}^\lambda$. Then, for $i \in [2, n - 1]$, it sequentially proceeds as follows:
   - Query $O'$ with the operation $Q_{i, 1} : \text{temp}_1 := O(\text{xor}[1 : i - 1], \bar{\alpha}_i)$.
   - Query $O'$ with the operation $Q_{i, 2} : \text{temp}_2 := \text{temp}_1 \oplus \text{xor}[1 : i - 1]$.
   - If $\bar{\alpha}_i = 0$, regard $\text{temp}_1$ as $s^\lambda_{\bar{\alpha}_i}$ and $\text{temp}_2$ as $\text{xor}[1 : i]$ (without new query).
   - If $\bar{\alpha}_i = 1$, regard $\text{temp}_1$ as $\text{xor}[1 : i]$ and $\text{temp}_2$ as $s^\lambda_{\bar{\alpha}_i}$ (without new query).

Finally, it queries $O'$ with the following operations:

$$Q_{n, 1} : \text{temp} := O(\text{xor}[1 : n - 1], 0),$$
$$Q_{n, 2} : \text{temp}' := \text{xor}[1 : n - 1] \oplus 1,$$
$$Q_{n, 3} : \text{temp}'' := O(\text{temp}', 0).$$

If $\bar{\alpha}_n = 0$, $B$ defines $R_1 := \text{temp}$ and $R_2 := \text{temp}''$; otherwise, it defines $R_1$ and $R_2$ reversely.

3. $B$ defines $s^n_{\bar{\alpha}_n} := \text{Convert}_{\mathcal{K}}(R_1)$, $y := \text{Convert}_{\mathcal{K}}(R_2)$, and computes $k_{pprf}\{\alpha\} := \{K^i_{\bar{\alpha}_i}\}_{i \in [1, n]}$ from $\{s^i_{\bar{\alpha}_i}\}_{i \in [1, n]}$ such that

$$\forall i \in [1, n - 1] : K^i_{\bar{\alpha}_i} = s^i_{\bar{\alpha}_i} \oplus \text{“other } \bar{\alpha}_i\text{-side nodes on the } i\text{-th level defined by the off-path nodes } \{s^j_{\bar{\alpha}_j}\}_{j \in [1, i-1]}\text{”},$$

$$K^n_{\bar{\alpha}_n} = s^n_{\bar{\alpha}_n} \oplus \text{“other } \alpha_n\text{-side nodes on the } n\text{-th level defined by the off-path nodes } \{s^j_{\bar{\alpha}_j}\}_{j \in [1, n-1]}\text{”}.$$ 

Then, $B$ sends $(k_{pprf}\{\alpha\}, y)$ to $A$ and outputs whatever $A$ outputs.
On the one hand, if $B$ is given $\text{Real}_{\frac{H}{S}, \Delta} (\cdot)$ for some $\Delta \leftarrow \mathbb{F}_{2^\lambda}$, then
\[
\begin{align*}
    s_{i}^\lambda = H_S \left( \Delta \oplus \bigoplus_{j \in [1,i-1]} s_{i}^j \right) \oplus \overline{r}_{i}, & \quad \forall i \in [2, n-1], \\
    R_1 = H_S \left( \Delta \oplus \bigoplus_{j \in [1,n-1]} s_{i}^j \right), & \quad R_2 = H_S \left( \Delta \oplus \bigoplus_{j \in [1,n-1]} s_{i}^j + \alpha_n \right).
\end{align*}
\]

The resulting $(k_{\text{pprf}} (\alpha), y)$ is identically distributed as that from $\text{Expt}_{\text{real}}(1^\lambda, \alpha)$. On the other hand, if $B$ is given $\text{Ideal}_{\frac{H}{S}} (\cdot)$, then it is clear that $\{s_{i}^\lambda\}_{i \in [1,n-1]}$ and $(R_1, R_2)$ are uniformly sampled. The resulting $(k_{\text{pprf}} (\alpha), y)$ is identically distributed as that from $\text{Expt}_{\text{hyb}}(1^\lambda, \alpha)$. Hence, it follows from the contradiction assumption that $B$ breaks Lemma 1 for the CCR hash function $H$ and $T = \{1\}$.

Second, it is easy to see that, due to the pseudorandomness of $\text{Convert}_K$,
\[
\begin{align*}
    \Pr \left[ (k_{\text{pprf}} (\alpha), y) \leftarrow \text{Expt}_{\text{hyb}}(1^\lambda, \alpha) : A(1^\lambda, \alpha, k_{\text{pprf}} (\alpha), y) = 1 \right] \\
    - \Pr \left[ (k_{\text{pprf}} (\alpha), y) \leftarrow \text{Expt}_{\text{rand}}(1^\lambda, \alpha) : A(1^\lambda, \alpha, k_{\text{pprf}} (\alpha), y) = 1 \right] \leq \text{negl}(\lambda). \tag{17}
\end{align*}
\]

Using (16) and (17), one can see that this lemma holds. \hfill \Box

**Lemma 5.** For any PPT adversary $A$, and any $\alpha \in \mathcal{X}$ chosen by $A$,
\[
\begin{align*}
    \Pr \left[ (k_{\text{pprf}} (\alpha), y) \leftarrow \text{Expt}_{\text{rand}}(1^\lambda, \alpha) : A(1^\lambda, \alpha, k_{\text{pprf}} (\alpha), y) = 1 \right] \\
    - \Pr \left[ (k_{\text{pprf}} (\alpha), y) \leftarrow \text{Expt}_{\text{ideal}}(1^\lambda, \alpha) : A(1^\lambda, \alpha, k_{\text{pprf}} (\alpha), y) = 1 \right] \leq \text{negl}(\lambda).
\end{align*}
\]

**Proof.** We define a distribution $\text{Expt}_{\text{hyb}}(1^\lambda, \alpha)$ that is identical to $\text{Expt}_{\text{rand}}(1^\lambda, \alpha)$, except that $s_{\pi n}^\alpha := \text{Convert}_K(R_1)$ for some $R_1 \leftarrow \mathbb{F}_{2^\lambda}$.

First, we prove that, for any PPT $A$ and any $\alpha \in \{0,1\}^n$ chosen by $A$,
\[
\begin{align*}
    \Pr \left[ (k_{\text{pprf}} (\alpha), y) \leftarrow \text{Expt}_{\text{ideal}}(1^\lambda, \alpha) : A(1^\lambda, \alpha, k_{\text{pprf}} (\alpha), y) = 1 \right] \\
    - \Pr \left[ (k_{\text{pprf}} (\alpha), y) \leftarrow \text{Expt}_{\text{hyb}}(1^\lambda, \alpha) : A(1^\lambda, \alpha, k_{\text{pprf}} (\alpha), y) = 1 \right] \leq \text{negl}(\lambda). \tag{18}
\end{align*}
\]

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

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

2. $B$ queries $O'$ with the operation $Q_1 : s_{\pi n}^\lambda := \text{xor}[1 : 1] \leftarrow \{0,1\}^\lambda$. Then, for $i \in [2, n-1]$, it sequentially proceeds as follows:
   - Query $O'$ with the operation $Q_{i,1} : \text{temp}_1 := O(\text{xor}[1 : i-1], \overline{r}_i)$.
   - Query $O'$ with the operation $Q_{i,2} : \text{temp}_2 := \text{temp}_1 \oplus \text{xor}[1 : i-1]$.
   - If $\overline{r}_i = 0$, regard $\text{temp}_1$ as $s_{\pi n}^\lambda$ and $\text{temp}_2$ as $\text{xor}[1 : i]$ (without new query).
• If \( \overline{\alpha}_i = 1 \), regard temp\(_1\) as xor\([1 : i]\) and temp\(_2\) as \( s_{\overline{\alpha}_i}^i \) (without new query).

Finally, it proceeds as follows:

• If \( \overline{\alpha}_n = 0 \), query \( \mathcal{O}' \) with the operation \( Q_{n,1} : R_1 := \mathcal{O} (\text{xor}[1 : n - 1], 0) \).
• If \( \overline{\alpha}_n = 1 \), query \( \mathcal{O}' \) with two operations:
  \[
  Q_{n,1} : \text{temp} := \text{xor}[1 : n - 1] \oplus 1, \\
  Q_{n,2} : R_1 := \mathcal{O} (\text{temp}, 0).
  \]

3. \( \mathcal{B} \) defines \( s_{\overline{\alpha}_n}^n := \text{Convert}_{\mathbb{K}} (R_1) \), \( y \leftarrow \mathbb{K} \), and computes \( k_{\text{pprf}} := \{ K_{\overline{\alpha}_i}^i \}_{i \in [1, n]} \) from \( \{ s_{\overline{\alpha}_i}^i \}_{i \in [1, n]} \) such that

\[
\forall i \in [1, n - 1] : K_{\overline{\alpha}_i}^i = s_{\overline{\alpha}_i}^i \oplus \text{“other } \overline{\alpha}_i\text{-side nodes on the } i\text{-th level defined} \\
\text{by the off-path nodes } \{ s_{\overline{\alpha}_j}^j \}_{j \in [1, i-1]} \text{”}, \\
K_{\overline{\alpha}_n}^n = s_{\overline{\alpha}_n}^n \oplus \text{“other } \overline{\alpha}_n\text{-side nodes on the } n\text{-th level defined} \\
\text{by the off-path nodes } \{ s_{\overline{\alpha}_j}^j \}_{j \in [1, n-1]} \text{”}.
\]

Then, \( \mathcal{B} \) sends \( (k_{\text{pprf}}(\alpha), y) \) to \( \mathcal{A} \) and outputs whatever \( \mathcal{A} \) outputs.

On the one hand, if \( \mathcal{B} \) is given \( \text{Real}_{\mathbb{H}_S, \Delta}(\cdot) \) for some \( \Delta \leftarrow \mathbb{F}_{2^\lambda} \), then the resulting \( (k_{\text{pprf}}(\alpha), y) \) is identically distributed as that from \( \text{Expt}_{\text{ideal}}(1^\lambda, \alpha) \). On the other hand, if \( \mathcal{B} \) is given \( \text{Ideal}_{\mathbb{H}_S}(\cdot) \), then \( \{ s_{\overline{\alpha}_i}^i \}_{i \in [1, n-1]} \) and \( R_1 \) are uniformly sampled. The resulting \( (k_{\text{pprf}}(\alpha), y) \) is identically distributed as that from \( \text{Expt}_{\text{hyb}}(1^\lambda, \alpha) \). Therefore, it follows from the contradiction assumption that \( \mathcal{B} \) breaks Lemma 1 for the CCR hash function \( \mathbb{H} \) and \( T = \{ 1 \} \).

Second, using the pseudorandomness of \( \text{Convert}_{\mathbb{K}} \), we have that

\[
\left| \Pr \left[ \frac{(k_{\text{pprf}}(\alpha), y)}{\text{Expt}_{\text{hyb}}(1^\lambda, \alpha)} : A(1^\lambda, \alpha, k_{\text{pprf}}(\alpha), y) = 1 \right] - \Pr \left[ \frac{(k_{\text{pprf}}(\alpha), y)}{\text{Expt}_{\text{rand}}(1^\lambda, \alpha)} : A(1^\lambda, \alpha, k_{\text{pprf}}(\alpha), y) = 1 \right] \right| \leq \text{negl}(\lambda).
\]

This lemma is immediate from (18) and (19). \( \square \)

Combining the above two lemmas, this theorem holds. \( \square \)

### D Security Proofs in Section 5

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

**Lemma 6.** Let \( \text{var} \) denote the variable with the name \( \text{var} \) in \( \text{DPF.Eval} \) to distinguish it from the variable with the same name in \( \text{DPF.Gen} \). For any \( n \in \mathbb{N} \), \( x, \alpha \in \{0, 1\}^n \), and \( i \in [0, n] \), it holds in \( \text{DPF.Eval} \) that

\[
(s_{\overline{\alpha}_n}^n \parallel t_{\overline{\alpha}_n}^i) \oplus (s_{\overline{\alpha}_i}^i \parallel t_{\overline{\alpha}_i}^i) = \begin{cases} 
\Delta & \text{, if } x_1 \ldots x_i = \alpha_1 \ldots \alpha_i, \ i < n \\
(\text{HCW} \oplus \text{HCW}^*) \parallel 1 & \text{, if } x = \alpha \\
0^\lambda & \text{, if } x_1 \ldots x_i \neq \alpha_1 \ldots \alpha_i
\end{cases}
\]

where \( \text{HCW}^* := \text{high}_0[\alpha_n] \oplus \text{high}_1[\alpha_n] \) is implicit in \( \text{DPF.Gen} \).

50
Proof. We prove Lemma 6 via induction. By construction, we can see that
\[(s_0^0 \parallel t_0^0) \oplus (s_i^0 \parallel t_i^0) = (s_0^0 \parallel t_0^0) \oplus (s_i^0 \parallel t_i^0) = \Delta,\]
which is exactly the base case \(i = 0\) of our induction (note that \(n \geq 1\)).

**Induction step.** We assume that Lemma 6 holds for \(i - 1 \geq 0\) case and consider the \(i \leq n\) case. We first give out the following claim:

**Claim 2.** For any \(i \in [1, n]\) and \(b \in \{0, 1\}\), if \(x_1 \ldots x_i = \alpha_1 \ldots \alpha_i\), then it holds that \((s_i^b \parallel t_i^b) = (s_i^0 \parallel t_i^0)\).

This claim follows from that \((s_i^b \parallel t_i^b)\) (resp., \((s_i^0 \parallel t_i^0)\)) depends on \((s_b^0 \parallel t_b^0)\), the prefix \(x_1 \ldots x_i\) (resp., \(\alpha_1 \ldots \alpha_i\)), and the same \(i\) public correction words \(CW_1, \ldots, CW_i\) in both DPF.Gen and DPF.Eval. We use this claim in two cases:

- \(i = n\). In this case, it holds that
  \[(20) = (\text{high}_0[x_n] \parallel \text{low}_0[x_n]) \oplus t_n^0 - 1 \cdot (\text{HCW} \parallel \text{LCW}[x_n])
  \oplus (\text{high}_1[x_n] \parallel \text{low}_1[x_n]) \oplus t_n^1 - 1 \cdot (\text{HCW} \parallel \text{LCW}[x_n])
  \oplus (\text{high}_0[x_n] \parallel \text{low}_0[x_n]) \oplus (\text{high}_1[x_n] \parallel \text{low}_1[x_n])
  \oplus (t_n^0 - 1 \oplus t_n^1 - 1) \cdot (\text{high}_0[\alpha_n] \parallel \text{high}_1[\alpha_n] \parallel \text{low}_0[x_n] \parallel \text{low}_1[x_n] \parallel \alpha_n \parallel \tau_n)\]

1. \(x_1 \ldots x_{n-1} = \alpha_1 \ldots \alpha_{n-1}\) and \(x_n = \alpha_n\) (i.e., \(x = \alpha\)). By the induction assumption, it holds that \((t_n^0 - 1 \oplus t_n^1 - 1) = \text{lsb}(\Delta) = 1\). Thus,
  \[(20) = (\text{HCW} \oplus \text{HCW}^*) \parallel (\alpha_n \parallel \tau_n) = (\text{HCW} \oplus \text{HCW}^*) \parallel 1,\]

since Claim 2 ensures \((s_n^0 - 1 \parallel t_n^0 - 1) = (s_b^0 - 1 \parallel t_b^0 - 1)\), which is turned into \((\text{high}_0[x_n] \parallel \text{low}_0[x_n]) = (\text{high}_0[x_n] \parallel \text{low}_0[x_n])\) by the deterministic \(H_S\).

2. \(x_1 \ldots x_{n-1} = \alpha_1 \ldots \alpha_{n-1}\) but \(x_n = \overline{\alpha}_n\) (i.e., \(x_1 \ldots x_n \neq \alpha_1 \ldots \alpha_n\)). The induction assumption also ensures \((t_n^0 - 1 \parallel t_n^1 - 1) = \text{lsb}(\Delta) = 1\). Thus,
  \[(20) = 0^{\lambda - 1} \parallel 0 = 0^\lambda.\]

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
  \[(s_n^0 - 1 \parallel t_n^0 - 1) \oplus (s_n^1 - 1 \parallel t_n^1 - 1) = 0^\lambda.\]

Since \(H_S\) is deterministic, this implies \((\text{high}_0[x_n] \parallel \text{low}_0[x_n]) = (\text{high}_1[x_n] \parallel \text{low}_1[x_n])\). Thus,
  \[(20) = 0^{\lambda - 1} \parallel 0 = 0^\lambda.\]

- \(i < n\). In this case, it holds that
  \[(20) = H_S(s_i^{i-1} \parallel t_i^{i-1}) \oplus x_i \cdot (s_i^{i-1} \parallel t_i^{i-1}) \oplus t_i^{i-1} \cdot CW_i
  \oplus H_S(s_i^{i-1} \parallel t_i^{i-1}) \oplus x_i \cdot (s_i^{i-1} \parallel t_i^{i-1}) \oplus t_i^{i-1} \cdot CW_i
  = H_S(s_i^{i-1} \parallel t_i^{i-1}) \oplus H_S(s_i^{i-1} \parallel t_i^{i-1}) \oplus x_i \cdot ((s_i^{i-1} \parallel t_i^{i-1}) \oplus (s_i^{i-1} \parallel t_i^{i-1}))
  \oplus (t_i^{i-1} \parallel t_i^{i-1}) \cdot (H_{S}(s_i^{i-1} \parallel t_i^{i-1}) \oplus H_S(s_i^{i-1} \parallel t_i^{i-1}) \parallel \overline{\alpha}_i \cdot \Delta)\]

1. \(x_1 \ldots x_{i-1} = \alpha_1 \ldots \alpha_{i-1}\) and \(x_i = \alpha_i\) (i.e., \(x_1 \ldots x_i = \alpha_1 \ldots \alpha_i\)). The induction assumption ensures \((s_i^{i-1} \parallel t_i^{i-1}) = \Delta\). By Claim 2,
  \[(20) = (x_i \oplus \overline{\alpha}_i) \cdot \Delta = \Delta.\]
2. $x_1 \ldots x_{i-1} = \alpha_1 \ldots \alpha_{i-1}$ but $x_i = \overline{\alpha}_i$ (i.e., $x_1 \ldots x_i \neq \alpha_1 \ldots \alpha_i$). The induction assumption ensures $(s_0^{-1} \parallel t_0^{-1}) \oplus (s_i^{-1} \parallel t_i^{-1}) = \Delta$. By Claim 2, 

$$\text{(20)} = 0 \cdot \Delta = 0^\lambda.$$

3. $x_1 \ldots x_{i-1} \neq \alpha_1 \ldots \alpha_{i-1}$ (i.e., $x_1 \ldots x_i \neq \alpha_1 \ldots \alpha_i$). The induction assumption implies $(s_0^{-1} \parallel t_0^{-1}) \oplus (s_i^{-1} \parallel t_i^{-1}) = 0^\lambda$. By the deterministic $H_S$, 

$$\text{(20)} = 0^\lambda.$$

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

\begin{proof}

Lemma 7. For any $n \in \mathbb{N}$, $x, \alpha \in \{0, 1\}^n$, let $p \in [0, n]$ be the largest index such that $x_1 \ldots x_p = \alpha_1 \ldots \alpha_p$. It holds in $DCF.Eval$ 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. Let $\text{var}$ be defined as in Lemma 6 for $DCF.Eval$. It holds that 

$$V_0^n + V_1^n = \sum_{i \in [1,n]} (\text{Convert}_G(v_i^0) + t_0^{-1} \cdot \text{VCW}_i) - \sum_{i \in [1,n]} (\text{Convert}_G(v_i^1) + t_1^{-1} \cdot \text{VCW}_i)$$

$$= \sum_{i \in [1,n]} (\text{Convert}_G(v_i^0) - \text{Convert}_G(v_i^1)) + \sum_{i \in [1,n]} ((t_0^{-1} - t_1^{-1}) \cdot \text{VCW}_i)$$

$$= \sum_{i \in [1,n]} (\text{Convert}_G(v_i^0) - \text{Convert}_G(v_i^1)) + \sum_{i \in [1,n]} ((t_0^{-1} - t_1^{-1}) \cdot \text{VCW}_i)$$

$$\quad + \sum_{i \in [1,n]} ((t_0^{-1} - t_1^{-1}) \cdot (t_0^{-1} - t_1^{-1}) \cdot (\text{Convert}_G(v_i^0) - \text{Convert}_G(v_i^1) + (\alpha_i - \alpha_{i-1}) \cdot \beta))$$

Consider the following two cases:

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

$$v_i^0 = H_S((s_b^{-1} \parallel t_b^{-1}) \oplus 2) = H_S((s_b^{-1} \parallel t_b^{-1}) \oplus 2) = v_b, \quad \forall b \in \{0, 1\},$$

$$t_0^{-1} - t_1^{-1} \cdot (t_0^{-1} - t_1^{-1}) = (t_0^{-1} - t_1^{-1}) \cdot (t_0^{-1} - t_1^{-1}) = 1.$$ 

Using (21) and (22), we can see that 

$$\text{(21)} = \sum_{i \in [1,n]} (\alpha_i - \alpha_{i-1}) \cdot \beta = \alpha_n \cdot \beta$$

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

$$v_i^0 = H_S((s_b^{-1} \parallel t_b^{-1}) \oplus 2) = H_S((s_b^{-1} \parallel t_b^{-1}) \oplus 2) = v_b, \quad \forall b \in \{0, 1\},$$

$$t_0^{-1} - t_1^{-1} \cdot (t_0^{-1} - t_1^{-1}) = (t_0^{-1} - t_1^{-1}) \cdot (t_0^{-1} - t_1^{-1}) = 1.$$ 

\end{proof}
Moreover, for every \( i \in [p + 2, n] \), \( x_1 \cdots x_{i-1} \neq \alpha_1 \cdots \alpha_{i-1} \) and thus
\[
\frac{(s_{i-1} \| t_{i-1}^{-1})}{v_n^i} = H_S((s_{i-1} \| t_{i-1}^{-1}) \oplus 2) = H_S((s_{i-1} \| t_{i-1}^{-1}) \oplus 2) = v_i^1
\]  
by Lemma 6. Using (21), (23), and (24), we have that
\[
(21) = \sum_{i \in [1, p+1]} (\alpha_i - \alpha_{i-1}) \cdot \beta = \alpha_{p+1} \cdot \beta.
\]
The above two cases conclude this proof. \( \square \)

D.2 Proof of Theorem 4

**Theorem 4.** Given CCR function \( H : \{0, 1\}^\lambda \rightarrow \{0, 1\}^\lambda \), function \( \text{Convert}_G : \{0, 1\}^{\lambda-1} \rightarrow \mathbb{G} \) (Appendix F.1), and the keyed hash function \( H_S(x) := H(S \oplus x) \) with key \( S \leftarrow \{0, 1\}^\lambda \), (DPF.Gen, DPF.Eval) (Figure 8) is a DPF scheme with domain \([0, N]\) and range \( \mathbb{G} \).

**Proof.** Let \( \text{var} \) be defined as per Lemma 6 for some variable \( \text{var} \) in DPF.Eval.

**Correctness.** The correctness follows from the construction where
\[
y_0 + y_1 = (\text{Convert}_G(s_0) + t_0 \cdot \text{CW}_{n+1}) - (\text{Convert}_G(s_1) + t_0 \cdot \text{CW}_{n+1})
= (\text{Convert}_G(s_0) - \text{Convert}_G(s_1)) + (t_0 - t_1) \cdot \text{CW}_{n+1}
= (\text{Convert}_G(s_0) - \text{Convert}_G(s_1)) + (t_0 - t_1) \cdot (\text{Convert}_G(s_1) - \text{Convert}_G(s_0) + \beta)
\]  
If \( x = \alpha \in \{0, 1\}^n \), Lemma 6 and Claim 2 ensures that
\[
\frac{(s_0 \| t_0)}{(s_1 \| t_1)} = \frac{(s_0 \| t_0)}{(s_1 \| t_1)} = \frac{(t_0 - t_1)}{(t_0 - t_1)} = 1.
\]
Thus, (25) = \( (\text{Convert}_G(s_0) - \text{Convert}_G(s_1)) + (\text{Convert}_G(s_0) - \text{Convert}_G(s_0) + \beta) = \beta \). Otherwise, using Lemma 6 for \( x \neq \alpha \) leads to (25) = 0 + 0 = 0.

**Security.** Consider the \( d \)-th hybrid \( \text{Hybrid}_{n, G, d} \) for \( d \in [0, 3] \) in Figure 15. We have the following lemmas.

**Lemma 8.** Let \( H : \{0, 1\}^\lambda \rightarrow \{0, 1\}^\lambda \) be a \( (T, 4n, \lambda - 1, \epsilon) \)-CCR function, let \( \chi \) be a distribution on \( \{0, 1\}^\lambda \) with min-entropy at least \( \lambda - 1 \), let \( S \leftarrow \chi \) be a public key, and let \( H_S(x) := H(S \oplus x) \) for \( x \in \{0, 1\}^\lambda \). There exists such a polynomial \( \text{poly}() \) that, for any \( b \in \{0, 1\} \), \( (\alpha, \beta) \in \{0, 1\}^n \times \mathbb{G} \), and any PPT adversary \( A \) running in time \( T \leq T - \text{poly}(\lambda) \), it holds that
\[
\Pr \left[ k_b \leftarrow \text{Hybrid}_{n, G, 0}(1^\lambda, b, \alpha, \beta) : A(1^\lambda, k_b) = 1 \right] - \Pr \left[ k_b \leftarrow \text{Hybrid}_{n, G, 1}(1^\lambda, b, \alpha, \beta) : A(1^\lambda, k_b) = 1 \right] \leq 2\epsilon + \frac{32n^2}{2^{\lambda+1}}.
\]

**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 \( A \) that can distinguish the two hybrids with advantage more than \( \epsilon \) within time \( T' \). We construct the following adversary \( B \) that is given an oracle \( O' \in \{\text{Real}_{H_S, \Delta}, \text{Ideal}_{H_S}\} \) and can use \( A \) to break Lemma 1. Let \( T = \{1\} \).
Hybrid_{n,G,d}(1^\lambda, b, \alpha, \beta):
1. Parse \alpha = (\alpha_1, ..., \alpha_n) \in \{0, 1\}^n and \beta \in G.
2. Sample \Delta \leftarrow \{0, 1\}^{\lambda - 1} \parallel 1 and (s_b^0 \parallel t_b^0) \leftarrow \{0, 1\}^\lambda.
3. for i \in [1, n - 1] do
   4. if d \in \{1, 2, 3\} then
   5. Sample CW_i \leftarrow \{0, 1\}^\lambda
   6. else // d = 0
   7. CW_i := H_S(s_b^{i-1} \parallel t_b^{i-1}) \oplus H_S((s_b^{i-1} \parallel t_b^{i-1}) \oplus \bar{\alpha}_i) \cdot \Delta
   8. (s_b^i \parallel t_b^i) := H_S(s_b^{i-1} \parallel t_b^{i-1}) \oplus \alpha_i \cdot (s_b^{i-1} \parallel t_b^{i-1}) \oplus t_b^{i-1} \cdot CW_i
   9. if d \in \{1, 2, 3\} then
10. Sample (\text{high}_{1-b}[0] \parallel \text{low}_{1-b}[0]), (\text{high}_{1-b}[1] \parallel \text{low}_{1-b}[1]) \leftarrow \{0, 1\}^\lambda
11. else // d = 0
12. (\text{high}_{1-b}[0] \parallel \text{low}_{1-b}[0]) := H_S((s_b^{n-1} \parallel t_b^{n-1}) \oplus \Delta)
13. (\text{high}_{1-b}[1] \parallel \text{low}_{1-b}[1]) := H_S((s_b^{n-1} \parallel t_b^{n-1}) \oplus \Delta \oplus 1)
14. if d \in \{2, 3\} then
15. Sample CW_n \leftarrow \{0, 1\}^{\lambda + 1} and HCW^* \leftarrow \{0, 1\}^{\lambda - 1}
16. else // d \in \{0, 1\}
17. (\text{high}_b[0] \parallel \text{low}_b[0]) := H_S(s_b^{n-1} \parallel t_b^{n-1})
18. (\text{high}_b[1] \parallel \text{low}_b[1]) := H_S((s_b^{n-1} \parallel t_b^{n-1}) \oplus 1)
19. HCW := \text{high}_b[\alpha_n] \oplus \text{high}_{1-b}[\bar{\alpha}_n]
20. LCW[0] := \text{low}_b[0] \oplus \bar{\alpha}_n \oplus \text{low}_{1-b}[0]
21. LCW[1] := \text{low}_b[1] \oplus \alpha_n \oplus \text{low}_{1-b}[1]
22. CW_n := (HCW \parallel LCW[0] \parallel LCW[1]) and HCW^* := \text{high}_b[\alpha_n] \oplus \text{high}_{1-b}[\bar{\alpha}_n]
23. if d = 3 then
24. Sample CW_{n+1} \leftarrow G
25. else // d \in \{0, 1, 2\}
26. (s_b^n \parallel t_b^n) := (\text{high}_b[\alpha_n] \parallel \text{low}_b[\alpha_n]) \oplus t_b^{n-1} \cdot (HCW \parallel LCW[\alpha_n])
27. CW_{n+1} := (-1)^b \cdot (t_b^n - (t_b^n \oplus 1)) \cdot ((-1)^b \cdot (\text{Convert}_G(s_b^n) - \text{Convert}_G(s_b^n \oplus HCW \oplus HCW^*)) + \beta)
28. return k_b := ((s_b^n \parallel t_b^n), \{CW_i\}_{i \in [1, n+1]})

Figure 15: The d-th hybrid for the DPF security.

1. \mathcal{B} queries \mathcal{O}' with the operation \mathcal{Q}_1 : (s_b^0 \parallel t_b^0) \leftarrow \{0, 1\}^\lambda. Then, for i \in [1, n - 1], it sequentially proceeds as follows:
   • Query \mathcal{O}' with the operation \mathcal{Q}_{i,1} : \text{temp}_1 := H_S(s_b^{i-1} \parallel t_b^{i-1}).
   • Query \mathcal{O}' with the operation \mathcal{Q}_{i,2} : \text{temp}_2 := \mathcal{O}(s_b^{i-1} \parallel t_b^{i-1} \parallel \bar{\alpha}_i).
   • Query \mathcal{O}' with the operation \mathcal{Q}_{i,3} : CW_i := \text{temp}_1 \oplus \text{temp}_2.
   • If \alpha_i = 0 and t_b^{i-1} = 0, regard \text{temp}_1 as (s_b^i \parallel t_b^i) (without new query).
   • If \alpha_i = 0 and t_b^{i-1} = 1, regard \text{temp}_2 as (s_b^i \parallel t_b^i) (without new query).
   • If \alpha_i = 1 and t_b^{i-1} = 0, query \mathcal{O}' with the operation \mathcal{Q}_{i,4} : (s_b^i \parallel t_b^i) := \text{temp}_1 \oplus (s_b^{i-1} \parallel t_b^{i-1}).
   • If \alpha_i = 1 and t_b^{i-1} = 1, query \mathcal{O}' with the operation \mathcal{Q}_{i,4} : (s_b^i \parallel t_b^i) := \text{temp}_2 \oplus (s_b^{i-1} \parallel t_b^{i-1}).

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

\mathcal{Q}_{n,1} : (\text{high}_{1-b}[0] \parallel \text{low}_{1-b}[0]) := \mathcal{O}(s_b^{n-1} \parallel t_b^{n-1}, 0),
\mathcal{Q}_{n,2} : \text{temp} := (s_b^{n-1} \parallel t_b^{n-1}) \oplus 1,
\mathcal{Q}_{n,3} : (\text{high}_{1-b}[1] \parallel \text{low}_{1-b}[1]) := \mathcal{O}(\text{temp}, 0).
2. \( B \) runs \( k_b \leftarrow \text{Hybrid}_{n,G,0}(1^\lambda, b, \alpha, \beta) \) except that it uses the responses from \( \mathcal{O}' \) to instantiate the variables under the same names in \( \text{Hybrid}_{n,G,0} \), instead of evaluating these variables by itself.

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

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

Here, \( \chi \) 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 \( \Delta \) are uniform. \( \chi \) has min-entropy \( \lambda - 1 \). On the one hand, if \( B \) is given \( \text{Real}_{H_S, \Delta}() \) for some \( \Delta \leftarrow \chi \), then

\[
\forall i \in [1, n - 1]: CW_i = H_S(s_b^{i-1} \parallel t_b^{i-1}) \oplus H_S((s_b^{i-1} \parallel t_b^{i-1}) \oplus \Delta) \oplus \overline{\pi}_i \cdot \Delta,
\]

\[
(\text{high}_{1-b}[0] \parallel \text{low}_{1-b}[0]) = H_S((s_b^{n-1} \parallel t_b^{n-1}) \oplus \Delta),
\]

\[
(\text{high}_{1-b}[1] \parallel \text{low}_{1-b}[1]) = H_S((s_b^{n-1} \parallel t_b^{n-1}) \oplus \Delta \oplus 1).
\]

The key \( k_b \) in this case is identically distributed as that in \( \text{Hybrid}_{n,G,0} \). Thus,

\[
\Pr_{\Delta \leftarrow \chi} [B_{\text{Real}_{H_S,\Delta}()}(1^\lambda) = 1] = \Pr [k_b \leftarrow \text{Hybrid}_{n,G,0}(1^\lambda, b, \alpha, \beta) : A(1^\lambda, k_b) = 1].
\] \hspace{1cm} (26)

On the other hand, if \( B \) is given \( \text{Ideal}_{H_S}() \), then

\[
\forall i \in [1, n - 1]: CW_i = H_S(s_b^{i-1} \parallel t_b^{i-1}) \oplus O(s_b^{i-1} \parallel t_b^{i-1}, \overline{\pi}_i),
\]

\[
(\text{high}_{1-b}[0] \parallel \text{low}_{1-b}[0]) = O(s_b^{n-1} \parallel t_b^{n-1}, 0),
\]

\[
(\text{high}_{1-b}[1] \parallel \text{low}_{1-b}[1]) = O((s_b^{n-1} \parallel t_b^{n-1}) \oplus 1, 0)
\]

are uniform since \( O \) returns a uniform string upon every invocation. Thus, the key \( k_b \) in this case is identically distributed as that in \( \text{Hybrid}_{n,G,1} \), and

\[
\Pr_{\Delta \leftarrow \chi} [B_{\text{Ideal}_{H_S}()}(1^\lambda) = 1] = \Pr [k_b \leftarrow \text{Hybrid}_{n,G,1}(1^\lambda, b, \alpha, \beta) : A(1^\lambda, k_b) = 1].
\] \hspace{1cm} (27)

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

\[
\left| \Pr_{\Delta \leftarrow \chi} [B_{\text{Real}_{H_S,\Delta}()}(1^\lambda) = 1] - \Pr_{\Delta \leftarrow \chi} [B_{\text{Ideal}_{H_S}()}(1^\lambda) = 1] \right| > 2\epsilon + \frac{32n^2}{2^{\lambda+1}},
\]

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

\[\square\]

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

\[
\{k_b \mid k_b \leftarrow \text{Hybrid}_{n,G,1}(1^\lambda, b, \alpha, \beta)\} \equiv \{k_b \mid k_b \leftarrow \text{Hybrid}_{n,G,2}(1^\lambda, b, \alpha, \beta)\}.
\]

**Proof.** Lemma 9 follows from the fact that, in \( \text{Hybrid}_{n,G,1} \), the string

\[
(\text{high}_{1-b}[\overline{\pi}_n] \parallel \text{low}_{1-b}[0] \parallel \text{low}_{1-b}[1] \parallel \text{high}_{1-b}[\alpha_n])
\]

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

\[
(\text{high}_b[\overline{\pi}_n] \parallel (\text{low}_b[0] \oplus \overline{\pi}_n) \parallel (\text{low}_b[1] \oplus \alpha_n) \parallel \text{high}_b[\alpha_n]).
\]

Therefore, the resulting ciphertext

\[
\text{HCW} \parallel \text{LCW}[0] \parallel \text{LCW}[1] \parallel \text{HCW}^*
\]

in \( \text{Hybrid}_{n,G,1} \) is as uniform as its counterpart in \( \text{Hybrid}_{n,G,2} \).

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

\[
\begin{align*}
\Pr\left[ k_b \leftarrow \text{Hybrid}_{n,G,2}(1^\lambda, b, \alpha, \beta) : A(1^\lambda, k_b) = 1 \right] - \Pr\left[ k_b \leftarrow \text{Hybrid}_{n,G,3}(1^\lambda, b, \alpha, \beta) : A(1^\lambda, k_b) = 1 \right] \leq \epsilon_{\text{conv}}.
\end{align*}
\]

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

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

\[
\text{CW}_{n+1} := (-1)^b \cdot \left( t_b^n - (t_b^n + 1) \right) \cdot ((-1)^1 - b) \cdot \left( \text{Convert}_G(s_b^n) - r \right) + \beta.
\]

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

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

If \(r := \text{Convert}_G(s)\) for some \(s \leftarrow \{0,1\}^{\lambda-1}\), then \(k_b\) is identically distributed as that in \(\text{Hybrid}_{n,G,2}\) since the uniform \(\text{HCW}^*\) in \(\text{Hybrid}_{n,G,2}\) ensures the uniform \(s_b^n \oplus \text{HCW} \oplus \text{HCW}^*\). Thus,

\[
\begin{align*}
\Pr\left[ s \leftarrow \{0,1\}^{\lambda-1}, r := \text{Convert}_G(s) : B(1^\lambda, r) = 1 \right] = \Pr\left[ k_b \leftarrow \text{Hybrid}_{n,G,2}(1^\lambda, b, \alpha, \beta) : A(1^\lambda, k_b) = 1 \right]
\end{align*}
\]

Instead, if \(r \leftarrow G\), then the distribution of \(k_b\) is identical to that in \(\text{Hybrid}_{n,G,3}\) since \(\text{CW}_{n+1}\) in this hybrid has the same uniform distribution as \(r\). Therefore,

\[
\begin{align*}
\Pr\left[ r \leftarrow G : B(1^\lambda, r) = 1 \right] = \Pr\left[ k_b \leftarrow \text{Hybrid}_{n,G,3}(1^\lambda, b, \alpha, \beta) : A(1^\lambda, k_b) = 1 \right]
\end{align*}
\]

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

\[
\begin{align*}
\Pr\left[ s \leftarrow \{0,1\}^{\lambda-1}, r := \text{Convert}_G(s) : B(1^\lambda, r) = 1 \right] - \Pr\left[ r \leftarrow G : B(1^\lambda, r) = 1 \right] \geq \epsilon_{\text{conv}},
\end{align*}
\]

which contradicts with the \((T_{\text{conv}}, \epsilon_{\text{conv}})\)-pseudorandomness of Convert\(_G\). \(\square\)

It follows from Lemma 6 that, for any \(b \in \{0,1\}\) and \((\alpha, \beta) \in \{0,1\}^n \times G\), \(k_b \leftarrow \text{Hybrid}_{n,G,0}(1^\lambda, b, \alpha, \beta) \equiv \left\{ k_b \mid (k_0, k_1) \leftarrow \text{DPF\_Gen}(1^\lambda, (\alpha, \beta, n, G)) \right\}\). \(\text{Hybrid}_{n,G,3}(1^\lambda, b, \alpha, \beta)\) implies a valid simulator \(\text{Sim}(1^\lambda, b, \text{Leak}(f_{\alpha, \beta}^*))\) that outputs a key of \(n \cdot \lambda + \lambda + 1 + \log |G|\) random bits, where \(n\) and \(G\) are given by \(\text{Leak}(f_{\alpha, \beta}^*)\). This theorem immediately follows from Lemma 8, 9 and 10. \(\square\)
D.3 Proof of Theorem 5

Theorem 5. Given CCR function $H : \{0,1\}^\lambda \rightarrow \{0,1\}^\lambda$, function $\text{Convert}_G : \{0,1\}^\ell \rightarrow G$ (Appendix F.1) with $\ell \in \{\lambda - 1, \lambda\}$, and the keyed hash function $H_S(x) := H(S \oplus x)$ with key $S \leftarrow \{0,1\}^{\lambda}$, (DCF.Gen,DCF.Eval) (Figure 9) is a DCF scheme with domain $[0,N)$ and range $G$.

Proof. Correctness. In DCF.Eval, it holds that

$$y_0 + y_1 = (\text{DPF.Eval}(0,k'_0,x) + V_0^n) + (\text{DPF.Eval}(1,k'_1,x) + V_1^n)$$  \hspace{1cm} (30)

If $x = 0 \in \{0,1\}^n$, Lemma 7 and the correctness of DPF ensure that (30) = $(-\alpha_n \cdot \beta) + \alpha_n \cdot \beta = 0$. If $x < 0$, there exists such an index $p \in [0,n-1]$ that

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

Using Lemma 7 and the correctness of DPF, we have (30) = $0 + \alpha_{p+1} \cdot \beta = \beta$. Similarly, if $x > 0$, we can also find an index $p \in [0,n-1]$ that results in

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

and (30) = $0 + \alpha_{p+1} \cdot \beta = 0$ holds due to Lemma 7 and the correctness of DPF. 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’s using $n$ additional hybrids $\text{Hybrid}_{n,G,3+1}, \ldots, \text{Hybrid}_{n,G,3+n}$. We first consider the $d$-th hybrid $\text{Hybrid}_{n,G,d}$ for $d \in [0,3]$ in Figure 16 and prove the following lemmas.

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

$$\left| \Pr \left[ k_b \leftarrow \text{Hybrid}_{n,G,0}(1^\lambda, b, \alpha, \beta) : A(1^\lambda, k_b) = 1 \right] - \Pr \left[ k_b \leftarrow \text{Hybrid}_{n,G,1}(1^\lambda, b, \alpha, \beta) : A(1^\lambda, k_b) = 1 \right]\right| \leq 2\epsilon + \frac{144n^2}{2^{\lambda+1}}.$$

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

1. $B$ queries $O'$ with the operation $Q_1 : (s_b^0 \parallel t_b^0) \leftarrow \{0,1\}^\lambda$. Then, for $i \in [1,n-1]$, it sequentially proceeds as follows:

   - Query $O'$ with the operation $Q_{i,1} : \text{temp}_1 := H_S(s_b^{i-1} \parallel t_b^{i-1})$.
   - Query $O'$ with the operation $Q_{i,2} : \text{temp}_2 := O(s_b^{i-1} \parallel t_b^{i-1}, \alpha_i)$.
   - Query $O'$ with the operation $Q_{i,3} : \text{CW}_i := \text{temp}_1 \oplus \text{temp}_2$.
   - Query $O'$ with the operation $Q_{i,4} : \text{temp}_3 := (s_b^{i-1} \parallel t_b^{i-1}) \oplus 2$.
   - Query $O'$ with the operation $Q_{i,5} : v_{i-b}^1 := O(\text{temp}_3, 0)$.
   - If $\alpha_i = 0$ and $t_b^{i-1} = 0$, regard $\text{temp}_1$ as $(s_b^i \parallel t_b^i)$ (without new query).
Figure 16: The d-th \((d \in \{0,3\})\) hybrid for the DCF security.

- If \(\alpha_i = 0\) and \(t^{i-1}_b = 1\), regard \(\text{temp}_2\) as \((s^i_b \parallel t^i_b)\) (without new query).
- If \(\alpha_i = 1\) and \(t^{i-1}_b = 0\), query \(\mathcal{O}'\) with the operation \(Q_{t,6} : (s^i_b \parallel t^i_b) := \text{temp}_1 \oplus (s^{i-1}_b \parallel t^{i-1}_b)\).
- If \(\alpha_i = 1\) and \(t^{i-1}_b = 1\), query \(\mathcal{O}'\) with the operation \(Q_{t,6} : (s^i_b \parallel t^i_b) := \text{temp}_2 \oplus (s^{i-1}_b \parallel t^{i-1}_b)\).
Finally, it queries $O'$ with the following operations:

\[
Q_{n,1} : \text{high}_{1-b}[0] \parallel \text{low}_{1-b}[0] := \mathcal{O}(s_{b}^{n-1} \parallel t_{b}^{n-1}, 0),
\]

\[
Q_{n,2} : \text{temp} := (s_{b}^{n-1} \parallel t_{b}^{n-1}) \oplus 1,
\]

\[
Q_{n,3} : \text{high}_{1-b}[1] \parallel \text{low}_{1-b}[1] := \mathcal{O}(\text{temp}, 0),
\]

\[
Q_{n,4} : \text{temp} := (s_{b}^{n-1} \parallel t_{b}^{n-1}) \oplus 2,
\]

\[
Q_{n,5} : v_{1-b}^{n} := \mathcal{O}(\text{temp}', 0).
\]

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

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

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

Here, $\chi$ 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 $\Delta$ are uniform. $\chi$ has min-entropy $\lambda - 1$. On the one hand, if $B$ is given $\text{Real}_{H_S, \Delta}(\cdot)$ for some $\Delta \leftarrow \chi$, then

\[
\forall i \in [1, n - 1] : \text{CW}_i = H_S(s_{b}^{i-1} \parallel t_{b}^{i-1}) \oplus H_S((s_{b}^{i-1} \parallel t_{b}^{i-1}) \oplus \Delta) \oplus \overline{\alpha}_i \cdot \Delta
\]

\[
(\text{high}_{1-b}[0] \parallel \text{low}_{1-b}[0]) = H_S((s_{b}^{n-1} \parallel t_{b}^{n-1}) \oplus \Delta),
\]

\[
(\text{high}_{1-b}[1] \parallel \text{low}_{1-b}[1]) = H_S((s_{b}^{n-1} \parallel t_{b}^{n-1}) \oplus 1) + 1).
\]

\[
\forall i \in [1, n] : v_{1-b}^{i} = H_S((s_{b}^{n-1} \parallel t_{b}^{n-1}) \oplus \Delta + 2)
\]

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

\[
\Pr_{\Delta \leftarrow \chi} [B_{\text{Real}_{H_S, \Delta}}(1^\lambda) = 1] = \Pr [k_b \leftarrow \text{Hybrid}_{n,G,0}(1^\lambda, b, \alpha, \beta) : A(1^\lambda, k_b) = 1]. \tag{31}
\]

On the other hand, if $B$ is given $\text{Ideal}_{H_S}(\cdot)$, then

\[
\forall i \in [1, n - 1] : \text{CW}_i = H_S(s_{b}^{i-1} \parallel t_{b}^{i-1}) \oplus \mathcal{O}(s_{b}^{i-1} \parallel t_{b}^{i-1}, \overline{\alpha}_i),
\]

\[
(\text{high}_{1-b}[0] \parallel \text{low}_{1-b}[0]) = \mathcal{O}(s_{b}^{n-1} \parallel t_{b}^{n-1}, 0),
\]

\[
(\text{high}_{1-b}[1] \parallel \text{low}_{1-b}[1]) = \mathcal{O}((s_{b}^{n-1} \parallel t_{b}^{n-1}) + 1, 0),
\]

\[
\forall i \in [1, n] : v_{1-b}^{i} = \mathcal{O}((s_{b}^{n-1} \parallel t_{b}^{n-1}) + 2, 0),
\]

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 $\text{Hybrid}_{n,G,1}$, and

\[
\Pr_B[B_{\text{Ideal}_{H_S}}(1^\lambda) = 1] = \Pr [k_b \leftarrow \text{Hybrid}_{n,G,1}(1^\lambda, b, \alpha, \beta) : A(1^\lambda, k_b) = 1]. \tag{32}
\]

Using the contradiction assumption and (31), (32), we can see that

\[
\Pr_{\Delta \leftarrow \chi} [B_{\text{Real}_{H_S, \Delta}}(1^\lambda) = 1] - \Pr_B[B_{\text{Ideal}_{H_S}}(1^\lambda) = 1] > 2\epsilon + \frac{144n^2}{2^\lambda + 1},
\]

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

59
Lemma 12. For any \( b \in \{0, 1\} \), and \( (\alpha, \beta) \in \{0, 1\}^n \times G \), it holds that
\[
\{ k_b \mid k_b \leftarrow \text{Hybrid}_{n,G,1}(1^\lambda, b, \alpha, \beta) \} \equiv \{ k_b \mid k_b \leftarrow \text{Hybrid}_{n,G,2}(1^\lambda, b, \alpha, \beta) \}.
\]

**Proof.** We prove the following lemma.

Lemma 13. There exists such a polynomial \( \text{poly} \conv(\cdot) \) that, for any \((T_{\conv}, \epsilon_{\conv})\)-pseudorandom \( \text{Convert}_G : \{0, 1\}^{\lambda - 1} \rightarrow G \), any \( b \in \{0, 1\} \), \( (\alpha, \beta) \in \{0, 1\}^n \times G \), and any PPT adversary \( A \) running in time \( T \leq T_{\conv} - \text{poly} \conv(\lambda) \), it holds that
\[
\left\lvert \Pr \left[ k_b \leftarrow \text{Hybrid}_{n,G,1}(1^\lambda, b, \alpha, \beta) : A(1^\lambda, k_b) = 1 \right] - \Pr \left[ k_b \leftarrow \text{Hybrid}_{n,G,2}(1^\lambda, b, \alpha, \beta) : A(1^\lambda, k_b) = 1 \right] \right\rvert \leq \epsilon_{\conv}.
\]

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

1. \( B \) receives an element \( r \in G \) from the challenger.
2. \( B \) follows the steps of \( \text{Hybrid}_{n,G,d-1}(1^\lambda, b, \alpha, \beta) \) except that it sets
   \[
   \text{VCW}_{d-3} := (-1)^b \cdot (t_b^{d-4} - (t_b^{d-4} \oplus 1)) \cdot ((-1)^{1-b} \cdot (\text{Convert}_G(v_b^{d-3}) - r) + (\alpha_{d-3} - \alpha_{d-4}) \cdot \beta).
   \]
   In the end, \( B \) generates a key \( k_b \).
3. \( B \) invokes \( A(1^\lambda, k_b) \) and outputs whatever \( A \) outputs.
The runtime of $B$ is at most $T + \text{poly}_c^\prime(\lambda) \leq T_{\text{conv}}$ for some implicit $\text{poly}_c^\prime(\cdot)$.

If $r := \text{Convert}_G(s)$ for some $s \leftarrow \{0,1\}^\lambda$, then $k_b$ is identically distributed as in $\text{Hybrid}_{n,G,d-1}$ since $v_{1-b}^{d-3}$ in $\text{Hybrid}_{n,G,d-1}$ is as uniform as $s$, and they lead to the same distribution of $\text{VCW}_{d-3}$. This indicates that

$$\Pr \left[ s \leftarrow \{0,1\}^\lambda, r := \text{Convert}_G(s) : B(1^\lambda, r) = 1 \right]$$
$$= \Pr \left[ k_b \leftarrow \text{Hybrid}_{n,G,d-1}(1^\lambda, b, \alpha, \beta) : A(1^\lambda, k_b) = 1 \right]$$

(33)

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

$$\Pr \left[ r \leftarrow G : B(1^\lambda, r) = 1 \right] = \Pr \left[ k_b \leftarrow \text{Hybrid}_{n,G,d}(1^\lambda, b, \alpha, \beta) : A(1^\lambda, k_b) = 1 \right]$$

(34)

Using the contradiction assumption and (33), (34), we can see that

$$\left| \Pr \left[ s \leftarrow \{0,1\}^\lambda, r := \text{Convert}_G(s) : B(1^\lambda, r) = 1 \right] - \Pr \left[ r \leftarrow G : B(1^\lambda, r) = 1 \right] \right| > \epsilon_{\text{conv}},$$

which contradicts with the $(T_{\text{conv}}, \epsilon_{\text{conv}})$-pseudorandomness of $\text{Convert}_G$. \hfill \square

It follows from Lemma 6 that, for any $b \in \{0,1\}$ and $(\alpha, \beta) \in \{0,1\}^n \times G$, $(k_b | k_b \leftarrow \text{Hybrid}_{n,G,n}(1^\lambda, b, \alpha, \beta)) \equiv (k_b | (k_0, k_1) := \text{DCF.Gen}(1^\lambda, (\alpha, \beta, n, G)))$. $\text{Hybrid}_{n,G,n+3}(1^\lambda, b, \alpha, \beta)$ implies a valid simulator $\text{Sim}(1^\lambda, b, \text{Leak}(f_{G,b}^\infty))$ that outputs a key $k_b$ of $n \cdot \lambda + (\lambda + 1) + (n + 1) \cdot \log |G|$ uniform bits. This theorem immediately follows from Lemma 11, 12, 13 and 14. \hfill \square

D.4 Proof of Theorem 6

Theorem 6. Given CCR function $H : \{0,1\}^\lambda \rightarrow \{0,1\}^\lambda$, function $\text{Convert}_R : \{0,1\}^{\lambda-1} \rightarrow R$ (Appendix F.1), and the keyed hash function $H_S(x) := H(S \oplus x)$ with key $S \leftarrow \{0,1\}^\lambda$, protocol $\Pi_{\text{DPF}}$ (Figure 10) UC-realizes functionality $\mathcal{F}_{\text{DPF}}$ (Figure 7) against any semi-honest adversary in the $(\mathcal{F}_{\text{COT}}, \mathcal{F}_{\text{Rand}}, \mathcal{F}_{\text{OLE}})$-hybrid model. If $R = \mathbb{F}_{2^m}$ for out $\in \mathbb{N}$, protocol $\Pi_{\text{DPF}}$ never invokes functionality $\mathcal{F}_{\text{OLE}}$.

Proof. We consider $t \geq 1$ parallel $\Pi_{\text{DPF}}$ executions, which have access to the same subroutine $\mathcal{F}_{\text{COT}}$, $\mathcal{F}_{\text{Rand}}$, and $\mathcal{F}_{\text{OLE}}$ instances. For simplicity of exposition, we present the simulator $\mathcal{S}$ for a single execution, and the simulator for the $t$ parallel executions interacts with $\mathcal{F}_{\text{DPF}}$ and simply runs $\mathcal{S}$ for each execution. $\mathcal{S}$ internally runs the real-world adversary $\mathcal{A}$ and relays messages between $\mathcal{A}$ and the environment $\mathcal{Z}$.

Correctness analysis. It would be helpful to provide a correctness analysis of protocol $\Pi_{\text{DPF}}$ before this security proof. In sub-protocol $\Pi_{\text{PREP}}$, we have that

$$\Delta = \langle \Delta \rangle_0 \oplus \langle \Delta \rangle_1 := \Delta_0' \oplus \Delta_1' \oplus (0^{\lambda-1} \parallel (\text{lsb}((\Delta_0') \oplus \text{lsb}((\Delta_1') \oplus 1)))$$

s.t. $\text{lsb}(\Delta) = 1$,

$$K[\langle \alpha_i \rangle_{1-b}] := K'[r_{1-b}^i] \oplus g_{1-b}^i \cdot \langle \Delta \rangle_b$$

$$= K'[r_{1-b}^i] \oplus (\langle \alpha_i \rangle_{1-b} \oplus r_{1-b}^i) \cdot \langle \Delta \rangle_b$$

$$= K'[r_{1-b}^i] \oplus r_{1-b}^i \cdot (\Delta_0' \oplus (0^{\lambda-1} \parallel (\text{lsb}((\Delta_1') \oplus b)))) \oplus \langle \alpha_i \rangle_{1-b} \cdot \langle \Delta \rangle_b$$

$$= M'[r_{1-b}^i] \oplus r_{1-b}^i \cdot (0^{\lambda-1} \parallel (\text{lsb}((\Delta_1') \oplus b))) \oplus \langle \alpha_i \rangle_{1-b} \cdot \langle \Delta \rangle_b$$

$$:= M[\langle \alpha_i \rangle_{1-b}] \oplus \langle \alpha_i \rangle_{1-b} \cdot \langle \Delta \rangle_b, \ \forall b \in \{0,1\}$$
as required in $\Pi_{\text{DPF}}$. In each execution, $(s_0^{0,0} \parallel t_0^{0,0}) \oplus (s_1^{0,0} \parallel t_1^{0,0}) = \Delta$ regardless of $W$. It is sufficient to show that each correction word securely computed in $\Pi_{\text{DPF}}$ is identically defined as in $\text{DPF.Gen}$ so that the correctness of our DPF scheme (c.f. Section D.2) applies. By the correctness Lemma 6, for $i \in [1, n - 1]$,

\[
\text{CW}_i = \langle \text{CW}_i \rangle_0 \oplus \langle \text{CW}_i \rangle_1
\]

\[
:= (\oplus_{j \in [0,2^{i-1}]}) \text{H}_S(s_0^{i-1,j} \parallel t_0^{i-1,j}) \oplus \langle \alpha \rangle_0 \cdot (\Delta)_0 \oplus \text{K}[\langle \alpha \rangle_1] \oplus \text{M}[\langle \alpha \rangle_0]
\]

\[
\oplus (\oplus_{j \in [0,2^{i-1}]}) \text{H}_S(s_1^{i-1,j} \parallel t_1^{i-1,j}) \oplus \langle \alpha \rangle_1 \cdot (\Delta)_1 \oplus \text{K}[\langle \alpha \rangle_0] \oplus \text{M}[\langle \alpha \rangle_1]
\]

\[
= \text{H}_S(s_0^{i-1,0} \otimes \cdots \otimes \alpha \otimes \cdots \otimes 1 \parallel t_0^{i-1,0} \otimes \cdots \otimes \alpha \otimes \cdots \otimes 1) \oplus \text{H}_S(s_1^{i-1,0} \otimes \cdots \otimes \alpha \otimes \cdots \otimes 1) \oplus \alpha \cdot \Delta.
\]

For $\text{CW}_n = \langle \text{HCW} \parallel \text{LCW}[0] \parallel \text{LCW}[1] \rangle := \langle \text{CW}_n \rangle_0 \oplus \langle \text{CW}_n \rangle_1$, it holds that

\[
\text{HCW} := \langle \text{HCW} \rangle_0 \oplus \langle \text{HCW} \rangle_1
\]

\[
= \text{XH}_0[\langle \alpha \rangle_0] \oplus \text{H}'(\mu_0 \oplus \text{K}[\langle \alpha \rangle_1]) \oplus \text{H}'(\mu_1 \oplus \text{M}[\langle \alpha \rangle_0]) \oplus \langle \alpha \rangle_0 \cdot d_1
\]

\[
\oplus \text{XH}_1[\langle \alpha \rangle_1] \oplus \text{H}'(\mu_1 \oplus \text{K}[\langle \alpha \rangle_1]) \oplus \text{H}'(\mu_0 \oplus \text{M}[\langle \alpha \rangle_1]) \oplus \langle \alpha \rangle_1 \cdot d_0
\]

\[
= \text{XH}_0[\langle \alpha \rangle_0] \oplus \text{XH}_1[\langle \alpha \rangle_1]
\]

\[
\oplus \langle \alpha \rangle_0 \cdot (\text{XH}_0[0] \oplus \text{XH}_1[1]) \oplus \langle \alpha \rangle_1 \cdot (\text{XH}_0[0] \oplus \text{XH}_0[1])
\]

\[
= \text{XH}_0[\alpha] \oplus \text{XH}_1[\alpha] = \text{high}_0[\alpha] \oplus \text{high}_1[\alpha], \quad \text{(Lemma 6)}
\]

\[
\text{LCW}[0] := \text{XL}_0[0] \oplus \text{XL}_1[1] \oplus \alpha_n = \text{low}_0[0] \oplus \text{low}_1[0] + \text{low}_1[1] \oplus \alpha_n, \quad \text{(Lemma 6)}
\]

\[
\text{LCW}[1] := \text{XL}_0[1] \oplus \text{XL}_1[1] \oplus \alpha_n = \text{low}_0[1] \oplus \text{low}_1[1] \oplus \alpha_n, \quad \text{(Lemma 6)}
\]

When $\mathcal{R}$ is a binary field (so that $+/-$ is essentially $\oplus$), Lemma 6 also ensures

\[
\text{CW}_{n+1} = \langle \text{CW}_{n+1} \rangle_0^\mathcal{A} + \langle \text{CW}_{n+1} \rangle_1^\mathcal{A}
\]

\[
:= (\sum_{j \in [0,N]} \text{Convert}_\mathcal{R}(s_0^{n,j})) + \langle \beta \rangle_0^\mathcal{A} + (\sum_{j \in [0,N]} \text{Convert}_\mathcal{R}(s_1^{n,j})) + \langle \beta \rangle_1^\mathcal{A}
\]

\[
= \text{Convert}_\mathcal{R}(s_0^{n,\alpha}) + \text{Convert}_\mathcal{R}(s_1^{n,\alpha}) + \beta
\]

\[
= (t_0^{n,\alpha} - t_1^{n,\alpha}) \cdot (\text{Convert}_\mathcal{R}(s_1^{n,\alpha}) - \text{Convert}_\mathcal{R}(s_0^{n,\alpha}) + \beta).
\]

When $\mathcal{R}$ is general, the OLE-based multiplication plus Lemma 6 ensure

\[
\text{CW}_{n+1} = \langle \text{CW}_{n+1} \rangle_0^\mathcal{A} + \langle \text{CW}_{n+1} \rangle_1^\mathcal{A}
\]

\[
:= t_0 \cdot s_0 + t_0 \cdot \gamma_1 - x_0[1] \cdot \gamma_1 + z_0[0] + z_0[1]
\]

\[
+ t_1 \cdot s_1 + t_1 \cdot \gamma_1 - x_1[0] \cdot \gamma_1 + z_1[0] + z_1[1]
\]

\[
= t_0 \cdot s_0 + t_0 \cdot s_1 + t_1 \cdot \gamma_1 - x_0[1] \cdot \gamma_1
\]

\[
+ t_1 \cdot s_1 + t_1 \cdot \gamma_1 - x_1[0] \cdot \gamma_1
\]

\[
+ x_0[0] \cdot x_1[0] + x_0[1] \cdot x_1[1]
\]

\[
= t_0 \cdot s_0 + t_0 \cdot (s_1 + x_1[1]) - x_0[1] \cdot (t_1 + x_1[1])
\]

\[
+ t_1 \cdot s_1 + t_1 \cdot (s_0 + x_0[1]) - x_1[0] \cdot (t_0 + x_0[0])
\]

\[
+ x_0[0] \cdot x_1[0] + x_0[1] \cdot x_1[1]
\]

\[
= (t_0 + t_1) \cdot (s_0 + s_1)
\]

\[
= (t_0^{n,\alpha} - t_1^{n,\alpha}) \cdot (\text{Convert}_\mathcal{R}(s_1^{n,\alpha}) - \text{Convert}_\mathcal{R}(s_0^{n,\alpha}) + \beta).
\]
The above correction words are consistent with those defined in DPF.Gen.

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

**Corrupted $P_b$.** In **Initialize** phase in sub-protocol $\Pi_{\text{PREP}},$

- Upon receiving (init) from $\mathcal{A}$ to $\mathcal{F}_{\text{COT}}^b$, $S$ waits for $\mathcal{A}$ to choose $\Delta'_b$ and records this value.
- Upon receiving (init) from $\mathcal{A}$ to $\mathcal{F}_{\text{COT}}^{1-b}$, $S$ sends a uniform bit $\text{lsb}(\Delta'_{1-b}) \leftarrow \{0, 1\}$ to $\mathcal{A}$.

Then,

1.1. Upon receiving (extend, $n$) from $\mathcal{A}$ to $\mathcal{F}_{\text{COT}}^b$, $S$ waits for $\mathcal{A}$ to choose its COT transcript $(K'[r^1_{1-b}], \ldots, K'[r^n_{1-b}]).$

1. Upon receiving (extend, $n$) from $\mathcal{A}$ to $\mathcal{F}_{\text{COT}}^{1-b}$, $S$ waits for $\mathcal{A}$ to choose its COT transcript $((r^1_b, \ldots, r^n_b), (M'[r^1_b], \ldots, M'[r^n_b])).$

1.2. $S$ sends uniform $(g^1_{1-b}, \ldots, g^n_{1-b}) \leftarrow \{0, 1\}^n$ to $\mathcal{A}$ and receives $(g^1_b, \ldots, g^n_b)$ from $\mathcal{A}$. Then, $S$ behaves as an honest $P_b$ in sub-protocol $\Pi_{\text{PREP}}$ to compute $(\langle \Delta \rangle_b, \{K[(\alpha_i)_{1-b}], M[(\alpha_i)_{1-b}]\}_{i \in [1, n]}).$

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

3. For $i \in [1, n-1]$, $S$ sends uniform $\langle CW_i \rangle_{1-b} \leftarrow \{0, 1\}^\lambda$ to $\mathcal{A}$ and receives $\langle CW_i \rangle_b$ from $\mathcal{A}$.

4. $S$ sends uniform $(\mu_{1-b}, d_{1-b}) \leftarrow \{0, 1\}^\lambda \times \{0, 1\}^{\lambda-1}$ to $\mathcal{A}$ and receives $(\mu_b, d_b)$ from $\mathcal{A}$. Then, $S$ sends uniform $\langle CW_n \rangle_{1-b} \leftarrow \{0, 1\}^{\lambda+1}$ to $\mathcal{A}$ and receives $\langle CW_n \rangle_b$ from $\mathcal{A}$.

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

     $S$ sends uniform $\langle CW_{n+1} \rangle^A_{1-b} \leftarrow \mathcal{R}$ to $\mathcal{A}$ and receives $\langle CW_{n+1} \rangle^A_b$ from $\mathcal{A}$.

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

     Upon receiving (extend, 2) from $\mathcal{A}$ to $\mathcal{F}_{\text{OLE}}$, $S$ waits for $\mathcal{A}$ to choose its output $(x_b, z_b)$. Then, $S$ sends uniform $(\gamma_{1-b}, \zeta_{1-b}) \leftarrow \mathcal{R} \times \mathcal{R}$ to $\mathcal{A}$ and receives $(\gamma_b, \zeta_b)$ from $\mathcal{A}$. Finally, $S$ sends uniform $\langle CW_{n+1} \rangle^A_{1-b} \leftarrow \mathcal{R}$ to $\mathcal{A}$ and receives $\langle CW_{n+1} \rangle^A_b$ from $\mathcal{A}$.

6. $S$ behaves as an honest $P_b$ in protocol $\Pi_{\text{DPF}}$ to compute its output $\langle r \rangle^A_b$ from $(\langle \alpha \rangle_b, \langle \beta \rangle^A_b)$, $(\langle \Delta \rangle_b, \{K[(\alpha_i)_{1-b}], M[(\alpha_i)_{1-b}]\}_{i \in [1, n]}), W,$ the transcripts sent by $\mathcal{A}$, the transcripts that $S$ chooses on behalf of $P_{1-b}$, and, if any, the OLE transcripts chosen by $\mathcal{A}$. Then, $S$ sends (gen, $(\alpha)_b$, $(\beta)_b^A$) and $\langle r \rangle^A_b$ to $\mathcal{F}_{\text{DPF}}$.

We prove that this ideal execution is computationally indistinguishable from the real one via the following hybrid argument.

- **Hybrid$_0$.** This is the real execution.

- **Hybrid$_1$.** This hybrid is identical to the previous one, except that $S$ is given both the corrupted party’s input $(\langle \alpha \rangle_b, \langle \beta \rangle^A_b)$ and the honest party’s input $(\langle \alpha \rangle_{1-b}, \langle \beta \rangle^A_{1-b})$, and proceeds as follows:

1. $S$ uses the given $(\langle \alpha \rangle_{1-b}, \langle \beta \rangle^A_{1-b})$ to play the role of $P_{1-b}$ to interact with $\mathcal{A}$ and emulates the two $\mathcal{F}_{\text{COT}}$ instances, and, if any, $\mathcal{F}_{\text{OLE}}$.

63
2. \( S \) computes \((\langle \Delta \rangle_b, \{ (K_i(\alpha_i), e_i), M_i(\alpha_i) \})_{i \in [1, n]} \) from the COT transcripts chosen by \( A \) and the transcripts that \( S \) chooses on behalf of \( P_{1-b} \).

3. \( S \) behaves as an honest \( P_{b} \) to compute its view-consistent share \( (r)_b^A \) from \((\langle \alpha \rangle_b, \langle \beta \rangle_b^A), \langle \Delta \rangle_b, \{ (K_i(\alpha_i), e_i), M_i(\alpha_i) \})_{i \in [1, n]} \) \( W \), the transcripts sent by \( A \), the transcripts that \( S \) chooses on behalf of \( P_{1-b} \), and, if any, the OLE transcripts chosen by \( A \).

4. \( S \) sends \((\text{gen}, \langle \alpha \rangle_b, \langle \beta \rangle_b^A)\) and \((r)_b^A\) to \( F_{DFP} \).

This hybrid leads to the same distribution as the previous one since (i) \( \Pi_{DFP} \) is correct so that \((r)_b^A + (r)_b^1\) equals the full-domain evaluation of \( f_{\alpha, \beta}^* \) in the previous hybrid, and (ii) the only difference is that \( S \) learns the input and output of the corrupted \( P_{b} \) played by \( A \), without skewing any distribution.

Notice that \( S \) can learn \( \alpha = \langle \alpha \rangle_b \oplus \langle \alpha \rangle_{1-b} \) and \( \beta = \langle \beta \rangle_b^A + \langle \beta \rangle_{1-b}^A \) in this hybrid. Moreover, \( S \) can compute each \( \langle CW_i \rangle_b \) to be sent by \( A \) before it is asked to send \( \langle CW_i \rangle_{1-b} \) since \( S \) learns all randomness used by \( A \) to define \( \langle CW_i \rangle_b \) at this time.

- **Hybrid**. This hybrid is identical to the previous one, except that \( S \) sends a random \( lsb(\Delta_{1-b}) \), emulates \( F_{Rand} \), and changes the way to compute the transcripts of the played \( P_{1-b} \). The emulation and the transcript computation use the real-world oracle \( \text{Real}_{HS, \Delta}() \) (c.f. Lemma 1) for some \( \Delta \leftarrow \{0, 1\}^\lambda \) such that \( lsb(\Delta) = 1 \). First, \( S \) queries \( \text{Real}_{HS, \Delta}() \) with \( Q_1 : (s_b^{i-1} \parallel t_b^{i-1}) \leftarrow \{0, 1\}^\lambda \). Then, for \( i \in [1, n-1] \), it sequentially proceeds as follows:
  - Query \( \text{Real}_{HS, \Delta}(\cdot) \) with \( Q_{i,1} : \text{temp}_1 := H_S(s_b^{i-1} \parallel t_b^{i-1}) \).
  - Query \( \text{Real}_{HS, \Delta}(\cdot) \) with \( Q_{i,2} : \text{temp}_2 := O(s_b^{i-1} \parallel t_b^{i-1}, \alpha_i) \).
  - Query \( \text{Real}_{HS, \Delta}(\cdot) \) with \( Q_{i,3} : CW_i := \text{temp}_1 \oplus \text{temp}_2 \).
  - If \( \alpha_i = 0 \) and \( t_b^{i-1} = 0 \), regard \( \text{temp}_1 \) as \( (s_b^i \parallel t_b^i) \) (without new query).
  - If \( \alpha_i = 0 \) and \( t_b^{i-1} = 1 \), regard \( \text{temp}_2 \) as \( (s_b^i \parallel t_b^i) \) (without new query).
  - If \( \alpha_i = 1 \) and \( t_b^{i-1} = 0 \), query \( \text{Real}_{HS, \Delta}(\cdot) \) with \( Q_{i,4} : (s_b^i \parallel t_b^i) := \text{temp}_1 \oplus (s_b^{i-1} \parallel t_b^{i-1}) \).
  - If \( \alpha_i = 1 \) and \( t_b^{i-1} = 1 \), query \( \text{Real}_{HS, \Delta}(\cdot) \) with \( Q_{i,4} : (s_b^i \parallel t_b^i) := \text{temp}_2 \oplus (s_b^{i-1} \parallel t_b^{i-1}) \).

Finally, it queries \( \text{Real}_{HS, \Delta}(\cdot) \) with the following operations:

\[
\begin{align*}
Q_{n,1} : (\text{high}_{1-b}[0] \parallel \text{low}_{1-b}[0]) & := O(s_b^{n-1} \parallel t_b^{n-1}, 0), \\
Q_{n,2} : \text{temp} & := (s_b^{n-1} \parallel t_b^{n-1}) \oplus 1, \\
Q_{n,3} : (\text{high}_{1-b}[1] \parallel \text{low}_{1-b}[1]) & := O(\text{temp}, 0), \\
Q_{n,4} : \text{rand} & := \{0, 1\}^\lambda, \\
Q_{n,5} : \text{pad} & := O(\text{rand}, 0).
\end{align*}
\]

Given these oracle responses, \( S \) proceeds as follows:

1. \( S \) emulates \( F_{Rand} \) by setting its output \( W := \langle \Delta \rangle_b \oplus (s_b^0 \parallel t_b^0) \).

2. For \( i \in [1, n-1] \), \( S \) sets \( \langle CW_i \rangle_{1-b} := CW_i \oplus \langle CW_i \rangle_b \). (\( S \) can behave as an honest \( P_b \) to compute \( \langle CW_i \rangle_b \) before sending \( \langle CW_i \rangle_{1-b} \).)
3. $S$ uses the intermediate results in the computation as an honest $P_b$ to set

$$(\text{high}_b[0] \parallel \text{low}_b[0]) := H_S(s_b^{n-1,\alpha_1...\alpha_n-1} || t_b^{n-1,\alpha_1...\alpha_n-1}),$$
$$(\text{high}_b[1] \parallel \text{low}_b[1]) := H_S(s_b^{n-1,\alpha_1...\alpha_n-1} || t_b^{n-1,\alpha_1...\alpha_n-1} \oplus 1),$$
$$(\text{XH}_{1-b}[0] \parallel \text{XL}_{1-b}[0]) := (\text{XH}_{0}[0] \parallel \text{XL}_{b}[0]) \oplus (\text{high}_b[0] \parallel \text{low}_b[0])$$
$$\oplus (\text{high}_1-b[0] \parallel \text{low}_{1-b}[0]),$$
$$(\text{XH}_{1-b}[1] \parallel \text{XL}_{1-b}[1]) := (\text{XH}_{b}[1] \parallel \text{XL}_{b}[1]) \oplus (\text{high}_b[1] \parallel \text{low}_b[1])$$
$$\oplus (\text{high}_1-b[1] \parallel \text{low}_{1-b}[1]),$$
$$\mu_{1-b} := \text{rand} \oplus M(\langle \alpha_n \rangle_b) \oplus \langle \Delta \rangle_b,$$
$$d_{1-b} := H'(\mu_{1-b} \oplus M(\langle \alpha_n \rangle_b)) \oplus \text{hb}(\text{pad})$$
$$\oplus (\text{XH}_{1-b}[0] \parallel \text{XH}_{1-b}[1]).$$

4. $S$ follows line 9-14 in DPF.Gen (c.f. Figure 8) to set $CW_n$ and $CW_{n+1}$, and sets $\langle CW_n \rangle_{1-b} := CW_n \oplus \langle CW_n \rangle_b$ and $\langle CW_{n+1} \rangle_{1-b} := CW_{n+1} - \langle CW_{n+1} \rangle_b$ ($S$ can behave as an honest $P_b$ to compute $\langle CW_n \rangle_b$ and $\langle CW_{n+1} \rangle_b$ before sending $\langle CW_n \rangle_{1-b}$ and $\langle CW_{n+1} \rangle_{1-b}$ on behalf of $P_{1-b}$, respectively).

Note that $\text{lsb}(\Delta_1-b), W$, and $\mu_{1-b}$ are uniform in the two hybrids. The transcript $(g_1^{1-b}, \ldots, g_n^{1-b})$ remains unchanged. It is clear that $\{\langle CW_i \rangle_{1-b} \mid i \in [1, n+1]\}$ and $d_{1-b}$ are equivalently defined as their counterparts in the previous hybrid. Since $\Delta, W$, and $\mu_{1-b}$ follow the same distribution in the two hybrids, we can see that this hybrid is identically distributed as the previous one.

- **Hybrid 3.** This hybrid is identical to the previous one, except that $S$ replaces $\text{Real}_{H_S, \Delta}()$ by the ideal-world oracle $\text{Ideal}_{H_S}()$. Lemma 1 ensures that this hybrid is computationally indistinguishable from the previous one. $\text{lsb}(\Delta_1-b), W$, $\{\langle CW_i \rangle_{1-b} \mid i \in [1, n]\}$, and $(\mu_{1-b}, d_{1-b})$ can be uniformly sampled instead.

- **Hybrid 4.** This hybrid is identical to the previous one, except that $S$ sends random $(\gamma_{1-b}, \zeta_{1-b})$ (if any) and $\langle CW_{n+1} \rangle_{1-b}$. This hybrid is computationally indistinguishable from the previous one due to the uniform $x_{1-b}$ from $\mathcal{F}_{\text{OLE}}$ and the pseudorandom $CW_{n+1}$ from $\text{Convert}_R$. Now, $(\gamma_{1-b}, \zeta_{1-b})$ (if any) and $\langle CW_{n+1} \rangle_{1-b}$ can be uniformly sampled instead.

- **Hybrid 5.** This hybrid is identical to the previous one, except that $S$ no longer uses $(\langle \alpha \rangle_{1-b}, \langle \beta \rangle_{1-b})$ and sends a random $(g_1^{1-b}, \ldots, g_n^{1-b})$. This hybrid is identically distributed as the previous one since the one-time pad $(r_1^{1-b}, \ldots, r_n^{1-b})$ is uniform.

In this hybrid, all transcripts are independent of $(\langle \alpha \rangle_{1-b}, \langle \beta \rangle_{1-b})$, the input of the honest $P_{1-b}$. It is clear that this hybrid is the ideal execution.

The above hybrid argument completes this proof. □

### D.5 Proof of Theorem 7

**Theorem 7.** Given CCR function $H : \{0, 1\}^\lambda \rightarrow \{0, 1\}^\lambda$, function $\text{Convert}_R : \{0, 1\}^\ell \rightarrow \mathcal{R}$ for $\ell \in \{\lambda - 1, \lambda\}$ (Appendix F.1), and the keyed hash function $H_S(x) := H(S \oplus x)$ with key $S \leftarrow \{0, 1\}^\lambda$, protocol $\Pi_{DCF}$ (Figure 12) UC-realizes functionality $\mathcal{F}_{DCF}$ (Figure 7) against any semi-honest adversary in the $(\mathcal{F}_{\text{COT}}, \mathcal{F}_{\text{Rand}}, \mathcal{F}_{\text{OLE}})$-hybrid model. If $\mathcal{R} = \mathbb{F}_{2^m}$ for $m \in \mathbb{N}$, protocol $\Pi_{DCF}$ never invokes functionality $\mathcal{F}_{\text{OLE}}$.  

65
**Proof.** We consider \( t \geq 1 \) parallel \( \Pi_{DCF} \) executions, which have access to the same subroutine \( F_{COT} \), \( F_{Rand} \), and \( F_{OLE} \) instances. For simplicity of exposition, we present the simulator \( S \) for a single execution, and the simulator for the \( t \) parallel executions interacts with \( F_{DCF} \) and simply runs \( S \) for each execution. \( S \) internally runs the real-world adversary \( A \) and relays messages between \( A \) and the environment \( Z \).

**Correctness analysis.** The correctness of protocol \( \Pi_{DCF} \) is inherited from the correctness of \( \Pi_{DPF} \) (c.f. Section D.4), with the additional requirement that the \( n \) value correction words \( \{VCW_i\}_{i\in[1,n]} \) are also computed as in \( DCF.Gen \). These correction words are securely computed like \( CW_{n+1} \), and the correctness can be checked likewise using Lemma 6, the fact that \( v^1_{b,j} \) depends on \( (s^i_{b} \parallel t^i_{b,j}) \), and the correctly shared \( \{\alpha_i \cdot \beta\}_{i\in[1,n]} \). We omit the checking for \( \{VCW_i\}_{i\in[1,n]} \) and only check that each \( \alpha_i \cdot \beta \) is correctly shared as follows:

\[
\langle \alpha_i \cdot \beta \rangle_0^A + \langle \alpha_i \cdot \beta \rangle_1^A \\
:= \langle \alpha_i \rangle_0 \cdot \langle \beta \rangle_0^A - H^*(x_0^i \oplus K[\langle \alpha_i \rangle_0]) + H^*(x_0^i \oplus M[\langle \alpha_i \rangle_0]) + \langle \alpha_i \rangle_0 \cdot y_0^i \\
+ \langle \alpha_i \rangle_1 \cdot \langle \beta \rangle_1^A - H^*(x_1^i \oplus K[\langle \alpha_i \rangle_0]) + H^*(x_1^i \oplus M[\langle \alpha_i \rangle_0]) + \langle \alpha_i \rangle_1 \cdot y_0^i \\
= \langle \alpha_i \rangle_0 \cdot \langle \beta \rangle_0^A + \langle \alpha_i \rangle_1 \cdot \langle \beta \rangle_1^A \\
- H^*(x_0^i \oplus K[\langle \alpha_i \rangle_0]) + H^*(x_0^i \oplus M[\langle \alpha_i \rangle_0]) + \langle \alpha_i \rangle_0 \cdot y_1^i \\
- H^*(x_1^i \oplus K[\langle \alpha_i \rangle_1]) + H^*(x_1^i \oplus M[\langle \alpha_i \rangle_1]) + \langle \alpha_i \rangle_1 \cdot y_0^i \\
= \langle \alpha_i \rangle_0 \cdot \langle \beta \rangle_0^A + \langle \alpha_i \rangle_1 \cdot \langle \beta \rangle_1^A \\
+ \langle \alpha_i \rangle_0 \cdot (\langle \beta \rangle_2^A - 2 \cdot \langle \alpha_i \rangle_1 \cdot \langle \beta \rangle_1^A) + \langle \alpha_i \rangle_1 \cdot (\langle \beta \rangle_2^A - 2 \cdot \langle \alpha_i \rangle_0 \cdot \langle \beta \rangle_0^A) \\
= \langle \alpha_i \rangle_0 + \langle \alpha_i \rangle_1 - 2 \cdot \langle \alpha_i \rangle_0 \cdot \langle \alpha_i \rangle_1 \cdot (\langle \beta \rangle_2^A + \langle \beta \rangle_2^A) = \alpha_i \cdot \beta.
\]

The security proof is similar to Appendix D.4. Let \( P_b \) be corrupted.

**Corrupted \( P_b \).** In **Initialize** phase in sub-protocol \( \Pi_{PREP} \),

- Upon receiving (init) from \( A \) to \( F_{COT}^b \), \( S \) waits for \( A \) to choose \( \Delta_{l-b}^2 \) and records this value.
- Upon receiving (init) from \( A \) to \( F_{COT}^{1-b} \), \( S \) sends a uniform bit \( \text{lsb}(\Delta_{1-b}^2) \leftarrow \{0,1\} \) to \( A \).

Then,

1-1. Upon receiving (extend, \( n \)) from \( A \) to \( F_{COT}^b \), \( S \) waits for \( A \) to choose its COT transcript \( (K'[r^1_{1-b}], \ldots, K'[r^n_{1-b}]) \).

  Upon receiving (extend, \( n \)) from \( A \) to \( F_{COT}^{1-b} \), \( S \) waits for \( A \) to choose its COT transcript \( ((r^1_{1-b}, \ldots, r^n_{1-b}), (M'[r^1_{1-b}], \ldots, M'[r^n_{1-b}])) \).

1-2. \( S \) sends uniform \( (g^1_{1-b}, \ldots, g^n_{1-b}) \leftarrow \{0,1\}^n \) to \( A \) and receives \( (g^1_{1-b}, \ldots, g^n_{1-b}) \) from \( A \). Then, \( S \) behaves as an honest \( P_b \) in sub-protocol \( \Pi_{PREP} \) to compute \( (\Delta)_{b}, \{(K[\langle \alpha_i \rangle_{1-b}], M[\langle \alpha_i \rangle_{1-b}])\}_{i \in [1,n]} \).

2. Upon receiving (sample, \( \lambda \)) from \( A \) to \( F_{Rand} \), \( S \) emulates \( F_{Rand} \) by sending uniform \( W \leftarrow \{0,1\}^\lambda \) to \( A \).

3. For \( i \in [1, n - 1] \), \( S \) sends uniform \( ((CW_i)_{1-b}, x^i_{1-b}, y^i_{1-b}) \leftarrow \{0,1\}^\lambda \times \{0,1\} \times \mathcal{R} \) to \( A \) and receives \( ((CW_i)_{1-b}, x^i_{1-b}, y^i_{1-b}) \) from \( A \).

4. \( S \) sends uniform \( (\mu_{1-b}, d_{1-b}) \leftarrow \{0,1\}^\lambda \times \{0,1\}^\lambda \) to \( A \) and receives \( (\mu_b, d_b) \) from \( A \). Then, \( S \) sends uniform \( ((CW_n)_{1-b}, x^n_{1-b}, y^n_{1-b}) \leftarrow \{0,1\}^{\lambda+1} \times \{0,1\} \times \mathcal{R} \) to \( A \) and receives \( ((CW_n)_{1-b}, x^n_{1-b}, y^n_{1-b}) \) from \( A \).
5. (Binary field $\mathcal{R} = \mathbb{F}_{2^n}$, without $\mathcal{F}_{OLE}$)

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

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

Upon receiving $(\text{extend}, 2)$ from $A$ to $\mathcal{F}_{OLE}$, $S$ waits for $A$ to choose its output $(x_b, z_b)$. Then, $S$ sends uniform $(\gamma_{1-b} \in \mathcal{R} \times \mathcal{R}$ to $A$ and receives $(\gamma_b, \zeta_b)$ from $A$. Finally, $S$ sends uniform $(\text{CW}_{n+1})_{1-b}^A \leftarrow \mathcal{R}$ to $A$ and receives $(\text{CW}_{n+1})_{b}^A$ from $A$.

Moreover, in the $i$-th parallel execution, $S$ proceeds as follows: upon receiving $(\text{extend}, 2)$ from $A$ to $\mathcal{F}_{OLE}$, $S$ waits for $A$ to choose $(x_{b,i}, z_{b,i})$. Then, $S$ sends uniform $(\gamma_{1-b,i} \in \mathcal{R} \times \mathcal{R}$ to $A$ and receives $(\gamma_{b,i}, \zeta_{b,i})$ from $A$. Finally, $S$ sends uniform $(\text{VCW}_i)^A \leftarrow \mathcal{R}$ to $A$ and receives $(\text{VCW}_i)^A$ from $A$.

6. $S$ behaves as an honest $P_b$ in protocol $\Pi_{DCF}$ to compute its output $\langle r \rangle^A_b$ from $(\langle \alpha \rangle_b, \langle \beta \rangle^A_b)$, $(\langle \Delta \rangle_b, \{K \langle \alpha_i \rangle_{1-b}, M \langle \alpha_i \rangle_b\}_{i \in [1,n]})$, $W$, the transcripts sent by $A$, the transcripts that $S$ chooses on behalf of $P_{1-b}$, and, if any, the OLE transcripts chosen by $A$. Then, $S$ sends $(\text{gen}, \langle \alpha \rangle_b, \langle \beta \rangle^A_b)$ and $(r)^A_b$ to $\mathcal{F}_{DCF}$.

We prove that this ideal execution is computationally indistinguishable from the real one via the following hybrid argument.

- **Hybrid$_0$.** This is the real execution.

- **Hybrid$_1$.** This hybrid is identical to the previous one, except that $S$ is given both the corrupted party’s input $(\langle \alpha \rangle_b, \langle \beta \rangle^A_b)$ and the honest party’s input $(\langle \alpha \rangle_{1-b}, \langle \beta \rangle^A_{1-b})$, and proceeds as follows:

1. $S$ uses the given $(\langle \alpha \rangle_{1-b}, \langle \beta \rangle^A_{1-b})$ to play the role of $P_{1-b}$ to interact with $A$ and emulates the two $\mathcal{F}_{COT}$ instances, and, if any, $\mathcal{F}_{OLE}$.
2. $S$ computes $(\langle \Delta \rangle_b, \{K \langle \alpha_i \rangle_{1-b}, M \langle \alpha_i \rangle_b\}_{i \in [1,n]})$ from the COT transcripts chosen by $A$ and the transcripts that $S$ chooses on behalf of $P_{1-b}$.
3. $S$ behaves as an honest $P_b$ to compute its view-consistent share $\langle r \rangle^A_b$ from $(\langle \alpha \rangle_b, \langle \beta \rangle^A_b)$, $(\langle \Delta \rangle_b, \{K \langle \alpha_i \rangle_{1-b}, M \langle \alpha_i \rangle_b\}_{i \in [1,n]})$, $W$, the transcripts sent by $A$, the transcripts that $S$ chooses on behalf of $P_{1-b}$, and, if any, the OLE transcripts chosen by $A$.
4. $S$ sends $(\text{gen}, \langle \alpha \rangle_b, \langle \beta \rangle^A_b)$ and $(r)^A_b$ to $\mathcal{F}_{DCF}$.

This hybrid leads to the same distribution as the previous one since (i) $\Pi_{DCF}$ is correct so that $\langle r \rangle^A_0 + \langle r \rangle^A_1$ equals the full-domain evaluation of $f^\leq_{\alpha,\beta}$ in the previous hybrid, and (ii) the only difference is that $S$ learns the input and output of the corrupted $P_b$ played by $A$, without skewing any distribution.

Notice that $S$ can learn $\alpha = \langle \alpha \rangle_b \oplus \langle \alpha \rangle_{1-b}$ and $\beta = \langle \beta \rangle^A_b + \langle \beta \rangle^A_{1-b}$ in this hybrid. Moreover, $S$ can compute each $(\text{CW}_i)^b$ (resp., $\{\text{VCW}_i^A\}_{i \in [1,n]}$) to be sent by $A$ before it is asked to send $(\text{CW})_{1-b}$ (resp., $\{\text{VCW}_i^A\}_{i \in [1,n]}$) since $S$ learns all randomness used by $A$ to define $(\text{CW}_i)^b$ and $\{\text{VCW}_i^A\}_{i \in [1,n]}$ at this time.

- **Hybrid$_2$.** This hybrid is identical to the previous one, except that $S$ sends a random $\text{lsb}(\Delta'_{1-b})$, emulates $\mathcal{F}_{Rand}$, and changes the way to compute the transcripts of the played $P_{1-b}$. The emulation and the transcript computation use the real-world oracle \text{Real}_{H_S,\Delta}(\cdot)$ (c.f. Lemma 1) for some $\Delta \leftarrow \{0,1\}^λ$ such that $\text{lsb}(\Delta) = 1$. First, $S$ queries \text{Real}_{H_S,\Delta}(\cdot)$ with $Q_1 : (s^0_b \parallel t^0_b) \leftarrow \{0,1\}^λ$. Then, for $i \in [1, n-1]$, it sequentially proceeds as follows:
2. For $1 \leq i \leq n - 1$, $S$ sets $\langle CW_i \rangle_{1-b} := CW_i \oplus \langle CW_i \rangle_b$ ($S$ can behave as an honest $P_b$ to compute $\langle CW_i \rangle_b$ before sending $\langle CW_i \rangle_{1-b}$) and

$$x^i_{1-b} := \text{rand}_i \oplus M[\langle \alpha_i \rangle_b] \oplus \langle \Delta \rangle_b,$$

$$y^i_{1-b} := (-1)^{\langle \alpha_i \rangle_b} \cdot (H^*(x^i_{1-b} \oplus M[\langle \alpha_i \rangle_b]) - \text{Convert}_R(\text{pad}_i)) + \langle \beta \rangle^{A}_{1-b} - 2 \cdot \langle \alpha_i \rangle_{1-b} \cdot \langle \beta \rangle^{A}_{1-b}.$$
3. $S$ uses the intermediate results in the computation as an honest $P_b$ to set

$$(\text{high}_b[0] \parallel \text{low}_b[0]) := H_S(t_b^{n-1,\alpha_1...\alpha_n-1} \parallel t_b^{n-1,\alpha_1...\alpha_n-1}),$$

$$(\text{high}_b[1] \parallel \text{low}_b[1]) := H_S((s_b^{n-1,\alpha_1...\alpha_n-1} \parallel t_b^{n-1,\alpha_1...\alpha_n-1}) + 1),$$

$$(XH_{1-b}[0] \parallel XL_{1-b}[0]) := (XH_b[0] \parallel XL_b[0]) \oplus ((\text{high}_b[0]) \parallel \text{low}_b[0])$$

$$\oplus (\text{high}_{1-b}[0] \parallel \text{low}_{1-b}[0]),$$

$$(XH_{1-b}[1] \parallel XL_{1-b}[1]) := (XH_b[1] \parallel XL_b[1]) \oplus ((\text{high}_b[1]) \parallel \text{low}_b[1])$$

$$\oplus (\text{high}_{1-b}[1] \parallel \text{low}_{1-b}[1]),$$

$$\mu_{1-b} := \text{rand} \oplus M[(\alpha_n)_b] \oplus (\Delta)_b,$$

$$d_{1-b} := H'(\mu_{1-b} \oplus M[(\alpha_n)_b]) \oplus b \cdot \text{pad}$$

$$\oplus (XH_{1-b}[0] \oplus XH_{1-b}[1]),$$

$$x_{1-b}^n := \text{rand}_n \oplus M[(\alpha_n)_b] \oplus (\Delta)_b,$$

$$y_{1-b}^n := (-1)^{(\alpha_n)_b} \cdot (H^*(x_{1-b}^n \oplus M[(\alpha_n)_b]) - \text{Convert}_R(\text{pad}_n))$$

$$\oplus (\beta)_{1-b} - 2 \cdot (\alpha_n)_{1-b} \oplus (\beta)_{1-b},$$

$$\forall i \in [1, n]: VCW_i := (t_{i-b}^{i-1} - (t_{i-b}^{i-1} + 1)) \cdot (\text{Convert}_R(v_{i-b}^{i-1}) - \text{Convert}_R(v_{i-b}^{i-1}))$$

$$+ (\alpha_i - \alpha_{i-1}) \cdot \beta.$$

4. $S$ follows line 9-14 in $DPF.Gen$ (c.f. Figure 8) to set $CW_n$ and $CW_{n+1}$, and sets $(CW_{n+1})_{1-b} := CW_n \oplus (CW_{n+1})_{1-b} := CW_{n+2} - (CW_{n+1})_{1-b}$, and $(VCW_i)_{1-b} := VCW_i - (VCW_i)_{1-b}$ for $i \in [1, n]$ ($S$ can behave as an honest $P_b$ to compute $(CW_{n+1})_{1-b}$, $(CW_{n+1})_{1-b}$, and $(VCW_i)_{1-b}$ before sending $(CW_{n+1})_{1-b}$, $(CW_{n+1})_{1-b}$, and $(VCW_i)_{1-b}$ on behalf of the played $P_{1-b}$, respectively).

Note that $\text{lsb}(\Delta'_{1-b})$, $W'$, $\mu_{1-b}$, and $\{x_{i-b}^n\}_{i \in [1, n]}$ are uniform in both two hybrids. The transcript $\{g_{1-b}[i], \ldots, g_{n-b}[i]\}$ remains unchanged. $\{CW_i)_{1-b}\}_{i \in [1, n]}$, $\{VCW_i)_{1-b}\}_{i \in [1, n]}$, $d_{1-b}$, and $\{y_{1-b}[i]\}_{i \in [1, n]}$ are equivalently defined as their counterparts in the previous hybrid. Since $\Delta$, $W$, $\mu_{1-b}$, and $\{x_{i-b}[i]\}_{i \in [1, n]}$ follow the same distribution in the two hybrids, we can see that this hybrid is identically distributed as the previous one.

- **Hybrid$_2$.** This hybrid is identical to the previous one, except that $S$ replaces $\text{real}_{H_{1-b}}(\cdot)$ by the ideal-world oracle $\text{ideal}_{H_{1-b}}(\cdot)$. Lemma 1 ensures that this hybrid is computationally indistinguishable from the previous one. $\text{lsb}(\Delta'_{1-b})$, $W$, $\{CW_{i-b}\}_{i \in [1, n]}$, $\{x_{i-b}[i], \mu_{1-b}, d_{1-b}\}$, $\{x_{i-b}[i], \text{pad}_i\}_{i \in [1, n]}$, and $\{x_{i-b}[i]\}_{i \in [1, n]}$ can be uniformly sampled instead.

- **Hybrid$_4$.** This hybrid is identical to the previous one, except that $S$ sends random $(\gamma_{1-b}, \zeta_{1-b})$ (if any), $\{(\gamma_{1-b}[i], \zeta_{1-b}[i])\}_{i \in [1, n]}$ (if any), $\{CW_{i-b}\}_{i \in [1, n]}$ and $\{(y_{1-b}[i], \text{VCW}_{i-b})\}_{i \in [1, n]}$. This hybrid is computationally indistinguishable from the previous one due to the uniform $x_{1-b}$, $\{x_{1-b}[i]\}_{i \in [1, n]}$ from $F_{\text{OLE}}$ and the pseudorandom $CW_{n+1}$, $\{VCW_i\}_{i \in [1, n]}$ from $\text{Convert}_R$. Now, $\{\gamma_{1-b}, \zeta_{1-b}\}$ (if any), $\{(\gamma_{1-b}[i], \zeta_{1-b}[i])\}_{i \in [1, n]}$ (if any), $(CW_{n+1})_{1-b}$ and $\{(y_{1-b}[i], \text{VCW}_{i-b})\}_{i \in [1, n]}$ can be uniformly sampled instead.

- **Hybrid$_5$.** This hybrid is identical to the previous one, except that $S$ no longer uses $(\alpha)_{1-b}$, $(\beta)_{1-b}$ but sends a random $(g_{1-b}^1, \ldots, g_{1-b}^n)$. This hybrid is identically distributed as the previous one since the one-time pad $(r_{1-b}, \ldots, r_{n-b})$ is uniform.

In this hybrid, all transcripts are independent of $(\alpha)_{1-b}$, $(\beta)_{1-b}$, the input of the honest $P_{1-b}$. It is clear that this hybrid is the ideal execution.

The above hybrid argument completes this proof. □


E  FSS Key Generation from FSS Correlation Generation

In contrast to the functionality $F_{\text{FSS}}$ 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. In this appendix, we show that our two FSS correlation generation protocols, $\Pi_{\text{DPF}}$ (Figure 10) and $\Pi_{\text{DCF}}$ (Figure 12), can be adapted for their respective key generation protocols by additionally using one $F_{\text{COT}}$ instance per party.

Consider the DPF key generation protocol $\Pi_{\text{DPF} \rightarrow \text{Gen}}$. Note that the two DPF keys to be output by this protocol are implicitly computed in $\Pi_{\text{DPF}}$, i.e., each DPF key $k_b$ contains $\langle \Delta \rangle_b \oplus R$ and the $n + 1$ correction words. In other words, $\Pi_{\text{DPF} \rightarrow \text{Gen}}$ follows $\Pi_{\text{DPF}}$ but only outputs the two parties’ keys. However, $\Pi_{\text{DPF} \rightarrow \text{Gen}}$ 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 $Z$ is given both parties’ DPF keys. Now, the environment 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 $R$ output by $F_{\text{Rand}}$. 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 high bits of $o_n$ (thus, $XH_{1-b}[o_n]$).

To address this subtle issue, we can use the same method as in our single-point sVOLE PCG (c.f. Section C.1). That is, each party can invoke an independent $F_{\text{COT}}$ instance with a global key $\Delta_b^*$ and use the COT correlations from this instance to define $d_{1-b}$. Since $Z$ cannot see $\Delta_b^*$, 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 $\Pi_{\text{DCF} \rightarrow \text{Gen}}$, which follows $\Pi_{\text{DCF}}$ except that each party $P_b$ uses one additional $F_{\text{COT}}$ instance to define its intermediate transcripts $d_b$ and $\{y_b^i\}_{i \in [1,n]}$ as $\Pi_{\text{DCF} \rightarrow \text{Gen}}$. Each DCF key $k_b$ to be output by $\Pi_{\text{DCF} \rightarrow \text{Gen}}$ consists of $\langle \Delta \rangle_b \oplus R$, $\{C_W^i\}_{i \in [1,n+1]}$, and $\{VCW^i\}_{i \in [1,n]}$, which are implicitly computed in $\Pi_{\text{DCF}}$.

F  Supplementary Preliminaries

F.1  Pseudorandom Conversion

Implementation of $\text{Convert}_G$. This function maps $\ell$-bit random strings into pseudorandom elements in $G$ and can be implemented as follows. Let $\rho \in \mathbb{N}$ be the statistical security parameter. On input an $\ell$-bit random string $x$: if $(2^\ell \mod |G|)/2^\ell \leq 2^{-\rho}$, it computes the remainder $x \mod |G|$ and outputs this remainder as a group element in $G$; otherwise, it computes $G_{\text{ext}}(x) \mod |G|$ for some PRG $G_{\text{ext}} : \{0,1\}^\ell \rightarrow \{0,1\}^{\ell_0}$, where $\ell_0 := \lfloor \log |G| \rfloor + \rho$, and outputs this remainder as a group element in $G$. 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 $\lambda$-bit random string to fit this string for the following $\text{Convert}_G$. This abuse is just for the simplicity of exposition.

Definition 5 (Pseudorandomness of $\text{Convert}_G$). $\text{Convert}_G : \{0,1\}^{\lambda-1} \rightarrow G$ is $(t, \epsilon)$-pseudorandom

\[ A special case is that $|G|$ is a power of two such that $|G|$ divides $2^\ell$. \]
Case (ii): ____

Case (i): ____

Proof. Define $\text{Per Definition 5}$, where polynomial $\text{poly}$ $\text{G}$ Theorem 9. Let $\text{If, for any distinguisher } D \text{ running in time at most } t, \text{ and any finite group } G$, it holds that

$$\left| \Pr \left[ s \leftarrow \{0,1\}^{\lambda-1}, r := \text{Convert}_G(s) : D(1^\lambda, r) = 1 \right] \right| - \Pr \left[ r \leftarrow G : D(1^\lambda, r) = 1 \right] \leq \epsilon.$$  

**Theorem 9.** Let $\rho \in \mathbb{N}$ be the statistical security parameter, $G$ be an arbitrary finite group, $G_{\text{ext}} : \{0,1\}^{\lambda-1} \rightarrow \{0,1\}^{\ell_0}$ be a $(t, \epsilon)$-secure PRG where $\ell_0 := \lceil \log |G| \rceil + \rho$. There exists such a polynomial $\text{poly}(\cdot)$ that the implementation of $\text{Convert}_G$ in this appendix is $(t', \epsilon')$-pseudorandom as per Definition 5, where

$$\langle t', \epsilon' \rangle = \begin{cases} (\infty, 2^{-\rho}) & \text{if } \frac{2^{\lambda-1} \text{mod } |G|}{2^{\lambda-1}} \leq 2^{-\rho}, \\ (t - \text{poly}(\lambda), \epsilon + 2^{-\rho}) & \text{otherwise} \end{cases}.$$  

**Proof.** Define $M := 2^{\lambda-1} \text{mod } |G|$. Consider two cases regarding $G$:

- **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., $s \leftarrow [0,2^{\lambda-1})$), where $s$ can be viewed as a string in $\{0,1\}^{\lambda-1}$. The statistical distance

$$\text{SD}(X, Y) = \frac{1}{2} \sum_{s' \in \text{supp}(X) \cup \text{supp}(Y)} |\Pr[X = s'] - \Pr[Y = s']|$$

$$= \frac{1}{2} \sum_{s' \in [0,2^{\lambda-1})} \left| \Pr[X = s'] - \Pr[Y = s'] \right|$$

$$= \frac{1}{2} \left( 2^{\lambda-1} - M \right) \left( \frac{1}{2^{\lambda-1} - M} - \frac{1}{2^{\lambda-1}} \right) + M \cdot \frac{1}{2^{\lambda-1}}$$

$$= \frac{M}{2^{\lambda-1}} \leq 2^{-\rho}.$$  

Since $\text{Convert}_G$ is a deterministic function using modulo operation and a bijective mapping, we have $\text{SD}(\text{Convert}_G(X), \text{Convert}_G(Y)) = \text{SD}(X, Y) \leq 2^{-\rho}$. Observe that $r := \text{Convert}_G(Y)$ is equivalent to the uniform sampling $r \leftarrow G$. 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 $D$.

- **Case (ii):** $M/2^{\lambda-1} > 2^{-\rho}$. In this case, $\text{Convert}_G$ will invoke the PRG $G_{\text{ext}}$. We assume that, for the sake of contradiction, $\text{Convert}_G$ is not $(t', \epsilon')$-pseudorandom for some distinguisher $D$. Let $\Phi : [0,|G|] \rightarrow G$ be the bijective mapping. We can construct a PRG adversary $A$ that, given a challenge $c \in \{0,1\}^{\ell_0}$ from the PRG challenger, interprets $c \in [0,2^{\ell_0})$, sends $r := \Phi(c \text{mod } |G|)$ to $D$, and outputs whatever $D$ outputs. The runtime of $A$ is at most $t' + \text{poly}(\lambda) \leq t$ for some fixed polynomial $\text{poly}(\cdot)$.

Similar to the analysis in **Case (i)**, the following computational indistinguishability is bounded.
by the statistical distance of $c$:

$$
\left| \Pr \left[ r \leftarrow G : D(1^\lambda, r) = 1 \right] \right.
\left. - \Pr \left[ c \leftarrow [0, 2^t), r := \Phi(c \mod |G|) : D(1^\lambda, r) = 1 \right] \right|
= \Pr \left[ c \leftarrow [0, 2^t - 2^t \mod |G|), r := \Phi(c \mod |G|) : D(1^\lambda, r) = 1 \right]
\left. - \Pr \left[ c \leftarrow [0, 2^t), r := \Phi(c \mod |G|) : D(1^\lambda, r) = 1 \right] \right|
\leq \frac{2^t \mod |G|}{2^t} < \frac{|G|}{2^t} \leq 2^{-\rho}.
$$

Due to the construction of $A$, we have that

$$
\left| \Pr \left[ c \leftarrow \{0,1\}^{t^2} : A(1^\lambda, c) = 1 \right] \right.
\left. - \Pr \left[ c \leftarrow [0, 2^t), r := \Phi(c \mod |G|) : D(1^\lambda, r) = 1 \right] \right|
= \Pr \left[ s \leftarrow \{0,1\}^{\lambda-1}, c := G_{ext}(s) : A(1^\lambda, c) = 1 \right]
\left. - \Pr \left[ s \leftarrow \{0,1\}^{\lambda-1}, r := Convert_G(s) : D(1^\lambda, r) = 1 \right] \right|
\leq 2^{t^2} \mod |G| < \frac{|G|}{2^t} \leq 2^{-\rho}.
$$

Using the contradiction assumption and (35), (36), we can see

$$
\left| \Pr \left[ s \leftarrow \{0,1\}^{\lambda-1}, c := G_{ext}(s) : A(1^\lambda, c) = 1 \right] \right.
\left. - \Pr \left[ c \leftarrow \{0,1\}^{t^2} : A(1^\lambda, c) = 1 \right] \right| > \epsilon - 2^\rho = \epsilon,
$$

contradicting the $(t, \epsilon)$-pseudorandomness of $G_{ext}$.

This theorem follows from the above two cases. □

### F.2 Coin-tossing

In Figure 17, we present the functionality $F_{Rand}$ 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 $F_{COT}$-hybrid model. The high-level idea is that a COT correlation $M[r_{1-b}] = K[r_{1-b}] \oplus r_{1-b} \cdot \Delta_b$, where $P_b$ has $(\Delta_b, K[r_{1-b}]) \in \{0,1\}^\lambda \times \{0,1\}^\lambda$ and $P_{1-b}$ has $(r_{1-b}, M[r_{1-b}]) \in \{0,1\} \times \{0,1\}^\lambda$, is an information-theoretic message authentication code (IT-MAC) [NNOB12, DPSZ12]. From this perspective, $r_{1-b}$ is a uniform bit authenticated under the global key $\Delta_b$ and the one-time key $K[r_{1-b}]$, and the resulting MAC is $M[r_{1-b}]$. This authenticated bit is then sent to $P_{1-b}$, who checks $M[r_{1-b}] = K[r_{1-b}] \oplus r_{1-b} \cdot \Delta_b$. By each party $P_b$ opening a COT correlation for its $r_b$, the two parties can flip a uniform coin $r := r_0 \oplus r_1$. To produce a random $\lambda$-bit string $R$, $\kappa$ COT correlations are required for each party.

The above opening can be batched using the technique [DNNR17] in the RPM. In this batched opening, each $P_b$ uses a compression function $f$ and sends $((r_{b,1}, \ldots, r_{b,\lambda}), f(M[r_{b,1}], \ldots, M[r_{b,\lambda}]))$ to $P_{1-b}$, who uses these values to check $f(M[r_{b,1}], \ldots, M[r_{b,\lambda}]) = f(K[r_{b,1}] \oplus r_{b,1} \cdot \Delta_{1-b}, \ldots, K[r_{b,\lambda}] \oplus r_{b,\lambda} \cdot \Delta_{1-b})$. This compression function $f$ can be implemented in the way that $f(x_1, \ldots, x_n) := \oplus_{i \in [1,n]} H(x_i)$, for some correlation-robust hash function $H : \{0,1\}^\lambda \rightarrow \{0,1\}^\lambda$, which can be instantiated in the RPM [GKWy20]. The resulting coin-tossing protocol is one-round in the $F_{COT}$-hybrid model, and each party sends $2\lambda$ bits.

72
### Functionality $F_{\text{Rand}}$

**Sample:** Upon receiving $(\text{sample}, \ell)$ from $P_0$ and $P_1$, sample $R \leftarrow \{0, 1\}^\ell$ and send $R$ to $P_0$ and $P_1$.

Figure 17: Functionality for coin-tossing.

### Functionality $F_{\text{OLE}}$

**Parameters:** Ring $\mathcal{R}$.

**Extend:** Upon receiving $(\text{extend}, N)$ from $P_0$ and $P_1$:

1. If both parties are honest, sample $(x_0, z_0), (x_1, z_1) \leftarrow \mathcal{R}^N \times \mathcal{R}^N$ such that $z_0 + z_1 = x_0 \odot x_1$; otherwise, receive $(x_b, z_b) \in \mathcal{R}^N \times \mathcal{R}^N$ from the adversary, sample $x_{1-b} \leftarrow \mathcal{R}^N$, and recompute $z_{1-b} := x_b \odot x_{1-b} - z_b \in \mathcal{R}^N$.

2. Send $(x_0, z_0)$ to $P_0$ and $(x_1, z_1)$ to $P_1$.

Figure 18: Functionality for OLE.

### F.3 Oblivious Linear Evaluation

In Figure 18, we present the functionality $F_{\text{OLE}}$ for oblivious linear evaluation (OLE), where two parties obtain their additive shares of the component-wise multiplication result $x_0 \odot x_1 \in \mathcal{R}^N$.

$F_{\text{OLE}}$ can be efficiently realized from, e.g., linearly homomorphic encryption [DPSZ12, KPR18], oblivious transfer [KOS16, GNN17], or DPF [BCG+20], under their respective assumptions.