

Collusion-Resistant Constrained PRFs for Compute-&-Compare Predicates from LWE

Jiaqi Cheng
UW-Madison*

Rishab Goyal
UW-Madison†

September 2, 2026

Abstract

We design the *first collusion-resistant* constrained PRFs (CPRFs) for a non-trivial and expressive class of constraints from *standard LWE*. The two predicate classes for which we design CPRFs are: compute-&-compare and predicated range constraints. We improve our CPRF for compute-&-compare predicates to also satisfy *collusion-resistant constraint privacy*. An additional feature of our CPRFs is that they also satisfy (almost-)key-homomorphic property. Prior to this work, we did not have any post-quantum collusion-resistant CPRF beyond prefix-fixing constraints, and collusion-resistant CPRFs for expressive predicates relied on either *code obfuscation* or *multilinear maps*.

As an immediate application, we obtain a two-sided predicate encryption (PE) and functional encryption (FE) for the compute-&-compare class in the symmetric-key setting. Prior to this work, we did not have any post-quantum construction for 2-sided PE/FE beyond inner product predicates. An important contribution of this work is to introduce a new framework of *purifying functionality*. The main motivation behind our new framework is to systematically eliminate zeroizing attacks, which have been a highly successful cryptanalysis paradigm for breaking various candidates for advanced cryptographic objects.

*Email: jiaqicheng@cs.wisc.edu.

†Email: rishab@cs.wisc.edu.

Contents

1	Introduction	1
1.1	Our Results	2
1.2	Applications	3
1.3	Related work	4
2	Technical Overview	6
2.1	Achieving collusion-resistance for compute-&-compare	9
2.2	Constraint privacy for compute-&-compare predicates	10
2.3	CPRFs for predicated range constraints	12
2.4	Applications to 2-Sided Predicate Encryption	12
2.5	Discussion and Future Directions	13
3	Lattice Preliminaries	14
3.1	Lattice Evaluation and Homomorphic Encryption of Matrix Functions	15
4	Cryptographic Preliminaries & Definitions	16
4.1	Constrained Pseudorandom Functions with Constraint Privacy	17
4.2	Secret-Key Predicate Encryption with Predicate Privacy	18
5	Purifying Functionality: CPRFs and PE	19
5.1	Collusion-Resistant CPRFs for Purified Constraints	19
5.2	Predicate Encryption for Purified Predicates	22
5.3	Candidate Purified Circuit Families	22
6	Collusion-Resistant CPRFs for Compute-&-Compare Constraints	24
6.1	Correctness	25
6.2	Purified Constraint Pseudorandomness	26
7	Constraint-Hiding CPRFs for Compute-&-Compare Constraints	38
7.1	Correctness	40
7.2	Purified Constraint Privacy	40
7.3	Purified Constraint Pseudorandomness	47
8	Collusion-Resistant CPRFs for Predicated Range Circuits	51
8.1	Correctness	52
8.2	Purified Constraint Pseudorandomness	52
8.3	Indistinguishability of Hybrids	58
9	2-Sided Predicate Encryption for Purified Class	59
9.1	Purified 2-Sided Security	60
9.2	Purified Predicate Privacy	65
10	CPRFs in the Static Constraint Model	67

11 Bootstrapping purified circuits to general (impure) circuits	68
11.1 Further discussion: beyond CPRFs, constraint hiding, and more	70
12 Candidate CPRFs for Purified NC¹	70
12.1 Preliminaries	70
12.2 Candidate Construction	73
12.3 Correctness	74
12.4 Purified Pseudorandomness and Constraint Hiding	74
A Matrix Evaluation of Fully Homomorphic Encryption	96

1 Introduction

Constrained pseudorandom functions [BW13, BGI14, KPTZ13] (CPRFs) are a powerful advancement of pseudorandom functions (PRFs) [GGM86]. Syntactically, a CPRF is defined nearly identical to a standard PRF, as in it has a **Setup** algorithm that takes a security parameter λ , and outputs the (master) PRF key K , and an **Eval** algorithm that takes K and input x , and outputs the PRF value $F_K(x)$. The crucial difference between traditional and constrained PRFs is that CPRFs have another special algorithm, **Constrain**. The **Constrain** algorithm takes as input, the master key K and a boolean predicate ϕ , and outputs a “constrained” key, K_ϕ . The desired functionality from K_ϕ is that one can compute the PRF output $F_K(x)$ on any x as long as x satisfies the predicate ϕ (i.e., $\phi(x) = 1$), yet K is independent of ϕ as one wants to choose ϕ dynamically, after setup.

Pseudorandomness. A critical feature for CPRFs is their ability to maintain pseudorandomness even under a collusion attack. This is formalized as *constrained pseudorandomness* property. It guarantees that any polytime adversary that corrupts a group of constrained keys $K_{\phi_1}, \dots, K_{\phi_Q}$, for Q predicates $\{\phi_i\}_i$, cannot distinguish the PRF output on a point x^* as long as none of the predicates satisfy it. Here Q denotes the collusion size, and CPRF is said to be collusion-resistant if Q can be any polynomial.

Privacy. Another crucial property for CPRFs is that of *constraint privacy* [BGI14, KPTZ13, BLW17]. It states an adversary cannot learn any non-trivial information about the predicates $\{\phi_i\}_i$ from a set of constrained keys $\{K_{\phi_i}\}_i$. The notion of non-trivial information has to be defined carefully as the attacker can pick an input x , use every constrained key to evaluate on x , and compare the resulting outputs to learn which constraints satisfy x [BLW17].

Current state of affairs. Despite a decade of intensive study, achieving collusion-resistant CPRFs for general predicates remains a formidable challenge. Without leveraging heavy cryptographic machinery (such as indistinguishability obfuscation [BGI⁺01], multilinear maps [BS02, GGH13a, CLT15, GGH15]), our ability to construct collusion-resistant schemes is remarkably limited. Even when constraint privacy is not a requirement, existing collusion-resistant CPRFs are restricted to a narrow class of functions: specifically, prefix-fixing constraints [GGM86, BW13, BGI14]¹ and left-right predicates [BW13]². Neither class provides the expressivity required to support the diverse array of applications envisioned for constrained pseudorandomness. Moreover, if one desires *post-quantum* security, the landscape narrows further; only prefix-fixing PRFs are currently known, as they follow directly from the seminal Goldreich-Goldwasser-Micali (GGM) construction [GGM86, BW13, BGI14].

More remarkably, designing CPRFs does not get easier even against a ‘fixed’ collusion size [DKXY02, SS10, GLW12, GVW12]. In the fixed collusion setting, Q is a-priori fixed during setup and security is only guaranteed if no more than Q keys get corrupted. For most primitives, such as encryption, signatures, and even certain forms of obfuscation, restricting to bounded collusion security makes the problem much easier [DKXY02, SS10, GLW12, GVW12, GKP⁺13]. In sharp contrast, CPRFs supporting a small (e.g., two) number of corrupted keys remain elusive to this day. All prior CPRFs that go beyond inner-product predicates can either resist a single collusion (i.e., $Q = 1$), or require obfuscation [BW13, KPTZ13, BGI14, BV15, DKW16, CC17, BTVW17, BKW17, BKM17, PS18,

¹A prefix-fixing constraint is defined using a string P such that x satisfies the constraint if P is a prefix of x .

²A left-right predicate is defined using a half-string w and a ‘left/right’ flag such that an input $x = (x_{\text{left}}, x_{\text{right}})$ satisfies it if $w = x_{\text{left}}$ and flag is set to ‘left’, or vice versa.

CVW18b, AMN⁺18, AMN⁺19, HKKW19, DKN⁺20, Ria24, KWZ22, CMPR23, CDD⁺24, SS24, BMO⁺25].

Why are CPRFs so hard to design? On a fundamental level, this large schism between CPRFs and other advanced primitives stems from the fact that CPRFs implicitly enforce a stronger consistency property. Two constrained keys for a CPRF must output *identical* PRF values on a common satisfying input. For other advanced primitives, this consistency property is not as stringent. As an example, consider attribute-based encryption (ABE) [SW05, GPSW06] where two predicate keys only need to recover the plaintext from a ciphertext and nothing else.

This strong consistency requirement raises the technical difficulty in designing CPRFs, which brings us to the central question that we study:

Q1. *Can we design collusion-resistant CPRFs from standard assumptions for any class of constraint predicates that cannot be captured as inner-products?*

Additionally, we ask:

Q2. *Can we design collusion-resistant CPRFs with constraint privacy for any non-trivial class of predicates from standard assumptions?*

As two-key constraint privacy for general circuits is known to imply obfuscation [CC17], thus one must carefully delineate a non-trivial class of predicates that provides meaningful expressivity without inadvertently collapsing into the regime of obfuscation.

1.1 Our Results

In this work, we give positive answers to both questions, Q1 and Q2, while emphasizing on realizing post-quantum security. We give the *first* post-quantum collusion-resistant CPRF, that goes beyond prefix-fixing [GGM86, BW13, BGI14], and also prove it to be constraint private. As a direct application of our CPRFs, we give the *first* post-quantum collusion-resistant 2-sided PE for a non-trivial class of predicates. A key technical contribution of this work is a new framework, that we call *Purified Circuit Class* framework, to eliminate susceptibility to zeroizing attacks [CLT14, CHL⁺14, CGH⁺15, BGH⁺15, BBKK18, CVW18b, AP20, HJL21, JLLS23] in the context of lattice-based CPRFs. Below we summarize our key results:

1. **Collusion-Resistant *Constraint-Private* CPRF for compute-&-compare predicates from **LWE**.** We construct a CPRF for compute-&-compare predicates (i.e., equality testing on function output). Each constraint predicate $CC[f, y]$ is associated with a ‘compute’ function f (that can be represented by a polynomial-sized circuit) and a function output y . A CPRF key for $CC[f, y]$ can be used to evaluate the PRF on any input x as long as $f(x) = y$. We prove our CPRF satisfies collusion-resistant constrained pseudorandomness and constraint privacy, under the Learning With Errors (LWE) assumption [Reg05], as long as the attacker corrupts constrained keys for the same compute function f but many different comparison points $y_1, \dots, y_Q$. Prior CPRFs were secure as long as an attacker gets *at most one* constrained key and, as we demonstrate in this work, prior lattice-based CPRFs can be easily broken if we corrupt just two keys for compute-&-compare predicates, even with same compute function f and two distinct comparison points.

2. **Collusion-Resistant CPRFs for predicated range from LWE.** We generalize our main CPRF construction to support *predicated range* constraints (i.e., range checking on function output). Each constraint predicate $\text{CC}[f, y, z]$ is associated with a ‘compute’ function f and an output range $[y, z]$, where an input x satisfies the predicate if $f(x) \in [y, z]$. We prove our CPRF for predicated range satisfies collusion-resistant constrained pseudorandomness as long as the ranges are pairwise disjoint. We leave it as an interesting open question to prove constraint privacy for this constraint class.
3. **2-Sided Predicate Encryption.** As a direct application of our CPRFs, we construct a collusion-resistant 2-sided (or strongly attribute-hiding) predicate encryption (PE) scheme [BW07, KSW08, GKW17a] for compute-&-compare predicates under standard LWE. Our encryption system is a symmetric-key scheme and also satisfies predicate privacy (i.e., decryption keys hide the predicate). Prior to this work, 2-sided PE could only be built for inner-product predicates [BW07, KSW08], or by invoking obfuscation and related primitives [GGH⁺13b, SW14, ABSV15].

Key-Homomorphism: an added benefit. Our CPRF constructions also satisfy (almost-)key-homomorphic property w.r.t. the master key as well as constrained keys [BLMR13, BFP⁺15, CVW⁺18a, GQWW19]. Briefly, (almost-)key-homomorphism states that $F_{K_1}(x) + F_{K_2}(x) \approx F_{K_1+K_2}(x)$, where K_1, K_2 are two master keys or two constrained keys for the same predicate. The key-homomorphism property has proven to be extremely valuable for a variety of applications [BLMR13, BFP⁺15, CVW⁺18a, GQWW19] and, to the best of our knowledge, we did not have any collusion-resistant key-homomorphic CPRF for any predicate other than prefix-fixing predicates from standard assumptions.

1.2 Applications

Our construction of a symmetric-key 2-sided PE scheme yields significant implications. Notably, this PE scheme inherently provides symmetric-key function-hiding functional encryption (FE) [SSW09, GKP⁺13, AAB⁺15] for the same class. Below, we outline several key applications.

Searchable and Order-Revealing Encryption. Searchable Symmetric Encryption (SSE) and Order-Revealing/Preserving Encryption (ORE/OPE) enable efficient querying over encrypted databases [SWP00, Goh03, AKSX04, CGKO06, BCLO09, CK10, BCO11, BLR⁺15]. Our 2-sided PE for compute-&-compare allows for secure sorting and searching while maintaining attribute hiding even for successful queries.

Consider an organization hosting employee records on a third-party cloud. The organization can delegate the ability to retrieve records within a specific salary interval $[t, 2t]$ or identify employees in a specific location. By utilizing our 2-sided security, the cloud server can execute these queries without learning the underlying salary values or geographical data, even for the records that match the query criteria.

Watermarkable and Traceable Constrained PRFs. Watermarkable [HMW07, BGI⁺01, CHN⁺16] and traceable PRFs [GKWW21] allow a master authority to embed and detect “watermarks” within secret keys to prevent unauthorized distribution or misuse. These are vital for challenge-response authentication and traitor tracing.

Existing schemes primarily support watermarking the master PRF key, which is insufficient for systems requiring fine-grained access control (e.g., identity-based traitor tracing [NWZ16,

GKW18]). By leveraging our collusion-resistant CPRFs with constraint privacy, one can design watermarkable constrained keys. This allows an authority to delegate restricted PRF capabilities while retaining the ability to trace the specific constrained key used in a leak or misuse.

Collusion-Resistant Private Constrained MACs. Private constrained MACs [BGI14, BLW17]—the symmetric analogue of constrained signatures—allow a master key holder to delegate the ability to tag messages that satisfy a specific predicate. Applications include enforcing departmental spending limits or delegating university purchase authorizations without revealing the underlying policy or employee credentials.

Prior constructions based on 1-key private CPRFs were limited to single-user security. Our results directly enable collusion-resistant private constrained MACs, allowing multiple delegated users to operate simultaneously without compromising the privacy of the organizational policies or the security of the tagging mechanism.

Fraud and Threat Detection. Our framework naturally supports “investigative” encryption models [BW07, SBC⁺07, GVW15], such as credit card fraud detection or suspicious content filtering. In these scenarios:

- *Fraud Investigation:* Investigators can get keys to identify transactions with specific fraudulent signatures (e.g., target ranges or specific metadata matches).
- *Threat Detection:* Cloud providers can host keys that trigger alerts for suspicious keywords or communication patterns.

Since the desired functionality can be naturally implemented as compute-&-compare and predicated range policies, thus the use of 2-sided PE ensures that the sensitive details of the transaction or communication remain hidden from the investigator until a “suspicious” item is flagged.

Beyond the specific applications discussed, the collusion-resistance and key-homomorphism properties of our construction offer broader utility in the design of advanced lattice-based primitives. Key-homomorphic CPRFs have played an important role in traitor tracing [CFN94, GKW18, CVW⁺18a] and trace-and-revoke systems [NP01, GQWW19]. By providing the first collusion-resistant realization for compute-&-compare and predicated range classes, our work effectively removes the ‘single-key barrier’ that has limited applications. We expect this framework to inspire new research into achieving collusion resistance for even more expressive, possible non-purified circuit classes under standard lattice assumptions.

1.3 Related work

The concept of CPRFs was concurrently introduced by Boneh and Waters, Boyle et al., and Kiayias et al. [BW13, BGI14, KPTZ13]. They observed that the GGM construction [GGM86] inherently provides a CPRF for prefix-fixing predicates under the minimal assumption of one-way functions (OWFs). Since the work of Sahai and Waters [SW14], puncturable PRFs have become a fundamental tool in applications of indistinguishability obfuscation (iO). Broadly, existing CPRF constructions can be categorized by their underlying cryptographic techniques: obfuscation-based, lattice-based, group-based, and combinatorial approaches.

Obfuscation-Based Constructions. Using iO or related “heavy hammers,” several constructions achieve collusion-resistant CPRFs for all circuits and even Turing machines [DKW16, BLW17, BKW17]. Some of these schemes [BKW17, BLW17] also achieve collusion-resistant constraint

Table 1: Comparison of Post-Quantum Constrained PRFs

Scheme	Constraint	Privacy	Coll.	Assumption
[BTVW17]	General Circuits	✓	1	LWE
[CC17]	NC ¹	✓	1	LWE
[DKN ⁺ 20]	Inner Product	✓	1	LWE
[DKN ⁺ 20]	t -CNF	✓	$O(1)$	OWF
[BFP ⁺ 15]	Prefix	✗	poly	LWE
[Yan23]	Puncturing	✓	poly	LWE
This Work	Comp-Compare	✓	poly	LWE
This Work	Range	✗	poly	LWE

Note: **Coll.:** Supported collusion keys (poly = unbounded). **RO:** Random Oracle. **Comp-Compare:** Compute-&-Compare. Comparisons are restricted to schemes without obfuscation.

privacy. While expressive, these constructions rely on the existence of iO, making them less suitable for deployment under standard, well-studied assumptions like LWE.

Lattice-Based Constructions. Under standard lattice assumptions, numerous constructions exist for general circuits [BV15, BKM17, CC17, BTVW17]. Several of these have been extended to provide 1-key constraint privacy. However, a significant limitation persists: all current lattice-based templates are restricted to single-key security. As discussed in our technical overview, these algebraic structures (e.g., BGG+ or GGH) are inherently vulnerable to key-subtraction or mix-and-match attacks when more than one key is issued.

Group-Based Constructions. In the random oracle model (ROM), Boneh and Waters [BW13] designed collusion-resistant CPRFs for left-right constraints using bilinear maps. Subsequent works [GHKW17, Bit17, AMN⁺18] achieved 1-key security for substring and NC¹ predicates under DDH-hard groups. More recently, Ishai et al. [ILL24] designed 1-key CPRFs for general constraints based on DCR or DDH with small-exponent assumptions. While recent works [CMPR23, BMO⁺25] have explored 1-key private CPRFs for inner-product predicates, none of these group-based schemes (save for the specialized ROM construction in [BW13]) support more than one key corruption or satisfy constraint privacy for general classes.

Combinatorial Approaches. A final direction involves restricting the predicate class to inner products or $O(1)$ -CNFs and leveraging combinatorial techniques [Tsa19, DKN⁺20, CDD⁺24]. While these approaches utilize mild assumptions (e.g., OWFs), they are subject to a “combinatorial explosion” regarding corruption sets; for instance, the scheme of Davidson et al. [DKN⁺20] resists only $O(1)$ corruptions. These methods appear fundamentally limited and unlikely to generalize to general constraints or full collusion resistance.

Summary. Prior to this work, the landscape of CPRFs was divided between iO-based schemes (which provide full collusion resistance for all circuits) and standard-assumption schemes (which are almost exclusively limited to single-key security). Our work bridges this gap, providing the first collusion-resistant CPRFs with constraint privacy from standard lattice assumptions for meaningful predicate classes.

Looking ahead. Over the last few years, numerous works have proposed new non-standard (and even non-falsifiable) lattice assumptions [Wee22, Tsa22, VWW22, WWW22, ARYY23, HLL23, Wee24, HLL24, AKY24, Wee25, HLL25] to go beyond established barriers in lattice-based cryptography. This has had mixed success. While relying on these non-standard assumptions led to the first provable designs for many advanced primitives [Wee22, Tsa22, VWW22, Wee24, HLL24, Wee25, HLL25] for which we do not yet have any constructions under standard lattice assumptions, their foundations have not resisted cryptanalysis [AMYY25, HHY25, DJM⁺25, BÜW24, HJL25]. Although recent works have made important progress towards the goal of lattice-based (post-quantum) functional cryptography, we believe a paradigm shift is essential in the community’s approach in order to design advanced primitives while reducing to standard LWE.

Taking a step back, we view our results as proposing a new avenue for bypassing zeroizing attacks in lattice-based cryptography. Such attacks have been a consistent source of successful cryptanalytic strategies on various lattice-based candidates. We believe that our purification framework could be widely applicable across other primitives in lattice-based cryptography to design new schemes with provable security under standard LWE.

2 Technical Overview

We start with the key technical barrier that deters realizing collusion-resistant CPRFs for circuits from lattices.

Zeroizing attacks: a key technical barrier. Zeroizing attacks are a powerful methodology to cryptanalyze candidates for advanced cryptographic objects [CLT14, CHL⁺14, CGH⁺15, BGH⁺15, BBKK18, CVW18b, AP20, HJL21, JLLS23]. Abstractly, a zeroizing (colluding) attacker in CPRFs works as follows: (1) corrupt two (or more) constrained keys, (2) evaluate them on an input x that satisfies all constraints (which must give the same output by CPRF correctness), and (3) algebraically combine these keys and outputs, and extract non-trivial (secret) encodings of ‘zero’.

Step (3) is the most crucial step and it reveals non-trivial information about the PRF master key msk in the lattice setting. Basically, each PRF output non-trivially depends on msk , and if two constrained keys can evaluate the PRF on a common point, then they can be used to learn the same msk encoding in two different ways. In lattice-based CPRFs, being able to compute the same encoding of msk in two different ways leaks underlying errors, eventually breaking pseudorandomness. Therefore, all prior lattice-based CPRFs [BV15, BTVW17, CC17, CVW18b, BKM17] succumb to zeroizing attacks. Since zeroizing attacks require at least two keys, thus prior works could prove 1-key constrained pseudorandomness, but fell apart when an attacker corrupts even two.

Our goal is to allow an arbitrary number of key corruptions, while avoiding zeroizing attacks. Towards that goal, we introduce a new notion that we call ‘purified circuit classes’ to eliminate susceptibility to zeroizing attacks.

“Purifying” Functionality: Eliminating Zeroizing Attacks. Our starting observation is that the abstract zeroizing attack methodology no longer works if an attacker corrupts *constraint keys that do not have an overlapping support*. Let $\hat{\mathcal{C}}$ be a class of predicates where every pair (distinct) predicates have a disjoint support. That is, $\forall C \neq C' \in \hat{\mathcal{C}}, \text{Supp}(C) \cap \text{Supp}(C') = \emptyset$.

We call such a predicate class, $\hat{\mathcal{C}}$, to be a *purified constraint class*. Intuitively, this enforces zero overlap between constraints. Essentially, each constrained key sk_C controls a unique and disjoint partition of the input space. Our intuition is that if attacker can only acquire secret keys that

correspond to a purified class, then zeroizing attack methodology might not work. Basically, the abstract zeroizing attack template stops at step (2) when an adversary cannot evaluate two keys on the same point anymore.

Naturally, moving to this framework invites three fundamental questions:

Q1 (Utility): Are there meaningful predicate classes that reside within this ‘purified’ regime?

Q2 (Complexity): Does the restriction to purified classes render the construction of collusion-resistant CPRFs trivial?

Q3 (Gap Analysis): Are existing lattice-based CPRFs already implicitly secure for such classes, or can we identify concrete vulnerabilities?

In the following, we address each of these in turn.

Addressing Q1: The Existence of Expressive Purified Predicates. We answer in the affirmative by identifying two significant and expressive families of purified predicates.

I. compute-&-Compare Predicates. Our first example is the class of compute-&-compare circuits. Informally, each circuit in this family checks whether a function f maps an input x to a target value y . Formally, for a function $f : \mathcal{X} \rightarrow \mathcal{Y}$ and a target $y \in \mathcal{Y}$, we define the circuit $C_{f,y}$ such that:

$$C_{f,y}(x) = 1 \iff f(x) = y$$

For a given f , any collection of circuits $\mathcal{F} \subseteq \{C_{f,y} : y \in \mathcal{Y}\}$ constitutes a compute-&-compare class. Notably, any such class is inherently purified: for any two distinct targets $y \neq y'$, the supports of $C_{f,y}$ and $C_{f,y'}$ are disjoint as f is a function (i.e., $\text{supp}(C_{f,y}) \cap \text{supp}(C_{f,y'}) = \emptyset$).

II. Predicated Range Constraints. We further generalize this to the predicated range class. Here, the constraint is satisfied if the function output falls within a specific interval. For a function f , we define the range circuit $C_{f,[y,z]}$ as:

$$C_{f,[y,z]}(x) = 1 \iff y \leq f(x) \leq z$$

A collection of such circuits $\{C_{f,[y_i,z_i]}\}$ is a purified predicated range class provided the corresponding intervals $[y_i, z_i]$ in the co-domain $\mathcal{Y}$ are pairwise non-overlapping. This class offers a powerful generalization, allowing for delegated evaluation over partitioned sub-domains of the function’s range.

These predicates have been studied in the context of code obfuscation [GKW17b, WZ17] and are fairly common in the searchable symmetric encryption (SSE) literature [AKSX04, BCLO09, BCO11, BLR⁺15]. We refer the reader to Section 1.2 for a detailed discussion on potential applications.

Addressing Q2: The Non-Triviality of Purified Constraints. The reader might be wondering: does restricting the predicate class to disjoint supports (the “purified” regime) render the construction of collusion-resistant CPRFs trivial? One might consider a naive approach that partitions the input space $\mathcal{X}$ according to the constraints and assigns independent PRF keys to each partition—for instance, defining $\text{Eval}(\text{msk}, x) = F(K_{\text{partition}(x)}, x)$.

However, such an approach fails to satisfy the fundamental requirement of a CPRF: independence of core PRF f. In a standard CPRF, the Setup and Eval algorithms must remain independent of the specific constraint issued. The master key msk defines a single, consistent pseudorandom function over the entire domain $\{0, 1\}^n$, whereas constraints can be selected dynamically via the Constrain algorithm later. A partition-based approach would require “fixing” the constraint (e.g., the function f in a

compute-&-compare class) during the **Setup** phase. This would yield a constraint-specific CPRF rather than a general-purpose CPRF. Our construction maintains a single msk capable of issuing keys for any predicate in the class (and for arbitrary functions f) without requiring apriori knowledge of these constraints at setup time.

We emphasize that the requirement of **Setup** and **Eval** being independent of the constraining predicates is essential for nearly all applications as well as it ensures the problem does not become trivial. In particular, designing CPRFs even for general constraints (i.e., without the purified restriction) is relatively straightforward if the **Setup** or **Eval** algorithm knows the set of constraints $C_1, \dots, C_Q$ for which a key will be generated. The idea is simple: partition the input space into 2^Q disjoint regions based on the validity vector $\mathbf{y} \in \{0, 1\}^n$ of the input $\mathbf{x}$ against the constraints (where $\mathbf{y}[i] = C_i(\mathbf{x})$). The system simply assigns an independent standard PRF key $K_{\mathbf{y}}$ for each possible vector $\mathbf{y}$, and a constrained key for C_i corresponds to $\{K_{\mathbf{y}}\}$ for all $\mathbf{y}$ such that $\mathbf{y}[i] = 1$. For completeness, we provide this simple construction for CPRFs in this so-called “static-constraint” model in Section 10.

Addressing Q3: The Failure of Existing Lattice-Based CPRFs. With the introduction of purified circuits, it is natural to ask whether existing lattice-based CPRFs [BV15, BTVW17, BKM17, CC17, CVW⁺18a] already satisfy collusion resistance when restricted to such classes. We answer this in the negative. We examine the two dominant algebraic paradigms—the BGG+ approach and the GGH approach—and demonstrate that both are fundamentally insecure against collusion, even in the purified setting.

I. Vulnerability of the BGG+ Approach. The constructions in [BV15, BTVW17, BKM17] utilize the homomorphic properties of the BGG+ framework [BGG⁺14]. In these schemes, the constrained key for a function f is typically an encoding of the form:

$$\text{sk}_f := \mathbf{s}^T \cdot (\mathbf{B} - f \otimes \mathbf{G}) + \mathbf{e}$$

where $\mathbf{s}$ is the LWE secret (serving as the PRF seed), $\mathbf{B}$ is a public matrix, and $\mathbf{G}$ is the gadget matrix. Correctness relies on the fact that for any x such that $f(x) = 0$, one can homomorphically derive $\mathbf{s}^T(\mathbf{B} \cdot \mathbf{H}_x) \approx \text{Eval}(\text{msk}, x)$.

The Attack: This design collapses under collusion for the compute-&-Compare class. Consider an adversary who obtains two keys sk_{y_0} and sk_{y_1} for the same function f but distinct targets y_0, y_1 . The keys take the form:

$$\text{sk}_{y_i} \approx \mathbf{s}^T \cdot (\mathbf{B} - [f \mid y_i] \otimes \mathbf{G})$$

By simply computing the difference $\text{sk}_{y_0} - \text{sk}_{y_1}$, the $\mathbf{s}^T \mathbf{B}$ and f -dependent terms cancel out, leaving a term of the form $\mathbf{s}^T \cdot ((y_1 - y_0) \otimes \mathbf{G}) + \mathbf{e}'$. Since $y_0 \neq y_1$, the adversary can use the gadget structure of $\mathbf{G}$ to recover the LWE secret $\mathbf{s}$ in its entirety. This total break of the master secret key demonstrates that the BGG+ framework cannot natively support collusion resistance.

II. Vulnerability of the GGH Approach. The second paradigm, based on GGH-style encodings [GGH15], constructs CPRFs for log-depth or log-space circuits using chains of matrices [CC17, CVW18b]. Here, the constrained key is a sequence of “links” that allow evaluation only if the input x traverses a valid path in a branching program.

The Attack: These constructions are susceptible to a mix-and-match attack. Even in the purified setting (e.g., bit-fixing constraints), an adversary holding keys for two disjoint constraints can “splice” the matrix chains together. For example, by combining the first half of a chain from Key A with the second half of Key B, the adversary can evaluate the PRF on inputs that satisfy neither

constraint. While this attack can be mitigated for single-key security, the algebraic structure of the GGH encodings inherently allows for the recombination of chain segments across multiple keys, effectively breaking collusion resistance.

The vulnerabilities identified in these two templates reveal a fundamental gap: existing lattice-based CPRFs are susceptible to direct algebraic breaks, even when the adversary is restricted to a purified class of constraints. This demonstrates that existing techniques are insufficient for achieving collusion resistance in this regime. Our construction addresses this gap by effectively immunizing constrained keys against “algebraic combination” attacks. We achieve this by expanding the lattice-based encodings and leveraging the inherent structure of the compute-&-compare and predicated range classes.

2.1 Achieving collusion-resistance for compute-&-compare

We first describe a baseline design that captures the main idea of our approach: algebraically decoupling the function description from the output value by introducing an explicit offset term that depends only on the target output. In the standard BGG+ instantiations for compute-&-compare, the target y is embedded directly into the homomorphic evaluation path (e.g., encoding the circuit $C_{f,y}$ such that $C_{f,y}(x) = 1$ if only if $f(x) = y$).

Our first idea is to sample a fresh random matrix $\{\mathbf{A}_y\}_y$ for each distinct target value. Given this matrix, we will keep the BGG+ encoding dependent on f only, and move target dependence into a separate offset. A direct way to enforce this is to represent constrained evaluation as a shifted version of master key evaluation:

$$(\mathbf{B} - \mathbf{f}^T \otimes \mathbf{G}) \cdot \mathbf{H}_{\mathcal{U}_{x,f}} = \mathbf{B} \cdot \mathbf{H}_{\mathcal{U}_x} - \mathbf{A}_{f(x)},$$

where $\mathbf{A}_{f(x)}$ is an offset family indexed by the output of target function f on input x . For a constrained key for function f and target y , the first component remains the standard BGG+ term $\text{sk}_{f,y}^{(1)} = \mathbf{s}^T \cdot (\mathbf{B} - \mathbf{f}^T \otimes \mathbf{G}) + \text{noise}$. Using this component, the evaluator computes

$$\text{sk}_{f,y}^{(1)} \cdot \mathbf{H}_{\mathcal{U}_{x,f}} \cdot \mathbf{G}^{-1}(\mathbf{C}_x) \approx \mathbf{s}^T \cdot (\mathbf{B} - \mathbf{A}_{f(x)}) \cdot \mathbf{H}_{\mathcal{U}_x} \cdot \mathbf{G}^{-1}(\mathbf{C}_x),$$

where $\mathbf{G}^{-1}(\mathbf{C}_x)$ is the standard “bookend” matrix to ensure the output looks pseudorandom, and not just unpredictable. So the baseline constrained evaluation is exactly the master key evaluation minus the offset indexed by $f(x)$. To ensure this offset can be canceled whenever the target matches, we give out a target-specific correction term as the second component of the constrained key: $\text{sk}_{f,y}^{(2)} := \mathbf{s}^T \cdot \mathbf{A}_y + \text{noise}_y$. The evaluator maps it to

$$\text{sk}_{f,y}^{(2)} \cdot \mathbf{G}^{-1}(\mathbf{C}_x) = (\mathbf{s}^T \cdot \mathbf{A}_y + \text{noise}_y) \cdot \mathbf{G}^{-1}(\mathbf{C}_x)$$

and adds this term to the expression above. Hence the final constrained output is

$$\underbrace{\mathbf{s}^T \cdot \mathbf{B} \cdot \mathbf{H}_{\mathcal{U}_x} \cdot \mathbf{G}^{-1}(\mathbf{C}_x)}_{\text{master key evaluation core}} + \underbrace{\mathbf{s}^T \cdot (\mathbf{A}_y - \mathbf{A}_{f(x)}) \cdot \mathbf{G}^{-1}(\mathbf{C}_x)}_{\text{residual offset}} + \text{noise}.$$

If $f(x) = y$, then $\mathbf{A}_y = \mathbf{A}_{f(x)}$, so the residual offset vanishes and we recover the master evaluation value. If $f(x) \neq y$, the residual offset remains and cancellation fails. This is precisely the compute-&-compare behavior we need. We point out that the first term will be (nearly) identical in different

constraint keys as long as they encode the same function f . Thus, an attacker cannot get any meaningful information from multiple encodings of the same function f it gets from two or more keys. Now while the second component is specialized to the target y , it cannot be used to mount a pseudorandomness attack. This is due the fact that $\mathbf{A}_y$ is randomly sampled for each target value y , thus looks like an LWE sample.

This separation is the key structural feature that enables many constrained keys for one fixed function without re-encoding targets in the shared BGG+ part. The above design relies on the matrix-valued homomorphic evaluation: the universal circuit $\mathcal{U}_x$ is defined to generate the matrix output, namely $\mathcal{U}_x(f) = \mathbf{A}_{f(x)}$. We remark that the matrix-valued homomorphic evaluation was originally developed in the context of proving constraint privacy for 1-key CPRFs [BTVW17], while we use it for homomorphically computing a ‘computation-dependent’ offset.

Handling inefficiency of Setup. In the baseline approach, Setup samples an independent random public matrix $\mathbf{A}_y$ for every $y \in \mathcal{Y}$, and the constrained key for target y includes the noisy linear term $\mathbf{s}^T \mathbf{A}_y + \text{noise}$. While sampling each $\mathbf{A}_y$ matrix randomly substantially helps in proving collusion-resistant constrained pseudorandomness, it makes the CPRF system to be highly inefficient. Note that the universal circuit $\mathcal{U}_x$ is effectively defined with this table hardwired (as this circuit must know which of the offset matrices need to added). Therefore, both parameter size and representation complexity scale with $|\mathcal{Y}|$. Effectively, when $|\mathcal{Y}|$ is exponential, this system becomes infeasible. Thus, we would need the number of targets $|\mathcal{Y}|$ to be polynomially bounded. This is a severe restriction that we would like to avoid.

To remove this bottleneck, we keep exactly the same offset-correction structure and replace only how $\mathbf{A}_y$ is generated. Instead of hardwiring an explicit table $\{\mathbf{A}_y\}_{y \in \mathcal{Y}}$ in pp, we instantiate $\mathbf{A}_y$ using the matrix-generation framework of any of the prior key-homomorphic PRF works such as [BLMR13, BP14]. In this approach, $\mathbf{A}_y$ is efficiently computable from y via a recursive or sequential gadget-based composition, so setup no longer needs to publish one fresh random matrix per target value. The constrained key still contains the same target handle $\mathbf{s}^T \mathbf{A}_y + \text{noise}_y$, and constrained evaluation is unchanged at the algebraic level. Hence correctness continues to follow from the same offset cancellation argument.

This replacement preserves the pseudorandomness argument as well. From the adversary’s view, each target handle remains an LWE sample indexed by the queried target. So for polynomially many queries, the joint distribution $(\mathbf{s}^T \mathbf{B} + \text{noise}, \mathbf{s}^T \mathbf{A}_{y_i} + \text{noise}_{y_i})_{i \in [Q]}$ is still pseudorandom under LWE. Therefore we keep the same collusion-resistant security intuition while eliminating the explicit $|\mathcal{Y}|$ -size public table.

We point out that the above construction naturally satisfies *key-homomorphism* property for, both, master keys as well as constrained keys. Key-homomorphism for master key follows the same argument as many prior works [BLMR13, BP14], which is the PRF computation is affine in the master secret key. For key-homomorphism of constrained keys, we can again rely on the same affine property as well as the fact that public-offset matrix $\mathbf{A}_y$ will be common across different systems.

2.2 Constraint privacy for compute-&-compare predicates

Our next goal is to also achieve constraint privacy for our compute-&-compare CPRF scheme. Observe that our previous construction already hides the target value y in each constrained key. In particular, the target-specific component is $\mathbf{s}^T \mathbf{A}_y + \text{noise}_y$, which is an LWE sample and is computationally indistinguishable from random. So the constrained key does not directly reveal the target value. However, this is not sufficient for constraint privacy. The evaluation algorithm still

requires the compute function f to compute $\mathbf{H}_{\mathcal{U}_x, f}$, so f must effectively be given to the evaluator. Clearly, this is not constraint private anymore.

To hide f , we follow the template of [GVW15, BTW17] and encrypt f under homomorphic encryption (HE) [GSW13]. At a high level, we replace the plaintext evaluation of f with homomorphic evaluation on a ciphertext $\text{ct} = \text{Enc}(f)$, while reusing the same LWE secret $\mathbf{s}$ in both the BGG+ layer and the HE layer. This secret reuse is the standard mechanism that allows the evaluator to combine homomorphic output with the constrained-key term and recover the correct value without learning the underlying circuit.

Our setting requires one additional modification. Prior boolean templates rely on decryption of a bit that directly cancels the gap between constrained and master evaluation. In our construction, that gap is matrix-valued and equals the offset indexed by $f(x)$. So we extend the GSW homomorphic evaluation to matrix outputs as well. Let $\mathcal{U}_x$ be the universal circuit with output $\mathbf{A}_{f(x)}$, and let $\text{MtrxEval}(\mathcal{U}_x, \text{ct})$ denote homomorphic matrix evaluation on encrypted f . We use AllButLastRow and LastRow for the top and bottom parts of this evaluated ciphertext, respectively. The constrained key is

$$\text{sk}_y = \left(\mathbf{s}^T (\mathbf{B} - \text{ct} \otimes \mathbf{G}) + \text{noise}, \mathbf{s}^T \mathbf{A}_y + \text{noise}_y \right),$$

and constrained evaluation combines the first term with $\text{LastRow}(\mathcal{U}_x, \text{ct})$ and the target correction term. The crucial algebraic fact is that after extending GSW homomorphic evaluation to matrix outputs, $\text{MtrxEval}(\mathcal{U}_x, \text{ct})$ has the form

$$\begin{bmatrix} \mathbf{C} \\ \mathbf{s}^T \mathbf{C} + \text{noise} \end{bmatrix} \mathbf{R} + \begin{bmatrix} \mathbf{A}_{f(x)} \\ \mathbf{0} \end{bmatrix},$$

so applying $[\mathbf{s}^T \mid -1]$ extracts (up to noise) the matrix offset $\mathbf{s}^T \mathbf{A}_{f(x)}$. Hence the same offset-cancellation argument from the plaintext construction goes through. When $f(x) = y$, the correction $\mathbf{s}^T \mathbf{A}_y + \text{noise}_y$ cancels the extracted offset and constrained evaluation matches master evaluation up to noise tolerance. When $f(x) \neq y$, cancellation fails. Thus, this construction hides, both, the target y and the function f while providing collusion-resistance.

We remark that our above construction only satisfies key-homomorphism property w.r.t. master key, and no longer satisfies key-homomorphism for constrained keys. This is due to the fact that the HE ciphertext computation is randomized, thus does not obey the same additive features that our previous CPRF construction did. We leave the problem of achieving key-homomorphism and collusion-resistant constraint privacy (simultaneously) as an open problem.

Adaptive security in the ROM. Our standard-model proof establishes selective security, where the adversary commits to the relevant compute function before the public parameters are generated. This restriction can be removed in the random oracle model by the standard output-hashing transformation. The variant keeps the same constrained keys and the same algebraic evaluation procedure, but replaces the final output z_x by $H(z_x)$. For an admissible challenge input x^* , the underlying value z_{x^*} remains hidden from the adversary by the security of the base construction. Thus, except with negligible probability, the adversary never queries z_{x^*} to the random oracle. Conditioned on this event, $H(z_{x^*})$ is a fresh random string, and the simulator may program this value as the challenge output. This gives the usual adaptive security upgrade in the ROM, following [AMN⁺18].

2.3 CPRFs for predicated range constraints

Next, we generalize our base CPRF construction for compute-&-compare to the class of predicated range circuits. At a high level, our plan is to combine our base CPRF construction with the prefix-fixing CRPF based on GGM [GGM86] PRFs. Essentially, our idea is to ensure that the user of the constrained keys can do partial evaluation over the tree instead of only given the leaf vectors. More concretely, given $\mathbf{s}^T \cdot \mathbf{A}_y + \text{noise}$ as part of the constrained secret key, the user can locally approximate the value of $\mathbf{s}^T \cdot \mathbf{A}_{y|y'}$ for any y' . To achieve this, rather than adopting the framework of Banerjee-Peikert [BP14], we instead follow the approach of Boneh et al. [BLMR13]. The issue with [BP14] is that it computes $\mathbf{A}_y$ using a binary tree structure rather than sequentially. While it improves concrete efficiency, it is unclear whether the prefix-fixing ideas could be applied there. On the other hand, evaluation in [BLMR13] proceeds sequentially. In particular, let ℓ be the length of y , $\mathbf{A}_y$ is defined as

$$\mathbf{A}_y = \mathbf{A}_{y[1]} \cdot \mathbf{G}^{-1}(\mathbf{A}_{y[2]}) \cdot \mathbf{G}^{-1}(\mathbf{A}_{y[3]}) \dots \mathbf{G}^{-1}(\mathbf{A}_{y[\ell]}),$$

which allows the key-constraining algorithm to assign a partial key $\mathbf{s}^T \cdot \mathbf{A}_y + \text{noise}_y$ for arbitrary lengths of y , after which the user can independently compute the approximate value of $\mathbf{s}^T \cdot \mathbf{A}_{y|y'}$. By combining the above with the fact that any range constraint (even of exponential size) can be decomposed into polynomially many prefix checks (using the power-of-two trick), we can give out constrained keys for arbitrary range predicates efficiently. To illustrate this, consider the following simple example: let $[y, z]$ be the desired range, where $y = 0010$ and $z = 1011$. The key-constraining algorithm will assign partial keys corresponding to $001\square$, $01\square\square$, and $10\square\square$, where $\square$ denotes a wildcard position. Using this, the constraining algorithm can generate compute-&-compare -style constrained keys for differing length targets.

We highlight that although our collusion-resistant CPRF construction naturally extends our compute-&-compare CPRF scheme, we are unable to generalize it to prove (full) constraint hiding. The issue is that, if we rely on a similar homomorphic-encryption-based constraint hiding template, evaluating $f(x)$ requires explicitly using f to ‘select’ the partial key corresponding to a prefix of $f(x)$. Thus, the evaluator needs some information about $f(x)$ (which is just the maximum length of the matching prefix as per above decomposition) in order to correctly evaluate the PRF. However, we note that if an evaluator gets this additional information, then we can indeed prove constraint hiding for predicated range circuits. We believe such a weaker notion of constraint hiding might be useful for certain applications, and we leave the problem of constraint hiding for predicated range circuits as an interesting open problem.

2.4 Applications to 2-Sided Predicate Encryption

We explore the implications of our construction to 2-sided predicate encryption (PE), contrasting it with the more common 1-sided variant. In a standard PE scheme [BW07, KSW08], a secret key sk_f enables the decryption of a ciphertext associated with attribute x if and only if $f(x) = 1$. In the 1-sided setting, attribute-hiding is required only when $f(x) = 0$. In the 2-sided setting, x must remain computationally hidden even from a user who possesses a key that successfully decrypts the ciphertext (i.e., when $f(x) = 1$). Achieving 2-sided security for general predicates is notoriously difficult, as it is essentially equivalent to collusion-resistant Functional Encryption [SW05, BW07, KSW08, BSW11].

We introduce a refined security notion termed 2-sided purified security. This notion guarantees attribute-hiding provided that either:

1. **Non-satisfying keys:** All queried keys sk_{f_i} satisfy $f_i(x) = 0$.
2. **Purified satisfying keys:** Any queried key sk_{f_j} that satisfies $f_j(x) = 1$ belongs to a purified circuit class (e.g., compute-&-Compare).

Crucially, we do not restrict non-satisfying queries, which may be arbitrary. We demonstrate that any 1-sided PE scheme (instantiable from LWE [GVW15, GKW17a, WZ17]) can be generically transformed into a 2-sided purified PE scheme using our collusion-resistant CPRFs.

To encrypt a message m under attribute x , the encryptor does the following:

1. **Attribute Encapsulation:** Computes a symmetric-key encryption of x as $\text{ct}_x \leftarrow \text{SKE.Enc}(k, x)$.
2. **Tag Generation:** Derives a tag $\tau \leftarrow \text{CPRF.Eval}(\text{cprf.msk}, \text{ct}_x)$.
3. **1-Sided Encryption:** Encrypts m using a 1-sided PE scheme under the augmented attribute (ct_x, τ) .

To create a secret key, we first sample a constrained PRF key for a compute-&-compare predicate (f, y) that decrypts ct_x and checks if $f(\text{Dec}(k, \text{ct}_x)) = y$. Next, we sample a 1-sided PE key that checks if the CPRF evaluation (using the above constrain key) on the ciphertext matches the tag τ . Because the message is “shielded” by the CPRF tag, an adversary learns only the bit representing the success of the evaluation, leaving x hidden even upon successful decryption.

Our scheme only works in the symmetric-key setting as the encryptor needs to evaluate the CPRF for obtaining the ‘tag’ in the ciphertext. We leave it as an exciting open problem to generalize our construction to the public-key setting.

2.5 Discussion and Future Directions

Boosting to Impure Classes. A natural question is whether CPRFs for purified constraints can serve as a mechanism to bootstrap to more expressive, “impure” classes. While a jump to general circuits is likely precluded by some complexity theoretic lower bounds, we provide partial positive results for “weakly impure” constraints in Section 11.

We define a circuit class $\mathcal{C}$ as *weakly impure* if any $C \in \mathcal{C}$ can be expressed as a sum of $\text{poly}(\lambda)$ purified circuits $C_1, \dots, C_k$ from a purified class $\mathcal{C}'$. To issue a key for $C \in \mathcal{C}$, the authority generates k independent CPRF keys for the constituent purified circuits. This allows us to handle predicates with *bounded overlaps*. For example, consider a range-based class where two keys might overlap on a small, fixed-length interval. By “purifying” these overlaps during the key-generation phase (i.e., treating the intersection as its own disjoint sub-predicate) we can extend the utility of our construction to non-disjoint settings.

Open Question: Towards Arbitrary Purified Constraints. Our work leaves open the challenge of designing CPRFs for arbitrary purified circuit classes. Specifically, can we construct collusion-resistant CPRFs for any poly-size constraints where the only guarantee is that their supports are disjoint?

An affirmative answer from standard lattice assumptions would be a major breakthrough, as it would suggest that “zeroizing” attacks are the primary obstacles to collusion resistance. In Section 12, we provide a candidate CPRF for log-depth constraints which we conjecture to be secure

under purified corruption. We provide a proof under an extremely strong non-falsifiable lattice assumption, and pose the formal proof under a falsifiable lattice assumption (or cryptanalysis) of this candidate as an open problem.

3 Lattice Preliminaries

Notation. Let PPT denote probabilistic polynomial-time. We will use lowercase bold letters for vectors (e.g. $\mathbf{v}$), uppercase bold letters for matrices (e.g. $\mathbf{A}$) and assume all vectors are row vectors. The j^{th} row of a matrix $\mathbf{A}$ is denoted by $\mathbf{A}[j]$. For any integer $q \geq 2$, we let $\mathbb{Z}_q$ denote the ring of integers modulo q . We represent $\mathbb{Z}_q$ as integers in the range $(-q/2, q/2]$. For a vector $\mathbf{v}$, we let $\|\mathbf{v}\|$ denote its ℓ_2 norm and $\|\mathbf{v}\|_\infty$ denote its infinity norm. Similarly, for matrices $\|\cdot\|$ and $\|\cdot\|_\infty$ denote their ℓ_2 and infinity norms (respectively).

Throughout the paper, for any boolean function $f : \{0, 1\}^\ell \rightarrow \{0, 1\}$, we use $\text{Supp}(f)$ to denote its support. That is, $\text{Supp}(f) = \{x \in \{0, 1\}^\ell : f(x) = 1\}$. We denote the set of all positive integers up to n as $[n] := \{1, \dots, n\}$. For any finite set S , $x \leftarrow S$ denotes a uniformly random element x from the set S . Similarly, for any distribution $\mathcal{D}$, $x \leftarrow \mathcal{D}$ denotes an element x drawn from distribution $\mathcal{D}$. The distribution $\mathcal{D}^n$ is used to represent a distribution over vectors of n components, where each component is drawn independently from the distribution $\mathcal{D}$.

For two distributions X, Y , over a finite domain Ω , the statistical distance between X and Y is defined as $\text{SD}(X, Y) \stackrel{\text{def}}{=} \frac{1}{2} \sum_{\omega \in \Omega} |X(\omega) - Y(\omega)|$. A family of distributions $\mathcal{D}_1 = \{\mathcal{D}_1(\lambda)\}_\lambda$ and $\mathcal{D}_2 = \{\mathcal{D}_2(\lambda)\}_\lambda$, parameterized by security parameter λ , are said to be statistically indistinguishable, represented by $\mathcal{D}_1 \approx_s \mathcal{D}_2$, if there exists a negligible function $\text{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $\text{SD}(\mathcal{D}_1(\lambda), \mathcal{D}_2(\lambda)) \leq \text{negl}(\lambda)$. For a family of distributions $\mathcal{D} = \{\mathcal{D}(\lambda)\}_\lambda$ over the integers, and integers bounds $B = \{B(\lambda)\}_\lambda$, we say that $\mathcal{D}$ is B -bounded if $\Pr[|x| \leq B(\lambda) : x \leftarrow \mathcal{D}(\lambda)] = 1$. In words, a B -bounded distribution is supported only on the range $[-B, B]$. Below we state the “smudging” lemma as it appears in prior works.

Generalized Learning with Errors For our constructions, we will use the following generalized LWE assumption, and it directly follows the LWE assumption by a simple hybrid argument.

Assumption 3.1 (Generalized Learning with Errors). Let n, m and q be positive integers and χ a noise distribution over $\mathbb{Z}_q$. We say that the generalized LWE assumption $\text{GLWE}_{n,m,q,\chi}$ holds if for any PPT adversary $\mathcal{A}$, there exists a negligible function $\text{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $q = q(\lambda)$, $n = n(\lambda)$, $\eta = \eta(\lambda)$, $\phi = \phi(\lambda)$ and $\chi = \chi(\lambda)$, $\text{Adv}_{\mathcal{A}}^{\text{GLWE}}(\lambda) \leq \text{negl}(\lambda)$, where

$$\text{Adv}_{\mathcal{A}}^{\text{GLWE}}(\lambda) = \Pr \left[\mathcal{A}^{O_{\mathbf{A}}} (1^\lambda) = 1 : \mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m} \right] - \Pr \left[\mathcal{A}^{O'} (1^\lambda = 1) \right],$$

and oracles $O_{\mathbf{A}}, O'$ are defined as follows: oracle $O_{\mathbf{A}}$ has $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ hardwired, and on each query it chooses $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, $\mathbf{e} \leftarrow \chi^m$ and outputs $(\mathbf{A}, \mathbf{s}^T \cdot \mathbf{A} + \mathbf{e}^T)$, and oracle O' (on each query) chooses $\mathbf{u} \leftarrow \mathbb{Z}_q^m$, and outputs $(\mathbf{A}, \mathbf{u}^T)$.

Min-Entropy and Randomness Extraction. The min-entropy of a random variable X is defined as $\mathbf{H}_\infty(X) \stackrel{\text{def}}{=} -\log_2(\max_x \Pr[X = x])$. Let $\text{SD}(X, Y)$ denote the statistical distance between two random variables X and Y . Below we state the Leftover Hash Lemma (LHL) from [HILL99, DRS04, DORS08].

Theorem 3.2. Let $\mathcal{H} = \{h : X \rightarrow Y\}_{h \in \mathcal{H}}$ be a universal hash family, then for any random variable W taking values in X , the following holds

$$\text{SD}((h, h(W)), (h, U_Y)) \leq \frac{1}{2} \sqrt{2^{-\mathbf{H}_\infty(W)} \cdot |Y|}.$$

We will use the following corollary, which follows from the Leftover Hash Lemma.

Corollary 3.3. Let $\ell > m \cdot n \log_2 q + \omega(\log n)$ and q a prime. Let $\mathbf{R}$ be an $k \times m$ matrix chosen as per distribution $\mathcal{R}$, where $k = k(n)$ is polynomial in n and $\mathbf{H}_\infty(\mathcal{R}) = \ell$. Let $\mathbf{A}$ and $\mathbf{B}$ be matrices chosen uniformly in $\mathbb{Z}_q n \times k$ and $\mathbb{Z}_q n \times m$, respectively. Then the statistical distance between the following distributions is negligible in n .

$$\{(\mathbf{A}, \mathbf{A} \cdot \mathbf{R})\} \approx_s \{(\mathbf{A}, \mathbf{B})\}$$

Lemma 3.4 (Smudging Lemma [AJL⁺12, Lemma 2.1, paraphrased]). Let B_1, B_2 be two polynomials over the integers and let $\mathcal{D} = \{\mathcal{D}(\lambda)\}_\lambda$ be any B_1 -bounded distribution family. Let $U = \{U(\lambda)\}_\lambda$ and $U(\lambda)$ denote the uniform distribution over integers $[-B_2(\lambda), B_2(\lambda)]$. The family of distributions $\mathcal{D}$ and U is statistically indistinguishable, $\mathcal{D} + U \approx_s U$, if there exists a negligible function $\text{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $B_1(\lambda)/B_2(\lambda) \leq \text{negl}(\lambda)$.

Truncated Discrete Gaussians. The truncated discrete Gaussian distribution over $\mathbb{Z}^m$ with parameter σ , denoted by $\tilde{\mathcal{D}}_{\mathbb{Z}^m, \sigma}$, is same as the discrete Gaussian distribution $\mathcal{D}_{\mathbb{Z}^m, \sigma}$ except it outputs 0 whenever the ℓ_∞ norm exceeds $\sqrt{m} \cdot \sigma$. Note that, by definition, $\tilde{\mathcal{D}}_{\mathbb{Z}^m, \sigma}$ is $\sqrt{m} \cdot \sigma$ -bounded. Also, by the above lemma we get that $\tilde{\mathcal{D}}_{\mathbb{Z}^m, \sigma} \approx_s \mathcal{D}_{\mathbb{Z}^m, \sigma}$.

3.1 Lattice Evaluation and Homomorphic Encryption of Matrix Functions

We start by introducing the lattice evaluation that was developed by the FHE and ABE schemes [GSW13, BGG⁺14].

Theorem 3.5. Given n, q, ℓ and $m = \Omega(n \log q)$, a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times \ell m}$, and a binary function $f : \{0, 1\}^\ell \rightarrow \{0, 1\}$ of depth d , there exists efficient deterministic algorithms EvalF and EvalFX such that for all $\mathbf{x} \in \{0, 1\}^\ell$, $\mathbf{H}_f = \text{EvalF}(f, \mathbf{A})$, and $\mathbf{H}_{f, \mathbf{x}} = \text{EvalFX}(f, \mathbf{x}, \mathbf{A})$, it satisfies that

$$(\mathbf{A} - \mathbf{x} \otimes \mathbf{G}) \cdot \mathbf{H}_{f, \mathbf{x}} = \mathbf{A}_f - f(\mathbf{x}) \otimes \mathbf{G} \tag{1}$$

where $\mathbf{A}_f = \mathbf{A} \cdot \mathbf{H}_f$, $\mathbf{H}_f \in \mathbb{Z}_q^{\ell m \times m}$, $\mathbf{A}_f \in \mathbb{Z}_q^{n \times m}$, $\mathbf{G} \in \mathbb{Z}_q^{n \times m}$ is the gadget matrix, and $\|\mathbf{H}_f\|_\infty, \|\mathbf{H}_{f, \mathbf{x}}\|_\infty \leq m^{O(d)}$.

Extending to the evaluation of matrix functions In [BTVW17], Brakerski et al. considers the following variant of the above lattice evaluation: Given n, m, q , then for $i \in [n], j \in [m]$ and $t = (i, j)$, we define matrix $\mathbf{E}_{t, \ell} \in \mathbb{Z}_q^{n \times m}$ as the matrix where all the entries of $\mathbf{E}_{t, \ell}$ are zeros except for the entry (i, j) , and such entry has value $2^{\ell-1}$. Then, given matrix $\mathbf{X}$ such that $x_{t, \ell}$ is the ℓ -th bit of entry t of $\mathbf{X}$, we have:

$$\mathbf{X} = \sum_{t, \ell} x_{t, \ell} \cdot \mathbf{E}_{t, \ell}.$$

Next, given function $f : \mathbf{x} \rightarrow \mathbf{X}_f$ ($\mathbf{x} \in \{0,1\}^\ell$, $\mathbf{X}_f \in \mathbb{Z}_q^{n \times m}$), let function $f_{t,\ell} : \mathbf{x} \rightarrow \{0,1\}$ be the binary function that outputs the ℓ -th bit of entry t of $\mathbf{X}_f$. Then for $\mathbf{A} \in \mathbb{Z}_q^{\ell m}$, let $\mathbf{H}_{f,t,\ell} = \text{EvalF}(f_{t,\ell}, \mathbf{A})$ and let $\mathbf{H}_{f,t,\ell,\mathbf{x}} = \text{EvalFX}(f_{t,\ell}, \mathbf{x}, \mathbf{A})$. Let

$$\mathbf{H}_f = \sum_{t,\ell} \mathbf{H}_{f,t,\ell} \cdot \mathbf{G}^{-1}(\mathbf{E}_{t,\ell}), \mathbf{H}_{f,\mathbf{x}} = \sum_{t,\ell} \mathbf{H}_{f,t,\ell,\mathbf{x}} \cdot \mathbf{G}^{-1}(\mathbf{E}_{t,\ell}).$$

Then, one could extend Eq. (1) as the following:

$$(\mathbf{A} - \mathbf{x} \otimes \mathbf{G}) \cdot \mathbf{H}_{f,\mathbf{x}} = \sum_{t,\ell} (\mathbf{A} - \mathbf{x} \otimes \mathbf{G}) \cdot \mathbf{H}_{f,t,\ell,\mathbf{x}} \cdot \mathbf{G}^{-1}(\mathbf{E}_{t,\ell}) = \sum_{t,\ell} (\mathbf{A} \cdot \mathbf{H}_{f,t,\ell} - f_{t,\ell}(\mathbf{x}) \cdot \mathbf{G}) \cdot \mathbf{G}^{-1}(\mathbf{E}_{t,\ell}) = \mathbf{A} \cdot \mathbf{H}_f - \mathbf{X}_f.$$

Assuming that function f is of depth d , then by Theorem 3.5, $\|\mathbf{H}_{f,t,\ell}\|_\infty, \|\mathbf{H}_{f,t,\ell,\mathbf{x}}\|_\infty \leq m^{O(d)}$. Thus, it also holds that $\|\mathbf{H}_f\|_\infty, \|\mathbf{H}_{f,\mathbf{x}}\|_\infty \leq m^{O(d)}$. Formally, we conclude the above using the following theorem:

Theorem 3.6 ([BTVW17], Section 4.1). Given n, q, ℓ and $m = \Omega(n \log q)$, a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times \ell m}$, and a function $f : \{0,1\}^\ell \rightarrow \mathbf{X}_f$ of depth d , $\mathbf{X}_f \in \mathbb{Z}_q^{n \times m}$, there exists efficient deterministic algorithms EvalF and EvalFX such that for all $\mathbf{x} \in \{0,1\}^\ell$, $\mathbf{H}_f = \text{EvalF}(f, \mathbf{A})$, and $\mathbf{H}_{f,\mathbf{x}} = \text{EvalFX}(f, \mathbf{x}, \mathbf{A})$, it satisfies that

$$(\mathbf{A} - \mathbf{x} \otimes \mathbf{G}) \cdot \mathbf{H}_{f,\mathbf{x}} = \mathbf{A}_f - \mathbf{X}_f,$$

where $\mathbf{A}_f = \mathbf{A} \cdot \mathbf{H}_f$, $\mathbf{H}_f \in \mathbb{Z}_q^{\ell m \times m}$, $\mathbf{A}_f \in \mathbb{Z}_q^{n \times m}$, $\mathbf{G} \in \mathbb{Z}_q^{n \times m}$ is the gadget matrix, and $\|\mathbf{H}_f\|_\infty, \|\mathbf{H}_{f,\mathbf{x}}\|_\infty \leq m^{O(d)}$.

Furthermore, we extend Theorem 3.6 to the [GSW13] fully homomorphic encryption scheme, which consists of algorithms (Setup, Enc, Eval, Dec). We additionally define algorithm MtrxEval and obtain Theorem 3.7, where the detailed analysis of the corollary can be found in Section A.

Corollary 3.7. Given $\lambda, q, n = \text{poly}(\lambda), m = \Omega(n \log q), \ell \in \mathbb{N}$, a function $f : \{0,1\}^\ell \rightarrow \mathbf{X}_f$ of depth d , $\mathbf{X}_f \in \mathbb{Z}_q^{n \times m}$, there exists efficient deterministic algorithms (Enc, Dec, MtrxEval) such that for all $\mathbf{x} \in \{0,1\}^\ell$, $(\text{sk}, \text{pk}) \leftarrow \text{Setup}(1^\lambda)$, $\text{ct}_i \leftarrow \text{Enc}(\text{pk}, \mathbf{x}[i])$ for all $i \in [\ell]$, $\text{ct} = \text{MtrxEval}(f, \text{ct}_1, \dots, \text{ct}_\ell)$, it satisfies that

$$\|[\mathbf{s}^T] - 1\| \cdot \text{ct} - [\mathbf{s}^T] - 1\| \cdot \mathbf{X}_f\|_\infty \leq \beta \cdot m^{O(d)},$$

where $\mathbf{s}$ is the secret key sk .

We additionally define function $\text{LastRow}(f, \text{ct}_1, \dots, \text{ct}_\ell)$ that outputs the last row of $\text{MtrxEval}(f, \text{ct}_1, \dots, \text{ct}_\ell)$ and function $\text{AllButLastRow}(f, \text{ct}_1, \dots, \text{ct}_\ell)$ that outputs such matrix except for the last row.

4 Cryptographic Preliminaries & Definitions

In this section, we recall the notion of constrained PRFs [BW13, KPTZ13, BGI14] and 2-sided predicate encryption/functional encryption [BW07, KSW08, BSW11] with full collusion resistance.

4.1 Constrained Pseudorandom Functions with Constraint Privacy

We begin by providing the syntax for a constrained pseudorandom function (CPRF). Consider $\{\mathcal{C}_\ell\}_{\ell \in \mathbb{N}}$ denote an ensemble of circuit classes, where $\mathcal{C}_\ell$ consists of circuits that read ℓ bits of input. A CPRF for constraint class $\{\mathcal{C}_\ell\}_\ell$ consists of the following four algorithms:

$\text{Setup}(1^\lambda, 1^\ell) \rightarrow \text{msk}$. The setup takes the security parameter λ and input length ℓ as inputs, and outputs a master secret key msk .

$\text{Eval}(\text{msk}, x) \rightarrow r$. The master key evaluation algorithm takes as input the msk and an ℓ -bit input x , and it generates the output r .

$\text{Constrain}(\text{msk}, C) \rightarrow \text{sk}_C$. The constrain algorithm takes as input the msk and a constraint circuit $C \in \mathcal{C}_\ell$, and outputs a constrained key sk_C .

$\text{CEval}(\text{sk}_C, x) \rightarrow r$. The constrained evaluation algorithm takes as input a constrained key sk_C and an input x , and generates the output r .

We require a CPRF $\mathcal{F} = (\text{Setup}, \text{Eval}, \text{Constrain}, \text{CEval})$ to satisfy the following properties:

Correctness of constrained evaluation. For every $\lambda, \ell \in \mathbb{N}$, circuit $C \in \mathcal{C}_\ell$, let $\text{msk} \leftarrow \text{Setup}(1^\lambda, 1^\ell)$ and $\text{sk}_C \leftarrow \text{Constrain}(\text{msk}, C)$. A CPRF $\mathcal{F}$ satisfies correctness if there exists a negligible function $\text{negl}(\cdot)$ such that, for every $x \in \{0, 1\}^\ell$, $\Pr[\text{Eval}(\text{msk}, x) = \text{CEval}(\text{sk}_C, x) \mid C(x) = 1] \geq 1 - \text{negl}(\lambda)$, where the probability is taken over the randomness of the Setup and Constrain procedures.

Collusion-resistant constrained pseudorandomness. Informally, it states that an attacker can not distinguish the CPRF output from random on any input x^* as long as the corrupted constrained keys can not be used to evaluated on x^* . In this work, we consider the notion of fully-collusion-resistant security. It is formally defined as follows.

Definition 4.1 (Constrained Pseudorandomness). A CPRF $\mathcal{F}$ satisfies (collusion-resistant) constrained pseudorandomness if for any stateful admissible PPT adversary $\mathcal{A}$, there exists a negligible function $\text{negl}(\cdot)$ such that for all $\lambda, \ell \in \mathbb{N}$, the following holds:

$$\Pr \left[\mathcal{A}^{\text{Constrain}(\text{msk}, \cdot), \text{Eval}(\text{msk}, \cdot)}(r_b) = b : \begin{array}{l} \text{msk} \leftarrow \text{Setup}(1^\lambda, 1^\ell); b \leftarrow \{0, 1\} \\ x^* \leftarrow \mathcal{A}^{\text{Constrain}(\text{msk}, \cdot), \text{Eval}(\text{msk}, \cdot)}(1^\lambda, 1^\ell) \\ r_0 = \text{Eval}(\text{msk}, x^*); r_1 \leftarrow \{0, 1\}^{|r_0|} \end{array} \right] \leq 1/2 + \text{negl}(\lambda),$$

where $\mathcal{A}$ is an admissible adversary if: (i) $C_i(x^*) = 0$ for every circuit C_i that $\mathcal{A}$ queries to the Constrain oracle, and (ii) x^* is not sent to the Eval oracle.

Collusion-resistant constraint hiding. Informally, constraint hiding security (also called constrained privacy) states that an attacker can not learn non-trivial information about the constrain circuit C from it corresponding constrained key sk_C . As above, we again consider general collusion resistant attackers that can corrupt more than one constrained key. As discussed earlier, this has to be carefully defined. It is formally defined as follows.

Definition 4.2 (Constraint Privacy). A CPRF $\mathcal{F}$ satisfies (collusion-resistant) constrained privacy if for any stateful admissible PPT adversary $\mathcal{A}$, there exists a negligible function $\text{negl}(\cdot)$ such that for all $\lambda, \ell \in \mathbb{N}$, the following holds:

$$\Pr \left[\mathcal{A}^{O_b(\text{msk}, \cdot, \cdot), \text{Eval}(\text{msk}, \cdot)}(1^\lambda, 1^\ell) = b : \text{msk} \leftarrow \text{Setup}(1^\lambda, 1^\ell); b \leftarrow \{0, 1\} \right] \leq 1/2 + \text{negl}(\lambda),$$

where $\mathcal{A}$ sends two circuits $C_{i,0}, C_{i,1}$ in each query to oracle O_b , and O_b answers it as $\text{Constrain}(\text{msk}, C_{i,b})$. That is, O_b generates the constrained key for the b -th circuit in each query. Moreover, $\mathcal{A}$ is an admissible adversary if:

1. for every (distinct) pair of constrain queries $(C_{i,0}, C_{i,1})$ and $(C_{j,0}, C_{j,1})$, we have that $\text{Supp}(C_{i,0}) \cap \text{Supp}(C_{j,0}) = \text{Supp}(C_{i,1}) \cap \text{Supp}(C_{j,1})$, and
2. every evaluation query x is outside the support of all queried circuits. That is, every query $x \notin \bigcup_{i,b} \text{Supp}(C_{i,b})$.³

Remark 4.3 (Indistinguishability vs. simulation-based security.). Although the above definition only consider an indistinguishability-based notion for constraint privacy, it can be easily generalized to simulation security. While prior works [CC17] have demonstrated strong barriers to designing collusion-resistant private CPRFs with simulation security due to the known implication to program obfuscation [BGI⁺01], our CPRFs do not imply program obfuscation and thus satisfy simulation security. For the sake of simplicity, we stick to the above indistinguishability-based definition, but highlight that our CPRFs are simulation-secure.

4.2 Secret-Key Predicate Encryption with Predicate Privacy

A secret-key predicate encryption (SK-PE) scheme, for set of attribute spaces $\mathcal{X} = \{\mathcal{X}_n\}_{n \in \mathbb{N}}$, predicate classes $\mathcal{C} = \{\mathcal{C}_n\}_{n \in \mathbb{N}}$ and message spaces $\mathcal{M} = \{\mathcal{M}_n\}_{n \in \mathbb{N}}$, consists of four polytime algorithms (Setup, Enc, KeyGen, Dec) with the following syntax:

Setup($1^\lambda, 1^n$) $\rightarrow$ **msk**. The setup algorithm takes as input the security parameter λ and a functionality index n , and outputs the master secret key **msk**.

Enc(**msk**, x, m) $\rightarrow$ **ct**. The encryption algorithm takes as input master key **msk**, an attribute $x \in \mathcal{X}_n$ and a message $m \in \mathcal{M}_n$. It outputs a ciphertext **ct**.

KeyGen(**msk**, C) $\rightarrow$ **sk_C**. The key generation algorithm takes as input master secret key **msk** and a predicate $C \in \mathcal{C}_n$. It outputs a secret key **sk_C**.

Dec(**sk_C**, **ct**) $\rightarrow m$ **or** $\perp$. The decryption algorithm takes as input a secret key **sk_C** and a ciphertext **ct**. It outputs either a message $m \in \mathcal{M}_n$ or a special symbol $\perp$.

We require an SK-PE scheme $\text{PE} = (\text{Setup}, \text{Enc}, \text{KeyGen}, \text{Dec})$ to satisfy the following properties:

Correctness. An SK-PE scheme PE is said to be correct if there exists a negligible function $\text{negl}(\cdot)$ such that for all $\lambda, n \in \mathbb{N}$, for all $x \in \mathcal{X}_n$, $C \in \mathcal{C}_n$, $m \in \mathcal{M}_n$, the following holds

$$C(x) = 1 \Rightarrow \Pr \left[\text{Dec}(\text{sk}_C, \text{ct}) = m : \begin{array}{l} \text{msk} \leftarrow \text{Setup}(1^\lambda, 1^n); \\ \text{sk}_C \leftarrow \text{KeyGen}(\text{msk}, C); \text{ct} \leftarrow \text{Enc}(\text{msk}, x, m) \end{array} \right] \geq 1 - \text{negl}(\lambda).$$

2-sided security. The standard notion of security for a predicate encryption scheme is that of 2-sided security (or full attribute hiding) [BW07, KSW08, GKW17a]. Informally, it captures two-sided threats– (i) a ciphertext must hide, *both*, message and attribute as long as none of the corrupted keys satisfy the attribute, and (ii) even when a corrupted key can decrypt the challenge ciphertext, ciphertext must hide all non-trivial information about the attribute. It is formally defined as follows.

³One could consider attackers that make evaluation queries on inputs that lie in the intersection of the support of the queried circuits too. We can handle such queries, but do not include these in the definition for simplicity.

Definition 4.4 (2-Sided Security). An SK-PE scheme PE is said to be (fully) 2-sided secure if for every stateful admissible PPT adversary $\mathcal{A}$, there exists a negligible function $\text{negl}(\cdot)$, such that for every $\lambda \in \mathbb{N}$ the following holds:

$$\Pr \left[\mathcal{A}^{\text{KeyGen}(\text{msk}, \cdot), \text{Enc}(\text{msk}, \cdot, \cdot)}(\text{ct}) = b : \begin{array}{l} 1^n \leftarrow \mathcal{A}(1^\lambda); \text{msk} \leftarrow \text{Setup}(1^\lambda, 1^n) \\ ((m_0, m_1), (x_0, x_1)) \leftarrow \mathcal{A}^{\text{KeyGen}(\text{msk}, \cdot), \text{Enc}(\text{msk}, \cdot, \cdot)} \\ b \leftarrow \{0, 1\}; \text{ct} \leftarrow \text{Enc}(\text{msk}, x_b, m_b) \end{array} \right] \leq \frac{1}{2} + \text{negl}(\lambda)$$

where $\mathcal{A}$ is an admissible adversary if: (a) *either* every predicate query C , made by adversary $\mathcal{A}$ to the **KeyGen** oracle, satisfies the condition that $C(x_0) = C(x_1) = 0$, *or* (b) $m_0 = m_1$ and every predicate query C made by $\mathcal{A}$ is such that $C(x_0) = C(x_1)$.

Predicate privacy. Additionally, we also consider the notion of predicate privacy which states that a secret key must not reveal the associated predicate. Formally, it is defined as follows.

Definition 4.5 (Predicate Privacy). An SK-PE scheme PE is said satisfy predicate privacy if for every stateful admissible PPT adversary $\mathcal{A}$, there exists a negligible function $\text{negl}(\cdot)$, such that for every $\lambda \in \mathbb{N}$ the following holds:

$$\Pr \left[\mathcal{A}^{O_b(\text{msk}, \cdot, \cdot), \text{Enc}(\text{msk}, \cdot, \cdot)} = b : \begin{array}{l} 1^n \leftarrow \mathcal{A}(1^\lambda); b \leftarrow \{0, 1\} \\ \text{msk} \leftarrow \text{Setup}(1^\lambda, 1^n) \end{array} \right] \leq \frac{1}{2} + \text{negl}(\lambda)$$

where $\mathcal{A}$ sends two circuits $C_{i,0}, C_{i,1}$ in each query to oracle O_b , and O_b answers it as $\text{KeyGen}(\text{msk}, C_{i,b})$. That is, O_b generates the predicate key for the b -th circuit. Moreover, $\mathcal{A}$ is an admissible adversary if $C_{i,0}(x) = C_{i,1}(x)$ for every attribute x that was queried by $\mathcal{A}$ to the **Enc** oracle.

Remark 4.6 (Secret-key functional encryption). We remark that the above notion of 2-sided (secret-key) predicate encryption implies (secret-key) functional encryption. By setting the plain-text message m to be 1, this would give a secret-key FE scheme with appropriate security.

5 Purifying Functionality: CPRFs and PE

In this work, we put forth the concept of *purified functionality* in functional cryptography. As discussed earlier, the community currently struggles with designing and proving security of many cryptographic primitives under general adversarial attack strategies. For instance, prior works in CPRFs and PE typically only focus on 1-key security. Our contribution is to get around this shortcoming by proposing a new notion of *purification* across different cryptographic primitives. As our main case study, we start with CPRFs and later extend it to PE as well.

5.1 Collusion-Resistant CPRFs for Purified Constraints

In this section, we recall the notion of constrained PRFs [BW13, KPTZ13, BGI14] and 2-sided predicate encryption/functional encryption [BW07, KSW08, BSW11] with full collusion resistance.

We begin by providing the syntax for a constrained pseudorandom function (CPRF). Consider $\{\mathcal{C}_\ell\}_{\ell \in \mathbb{N}}$ denote an ensemble of circuit classes, where $\mathcal{C}_\ell$ consists of circuits that read ℓ bits of input. A CPRF for constraint class $\{\mathcal{C}_\ell\}_\ell$ consists of the following four algorithms:

Setup $(1^\lambda, 1^\ell) \rightarrow \text{msk}$. The setup takes the security parameter λ and input length ℓ as inputs, and outputs a master secret key msk .

$\text{Eval}(\text{msk}, x) \rightarrow r$. The master key evaluation algorithm takes as input the msk and an ℓ -bit input x , and it generates the output r .

$\text{Constrain}(\text{msk}, C) \rightarrow \text{sk}_C$. The constrain algorithm takes as input the msk and a constraint circuit $C \in \mathcal{C}_\ell$, and outputs a constrained key sk_C .

$\text{CEval}(\text{sk}_C, x) \rightarrow r$. The constrained evaluation algorithm takes as input a constrained key sk_C and an input x , and generates the output r .

We require a CPRF $\mathcal{F} = (\text{Setup}, \text{Eval}, \text{Constrain}, \text{CEval})$ to satisfy the following properties:

Correctness of constrained evaluation. For every $\lambda, \ell \in \mathbb{N}$, circuit $C \in \mathcal{C}_\ell$, let $\text{msk} \leftarrow \text{Setup}(1^\lambda, 1^\ell)$ and $\text{sk}_C \leftarrow \text{Constrain}(\text{msk}, C)$. A CPRF $\mathcal{F}$ satisfies correctness if there exists a negligible function $\text{negl}(\cdot)$ such that, for every $x \in \{0, 1\}^\ell$, $\Pr[\text{Eval}(\text{msk}, x) = \text{CEval}(\text{sk}_C, x) \mid C(x) = 1] \geq 1 - \text{negl}(\lambda)$, where the probability is taken over the randomness of the Setup and Constrain procedures.

Informally, a *purified* CPRF is defined exactly as a regular CPRF (see Section 4.1). That is, it is described using the same set of four algorithms with identical correctness guarantees. The main difference is that a purified CPRF can guarantee (full) collusion-resistant constrained pseudorandomness and constraint hiding as long as the constrained keys corrupted by an attacker correspond to a *purified class of constraints*. Here by a purified class, we mean that no two constrained keys can be used to evaluate the PRF on the same point. That is, the support for corrupted constrained keys do not overlap. As we demonstrate later via multiple constructions for purified CPRFs with provable security, this purification process enables going beyond the collusion-resistance barrier in CPRFs. Next, we formally introduce the notion of purified constraints, and later incorporate this with CPRFs.

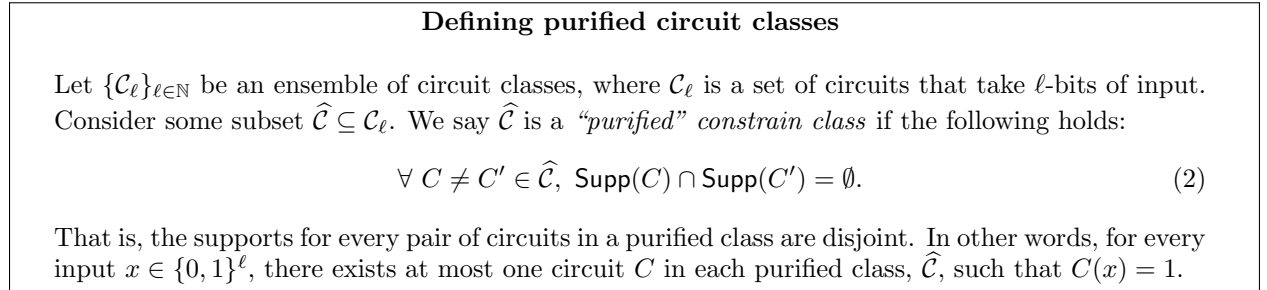


Figure 1: Purified Circuit Classes

Under the above abstraction, we present a new notion of CPRF security w.r.t. purified classes.

Collusion-resistant constrained pseudorandomness. Informally, it states that an attacker can not distinguish the CPRF output from random on any input x^* as long as the corrupted constrained keys can not be used to evaluated on x^* . In this work, we consider the notion of purified constrained pseudorandomness, which further requires that all constraint queries made by the adversary belong to a purified class. It is formally defined as follows.

Definition 5.1 (*Purified Constrained Pseudorandomness*). A CPRF $\mathcal{F}$ satisfies (collusion-resistant) constrained pseudorandomness *for purified constraints* if for every admissible PPT adversary $\mathcal{A}$,

there exists a negligible function $\text{negl}(\cdot)$ such that for all $\lambda, \ell \in \mathbb{N}$, the following holds:

$$\Pr \left[\mathcal{A}^{\text{Constrain}(\text{msk}, \cdot), \text{Eval}(\text{msk}, \cdot)}(r_b) = b : \begin{array}{l} \text{msk} \leftarrow \text{Setup}(1^\lambda, 1^\ell); b \leftarrow \{0, 1\} \\ x^* \leftarrow \mathcal{A}^{\text{Constrain}(\text{msk}, \cdot), \text{Eval}(\text{msk}, \cdot)}(1^\lambda, 1^\ell) \\ r_0 = \text{Eval}(\text{msk}, x^*); r_1 \leftarrow \{0, 1\}^{|r_0|} \end{array} \right] \leq 1/2 + \text{negl}(\lambda),$$

where $\mathcal{A}$ is an admissible adversary if: (i) $C_i(x^*) = 0$ for every circuit C_i that $\mathcal{A}$ queries to the **Constrain** oracle, (ii) x^* is not sent to the **Eval** oracle, and (iii) if $\mathcal{C}_{\mathcal{A}} = \{C_i\}_i$ denotes all circuits queried by $\mathcal{A}$ to the **Constrain** oracle, $\mathcal{C}_{\mathcal{A}}$ must be a purified constrain class as per Eq. (2) for $\mathcal{A}$ to be admissible.

Collusion-resistant constraint hiding. Informally, constraint hiding security (also called constrained privacy) states that an attacker can not learn non-trivial information about the constrain circuit C from its corresponding constrained key sk_C . As above, we again consider general collusion resistant attackers that can corrupt more than one constrained key. In the purified variant, we additionally require that both query families $\{C_{i,0}\}_i, \{C_{i,1}\}_i$ are purified classes, where these are the two circuit families submitted by $\mathcal{A}$ in its challenge queries. As discussed earlier, this has to be carefully defined. It is formally defined as follows.

Definition 5.2 (*Purified Constraint Privacy*). A CPRF $\mathcal{F}$ satisfies (collusion-resistant) constrained privacy for purified constraints if for any stateful admissible PPT adversary $\mathcal{A}$, there exists a negligible function $\text{negl}(\cdot)$ such that for all $\lambda, \ell \in \mathbb{N}$, the following holds:

$$\Pr \left[\mathcal{A}^{O_b(\text{msk}, \cdot, \cdot), \text{Eval}(\text{msk}, \cdot)}(1^\lambda, 1^\ell) = b : \text{msk} \leftarrow \text{Setup}(1^\lambda, 1^\ell); b \leftarrow \{0, 1\} \right] \leq 1/2 + \text{negl}(\lambda),$$

where $\mathcal{A}$ sends two circuits $C_{i,0}, C_{i,1}$ in each query to oracle O_b , and O_b answers it as $\text{Constrain}(\text{msk}, C_{i,b})$. That is, O_b generates the constrained key for the b -th circuit in each query. Moreover, $\mathcal{A}$ is an admissible adversary if:

1. for every (distinct) pair of constrain queries $(C_{i,0}, C_{i,1})$ and $(C_{j,0}, C_{j,1})$, we have that $\text{Supp}(C_{i,0}) \cap \text{Supp}(C_{j,0}) = \text{Supp}(C_{i,1}) \cap \text{Supp}(C_{j,1})$, and
2. every evaluation query x is outside the support of all queried circuits. That is, every query $x \notin \bigcup_{i,b} \text{Supp}(C_{i,b})$.⁴
3. $\{C_{i,0}\}_i$ and $\{C_{i,1}\}_i$ must both be purified constrain classes, individually, as per Eq. (2).

Remark 5.3 (Indistinguishability vs. simulation-based security.). Although the above definition only considers an indistinguishability-based notion for constraint privacy, it can be easily generalized to simulation security. While prior works [CC17] have demonstrated strong barriers to designing collusion-resistant private CPRFs with simulation security due to the known implication to program obfuscation [BGI⁺01], our CPRFs do not imply program obfuscation and thus satisfy simulation security. For the sake of simplicity, we stick to the above indistinguishability-based definition, but highlight that our CPRFs are simulation-secure.

⁴One could consider attackers that make evaluation queries on inputs that lie in the intersection of the support of the queried circuits too. We can handle such queries, but do not include these in the definition for simplicity.

5.2 Predicate Encryption for Purified Predicates

Similar to CPRFs, we propose the notion of purified predicates in the context of (secret-key) predicate encryption. Informally, a *purified* (secret-key) PE system is defined exactly as a regular PE system (see Section 4.2), i.e. with same syntax and correctness. As for purified CPRFs, we define predicate privacy for PE in a similar manner, which is the predicate keys corrupted by an attacker must correspond to a *purified class of predicates* (as in Fig. 1).

However, unlike purified CPRFs, we define the new purified security properties for PE in a slightly stronger way. A purified PE systems guarantees 2-sided security as long as the attacker either makes arbitrary unsatisfying predicate queries (i.e., do not require queries to correspond to a purified circuit class), or if it makes a satisfying predicate query, then all predicates correspond to a purified circuit class (as in Fig. 1). Thus, a purified PE provides *strictly stronger* security when compared to a regular 1-sided predicate encryption system [GKW17b]. We explain it formally below.

Definition 5.4 (*Purified 2-Sided Security*). An SK-PE scheme PE is said to be (fully) 2-sided secure *for purified predicates* if every admissible PPT adversary $\mathcal{A}$ has negligible advantage in the 2-sided security game as defined in Theorem 4.4, where $\mathcal{A}$ must satisfy the following *additional admissibility condition*: let $\mathcal{C}_{\mathcal{A}} = \{C_i\}_i$ denote the set of all circuits queried by $\mathcal{A}$ to the KeyGen oracle; if $C_i(x_0) = C_i(x_1) = 1$ for some queried circuit C_i , then $\mathcal{C}_{\mathcal{A}}$ must be a purified circuit class as per Eq. (2) for $\mathcal{A}$ to be admissible.

In words, the admissibility condition is only changed for ‘one-side’. That is, the corrupted keys must correspond to a purified circuit class *only* when an attacker corrupts at least one satisfying secret key.

Definition 5.5 (*Purified Predicate Privacy*). An SK-PE scheme PE is said satisfy predicate privacy *for purified predicates* if every admissible PPT adversary $\mathcal{A}$ has negligible advantage in the predicate privacy security game as defined in Theorem 4.5, where $\mathcal{A}$ must satisfy the following *additional admissibility condition*: let $\{(C_{i,0}, C_{i,1})\}_i$ denote the set of circuit-pairs queried by $\mathcal{A}$ to the challenge key-gen oracle O_b and $\mathcal{C}_{\mathcal{A},b} = \{C_{i,b}\}_i$; $\mathcal{C}_{\mathcal{A},0}$ and $\mathcal{C}_{\mathcal{A},1}$ must both be purified constrain classes, individually, as per Eq. (2) for $\mathcal{A}$ to be admissible.

5.3 Candidate Purified Circuit Families

In this section, we describe three families of purified circuits with increasing expressivity. Looking ahead, we provide new constructions for CPRFs and PE for all three purified circuit families – compute-&-compare, predicate range, and purified NC¹. Below we formally describe them.

- (1) **Compute-&-compare** – Each circuit class in this family is associated with a function $f : \mathcal{X} \rightarrow \mathcal{Y}$, with domain and co-domain $\mathcal{X}$ and $\mathcal{Y}$. That is, each class $\widehat{\mathcal{C}}_{\text{CC},f}$ is annotated with a function f , where f can be represented as an arithmetic circuit of polynomial size. For any function f and element y in its co-domain $\mathcal{Y}$, let $\text{CC}[f, y]$ correspond to the ‘compute-&-compare’ circuit that, on input $x \in \mathcal{X}$, outputs 1 if and only if $f(x) = y$. In short,

$$\forall x \in \mathcal{X}, \quad \text{CC}[f, y](x) = (f(x) \stackrel{?}{=} y).$$

Consider the following class of efficiently enumerable circuits – $\text{CC}_f = \{\text{CC}[f, y] : y \in \mathcal{Y}\}$. We call any subset of CC_f as a compute-&-compare family of circuits. It is straightforward to

check that every $\hat{\mathcal{C}} \subseteq \text{CC}_f$ will indeed be a purified circuit family class as per Fig. 1. This is because for any two distinct circuits in $\mathcal{C}, \mathcal{C}' \in \text{CC}_f$, where $\mathcal{C} = \text{CC}[f, y]$ and $\mathcal{C}' = \text{CC}[f, y']$ for some $y \neq y' \in \mathcal{Y}$, we have that $\text{Supp}(\mathcal{C}) = \{x : x \in f^{-1}(y)\}$ and $\text{Supp}(\mathcal{C}') = \{x : x \in f^{-1}(y')\}$. Since f is a function, thus $\text{Supp}(\mathcal{C}) \cap \text{Supp}(\mathcal{C}')$ will always be $\emptyset$ as long as $\mathcal{C} \neq \mathcal{C}'$.

Popular examples of compute-&-compare . The most basic example of a compute-&-compare class would be if the function f is the identity function, thus $\mathcal{X} = \mathcal{Y}$. Thus, one could view ID-based predicates [Sha84, BF01, Coc01] as the simplest compute-&-compare circuit class. More generally, compute-&-compare predicates have also been considered in the context of code obfuscation, popularly known as lockable obfuscation [GKW17b, WZ17]. Moreover, one could also write down various order-revealing/preserving predicates (used in various searchable encryption systems) [AKSX04, BCLO09, BCO11, BLR⁺15] as special forms of compute-&-compare predicates.

(2) Predicated range – This class is a generalization of the compute-&-compare class, wherein every circuit class is again associated with a function $f : \mathcal{X} \rightarrow \mathcal{Y}$. The difference is that instead of a comparison check to a single element, each circuit checks whether the function output lies in a pre-specified range. Concretely, let $y_1 < y_2 \in \mathcal{Y}$ be two co-domain elements.⁵ Then a predicate range circuit, denoted as $\text{CR}[f, y_1, y_2]$, is defined as:

$$\forall x \in \mathcal{X}, \quad \text{CR}[f, y_1, y_2](x) = (f(x) \stackrel{?}{\in} [y_1, y_2]).$$

For any function f , and any sequence of increasingly ordered pairs $(y_1, y_2), (y_3, y_4), \dots$ (where $y_i < y_{i+1}$ and $y_i \in Y$ for every i), the following class of circuits $\{\text{CR}[f, y_1, y_2], \text{CR}[f, y_3, y_4], \dots\}$ is referred to as a ‘*predicated range*’ circuit class. In words, a predicated range circuit class is such that any two predicated range circuits (in that class) are associated with two non-overlapping range subsets of the co-domain $\mathcal{Y}$. Again, it is straightforward to check that every predicated range circuit class is indeed a purified circuit family class as per Fig. 1. $\mathcal{C} = \text{CR}[f, y_1, y_2]$ and $\mathcal{C}' = \text{CR}[f, y_3, y_4]$ have disjoint supports if $y_1, y_2 < y_3, y_4$. This is because f is a function, and $\text{Supp}(\mathcal{C}) = \{x : x \in f^{-1}([y_1, y_2])\}$, $\text{Supp}(\mathcal{C}') = \{x : x \in f^{-1}([y_3, y_4])\}$ do not intersect.

Popular examples of predicated range. In addition to naturally generalizing the compute-&-compare class, it captures various commonly studied circuit classes. For example, it captures fuzzy ID-based predicates [SW05] if we consider f to be an identity function. Moreover, this also captures range-based search predicates as considered in many searchable encryption systems [AKSX04, BCLO09, BCO11, BLR⁺15]. Broadly, these can also be further extended to consider hierarchical predicates [HL02, GS02, GVW13, BGG⁺14], where a given predicated range circuit could be restricted to define another predicated range circuit or a compute-&-compare circuit.

(3) Purified NC¹ – Finally, we start with the class of arbitrary log-depth circuits, NC¹, and define a sub-class called purified NC¹ circuits, called Pure-NC¹. Rather than define them in a structured manner, we say a circuit class $\hat{\mathcal{C}} \in \text{Pure-NC}^1$ (i.e., $\hat{\mathcal{C}}$ is a purified log-depth circuit class) if the pairwise intersection of supports of two distinct circuits in $\hat{\mathcal{C}}$ is empty. Concretely,

$$\text{Pure-NC}^1 = \left\{ \hat{\mathcal{C}} : \hat{\mathcal{C}} \subset \text{NC}^1 \wedge \left(\forall \mathcal{C} \neq \mathcal{C}' \in \hat{\mathcal{C}}, \text{Supp}(\mathcal{C}) \cap \text{Supp}(\mathcal{C}') = \emptyset \right) \right\}$$

⁵We are assuming that $\mathcal{Y}$ is a totally ordered set.

Briefly, Pure-NC^1 is the broadest class of purified circuits that is a subset of *general* NC^1 circuits.

Note. We could also define arbitrary purified circuits by generalizing Pure-NC^1 while removing the log-depth constraint in the above.

6 Collusion-Resistant CPRFs for Compute-&-Compare Constraints

In this section, we provide our first construction. We construct collusion-resistant CPRFs for compute-&-compare constraints (see Section 5.3). Recall each compute-&-compare circuit $\text{CC}[f, y]$ is associated with a function f and co-domain element y . For any $\ell, \ell', L, d \in \mathbb{N}$, we construct CPRFs for depth- d arithmetic circuits that take ℓ -bits of input and output ℓ' -bits, and have a description length L . The tools required for this construction are basic preliminaries described in Section 3. We start by describing some useful parameters that will be useful for proving correctness and security of our CPRFs.

Let χ_0 and χ_1 be two truncated discrete gaussian distributions set as $\tilde{D}_{\mathbb{Z}, \sigma}$ and $\tilde{D}_{\mathbb{Z}, \sigma'}$, respectively. Fix any $\epsilon < 1/2$. Let n, m, q, σ, σ' be parameters with the following constraints:

- $n = \text{poly}(\lambda)$, $q \leq 2^{n^\epsilon}$, $m = \Omega(n \log q)$ (for LWE security)
- $\sigma' \geq \sigma \cdot m^{\ell'+1} \cdot 2^{\Omega(\lambda)}$ (for smudging)
- $\frac{q}{p} \geq \sigma' \cdot (L \cdot m^{\Omega(d)} + m^\ell) \cdot 2^{\Omega(\lambda)}$ (for correctness/smudging)

One possible parameter setting that satisfies the above requirement is as follows:

$$n = 2\lambda(d + \ell + \ell')/\epsilon, m = n \log q, p = 2, q = 2^{n^\epsilon}, \sigma = n, \sigma' = \sigma \cdot m^{\ell'+1} \cdot 2^\lambda.$$

Notation. We define function $\mathbf{A}(\mathbf{x})$ as $\mathbf{A}_{\mathbf{x}[1]} \cdot \prod_{i=2}^\ell \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{x}[i]})$, given matrices $\mathbf{A}_0, \mathbf{A}_1$, gadget matrix $\mathbf{G}$, and length ℓ of $\mathbf{x}$. In the following construction, $\lceil v \rceil$ means multiplying v by p/q and rounding to the nearest integer. Unless otherwise specified, these notations apply to the rest of this paper. We will now describe our CPRF algorithms.

Setup($1^\lambda, 1^\ell, 1^{\ell'}, 1^L, 1^d$): It takes as input security parameter λ , input length ℓ , output length ℓ' , description length L of constraint functions, and depth d . It samples $\mathbf{B}_1, \dots, \mathbf{B}_L \leftarrow \mathbb{Z}_q^{n \times m}$, $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, $\mathbf{A}_0, \mathbf{A}_1 \leftarrow \mathbb{Z}_q^{n \times m}$. It outputs $\text{pp} := (\mathbf{B}_1, \dots, \mathbf{B}_L, \mathbf{A}_0, \mathbf{A}_1)$, $\text{msk} := \mathbf{s}$.

Eval($\text{msk}, \mathbf{x}$): It takes as input master secret key msk and input $\mathbf{x}$. We define circuit $\mathcal{U}_{\mathbf{x}}$ as the circuit which takes as input a description of a function f and outputs $\mathbf{A}(f(\mathbf{x}))$.

$$\mathbf{H}_{\mathcal{U}_{\mathbf{x}}} := \text{EvalF}(\mathcal{U}_{\mathbf{x}}, \{\mathbf{B}_i\}_{i \in [L]}), \mathbf{B}_{\mathcal{U}_{\mathbf{x}}} := [\mathbf{B}_1 \mid \dots \mid \mathbf{B}_L] \cdot \mathbf{H}_{\mathcal{U}_{\mathbf{x}}}.$$

The PRF output is $\lceil \mathbf{s}^T \cdot \mathbf{B}_{\mathcal{U}_{\mathbf{x}}} \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \rceil$.

Constrain($\text{msk}, C_{f, \mathbf{y}}$): It takes as input msk and a compute-&-compare constraint circuit $C_{f, \mathbf{y}}$. It samples $\mathbf{e} \leftarrow \chi_1^{Lm}$ and $\mathbf{e}_{\mathbf{y}} \leftarrow \chi_1^m$, and outputs

$$\text{sk} := (\mathbf{s}^T \cdot [\mathbf{B}_1 - f[1]\mathbf{G} \mid \dots \mid \mathbf{B}_L - f[L]\mathbf{G}] + \mathbf{e}^T, \mathbf{s}^T \cdot \mathbf{A}(\mathbf{y}) + \mathbf{e}_{\mathbf{y}}^T).$$

CEval($\text{sk}, \mathbf{x}$): Parse sk as $\mathbf{b}^T, \mathbf{c}_{\mathbf{y}}^T$. It computes $\mathbf{H}_{\mathcal{U}_{\mathbf{x}, f}} := \text{EvalFX}(\mathcal{U}_{\mathbf{x}}, f, \{\mathbf{B}_i\}_{i \in [L]})$. The PRF output is $\lceil (\mathbf{b}^T \cdot \mathbf{H}_{\mathcal{U}_{\mathbf{x}, f}} + \mathbf{c}_{\mathbf{y}}^T) \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \rceil$.

Master-key homomorphism. Recall that $\text{Eval}(\mathbf{s}, \mathbf{x}) = \lceil \mathbf{s}^T \cdot \mathbf{B}_{\mathcal{U}_{\mathbf{x}}} \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \rceil$, where $\mathbf{B}_{\mathcal{U}_{\mathbf{x}}}$ depends only on public parameters. For two master keys $\mathbf{s}_1, \mathbf{s}_2$, by linearity of matrix multiplication we have

$$\mathbf{s}_1^T \cdot \mathbf{B}_{\mathcal{U}_{\mathbf{x}}} \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) + \mathbf{s}_2^T \cdot \mathbf{B}_{\mathcal{U}_{\mathbf{x}}} \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) = (\mathbf{s}_1 + \mathbf{s}_2)^T \cdot \mathbf{B}_{\mathcal{U}_{\mathbf{x}}} \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})).$$

Therefore, before the rounding step, evaluation under $\mathbf{s}_1 + \mathbf{s}_2$ coincides exactly with the sum of evaluations under $\mathbf{s}_1$ and $\mathbf{s}_2$. The only deviation between $\text{Eval}(\mathbf{s}_1 + \mathbf{s}_2, \mathbf{x})$ and $\text{Eval}(\mathbf{s}_1, \mathbf{x}) + \text{Eval}(\mathbf{s}_2, \mathbf{x})$ comes from the final rounding operation. This rounding error can be analyzed in exactly the same manner as in the correctness proof. In particular, the magnitude of error is negligible compared to q/p , and so is the rounding discrepancy. Consequently, we obtain that

$$\text{Eval}(\mathbf{s}_1 + \mathbf{s}_2, \mathbf{x}) \approx \text{Eval}(\mathbf{s}_1, \mathbf{x}) + \text{Eval}(\mathbf{s}_2, \mathbf{x}),$$

which establishes almost key homomorphism for master keys.

Constrained-key homomorphism. For two master keys $\mathbf{s}_1, \mathbf{s}_2$ and the corresponding constrained keys sk_1, sk_2 for the same constraint, by adding the two keys gives $\text{sk}_1 + \text{sk}_2 = ((\mathbf{s}_1 + \mathbf{s}_2)^T \cdot [\mathbf{B}_1 - f[1]\mathbf{G} | \dots])$. Since CEval is linear in these two components before the final rounding step, the only deviation again comes from the noise terms. As in the correctness proof, the accumulated error is negligible compared to q/p under our parameter setting.

Adaptive security in the random oracle model. We note that the selective restriction in the above standard-model theorem can be removed in the random oracle model. Let $H : \mathcal{Y}_{\text{PRF}} \rightarrow \{0, 1\}^\lambda$ be modeled as a random oracle, where $\mathcal{Y}_{\text{PRF}}$ is the output space of the algebraic PRF value computed above. The ROM variant keeps Setup , Constrain , and CEval unchanged up to the computation of the original output, and only hashes the final value. Namely, if the adaptive CPRF evaluation algorithm computes $H(\lceil \mathbf{s}^T \cdot \mathbf{B}_{\mathcal{U}_{\mathbf{x}}} \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \rceil)$. The constrained evaluation algorithm is modified analogously: it first computes the same value as CEval and then applies H .

For an admissible challenge input $\mathbf{x}^*$, let $\mathbf{z}_{\mathbf{x}^*}$ denote the CPRF value computed before applying H . In the ROM proof, the simulator answers the challenge query by a fresh random string and programs H at $\mathbf{z}_{\mathbf{x}^*}$. This programming is consistent unless the adversary has already queried $\mathbf{z}_{\mathbf{x}^*}$ to H , or obtains the same oracle point through another evaluation query. Thus the only bad events are that the adversary queries $\mathbf{z}_{\mathbf{x}^*}$ to H , or finds some evaluation query $\mathbf{x} \neq \mathbf{x}^*$ such that $\mathbf{z}_{\mathbf{x}} = \mathbf{z}_{\mathbf{x}^*}$. The first event is ruled out by the underlying hidden-preimage property of the algebraic construction: for an admissible $\mathbf{x}^*$, the value $\mathbf{z}_{\mathbf{x}^*}$ cannot be computed from the public parameters, evaluation queries, and admissible constrained keys. The second event is ruled out by the standard collision argument for the underlying algebraic evaluation map. Hence, conditioned on these bad events not occurring, the value $H(\mathbf{z}_{\mathbf{x}^*})$ is unseen by the adversary and is distributed as a fresh random string. This gives the ROM adaptive-security upgrade via the standard output-hashing paradigm of [AMN⁺18].

6.1 Correctness

For the correctness property to hold, we need to prove that if $f(\mathbf{x}) = \mathbf{y}$, then with high probability it holds that $\text{Eval}(\text{msk}, \mathbf{x}) = \text{CEval}(\text{sk}, \mathbf{x})$ for $\text{msk} \leftarrow \text{Setup}(1^\lambda, 1^\ell, 1^{\ell'}, 1^L, 1^d)$, $\text{sk} \leftarrow \text{Constrain}(\text{msk}, f, \mathbf{y})$.

First by Theorem 3.6, we have $\|\mathbf{H}_{\mathcal{U}_{\mathbf{x}}}\|_{\infty}, \|\mathbf{H}_{\mathcal{U}_{\mathbf{x},f}}\|_{\infty} \leq m^{O(d)}$, and

$$[\mathbf{B}_1 - f[1]\mathbf{G} \mid \dots \mid \mathbf{B}_L - f[L]\mathbf{G}] \cdot \mathbf{H}_{\mathcal{U}_{\mathbf{x},f}} = [\mathbf{B}_1 \mid \dots \mid \mathbf{B}_L] \cdot \mathbf{H}_{\mathcal{U}_{\mathbf{x}}} - \mathbf{A}(f(\mathbf{x})).$$

We then rewrite the equation of $\text{CEval}(\text{sk}, \mathbf{x})$ and recall that $f(\mathbf{x}) = \mathbf{y}$:

$$\begin{aligned} \text{CEval}(\text{sk}, \mathbf{x}) &= \lceil ((\mathbf{s}^T \cdot [\mathbf{B}_1 - f[1]\mathbf{G} \mid \dots \mid \mathbf{B}_L - f[L]\mathbf{G}] + \mathbf{e}^T) \cdot \mathbf{H}_{\mathcal{U}_{\mathbf{x},f}} + \mathbf{s}^T \cdot \mathbf{A}(\mathbf{y}) + \mathbf{e}_{\mathbf{y}}^T) \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \rceil \\ &= \lceil (\mathbf{s}^T \cdot ([\mathbf{B}_1 - f[1]\mathbf{G} \mid \dots \mid \mathbf{B}_L - f[L]\mathbf{G}] \cdot \mathbf{H}_{\mathcal{U}_{\mathbf{x},f}} + \mathbf{A}(\mathbf{y})) + (\mathbf{e}^T \cdot \mathbf{H}_{\mathcal{U}_{\mathbf{x},f}} + \mathbf{e}_{\mathbf{y}}^T)) \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \rceil \\ &= \lceil \mathbf{s}^T \cdot [\mathbf{B}_1 \mid \dots \mid \mathbf{B}_L] \cdot \mathbf{H}_{\mathcal{U}_{\mathbf{x}}} \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) + (\mathbf{e}^T \cdot \mathbf{H}_{\mathcal{U}_{\mathbf{x},f}} + \mathbf{e}_{\mathbf{y}}^T) \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \rceil. \end{aligned}$$

The above equation differs from $\text{Eval}(\text{msk}, \mathbf{x})$ by the highlighted part:

$$\text{Eval}(\text{msk}, \mathbf{x}) = \lceil \mathbf{s}^T \cdot [\mathbf{B}_1 \mid \dots \mid \mathbf{B}_L] \cdot \mathbf{H}_{\mathcal{U}_{\mathbf{x}}} \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \rceil.$$

We analyze the norm bound of the highlighted expression. By our parameter setting and the norm bound of truncated discrete gaussian distribution, we have that $\|\mathbf{e}^T\|_{\infty}, \|\mathbf{e}_{\mathbf{y}}^T\|_{\infty} \leq \sigma' \cdot m$, thus

$$\|(\mathbf{e}^T \cdot \mathbf{H}_{\mathcal{U}_{\mathbf{x},f}} + \mathbf{e}_{\mathbf{y}}^T) \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x}))\|_{\infty} \leq (\sigma' \cdot m \cdot m^{O(d)} \cdot L \cdot m + \sigma' \cdot m) \cdot m \leq \sigma' \cdot L \cdot m^{O(d)} \ll q/p.$$

We conclude that $\text{Eval}(\text{msk}, \mathbf{x}) = \text{CEval}(\text{sk}, \mathbf{x})$ with high probability for $f(\mathbf{x}) = \mathbf{y}$.

6.2 Purified Constraint Pseudorandomness

Next, we prove that our CPRF satisfies *semi-adaptive purified constraint pseudorandomness*, where the attacker can corrupt any polynomials number of constraint keys as long as they correspond to the compute-&-compare class. (Refer to Section 5.1.) We point out that we only prove semi-adaptive security, which is identical to as defined Theorem 5.1, except we require the attacker to specify the function f at the beginning. For completeness, we formally define it below.

Definition 6.1 (Semi-Adaptive Purified Constrained Pseudorandomness). A CPRF $\mathcal{F}$ satisfies (collusion-resistant) semi-adaptive constrained pseudorandomness for compute-&-compare circuits, if for any stateful admissible PPT adversary $\mathcal{A}$, there exists a negligible function $\text{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, the following holds:

$$\Pr \left[\begin{array}{l} \mathcal{A}^{\text{Constrain}(\text{msk}, \cdot), \text{Eval}(\text{msk}, \cdot)}(r_b) = b : \\ \begin{array}{l} (\ell, \ell', L, d, f) \leftarrow \mathcal{A}(1^\lambda); b \leftarrow \{0, 1\} \\ \text{msk} \leftarrow \text{Setup}(1^\lambda, 1^\ell, 1^{\ell'}, 1^L, 1^d) \\ x^* \leftarrow \mathcal{A}^{\text{Constrain}(\text{msk}, \cdot), \text{Eval}(\text{msk}, \cdot)} \\ r_0 = \text{Eval}(\text{msk}, x^*); r_1 \leftarrow \{0, 1\}^{|r_0|} \end{array} \end{array} \right] \leq 1/2 + \text{negl}(\lambda),$$

where $\mathcal{A}$ is an admissible adversary if: (i) each C_i queried by $\mathcal{A}$ to the **Constrain** oracle is of the form $C_{f, \mathbf{y}_i}$ for some $\mathbf{y}_i$ and $C_i(x^*) = 0$ for every i , and (ii) x^* is not sent to the **Eval** oracle.

We refer to the above as the “semi-adaptive” experiment because the attacker only has to selectively choose function f , but it could still adaptively query constraint keys with respect to different function outputs after msk is set. By a standard complexity leveraging argument, this could be improved to standard adaptive security. We leave the problem of proving adaptive security without leveraging as an interesting open problem.

Formally, we prove the following.

Theorem 6.2. Assuming that $\text{GLWE}_{n,(L+2)m,q,\chi_0}$ (Theorem 3.1) holds, the above CPRF satisfies semi-adaptive constrained pseudorandomness for compute-&-compare circuits, as per Theorem 6.1.

The proof is describe using a sequence of hybrid games. First, we describe the hybrid games, and later prove indistinguishability of any two consecutive games. For ease of exposition, we introduce some useful notation for the rest of the proof.

Notations. We adopt the convention that for any binary vector $\mathbf{x}$ (or $\mathbf{y}$), the notation $\mathbf{x}[[i]]$ refers to the prefix containing the first i bits of $\mathbf{x}$. The notation $\mathbf{x}[i]$ refers to the i -th bit of $\mathbf{x}$. Unless otherwise specified, the notations apply to all binary vectors considered.

6.2.1 Sequence of Hybrid Games

In the sequence of the hybrids described below, Hybrid 0 will correspond to the original game. Hybrid 1 ensures that any constrained secret key sk will be independent from the description of function f during constrained key queries. Challenger of Hybrid 2 will generate constrained secret keys during evaluation and challenge queries, and will answer those queries following the process of the constrained evaluation instead of the original evaluation algorithm. Thus after Hybrid 2, all of the queries will be independent from the description of function f , and then Hybrid 3 will generate uniformly random constrained keys. Lastly, Hybrid 4 will answer the challenge query using random strings.

Hybrid 0: In this game, the challenger generates the public parameters and answers all the queries honestly.

1. The attacker outputs the constraining function f .
2. The challenger starts by running the setup algorithm. It samples $\mathbf{B}_1, \dots, \mathbf{B}_L \leftarrow \mathbb{Z}_q^{n \times m}$, $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, $\mathbf{A}_0, \mathbf{A}_1 \leftarrow \mathbb{Z}_q^{n \times m}$, $\mathbf{e} \leftarrow \chi_0^{Lm}$. The challenger outputs $\mathbf{B}_1, \dots, \mathbf{B}_L, \mathbf{A}_0, \mathbf{A}_1$. It samples a bit $b \leftarrow \{0, 1\}$.
3. The attacker will make polynomial evaluation and constrained key queries. Let $Q = \text{poly}(\lambda)$ be the number of the evaluation queries.
 - During each constrained key query, the attacker outputs $\mathbf{y}$. The challenger responds using the constraint key generation algorithm $\text{Constrain}(\text{msk}, C_{f,\mathbf{y}})$: It samples $\mathbf{e} \leftarrow \chi_1^{Lm}$, $\mathbf{e}_y \leftarrow \chi_1^m$, and outputs $\text{sk} := (\mathbf{s}^T \cdot [\mathbf{B}_1 - f[1]\mathbf{G}] \dots [\mathbf{B}_L - f[L]\mathbf{G}] + \mathbf{e}^T, \mathbf{s}^T \cdot \mathbf{A}(\mathbf{y}) + \mathbf{e}_y^T)$. The challenger adds $\mathbf{y}$ to set $\mathcal{Y}$.
 - During each evaluation query, the attacker outputs an input $\mathbf{x}$. The challenger responds with $\text{Eval}(\text{sk}, \mathbf{x})$: It computes $\mathbf{H}_{\mathcal{U}_{\mathbf{x}}} := \text{EvalF}(\mathcal{U}_{\mathbf{x}}, \{\mathbf{B}_i\}_{i \in L})$, $\mathbf{B}_{\mathcal{U}_{\mathbf{x}}} := [\mathbf{B}_1 | \dots | \mathbf{B}_L] \cdot \mathbf{H}_{\mathcal{U}_{\mathbf{x}}}$, and outputs $\lceil \mathbf{s}^T \cdot \mathbf{B}_{\mathcal{U}_{\mathbf{x}}} \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \rceil$. It adds $\mathbf{x}$ to set $\mathcal{X}$.
4. During the challenge query, the attacker outputs an input $\mathbf{x}^*$. If $\mathbf{x}^* \in \mathcal{X}$ or $f(\mathbf{x}^*) \in \mathcal{Y}$, the attacker fails and the game aborts. If $b = 0$, the challenger responds with $\text{Eval}(\text{sk}, \mathbf{x}^*)$: It computes $\mathbf{H}_{\mathcal{U}_{\mathbf{x}^*}} := \text{EvalF}(\mathcal{U}_{\mathbf{x}^*}, \{\mathbf{B}_i\}_{i \in L})$, $\mathbf{B}_{\mathcal{U}_{\mathbf{x}^*}} := [\mathbf{B}_1 | \dots | \mathbf{B}_L] \cdot \mathbf{H}_{\mathcal{U}_{\mathbf{x}^*}}$, and outputs $\lceil \mathbf{s}^T \cdot \mathbf{B}_{\mathcal{U}_{\mathbf{x}^*}} \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x}^*)) \rceil$. Otherwise the challenger samples $\mathbf{r}_{\mathbf{x}^*} \leftarrow \mathbb{Z}_q^m$ and outputs $\lceil \mathbf{r}_{\mathbf{x}^*}^T \rceil$.
5. It repeats step 3, except that if $\mathcal{A}$ queries $f(\mathbf{x}^*)$ during constrained key queries or $\mathbf{x}^*$ during evaluation queries, the attacker immediately fails and the game aborts.
6. The attacker outputs b' and wins if and only if $b' = b$.

Hybrid 1: Everything is the same as Hybrid 0, except for the highlighted part. In Hybrid 1, the challenger sets public parameter pp as $\mathbf{B}_1 + f[1]\mathbf{G}, \dots, \mathbf{B}_L + f[L]\mathbf{G}, \mathbf{A}_0, \mathbf{A}_1$, so that the constrained key queries will be independent from the description of function f .

2. The challenger starts by running the setup algorithm. It samples $\mathbf{B}_1, \dots, \mathbf{B}_L \leftarrow \mathbb{Z}_q^{n \times m}, \mathbf{s} \leftarrow \mathbb{Z}_q^n, \mathbf{A}_0, \mathbf{A}_1 \leftarrow \mathbb{Z}_q^{n \times m}, \mathbf{e} \leftarrow \chi_0^{Lm}$. The challenger outputs $\mathbf{B}_1 + f[1]\mathbf{G}, \dots, \mathbf{B}_L + f[L]\mathbf{G}, \mathbf{A}_0, \mathbf{A}_1$. It samples a bit $b \leftarrow \{0, 1\}$.
3. The attacker will make polynomial evaluation and constrained key queries. Let $Q = \text{poly}(\lambda)$ be the number of the evaluation queries.
 - During each constrained key query, the attacker outputs $\mathbf{y}$. The challenger samples $\mathbf{e}_y \leftarrow \chi_1^m$, and outputs $\text{sk} := (\mathbf{s}^T \cdot [\mathbf{B}_1 | \dots | \mathbf{B}_L] + \mathbf{e}^T, \mathbf{s}^T \cdot \mathbf{A}(\mathbf{y}) + \mathbf{e}_y^T)$. It adds $\mathbf{y}$ to set $\mathcal{Y}$.
 - During each evaluation query, the attacker outputs an input $\mathbf{x}$. If $\mathbf{x}^* \in \mathcal{X}$ or $f(\mathbf{x}^*) \in \mathcal{Y}$, the attacker fails and the game aborts. The challenger responds with $\text{Eval}(\text{sk}, \mathbf{x})$: It computes $\mathbf{H}_{\mathcal{U}_x} := \text{EvalF}(\mathcal{U}_x, \{\mathbf{B}_i\}_{i \in L})$ and $\mathbf{B}_{\mathcal{U}_x} := [\mathbf{B}_1 + f[1]\mathbf{G} | \dots | \mathbf{B}_L + f[L]\mathbf{G}] \cdot \mathbf{H}_{\mathcal{U}_x}$, and outputs $\lceil \mathbf{s}^T \cdot \mathbf{B}_{\mathcal{U}_x} \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \rceil$. It adds $\mathbf{x}$ to set $\mathcal{X}$.
4. During the challenge query, the attacker outputs an input $\mathbf{x}^*$. If $b = 0$, the challenger responds with $\text{Eval}(\text{sk}, \mathbf{x}^*)$: It computes $\mathbf{H}_{\mathcal{U}_{x^*}} := \text{EvalF}(\mathcal{U}_{x^*}, \{\mathbf{B}_i\}_{i \in L})$, $\mathbf{B}_{\mathcal{U}_{x^*}} := [\mathbf{B}_1 + f[1]\mathbf{G} | \dots | \mathbf{B}_L + f[L]\mathbf{G}] \cdot \mathbf{H}_{\mathcal{U}_{x^*}}$, and outputs $\lceil \mathbf{s}^T \cdot \mathbf{B}_{\mathcal{U}_{x^*}} \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x}^*)) \rceil$. Otherwise the challenger samples $\mathbf{r}_{x^*} \leftarrow \mathbb{Z}_q^m$ and outputs $\lceil \mathbf{r}_{x^*}^T \rceil$.

Hybrid 2: The challenger generates constrained keys during the evaluation and the challenge queries. For example, given $\mathbf{x}$, the challenger sets $\mathbf{y}$ as $f(\mathbf{x})$, generates constrained key using $\mathbf{y}$ following the process of constrained key queries, and then answers the queries following the constrained evaluation algorithm.

3. The attacker will make polynomial evaluation and constrained key queries. Let $Q = \text{poly}(\lambda)$ be the number of the evaluation queries.
 - During each constrained key query, the attacker outputs $\mathbf{y}$. The challenger samples $\mathbf{e} \leftarrow \chi_1^{Lm}, \mathbf{e}_y \leftarrow \chi_1^m$, and outputs $\text{sk} := (\mathbf{s}^T \cdot [\mathbf{B}_1 | \dots | \mathbf{B}_L] + \mathbf{e}^T, \mathbf{s}^T \cdot \mathbf{A}(\mathbf{y}) + \mathbf{e}_y^T)$. It adds $\mathbf{y}$ to set $\mathcal{Y}$.
 - During each evaluation query, the attacker outputs an input $\mathbf{x}$. **The challenger computes $\mathbf{H}_{\mathcal{U}_{x,f}} := \text{EvalFX}(\mathcal{U}_x, f, \{\mathbf{B}_i\}_{i \in L})$. Let $\mathbf{y}$ be the output of $f(\mathbf{x})$, the challenger samples $\mathbf{e} \leftarrow \chi_1^{Lm}, \mathbf{e}_y \leftarrow \chi_1^m$. It outputs**

$$\lceil ((\mathbf{s}^T \cdot [\mathbf{B}_1 | \dots | \mathbf{B}_L] + \mathbf{e}^T) \cdot \mathbf{H}_{\mathcal{U}_{x,f}} + \mathbf{s}^T \cdot \mathbf{A}(\mathbf{y}) + \mathbf{e}_y^T) \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \rceil.$$
It adds $\mathbf{x}$ to $\mathcal{X}$.
4. During the challenge query, the attacker outputs an input $\mathbf{x}^*$. If $\mathbf{x}^* \in \mathcal{X}$ or $f(\mathbf{x}^*) \in \mathcal{Y}$, the attacker fails and the game aborts. Let $\mathbf{y}^*$ be $f(\mathbf{x}^*)$. If $b = 0$, it samples $\mathbf{e} \leftarrow \chi_1^{Lm}, \mathbf{e}_{y^*} \leftarrow \chi_1^m$, and computes $\mathbf{H}_{\mathcal{U}_{x^*,f}} := \text{EvalFX}(\mathcal{U}_{x^*}, f, \{\mathbf{B}_i\}_{i \in L})$. It outputs

$$\lceil ((\mathbf{s}^T \cdot [\mathbf{B}_1 | \dots | \mathbf{B}_L] + \mathbf{e}^T) \cdot \mathbf{H}_{\mathcal{U}_{x^*,f}} + \mathbf{s}^T \cdot \mathbf{A}(\mathbf{y}^*) + \mathbf{e}_{y^*}^T) \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x}^*)) \rceil.$$

Otherwise the challenger samples $\mathbf{r}_{x^*} \leftarrow \mathbb{Z}_q^m$ and outputs $\lceil \mathbf{r}_{x^*}^T \rceil$.

Between Hybrid 2 and 3, we introduce the following intermediate hybrids:

Hybrid 2.i.0: Everything of Hybrid 2.i.0 is the same as Hybrid 2, except for the highlighted part. In this experiment, the challenger samples uniformly random LWE secret $\mathbf{s}_{\mathbf{y}[[i-1]]}$ and random $\mathbf{r}_{\mathbf{y}[[j]]}$ for $j \in [i-1]$, where such vectors are indexed by prefixes of $\mathbf{y}$. It uses these $\mathbf{s}_{\mathbf{y}[[i-1]]}$, $\mathbf{r}_{\mathbf{y}[[j]]}$ to compute and answer each query by $\mathcal{A}$. For $i = 1$ where $\mathbf{y}[[i-1]]$ is an empty string, it is identical to Hybrid 2.

3. The attacker will make polynomial evaluation and constrained key queries. Let $Q = \text{poly}(\lambda)$ be the number of the evaluation queries.

- During each constrained key query, the attacker outputs $\mathbf{y}$. The challenger samples fresh $\mathbf{e} \leftarrow \chi_1^{Lm}$, $\mathbf{e}_{\mathbf{y}} \leftarrow \chi_1^m$. Next, it takes out the first i bits of $\mathbf{y}$ as $\mathbf{y}_i$. It samples $\mathbf{s}_{\mathbf{y}[[i-1]]} \leftarrow \mathbb{Z}_q^n$ and $\mathbf{r}_{\mathbf{y}[[j]]} \leftarrow \mathbb{Z}_q^m$ for all $j \in [i-1]$ (if such $\mathbf{s}_{\mathbf{y}[[i]]}$ or $\mathbf{r}_{\mathbf{y}[[j]]}$ was not chosen in any of the previous queries). If $i = 1$, it sets $\mathbf{b}$ such that $\mathbf{b}^T = \mathbf{s}_{\mathbf{y}[[i-1]]}^T \cdot [\mathbf{B}_1] \dots [\mathbf{B}_L] + \mathbf{e}^T$, otherwise for $2 \leq i \leq \ell'$, the challenger samples $\mathbf{b} \leftarrow \mathbb{Z}_q^{Lm}$ (if $\mathbf{b}$ was not sampled in any previous query). Next, the challenger samples fresh $\mathbf{e}_{\mathbf{y}} \leftarrow \chi_1^m$. It sets $\mathbf{c}_{\mathbf{y}}$ such that

$$\mathbf{c}_{\mathbf{y}}^T = \mathbf{s}_{\mathbf{y}[[i-1]]}^T \cdot \mathbf{A}_{\mathbf{y}[[i]]} \prod_{j=i+1}^{\ell'} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{y}[[j]])} + \sum_{j=1}^{i-1} \left(\mathbf{r}_{\mathbf{y}[[j]]}^T \prod_{k=j+1}^{\ell'} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{y}[[k]])} \right) + \mathbf{e}_{\mathbf{y}}^T.$$

The challenger outputs $\text{sk} := (\mathbf{b}^T + \mathbf{e}^T, \mathbf{c}_{\mathbf{y}}^T)$. It adds $\mathbf{y}$ to set $\mathcal{Y}$.

- During each evaluation query, the attacker outputs an input $\mathbf{x}$. Let $\mathbf{y}$ be the output of $f(\mathbf{x})$. The challenger computes $\mathbf{H}_{\mathcal{U}_{\mathbf{x},f}} := \text{EvalFX}(\mathcal{U}_{\mathbf{x},f}, \{\mathbf{B}_i\}_{i \in L})$. It samples $\mathbf{s}_{\mathbf{y}[[i-1]]} \leftarrow \mathbb{Z}_q^n$ and $\mathbf{r}_{\mathbf{y}[[j]]} \leftarrow \mathbb{Z}_q^m$ for all $j \in [i-1]$ (if such $\mathbf{s}_{\mathbf{y}[[i]]}$ or $\mathbf{r}_{\mathbf{y}[[j]]}$ was not chosen). If $i = 1$, it sets $\mathbf{b}$ such that $\mathbf{b}^T = \mathbf{s}_{\mathbf{y}[[i-1]]}^T \cdot [\mathbf{B}_1] \dots [\mathbf{B}_L] + \mathbf{e}^T$, otherwise for $2 \leq i \leq \ell'$, the challenger samples $\mathbf{b} \leftarrow \mathbb{Z}_q^{Lm}$ (if $\mathbf{b}$ was not sampled before). Next, the challenger samples fresh $\mathbf{e}_{\mathbf{y}} \leftarrow \chi_1^m$. It sets $\mathbf{c}_{\mathbf{y}}$ such that

$$\mathbf{c}_{\mathbf{y}}^T = \mathbf{s}_{\mathbf{y}[[i-1]]}^T \cdot \mathbf{A}_{\mathbf{y}[[i]]} \prod_{j=i+1}^{\ell'} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{y}[[j]])} + \sum_{j=1}^{i-1} \left(\mathbf{r}_{\mathbf{y}[[j]]}^T \prod_{k=j+1}^{\ell'} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{y}[[k]])} \right) + \mathbf{e}_{\mathbf{y}}^T.$$

It outputs $\lceil (\mathbf{b}^T \cdot \mathbf{H}_{\mathcal{U}_{\mathbf{x},f}} + \mathbf{c}_{\mathbf{y}}^T) \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \rceil$.

4. During the challenge query, the attacker outputs an input $\mathbf{x}^*$. If $\mathbf{x}^* \in \mathcal{X}$ or $f(\mathbf{x}^*) \in \mathcal{Y}$, the attacker fails and the game aborts. Let $\mathbf{y}^*$ be $f(\mathbf{x}^*)$. If $b = 0$, the challenger computes $\mathbf{H}_{\mathcal{U}_{\mathbf{x}^*,f}} := \text{EvalFX}(\mathcal{U}_{\mathbf{x}^*,f}, \{\mathbf{B}_i\}_{i \in L})$. It samples $\mathbf{s}_{\mathbf{y}^*[[i-1]]} \leftarrow \mathbb{Z}_q^n$ and $\mathbf{r}_{\mathbf{y}^*[[j]]} \leftarrow \mathbb{Z}_q^m$ for all $j \in [i-1]$ (if such $\mathbf{s}_{\mathbf{y}^*[[i]]}$ or $\mathbf{r}_{\mathbf{y}^*[[j]]}$ was not chosen). If $i = 1$, it sets $\mathbf{b}$ such that $\mathbf{b}^T = \mathbf{s}_{\mathbf{y}^*[[i-1]]}^T \cdot [\mathbf{B}_1] \dots [\mathbf{B}_L] + \mathbf{e}^T$, otherwise for $2 \leq i \leq \ell'$, the challenger samples $\mathbf{b} \leftarrow \mathbb{Z}_q^{Lm}$ (if $\mathbf{b}$ was not sampled before). Next, the challenger samples fresh $\mathbf{e}_{\mathbf{y}^*} \leftarrow \chi_1^m$. It sets $\mathbf{c}_{\mathbf{y}^*}$ such that

$$\mathbf{c}_{\mathbf{y}^*}^T = \mathbf{s}_{\mathbf{y}^*[[i-1]]}^T \cdot \mathbf{A}_{\mathbf{y}^*[[i]]} \prod_{j=i+1}^{\ell'} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{y}^*[[j]])} + \sum_{j=1}^{i-1} \left(\mathbf{r}_{\mathbf{y}^*[[j]]}^T \prod_{k=j+1}^{\ell'} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{y}^*[[k]])} \right) + \mathbf{e}_{\mathbf{y}^*}^T.$$

It outputs $\lceil (\mathbf{b}^T \cdot \mathbf{H}_{\mathcal{U}_{\mathbf{x}^*,f}} + \mathbf{c}_{\mathbf{y}^*}^T) \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \rceil$. Otherwise if $b = 1$, the challenger samples $\mathbf{r}_{\mathbf{x}^*} \leftarrow \mathbb{Z}_q^m$ and outputs $\lceil \mathbf{r}_{\mathbf{x}^*}^T \rceil$.

5. It repeats step 3, **except that the challenger does not resample $\mathbf{b}$** , and if $\mathcal{A}$ queries $f(\mathbf{x}^*)$ during constrained key queries or $\mathbf{x}^*$ during evaluation queries, the game immediately fails and aborts.

Hybrid 2.i.1: It is the same as Hybrid 2.i.0, except that the challenger samples $\mathbf{e}'_{\mathbf{y}[[i]]}$ from noise distribution χ_0 , and includes those errors when computing $\mathbf{c}_{\mathbf{y}}$. If $i = 1$, it additionally samples $\mathbf{e}'$ and adds it to $\mathbf{b}$.

3. The attacker will make polynomial evaluation and constrained key queries. Let $Q = \text{poly}(\lambda)$ be the number of the evaluation queries.
 - During each constrained key query, the attacker outputs $\mathbf{y}$. The challenger samples fresh $\mathbf{e} \leftarrow \chi_1^{Lm}$, $\mathbf{e}_{\mathbf{y}} \leftarrow \chi_1^m$. Next, it takes out the first i bits of $\mathbf{y}$ as $\mathbf{y}_i$. It samples $\mathbf{e}' \leftarrow \chi_0^{Lm}$, $\mathbf{s}_{\mathbf{y}[[i-1]]} \leftarrow \mathbb{Z}_q^n$, $\mathbf{e}'_{\mathbf{y}[[i]]} \leftarrow \chi_0^m$, and $\mathbf{r}_{\mathbf{y}[[j]]} \leftarrow \mathbb{Z}_q^m$ for all $j \in [i-1]$ (for $\mathbf{e}'$, $\mathbf{e}'_{\mathbf{y}[[i]]}$, $\mathbf{s}_{\mathbf{y}[[i]]}$, or $\mathbf{r}_{\mathbf{y}[[j]]}$ that was not sampled in previous queries). If $i = 1$, it sets $\mathbf{b}^T$ such that $\mathbf{b}^T = \mathbf{s}_{\mathbf{y}[[i-1]]}^T \cdot [\mathbf{B}_1 | \dots | \mathbf{B}_L] + \mathbf{e}'^T$, otherwise for $2 \leq i \leq \ell'$, the challenger samples $\mathbf{b} \leftarrow \mathbb{Z}_q^{Lm}$ (if $\mathbf{b}$ was not sampled in any previous query). It sets $\mathbf{c}_{\mathbf{y}}$ such that

$$\mathbf{c}_{\mathbf{y}}^T = \left(\mathbf{s}_{\mathbf{y}[[i-1]]}^T \cdot \mathbf{A}_{\mathbf{y}[[i]]} + \mathbf{e}'_{\mathbf{y}[[i]]}^T \right) \prod_{j=i+1}^{\ell'} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{y}[[j]])} + \sum_{j=1}^{i-1} \left(\mathbf{r}_{\mathbf{y}[[j]]}^T \prod_{k=j+1}^{\ell'} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{y}[[k]])} \right) + \mathbf{e}_{\mathbf{y}}^T.$$

The challenger outputs $\text{sk} := (\mathbf{b}^T + \mathbf{e}^T, \mathbf{c}_{\mathbf{y}}^T)$. It adds $\mathbf{y}$ to set $\mathcal{Y}$.

- During each evaluation query, the attacker outputs an input $\mathbf{x}$. Let $\mathbf{y}$ be the output of $\mathbf{A}(\mathbf{x})$. **Similar to the changes from Hybrid 2.i.0 to 2.i.1 in the constrained key queries, the challenger additionally samples $\mathbf{e}' \leftarrow \chi_0^{Lm}$, $\mathbf{e}'_{\mathbf{y}[[i]]} \leftarrow \chi_0^m$ if they were not sampled in any previous query, and sets $\mathbf{c}_{\mathbf{y}}$ the same way as the above constrained key queries.** It outputs $\lceil (\mathbf{b}^T \cdot \mathbf{H}_{\mathcal{U}_{\mathbf{x},f}} + \mathbf{c}_{\mathbf{y}}^T) \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \rceil$.
4. During the challenge query, the attacker outputs an input $\mathbf{x}^*$. If $\mathbf{x}^* \in \mathcal{X}$ or $f(\mathbf{x}^*) \in \mathcal{Y}$, the attacker fails and the game aborts. Let $\mathbf{y}^*$ be $f(\mathbf{x}^*)$. **If $b = 0$, the challenger additionally samples $\mathbf{e}' \leftarrow \chi_0^{Lm}$, $\mathbf{e}'_{\mathbf{y}^*[[i]]} \leftarrow \chi_0^m$ if they were not sampled before, and sets $\mathbf{c}_{\mathbf{y}^*}$ in the way identical to the constrained key and evaluation queries.** It outputs $\lceil (\mathbf{b}^T \cdot \mathbf{H}_{\mathcal{U}_{\mathbf{x}^*,f}} + \mathbf{c}_{\mathbf{y}^*}^T) \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \rceil$. Otherwise if $b = 1$, the challenger samples $\mathbf{r}_{\mathbf{x}^*} \leftarrow \mathbb{Z}_q^m$ and outputs $\lceil \mathbf{r}_{\mathbf{x}^*}^T \rceil$.

Hybrid 2.i.2: It is identical to the Hybrid 2.i.1, except that the challenger uses uniformly random $\mathbf{r}_{\mathbf{y}[[j]]} \leftarrow \mathbb{Z}_q^m$ to replace $\mathbf{s}_{\mathbf{y}[[i-1]]}^T \cdot \mathbf{A}_{\mathbf{y}[[i]]} + \mathbf{e}'_{\mathbf{y}[[i]]}^T$ when computing $\mathbf{c}_{\mathbf{y}}$. Additionally for $i = 1$ it uses uniformly random $\mathbf{b}$. When $i = \ell'$, this is statistically close to the distribution of Hybrid 3.

3. The attacker will make polynomial evaluation and constrained key queries. Let $Q = \text{poly}(\lambda)$ be the number of the evaluation queries.
 - During each constrained key query, the attacker outputs $\mathbf{y}$. The challenger takes out the first i bits of $\mathbf{y}$ as $\mathbf{y}_i$. It samples $\mathbf{r}_{\mathbf{y}[[j]]} \leftarrow \mathbb{Z}_q^m$ for all $j \in [i]$ (if such $\mathbf{r}_{\mathbf{y}[[j]]}$ was not

chosen) and fresh $\mathbf{e} \leftarrow \chi_1^{Lm}$, $\mathbf{e}_y \leftarrow \chi_1^m$. The challenger samples $\mathbf{b} \leftarrow \mathbb{Z}_q^{Lm}$ (if $\mathbf{b}$ was not sampled in the other queries before). It sets $\mathbf{c}_y$ such that

$$\mathbf{c}_y^T = \mathbf{r}_{\mathbf{y}[[i]]}^T \prod_{j=i+1}^{\ell'} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{y}[j]}) + \sum_{j=1}^{i-1} \left(\mathbf{r}_{\mathbf{y}[[j]]}^T \prod_{k=j+1}^{\ell'} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{y}[k]}) \right) + \mathbf{e}_y^T.$$

The challenger outputs $\mathbf{sk} := (\mathbf{b}^T + \mathbf{e}^T, \mathbf{c}_y^T)$. It adds $\mathbf{y}$ to set $\mathcal{Y}$.

- During each evaluation query, the attacker outputs an input $\mathbf{x}$. Let $\mathbf{y}$ be the output of $\mathbf{A}(\mathbf{x})$. The challenger then samples fresh $\mathbf{e} \leftarrow \chi_1^{Lm}$, $\mathbf{e}_y \leftarrow \chi_1^m$. **Next, it samples $\mathbf{r}_{\mathbf{y}[[j]]}$ for all $j \in [i]$ (if $\mathbf{r}_{\mathbf{y}[[j]]}$ was not chosen before) and sets $\mathbf{c}_y$ in the way that is identical to the above constrained key queries.** It outputs $\lceil (\mathbf{b}^T \cdot \mathbf{H}_{\mathcal{U}_{\mathbf{x},f}} + \mathbf{c}_y^T) \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \rceil$.
- 4. For the challenge query, the only change is as follows: **The challenger samples $\mathbf{r}_{\mathbf{y}^*[[i]]}$ and sets $\mathbf{c}_{\mathbf{y}^*}$ as in the constrained key/evaluation queries.** For $b = 0$, it again outputs $\lceil (\mathbf{b}^T \cdot \mathbf{H}_{\mathcal{U}_{\mathbf{x}^*,f}} + \mathbf{c}_{\mathbf{y}^*}^T) \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \rceil$. And otherwise if $b = 1$, the challenger samples $\mathbf{r}_{\mathbf{x}^*} \leftarrow \mathbb{Z}_q^m$ and outputs $\lceil \mathbf{r}_{\mathbf{x}^*}^T \rceil$.

Hybrid 3: Hybrid 3 is the same as Hybrid 2. ℓ' .2, except for the highlighted part. In the view of $\mathcal{A}$, the distribution of Hybrid 3 is statistically close to Hybrid 2. ℓ' .2. The only difference is that the challenger removes LWE error $\mathbf{e}_y$ from the equation of $\mathbf{c}_y$ when answering the evaluation queries. To see that the distributions are statistically close, we remark that $\mathbf{r}_{\mathbf{y}[[\ell']]}$ in Hybrid 2. ℓ' .2 and $\mathbf{c}_y$ in Hybrid 3 are both uniformly random.

3. The attacker will make polynomial evaluation and constrained key queries. Let $Q = \text{poly}(\lambda)$ be the number of the evaluation queries.
 - During each constrained key query, the attacker outputs $\mathbf{y}$. The challenger samples $\mathbf{e} \leftarrow \chi_1^{Lm}$, $\mathbf{e}_y \leftarrow \chi_1^m$. (It samples fresh $\mathbf{e}, \mathbf{e}_y$ if they were already sampled in previous queries.) **It additionally samples $\mathbf{c}_y \leftarrow \mathbb{Z}_q^m$ if $\mathbf{c}_y$ was not never sampled in previous queries.** It outputs $\mathbf{sk} := (\mathbf{b}^T + \mathbf{e}^T, \mathbf{c}_y^T + \mathbf{e}_y^T)$, and adds $\mathbf{y}$ to $\mathcal{Y}$.
 - During each evaluation query, the attacker outputs an input $\mathbf{x}$. Let $\mathbf{y}$ be the output of $f(\mathbf{x})$. **It computes $\mathbf{H}_{\mathcal{U}_{\mathbf{x},f}} := \text{EvalFX}(\mathcal{U}_{\mathbf{x},f}, \{\mathbf{B}_i\}_{i \in L})$. If $\mathbf{c}_y$ was not sampled before, the challenger additionally samples $\mathbf{c}_y \leftarrow \mathbb{Z}_q^m$.** It outputs

$$\lceil (\mathbf{b}^T \cdot \mathbf{H}_{\mathcal{U}_{\mathbf{x},f}} + \mathbf{c}_y^T) \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \rceil.$$

It adds $\mathbf{x}$ to $\mathcal{X}$.

4. During the challenge query, the attacker outputs an input $\mathbf{x}^*$. If $\mathbf{x}^* \in \mathcal{X}$ or $f(\mathbf{x}^*) \in \mathcal{Y}$, the attacker fails and the game aborts. Let $\mathbf{y}^*$ be $f(\mathbf{x}^*)$. **If $b = 0$, the challenger computes $\mathbf{H}_{\mathcal{U}_{\mathbf{x}^*,f}} := \text{EvalFX}(\mathcal{U}_{\mathbf{x}^*,f}, \{\mathbf{B}_i\}_{i \in L})$. If $\mathbf{c}_{\mathbf{y}^*}$ was not sampled before, the challenger additionally samples $\mathbf{c}_{\mathbf{y}^*} \leftarrow \mathbb{Z}_q^m$.** It outputs

$$\lceil (\mathbf{b}^T \cdot \mathbf{H}_{\mathcal{U}_{\mathbf{x}^*,f}} + \mathbf{c}_{\mathbf{y}^*}^T) \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \rceil.$$

Otherwise the challenger samples $\mathbf{r}_{\mathbf{x}^*} \leftarrow \mathbb{Z}_q^m$ and outputs $\lceil \mathbf{r}_{\mathbf{x}^*}^T \rceil$.

The following intermediate hybrids are between Hybrid 3 and 4:

Hybrid 3.i.0: It is the same as Hybrid 3, except for the highlighted part. In Hybrid 3.i.0, the challenger guesses a random index r from $[Q + 1]$. Let $\mathbf{x}$ be the r -th evaluation query, $\mathbf{y}$ be $f(\mathbf{x})$, and $\mathbf{y}^*$ be the challenge query, the guess is successful only if $\mathbf{y} = \mathbf{y}^*$, or $\mathbf{y}^*$ never appears in any of the evaluation queries. The game is artificially aborted if the guess fails. Next for any evaluation query where $\mathbf{y} = \mathbf{y}^*$ (assuming that guess is successful), the challenger random vectors $\mathbf{r}_{\mathbf{y}, \mathbf{x}[[j]]}$ for $j \in [i]$ which is indexed by $\mathbf{y}$ and prefix of $\mathbf{x}$ of length i . It computes the output of evaluation and challenge queries using such $\mathbf{r}_{\mathbf{x}_i}$. Hybrid 3.0.0 will be identically distributed as Hybrid 3 in the view of $\mathcal{A}$.

3. The attacker will make polynomial evaluation and constrained key queries. Let $Q = \text{poly}(\lambda)$ be the number of the evaluation queries. Before the queries begin, the challenger samples $\mathbf{b} \leftarrow \mathbb{Z}_q^{Lm}$, and $r \leftarrow [Q + 1]$.

- It remains unchanged from Hybrid 3, **except that if $\mathbf{y}^*$ has been sampled and $\mathbf{y} = \mathbf{y}^*$, the challenger aborts the game artificially.**
- During each evaluation query, the attacker outputs an input $\mathbf{x}$. Then the challenger computes $\mathbf{H}_{\mathcal{U}_{\mathbf{x}, f}} := \text{EvalFX}(\mathcal{U}_x, f, \{\mathbf{B}_i\}_{i \in L})$. Let $\mathbf{y}$ be $f(\mathbf{x})$. **If this is the r -th evaluation query, and $\mathbf{y}$ has never been queried during the constrained key queries, and $f(\mathbf{x}') \neq \mathbf{y}$ for all $\mathbf{x}' \in \mathcal{X}$, then the challenger sets $\mathbf{y}^*$ as $\mathbf{y}$. Next, if $\mathbf{y}^* = \mathbf{y}$ (either this is the r -th query and the above conditions hold, or this is some evaluation query after the r -th), the challenger samples $\mathbf{r}_{\mathbf{y}} \leftarrow \mathbb{Z}_q^m$ and $\mathbf{r}_{\mathbf{y}, \mathbf{x}[[j]]} \leftarrow \mathbb{Z}_q^m$ for all $j \in [i]$ (if such $\mathbf{r}_{\mathbf{y}}$ or $\mathbf{r}_{\mathbf{y}, \mathbf{x}[[j]]}$ was not chosen). Next, the challenger sets $\mathbf{b}_{\mathbf{x}, f}$ such that $\mathbf{b}_{\mathbf{x}, f}^T = \mathbf{b}^T \cdot \mathbf{H}_{\mathcal{U}_{\mathbf{x}, f}} \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x}))$. It outputs**

$$\left[\mathbf{b}_{\mathbf{x}, f}^T + \mathbf{r}_{\mathbf{y}, \mathbf{x}[[i]]}^T \prod_{j=i+1}^{\ell} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{x}[j]}) + \sum_{j=1}^{i-1} \left(\mathbf{r}_{\mathbf{y}, \mathbf{x}[[j]]}^T \prod_{k=j+1}^{\ell} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{x}[k]}) \right) + \mathbf{r}_{\mathbf{y}}^T \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \right].$$

Otherwise, the challenger samples $\mathbf{c}_{\mathbf{y}} \leftarrow \mathbb{Z}_q^m$ (if such $\mathbf{c}_{\mathbf{y}}$ has not been sampled before) and outputs $\lceil (\mathbf{b}^T \cdot \mathbf{H}_{\mathcal{U}_{\mathbf{x}, f}} + \mathbf{c}_{\mathbf{y}}^T) \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \rceil$. The challenger adds $\mathbf{x}$ to set $\mathcal{X}$.

4. During the challenge query, the attacker outputs $\mathbf{x}^*$. If $\mathbf{x}^* \in \mathcal{X}$ or $f(\mathbf{x}^*) \in \mathcal{Y}$, the attacker fails and the game aborts. Additionally, if $f(\mathbf{x}^*) = f(\mathbf{x})$ for any $\mathbf{x} \in \mathcal{X}$ and $f(\mathbf{x}^*) \neq \mathbf{y}^*$, the challenger aborts the game. Next, if $\mathbf{y}^*$ has not yet been set, the challenger sets $\mathbf{y}^*$ as $f(\mathbf{x}^*)$. Let $\mathbf{y}^*$ be $f(\mathbf{x}^*)$. **If $b = 0$, the challenger samples $\mathbf{r}_{\mathbf{y}^*} \leftarrow \mathbb{Z}_q^m$ and $\mathbf{r}_{\mathbf{y}^*, \mathbf{x}^*[[j]]} \leftarrow \mathbb{Z}_q^m$ for all $j \in [i]$ (if such $\mathbf{r}_{\mathbf{y}^*}$ or $\mathbf{r}_{\mathbf{y}^*, \mathbf{x}^*[[j]]}$ was not chosen). Next, the challenger sets $\mathbf{b}_{\mathbf{x}^*, f}$ such that $\mathbf{b}_{\mathbf{x}^*, f}^T = \mathbf{b}^T \cdot \mathbf{H}_{\mathcal{U}_{\mathbf{x}^*, f}} \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x}^*))$. It outputs**

$$\left[\mathbf{b}_{\mathbf{x}^*, f}^T + \mathbf{r}_{\mathbf{y}^*, \mathbf{x}^*[[i]]}^T \prod_{j=i+1}^{\ell} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{x}^*[j]}) + \sum_{j=1}^{i-1} \left(\mathbf{r}_{\mathbf{y}^*, \mathbf{x}^*[[j]]}^T \prod_{k=j+1}^{\ell} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{x}^*[k]}) \right) + \mathbf{r}_{\mathbf{y}^*}^T \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x}^*)) \right].$$

Otherwise if $b = 1$, the challenger samples $\mathbf{r}_{\mathbf{x}^*} \leftarrow \mathbb{Z}_q^m$ and outputs $\lceil \mathbf{r}_{\mathbf{x}^*}^T \rceil$.

5. It repeats step 3, except for the following: The challenger does not resample $\mathbf{b}$, and if $\mathcal{A}$ queries $f(\mathbf{x}^*)$ during constrained key queries or $\mathbf{x}^*$ during evaluation queries, the game immediately fails and aborts. **Furthermore, the challenger no longer considers the guessed index r after the challenge query, so that during each evaluation query where the attacker queries $\mathbf{x}$, the challenger samples uniformly random $\mathbf{r}_{\mathbf{y}}, \mathbf{r}_{\mathbf{y}, \mathbf{x}[[j]]}$ to answer evaluation queries only when $f(\mathbf{x}) = \mathbf{y}^*$.**

Hybrid 3.i.1: It is the same as Hybrid 3.i.0, except for the highlighted sentences: It replaces $\mathbf{r}_{\mathbf{y},\mathbf{x}[[i]]}^T$ with $\mathbf{s}_{\mathbf{y},\mathbf{x}[[i]]}^T \cdot \mathbf{G} + \mathbf{r}_{\mathbf{y},\mathbf{x}[[i]]}^T$ for some uniform random $\mathbf{s}_{\mathbf{y},\mathbf{x}[[i]]}$ during the guessed evaluation/challenge queries.

3. The attacker will make polynomial evaluation and constrained key queries. Let $Q = \text{poly}(\lambda)$ be the number of the evaluation queries. Before the queries begin, the challenger samples $\mathbf{b} \leftarrow \mathbb{Z}_q^{Lm}$, and $r \leftarrow [Q + 1]$.

- During each evaluation query, the attacker outputs $\mathbf{x}$. Next the challenger computes $\mathbf{H}_{\mathcal{U}_{\mathbf{x},f}} := \text{EvalFX}(\mathcal{U}_x, f, \{\mathbf{B}_i\}_{i \in L})$. Let $\mathbf{y}$ be $f(\mathbf{x})$. If this is the r -th evaluation query, and $\mathbf{y}$ has never been queried during the constrained key queries, and $f(\mathbf{x}') \neq \mathbf{y}$ for all $\mathbf{x}' \in \mathcal{X}$, then the challenger sets $\mathbf{y}^*$ as $\mathbf{y}$. Next, if $\mathbf{y}^* = \mathbf{y}$, then the challenger samples $\mathbf{r}_{\mathbf{y}} \leftarrow \mathbb{Z}_q^m$, $\mathbf{s}_{\mathbf{y},\mathbf{x}[[i]]} \leftarrow \mathbb{Z}_q^n$, and $\mathbf{r}_{\mathbf{y},\mathbf{x}[[j]]} \leftarrow \mathbb{Z}_q^m$ for all $j \in [i]$ (if such $\mathbf{r}_{\mathbf{y}}$, $\mathbf{s}_{\mathbf{y},\mathbf{x}[[i]]}$, or $\mathbf{r}_{\mathbf{y},\mathbf{x}[[j]]}$ was not chosen in any previous query). It sets $\mathbf{b}_{\mathbf{x},f}$ such that $\mathbf{b}_{\mathbf{x},f}^T = \mathbf{b}^T \cdot \mathbf{H}_{\mathcal{U}_{\mathbf{x},f}} \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x}))$. It outputs

$$\left[\mathbf{b}_{\mathbf{x},f}^T + \left(\mathbf{s}_{\mathbf{y},\mathbf{x}[[i]]}^T \cdot \mathbf{G} + \mathbf{r}_{\mathbf{y},\mathbf{x}[[i]]}^T \right) \prod_{j=i+1}^{\ell} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{x}[j]}) + \sum_{j=1}^{i-1} \left(\mathbf{r}_{\mathbf{y},\mathbf{x}[[j]]}^T \prod_{k=j+1}^{\ell} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{x}[k]}) \right) + \mathbf{r}_{\mathbf{y}}^T \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \right].$$

Otherwise, the challenger samples $\mathbf{c}_{\mathbf{y}} \leftarrow \mathbb{Z}_q^m$ (if such $\mathbf{c}_{\mathbf{y}}$ has not been sampled before) and outputs $\lceil (\mathbf{b}^T \cdot \mathbf{H}_{\mathcal{U}_{\mathbf{x},f}} + \mathbf{c}_{\mathbf{y}}^T) \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \rceil$. The challenger adds $\mathbf{x}$ to set $\mathcal{X}$.

4. Similar to the evaluation queries, the challenger of the challenge query of Hybrid 3.i.1 only additionally samples $\mathbf{s}_{\mathbf{y}^*,\mathbf{x}^*[[i]]} \leftarrow \mathbb{Z}_q^n$ if it was not sampled in any previous query, and the output becomes

$$\left[\mathbf{b}_{\mathbf{x}^*,f}^T + \left(\mathbf{s}_{\mathbf{y}^*,\mathbf{x}^*[[i]]}^T \cdot \mathbf{G} + \mathbf{r}_{\mathbf{y}^*,\mathbf{x}^*[[i]]}^T \right) \prod_{j=i+1}^{\ell} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{x}^*[j]}) + \sum_{j=1}^{i-1} \left(\mathbf{r}_{\mathbf{y}^*,\mathbf{x}^*[[j]]}^T \prod_{k=j+1}^{\ell} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{x}^*[k]}) \right) + \mathbf{r}_{\mathbf{y}^*}^T \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x}^*)) \right].$$

The other steps directly follow from the challenge query of Hybrid 3.i.0.

Hybrid 3.i.2: It is the same as Hybrid 3.i.1, except that the challenger samples $\mathbf{e}_{\mathbf{y},\mathbf{x}[[i+1]]}$ from error distribution χ_0 and the error is included when answering the guessed evaluation/challenge queries.

3. The attacker will make polynomial evaluation and constrained key queries. Let $Q = \text{poly}(\lambda)$ be the number of the evaluation queries. Before the queries begin, the challenger samples $\mathbf{b} \leftarrow \mathbb{Z}_q^{Lm}$, and $r \leftarrow [Q + 1]$.

- During each evaluation query, the attacker outputs $\mathbf{x}$. Next the challenger computes $\mathbf{H}_{\mathcal{U}_{\mathbf{x},f}} := \text{EvalFX}(\mathcal{U}_x, f, \{\mathbf{B}_i\}_{i \in L})$. Let $\mathbf{y}$ be $f(\mathbf{x})$. If this is the r -th evaluation query, and $\mathbf{y}$ has never been queried during the constrained key queries, and $f(\mathbf{x}') \neq \mathbf{y}$ for all $\mathbf{x}' \in \mathcal{X}$, then the challenger sets $\mathbf{y}^*$ as $\mathbf{y}$. Next, if $\mathbf{y}^* = \mathbf{y}$, then the challenger

samples $\mathbf{r}_y \leftarrow \mathbb{Z}_q^m$, $\mathbf{r}_y \leftarrow \mathbb{Z}_q^m$, $\mathbf{s}_{y,x[[i]]} \leftarrow \mathbb{Z}_q^n$, $\mathbf{e}_{y,x[[i+1]]} \leftarrow \chi_0^m$, and $\mathbf{r}_{y,x[[j]]} \leftarrow \mathbb{Z}_q^m$ for all $j \in [i]$ (if such $\mathbf{r}_y$, $\mathbf{s}_{y,x[[i]]}$, $\mathbf{e}_{y,x[[i+1]]}$ or $\mathbf{r}_{y,x[[j]]}$ was not chosen). It sets $\mathbf{b}_{x,f}$ such that $\mathbf{b}_{x,f}^T = \mathbf{b}^T \cdot \mathbf{H}_{\mathcal{U}_{x,f}} \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x}))$. It outputs

$$\left[\mathbf{b}_{x,f}^T + \left(\mathbf{s}_{y,x[[i]]}^T \cdot \mathbf{A}_{x[i+1]} + \mathbf{e}_{y,x[[i+1]]}^T \right) \prod_{j=i+2}^{\ell} \mathbf{G}^{-1}(\mathbf{A}_{x[j]}) + \sum_{j=1}^i \left(\mathbf{r}_{y,x[[j]]}^T \prod_{k=j+1}^{\ell} \mathbf{G}^{-1}(\mathbf{A}_{x[k]}) \right) + \mathbf{r}_y^T \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \right].$$

Otherwise, the challenger samples $\mathbf{c}_y \leftarrow \mathbb{Z}_q^m$ (if such $\mathbf{c}_y$ has not been sampled before) and outputs $\lceil (\mathbf{b}^T \cdot \mathbf{H}_{\mathcal{U}_{x,f}} + \mathbf{c}_y^T) \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \rceil$. The challenger adds $\mathbf{x}$ to set $\mathcal{X}$.

4. During the challenge query, the challenger of Hybrid 3.i.2 additionally samples $\mathbf{e}_{y^*,x^*[[i+1]]} \leftarrow \chi_0^m$ if it was not sampled in any previous query, and the output becomes

$$\left[\mathbf{b}_{x^*,f}^T + \left(\mathbf{s}_{y,x[[i]]}^T \cdot \mathbf{A}_{x[i+1]} + \mathbf{e}_{y^*,x^*[[i+1]]}^T \right) \prod_{j=i+2}^{\ell} \mathbf{G}^{-1}(\mathbf{A}_{x^*[j]}) + \sum_{j=1}^i \left(\mathbf{r}_{y^*,x^*[[j]]}^T \prod_{k=j+1}^{\ell} \mathbf{G}^{-1}(\mathbf{A}_{x^*[k]}) \right) + \mathbf{r}_{y^*}^T \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x}^*)) \right].$$

The other steps are identical to the challenge query of Hybrid 3.i.1.

Hybrid 4: Hybrid 4 is identical to Hybrid 3.ℓ.0, except for the highlighted sentences. The challenger will answer the guessed evaluation queries and challenge query using truly random outputs.

3. The attacker will make polynomial evaluation and constrained key queries. Let $Q = \text{poly}(\lambda)$ be the number of the evaluation queries. Before the queries begin, the challenger samples $\mathbf{b} \leftarrow \mathbb{Z}_q^{Lm}$, and $r \leftarrow [Q + 1]$.
 - During each evaluation query, the attacker outputs an input $\mathbf{x}$. Let $\mathbf{y}$ be $f(\mathbf{x})$. If this is the r -th evaluation query, and $\mathbf{y}$ has never been queried during the constrained key queries, and $f(\mathbf{x}') \neq \mathbf{y}$ for all $\mathbf{x}' \in \mathcal{X}$, then the challenger sets $\mathbf{y}^*$ as $\mathbf{y}$. **Next, if $\mathbf{y}^* = \mathbf{y}$, then the challenger samples $\mathbf{r}_x \leftarrow \mathbb{Z}_q^m$ (if it hasn't already been sampled before) and outputs $\lceil \mathbf{r}_x^T \rceil$.** Otherwise, it computes $\mathbf{H}_{\mathcal{U}_{x,f}} := \text{EvalFX}(\mathcal{U}_x, f, \{\mathbf{B}_i\}_{i \in L})$. If $\mathbf{c}_y$ was not sampled before, the challenger additionally samples $\mathbf{c}_y \leftarrow \mathbb{Z}_q^m$. It outputs

$$\lceil (\mathbf{b}^T \cdot \mathbf{H}_{\mathcal{U}_{x,f}} + \mathbf{c}_y^T) \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \rceil.$$

The challenger adds $\mathbf{x}$ to set $\mathcal{X}$.

4. During the challenge query, the attacker outputs $\mathbf{x}^*$. If $\mathbf{x}^* \in \mathcal{X}$ or $f(\mathbf{x}^*) \in \mathcal{Y}$, the attacker fails and the game aborts. Additionally, if $f(\mathbf{x}^*) = f(\mathbf{x})$ for any $\mathbf{x} \in \mathcal{X}$ and $f(\mathbf{x}^*) \neq \mathbf{y}^*$, the challenger aborts the game. Next, if $\mathbf{y}^*$ has not yet been set, the challenger sets $\mathbf{y}^*$ as $f(\mathbf{x}^*)$. Let $\mathbf{y}^*$ be $f(\mathbf{x}^*)$. **For both $b = 0$ and $b = 1$, the challenger samples $\mathbf{r}_{x^*} \leftarrow \mathbb{Z}_q^m$ and outputs $\lceil \mathbf{r}_{x^*}^T \rceil$.**

6.2.2 Indistinguishability of Hybrids

We will now show how the above hybrid experiments are computationally indistinguishable. For any adversary $\mathcal{A}$, let $\text{Adv}_{\mathcal{A}}^i$ denote the probability that the adversary wins the game during Hybrid i .

Lemma 6.3. For any adversary $\mathcal{A}$ of unbounded running time, $\text{Adv}_{\mathcal{A}}^0 - \text{Adv}_{\mathcal{A}}^1 = 0$ (i.e. Hybrid 0 and Hybrid 1 are identically distributed).

Proof. This is because the distribution of $(\mathbf{B}_i, \mathbf{B}_i + f[i]\mathbf{G})$ is identical to $(\mathbf{B}_i - f[i]\mathbf{G}, \mathbf{B}_i)$, given that $\mathbf{B}_i \leftarrow \mathbb{Z}_q^{n \times m}$. $\square$

Lemma 6.4. For any adversary $\mathcal{A}$ of unbounded running time, there exists a negligible function $\text{negl}(\cdot)$ such that $\text{Adv}_{\mathcal{A}}^1 - \text{Adv}_{\mathcal{A}}^2 \leq \text{negl}(\lambda)$ (i.e. Hybrid 1 and Hybrid 2 are statistically close).

Proof. During the evaluation queries and in Hybrid 1, the challenger outputs

$$\left[\mathbf{s}^T \cdot [\mathbf{B}_1 + f[1]\mathbf{G} \mid \dots \mid \mathbf{B}_\ell + f[L]\mathbf{G}] \cdot \mathbf{H}_{\mathcal{U}_{\mathbf{x}}} \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \right],$$

whereas in Hybrid 2, the challenger outputs

$$\left[((\mathbf{s}^T \cdot [\mathbf{B}_1 \mid \dots \mid \mathbf{B}_L] + \mathbf{e}^T) \cdot \mathbf{H}_{\mathcal{U}_{\mathbf{x},f}} + \mathbf{s}^T \cdot \mathbf{A}(\mathbf{y}) + \mathbf{e}_{\mathbf{y}}^T) \cdot \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{x}}) \right].$$

The difference is that the challenger uses constrained evaluation to answer the evaluation queries instead of the original evaluation algorithm. The analysis is omitted as it is identical to the correctness argument. $\square$

Lemma 6.5. For any unbounded adversary $\mathcal{A}$, $\text{Adv}_{\mathcal{A}}^2 - \text{Adv}_{\mathcal{A}}^{2.1.0} = 0$.

Proof. This holds because $\mathbf{s}$ in Hybrid 2 and $\mathbf{s}_{\mathbf{y}[[0]]}$ in Hybrid 2.1.0 are both uniformly randomly sampled. In the view of $\mathcal{A}$, the two hybrid experiments are identically distributed. $\square$

Lemma 6.6. For all $i \in [\ell']$, for any adversary $\mathcal{A}$ of unbounded running time, there exists a negligible function $\text{negl}(\cdot)$ such that $\text{Adv}_{\mathcal{A}}^{2.i.0} - \text{Adv}_{\mathcal{A}}^{2.i.1} \leq \text{negl}(\lambda)$.

Proof. By truncated discrete gaussian, $\left\| \mathbf{e}_{\mathbf{y}[[i]]}^T \right\|_{\infty} \leq \sigma \cdot m$. Given that $\mathbf{G}^{-1}(\cdot)$ is a binary matrix, $\left\| \mathbf{e}_{\mathbf{y}[[i]]}^T \cdot \prod_{j=i+1}^{\ell'} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{y}[j]}) \right\|_{\infty} \leq \sigma \cdot m \cdot m^{\ell'}$. By our parameter setting, $\sigma' \geq \sigma \cdot m^{\ell'+1} \cdot 2^{\Omega(\lambda)}$. Thus by Theorem 3.4, for $\mathbf{e}_{\mathbf{y}} \leftarrow \tilde{\mathcal{D}}_{Z^m, \sigma'}$, $\mathbf{e}_{\mathbf{y}[[i]]}^T \cdot \prod_{j=i+1}^{\ell'} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{y}[j]}) + \mathbf{e}_{\mathbf{y}}^T$ and $\mathbf{e}_{\mathbf{y}}^T$ are statistically close. Our claim follows. $\square$

Lemma 6.7. Assuming that $\text{GLWE}_{n, (L+2)m, q, \chi_0}$ holds, then for any PPT adversary $\mathcal{A}$, there exists a negligible function $\text{negl}(\cdot)$ such that $\text{Adv}_{\mathcal{A}}^{2.i.1} - \text{Adv}_{\mathcal{A}}^{2.i.2} \leq \text{negl}(\lambda)$ for all $i \in [\ell']$.

Proof. Assume that there exists a polynomial time adversary $\mathcal{A}$, such that $\text{Adv}_{\mathcal{A}}^{2.i.1} - \text{Adv}_{\mathcal{A}}^{2.i.2} \geq \eta(\lambda)$ for some non-negligible function $\eta(\cdot)$, we design a reduction algorithm $\mathcal{B}$ that breaks the underlying LWE assumption.

The LWE challenger starts by outputting $[\mathbf{B}_1 \mid \dots \mid \mathbf{B}_L \mid \mathbf{A}_0 \mid \mathbf{A}_1]$ such that $\mathbf{B}_1, \dots, \mathbf{B}_L, \mathbf{A}_0, \mathbf{A}_1 \in \mathbb{Z}_q^{n \times m}$. Reduction algorithm $\mathcal{B}$ then tosses a coin $b \leftarrow \{0, 1\}$. For each $\mathbf{y}$ from $\mathcal{A}$ during those constrained key/evaluation/challenge queries (let $\mathbf{y}$ be $f(\mathbf{x})$ or $f(\mathbf{x}^*)$ during evaluation/challenge

queries), we define $\mathbf{y}[[i-1, 0]]$ as $\begin{bmatrix} \mathbf{y}[[i-1]] \\ 0 \end{bmatrix}$ and $\mathbf{y}[[i-1, 1]]$ as $\begin{bmatrix} \mathbf{y}[[i-1]] \\ 1 \end{bmatrix}$. During such query, $\mathcal{B}$ samples fresh $\mathbf{e} \leftarrow \chi_1^{Lm}, \mathbf{e}_y \leftarrow \chi_1^m$. Then, it samples $\mathbf{r}_{\mathbf{y}[[j]]} \leftarrow \mathbb{Z}_q^m$ for all $j \in [i-1]$ (if such $\mathbf{r}_{\mathbf{y}[[j]]}$ was not chosen in any previous query). If $2 \leq i \leq \ell'$ and if $\mathbf{b}$ was not set up before, $\mathcal{B}$ also samples $\mathbf{b} \leftarrow \mathbb{Z}_q^{Lm}$. If $\mathcal{A}$ has never issued a query on any $\mathbf{y}'$ such that the first $(i-1)$ components of $\mathbf{y}'$ match those of either $\mathbf{y}[[i-1, 0]]$ or $\mathbf{y}[[i-1, 1]]$ during any previous constrained key queries or evaluation queries, then $\mathcal{B}$ queries the LWE challenger and the challenger responds with $\mathbf{r} \in \mathbb{Z}_q^{(L+2)m}$. If $i = 1$ and $\mathbf{b}$ has not yet been initialized (i.e., this is the first query by $\mathcal{A}$), then $\mathcal{B}$ parses $\mathbf{r}^T$ as $\left[\mathbf{b}^T \middle| \mathbf{r}_{\mathbf{y}[[i-1, 0]]}^T \middle| \mathbf{r}_{\mathbf{y}[[i-1, 1]]}^T \right]$ where $\mathbf{b} \in \mathbb{Z}_q^{Lm}$ and $\mathbf{y}_{i-1,0}, \mathbf{y}_{i-1,1} \in \mathbb{Z}_q^m$. Otherwise if $\mathbf{b}$ has already been set, $\mathcal{B}$ parses $\mathbf{r}^T$ as $\left[\dots \middle| \mathbf{r}_{\mathbf{y}[[i-1, 0]]}^T \middle| \mathbf{r}_{\mathbf{y}[[i-1, 1]]}^T \right]$ with the same component dimensions. $\mathcal{B}$ then samples $\mathbf{e}_y \leftarrow \chi_1$ and sets $\mathbf{c}_y$ such that

$$\mathbf{c}_y^T = \mathbf{r}_{\mathbf{y}[[i]]}^T \prod_{j=i+1}^{\ell'} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{y}[j]}) + \sum_{j=1}^{i-1} \left(\mathbf{r}_{\mathbf{y}[[j]]}^T \prod_{k=j+1}^{\ell'} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{y}[k]}) \right) + \mathbf{e}_y^T.$$

If $\mathcal{A}$'s query is a constrained key query, $\mathcal{B}$ outputs $\text{sk} := (\mathbf{b}^T + \mathbf{e}^T, \mathbf{c}_y^T)$ and adds $\mathbf{y}$ to $\mathcal{Y}$. Otherwise, if it is an evaluation query, $\mathcal{B}$ computes $\mathbf{H}_{\mathcal{U}_{\mathbf{x},f}} := \text{EvalFX}(\mathcal{U}_{\mathbf{x}}, f, \{\mathbf{B}_i\}_{i \in L})$. It adds $\mathbf{x}$ to set $\mathcal{X}$, and outputs $\lceil (\mathbf{b}^T \cdot \mathbf{H}_{\mathcal{U}_{\mathbf{x},f}} + \mathbf{c}_y^T) \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \rceil$. If it is the challenge query and if $\mathbf{x}^* \in \mathcal{X}$ or $f(\mathbf{x}^*) \in \mathcal{Y}$, $\mathcal{B}$ aborts the game. If the game continues and $b = 0$, $\mathcal{B}$ computes $\mathbf{H}_{\mathcal{U}_{\mathbf{x}^*}} := \text{EvalFX}(\mathcal{U}_{\mathbf{x}^*}, \{\mathbf{B}_i\}_{i \in L})$, and outputs $\lceil (\mathbf{b}^T \cdot \mathbf{H}_{\mathcal{U}_{\mathbf{x}^*}} + \mathbf{c}_{y^*}^T) \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \rceil$ ($\mathbf{c}_{y^*}$ follows from $\mathbf{c}_y$ as explained above). If $b = 1$, $\mathcal{B}$ samples $\mathbf{r}_{\mathbf{x}^*} \leftarrow \mathbb{Z}_q^m$ and outputs $\lceil \mathbf{r}_{\mathbf{x}^*}^T \rceil$.

If the LWE challenger sets $\mathbf{r}$ such that $\mathbf{r}^T = \mathbf{s}^T \cdot [\mathbf{B}_1 | \dots | \mathbf{B}_L | \mathbf{A}_0 | \mathbf{A}_1] + \mathbf{e}^T$ by sampling fresh $\mathbf{s} \leftarrow \mathbb{Z}_q^n, \mathbf{e} \leftarrow \chi_0^{(L+2)m}$ whenever $\mathcal{B}$ queries it, then the above game is exactly the same as Hybrid 2.i.1 in the view of $\mathcal{A}$. Otherwise, the above game is the same as Hybrid 2.i.2. Thus if $\text{Adv}_{\mathcal{A}}^{2.i.1} - \text{Adv}_{\mathcal{A}}^{2.i.2}$ is non-negligible, $\mathcal{B}$ breaks the LWE security game with non-negligible advantage. $\square$

Lemma 6.8. For all $i \in [\ell' - 1]$ and for any adversary $\mathcal{A}$ of unbounded running time, it holds that $\text{Adv}_{\mathcal{A}}^{2.i.2} - \text{Adv}_{\mathcal{A}}^{2.(i+1).0} = 0$.

Proof. In Hybrid 2.(i+1).0 we defined $\mathbf{c}_y$ for both constrained key queries and evaluation queries as the following:

$$\mathbf{c}_y^T = \mathbf{s}_{\mathbf{y}[[i]]}^T \cdot \mathbf{A}_{\mathbf{y}[i]} \prod_{j=i+2}^{\ell'} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{y}[j]}) + \sum_{j=1}^i \left(\mathbf{r}_{\mathbf{y}[[j]]}^T \prod_{k=j+1}^{\ell'} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{y}[k]}) \right) + \mathbf{e}_y^T,$$

where $\mathbf{s}_{\mathbf{y}[[i]]} \leftarrow \mathbb{Z}_q^n$ and $\mathbf{r}_{\mathbf{y}[[j]]} \leftarrow \mathbb{Z}_q^m$ for all $j \in [i]$. It implies that

$$\mathbf{c}_y^T = (\mathbf{s}_{\mathbf{y}[[i]]}^T \cdot \mathbf{G} + \mathbf{r}_{\mathbf{y}[[i]]}^T) \prod_{j=i+1}^{\ell'} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{y}[j]}) + \sum_{j=1}^{i-1} \left(\mathbf{r}_{\mathbf{y}[[j]]}^T \prod_{k=j+1}^{\ell'} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{y}[k]}) \right) + \mathbf{e}_y^T.$$

Recall that in Hybrid 2.i.2, $\mathbf{c}_y$ was defined as the following for $\mathbf{r}_{\mathbf{y}[[j]]} \leftarrow \mathbb{Z}_q^m$:

$$\mathbf{c}_y^T = \mathbf{r}_{\mathbf{y}[[i]]}^T \prod_{j=i+1}^{\ell'} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{y}[j]}) + \sum_{j=1}^{i-1} \left(\mathbf{r}_{\mathbf{y}[[j]]}^T \prod_{k=j+1}^{\ell'} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{y}[k]}) \right) + \mathbf{e}_y^T$$

One could observe that Hybrid 2.i.2 and Hybrid 2.(i + 1).0 are identically distributed, because $\mathbf{s}_{\mathbf{y}[[i]]}^T \cdot \mathbf{G} + \mathbf{r}_{\mathbf{y}[[i]]}^T$ has the same distribution as $\mathbf{r}_{\mathbf{y}[[i]]}^T$. The rest of the Hybrid 2.i.2 and Hybrid 2.(i + 1).0 are all identical, thus our claim follows. $\square$

Lemma 6.9. For any adversary $\mathcal{A}$ of unbounded running time, there exists a negligible function $\text{negl}(\cdot)$, such that $\text{Adv}_{\mathcal{A}}^{2.\ell'.2} - \text{Adv}_{\mathcal{A}}^3 \leq \text{negl}(\lambda)$.

Proof. The only difference between Hybrid 2. ℓ .2 and Hybrid 3 is that $\mathbf{c}_{\mathbf{y}}^T$ of each evaluation queries is set as $\mathbf{r}_{\mathbf{y}}^T + \mathbf{e}_{\mathbf{y}}^T$, where $\mathbf{r}_{\mathbf{y}}$ is sampled from $\mathbb{Z}_q^m$ and is fixed for the same $\mathbf{y}$ across all the evaluation queries and $\mathbf{e}_{\mathbf{y}}^T \leftarrow \chi_1^m$ is freshly generated for each evaluation query. Since $\|\mathbf{e}_{\mathbf{y}}^T \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x}))\|_{\infty} \leq \sigma' \cdot m^{\ell} \ll q/p$, Hybrid 2. ℓ .2 and Hybrid 3 are statistically close. $\square$

Lemma 6.10. For any unbounded adversary $\mathcal{A}$, $\text{Adv}_{\mathcal{A}}^3 - 1/2 = (Q + 1)\text{Adv}_{\mathcal{A}}^{3.0.0} - 1/2$.

Proof. In the view of $\mathcal{A}$, Hybrid 3 and 3.0.0 are identically distributed, because $\mathbf{c}_{\mathbf{y}}$ and $\mathbf{r}_{\mathbf{y}, \mathbf{x}[[0]]}$ are sampled from the same distribution. Note that the attacker wins only if the challenger successfully guessed index r , and the probability of guessing correctly is $1/(Q + 1)$ because the choice of r is independent from the output of $\mathcal{A}$. $\square$

Lemma 6.11. For all $i \in \{0, \dots, \ell - 1\}$, for any unbounded adversary $\mathcal{A}$, $\text{Adv}_{\mathcal{A}}^{3.i.0} - \text{Adv}_{\mathcal{A}}^{3.i.1} = 0$.

Proof. Since $\mathbf{r}_{\mathbf{y}, \mathbf{x}[[i]]}^T$ and $\mathbf{s}_{\mathbf{y}, \mathbf{x}[[i]]}^T \cdot \mathbf{G} + \mathbf{r}_{\mathbf{y}, \mathbf{x}[[i]]}^T$ are identically distributed for uniformly random $\mathbf{r}_{\mathbf{y}, \mathbf{x}[[i]]}$, our claim follows for $i \in [\ell - 1]$. The same analysis applies $i = 0$, and our claim holds. $\square$

Lemma 6.12. For all $i \in \{0, \dots, \ell - 1\}$, for any unbounded adversary $\mathcal{A}$, there exists a negligible function $\text{negl}(\cdot)$ such that $\text{Adv}_{\mathcal{A}}^{3.i.1} - \text{Adv}_{\mathcal{A}}^{3.i.2} \leq \text{negl}(\lambda)$.

Proof. To prove our claim, we only need to show that $\left\| \mathbf{e}_{\mathbf{y}, \mathbf{x}[[i+1]]}^T \cdot \prod_{j=i+2}^{\ell} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{x}[j]}) \right\|_{\infty} \ll q/p$ during the evaluation query and $\left\| \mathbf{e}_{\mathbf{y}^*, \mathbf{x}^*[[i+1]]}^T \cdot \prod_{j=i+2}^{\ell} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{x}^*[j]}) \right\|_{\infty} \ll q/p$ for the challenge query, where both $\mathbf{e}_{\mathbf{y}, \mathbf{x}[[i+1]]}$ and $\mathbf{e}_{\mathbf{y}^*, \mathbf{x}^*[[i+1]]}$ are sampled from χ_0^m . By truncated discrete gaussian distribution and our parameter setting, we have $\left\| \mathbf{e}_{\mathbf{y}, \mathbf{x}[[i+1]]}^T \cdot \prod_{j=i+2}^{\ell} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{x}[j]}) \right\|_{\infty} \leq \sigma \cdot m \cdot m^{\ell} \ll q/p$, and the same analysis applies to the challenge query. $\square$

Lemma 6.13. Assuming that $\text{GLWE}_{n, 2m, q, \chi_0}$ holds, then for any PPT adversary $\mathcal{A}$, there exists a negligible function $\text{negl}(\cdot)$ such that $\text{Adv}_{\mathcal{A}}^{3.i.2} - \text{Adv}_{\mathcal{A}}^{3.(i+1).0} \leq \text{negl}(\lambda)$ for all $i \in \{0, \dots, \ell - 1\}$.

Proof. Assume that there exists a polynomial time adversary $\mathcal{A}$, such that $\text{Adv}_{\mathcal{A}}^{3.i.1} - \text{Adv}_{\mathcal{A}}^{3.i.2} \geq \eta(\lambda)$ for some non-negligible function $\eta(\cdot)$ for any $i \in [\ell - 1]$, we design a reduction algorithm $\mathcal{B}$ that breaks the underlying LWE assumption.

The LWE challenger starts by outputting $\mathbf{A} \in \mathbb{Z}_q^{n \times 2m}$. The reduction algorithm $\mathcal{B}$ sets $\mathbf{A}_0$ and $\mathbf{A}_1$ such that $\mathbf{A} = [\mathbf{A}_0 | \mathbf{A}_1]$ for $\mathbf{A}_0, \mathbf{A}_1 \in \mathbb{Z}_q^{n \times m}$. Next, $\mathcal{B}$ tosses a coin $b \leftarrow \{0, 1\}$, samples $r \leftarrow [Q + 1]$, and samples $\mathbf{b} \leftarrow \mathbb{Z}_q^{Lm}$. Then, the attacker $\mathcal{A}$ is allowed polynomial constrained key queries and evaluation queries. For each constrained key query, $\mathcal{A}$ queries $\mathbf{y}$. If $\mathbf{y}^*$ has already been set and $\mathbf{y} = \mathbf{y}^*$, $\mathcal{B}$ aborts the game. Otherwise, $\mathcal{B}$ samples $\mathbf{c}_{\mathbf{y}} \leftarrow \mathbb{Z}_q^m$ if it was not sampled before. Next, $\mathcal{B}$ samples fresh $\mathbf{e} \leftarrow \chi_1^{Lm}$, $\mathbf{e}_{\mathbf{y}} \leftarrow \chi_1^m$. It outputs $(\mathbf{b}^T + \mathbf{e}^T, \mathbf{c}_{\mathbf{y}}^T + \mathbf{e}_{\mathbf{y}}^T)$. $\mathcal{B}$ also adds $\mathbf{y}$ to $\mathcal{Y}$. During each evaluation query, $\mathcal{A}$ outputs $\mathbf{x}$. If it is the r -th evaluation query, and $f(\mathbf{x}) \notin \mathcal{Y}$, and $f(\mathbf{x}) \neq f(\mathbf{x}')$ for all $\mathbf{x}' \in \mathcal{X}$, it sets $\mathbf{y}^*$ as $f(\mathbf{x})$. During the challenge query, $\mathcal{A}$ outputs $\mathbf{x}^*$. If

$\mathbf{x}^* \in \mathcal{X}$ or $f(\mathbf{x}^*) \in \mathcal{Y}$, the attacker fails and the game aborts. If $f(\mathbf{x}^*) = f(\mathbf{x})$ for any $\mathbf{x} \in \mathcal{X}$ and $f(\mathbf{x}^*) \neq \mathbf{y}^*$, $\mathcal{B}$ artificially aborts the game. Next, we combine the cases of a evaluation query where it is the evaluation query such that $f(\mathbf{x}) = \mathbf{y}^*$, and the challenge query where the game is not aborted. For ease of notation, we simply let $\mathbf{x}$ denote $\mathbf{x}^*$ in the challenge query and let $\mathbf{y}$ be $f(\mathbf{x})$. Next, we define $\mathbf{x}[[i, 0]]$ as $\begin{bmatrix} \mathbf{x}[[i]] \\ 0 \end{bmatrix}$ and $\mathbf{x}[[i, 1]]$ as $\begin{bmatrix} \mathbf{x}[[i]] \\ 1 \end{bmatrix}$. Note that for $i = 0$, we consider $\mathbf{x}[[i]]$ as the empty string. If $\mathcal{A}$ has never issued a query on any $\mathbf{x}'$ such that the first i components of $\mathbf{x}'$ match those of either $\mathbf{x}[[i, 0]]$ or $\mathbf{x}[[i, 1]]$ and $f(\mathbf{x}') = f(\mathbf{x})$, then $\mathcal{B}$ queries the LWE challenger and the challenger outputs $\mathbf{r} \in \mathbb{Z}_q^{2m}$. $\mathcal{B}$ then parses $\mathbf{r}$ as $\begin{bmatrix} \mathbf{r}_{\mathbf{y}, \mathbf{x}[[i, 0]]} \\ \mathbf{r}_{\mathbf{y}, \mathbf{x}[[i, 1]]} \end{bmatrix}$ such that $\mathbf{r}_{\mathbf{y}, \mathbf{x}[[i, 0]]}, \mathbf{r}_{\mathbf{y}, \mathbf{x}[[i, 1]]} \in \mathbb{Z}_q^m$. At this point, $\mathcal{B}$ has either obtained $\mathbf{r}_{\mathbf{y}, \mathbf{x}[[i+1]]}$ during the current evaluation query or received it during an earlier one. Next if $b = 0$, $\mathcal{B}$ sets $\mathbf{b}_{\mathbf{x}, f}$ such that $\mathbf{b}_{\mathbf{x}, f}^T = \mathbf{b}^T \cdot \mathbf{H}_{\mathcal{U}_{\mathbf{x}, f}} \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x}))$. $\mathcal{B}$ samples $\mathbf{r}_{\mathbf{y}} \leftarrow \mathbb{Z}_q^m$ and $\mathbf{r}_{\mathbf{y}, \mathbf{x}[[j]]} \leftarrow \mathbb{Z}_q^m$ for all $j \in [i]$ (if such $\mathbf{r}_{\mathbf{y}}$ or $\mathbf{r}_{\mathbf{y}, \mathbf{x}[[j]]}$ was not chosen). It returns

$$\left[\mathbf{b}_{\mathbf{x}, f}^T + \mathbf{r}_{\mathbf{y}, \mathbf{x}[[i+1]]}^T \prod_{j=i+2}^{\ell} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{x}[j]}) + \sum_{j=1}^i \left(\mathbf{r}_{\mathbf{y}, \mathbf{x}[[j]]}^T \prod_{k=j+1}^{\ell} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{x}[k]}) \right) + \mathbf{r}_{\mathbf{y}}^T \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \right].$$

Otherwise if it is a evaluation query that does not satisfy the above condition, $\mathcal{B}$ just samples $\mathbf{c}_{\mathbf{y}} \leftarrow \mathbb{Z}_q^m$ (if such $\mathbf{c}_{\mathbf{y}}$ has not been sampled before) and outputs $\lceil (\mathbf{b}^T \cdot \mathbf{H}_{\mathcal{U}_{\mathbf{x}, f}} + \mathbf{c}_{\mathbf{y}}^T) \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \rceil$.

If the LWE challenger always sets $\mathbf{r}$ so that $\mathbf{r}^T = \mathbf{s}^T \cdot [\mathbf{A}_0 | \mathbf{A}_1] + \mathbf{e}^T$ by sampling fresh $\mathbf{s} \leftarrow \mathbb{Z}_q^n, \mathbf{e} \leftarrow \mathbb{Z}^2 m_q$ whenever $\mathcal{B}$ queries it, then the above game is exactly identical to Hybrid 3.i.2 in the view of $\mathcal{A}$. Otherwise the LWE challenger just randomly samples $\mathbf{r}$, and the above game would be the same as Hybrid 3.(i+1).0. Thus, if $\text{Adv}_{\mathcal{A}}^{3.i.2} - \text{Adv}_{\mathcal{A}}^{3.(i+1).0}$ is non-negligible for any $i \in [\ell - 1]$, then $\mathcal{B}$ breaks the LWE security game with non-negligible advantage. We omit the proof for the case where $i = 0$ as is nearly identical to the above reduction. $\square$

Lemma 6.14. For any unbounded adversary $\mathcal{A}$, $\text{Adv}_{\mathcal{A}}^{3.\ell.0} - \text{Adv}_{\mathcal{A}}^4 = 0$.

Proof. The above holds, because $\mathbf{r}_{\mathbf{x}, \mathbf{y}}$ in Hybrid 3.ℓ.0 is identically distributed as $\mathbf{r}_{\mathbf{x}}$ and $\mathbf{r}_{\mathbf{x}^*}$ in Hybrid 4. $\square$

The adversary has at most $\frac{1}{2(Q+1)}$ advantage in Hybrid 4 because as long as the game is not aborted, the adversary's advantage is at most 1/2, and the probability that the challenger does not artificially aborts the game is $1/(Q+1)$. Thus combining Theorems 6.3 to 6.14, we conclude that Theorem 6.2 holds.

7 Constraint-Hiding CPRFs for Compute-&-Compare Constraints

In this section, we additionally achieve constraint-hiding for the constrained PRFs of the above circuit class. The construction relies on the [GSW13] fully homomorphic encryption design and the definition of `MtrxEval`, `AllButLastRow`, `LastRow` are as explained in Section 3.1. The design applies the “dual use” technique in [BTVW17] to achieve constraint hiding. We define the algorithms here and additionally define the stateful simulator for the constraint hiding experiment.

Let χ_0 and χ_1 be two truncated discrete gaussian distributions set as $\tilde{D}_{\mathbb{Z}, \sigma}$ and $\tilde{D}_{\mathbb{Z}, \sigma'}$, respectively. Fix any $\epsilon < 1/2$. Let n, m, q, σ, σ' be parameters with the following constraints:

- $n = \text{poly}(\lambda)$, $q \leq 2^{n^\epsilon}$, $m = \Omega(n \log q)$ (for LWE security)
- $\sigma' \geq \sigma \cdot m^{\ell'+1} \cdot 2^{\Omega(\lambda)}$ (for smudging)
- $\frac{q}{p} \geq \sigma' \cdot (L \cdot m^{\Omega(d)} + m^\ell + m^{\ell'+1}) \cdot 2^{\Omega(\lambda)}$ (for correctness/smudging)

One possible parameter setting is as follows:

$$n = 2\lambda(d + \ell + \ell')/\epsilon, m = n \log q, p = 2, q = 2^{n^\epsilon}, \sigma = n, \sigma' = \sigma \cdot m^{\ell'+1} \cdot 2^\lambda.$$

Setup($1^\lambda, 1^\ell, 1^{\ell'}, 1^d$): It takes as input security parameter λ , description length ℓ_f of constraint functions, and depth d . Let $L = \ell_f \cdot (n + 1)^2 \log^2 q$. It samples

$$\begin{aligned} \mathbf{B}, \mathbf{B}_1, \dots, \mathbf{B}_L &\leftarrow \mathbb{Z}_q^{n \times (m + \log q)}, \mathbf{s} \leftarrow \mathbb{Z}_q^n, \mathbf{A}_0, \mathbf{A}_1 \leftarrow \mathbb{Z}_q^{n \times m}, \\ \mathbf{R}_1, \dots, \mathbf{R}_{\ell_f} &\leftarrow \{0, 1\}^{(m + \log q) \times (m + \log q)}, \mathbf{e}_f \leftarrow \chi_0^{m + \log q}. \end{aligned}$$

It outputs

$$\text{pp} := (\mathbf{B}, \mathbf{B}_1, \dots, \mathbf{B}_L, \mathbf{A}_0, \mathbf{A}_1), \text{msk} := (\mathbf{s}, \mathbf{R}_1, \dots, \mathbf{R}_{\ell_f}, \mathbf{e}_f).$$

Eval($\text{msk}, \mathbf{x}$): It takes as input master secret key msk and input $\mathbf{x}$. We define circuit $\mathcal{U}_{\mathbf{x}}$ as the universal circuit that takes as input a description of a function f and outputs $\begin{bmatrix} \mathbf{A}(f(\mathbf{x})) \\ \mathbf{0} \end{bmatrix} \in \mathbb{Z}_q^{(n+1) \times m}$. Let circuit $\widehat{\mathcal{U}}_{\mathbf{x}}$ be the circuit that takes as input ct_f (the encryption of function f) and outputs $\text{AllButLastRow}(\mathcal{U}_{\mathbf{x}}, \text{ct}_f)$ (AllButLastRow as defined in Section 3.1).

$$\mathbf{H}_{\widehat{\mathcal{U}}_{\mathbf{x}}} := \text{EvalF}(\widehat{\mathcal{U}}_{\mathbf{x}}, \{\mathbf{B}_i\}_{i \in [L]}), \mathbf{B}_{\widehat{\mathcal{U}}_{\mathbf{x}}} := [\mathbf{B}_1 | \dots | \mathbf{B}_L] \cdot \mathbf{H}_{\widehat{\mathcal{U}}_{\mathbf{x}}}.$$

The PRF output is $\left\lceil \mathbf{s}^T \cdot \mathbf{B}_{\widehat{\mathcal{U}}_{\mathbf{x}}} \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \right\rceil$.

Constrain($\text{msk}, C_{f, \mathbf{y}}$): It takes as input a circuit $C_{f, \mathbf{y}}$ with respect to function f and target $\mathbf{y}$. It then encrypts function f bit by bit using fully homomorphic encryption, such that for all $i \in [\ell_f]$,

$$\text{ct}_i = \begin{bmatrix} \mathbf{B} \\ \mathbf{s}^T \mathbf{B} + \mathbf{e}_f^T \end{bmatrix} \mathbf{R}_i + f_i \mathbf{G}.$$

Let ct denote $(\text{ct}_1, \dots, \text{ct}_{\ell_f})$. Next, for all $i \in [L]$, let ψ_i denote the i -th bit of the binary representation of ct . It samples $\mathbf{e} \leftarrow \chi_1^{Lm}$, $\mathbf{e}_y \leftarrow \chi_1^m$ and outputs

$$\text{sk} := (\text{ct}, \mathbf{s}^T \cdot [\mathbf{B}_1 - \psi_1 \mathbf{G} | \dots | \mathbf{B}_L - \psi_L \mathbf{G}] + \mathbf{e}^T, \mathbf{s}^T \cdot \mathbf{A}(\mathbf{y}) + \mathbf{e}_y^T).$$

CEval($\text{sk}, \mathbf{x}$): It parses sk as $(\text{ct}, \mathbf{b}^T, \mathbf{c}_y^T)$. Again let circuit $\mathcal{U}_{\mathbf{x}}$ be the circuit which takes as input a description of a function f and outputs $\mathbf{A}(f(x))$. Let circuit $\widehat{\mathcal{U}}_{\mathbf{x}}$ be the circuit that takes as input ct (the encryption of function f) and outputs $\text{AllButLastRow}(\mathcal{U}_{\mathbf{x}}, \text{ct})$. It computes $\mathbf{H}_{\widehat{\mathcal{U}}_{\mathbf{x}, \text{ct}}} := \text{EvalFX}(\widehat{\mathcal{U}}_{\mathbf{x}, \text{ct}}, \{\mathbf{B}_i\}_{i \in [L]})$. Next, it computes the PRF output as

$$\left\lceil \left(\mathbf{b}^T \cdot \mathbf{H}_{\widehat{\mathcal{U}}_{\mathbf{x}, \text{ct}}} + \text{LastRow}(\mathcal{U}_{\mathbf{x}}, \text{ct}) + \mathbf{c}_y^T \right) \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \right\rceil.$$

7.1 Correctness

By Theorem 3.6, for $\mathbf{H}_{\hat{\mathcal{U}}_{\mathbf{x}}} = \text{EvalF}(\hat{\mathcal{U}}_{\mathbf{x}}, \{\mathbf{B}_i\}_{i \in L})$ and $\mathbf{H}_{\hat{\mathcal{U}}_{\mathbf{x}, \text{ct}}} = \text{EvalFX}(\hat{\mathcal{U}}_{\mathbf{x}, \text{ct}}, \{\mathbf{B}_i\}_{i \in L})$, we have $\|\mathbf{H}_{\hat{\mathcal{U}}_{\mathbf{x}}}\|_{\infty}, \|\mathbf{H}_{\hat{\mathcal{U}}_{\mathbf{x}, \text{ct}}}\|_{\infty} \leq m^{O(d)}$. Next, consider the constrained evaluation algorithm:

$$\text{CEval}(\text{sk}, \mathbf{x}) = \left\lceil \left(\mathbf{b}^T \cdot \mathbf{H}_{\hat{\mathcal{U}}_{\mathbf{x}, \text{ct}}} + \text{LastRow}(\mathcal{U}_{\mathbf{x}}, \text{ct}) + \mathbf{c}_{\mathbf{y}}^T \right) \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \right\rceil, \text{ where}$$

$$\mathbf{b}^T \cdot \mathbf{H}_{\hat{\mathcal{U}}_{\mathbf{x}, \text{ct}}} = (\mathbf{s}^T \cdot [\mathbf{B}_1 - \psi_1 \mathbf{G} \mid \dots \mid \mathbf{B}_L - \psi_L \mathbf{G}] + \mathbf{e}^T) \cdot \mathbf{H}_{\hat{\mathcal{U}}_{\mathbf{x}, \text{ct}}} = \mathbf{s}^T \left([\mathbf{B}_1 \mid \dots \mid \mathbf{B}_L] \cdot \mathbf{H}_{\hat{\mathcal{U}}_{\mathbf{x}}} - \text{AllButLastRow}(\mathcal{U}_{\mathbf{x}}, \text{ct}) \right) + \mathbf{e}^T \cdot \mathbf{H}_{\hat{\mathcal{U}}_{\mathbf{x}, \text{ct}}}.$$

Let $\mathbf{y}$ be $f(\mathbf{x})$ and define $\mathbf{c}_{\mathbf{y}}$ as $\mathbf{s}^T \cdot \mathbf{A}(\mathbf{y})$. One could rewrite the output of CEval as:

$$\left\lceil \left(\mathbf{s}^T [\mathbf{B}_1 \mid \dots \mid \mathbf{B}_L] \cdot \mathbf{H}_{\hat{\mathcal{U}}_{\mathbf{x}}} - [\mathbf{s}^T \mid -1] \cdot \text{MtrxEval}(\mathcal{U}_{\mathbf{x}}, \text{ct}) + \mathbf{c}_{\mathbf{y}}^T + \mathbf{e}^T \cdot \mathbf{H}_{\hat{\mathcal{U}}_{\mathbf{x}, \text{ct}}} \right) \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \right\rceil$$

Observe that

$$\left\| -[\mathbf{s}^T \mid -1] \cdot \text{MtrxEval}(\mathcal{U}_{\mathbf{x}}, \text{ct}) + \mathbf{c}_{\mathbf{y}}^T \right\|_{\infty} = \left\| -[\mathbf{s}^T \mid -1] \cdot \text{MtrxEval}(\mathcal{U}_{\mathbf{x}}, \text{ct}) + [\mathbf{s}^T \mid -1] \cdot \begin{bmatrix} \mathbf{A}(\mathbf{y}) \\ \mathbf{0} \end{bmatrix} + \mathbf{e}_{\mathbf{y}}^T \right\|_{\infty}.$$

By Theorem 3.7, when $\mathcal{U}_{\mathbf{x}}(f) = \begin{bmatrix} \mathbf{A}(\mathbf{y}) \\ \mathbf{0} \end{bmatrix}$, it turns out that the above norm is bounded by the norm of error $\mathbf{e}_f$, $\mathbf{e}_{\mathbf{y}}$, and depth $O(d)$ of the $\hat{\mathcal{U}}_{\mathbf{x}}$, where

$$\left\| (-[\mathbf{s}^T \mid -1] \cdot \text{MtrxEval}(\mathcal{U}_{\mathbf{x}}, \text{ct}) + \mathbf{c}_{\mathbf{y}}^T) \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \right\|_{\infty} \leq (\sigma \cdot m^{O(d)} + \sigma' \cdot m) \cdot m \ll q/p.$$

Then with high probability the output of CEval is equivalent to

$$\left\lceil \left(\mathbf{s}^T \cdot [\mathbf{B}_1 \mid \dots \mid \mathbf{B}_L] \cdot \mathbf{H}_{\hat{\mathcal{U}}_{\mathbf{x}}} + \mathbf{e}^T \cdot \mathbf{H}_{\hat{\mathcal{U}}_{\mathbf{x}, \text{ct}}} \right) \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \right\rceil$$

By our parameter setting,

$$\left\| \mathbf{e}^T \cdot \mathbf{H}_{\hat{\mathcal{U}}_{\mathbf{x}, \text{ct}}} \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \right\|_{\infty} \leq (\sigma' \cdot L \cdot m) \cdot m.$$

Such norm bound is exponentially smaller than q/p . We conclude that $\text{CEval}(\text{sk}, \mathbf{x})$ gives the same output as $\text{Eval}(\text{msk}, \mathbf{x})$ with high probability.

7.2 Purified Constraint Privacy

The constraint hiding property is nearly identical to Theorem 5.2, except that we require the attacker to specify the function f_0 and f_1 at the beginning of the experiment and before msk is set, and all of the circuits must lie in the compute-&-compare class with respect to f_0 or f_1 . We define oracle $O_b(\text{msk}, C_0, C_1)$ as the following: It takes as input circuits C_0, C_1 of the form $C_{f_0, \mathbf{y}_0}, C_{f_1, \mathbf{y}_1}$ and outputs $\text{Constrain}(\text{msk}, C_b)$.

Definition 7.1 (Semi-Adaptive Purified Constraint Hiding). A CPRF $\mathcal{F}$ satisfies (collusion-resistant) semi-adaptive constraint hiding for compute-&-compare circuits, if for any stateful admissible PPT adversary $\mathcal{A}$, there exists a negligible function $\text{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, the following holds:

$$\Pr \left[\mathcal{A}^{\text{Eval}(\text{msk}, \cdot), O_b(\text{msk}, \cdot, \cdot)} = b : \begin{array}{l} (\ell, \ell', L, d, f_0, f_1) \leftarrow \mathcal{A}(1^\lambda); b \leftarrow \{0, 1\} \\ \text{msk} \leftarrow \text{Setup}(1^\lambda, 1^\ell, 1^{\ell'}, 1^L, 1^d) \end{array} \right] \leq 1/2 + \text{negl}(\lambda).$$

Consider that $\mathcal{A}$ queries $(C_{i,0}, C_{i,1})$ for each constrained key query, then $\mathcal{A}$ is an admissible adversary if: (i) for all i , each $C_{i,\beta}$ queried by $\mathcal{A}$ to the **Constrain** oracle is of the form $C_{f_\beta, \mathbf{y}_{i,\beta}}$, (ii) $\mathbf{y}_{i,\beta} \neq \mathbf{y}_{j,\beta}$ for any $i \neq j, \beta \in \{0, 1\}$, and (iii) for all $i, \beta \in \{0, 1\}$, $C_{i,\beta}(\mathbf{x}) = 0$.

Theorem 7.2. Assuming that $\text{GLWE}_{n, L(m+\log q)+2m, q, \chi}$ assumption (Theorem 3.1) holds, then our design satisfies semi-adaptive compute-&-compare constraint-hiding property.

Before the we present hybrid arguments, we define the following simulation algorithm which simulates the constrained key queries. And in the proof, we achieve the slightly stronger simulation based constraint-hiding definition which implies the indistinguishability based constraint-hiding definition.

Sim $(1^\lambda, 1^\ell, 1^{\ell'}, 1^L, 1^d)$: At the beginning of the constraint-hiding experiment, the simulator samples $\mathbf{B}'_i \leftarrow \mathbb{Z}_q^{(n+1) \times (m+\log q)}$ for all $i \in [\ell_f]$, $\mathbf{b}' \leftarrow \mathbb{Z}_q^{Lm}$. Next, to simulate each constrained key query, it sets $\text{ct}_i := \mathbf{B}'_i$ for all $i \in [\ell_f]$ (i.e. $\text{ct}_1, \dots, \text{ct}_{\ell_f}$ are random matrices instead of encryption of function f). It samples dummy random string $\mathbf{r}_\mathbf{y} \leftarrow \mathbb{Z}_q^m$. It also samples $\mathbf{e} \leftarrow \chi_1^{L(m+\log q)}$ and outputs $\text{sk} := (\text{ct}, \mathbf{b}'^T + \mathbf{e}^T, \mathbf{r}_\mathbf{y}^T)$.

7.2.1 Sequence of Hybrid Games

In the following sequence of hybrid games, Hybrid 0 corresponds to the simulation experiment where the constrained key queries are answered honestly if $b = 0$ and are simulated if $b = 1$. Such experiment implies the indistinguishability based constraint hiding. Next, Hybrid 1 will ensure that the secret keys are independent from the encryption of function f . Hybrid 2 will answer each evaluation query following the constrained evaluation algorithm. Hybrid 3 will set each constrained key uniformly at random if $b = 0$. Hybrid 4 will then simulate the evaluation queries by generating random outputs.

Hybrid 0: This is the simulation based constraint hiding experiment. The adversary outputs a single function f at the beginning. Next if $b = 0$, the challenger answers $\mathcal{A}$'s constrained key queries honestly. Otherwise for $b = 1$, it answers $\mathcal{A}$'s queries using the simulator.

1. The attacker outputs the constraining function f .
2. The challenger starts by running the setup algorithm. It samples $\mathbf{B}, \mathbf{B}_1, \dots, \mathbf{B}_L \leftarrow \mathbb{Z}_q^{n \times (m+\log q)}$, $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, $\mathbf{A}_0, \mathbf{A}_1 \leftarrow \mathbb{Z}_q^{n \times m}$, $\mathbf{R}_1, \dots, \mathbf{R}_{\ell_f} \leftarrow \{0, 1\}^{(m+\log q) \times (m+\log q)}$, $\mathbf{e}_f \leftarrow \chi_0^{m+\log q}$. The challenger outputs $\mathbf{B}, \mathbf{B}_1, \dots, \mathbf{B}_L, \mathbf{A}_0, \mathbf{A}_1$. It samples a bit $b \leftarrow \{0, 1\}$.
3. Before the queries begin, it samples $\mathbf{B}'_i \leftarrow \mathbb{Z}_q^{(n+1) \times (m+\log q)}$ for all $i \in [\ell_f]$, $\mathbf{b}' \leftarrow \mathbb{Z}_q^{Lm}$ by definition of the simulator.

4. The attacker is allowed to make polynomial evaluation and constrained key queries.

- During each constrained key query, the attacker outputs $\mathbf{y}$. If $\mathbf{y} \in \mathcal{Y}$, or there exists any $\mathbf{x} \in \mathcal{X}$ such that $f(\mathbf{x}) = \mathbf{y}$, the game aborts and the attacker fails. Otherwise, the challenger adds the target $\mathbf{y}$ to $\mathcal{Y}$. There are two cases:

The first case is that $b = 0$. In this case, the challenger responds the query following $\text{Constrain}(\text{msk}, C_{f,\mathbf{y}})$: The challenger then encrypts function f bit by bit, such that for all

$i \in [\ell_f]$, $\text{ct}_i = \begin{bmatrix} \mathbf{B} \\ \mathbf{s}^T \mathbf{B} + \mathbf{e}_f^T \end{bmatrix} \mathbf{R}_i + f_i \mathbf{G}$. Let ct denote $(\text{ct}_1, \dots, \text{ct}_{\ell_f})$. Next, for all $i \in [L]$, let

ψ_i denote the i -th bit of the binary representation of ct . It samples $\mathbf{e} \leftarrow \chi_1^{L(m+\log q)}$, $\mathbf{e}_y \leftarrow \chi_1^m$ and outputs sk as $(\text{ct}, \mathbf{s}^T \cdot [\mathbf{B}_1 - \psi_1 \mathbf{G}] \dots [\mathbf{B}_L - \psi_L \mathbf{G}] + \mathbf{e}^T, \mathbf{s}^T \cdot \mathbf{A}(\mathbf{y}) + \mathbf{e}_y^T)$.

The second case is that $b = 1$: The challenger sets $\text{ct}_i = \mathbf{B}'_i$. Next, it samples a random strings $\mathbf{r}_y \leftarrow \mathbb{Z}_q^m$. It also samples $\mathbf{e} \leftarrow \chi_1^{L(m+\log q)}$ and outputs $\text{sk} := (\text{ct}, \mathbf{b}^T + \mathbf{e}^T, \mathbf{r}_y^T)$.

- During each evaluation query, the attacker outputs $\mathbf{x}$. If $f(\mathbf{x}) \in \mathcal{Y}$, the game aborts and the attacker fails. Otherwise, it adds $\mathbf{x}$ to set $\mathcal{X}$. The challenger responds with $\text{Eval}(\text{sk}, \mathbf{x})$: It computes $\mathbf{H}_{\hat{\mathcal{U}}_{\mathbf{x}}} := \text{EvalF}(\hat{\mathcal{U}}_{\mathbf{x}}, \{\mathbf{B}_i\}_{i \in L})$, $\mathbf{B}_{\hat{\mathcal{U}}_{\mathbf{x}}} := [\mathbf{B}_1 | \dots | \mathbf{B}_L] \cdot \mathbf{H}_{\hat{\mathcal{U}}_{\mathbf{x}}}$, and outputs $\lceil \mathbf{s}^T \cdot \mathbf{B}_{\hat{\mathcal{U}}_{\mathbf{x}}} \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \rceil$.

5. The attacker outputs b' and wins if and only if $b' = b$.

Hybrid 1: Hybrid 1 is the same as Hybrid 0, and the only difference is highlighted: In Hybrid 1, the challenger sets pp as $\mathbf{B}_1 + \psi_1 \mathbf{G}, \dots, \mathbf{B}_L + \psi_L \mathbf{G}, \mathbf{A}_0, \mathbf{A}_1$, where $(\psi_1, \dots, \psi_L)$ is the encryption of function f . As a result, constrained keys become $(\mathbf{s}^T \cdot [\mathbf{B}_1 | \dots | \mathbf{B}_L] + \mathbf{e}^T, \mathbf{s}^T \cdot \mathbf{A}(\mathbf{y}) + \mathbf{e}_y^T)$ and are independent from function f .

2. The challenger samples $\mathbf{B}, \mathbf{B}_1, \dots, \mathbf{B}_L \leftarrow \mathbb{Z}_q^{n \times (m+\log q)}$, $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, $\mathbf{A}_0, \mathbf{A}_1 \leftarrow \mathbb{Z}_q^{n \times m}$, $\mathbf{R}_1, \dots, \mathbf{R}_{\ell_f} \leftarrow \{0, 1\}^{(m+\log q) \times (m+\log q)}$, $\mathbf{e}_f \leftarrow \chi_0^{m+\log q}$. The challenger outputs $\mathbf{B}, \mathbf{B}_1 + \psi_1 \mathbf{G}, \dots, \mathbf{B}_L + \psi_L \mathbf{G}, \mathbf{A}_0, \mathbf{A}_1$. It samples a bit $b \leftarrow \{0, 1\}$.

4. The attacker is allowed to make polynomial evaluation and constrained key queries.

- The constrained key query is nearly identical to Hybrid 0, except that for $b = 0$, the challenger sets sk as $(\text{ct}, \mathbf{s}^T \cdot [\mathbf{B}_1 | \dots | \mathbf{B}_L] + \mathbf{e}^T, \mathbf{s}^T \cdot \mathbf{A}(\mathbf{y}) + \mathbf{e}_y^T)$.
- The evaluation query is the same as Hybrid 0, except that the challenger sets $\mathbf{B}_{\hat{\mathcal{U}}_{\mathbf{x}}} := [\mathbf{B}_1 + \psi_1 \mathbf{G} | \dots | \mathbf{B}_L + \psi_L \mathbf{G}] \cdot \mathbf{H}_{\hat{\mathcal{U}}_{\mathbf{x}}}$.

Hybrid 2: It is the same as Hybrid 1, except that the challenger answers evaluation queries by generating a constrained key and then following the process of the CEval algorithm:

4. The attacker is allowed to make polynomial evaluation and constrained key queries.

- The constrained key queries remain the same as Hybrid 1.
- During each evaluation query, the attacker outputs $\mathbf{x}$. If $f(\mathbf{x}) \in \mathcal{Y}$, the game aborts and the attacker fails. Otherwise, it adds $\mathbf{x}$ to set $\mathcal{X}$. **Next, the challenger computes**

$\mathbf{H}_{\hat{\mathcal{U}}_{\mathbf{x}, \text{ct}}} := \text{EvalFX}(\hat{\mathcal{U}}_{\mathbf{x}}, \text{ct}, \{\mathbf{B}_i\}_{i \in L})$. Let $\mathbf{y}$ be the output of $f(\mathbf{x})$, the challenger samples $\mathbf{e} \leftarrow \chi_1^{Lm}$, $\mathbf{e}_y \leftarrow \chi_1^m$. It outputs

$$\left[\left((\mathbf{s}^T \cdot [\mathbf{B}_1] \dots [\mathbf{B}_L] + \mathbf{e}^T) \cdot \mathbf{H}_{\hat{\mathcal{U}}_{\mathbf{x}, \text{ct}}} + \text{LastRow}(\mathcal{U}_{\mathbf{x}}, \text{ct}) + \mathbf{s}^T \cdot \mathbf{A}(\mathbf{y}) + \mathbf{e}_y^T \right) \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \right].$$

We then introduce the intermediate hybrids between Hybrid 2 and 3.

Hybrid 2.i.0 ($1 \leq i \leq \ell'$): It is the same as Hybrid 2, except for the highlighted sentences: For each $\mathbf{y}$ during the constrained key queries, the challenger samples uniformly random $\mathbf{s}_{\mathbf{y}[[i-1]]}$ and $\mathbf{r}_{\mathbf{y}[[j]]}$ for $j \in [i-1]$ and computes $\mathbf{c}_{\mathbf{y}}$ by the equation explained below. During each evaluation query where $\mathcal{A}$ queries $\mathbf{x}$, let $\mathbf{y}$ be $f(\mathbf{x})$, the challenger computes $\mathbf{c}_{\mathbf{y}}$ by sampling uniformly random $\mathbf{s}_{\mathbf{y}[[i-1]]}$ and $\mathbf{r}_{\mathbf{y}[[j]]}$.

3. The attacker is allowed to make polynomial evaluation and constrained key queries.

- During each constrained key query, the attacker outputs $\mathbf{y}$. The case where $b = 1$ (where the challenger uses the simulator) remains the same as Hybrid 2. We only explain the changes where $b = 0$. If $\mathbf{y} \in \mathcal{Y}$, or there exists any $\mathbf{x} \in \mathcal{X}$ such that $f(\mathbf{x}) = \mathbf{y}$, the game aborts and the attacker fails. Otherwise, the challenger adds $\mathbf{y}$ to $\mathcal{Y}$. The challenger samples fresh $\mathbf{e}_y \leftarrow \chi_1^m$. Next, it takes out the first i bits of $\mathbf{y}$ as $\mathbf{y}_i$. It samples $\mathbf{s}_{\mathbf{y}[[i-1]]} \leftarrow \mathbb{Z}_q^n$ and $\mathbf{r}_{\mathbf{y}[[j]]} \leftarrow \mathbb{Z}_q^m$ for all $j \in [i-1]$ (if such $\mathbf{s}_{\mathbf{y}[[i]]}$ or $\mathbf{r}_{\mathbf{y}[[j]]}$ was not chosen in any of the previous queries). Next, the challenger samples fresh $\mathbf{e}_y \leftarrow \chi_1^m$. It sets $\mathbf{c}_{\mathbf{y}}$ such that

$$\mathbf{c}_{\mathbf{y}}^T = \mathbf{s}_{\mathbf{y}[[i-1]]}^T \cdot \mathbf{A}_{\mathbf{y}[i]} \prod_{j=i+1}^{\ell'} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{y}[j]}) + \sum_{j=1}^{i-1} \left(\mathbf{r}_{\mathbf{y}[[j]]}^T \prod_{k=j+1}^{\ell'} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{y}[k]}) \right) + \mathbf{e}_y^T.$$

After setting up $\mathbf{c}_{\mathbf{y}}$, the challenger sets $\mathbf{b}, \mathbf{b}_f$: If $i = 1$, it sets $\mathbf{b}, \mathbf{b}_f$ such that $\mathbf{b}^T = \mathbf{s}_{\mathbf{y}[[i-1]]}^T \cdot [\mathbf{B}_1] \dots [\mathbf{B}_L]$, $\mathbf{b}_f = \mathbf{s}_{i-1}^T \cdot \mathbf{B} + \mathbf{e}_f$, otherwise for $2 \leq i \leq \ell'$, the challenger samples $\mathbf{b} \leftarrow \mathbb{Z}_q^{L(m+\log q)}$, $\mathbf{b}_f \leftarrow \mathbb{Z}_q^{m+\log q}$ (if $\mathbf{b}, \mathbf{b}_f$ was not sampled in any previous query). Next, the challenger samples fresh $\mathbf{e} \leftarrow \chi_1^{L(m+\log q)}$. Then, it sets ct_i as $\text{ct}_i = \begin{bmatrix} \mathbf{B} \\ \mathbf{b}_f^T \end{bmatrix} \mathbf{R}_i + f_i \mathbf{G}$.

Let ct be $(\text{ct}_1, \dots, \text{ct}_{\ell_f})$. It outputs $(\text{ct}, \mathbf{b}^T + \mathbf{e}^T, \mathbf{c}_{\mathbf{y}}^T)$.

- During each evaluation query, the attacker outputs $\mathbf{x}$. If $f(\mathbf{x}) \in \mathcal{Y}$, the game aborts and the attacker fails. Otherwise, it adds $\mathbf{x}$ to set $\mathcal{X}$. The challenger computes $\mathbf{H}_{\hat{\mathcal{U}}_{\mathbf{x}, \text{ct}}} := \text{EvalFX}(\hat{\mathcal{U}}_{\mathbf{x}}, f, \{\mathbf{B}_i\}_{i \in L})$. It samples $\mathbf{s}_{\mathbf{y}[[i-1]]} \leftarrow \mathbb{Z}_q^n$ and $\mathbf{r}_{\mathbf{y}[[j]]} \leftarrow \mathbb{Z}_q^m$ for all $j \in [i-1]$ (if such $\mathbf{s}_{\mathbf{y}[[i]]}$ or $\mathbf{r}_{\mathbf{y}[[j]]}$ was not chosen). If $i = 1$, it sets $\mathbf{b}, \mathbf{b}_f$ such that $\mathbf{b}^T = \mathbf{s}_{\mathbf{y}[[i-1]]}^T \cdot [\mathbf{B}_1] \dots [\mathbf{B}_L]$, $\mathbf{b}_f = \mathbf{s}_{i-1}^T \cdot \mathbf{B} + \mathbf{e}_f$, otherwise for $2 \leq i \leq \ell'$, the challenger samples $\mathbf{b} \leftarrow \mathbb{Z}_q^{Lm}$, $\mathbf{b}_f \leftarrow \mathbb{Z}_q^{m+\log q}$ (if $\mathbf{b}, \mathbf{b}_f$ was not sampled before). Then, it sets ct_i as $\text{ct}_i = \begin{bmatrix} \mathbf{B} \\ \mathbf{b}_f^T \end{bmatrix} \mathbf{R}_i + f_i \mathbf{G}$. Let ct be $(\text{ct}_1, \dots, \text{ct}_{\ell_f})$. Next, the challenger samples fresh $\mathbf{e}_y \leftarrow \chi_1^m$. It sets $\mathbf{c}_{\mathbf{y}}$ such that

$$\mathbf{c}_{\mathbf{y}}^T = \mathbf{s}_{\mathbf{y}[[i-1]]}^T \cdot \mathbf{A}_{\mathbf{y}[i]} \prod_{j=i+1}^{\ell'} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{y}[j]}) + \sum_{j=1}^{i-1} \left(\mathbf{r}_{\mathbf{y}[[j]]}^T \prod_{k=j+1}^{\ell'} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{y}[k]}) \right) + \mathbf{e}_y^T.$$

It outputs $\left[\left(\mathbf{b}^T \cdot \mathbf{H}_{\widehat{\mathcal{U}}_{\mathbf{x}, \text{ct}}} + \text{LastRow}(\mathcal{U}_{\mathbf{x}, \text{ct}}) + \mathbf{c}_{\mathbf{y}}^T \right) \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \right]$.

Hybrid 2.i.1 ($1 \leq i \leq \ell'$): It is the same as Hybrid 2.i.0, except for the highlighted part: The challenger additionally samples LWE error $\mathbf{e}'_{\mathbf{y}[[i]]}$ (and $\mathbf{e}'$ for $i = 1$) from error distribution χ_0 , which are included in the computation of $\mathbf{c}_{\mathbf{y}}$ and $\mathbf{b}$.

3. The attacker is allowed to make polynomial evaluation and constrained key queries.

- During each constrained key query, the attacker outputs $\mathbf{y}$. The case where $b = 1$ (where the challenger uses the simulator) remains the same as Hybrid 2. We only explain the changes where $b = 0$. If $\mathbf{y} \in \mathcal{Y}$ or there exists any $\mathbf{x} \in \mathcal{X}$ such that $f(\mathbf{x}) = \mathbf{y}$, the game aborts and the attacker fails. Otherwise, the challenger adds the target $\mathbf{y}$ to $\mathcal{Y}$. The challenger samples fresh $\mathbf{e}_{\mathbf{y}} \leftarrow \chi_1^m$. Next, it takes out the first i bits of $\mathbf{y}$ as $\mathbf{y}_i$. It samples $\mathbf{e}' \leftarrow \chi_0^{L(m+\log q)}$, $\mathbf{s}_{\mathbf{y}[[i-1]]} \leftarrow \mathbb{Z}_q^n$, $\mathbf{e}'_{\mathbf{y}[[i]]} \leftarrow \chi_0^m$, and $\mathbf{r}_{\mathbf{y}[[j]]} \leftarrow \mathbb{Z}_q^m$ for all $j \in [i-1]$ (if such $\mathbf{e}'$, $\mathbf{s}_{\mathbf{y}[[i]]}$, $\mathbf{e}_{\mathbf{y}[[i]]}$, or $\mathbf{r}_{\mathbf{y}[[j]]}$ was not chosen in any of the previous queries). Next, the challenger samples fresh $\mathbf{e}_{\mathbf{y}} \leftarrow \chi_1^m$. It sets $\mathbf{c}_{\mathbf{y}}$ such that

$$\mathbf{c}_{\mathbf{y}}^T = \left(\mathbf{s}_{\mathbf{y}[[i-1]]}^T \cdot \mathbf{A}_{\mathbf{y}[[i]]} + \mathbf{e}'_{\mathbf{y}[[i]]} \right) \prod_{j=i+1}^{\ell'} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{y}[[j]])} + \sum_{j=1}^{i-1} \left(\mathbf{r}_{\mathbf{y}[[j]]}^T \prod_{k=j+1}^{\ell'} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{y}[[k]])} \right) + \mathbf{e}_{\mathbf{y}}^T.$$

After setting up $\mathbf{c}_{\mathbf{y}}$, the challenger sets $\mathbf{b}, \mathbf{b}_f$: If $i = 1$, it sets $\mathbf{b}, \mathbf{b}_f$ such that $\mathbf{b}^T = \mathbf{s}_{\mathbf{y}[[i-1]]}^T \cdot [\mathbf{B}_1 | \dots | \mathbf{B}_L] + \mathbf{e}'^T$, $\mathbf{b}_f = \mathbf{s}_{i-1}^T \cdot \mathbf{B} + \mathbf{e}_f$, otherwise for $2 \leq i \leq \ell'$, the challenger samples $\mathbf{b} \leftarrow \mathbb{Z}_q^{L(m+\log q)}$, $\mathbf{b}_f \leftarrow \mathbb{Z}_q^{m+\log q}$ (if $\mathbf{b}, \mathbf{b}_f$ was not sampled in any previous query). Next, the challenger samples fresh $\mathbf{e} \leftarrow \chi_1^{L(m+\log q)}$. Then, it sets ct_i as $\text{ct}_i = \begin{bmatrix} \mathbf{B} \\ \mathbf{b}_f^T \end{bmatrix} \mathbf{R}_i + f_i \mathbf{G}$. Let ct be $(\text{ct}_1, \dots, \text{ct}_{\ell_f})$. It outputs $(\text{ct}, \mathbf{b}^T + \mathbf{e}^T, \mathbf{c}_{\mathbf{y}}^T)$.

- During each evaluation query, the attacker outputs $\mathbf{x}$. If $f(\mathbf{x}) \in \mathcal{I}$ for any $\mathcal{I} \in \mathcal{Y}$, the game aborts and the attacker fails. Otherwise, it adds $\mathbf{x}$ to set $\mathcal{X}$. The challenger computes $\mathbf{H}_{\widehat{\mathcal{U}}_{\mathbf{x}, \text{ct}}} := \text{EvalFX}(\widehat{\mathcal{U}}_{\mathbf{x}}, f, \{\mathbf{B}_i\}_{i \in L})$. It samples $\mathbf{e}' \leftarrow \chi_0^{L(m+\log q)}$, $\mathbf{s}_{\mathbf{y}[[i-1]]} \leftarrow \mathbb{Z}_q^n$, $\mathbf{e}'_{\mathbf{y}[[i]]} \leftarrow \chi_0^m$, and $\mathbf{r}_{\mathbf{y}[[k]]} \leftarrow \mathbb{Z}_q^m$ for all $k \in [i-1]$ (if such $\mathbf{e}'$, $\mathbf{s}_{\mathbf{y}[[i]]}$, $\mathbf{e}_{\mathbf{y}[[i]]}$, or $\mathbf{r}_{\mathbf{y}[[k]]}$ was not chosen in any of the previous queries). If $i = 1$, it sets $\mathbf{b}, \mathbf{b}_f$ such that $\mathbf{b}^T = \mathbf{s}_{\mathbf{y}[[i-1]]}^T \cdot [\mathbf{B}_1 | \dots | \mathbf{B}_L] + \mathbf{e}'^T$, $\mathbf{b}_f = \mathbf{s}_{i-1}^T \cdot \mathbf{B} + \mathbf{e}_f$, otherwise for $2 \leq i \leq \ell'$, the challenger samples $\mathbf{b} \leftarrow \mathbb{Z}_q^{Lm}$, $\mathbf{b}_f \leftarrow \mathbb{Z}_q^{m+\log q}$ (if $\mathbf{b}, \mathbf{b}_f$ was not sampled before). Then, it sets ct_i as $\text{ct}_i = \begin{bmatrix} \mathbf{B} \\ \mathbf{b}_f^T \end{bmatrix} \mathbf{R}_i + f_i \mathbf{G}$. Let ct be $(\text{ct}_1, \dots, \text{ct}_{\ell_f})$. Next, the challenger samples fresh $\mathbf{e}_{\mathbf{y}} \leftarrow \chi_1^m$. It sets $\mathbf{c}_{\mathbf{y}}$ such that

$$\mathbf{c}_{\mathbf{y}}^T = \left(\mathbf{s}_{\mathbf{y}[[i-1]]}^T \cdot \mathbf{A}_{\mathbf{y}[[i]]} + \mathbf{e}'_{\mathbf{y}[[i]]} \right) \prod_{j=i+1}^{\ell'} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{y}[[j]])} + \sum_{j=1}^{i-1} \left(\mathbf{r}_{\mathbf{y}[[j]]}^T \prod_{k=j+1}^{\ell'} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{y}[[k]])} \right) + \mathbf{e}_{\mathbf{y}}^T.$$

It outputs $\left[\left(\mathbf{b}^T \cdot \mathbf{H}_{\widehat{\mathcal{U}}_{\mathbf{x}, \text{ct}}} + \text{LastRow}(\mathcal{U}_{\mathbf{x}, \text{ct}}) + \mathbf{c}_{\mathbf{y}}^T \right) \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \right]$.

Hybrid 2.i.2 ($1 \leq i \leq \ell'$): This is identical to Hybrid 2.i.1, except that the challenger replaces $\mathbf{s}_{\mathbf{y}[[i-1]]}^T \cdot \mathbf{A}_{\mathbf{y}[[i]]} + \mathbf{e}_{\mathbf{y}[[i]]}^T$ with $\mathbf{r}_{\mathbf{y}[[i]]}$, and replaces $\mathbf{b}$ with a uniform random string if $i = 1$.

3. The attacker is allowed to make polynomial evaluation and constrained key queries.

- During each constrained key query, the attacker outputs $\mathbf{y}$. The case where $b = 1$ (where the challenger uses the simulator) remains the same as Hybrid 2. We only explain the changes where $b = 0$. If $\mathbf{y} \in \mathcal{Y}$, or there exists any $\mathbf{x} \in \mathcal{X}$ such that $f(\mathbf{x}) = \mathbf{y}$, the game aborts and the attacker fails. Otherwise, the challenger adds $\mathbf{y}$ to $\mathcal{Y}$. The challenger samples fresh $\mathbf{e}_{\mathbf{y}} \leftarrow \chi_1^m$. Next, it takes out the first i bits of $\mathbf{y}$ as $\mathbf{y}_i$. It samples $\mathbf{r}_{\mathbf{y}[[j]]} \leftarrow \mathbb{Z}_q^m$ for all $j \in [i]$ (if such $\mathbf{r}_{\mathbf{y}[[j]]}$ was not chosen). Next, the challenger samples fresh $\mathbf{e}_{\mathbf{y}} \leftarrow \chi_1^m$. It sets $\mathbf{c}_{\mathbf{y}}$ such that

$$\mathbf{c}_{\mathbf{y}}^T = \mathbf{r}_{\mathbf{y}[[i]]}^T \prod_{j=i+1}^{\ell'} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{y}[[j]])} + \sum_{j=1}^{i-1} \left(\mathbf{r}_{\mathbf{y}[[j]]}^T \prod_{k=j+1}^{\ell'} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{y}[[k]])} \right) + \mathbf{e}_{\mathbf{y}}^T.$$

After setting up $\mathbf{c}_{\mathbf{y}}$, the challenger sets $\mathbf{b}, \mathbf{b}_f$: It samples $\mathbf{b} \leftarrow \mathbb{Z}_q^{L(m+\log q)}, \mathbf{b}_f \leftarrow \mathbb{Z}_q^{m+\log q}$ (if $\mathbf{b}, \mathbf{b}_f$ was not sampled in any previous query). Next, the challenger samples fresh $\mathbf{e} \leftarrow \chi_1^{L(m+\log q)}$. Then, it sets ct_i as $\text{ct}_i = \begin{bmatrix} \mathbf{B} \\ \mathbf{b}_f^T \end{bmatrix} \mathbf{R}_i + f_i \mathbf{G}$. Let ct be $(\text{ct}_1, \dots, \text{ct}_{\ell_f})$. It outputs $(\text{ct}, \mathbf{b}^T + \mathbf{e}^T, \mathbf{c}_{\mathbf{y}}^T)$.

- During each evaluation query, the attacker outputs $\mathbf{x}$. If $f(\mathbf{x}) \in \mathcal{I}$ for any $\mathcal{I} \in \mathcal{Y}$, the game aborts and the attacker fails. Otherwise, it adds $\mathbf{x}$ to set $\mathcal{X}$. The challenger computes $\mathbf{H}_{\hat{\mathcal{U}}_{\mathbf{x}, \text{ct}}} := \text{EvalFX}(\hat{\mathcal{U}}_{\mathbf{x}}, f, \{\mathbf{B}_i\}_{i \in L})$. It $\mathbf{r}_{\mathbf{y}[[k]]} \leftarrow \mathbb{Z}_q^m$ for all $k \in [i]$ (if such $\mathbf{r}_{\mathbf{y}[[k]]}$ was not chosen in any of the previous queries). Next, the challenger samples $\mathbf{b} \leftarrow \mathbb{Z}_q^{L(m+\log q)}, \mathbf{b}_f \leftarrow \mathbb{Z}_q^{m+\log q}$ (if $\mathbf{b}, \mathbf{b}_f$ was not sampled before). Then, it sets ct_i as $\text{ct}_i = \begin{bmatrix} \mathbf{B} \\ \mathbf{b}_f^T \end{bmatrix} \mathbf{R}_i + f_i \mathbf{G}$. Let ct be $(\text{ct}_1, \dots, \text{ct}_{\ell_f})$. Then the challenger samples fresh $\mathbf{e}_{\mathbf{y}} \leftarrow \chi_1^m$. It sets $\mathbf{c}_{\mathbf{y}}$ such that

$$\mathbf{c}_{\mathbf{y}}^T = \mathbf{r}_{\mathbf{y}[[i]]}^T \prod_{j=i+1}^{\ell'} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{y}[[j]])} + \sum_{j=1}^{i-1} \left(\mathbf{r}_{\mathbf{y}[[j]]}^T \prod_{k=j+1}^{\ell'} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{y}[[k]])} \right) + \mathbf{e}_{\mathbf{y}}^T.$$

It outputs $\left[\left(\mathbf{b}^T \cdot \mathbf{H}_{\hat{\mathcal{U}}_{\mathbf{x}, \text{ct}}} + \text{LastRow}(\mathcal{U}_{\mathbf{x}}, \text{ct}) + \mathbf{c}_{\mathbf{y}}^T \right) \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \right]$.

Hybrid 3: It is the same as the previous Hybrid, except that the challenger sets $\mathbf{b}, \mathbf{b}_f$, and $\mathbf{c}_{\mathbf{y}}$ uniformly at random, and the constrained keys no longer depend on the values of $\mathbf{y}$.

4. The attacker is allowed to make polynomial evaluation and constrained key queries. **Before the queries begin, the challenger samples $\mathbf{b} \leftarrow \mathbb{Z}_q^{L(m+\log q)}, \mathbf{b}_f \leftarrow \mathbb{Z}_q^{m+\log q}$.**
- During each constrained key query, the attacker outputs $\mathbf{y}$. If $\mathbf{y} \in \mathcal{Y}$ or there exists any $\mathbf{x} \in \mathcal{X}$ such that $f(\mathbf{x}) = \mathbf{y}$, the game aborts and the attacker fails. Otherwise, the challenger adds the target $\mathbf{y}$ to $\mathcal{Y}$. It samples $\mathbf{c}_{\mathbf{y}} \leftarrow \mathbb{Z}_q^m$. After setting up $\mathbf{c}_{\mathbf{y}}$,

the challenger then encrypts function f bit by bit, such that for all $i \in [\ell_f]$, $\text{ct}_i = \begin{bmatrix} \mathbf{B} \\ \mathbf{b}_f \end{bmatrix} \mathbf{R}_i + f_i \mathbf{G}$. Let ct denote $(\text{ct}_1, \dots, \text{ct}_{\ell_f})$. Next, for all $i \in [L]$, let ψ_i denote the i -th bit of the binary representation of ct . It samples $\mathbf{e} \leftarrow \chi_1^{Lm}$ and outputs sk as $(\text{ct}, \mathbf{b}^T + \mathbf{e}^T, \mathbf{c}_y^T)$. The second case where $b = 1$ maintains the same as the previous hybrid.

- During each evaluation query, the attacker outputs $\mathbf{x}$. If $f(\mathbf{x}) \in \mathcal{I}$ for any $\mathcal{I} \in \mathcal{Y}$, the game aborts and the attacker fails. Otherwise, it adds $\mathbf{x}$ to set $\mathcal{X}$. Let $\mathbf{y}$ be $f(\mathbf{x})$. Next, it computes $\mathbf{H}_{\mathcal{U}_{\mathbf{x},f}} := \text{EvalFX}(\mathcal{U}_{\mathbf{x},f}, \{\mathbf{B}_i\}_{i \in L})$. If $\mathbf{c}_y$ was not sampled before, the challenger additionally samples $\mathbf{c}_y \leftarrow \mathbb{Z}_q^m$. It outputs

$$\lceil (\mathbf{b}^T \cdot \mathbf{H}_{\mathcal{U}_{\mathbf{x},f}} + \text{LastRow}(\mathcal{U}_{\mathbf{x},f}, \text{ct}) + \mathbf{c}_y^T) \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \rceil.$$

Hybrid 4: All of the evaluation queries are answered using random strings.

4. The attacker is allowed to make polynomial evaluation and constrained key queries. Before the queries begin, the challenger samples $\mathbf{b} \leftarrow \mathbb{Z}_q^{L(m+\log q)}$, $\mathbf{b}_f \leftarrow \mathbb{Z}_q^{m+\log q}$.
 - Constrained key queries remain unchanged from the previous hybrid.
 - During each evaluation query, the attacker outputs $\mathbf{x}$. If $f(\mathbf{x}) \in \mathcal{I}$ for any $\mathcal{I} \in \mathcal{Y}$, the game aborts and the attacker fails. The challenger adds $\mathbf{x}$ to $\mathcal{X}$. Next, the challenger samples $\mathbf{r}_x \leftarrow \mathbb{Z}_q^m$ (if $\mathbf{r}_x$ was not sampled in any previous query) and outputs $\lceil \mathbf{r}_x^T \rceil$.

7.2.2 Indistinguishability of Hybrids

Lemma 7.3. For any adversary $\mathcal{A}$ of unbounded running time, $|\text{Adv}_{\mathcal{A}}^0 - \text{Adv}_{\mathcal{A}}^1| = 0$ (i.e. Hybrid 0 and Hybrid 1 are identically distributed).

Proof. We omit the proof as it is identical to Theorem 6.3, Theorem 8.3. $\square$

Lemma 7.4. For any adversary $\mathcal{A}$ of unbounded running time, there exists a negligible function $\text{negl}(\cdot)$ such that $|\text{Adv}_{\mathcal{A}}^1 - \text{Adv}_{\mathcal{A}}^2| \leq \text{negl}(\lambda)$ (i.e. Hybrid 1 and Hybrid 2 are statistically close).

Proof. The proof is omitted as it follows from the completeness analysis. $\square$

Lemma 7.5. Assuming that $\text{GLWE}_{n,L(m+\log q)+2m,q,\chi_0}$ assumption holds, then for any PPT adversary $\mathcal{A}$, there exists a negligible function $\text{negl}(\cdot)$ such that $|\text{Adv}_{\mathcal{A}}^2 - \text{Adv}_{\mathcal{A}}^3| \leq \text{negl}(\lambda)$.

Proof. We will argue that the intermediate hybrids are indistinguishable.

Claim 7.6. For all $i \in [\ell']$, for any adversary $\mathcal{A}$ of unbounded running time, there exists a negligible function $\text{negl}(\cdot)$ such that $|\text{Adv}_{\mathcal{A}}^{2,i,0} - \text{Adv}_{\mathcal{A}}^{2,i,1}| \leq \text{negl}(\lambda)$.

Proof. Omitted by Theorem 6.6, Theorem 8.5. $\square$

Claim 7.7. Assuming that $\text{GLWE}_{n,L(m+\log q)+2m,q,\chi_0}$ holds, then for any PPT adversary $\mathcal{A}$, there exists a negligible function $\text{negl}(\cdot)$ such that $|\text{Adv}_{\mathcal{A}}^{2,i,1} - \text{Adv}_{\mathcal{A}}^{2,i,2}| \leq \text{negl}(\lambda)$ for all $i \in [\ell']$.

Proof. Omitted by Theorem 6.7, Theorem 8.6. $\square$

Claim 7.8. For all $i \in [\ell' - 1]$ and for any adversary $\mathcal{A}$ of unbounded running time, it holds that $|\text{Adv}_{\mathcal{A}}^{2,i,2} - \text{Adv}_{\mathcal{A}}^{2,(i+1),0}| = 0$.

Proof. Omitted by Theorem 6.8, Theorem 8.7. $\square$

Claim 7.9. For any adversary $\mathcal{A}$ of unbounded running time, there exists a negligible function $\text{negl}(\cdot)$, such that $|\text{Adv}_{\mathcal{A}}^{2,\ell,2} - \text{Adv}_{\mathcal{A}}^3| \leq \text{negl}(\lambda)$.

Proof. Omitted by Theorem 6.9, Theorem 8.8. $\square$

Hybrid 2.1.0 is identical to Hybrid 2, and Hybrid 2. ℓ' .2 is statistically close to Hybrid 3. Thus, our lemma follows. $\square$

Lemma 7.10. Assuming that $\text{GLWE}_{n,2m,q,\chi_0}$ assumption holds, then for any PPT adversary $\mathcal{A}$, there exists a negligible function $\text{negl}(\cdot)$ such that $|\text{Adv}_{\mathcal{A}}^3 - \text{Adv}_{\mathcal{A}}^4| \leq \text{negl}(\lambda)$.

Proof. We omit the proof as it is identical to Theorems 6.10 to 6.14, Theorem 8.9. $\square$

In Hybrid 4, constrained key queries are independent from evaluation queries. Furthermore, by leftover hash lemma, $\begin{bmatrix} \mathbf{B} \\ \mathbf{b}_f \end{bmatrix} \mathbf{R}_i + f_i \mathbf{G}$ is statistically close to $\mathbf{B}' \leftarrow \mathbb{Z}_q^{(n+1) \times (m \log q)}$ given $\mathbf{R}_i \leftarrow \{0,1\}^{(m+\log q) \times (m+\log q)}$, thus the outputs of constrained key queries for $b = 0$ and $b = 1$ are statistically close. Thus in Hybrid 4, attacker $\mathcal{A}$ has at most $1/2 + \text{negl}(\lambda)$ advantage for some negligible function $\text{negl}(\cdot)$. Theorem 7.2 follows.

7.3 Purified Constraint Pseudorandomness

Theorem 7.11. Assuming that $\text{GLWE}_{n,(L+2)m,q,\chi}$ (Theorem 3.1) assumption holds, then our design satisfies semi-adaptive pseudorandomness (Theorem 6.1).

To prove that our theorem holds, we will use the following hybrids.

7.3.1 Sequence of Hybrid games

Hybrid 0: This is the original pseudorandomness experiment.

1. The attacker outputs the constraining function f .
2. The challenger starts by running the setup algorithm. It samples $\mathbf{B}, \mathbf{B}_1, \dots, \mathbf{B}_L \leftarrow \mathbb{Z}_q^{n \times (m+\log q)}$, $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, $\mathbf{A}_0, \mathbf{A}_1 \leftarrow \mathbb{Z}_q^{n \times m}$, $\mathbf{R}_1, \dots, \mathbf{R}_{\ell_f} \leftarrow \{0,1\}^{(m+\log q) \times (m+\log q)}$, $\mathbf{e}_f \leftarrow \chi_0^{m+\log q}$. The challenger outputs $\mathbf{B}, \mathbf{B}_1, \dots, \mathbf{B}_L, \mathbf{A}_0, \mathbf{A}_1$. It samples a bit $b \leftarrow \{0,1\}$.
3. The attacker is allowed to make polynomial evaluation and constrained key queries.
 - During each constrained key query, the attacker outputs $\mathbf{y}$. If $\mathbf{y} \in \mathcal{Y}$, or there exists any $\mathbf{x} \in \mathcal{X}$ such that $f(\mathbf{x}) = \mathbf{y}$, the game aborts and the attacker fails. Otherwise, the challenger adds $\mathbf{y}$ to $\mathcal{Y}$, and responds the query following $\text{Constrain}(\text{msk}, C_{f,\mathbf{y}})$: It samples $\mathbf{e}_{\mathbf{y}} \leftarrow \chi_1^m$. It sets $\mathbf{c}_{\mathbf{y}}$ such that $\mathbf{c}_{\mathbf{y}}^T = \mathbf{s}^T \cdot \mathbf{A}(\mathbf{y}) + \mathbf{e}_{\mathbf{y}}^T$. After setting up the range key $\mathbf{c}_{\mathbf{y}}$, the challenger then then encrypts function f bit by bit, such that for all $i \in [\ell_f]$,

$\text{ct}_i = \left[\begin{smallmatrix} \mathbf{B} \\ \mathbf{s}^T \mathbf{B} + \mathbf{e}_f^T \end{smallmatrix} \right] \mathbf{R}_i + f_i \mathbf{G}$. Let ct denote $(\text{ct}_1, \dots, \text{ct}_{\ell_f})$. Next, for all $i \in [L]$, let ψ_i denote the i -th bit of the binary representation of ct . It samples $\mathbf{e} \leftarrow \chi_1^{Lm}$ and outputs sk as $(\text{ct}, \mathbf{s}^T \cdot [\mathbf{B}_1 - \psi_1 \mathbf{G} \mid \dots \mid \mathbf{B}_L - \psi_L \mathbf{G}] + \mathbf{e}^T, \mathbf{c}_y^T)$.

- During each evaluation query, the attacker outputs $\mathbf{x}$. If $f(\mathbf{x}) \in \mathcal{Y}$, the game aborts and the attacker fails. The challenger responds with $\text{Eval}(\text{sk}, \mathbf{x})$: It computes $\mathbf{H}_{\hat{\mathcal{U}}_{\mathbf{x}}} := \text{EvalF}(\hat{\mathcal{U}}_{\mathbf{x}}, \{\mathbf{B}_i\}_{i \in L}), \mathbf{B}_{\hat{\mathcal{U}}_{\mathbf{x}}} := [\mathbf{B}_1 \mid \dots \mid \mathbf{B}_L] \cdot \mathbf{H}_{\hat{\mathcal{U}}_{\mathbf{x}}}$, and outputs $\lceil \mathbf{s}^T \cdot \mathbf{B}_{\hat{\mathcal{U}}_{\mathbf{x}}} \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \rceil$.
- 4. During the challenge query, the attacker outputs an input $\mathbf{x}^*$. If $\mathbf{x}^* \in \mathcal{X}$ or $f(\mathbf{x}^*) \in \mathcal{Y}$, the attacker fails and the game aborts. If $b = 0$, the challenger responds with $\text{Eval}(\text{sk}, \mathbf{x}^*)$: It computes $\mathbf{H}_{\hat{\mathcal{U}}_{\mathbf{x}^*}} := \text{EvalF}(\hat{\mathcal{U}}_{\mathbf{x}^*}, \{\mathbf{B}_i\}_{i \in L}), \mathbf{B}_{\hat{\mathcal{U}}_{\mathbf{x}^*}} := [\mathbf{B}_1 \mid \dots \mid \mathbf{B}_L] \cdot \mathbf{H}_{\hat{\mathcal{U}}_{\mathbf{x}^*}}$, and outputs $\lceil \mathbf{s}^T \cdot \mathbf{B}_{\hat{\mathcal{U}}_{\mathbf{x}^*}} \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x}^*)) \rceil$. For $b = 1$, the challenger samples $\mathbf{r}_{\mathbf{x}^*} \leftarrow \mathbb{Z}_q^m$ if $\mathbf{r}_{\mathbf{x}^*}$ was not sampled in any previous query and outputs $\lceil \mathbf{r}_{\mathbf{x}^*}^T \rceil$.
- 5. It repeats step 3, except that if $\mathcal{A}$ queries $f(\mathbf{x}^*)$ during constrained key queries or $\mathbf{x}^*$ during evaluation queries, the attacker immediately fails and the game aborts.
- 6. The attacker outputs b' and wins if and only if $b' = b$.

Hybrid 1: This is the same as Hybrid 0, except that in Hybrid 1, the challenger sets pp as $\mathbf{B}_1 + \psi_1 \mathbf{G}, \dots, \mathbf{B}_L + \psi_L \mathbf{G}, \mathbf{A}_0, \mathbf{A}_1$, where $(\psi_1, \dots, \psi_L)$ is the encryption of function f . The constrained keys become independent from function f .

- 2. The challenger samples $\mathbf{B}, \mathbf{B}_1, \dots, \mathbf{B}_L \leftarrow \mathbb{Z}_q^{n \times (m + \log q)}, \mathbf{s} \leftarrow \mathbb{Z}_q^n, \mathbf{A}_0, \mathbf{A}_1 \leftarrow \mathbb{Z}_q^{n \times m}, \mathbf{R}_1, \dots, \mathbf{R}_{\ell_f} \leftarrow \{0, 1\}^{(m + \log q) \times (m + \log q)}, \mathbf{e}_f \leftarrow \chi_0^{m + \log q}$. The challenger outputs $\mathbf{B}, \mathbf{B}_1 + \psi_1 \mathbf{G}, \dots, \mathbf{B}_L + \psi_L \mathbf{G}, \mathbf{A}_0, \mathbf{A}_1$. It samples a bit $b \leftarrow \{0, 1\}$.
- 3. The attacker is allowed to make polynomial evaluation and constrained key queries.
 - The constrained key query is nearly identical to Hybrid 0, except that the challenger sets sk as $(\text{ct}, \mathbf{s}^T \cdot [\mathbf{B}_1 \mid \dots \mid \mathbf{B}_L] + \mathbf{e}_f^T, \mathcal{C})$.
 - The evaluation query is the same as Hybrid 0, except that if $b = 0$, the challenger sets $\mathbf{B}_{\hat{\mathcal{U}}_{\mathbf{x}}} := [\mathbf{B}_1 + \psi_1 \mathbf{G} \mid \dots \mid \mathbf{B}_L + \psi_L \mathbf{G}] \cdot \mathbf{H}_{\hat{\mathcal{U}}_{\mathbf{x}}}$.
- 4. The challenge query is the same as Hybrid 0, except that if $b = 0$, the challenger computes $\mathbf{H}_{\hat{\mathcal{U}}_{\mathbf{x}^*}} := \text{EvalF}(\hat{\mathcal{U}}_{\mathbf{x}^*}, \{\mathbf{B}_i\}_{i \in L}), \mathbf{B}_{\hat{\mathcal{U}}_{\mathbf{x}^*}} := [\mathbf{B}_1 + \psi_1 \mathbf{G} \mid \dots \mid \mathbf{B}_L + \psi_L \mathbf{G}] \cdot \mathbf{H}_{\hat{\mathcal{U}}_{\mathbf{x}^*}}$, and outputs $\lceil \mathbf{s}^T \cdot \mathbf{B}_{\hat{\mathcal{U}}_{\mathbf{x}^*}} \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x}^*)) \rceil$.

Hybrid 2: The challenger answers evaluation/challenge queries using the CEval algorithm.

- 3. The attacker is allowed to make polynomial evaluation and constrained key queries.
 - The constrained key queries remain the same as Hybrid 1.
 - During each evaluation query, the attacker outputs $\mathbf{x}$. If $f(\mathbf{x}) \in \mathcal{Y}$, the game aborts and the attacker fails. Otherwise, it adds $\mathbf{x}$ to set $\mathcal{X}$. If $b = 0$, the challenger computes

$\mathbf{H}_{\hat{\mathcal{U}}_{\mathbf{x}, \text{ct}}} := \text{EvalFX}(\hat{\mathcal{U}}_{\mathbf{x}}, \text{ct}, \{\mathbf{B}_i\}_{i \in L})$. Let $\mathbf{y}$ be the output of $f(\mathbf{x})$, the challenger samples $\mathbf{e} \leftarrow \chi_1^{Lm}$, $\mathbf{e}_y \leftarrow \chi_1^m$. It outputs

$$\left\lceil \left((\mathbf{s}^T \cdot [\mathbf{B}_1 | \dots | \mathbf{B}_L] + \mathbf{e}^T) \cdot \mathbf{H}_{\hat{\mathcal{U}}_{\mathbf{x}, \text{ct}}} + \text{LastRow}(\mathcal{U}_{\mathbf{x}}, \text{ct}) + \mathbf{s}^T \cdot \mathbf{A}(\mathbf{y}) + \mathbf{e}_y^T \right) \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \right\rceil.$$

For $b = 1$, the challenger samples $\mathbf{r}_{\mathbf{x}} \leftarrow \mathbb{Z}_q^m$ if $\mathbf{r}_{\mathbf{x}}$ was not sampled in any previous query and outputs $\lceil \mathbf{r}_{\mathbf{x}}^T \rceil$.

4. During the challenge query, the attacker outputs an input $\mathbf{x}^*$. If $\mathbf{x}^* \in \mathcal{X}$ or $f(\mathbf{x}^*) \in \mathcal{Y}$, the attacker fails and the game aborts. Let $\mathbf{y}^*$ be $f(\mathbf{x}^*)$. If $b = 0$, it samples $\mathbf{e} \leftarrow \chi_1^{Lm}$, $\mathbf{e}_{\mathbf{y}^*} \leftarrow \chi_1^m$, and computes $\mathbf{H}_{\hat{\mathcal{U}}_{\mathbf{x}^*, \text{ct}}} := \text{EvalFX}(\hat{\mathcal{U}}_{\mathbf{x}^*}, \text{ct}, \{\mathbf{B}_i\}_{i \in L})$. It outputs

$$\left\lceil \left((\mathbf{s}^T \cdot [\mathbf{B}_1 | \dots | \mathbf{B}_L] + \mathbf{e}^T) \cdot \mathbf{H}_{\hat{\mathcal{U}}_{\mathbf{x}^*, \text{ct}}} + \text{LastRow}(\mathcal{U}_{\mathbf{x}^*}, \text{ct}) + \mathbf{s}^T \cdot \mathbf{A}(\mathbf{y}) + \mathbf{e}_y^T \right) \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \right\rceil.$$

Hybrid 3: The challenger sets $\mathbf{b}$, $\mathbf{b}_f$, and $\mathbf{c}_{\mathbf{y}}$ uniformly at random.

3. The attacker is allowed to make polynomial evaluation and constrained key queries. **Before the queries begin, the challenger samples $\mathbf{b} \leftarrow \mathbb{Z}_q^{L(m+\log q)}$, $\mathbf{b}_f \leftarrow \mathbb{Z}_q^{m+\log q}$.**
 - During each constrained key query, the attacker outputs $\mathbf{y}$. If $\mathbf{y} \in \mathcal{Y}$, or there exists any $\mathbf{x} \in \mathcal{X}$ such that $f(\mathbf{x}) = \mathbf{y}$, the game aborts and the attacker fails. Otherwise, the challenger adds $\mathbf{y}$ to $\mathcal{Y}$. After setting up $\mathbf{c}_{\mathbf{y}}$, the challenger then encrypts function f bit by bit, such that for all $i \in [\ell_f]$, $\text{ct}_i = \begin{bmatrix} \mathbf{B} \\ \mathbf{b}_f \end{bmatrix} \mathbf{R}_i + f_i \mathbf{G}$. Let ct denote $(\text{ct}_1, \dots, \text{ct}_{\ell_f})$. Next, for all $i \in [L]$, let ψ_i denote the i -th bit of the binary representation of ct . It samples $\mathbf{e} \leftarrow \chi_1^{Lm}$ and outputs sk as $(\text{ct}, \mathbf{b}^T + \mathbf{e}^T, \mathbf{c}_{\mathbf{y}}^T)$.
 - During each evaluation query, the attacker outputs $\mathbf{x}$. If $f(\mathbf{x}) \in \mathcal{Y}$, the game aborts and the attacker fails. Otherwise, it adds $\mathbf{x}$ to set $\mathcal{X}$. Let $\mathbf{y}$ be $f(\mathbf{x})$. **If $b = 0$, it computes $\mathbf{H}_{\hat{\mathcal{U}}_{\mathbf{x}, \text{ct}}} := \text{EvalFX}(\hat{\mathcal{U}}_{\mathbf{x}}, \text{ct}, \{\mathbf{B}_i\}_{i \in L})$. If $\mathbf{c}_{\mathbf{y}}$ was not sampled before, the challenger additionally samples $\mathbf{c}_{\mathbf{y}} \leftarrow \mathbb{Z}_q^m$. It outputs**

$$\lceil (\mathbf{b}^T \cdot \mathbf{H}_{\hat{\mathcal{U}}_{\mathbf{x}, \text{ct}}} + \text{LastRow}(\mathcal{U}_{\mathbf{x}}, \text{ct}) + \mathbf{c}_{\mathbf{y}}^T) \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \rceil.$$

Otherwise the challenger samples $\mathbf{r}_{\mathbf{x}} \leftarrow \mathbb{Z}_q^m$ if $\mathbf{r}_{\mathbf{x}}$ was not sampled in any previous query and outputs $\lceil \mathbf{r}_{\mathbf{x}}^T \rceil$.

4. During the challenge query, the attacker outputs an input $\mathbf{x}^*$. If $\mathbf{x}^* \in \mathcal{X}$ or $f(\mathbf{x}^*) \in \mathcal{Y}$, the attacker fails and the game aborts. Let $\mathbf{y}^*$ be $f(\mathbf{x}^*)$. **If $b = 0$, the challenger computes $\mathbf{H}_{\hat{\mathcal{U}}_{\mathbf{x}^*, \text{ct}}} := \text{EvalFX}(\hat{\mathcal{U}}_{\mathbf{x}^*}, \text{ct}, \{\mathbf{B}_i\}_{i \in L})$. If $\mathbf{c}_{\mathbf{y}^*}$ was not sampled before, the challenger additionally samples $\mathbf{c}_{\mathbf{y}^*} \leftarrow \mathbb{Z}_q^m$. It outputs**

$$\lceil (\mathbf{b}^T \cdot \mathbf{H}_{\hat{\mathcal{U}}_{\mathbf{x}^*, \text{ct}}} + \text{LastRow}(\mathcal{U}_{\mathbf{x}^*}, \text{ct}) + \mathbf{c}_{\mathbf{y}^*}^T) \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \rceil.$$

Otherwise the challenger samples $\mathbf{r}_{\mathbf{x}^*} \leftarrow \mathbb{Z}_q^m$ and outputs $\lceil \mathbf{r}_{\mathbf{x}^*}^T \rceil$.

Hybrid 4: The challenger answers evaluation queries using random strings.

3. The attacker will make polynomial evaluation and constrained key queries. Let $Q = \text{poly}(\lambda)$ be the number of the evaluation queries. Before the queries begin, the challenger samples $r \leftarrow [Q + 1]$, $\mathbf{b} \leftarrow \mathbb{Z}_q^{L(m+\log q)}$, $\mathbf{b}_f \leftarrow \mathbb{Z}_q^{m+\log q}$.

- Constrained key queries remain unchanged from the previous hybrid.
- During each evaluation query, the attacker outputs an input $\mathbf{x}$. Let $\mathbf{y}$ be $f(\mathbf{x})$. If this is the r -th evaluation query, and $\mathbf{y}$ has never been queried during the constrained key queries, and $f(\mathbf{x}') \neq \mathbf{y}$ for all $\mathbf{x}' \in \mathcal{X}$, then the challenger sets $\mathbf{y}^*$ as $\mathbf{y}$. **Next, if $\mathbf{y}^* = \mathbf{y}$, then the challenger samples $\mathbf{r}_{\mathbf{x}} \leftarrow \mathbb{Z}_q^m$ (if it hasn't already been sampled before) and outputs $[\mathbf{r}_{\mathbf{x}}^T]$.** Otherwise, it computes $\mathbf{H}_{\widehat{\mathcal{U}}_{\mathbf{x}, \text{ct}}} := \text{EvalFX}(\widehat{\mathcal{U}}_{\mathbf{x}}, \text{ct}, \{\mathbf{B}_i\}_{i \in L})$. If $\mathbf{c}_{\mathbf{y}}$ was not sampled before, the challenger additionally samples $\mathbf{c}_{\mathbf{y}} \leftarrow \mathbb{Z}_q^m$. It outputs

$$\lceil (\mathbf{b}^T \cdot \mathbf{H}_{\widehat{\mathcal{U}}_{\mathbf{x}, \text{ct}}} + \text{LastRow}(\mathcal{U}_{\mathbf{x}}, \text{ct}) + \mathbf{c}_{\mathbf{y}}^T) \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \rceil.$$

The challenger adds $\mathbf{x}$ to set $\mathcal{X}$.

4. During the challenge query, the attacker outputs $\mathbf{x}^*$. If $\mathbf{x}^* \in \mathcal{X}$ or $f(\mathbf{x}^*) \in \mathcal{Y}$, the attacker fails and the game aborts. Additionally, if $f(\mathbf{x}^*) = f(\mathbf{x})$ for any $\mathbf{x} \in \mathcal{X}$ and $f(\mathbf{x}^*) \neq \mathbf{y}^*$, the challenger aborts the game. Next, if $\mathbf{y}^*$ has not yet been set, the challenger sets $\mathbf{y}^*$ as $f(\mathbf{x}^*)$. Let $\mathbf{y}^*$ be $f(\mathbf{x}^*)$. **For both $b = 0$ and $b = 1$, the challenger samples $\mathbf{r}_{\mathbf{x}^*} \leftarrow \mathbb{Z}_q^m$ and outputs $[\mathbf{r}_{\mathbf{x}^*}^T]$.**

7.3.2 Indistinguishability of Hybrids

Lemma 7.12. For any adversary $\mathcal{A}$ of unbounded running time, $|\text{Adv}_{\mathcal{A}}^0 - \text{Adv}_{\mathcal{A}}^1| = 0$ (i.e. Hybrid 0 and Hybrid 1 are identically distributed).

Proof. Omitted as it is nearly identical to Theorem 6.3, Theorem 7.3. □

Lemma 7.13. For any adversary $\mathcal{A}$ of unbounded running time, there exists a negligible function $\text{negl}(\cdot)$ such that $|\text{Adv}_{\mathcal{A}}^1 - \text{Adv}_{\mathcal{A}}^2| \leq \text{negl}(\lambda)$ (i.e. Hybrid 1 and Hybrid 2 are statistically close).

Proof. Omitted as it follows from smudging and the error bound analysis is nearly identical to the completeness analysis. □

Lemma 7.14. Assuming that $\text{GLWE}_{n, L(m+\log q)+2m, q, \chi_0}$ assumption holds, then for any PPT adversary $\mathcal{A}$, there exists a negligible function $\text{negl}(\cdot)$ such that $|\text{Adv}_{\mathcal{A}}^2 - \text{Adv}_{\mathcal{A}}^3| \leq \text{negl}(\lambda)$.

Proof. Omitted as it is nearly identical to the proof of Theorem 7.5. □

Lemma 7.15. Assuming that $\text{GLWE}_{n, 2m, q, \chi_0}$ assumption holds, then for any PPT adversary $\mathcal{A}$, there exists a negligible function $\text{negl}(\cdot)$ such that $|\text{Adv}_{\mathcal{A}}^3 - \text{Adv}_{\mathcal{A}}^4| \leq \text{negl}(\lambda)$.

Proof. We omit the proof as it is nearly identical to Theorems 6.10 to 6.14, Theorem 7.10. □

Because the adversary has at most $1/2$ advantage in Hybrid 4, we conclude that Theorem 7.11 holds.

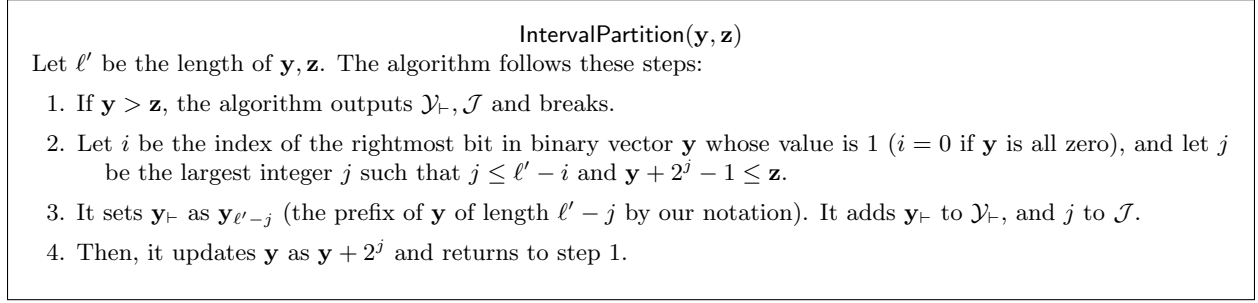


Figure 2: The binary interval partitioning algorithm

8 Collusion-Resistant CPRFs for Predicated Range Circuits

In this section, we construct our collusion-resistant CPRFs for the predicated range class. (See Section 5.3). Let χ_0 and χ_1 be two truncated discrete gaussian distributions set as $\tilde{D}_{\mathbb{Z}, \sigma}$ and $\tilde{D}_{\mathbb{Z}, \sigma'}$, respectively. Fix any $\epsilon < 1/2$. Let n, m, q, σ, σ' be parameters with the following constraints:

- $n = \text{poly}(\lambda)$, $q \leq 2^{n^\epsilon}$, $m = \Omega(n \log q)$ (for LWE security)
- $\sigma' \geq \sigma \cdot m^{\ell'+1} \cdot 2^{\Omega(\lambda)}$ (for smudging)
- $\frac{q}{p} \geq \sigma' \cdot (L \cdot m^{\Omega(d)} + m^\ell + m^{\ell'+1}) \cdot 2^{\Omega(\lambda)}$ (for correctness/smudging)

One possible parameter setting is as follows:

$$n = 2\lambda(d + \ell + \ell')/\epsilon, m = n \log q, p = 2, q = 2^{n^\epsilon}, \sigma = n, \sigma' = \sigma \cdot m^{\ell'+1} \cdot 2^\lambda.$$

For our design, the notations, definition of function $\mathbf{A}(\cdot)$, and algorithms **Setup**, **Eval** are identical to the design of Section 6. The rest of the algorithms are as follows:

Constrain($\text{msk}, C_{f, \mathbf{y}, \mathbf{z}}$): It takes as input msk and range constraint circuit C consisting of function f and range bound $\mathbf{y}, \mathbf{z}$. It runs the interval partitioning algorithm **IntervalPartition**($\mathbf{y}, \mathbf{z}$) as defined in Fig. 2, and obtains set $\mathcal{Y}_\perp$. For each $\mathbf{y}_\perp \in \mathcal{Y}_\perp$ and the corresponding $j \in \mathcal{J}$, it samples $\mathbf{e}_{\mathbf{y}_\perp} \leftarrow \chi_1^m$, sets $\mathbf{c}_{\mathbf{y}_\perp}$ such that $\mathbf{c}_{\mathbf{y}_\perp}^T = \mathbf{s}^T \cdot \mathbf{A}(\mathbf{y}_\perp) + \mathbf{e}_{\mathbf{y}_\perp}^T$. It adds $\mathbf{c}_{\mathbf{y}_\perp}$ to the set $\mathcal{C}$. Such $\mathbf{c}_{\mathbf{y}_\perp}$ corresponds to the range between $\mathbf{y}$ and $\mathbf{y} + 2^j - 1$ (including both ends). Next, it samples $\mathbf{e} \leftarrow \chi_1^{Lm}$ and outputs

$$\text{sk} := (\mathbf{s}^T \cdot [\mathbf{B}_1 - f[1]\mathbf{G}] \dots [\mathbf{B}_L - f[L]\mathbf{G}] + \mathbf{e}^T, \mathcal{C}).$$

CEval($\text{sk}, \mathbf{x}$): Parse sk as $\mathbf{b}^T, \mathcal{C}$. Let $\mathbf{y}$ be $f(\mathbf{x})$. It takes out the key $\mathbf{c}_{\mathbf{y}_\perp} \in \mathcal{C}$ where $\mathbf{c}_{\mathbf{y}_\perp}$ corresponds to the range of $2^j \cdot \mathbf{y}_\perp$ and $2^j \cdot \mathbf{y}_\perp + 2^j - 1$ for some $j \in \{0, \dots, \ell'\}$ and $\mathbf{y}_\perp$ as a prefix of $\mathbf{y}$ of length $\ell' - j$. Note that by our design, there must exist such $\mathbf{c}_{\mathbf{y}_\perp}$ corresponding to some prefix of $\mathbf{y}$ as long as the constrained key covers $\mathbf{y}$. Let $\mathbf{y}_\perp$ be the suffix of $\mathbf{y}$ of length j . It computes $\mathbf{H}_{\mathcal{U}_{\mathbf{x}, f}} := \text{EvalFX}(\mathcal{U}_{\mathbf{x}, f}, f, \{\mathbf{B}_i\}_{i \in L})$. The PRF output is

$$\left[\left(\mathbf{b}^T \cdot \mathbf{H}_{\mathcal{U}_{\mathbf{x}, f}} + \mathbf{c}_{\mathbf{y}_\perp}^T \cdot \prod_{i=1}^{\ell'-j} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{y}_\perp[i]}) \right) \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \right].$$

8.1 Correctness

We show that if $\mathbf{y} \leq f(\mathbf{x}) \leq \mathbf{z}$, then with high probability it satisfies that $\text{Eval}(\text{msk}, \mathbf{x}) = \text{CEval}(\text{sk}, \mathbf{x})$ for $\text{msk} \leftarrow \text{Setup}(1^\lambda, 1^\ell, 1^L, 1^d)$, $\text{sk} \leftarrow \text{Constrain}(\text{msk}, C_{f, \mathbf{y}, \mathbf{z}})$. Given a constrained key sk that is correctly computed, and its range covers $\mathbf{x}$ and let $\mathbf{y}$ be $f(\mathbf{x})$, then the output of $\text{CEval}(\text{sk}, \mathbf{x})$ could be rewritten as:

$$\left[\left(\mathbf{s}^T \cdot [\mathbf{B}_1 - f[1]\mathbf{G}] \dots [\mathbf{B}_L - f[L]\mathbf{G}] + \mathbf{e}^T \right) \cdot \mathbf{H}_{\mathcal{U}_{\mathbf{x},f}} + \left(\mathbf{s}^T \cdot \mathbf{A}(\mathbf{y}_+) + \mathbf{e}_{\mathbf{y}_+}^T \right) \cdot \prod_{i=1}^{\ell-j} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{y}_{-}[i]}) \right] \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \Bigg].$$

By Theorem 3.6, it holds that $\|\mathbf{H}_{\mathcal{U}_{\mathbf{x}}}\|_\infty, \|\mathbf{H}_{\mathcal{U}_{\mathbf{x},f}}\|_\infty \leq m^{O(d)}$, and $[\mathbf{B}_1 - f[1]\mathbf{G}] \dots [\mathbf{B}_L - f[L]\mathbf{G}] \cdot$

$\mathbf{H}_{\mathcal{U}_{\mathbf{x},f}} = [\mathbf{B}_1] \dots [\mathbf{B}_L] \cdot \mathbf{H}_{\mathcal{U}_{\mathbf{x}}} - \mathbf{A}(f(\mathbf{x}))$. And because $f(\mathbf{x}) = \begin{bmatrix} \mathbf{y}_+ \\ \mathbf{y}_{-} \end{bmatrix}$, it holds that $\mathbf{A}(\mathbf{y}_+) \cdot \prod_{i=1}^{\ell-j} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{y}_{-}[i]}) = \mathbf{A}(f(\mathbf{x}))$. Thus, one could simplify the above equation of CEval as the following:

$$\left[\left(\mathbf{s}^T \cdot ([\mathbf{B}_1] \dots [\mathbf{B}_L] \cdot \mathbf{H}_{\mathcal{U}_{\mathbf{x},f}}) + \mathbf{e}^T \cdot \mathbf{H}_{\mathcal{U}_{\mathbf{x},f}} + \mathbf{e}_{\mathbf{y}_+}^T \cdot \prod_{i=1}^{\ell-j} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{y}_{-}[i]}) \right) \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \right].$$

Observe that the highlighted part of such expression is the only difference between $\text{CEval}(\text{sk}, \mathbf{x})$ and $\text{Eval}(\text{msk}, \mathbf{x})$. Next, we analyze the norm bound of that highlighted expression:

$$\left\| \left(\mathbf{e}^T \cdot \mathbf{H}_{\mathcal{U}_{\mathbf{x},f}} + \mathbf{e}_{\mathbf{y}_+}^T \cdot \prod_{i=1}^{\ell-j} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{y}_{-}[i]}) \right) \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \right\|_\infty \leq (\sigma' \cdot m^{O(d)} \cdot L \cdot m + \sigma' \cdot m \cdot m^{\ell'}) \cdot m.$$

Such norm bound is exponentially smaller than q/p . Thus we conclude that $\text{CEval}(\text{sk}, \mathbf{x})$ gives the same output as $\text{Eval}(\text{msk}, \mathbf{x})$ with high probability.

8.2 Purified Constraint Pseudorandomness

We prove that our CPRF satisfies semi-adaptive pseudorandomness security as in Theorem 6.1, except we also require each input (sent by $\mathcal{A}$ during evaluation query) is not a satisfying input for any constraint that $\mathcal{A}$ gets a constrained key for. For simplicity, we still refer to this as *semi-adaptive purified constraint pseudorandomness*. The reason this restriction was not needed for compute-&-compare class is that in compute-&-compare class, the constrained keys generated for answering the evaluation queries will later be reused for the constrained key query, however in predicated range class, the challenger cannot predict the range of the constrained key queries and thus cannot reuse the keys. At a very high level, this issue is very similar to the difficulty of answering secret key queries in hierarchical and standard IBE [GKR25].

Definition 8.1 (Semi-Adaptive Purified Constrained Pseudorandomness). A CPRF $\mathcal{F}$ satisfies (collusion-resistant) semi-adaptive constrained pseudorandomness for predicated range circuits if the advantage of any PPT adversary in the experiment described in Theorem 6.1 (generalized to predicated range constraints) is always negligible, where the adversary must also adhere to the restriction that each of $\mathcal{A}$'s evaluation query $\mathbf{x}$ (and the challenge query $\mathbf{x}^*$) must not satisfy the predicated range constraint for any constrained key query (i.e., $f(\mathbf{x}) \notin [\mathbf{y}_i, \mathbf{z}_i]$ for all i).

Theorem 8.2. Assuming that $\text{GLWE}_{n,(L+2)m,q,\chi_0}$ (Theorem 3.1) assumption holds, then the above construction satisfies semi-adaptive pseudorandomness for predicated range class, as per Section 8.2.

To prove the above theorem, we start by introducing a series of hybrids.

8.2.1 Sequence of Hybrid Games

In the sequence of the hybrid experiments, Hybrid 0 is the original game and the challenger is honest. In Hybrid 1, the challenger will set msk according to the adversary's choice of function f and the constraint secret keys will no longer depend on the description of function f . Hybrid 2 will answer all of the evaluation queries following the process of constrained evaluation algorithm as long as $b = 0$. Next, Hybrid 3 will set all of the constrained keys uniformly at random and Hybrid 4 answers all the evaluation queries using uniform random strings.

Hybrid 0: This is the original semi-adaptive pseudorandomness experiment.

1. The attacker outputs the constraining function f .
2. The challenger starts by running the setup algorithm. It samples $\mathbf{B}_1, \dots, \mathbf{B}_L \leftarrow \mathbb{Z}_q^{n \times m}$, $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, $\mathbf{A}_0, \mathbf{A}_1 \leftarrow \mathbb{Z}_q^{n \times m}$. The challenger outputs $\mathbf{B}_1, \dots, \mathbf{B}_L, \mathbf{A}_0, \mathbf{A}_1$. It samples a bit $b \leftarrow \{0, 1\}$.
3. The attacker is allowed to make polynomial evaluation and constrained key queries.
 - During each constrained key query, the attacker outputs $\mathbf{y}, \mathbf{z}$. If there exists any range $\mathcal{I} \in \mathcal{Y}$ such that $[\mathbf{y}, \mathbf{z}]$ intersects with $\mathcal{I}$, or there exists any $\mathbf{x} \in \mathcal{X}$ such that $f(\mathbf{x}) \in [\mathbf{y}, \mathbf{z}]$, the game aborts and the attacker fails. Otherwise, the challenger adds the interval $[\mathbf{y}, \mathbf{z}]$ to $\mathcal{I}$, and responds the query following $\text{Constrain}(\text{msk}, C_{f,\mathbf{y},\mathbf{z}})$: It runs $\text{IntervalPartition}(\mathbf{y}, \mathbf{z})$ and obtains $\mathcal{Y}_-, \mathcal{J}$. For each $\mathbf{y} \in \mathcal{Y}$ and the corresponding $j \in \mathcal{J}$, it sets $\mathbf{y}_-$ as $\mathbf{y}_{\ell-j}$ and samples $\mathbf{e}_{\mathbf{y}_-} \leftarrow \chi_1^m$. It sets $\mathbf{c}_{\mathbf{y}_-}$ such that $\mathbf{c}_{\mathbf{y}_-}^T = \mathbf{s}^T \cdot \mathbf{A}(\mathbf{y}_-) + \mathbf{e}_{\mathbf{y}_-}^T$, and adds $\mathbf{c}_{\mathbf{y}_-}$ to the set $\mathcal{C}$. After setting up the range key $\mathbf{c}_{\mathbf{y}_-}$ for all $\mathbf{y} \in \mathcal{Y}_-$, the challenger samples $\mathbf{e} \leftarrow \chi_1^{Lm}$ and outputs $\text{sk} := (\mathbf{s}^T \cdot [\mathbf{B}_1 - f[1]\mathbf{G}] \dots [\mathbf{B}_L - f[L]\mathbf{G}] + \mathbf{e}^T, \mathcal{C})$.
 - During each evaluation query, the attacker outputs $\mathbf{x}$. If $f(\mathbf{x}) \in \mathcal{I}$ for any $\mathcal{I} \in \mathcal{Y}$, the game aborts and the attacker fails. Otherwise, it adds $\mathbf{x}$ to set $\mathcal{X}$. The challenger responds with $\text{Eval}(\text{sk}, \mathbf{x})$: It computes $\mathbf{H}_{\mathcal{U}_{\mathbf{x}}} := \text{EvalF}(\mathcal{U}_{\mathbf{x}}, \{\mathbf{B}_i\}_{i \in L})$, $\mathbf{B}_{\mathcal{U}_{\mathbf{x}}} := [\mathbf{B}_1 | \dots | \mathbf{B}_L] \cdot \mathbf{H}_{\mathcal{U}_{\mathbf{x}}}$, and outputs $\lceil \mathbf{s}^T \cdot \mathbf{B}_{\mathcal{U}_{\mathbf{x}}} \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \rceil$.
4. During the challenge query, the attacker outputs $\mathbf{x}^*$. If $f(\mathbf{x}^*) \in \mathcal{I}$ for any $\mathcal{I} \in \mathcal{Y}$, or if $\mathbf{x}^* \in \mathcal{X}$, the game aborts and the attacker fails. Otherwise, it adds $\mathbf{x}^*$ to set $\mathcal{X}$. If $b = 0$, the challenger will answer the query in the way identical to the above evaluation query: It computes $\mathbf{H}_{\mathcal{U}_{\mathbf{x}^*}} := \text{EvalF}(\mathcal{U}_{\mathbf{x}^*}, \{\mathbf{B}_i\}_{i \in L})$, $\mathbf{B}_{\mathcal{U}_{\mathbf{x}^*}} := [\mathbf{B}_1 | \dots | \mathbf{B}_L] \cdot \mathbf{H}_{\mathcal{U}_{\mathbf{x}^*}}$, and outputs $\lceil \mathbf{s}^T \cdot \mathbf{B}_{\mathcal{U}_{\mathbf{x}^*}} \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x}^*)) \rceil$. For $b = 1$, the challenger samples $\mathbf{r}_{\mathbf{x}^*} \leftarrow \mathbb{Z}_q^m$ and outputs $\lceil \mathbf{r}_{\mathbf{x}^*}^T \rceil$.
5. Repeat step 3.
6. The attacker outputs b' and wins if and only if $b' = b$.

Hybrid 1: In Hybrid 1, the challenger sets pp as $\mathbf{B}_1 + f[1]\mathbf{G}, \dots, \mathbf{B}_L + f[L]\mathbf{G}, \mathbf{A}_0, \mathbf{A}_1$. As a result, constrained keys will be in the form of $(\mathbf{s}^T \cdot [\mathbf{B}_1 | \dots | \mathbf{B}_L] + \mathbf{e}^T, \mathcal{C})$.

2. The challenger samples $\mathbf{B}_1, \dots, \mathbf{B}_L \leftarrow \mathbb{Z}_q^{n \times m}$, $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, $\mathbf{A}_0, \mathbf{A}_1 \leftarrow \mathbb{Z}_q^{n \times m}$. The challenger outputs $\mathbf{B}_1 + f[1]\mathbf{G}, \dots, \mathbf{B}_L + f[L]\mathbf{G}, \mathbf{A}_0, \mathbf{A}_1$. It samples a bit $b \leftarrow \{0, 1\}$.
3. The attacker is allowed to make polynomial evaluation and constrained key queries.
 - The constrained key query is nearly identical to Hybrid 0, except that the challenger sets sk as $(\mathbf{s}^T \cdot [\mathbf{B}_1 | \dots | \mathbf{B}_L] + \mathbf{e}^T, \mathcal{C})$.
 - The evaluation query is the same as Hybrid 0, except that if $b = 0$, the challenger sets $\mathbf{B}_{\mathcal{U}_x} := [\mathbf{B}_1 + f[1]\mathbf{G} | \dots | \mathbf{B}_L + f[L]\mathbf{G}] \cdot \mathbf{H}_{\mathcal{U}_x}$.

Hybrid 2: During each evaluation/challenge query, $\mathcal{A}$ outputs $\mathbf{x}$ (or $\mathbf{x}^*$). The challenger answers those queries using the constrained evaluation algorithm by generating constrained keys for $f(\mathbf{x})$ in the way that is identical to the constrained key queries.

3. The attacker is allowed to make polynomial evaluation and constrained key queries.
 - The constrained key queries remain the same as Hybrid 1.
 - During each evaluation query, the attacker outputs $\mathbf{x}$. If $f(\mathbf{x}) \in \mathcal{I}$ for any $\mathcal{I} \in \mathcal{Y}$, the game aborts and the attacker fails. Otherwise, it adds $\mathbf{x}$ to set $\mathcal{X}$. The challenger computes $\mathbf{H}_{\mathcal{U}_{x,f}} := \text{EvalFX}(\mathcal{U}_x, f, \{\mathbf{B}_i\}_{i \in L})$. Let $\mathbf{y}$ be the output of $f(\mathbf{x})$, the challenger samples $\mathbf{e} \leftarrow \chi_1^{Lm}$, $\mathbf{e}_y \leftarrow \chi_1^m$. It outputs

$$\left[((\mathbf{s}^T \cdot [\mathbf{B}_1 | \dots | \mathbf{B}_L] + \mathbf{e}^T) \cdot \mathbf{H}_{\mathcal{U}_{x,f}} + \mathbf{s}^T \cdot \mathbf{A}(\mathbf{y}) + \mathbf{e}_y^T) \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \right].$$

4. During the challenge query, the attacker outputs $\mathbf{x}^*$. If $\mathbf{x}^* \in \mathcal{X}$, the game aborts and the challenger fails immediately. Otherwise if $b = 0$, the challenge query will be handled identically as the above evaluation query (using $\mathbf{x}^*$ instead of $\mathbf{x}$). If $b = 1$, the challenger samples $\mathbf{r}_{\mathbf{x}^*} \leftarrow \mathbb{Z}_q^m$ and outputs $\lceil \mathbf{r}_{\mathbf{x}^*}^T \rceil$.

Hybrid 2.i.0 ($1 \leq i \leq \ell'$): It is the same as Hybrid 2, except for the highlighted part: For each $\mathbf{y}_\perp$ during the constrained key queries, if length of $\mathbf{y}_\perp$ is larger than i , the challenger samples uniformly random $\mathbf{s}_{\mathbf{y}[[i-1]]}$ and $\mathbf{r}_{\mathbf{y}[[k]]}$ for $k \in [j-1]$ and uses these vectors to compute $\mathbf{c}_{\mathbf{y}_\perp}$. Otherwise, the challenger directly samples uniformly random $\mathbf{c}_{\mathbf{y}_\perp}$. During each evaluation query, the challenger samples uniformly random $\mathbf{s}_{\mathbf{y}[[i-1]]}$ and $\mathbf{r}_{\mathbf{y}[[j]]}$, and computes $\mathbf{c}_{\mathbf{y}}$ accordingly.

3. The attacker is allowed to make polynomial evaluation and constrained key queries.
 - During each constrained key query, the attacker outputs $\mathbf{y}, \mathbf{z}$. If there exists any range $\mathcal{I} \in \mathcal{Y}$ such that $[\mathbf{y}, \mathbf{z}]$ intersects with $\mathcal{I}$, or there exists any $\mathbf{x} \in \mathcal{X}$ such that $f(\mathbf{x}) \in [\mathbf{y}, \mathbf{z}]$, the game aborts and the attacker fails. Otherwise, the challenger adds the interval $[\mathbf{y}, \mathbf{z}]$ to $\mathcal{I}$. Next, it runs $\text{IntervalPartition}(\mathbf{y}, \mathbf{z})$ and obtains $\mathcal{Y}_\perp, \mathcal{J}$. For each $\mathbf{y}_\perp \in \mathcal{Y}_\perp$ and the corresponding $j \in \mathcal{J}$, there are two cases:

First consider that $\ell' - j \geq i$: The challenger samples fresh $\mathbf{e}_{\mathbf{y}_\perp} \leftarrow \chi_1^m$. Next, it takes out the first i bits of $\mathbf{y}$ as $\mathbf{y}_i$. It samples $\mathbf{s}_{\mathbf{y}[[i-1]]} \leftarrow \mathbb{Z}_q^n$ and $\mathbf{r}_{\mathbf{y}[[k]]} \leftarrow \mathbb{Z}_q^m$ for all $k \in [i-1]$

(if such $\mathbf{s}_{\mathbf{y}[[i]]}$ or $\mathbf{r}_{\mathbf{y}[[k]]}$ was not chosen in any of the previous queries). It sets $\mathbf{c}_{\mathbf{y}_\perp}$ such that

$$\mathbf{c}_{\mathbf{y}_\perp}^T = \mathbf{s}_{\mathbf{y}[[i-1]]}^T \cdot \mathbf{A}_{\mathbf{y}[[i]]} \prod_{k=i+1}^{\ell'-j} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{y}[[k]]}) + \sum_{k=1}^{i-1} \left(\mathbf{r}_{\mathbf{y}[[k]]}^T \prod_{t=k+1}^{\ell'-j} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{y}[[t]]}) \right) + \mathbf{e}_{\mathbf{y}_\perp}^T.$$

Second if $\ell' - j < i$: The challenger samples $\mathbf{c}_{\mathbf{y}_\perp} \leftarrow \mathbb{Z}_q^m$ and adds $\mathbf{c}_{\mathbf{y}_\perp}$ to the set $\mathcal{C}$.

After setting up $\mathbf{c}_{\mathbf{y}_\perp}$ for all $\mathbf{y}_\perp \in \mathcal{Y}_\perp$, the challenger sets $\mathbf{b}$: If $i = 1$, it sets $\mathbf{b}$ such that $\mathbf{b}^T = \mathbf{s}_{\mathbf{y}[[i-1]]}^T \cdot [\mathbf{B}_1 | \dots | \mathbf{B}_L]$, otherwise for $2 \leq i \leq \ell'$, the challenger samples $\mathbf{b} \leftarrow \mathbb{Z}_q^{Lm}$ (if $\mathbf{b}$ was not sampled in any previous query). Next, the challenger samples fresh $\mathbf{e} \leftarrow \chi_1^{Lm}$. It outputs $(\mathbf{b}^T + \mathbf{e}^T, \mathcal{C})$.

- During each evaluation query, the attacker outputs $\mathbf{x}$. If $f(\mathbf{x}) \in \mathcal{I}$ for any $\mathcal{I} \in \mathcal{Y}$, the game aborts and the attacker fails. Otherwise, it adds $\mathbf{x}$ to set $\mathcal{X}$. The challenger computes $\mathbf{H}_{\mathcal{U}_{\mathbf{x},f}} := \text{EvalFX}(\mathcal{U}_{\mathbf{x},f}, \{\mathbf{B}_i\}_{i \in L})$. It samples $\mathbf{s}_{\mathbf{y}[[i-1]]} \leftarrow \mathbb{Z}_q^n$ and $\mathbf{r}_{\mathbf{y}[[j]]} \leftarrow \mathbb{Z}_q^m$ for all $j \in [i-1]$ (if such $\mathbf{s}_{\mathbf{y}[[i]]}$ or $\mathbf{r}_{\mathbf{y}[[j]]}$ was not chosen). If $i = 1$, it sets $\mathbf{b}$ such that $\mathbf{b}^T = \mathbf{s}_{\mathbf{y}[[i-1]]}^T \cdot [\mathbf{B}_1 | \dots | \mathbf{B}_L]$, otherwise for $2 \leq i \leq \ell'$, the challenger samples $\mathbf{b} \leftarrow \mathbb{Z}_q^{Lm}$ (if $\mathbf{b}$ was not sampled before). Next, the challenger samples fresh $\mathbf{e}_{\mathbf{y}} \leftarrow \chi_1^m$. It sets $\mathbf{c}_{\mathbf{y}}$ such that

$$\mathbf{c}_{\mathbf{y}}^T = \mathbf{s}_{\mathbf{y}[[i-1]]}^T \cdot \mathbf{A}_{\mathbf{y}[[i]]} \prod_{j=i+1}^{\ell'} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{y}[[j]]}) + \sum_{j=1}^{i-1} \left(\mathbf{r}_{\mathbf{y}[[j]]}^T \prod_{k=j+1}^{\ell'} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{y}[[k]]}) \right) + \mathbf{e}_{\mathbf{y}}^T.$$

It outputs $\lceil (\mathbf{b}^T \cdot \mathbf{H}_{\mathcal{U}_{\mathbf{x},f}} + \mathbf{c}_{\mathbf{y}}^T) \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \rceil$.

4. During the challenge query, the attacker outputs $\mathbf{x}^*$. If $\mathbf{x}^* \in \mathcal{X}$, the game aborts and the attacker fails. Otherwise if $b = 0$, the challenger will reply to the challenge query according to the process of the above evaluation query (using $\mathbf{x}^*$ instead of $\mathbf{x}$). If $b = 1$, the challenger samples $\mathbf{r}_{\mathbf{x}^*} \leftarrow \mathbb{Z}_q^m$ and outputs $\lceil \mathbf{r}_{\mathbf{x}^*}^T \rceil$.

Hybrid 2.i.1 ($1 \leq i \leq \ell'$): It is the same as Hybrid 2.i.0, except for the highlighted sentences: The challenger additionally samples LWE error $\mathbf{e}'_{\mathbf{y}[[i]]}$ (and $\mathbf{e}'$ for $i = 1$) from error distribution χ_0 . These errors are included in the computation of $\mathbf{c}_{\mathbf{y}_\perp}$ and $\mathbf{b}$.

3. The attacker is allowed to make polynomial evaluation and constrained key queries.

- During each constrained key query, the attacker outputs $\mathbf{y}, \mathbf{z}$. If there exists any range $\mathcal{I} \in \mathcal{Y}$ such that $[\mathbf{y}, \mathbf{z}]$ intersects with $\mathcal{I}$, or there exists any $\mathbf{x} \in \mathcal{X}$ such that $f(\mathbf{x}) \in [\mathbf{y}, \mathbf{z}]$, the game aborts and the attacker fails. Otherwise, the challenger adds the interval $[\mathbf{y}, \mathbf{z}]$ to $\mathcal{I}$. Next, it runs $\text{IntervalPartition}(\mathbf{y}, \mathbf{z})$ and obtains $\mathcal{Y}_\perp, \mathcal{J}$. For each $\mathbf{y}_\perp \in \mathcal{Y}_\perp$ and the corresponding $j \in \mathcal{J}$, there are two cases:

First consider that $\ell' - j \geq i$: The challenger samples fresh $\mathbf{e}_{\mathbf{y}_\perp} \leftarrow \chi_1^m$. Next, it takes out the first i bits of $\mathbf{y}$ as $\mathbf{y}_i$. It samples $\mathbf{e}' \leftarrow \chi_0^{Lm}$, $\mathbf{s}_{\mathbf{y}[[i-1]]} \leftarrow \mathbb{Z}_q^n$, $\mathbf{e}'_{\mathbf{y}[[i]]} \leftarrow \chi_0^m$, and $\mathbf{r}_{\mathbf{y}[[k]]} \leftarrow \mathbb{Z}_q^m$ for all $k \in [i-1]$ (if such $\mathbf{e}'$, $\mathbf{s}_{\mathbf{y}[[i]]}$, $\mathbf{e}_{\mathbf{y}[[i]]}$, or $\mathbf{r}_{\mathbf{y}[[k]]}$ was not chosen in any

of the previous queries). Next, the challenger samples fresh $\mathbf{e}_{\mathbf{y}_+} \leftarrow \chi_1^m$. It sets $\mathbf{c}_{\mathbf{y}_+}$ such that

$$\mathbf{c}_{\mathbf{y}_+}^T = \left(\mathbf{s}_{\mathbf{y}_{[[i-1]]}}^T \cdot \mathbf{A}_{\mathbf{y}_{[i]}} + \mathbf{e}_{\mathbf{y}_{[[i]]}}'^T \right) \prod_{k=i+1}^{\ell'-j} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{y}_{[k]}}) + \sum_{k=1}^{i-1} \left(\mathbf{r}_{\mathbf{y}_{[[k]]}}^T \prod_{t=k+1}^{\ell'-j} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{y}_{[t]}}) \right) + \mathbf{e}_{\mathbf{y}_+}^T.$$

Second if $\ell' - j < i$: The challenger samples $\mathbf{c}_{\mathbf{y}_+} \leftarrow \mathbb{Z}_q^m$ and adds $\mathbf{c}_{\mathbf{y}_+}$ to the set $\mathcal{C}$.

After setting up $\mathbf{c}_{\mathbf{y}_+}$ for all $\mathbf{y}_+$, the challenger sets $\mathbf{b}$: If $i = 1$, it sets $\mathbf{b}$ such that $\mathbf{b}^T = \mathbf{s}_{\mathbf{y}_{[[i-1]]}}^T \cdot [\mathbf{B}_1 | \dots | \mathbf{B}_L] + \mathbf{e}'^T$, otherwise for $2 \leq i \leq \ell'$, the challenger samples $\mathbf{b} \leftarrow \mathbb{Z}_q^{Lm}$ (if $\mathbf{b}$ was not sampled in any previous query). Next, the challenger samples fresh $\mathbf{e}_{\mathbf{y}} \leftarrow \chi_1^m$. It outputs $(\mathbf{b}^T + \mathbf{e}^T, \mathcal{C})$.

- During each evaluation query, the attacker outputs $\mathbf{x}$. If $f(\mathbf{x}) \in \mathcal{I}$ for any $\mathcal{I} \in \mathcal{Y}$, the game aborts and the attacker fails. Otherwise, it adds $\mathbf{x}$ to set $\mathcal{X}$. If $b = 0$, the challenger computes $\mathbf{H}_{\mathcal{U}_{\mathbf{x},f}} := \text{EvalFX}(\mathcal{U}_{\mathbf{x},f}, \{\mathbf{B}_i\}_{i \in L})$. It samples $\mathbf{e}' \leftarrow \chi_0^{Lm}$, $\mathbf{s}_{\mathbf{y}_{[[i-1]]}} \leftarrow \mathbb{Z}_q^n$, $\mathbf{e}'_{\mathbf{y}_{[[i]]}} \leftarrow \chi_0^m$, and $\mathbf{r}_{\mathbf{y}_{[[j]]}} \leftarrow \mathbb{Z}_q^m$ for all $j \in [i-1]$ (if such $\mathbf{e}'$, $\mathbf{s}_{\mathbf{y}_{[[i]]}}$, $\mathbf{e}_{\mathbf{y}_{[[i]]}}$, or $\mathbf{r}_{\mathbf{y}_{[[k]]}}$ was not chosen in any of the previous queries). If $i = 1$, it sets $\mathbf{b}$ such that $\mathbf{b}^T = \mathbf{s}_{\mathbf{y}_{[[i-1]]}}^T \cdot [\mathbf{B}_1 | \dots | \mathbf{B}_L] + \mathbf{e}'^T$, otherwise for $2 \leq i \leq \ell'$, the challenger samples $\mathbf{b} \leftarrow \mathbb{Z}_q^{Lm}$ (if $\mathbf{b}$ was not sampled before). Next, the challenger samples fresh $\mathbf{e}_{\mathbf{y}} \leftarrow \chi_1^m$. It sets $\mathbf{c}_{\mathbf{y}}$ such that

$$\mathbf{c}_{\mathbf{y}}^T = \left(\mathbf{s}_{\mathbf{y}_{[[i-1]]}}^T \cdot \mathbf{A}_{\mathbf{y}_{[i]}} + \mathbf{e}'_{\mathbf{y}_{[[i]]}}'^T \right) \prod_{j=i+1}^{\ell'} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{y}_{[j]}}) + \sum_{j=1}^{i-1} \left(\mathbf{r}_{\mathbf{y}_{[[j]]}}^T \prod_{k=j+1}^{\ell'} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{y}_{[k]}}) \right) + \mathbf{e}_{\mathbf{y}}^T.$$

It outputs $\lceil (\mathbf{b}^T \cdot \mathbf{H}_{\mathcal{U}_{\mathbf{x},f}} + \mathbf{c}_{\mathbf{y}}^T) \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \rceil$. Otherwise if $b = 1$ the challenger samples $\mathbf{r}_{\mathbf{x}} \leftarrow \mathbb{Z}_q^m$ if $\mathbf{r}_{\mathbf{x}}$ was not sampled in any previous query and outputs $\lceil \mathbf{r}_{\mathbf{x}}^T \rceil$.

4. During the challenge query, the attacker outputs $\mathbf{x}^*$. If $\mathbf{x}^* \in \mathcal{X}$, the game aborts and the attacker fails. Otherwise if $b = 0$, the challenger will follow the above evaluation query. If $b = 1$, the challenger samples $\mathbf{r}_{\mathbf{x}^*} \leftarrow \mathbb{Z}_q^m$ and outputs $\lceil \mathbf{r}_{\mathbf{x}^*}^T \rceil$.

Hybrid 2.i.2 ($1 \leq i \leq \ell'$): The difference between Hybrid 2.i.1 and Hybrid 2.i.2 is that the challenger replaces $\mathbf{s}_{\mathbf{y}_{[[i-1]]}}^T \cdot \mathbf{A}_{\mathbf{y}_{[i]}} + \mathbf{e}'_{\mathbf{y}_{[[i]]}}'^T$ using $\mathbf{r}_{\mathbf{y}_{[[i]]}}$ when generating $\mathbf{c}_{\mathbf{y}_+}$, and replaces $\mathbf{b}$ with a uniform random string if $i = 1$.

3. The attacker is allowed to make polynomial evaluation and constrained key queries.

- During each constrained key query, the attacker outputs $\mathbf{y}, \mathbf{z}$. If there exists any range $\mathcal{I} \in \mathcal{Y}$ such that $[\mathbf{y}, \mathbf{z}]$ intersects with $\mathcal{I}$, or there exists any $\mathbf{x} \in \mathcal{X}$ such that $f(\mathbf{x}) \in [\mathbf{y}, \mathbf{z}]$, the game aborts and the attacker fails. Next, it runs $\text{IntervalPartition}(\mathbf{y}, \mathbf{z})$ and obtains $\mathcal{Y}_+, \mathcal{J}$. For each $\mathbf{y}_+ \in \mathcal{Y}_+$ and the corresponding $j \in \mathcal{J}$, there are two cases:
First consider that $\ell' - j \geq i$: The challenger samples fresh $\mathbf{e}_{\mathbf{y}_+} \leftarrow \chi_1^m$. Next, it takes out the first i bits of $\mathbf{y}$ as $\mathbf{y}_i$. It samples $\mathbf{r}_{\mathbf{y}_{[[k]]}} \leftarrow \mathbb{Z}_q^m$ for all $k \in [i]$ (if such $\mathbf{r}_{\mathbf{y}_{[[k]]}}$ was not chosen). Next, the challenger samples $\mathbf{b} \leftarrow \mathbb{Z}_q^{Lm}$ (if $\mathbf{b}$ was not sampled before). Then the challenger samples fresh $\mathbf{e}_{\mathbf{y}_+} \leftarrow \chi_1^m$. It sets $\mathbf{c}_{\mathbf{y}_+}$ such that

$$\mathbf{c}_{\mathbf{y}_+}^T = \mathbf{r}_{\mathbf{y}[[i]]}^T \prod_{k=i+1}^{\ell'-j} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{y}[k]}) + \sum_{k=1}^{i-1} \left(\mathbf{r}_{\mathbf{y}[[k]]}^T \prod_{t=k+1}^{\ell'-j} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{y}[t]}) \right) + \mathbf{e}_{\mathbf{y}_+}^T.$$

Second if $\ell' - j < i$: The challenger samples $\mathbf{c}_{\mathbf{y}_+} \leftarrow \mathbb{Z}_q^m$ and adds $\mathbf{c}_{\mathbf{y}_+}$ to the set $\mathcal{C}$. The challenger outputs $\mathbf{sk} := (\mathbf{b}^T + \mathbf{e}^T, \mathcal{C})$.

- During each evaluation query, the attacker outputs $\mathbf{x}$. If $f(\mathbf{x}) \in \mathcal{I}$ for any $\mathcal{I} \in \mathcal{Y}$, the game aborts and the attacker fails. Otherwise, it adds $\mathbf{x}$ to set $\mathcal{X}$. If $b = 0$, the challenger computes $\mathbf{H}_{\mathcal{U}_{\mathbf{x},f}} := \text{EvalFX}(\mathcal{U}_{\mathbf{x}}, f, \{\mathbf{B}_i\}_{i \in L})$. It samples $\mathbf{r}_{\mathbf{y}[[j]]} \leftarrow \mathbb{Z}_q^m$ for all $j \in [i]$ (if such $\mathbf{s}_{\mathbf{y}[[i]]}$ or $\mathbf{r}_{\mathbf{y}[[j]]}$ was not chosen). The challenger samples fresh $\mathbf{e} \leftarrow \chi_1^{Lm}$. Next, the challenger samples $\mathbf{b} \leftarrow \mathbb{Z}_q^{Lm}$ (if $\mathbf{b}$ was not sampled before). Then the challenger samples fresh $\mathbf{e}_{\mathbf{y}} \leftarrow \chi_1^m$. It sets $\mathbf{c}_{\mathbf{y}}$ such that

$$\mathbf{c}_{\mathbf{y}}^T = \mathbf{r}_{\mathbf{y}[[i]]}^T \prod_{j=i+1}^{\ell'} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{y}[j]}) + \sum_{j=1}^{i-1} \left(\mathbf{r}_{\mathbf{y}[[j]]}^T \prod_{k=j+1}^{\ell'} \mathbf{G}^{-1}(\mathbf{A}_{\mathbf{y}[k]}) \right) + \mathbf{e}_{\mathbf{y}}^T.$$

It outputs $\lceil (\mathbf{b}^T \cdot \mathbf{H}_{\mathcal{U}_{\mathbf{x},f}} + \mathbf{c}_{\mathbf{y}}^T) \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \rceil$.

4. During the challenge query, the attacker outputs $\mathbf{x}^*$. If $\mathbf{x}^* \in \mathcal{X}$, the game aborts and the attacker fails. Otherwise if $b = 0$, the challenger will follow the above evaluation query. If $b = 1$, the challenger samples $\mathbf{r}_{\mathbf{x}^*} \leftarrow \mathbb{Z}_q^m$ and outputs $\lceil \mathbf{r}_{\mathbf{x}^*}^T \rceil$.

Hybrid 3: It is identical to Hybrid 2.ℓ.2, except for the highlighted part. Hybrid 3 is also statistically close to Hybrid 2.ℓ'.2.

3. The attacker is allowed to make polynomial evaluation and constrained key queries. **Before the queries begin, the challenger samples $\mathbf{b} \leftarrow \mathbb{Z}_q^{Lm}$.**

- During each constrained key query, the attacker outputs $\mathbf{y}, \mathbf{z}$. If there exists any range $\mathcal{I} \in \mathcal{Y}$ such that $[\mathbf{y}, \mathbf{z}]$ intersects with $\mathcal{I}$, or there exists any $\mathbf{x} \in \mathcal{X}$ such that $f(\mathbf{x}) \in [\mathbf{y}, \mathbf{z}]$, the game aborts and the attacker fails. Otherwise, the challenger adds the interval $[\mathbf{y}, \mathbf{z}]$ to $\mathcal{I}$. Next, it runs $\text{IntervalPartition}(\mathbf{y}, \mathbf{z})$ and obtains $\mathcal{Y}_+, \mathcal{J}$. For each $\mathbf{y}_+ \in \mathcal{Y}_+$, it samples $\mathbf{c}_{\mathbf{y}_+} \leftarrow \mathbb{Z}_q^m$ and adds $\mathbf{c}_{\mathbf{y}_+}$ to the set $\mathcal{C}$. After setting up the range keys for all $\mathbf{y} \in \mathcal{Y}_+$, the challenger samples $\mathbf{e} \leftarrow \chi_1^{Lm}$ and outputs $\mathbf{sk} := (\mathbf{b}^T + \mathbf{e}^T, \mathcal{C})$.
- During each evaluation query, the attacker outputs $\mathbf{x}$. If $f(\mathbf{x}) \in \mathcal{I}$ for any $\mathcal{I} \in \mathcal{Y}$, the game aborts and the attacker fails. Otherwise, it adds $\mathbf{x}$ to set $\mathcal{X}$. Let $\mathbf{y}$ be $f(\mathbf{x})$. **It computes $\mathbf{H}_{\mathcal{U}_{\mathbf{x},f}} := \text{EvalFX}(\mathcal{U}_{\mathbf{x}}, f, \{\mathbf{B}_i\}_{i \in L})$. If $\mathbf{c}_{\mathbf{y}}$ was not sampled before, the challenger additionally samples $\mathbf{c}_{\mathbf{y}} \leftarrow \mathbb{Z}_q^m$. It outputs**

$$\lceil (\mathbf{b}^T \cdot \mathbf{H}_{\mathcal{U}_{\mathbf{x},f}} + \mathbf{c}_{\mathbf{y}}^T) \cdot \mathbf{G}^{-1}(\mathbf{A}(\mathbf{x})) \rceil.$$

4. During the challenge query, the attacker outputs $\mathbf{x}^*$. If $\mathbf{x}^* \in \mathcal{X}$, the game aborts and the attacker fails. **Otherwise if $b = 0$, the challenger will follow the above evaluation query.** If $b = 1$, the challenger samples $\mathbf{r}_{\mathbf{x}^*} \leftarrow \mathbb{Z}_q^m$ and outputs $\lceil \mathbf{r}_{\mathbf{x}^*}^T \rceil$.

We will omit the intermediate hybrids between Hybrid 3 and Hybrid 4 since the analysis is identical to 6.2.2.

Hybrid 4: All of the evaluation queries and the challenge query in Hybrid 4 will be answered using random strings.

3. The attacker is allowed to make polynomial evaluation and constrained key queries. Before the queries begin, the challenger samples $\mathbf{b} \leftarrow \mathbb{Z}_q^{Lm}$.
 - Constrained key queries remain unchanged from the previous hybrid.
 - During each evaluation query, the attacker outputs $\mathbf{x}$. If $f(\mathbf{x}) \in \mathcal{I}$ for any $\mathcal{I} \in \mathcal{Y}$, the game aborts and the attacker fails. The challenger adds $\mathbf{x}$ to $\mathcal{X}$. **The challenger samples $\mathbf{r}_{\mathbf{x}} \leftarrow \mathbb{Z}_q^m$ (if $\mathbf{r}_{\mathbf{x}}$ was not sampled in any previous query) and outputs $\lceil \mathbf{r}_{\mathbf{x}}^T \rceil$.**
4. During the challenge query, the attacker outputs $\mathbf{x}^*$. If $f(\mathbf{x}^*) \in \mathcal{I}$ for any $\mathcal{I} \in \mathcal{Y}$, the game aborts and the attacker fails. The challenger adds $\mathbf{x}^*$ to $\mathcal{X}$. **For both $b = 0$ and $b = 1$, the challenger samples $\mathbf{r}_{\mathbf{x}^*} \leftarrow \mathbb{Z}_q^m$ and outputs $\lceil \mathbf{r}_{\mathbf{x}^*}^T \rceil$.**

We will show that the above hybrid experiments are computationally indistinguishable. For any adversary $\mathcal{A}$, let $\text{Adv}_{\mathcal{A}}^i$ denote the probability that the adversary wins the game during Hybrid i .

8.3 Indistinguishability of Hybrids

Lemma 8.3. For any adversary $\mathcal{A}$ of unbounded running time, $|\text{Adv}_{\mathcal{A}}^0 - \text{Adv}_{\mathcal{A}}^1| = 0$.

Proof. We omit the proof as it is identical to Theorem 6.3. □

Lemma 8.4. For any adversary $\mathcal{A}$ of unbounded running time, there exists a negligible function $\text{negl}(\cdot)$ such that $|\text{Adv}_{\mathcal{A}}^1 - \text{Adv}_{\mathcal{A}}^2| \leq \text{negl}(\lambda)$.

Proof. We omit the proof as the norm bound analysis is nearly identical to the completeness analysis. The difference between Hybrid 1 and 2 is that the challenge answers evaluation queries in the way that is identical to constrained secret key evaluations. □

Lemma 8.5. For all $i \in [\ell']$, Hybrid 2. i .0 and Hybrid 2. i .1 are statistically close.

Proof. Omitted as it is nearly identical to the proof of Theorem 6.6. □

Lemma 8.6. Assuming that $\text{GLWE}_{n, (L+2)m, q, \chi_0}$ holds, then for any PPT adversary $\mathcal{A}$, there exists a negligible function $\text{negl}(\cdot)$ such that $|\text{Adv}_{\mathcal{A}}^{2.i.1} - \text{Adv}_{\mathcal{A}}^{2.i.2}| \leq \text{negl}(\lambda)$ for all $i \in [\ell']$.

Proof. Omitted as it is nearly identical to the proof of Theorem 6.7. □

Lemma 8.7. For all $i \in [\ell' - 1]$ and for any unbounded adversary $\mathcal{A}$, $|\text{Adv}_{\mathcal{A}}^{2.i.2} - \text{Adv}_{\mathcal{A}}^{2.(i+1).0}| = 0$.

Proof. Omitted as it is nearly identical to the proof of Theorem 6.8. □

Lemma 8.8. For any adversary $\mathcal{A}$ of unbounded running time, there exists a negligible function $\text{negl}(\cdot)$, such that $|\text{Adv}_{\mathcal{A}}^{2.\ell'.2} - \text{Adv}_{\mathcal{A}}^3| \leq \text{negl}(\lambda)$.

Proof. The proof is identical to the analysis of Theorem 6.9. □

Lemma 8.9. Assuming that $\text{GLWE}_{n, 2m, q, \chi_0}$ assumption holds, then for any PPT adversary $\mathcal{A}$, there exists a negligible function $\text{negl}(\cdot)$ such that $|\text{Adv}_{\mathcal{A}}^3 - \text{Adv}_{\mathcal{A}}^4| \leq \text{negl}(\lambda)$.

Proof. We omit the proof as it is identical to the argument by Theorems 6.10 to 6.14. □

Combining all of the above lemmas, Theorem 8.2 holds.

9 2-Sided Predicate Encryption for Purified Class

In this section, we design predicate encryption for compute-&-compare circuits of the purified class. And our design can also be generalized to the predicated range class. Our design relies on constrained PRF for compute-&-compare (CPRF.Setup, CPRF.Eval, CPRF.Constrain, CPRF.CEval), secret key encryption scheme (SKE.Setup, SKE.Enc, SKE.Dec), and 1-sided secret key predicate encryption (PE.Setup, PE.Enc, PE.KeyGen, PE.Dec).

Setup($1^\lambda, 1^{\ell_m}, 1^{\ell_x}, 1^{\ell'}, 1^{L_f}, 1^d$): It takes as input security parameter λ , message length ℓ_m , attribute length ℓ_x , function output length ℓ' , description length L_f of the function, and function depth d . Let ℓ be the length of the SKE ciphertext that encrypts an attribute of length ℓ_x . Let L be the length of function CPRF. f where CPRF. f is as defined in the KeyGen algorithm. It generates $(\text{cprf.pp}, \text{cprf.msk}) \leftarrow \text{CPRF.Setup}(1^\lambda, 1^\ell, 1^{\ell'}, 1^L, 1^d)$. Next, it generates $\text{ske.sk} \leftarrow \text{SKE.Setup}(1^\lambda)$, $\text{pe.msk} \leftarrow \text{PE.Setup}(1^\lambda)$. It outputs cprf.pp as pp , and $(\text{cprf.msk}, \text{ske.sk})$ as master secret key msk .

KeyGen($\text{msk}, C_{f,y}$) : It takes as input master secret key msk , and a circuit C defined by function f and y . First define function CPRF. f as follows: Function CPRF. f has ske.sk hardwired and takes as input ske.ct . It outputs $f(\text{Dec}(\text{ske.sk}, \text{ske.ct}))$ and outputs $\mathbf{0}$ if $\text{Dec}(\text{ske.sk}, \text{ske.ct}) = \perp$. The KeyGen algorithm parses msk as $(\text{cprf.msk}, \text{ske.sk})$. Let circuit C be the compute-&-compare circuit specified by function CPRF. f and y . The KeyGen algorithm sets cprf.sk as $\text{Constrain}(\text{cprf.msk}, C)$. Next, let PE. f be the function that takes as input $\text{ske.ct}, \tau$, and has the above cprf.sk hardwired. It outputs 1 if and only if $\text{CPRF.Eval}(\text{cprf.sk}, \text{ske.ct}) = \tau$. The KeyGen algorithm generates and outputs $\text{sk} \leftarrow \text{PE.KeyGen}(\text{pe.msk}, \text{PE.f})$.

Enc($\text{msk}, \mathbf{x}, \mathbf{m}$) : It takes as input master secret key msk , attribute $\mathbf{x}$, and message $\mathbf{m}$. It generates $\text{ske.ct} \leftarrow \text{SKE.Enc}(\text{ske.sk}, \mathbf{x})$, $\tau := \text{CPRF.Eval}(\text{cprf.msk}, \text{ske.ct})$, $\text{ct} \leftarrow \text{PE.Enc}(\text{pe.msk}, (\text{ske.ct}, \tau), \mathbf{m})$. It outputs ct .

Dec(sk, ct) : It outputs $\text{PE.Dec}(\text{sk}, \text{ct})$.

Proof sketch. We provide an overview of the Hybrid experiments and security proof of purified 2-sided security. Hybrid 0 corresponds to the original security experiment. There are two types of attackers:

For type 1 attacker $\mathcal{A}$, both challenge attributes x_0, x_1 lie outside the support of the predicate associated with every secret key query. In this case, reduction can simply choose τ uniformly at random by pseudorandomness of CPRF. By the security guarantee of the underlying one-sided predicate encryption scheme, such attacker cannot have more than $1/2 + \text{negl}(\lambda)$ advantage.

Type 2 attacker corresponds to the case where the challenge messages are identical ($m_0 = m_1$). Let b be the challenger's random bit. We show that even if the adversary is able to decrypt the ciphertext containing m_b , the attribute x_b remains hidden. In Hybrid 1, the challenger generates constrained keys for the encryption and challenge queries and computes CPRF output τ using the constrained evaluation algorithm. Next in Hybrid 2, the constraining function of the underlying compute-&-compare is replaced with a dummy function that always outputs 0. Then, all constrained secret keys generated in this hybrid will be sampled as dummy keys since all possible inputs lie outside the supports of the constraints. In Hybrid 3, the challenger instead uses the all-zero attribute to compute τ during the challenge phase. Hybrid 0 and 1 are indistinguishable

by completeness of the underlying CPRF. Hybrid 1 is indistinguishable from Hybrid 2 by the constraint-hiding of the underlying CPRF. Hybrid 2 is indistinguishable from Hybrid 3 by semantic security of ske , since the CPRF keys are independent of the ske key. We conclude that both type 1 and type 2 attacker has at most $1/2 + \text{negl}(\lambda)$.

Completeness The completeness of our design directly follows from the completeness of the underlying constrained PRF CPRF, secret key encryption, and predicate encryption PE.

9.1 Purified 2-Sided Security

The 2-sided security of predicate encryption for compute-&-compare class follows from Theorem 4.4, except that the adversary has to specify function f before the master secret key is set. Formally, we give the following definition:

Definition 9.1 (Purified 2-Sided Security). An SK-PE scheme PE is said to be (fully) 2-sided secure for compute-&-compare circuits, if for every stateful admissible PPT adversary $\mathcal{A}$, there exists a negligible function $\text{negl}(\cdot)$, such that for every $\lambda \in \mathbb{N}$ the following holds:

$$\Pr \left[\mathcal{A}^{\text{KeyGen}(\text{msk}, \cdot), \text{Enc}(\text{msk}, \cdot, \cdot)}(\text{ct}) = b : \begin{array}{l} (\ell_m, \ell_x, \ell', L_f, d, f) \leftarrow \mathcal{A}(1^\lambda); \\ \text{msk} \leftarrow \text{Setup}(1^\lambda, 1^{\ell_m}, 1^{\ell_x}, 1^{\ell'}, 1^{L_f}, 1^d) \\ ((m_0, m_1), (x_0, x_1)) \leftarrow \mathcal{A}^{\text{KeyGen}(\text{msk}, \cdot), \text{Enc}(\text{msk}, \cdot, \cdot)} \\ b \leftarrow \{0, 1\}; \text{ct} \leftarrow \text{Enc}(\text{msk}, x_b, m_b) \end{array} \right] \leq \frac{1}{2} + \text{negl}(\lambda)$$

where $\mathcal{A}$ is an admissible adversary if: (a) *either* every predicate query C , made by adversary $\mathcal{A}$ to the KeyGen oracle, satisfies the condition that $C(x_0) = C(x_1) = 0$, *or* (b) $m_0 = m_1$, and every predicate query C made by $\mathcal{A}$ satisfies $C(x_0) = C(x_1)$, has the form of $C_{f,y}$, and is never repeated.

Theorem 9.2. Assuming that the underlying compute-&-compare constrained PRF CPRF satisfies pseudorandomness and constraint-hiding, the secret key encryption scheme SKE satisfies semantic security, then the above secret key predicate encryption for compute-&-compare predicate class satisfies attribute hiding.

Proof. In the case that every predicate query C by adversary $\mathcal{A}$ satisfies the condition that $C(x_0) = C(x_1) = 0$,

the predicate hiding property directly follows from the predicate encryption scheme PE.

In our proof, we only focus on the case where $m_0 = m_1$, every predicate query by $\mathcal{A}$ satisfies $C(x_0) = C(x_1)$, and all these circuits are under the compute-&-compare class (the case by 2-sided security).

We start by presenting the following hybrid experiments:

Hybrid 0: This is the original security game.

1. Attacker $\mathcal{A}$ outputs a function f .
2. Challenger tosses coin $b \leftarrow \{0, 1\}$, generates $\text{ske.sk} \leftarrow \text{SKE.Setup}(1^\lambda)$. Let $\text{CPRF}.f$ be the function that has ske.sk hardwired and takes as input ct . Function $\text{CPRF}.f$ outputs $f(\text{Dec}(\text{ske.sk}, \text{ct}))$ and outputs $\mathbf{0}$ if $\text{Dec}(\text{ske.sk}, \text{ct}) = \perp$. Next, it generates $(\text{cprf.pp}, \text{cprf.msk}) \leftarrow \text{CPRF.Setup}(1^\lambda, 1^\ell, 1^{\ell'}, 1^L, 1^d)$, $\text{pe.msk} \leftarrow \text{PE.Setup}(1^\lambda)$. It sets cprf.pp as pp , and $(\text{cprf.msk}, \text{ske.sk}, \text{pe.msk})$ as master secret key msk . It sends pp to $\mathcal{A}$.

3. Attacker $\mathcal{A}$ is allowed polynomial secret key and message queries.

- In each secret key query, $\mathcal{A}$ outputs $\mathbf{y}$. If $\mathbf{y} \in \mathcal{Y}$, the attacker fails and the game aborts. Otherwise, let circuit C be the compute-&-compare circuit over function $\text{CPRF}.f$ and $\mathbf{y}$. The challenger generates $\text{cprf.sk}_{\mathbf{y}} \leftarrow \text{Constrain}(\text{cprf.msk}, C_{\text{CPRF}.f, \mathbf{y}})$. Next, let $\text{PE}.f$ be the function that takes as input $\text{ske.ct}, \tau$, and has the above $\text{cprf.sk}_{\mathbf{y}}$ hardwired. It outputs 1 if and only if $\text{CPRF.Eval}(\text{cprf.sk}_{\mathbf{y}}, \text{ske.ct}) = \tau$. The challenger outputs $\text{sk} \leftarrow \text{PE.KeyGen}(\text{pe.msk}, \text{PE}.f)$. It adds $\mathbf{y}$ to $\mathcal{Y}$.
- In each message query, $\mathcal{A}$ outputs message $\mathbf{m}$ and attribute $\mathbf{x}$. The challenger then generates $\text{ske.ct} \leftarrow \text{SKE.Enc}(\text{ske.sk}, \mathbf{x})$, and $\tau := \text{CPRF.Eval}(\text{cprf.msk}, \text{ske.ct})$. It outputs $\text{ct} \leftarrow \text{PE.Enc}(\text{pe.msk}, (\text{ske.ct}, \tau), \mathbf{m})$.

4. During the challenge query, $\mathcal{A}$ outputs message $\mathbf{m}_0, \mathbf{m}_1$ and attributes $\mathbf{x}_0, \mathbf{x}_1$. If $f(\mathbf{x}_0) \in \mathcal{Y}$ or $f(\mathbf{x}_1) \in \mathcal{Y}$, but $f(\mathbf{x}_0) \neq f(\mathbf{x}_1)$ or $\mathbf{m}_0 \neq \mathbf{m}_1$, then the attacker fails and the game aborts. The challenger then generates $\text{ske.ct} \leftarrow \text{SKE.Enc}(\text{ske.sk}, \mathbf{x}_b)$, and $\tau := \text{CPRF.Eval}(\text{cprf.msk}, \text{ske.ct})$. It outputs $\text{ct} \leftarrow \text{PE.Enc}(\text{pe.msk}, (\text{ske.ct}, \tau), \mathbf{m}_b)$.

5. This is identical to step 3, except that during each secret key query where $\mathcal{A}$ queries $\mathbf{y}$, if $\mathbf{y} = f(\mathbf{x}_0)$ or $\mathbf{y} = f(\mathbf{x}_1)$, but $f(\mathbf{x}_0) \neq f(\mathbf{x}_1)$ or $\mathbf{m}_0 \neq \mathbf{m}_1$, then the attacker fails and the game aborts.

6. The attacker outputs b' . It wins if and only if $b' = b$.

There are two types of attackers: Type 1 attacker is that for every secret key query on predicate C by adversary $\mathcal{A}$ satisfies the condition that $C(\mathbf{x}_0) = C(\mathbf{x}_1) = 0$. Type 2 attacker corresponds to the case where $\mathbf{m}_0 = \mathbf{m}_1$. For type 1 attacker $\mathcal{A}$, we provide the following hybrid after Hybrid 0:

Type 1 attacker Hybrid 1: This is the same as Hybrid 0, except that the challenger sets τ as a random string during the challenge query.

- 4. During the challenge query, $\mathcal{A}$ outputs message $\mathbf{m}_0, \mathbf{m}_1$ and attributes $\mathbf{x}_0, \mathbf{x}_1$. If $f(\mathbf{x}_0) \in \mathcal{Y}$ or $f(\mathbf{x}_1) \in \mathcal{Y}$, but $f(\mathbf{x}_0) \neq f(\mathbf{x}_1)$ or $\mathbf{m}_0 \neq \mathbf{m}_1$, then the attacker fails and the game aborts. The challenger then generates $\text{ske.ct} \leftarrow \text{SKE.Enc}(\text{ske.sk}, \mathbf{x}_b)$, and $\tau \leftarrow \{0, 1\}^L$ where L is the output length of the underlying CPRF. It outputs $\text{ct} \leftarrow \text{PE.Enc}(\text{pe.msk}, (\text{ske.ct}, \tau), \mathbf{m}_b)$.

We will show that type 1 adversary only has negligible advantage in the above experiment.

Lemma 9.3. Assuming that CPRF satisfies pseudorandomness, then for all polynomial time type 1 attacker $\mathcal{A}$, there exists a negligible function $\text{negl}(\cdot)$ such that $\text{Adv}_{\mathcal{A}}^0 - \text{Adv}_{\mathcal{A}}^1 \leq \text{negl}(\lambda)$.

Proof. Assuming that there exists an adversary $\mathcal{A}$ such that $\text{Adv}_{\mathcal{A}}^0 - \text{Adv}_{\mathcal{A}}^1 \geq \epsilon(\lambda)$ where $\epsilon(\cdot)$ is non-negligible, we design a reduction algorithm $\mathcal{B}$ that breaks the pseudorandomness of the underlying CPRF.

The attacker $\mathcal{A}$ outputs function f . $\mathcal{B}$ first tosses a coin $b \leftarrow \{0, 1\}$ and generates $\text{ske.sk} \leftarrow \text{SKE.Setup}(1^\lambda)$. Next, $\mathcal{B}$ outputs f and the constrained PRF challenger outputs cprf.pp . $\mathcal{B}$ then generates $\text{pe.msk} \leftarrow \text{PE.Setup}(1^\lambda)$, and sends cprf.pp to $\mathcal{A}$. Next, during each secret key query, $\mathcal{A}$ outputs $\mathbf{y}$. If $\mathbf{y} \in \mathcal{Y}$, the attacker fails and the game aborts. Additionally, if the challenge query has already been made, and $\mathbf{y} = f(\mathbf{x}_0)$ or $\mathbf{y} = f(\mathbf{x}_1)$, but $f(\mathbf{x}_0) \neq f(\mathbf{x}_1)$ or $\mathbf{m}_0 \neq \mathbf{m}_1$, the attacker

fails and the game aborts. Otherwise, if $\text{sk}_{\mathbf{y}}$ does not exist, $\mathcal{B}$ queries $\mathbf{y}$ and the constrained PRF challenger responds with $\text{sk}_{\mathbf{y}}$. Let $\text{PE}.f$ be the function that takes as input $\text{ske.ct}, \tau$, and has the above $\text{cprf.sk}_{\mathbf{y}}$ hardwired. It outputs 1 if and only if $\text{CPRF.Eval}(\text{cprf.sk}_{\mathbf{y}}, \text{ske.ct}) = \tau$. $\mathcal{B}$ outputs $\text{sk} \leftarrow \text{PE.KeyGen}(\text{pe.msk}, \text{PE}.f)$. $\mathcal{B}$ also adds $\mathbf{y}$ to $\mathcal{Y}$. During each message query, $\mathcal{A}$ queries $\mathbf{m}, \mathbf{x}$. $\mathcal{B}$ then generates $\text{ske.ct} \leftarrow \text{SKE.Enc}(\text{ske.sk}, \mathbf{x})$, and queries the CPRF challenger using ske.ct where the challenger responds using τ . $\mathcal{B}$ outputs $\text{ct} \leftarrow \text{PE.Enc}(\text{pe.msk}, (\text{ske.ct}, \tau), \mathbf{m})$. During the challenge query, $\mathcal{A}$ outputs message $\mathbf{m}_0, \mathbf{m}_1$ and attributes $\mathbf{x}_0, \mathbf{x}_1$. If $f(\mathbf{x}_0) \in \mathcal{Y}$ or $f(\mathbf{x}_1) \in \mathcal{Y}$, but $f(\mathbf{x}_0) \neq f(\mathbf{x}_1)$ or $\mathbf{m}_0 \neq \mathbf{m}_1$, then $\mathcal{A}$ fails and the game aborts. Let $\mathbf{y}_b$ be $f(\mathbf{x}_b)$. $\mathcal{B}$ then generates $\text{ske.ct} \leftarrow \text{SKE.Enc}(\text{ske.sk}, \mathbf{x}_b)$, and queries the CPRF challenger using ske.ct where the challenger responds using τ . $\mathcal{B}$ outputs $\text{ct} \leftarrow \text{PE.Enc}(\text{pe.msk}, (\text{ske.ct}, \tau), \mathbf{m}_b)$. After all of the above queries, $\mathcal{A}$ outputs b' and $\mathcal{B}$ outputs 1 if and only if $b' = b$. $\square$

Lemma 9.4. Assuming that the underlying predicate encryption PE is secure, then there exists a negligible function $\text{negl}(\cdot)$ such that $\text{Adv}_{\mathcal{A}}^1 \leq 1/2 + \text{negl}(\lambda)$.

Proof. Note that in Hybrid 2 for type 1 attacker, τ is randomly sampled. Thus, with only negligible probability, the keys assigned to the adversary can be used to decrypt the message $\mathbf{m}_0$ or $\mathbf{m}_1$. The security of our predicate encryption against type 1 attacker then directly follows from the security of the underlying 1-sided PE. $\square$

Next, for type 2 attacker $\mathcal{A}$, we provide the following hybrids after Hybrid 0:

Type 2 attacker Hybrid 1: It is the same as Hybrid 0, except that the challenger generates constrained keys $\text{cprf.sk}_{\mathbf{y}}$ (for $\mathbf{y} = f(\mathbf{x})$) during the message and challenge queries and uses the constrained evaluation algorithm to compute the pseudorandom function output τ .

3. Attacker $\mathcal{A}$ is allowed polynomial secret key and message queries.

- In each secret key query, $\mathcal{A}$ outputs $\mathbf{y}$. If $\mathbf{y} \in \mathcal{Y}$, the attacker fails and the game aborts. Otherwise, the challenger generates $\text{cprf.sk}_{\mathbf{y}} \leftarrow \text{Constrain}(\text{cprf.msk}, C_{\text{CPRF}.f, \mathbf{y}})$ if $\text{cprf.sk}_{\mathbf{y}}$ does not exist. Next, let $\text{PE}.f$ be the function that takes as input $\text{ske.ct}, \tau$, and has the above $\text{cprf.sk}_{\mathbf{y}}$ hardwired. It outputs 1 if and only if $\text{CPRF.Eval}(\text{cprf.sk}_{\mathbf{y}}, \text{ske.ct}) = \tau$. The challenger outputs $\text{sk} \leftarrow \text{PE.KeyGen}(\text{pe.msk}, \text{PE}.f)$. It adds $\mathbf{y}$ to $\mathcal{Y}$.
- In each message query, $\mathcal{A}$ outputs message $\mathbf{m}$ and attribute $\mathbf{x}$. Let $\mathbf{y}$ be $f(\mathbf{x})$. If $\text{cprf.sk}_{\mathbf{y}}$ does not exist, the challenger generates $\text{cprf.sk}_{\mathbf{y}} \leftarrow \text{Constrain}(\text{cprf.msk}, C_{\text{CPRF}.f, \mathbf{y}})$. The challenger then generates $\text{ske.ct} \leftarrow \text{SKE.Enc}(\text{ske.sk}, \mathbf{x})$, and $\tau := \text{CPRF.CEval}(\text{cprf.sk}_{\mathbf{y}}, \text{ske.ct})$. It outputs $\text{ct} \leftarrow \text{PE.Enc}(\text{pe.msk}, (\text{ske.ct}, \tau), \mathbf{m})$.

4. During the challenge query, $\mathcal{A}$ outputs message $\mathbf{m}_0, \mathbf{m}_1$ and attributes $\mathbf{x}_0, \mathbf{x}_1$. If $f(\mathbf{x}_0) \in \mathcal{Y}$ or $f(\mathbf{x}_1) \in \mathcal{Y}$, but $f(\mathbf{x}_0) \neq f(\mathbf{x}_1)$ or $\mathbf{m}_0 \neq \mathbf{m}_1$, then the attacker fails and the game aborts. Let $\mathbf{y}$ be $f(\mathbf{x})$. If $\text{cprf.sk}_{\mathbf{y}}$ does not exist, the challenger generates $\text{cprf.sk}_{\mathbf{y}} \leftarrow \text{Constrain}(\text{cprf.msk}, C_{\text{CPRF}.f, \mathbf{y}})$. The challenger then generates $\text{ske.ct} \leftarrow \text{SKE.Enc}(\text{ske.sk}, \mathbf{x}_b)$, and $\tau := \text{CPRF.CEval}(\text{cprf.sk}_{\mathbf{y}}, \text{ske.ct})$. It outputs $\text{ct} \leftarrow \text{PE.Enc}(\text{pe.msk}, (\text{ske.ct}, \tau), \mathbf{m}_b)$.

Type 2 attacker Hybrid 2: It is identical to Hybrid 1, except for the following: The challenger will set the constrained function $\text{CPRF}.f$ as a function that always outputs $\mathbf{0}$. Additionally, the challenger uses random $\mathbf{y}'$ to generate constrained keys.

2. Challenger tosses coin $b \leftarrow \{0, 1\}$, generates $\text{ske.sk} \leftarrow \text{SKE.Setup}(1^\lambda)$. Let $\text{CPRF}.f$ be the function that has input length ℓ and output length ℓ' : It takes as input in $\{0, 1\}^\ell$ and always outputs $\mathbf{0}$. Next, it generates $(\text{cprf.pp}, \text{cprf.msk}) \leftarrow \text{CPRF.Setup}(1^\lambda, 1^\ell, 1^{\ell'}, 1^L, 1^d)$, $\text{pe.msk} \leftarrow \text{PE.Setup}(1^\lambda)$. It sets cprf.pp as pp , and $(\text{cprf.msk}, \text{ske.sk}, \text{pe.msk})$ as master secret key msk . It sends pp to $\mathcal{A}$.
3. Attacker $\mathcal{A}$ is allowed polynomial secret key and message queries.
 - In each secret key query, $\mathcal{A}$ outputs $\mathbf{y}$. If $\mathbf{y} \in \mathcal{Y}$, the attacker fails and the game aborts. Otherwise, if $\text{cprf.sk}_{\mathbf{y}}$ does not exist, the challenger samples $\mathbf{y}' \leftarrow \{0, 1\}^{\ell'}$ and generates $\text{cprf.sk}_{\mathbf{y}} \leftarrow \text{Constrain}(\text{cprf.msk}, C_{\text{CPRF}.f, \mathbf{y}'})$. Next, let $\text{PE}.f$ be the function that takes as input $\text{ske.ct}, \tau$, and has the above $\text{cprf.sk}_{\mathbf{y}}$ hardwired. It outputs 1 if and only if $\text{CPRF.Eval}(\text{cprf.sk}_{\mathbf{y}}, \text{ske.ct}) = \tau$. The challenger outputs $\text{sk} \leftarrow \text{PE.KeyGen}(\text{pe.msk}, \text{PE}.f)$. It adds $\mathbf{y}$ to $\mathcal{Y}$.
 - In each message query, $\mathcal{A}$ outputs message $\mathbf{m}$ and attribute $\mathbf{x}$. Let $\mathbf{y}$ be $f(\mathbf{x})$. If $\text{cprf.sk}_{\mathbf{y}}$ does not exist, the challenger samples $\mathbf{y}' \leftarrow \{0, 1\}^{\ell'}$ and generates $\text{cprf.sk}_{\mathbf{y}} \leftarrow \text{Constrain}(\text{cprf.msk}, C_{\text{CPRF}.f, \mathbf{y}'})$. The challenger then generates $\text{ske.ct} \leftarrow \text{SKE.Enc}(\text{ske.sk}, \mathbf{x})$, and $\tau := \text{CPRF.CEval}(\text{cprf.sk}_{\mathbf{y}}, \text{ske.ct})$. It outputs $\text{ct} \leftarrow \text{PE.Enc}(\text{pe.msk}, (\text{ske.ct}, \tau), \mathbf{m}_b)$.
4. During the challenge query, $\mathcal{A}$ outputs message $\mathbf{m}_0, \mathbf{m}_1$ and attributes $\mathbf{x}_0, \mathbf{x}_1$. If $f(\mathbf{x}_0) \in \mathcal{Y}$ or $f(\mathbf{x}_1) \in \mathcal{Y}$, but $f(\mathbf{x}_0) \neq f(\mathbf{x}_1)$ or $\mathbf{m}_0 \neq \mathbf{m}_1$, then the attacker fails and the game aborts. Let $\mathbf{y}$ be $f(\mathbf{x})$. If $\text{cprf.sk}_{\mathbf{y}}$ does not exist, the challenger samples $\mathbf{y}' \leftarrow \{0, 1\}^{\ell'}$ and generates $\text{cprf.sk}_{\mathbf{y}} \leftarrow \text{Constrain}(\text{cprf.msk}, C_{\text{CPRF}.f, \mathbf{y}'})$. The challenger then generates $\text{ske.ct} \leftarrow \text{SKE.Enc}(\text{ske.sk}, \mathbf{x}_b)$, and $\tau := \text{CPRF.CEval}(\text{cprf.sk}_{\mathbf{y}}, \text{ske.ct})$. It outputs $\text{ct} \leftarrow \text{PE.Enc}(\text{pe.msk}, (\text{ske.ct}, \tau), \mathbf{m}_b)$.

Type 2 attacker Hybrid 3: In this Hybrid, the challenger will encrypt $\mathbf{0}$.

4. Challenge queries are identical to the previous hybrid, except that the challenger generates $\text{ske.ct} \leftarrow \text{SKE.Enc}(\text{ske.sk}, \mathbf{0})$.

Lemma 9.5. Assuming that CPRF satisfies completeness, then for all polynomial time type 2 adversary $\mathcal{A}$, there exists a negligible function $\text{negl}(\cdot)$ such that $\text{Adv}_{\mathcal{A}}^0 - \text{Adv}_{\mathcal{A}}^1 \leq \text{negl}(\lambda)$.

Proof. This holds by completeness of the underlying CPRF, because the only difference between Hybrid 0 and 1 is that the challenger uses CEval instead of Eval . $\square$

Lemma 9.6. Assuming that CPRF satisfies constraint-hiding, then for all polynomial time type 2 adversary $\mathcal{A}$, there exists a negligible function $\text{negl}(\cdot)$ such that $\text{Adv}_{\mathcal{A}}^1 - \text{Adv}_{\mathcal{A}}^2 \leq \text{negl}(\lambda)$.

Proof. Assuming that there exists a type 2 adversary $\mathcal{A}$ such that $\text{Adv}_{\mathcal{A}}^1 - \text{Adv}_{\mathcal{A}}^2 \geq \epsilon(\lambda)$ where $\epsilon(\cdot)$ is non-negligible, we design a reduction algorithm $\mathcal{B}$ that breaks the constraint-hiding property of the underlying CPRF.

The attacker $\mathcal{A}$ outputs function f . $\mathcal{B}$ first tosses a coin $b \leftarrow \{0, 1\}$ and generates $\text{ske.sk} \leftarrow \text{SKE.Setup}(1^\lambda)$. Let f_0 be the function that outputs $f(\text{Dec}(\text{ske.sk}, \text{ct}))$ and outputs $\mathbf{0}$ if $\text{Dec}(\text{ske.sk}, \text{ct}) = \perp$,

and f_1 be the function of input length ℓ and output length ℓ' which always outputs $\mathbf{0}$. Next, $\mathcal{B}$ outputs f_0, f_1 and the constrained PRF challenger outputs cprf.pp . $\mathcal{B}$ then generates $\text{pe.msk} \leftarrow \text{PE.Setup}(1^\lambda)$, and sends cprf.pp to $\mathcal{A}$. Next, during each secret key query, $\mathcal{A}$ outputs $\mathbf{y}$. If $\mathbf{y} \in \mathcal{Y}$, the attacker fails and the game aborts. Additionally, if the challenge query has already been made, and $\mathbf{y} = f(\mathbf{x}_0)$ or $\mathbf{y} = f(\mathbf{x}_1)$, but $f(\mathbf{x}_0) \neq f(\mathbf{x}_1)$ or $\mathbf{m}_0 \neq \mathbf{m}_1$, the attacker fails and the game aborts. Otherwise, if $\text{sk}_{\mathbf{y}}$ does not exist, $\mathcal{B}$ samples $\mathbf{y}' \leftarrow \{0, 1\}^{\ell'}$. It queries $\mathbf{y}, \mathbf{y}'$ and the constrained PRF challenger responds with $\text{sk}_{\mathbf{y}}$. Let $\text{PE}.f$ be the function that takes as input $\text{ske.ct}, \tau$, and has the above $\text{cprf.sk}_{\mathbf{y}}$ hardwired. It outputs 1 if and only if $\text{CPRF.Eval}(\text{cprf.sk}_{\mathbf{y}}, \text{ske.ct}) = \tau$. $\mathcal{B}$ outputs $\text{sk} \leftarrow \text{PE.KeyGen}(\text{pe.msk}, \text{PE}.f)$. $\mathcal{B}$ also adds $\mathbf{y}$ to $\mathcal{Y}$. During each message query, $\mathcal{A}$ queries $\mathbf{m}, \mathbf{x}$. Let $\mathbf{y}$ be $f(\mathbf{x})$. If $\text{sk}_{\mathbf{y}}$ does not exist, $\mathcal{B}$ queries $\mathbf{y}$ and the constrained PRF challenger responds with $\text{sk}_{\mathbf{y}}$. $\mathcal{B}$ then generates $\text{ske.ct} \leftarrow \text{SKE.Enc}(\text{ske.sk}, \mathbf{x})$, and $\tau := \text{CPRF.CEval}(\text{sk}_{\mathbf{y}}, \text{ske.ct})$. $\mathcal{B}$ outputs $\text{ct} \leftarrow \text{PE.Enc}(\text{pe.msk}, (\text{ske.ct}, \tau), \mathbf{m})$. During the challenge query, $\mathcal{A}$ outputs message $\mathbf{m}_0, \mathbf{m}_1$ and attributes $\mathbf{x}_0, \mathbf{x}_1$. If $f(\mathbf{x}_0) \in \mathcal{Y}$ or $f(\mathbf{x}_1) \in \mathcal{Y}$, but $f(\mathbf{x}_0) \neq f(\mathbf{x}_1)$ or $\mathbf{m}_0 \neq \mathbf{m}_1$, then $\mathcal{A}$ fails and the game aborts. Let $\mathbf{y}_b$ be $f(\mathbf{x}_b)$. If $\text{cprf.sk}_{\mathbf{y}_b}$ does not exist, $\mathcal{B}$ samples $\mathbf{y}' \leftarrow \{0, 1\}^{\ell'}$ and queries the CPRF challenger using $\mathbf{y}, \mathbf{y}'$ and the constrained PRF challenger responds with $\text{sk}_{\mathbf{y}_b}$. $\mathcal{B}$ then generates $\text{ske.ct} \leftarrow \text{SKE.Enc}(\text{ske.sk}, \mathbf{x}_b)$, and $\tau := \text{CPRF.CEval}(\text{cprf.sk}_{\mathbf{y}_b}, \text{ske.ct})$. $\mathcal{B}$ outputs $\text{ct} \leftarrow \text{PE.Enc}(\text{pe.msk}, (\text{ske.ct}, \tau), \mathbf{m}_b)$. After all of the above queries, $\mathcal{A}$ outputs b' and $\mathcal{B}$ outputs 1 if and only if $b' = b$.

Note that in the view of $\mathcal{A}$, the experiment is identical to Hybrid 1 if the challenger chooses its random coin as 0, and the experiment is identical to Hybrid 2 if the coin is 1. If $\text{Adv}_{\mathcal{A}}^1 - \text{Adv}_{\mathcal{A}}^2 \geq \epsilon(\lambda)$ for some non-negligible $\epsilon(\cdot)$, then $\mathcal{B}$ breaks the constraint-hiding property with advantage greater than or equal to $1/2 + \epsilon(\lambda)$. Thus our lemma follows. $\square$

Lemma 9.7. Assuming that SKE satisfies semantic security, then for all polynomial time adversary $\mathcal{A}$, there exists a negligible function $\text{negl}(\cdot)$ such that $\text{Adv}_{\mathcal{A}}^2 - \text{Adv}_{\mathcal{A}}^3 \leq \text{negl}(\lambda)$.

Proof. Assuming that there exists a type 2 adversary $\mathcal{A}$ such that $\text{Adv}_{\mathcal{A}}^2 - \text{Adv}_{\mathcal{A}}^3 \geq \epsilon(\lambda)$ for some non-negligible $\epsilon(\cdot)$, then we design a reduction algorithm $\mathcal{B}$ that breaks the semantic security of SKE.

$\mathcal{A}$ outputs f . $\mathcal{B}$ sets function $\text{CPRF}.f$ as a function of input length ℓ and output length ℓ' that always outputs $\mathbf{0}$. Next, it tosses a coin $b \leftarrow \{0, 1\}$ and generates $(\text{cprf.pp}, \text{cprf.msk}) \leftarrow \text{CPRF.Setup}(1^\lambda, 1^\ell, 1^{\ell'}, 1^L, 1^d)$, $\text{pe.msk} \leftarrow \text{PE.Setup}(1^\lambda)$. It then sends cprf.pp to $\mathcal{A}$. Next, during each secret key query, $\mathcal{A}$ outputs $\mathbf{y}$. If $\mathbf{y} \in \mathcal{Y}$, the attacker fails and the game aborts. Additionally, if the challenge query has already been made, and $\mathbf{y} = f(\mathbf{x}_0)$ or $\mathbf{y} = f(\mathbf{x}_1)$, but $f(\mathbf{x}_0) \neq f(\mathbf{x}_1)$ or $\mathbf{m}_0 \neq \mathbf{m}_1$, the attacker fails and the game aborts. Otherwise, if $\text{sk}_{\mathbf{y}}$ does not exist, $\mathcal{B}$ samples $\mathbf{y} \leftarrow \{0, 1\}^{\ell'}$ and generates $\text{sk}_{\mathbf{y}} \leftarrow \text{CPRF.Constrain}(\text{cprf.msk}, C_{\text{CPRF}.f, \mathbf{y}'})$. Let $\text{PE}.f$ be the function that takes as input $\text{ske.ct}, \tau$, and has the above $\text{cprf.sk}_{\mathbf{y}}$ hardwired. It outputs 1 if and only if $\text{CPRF.Eval}(\text{cprf.sk}_{\mathbf{y}}, \text{ske.ct}) = \tau$. $\mathcal{B}$ outputs $\text{sk} \leftarrow \text{PE.KeyGen}(\text{pe.msk}, \text{PE}.f)$. $\mathcal{B}$ also adds $\mathbf{y}$ to $\mathcal{Y}$. In each message query, $\mathcal{A}$ queries $\mathbf{m}, \mathbf{x}$. Let $\mathbf{y}$ be $f(\mathbf{m}_b)$. If $\text{sk}_{\mathbf{y}}$ does not exist yet, $\mathcal{B}$ samples $\mathbf{y}' \leftarrow \{0, 1\}^{\ell'}$ generates $\text{sk}_{\mathbf{y}} \leftarrow \text{CPRF.Constrain}(\text{cprf.msk}, \text{CPRF}.f, \mathbf{y}')$. $\mathcal{B}$ then queries the SKE challenger using $\mathbf{x}$ where the challenger responds with ske.ct . $\mathcal{B}$ then sets $\tau := \text{CPRF.CEval}(\text{sk}_{\mathbf{y}}, \text{ske.ct})$. $\mathcal{B}$ outputs $\text{ct} \leftarrow \text{PE.Enc}(\text{pe.msk}, (\text{ske.ct}, \tau), \mathbf{m})$. During the challenge query, $\mathcal{A}$ outputs message $\mathbf{m}_0, \mathbf{m}_1$ and attributes $\mathbf{x}_0, \mathbf{x}_1$. If $f(\mathbf{x}_0) \in \mathcal{Y}$ or $f(\mathbf{x}_1) \in \mathcal{Y}$, but $f(\mathbf{x}_0) \neq f(\mathbf{x}_1)$ or $\mathbf{m}_0 \neq \mathbf{m}_1$, then $\mathcal{A}$ fails and the game aborts. Let $\mathbf{y}_b$ be $f(\mathbf{x}_b)$. If $\text{cprf.sk}_{\mathbf{y}_b}$ does not exist, $\mathcal{B}$ samples $\mathbf{y}' \leftarrow \{0, 1\}^{\ell'}$ and generates $\text{sk}_{\mathbf{y}_b} \leftarrow \text{CPRF.Constrain}(\text{cprf.msk}, C_{\text{CPRF}.f, \mathbf{y}'})$. $\mathcal{B}$ then queries the secret key encryption challenger using $\mathbf{x}_b$ and $\mathbf{0}$ and the challenger responds with ske.ct . $\mathcal{B}$ then sets $\tau :=$

$\text{CPRF.CEval}(\text{cprf.sk}_y, \text{ske.ct})$. $\mathcal{B}$ outputs $\text{ct} \leftarrow \text{PE.Enc}(\text{pe.msk}, (\text{ske.ct}, \tau), \mathbf{m}_b)$. After all queries, $\mathcal{A}$ outputs b' and $\mathcal{B}$ outputs 1 if and only if $b' = b$.

Observe that if the challenger uses the encryption of $\mathbf{x}_b$, then the experiment is identical to Hybrid 2. Otherwise, the above experiment is identical to Hybrid 3. Thus if $\text{Adv}_{\mathcal{A}}^2 - \text{Adv}_{\mathcal{A}}^3$ is non-negligible, $\mathcal{B}$ breaks the SKE semantic security with non-negligible advantage. $\square$

Lemma 9.8. Assuming that PE satisfies semantic security, then for all polynomial time type 2 adversary $\mathcal{A}$, there exists a negligible function $\text{negl}(\cdot)$ such that $\text{Adv}_{\mathcal{A}}^3 \leq 1/2 + \text{negl}(\lambda)$.

Proof. There are two cases: The first case is that $f(\mathbf{x}_0), f(\mathbf{x}_1)$ was queried during the secret key queries and $f(\mathbf{x}_0) = f(\mathbf{x}_1)$, $\mathbf{m}_0 = \mathbf{m}_1$. In this case $\mathcal{A}$ has exactly $1/2$ advantage in Hybrid 3. The second case is that $f(\mathbf{x}_0), f(\mathbf{x}_1) \notin \mathcal{Y}$. In this case, assuming that there exists an adversary $\mathcal{A}$ such that $\text{Adv}_{\mathcal{A}}^3 \geq 1/2 + \epsilon(\lambda)$, we design a reduction algorithm $\mathcal{B}$ that breaks the semantic security of the underlying secret key PE.

$\mathcal{A}$ outputs f . $\mathcal{B}$ sets function $\text{CPRF}.f$ as a function of input length ℓ and output length ℓ' that always outputs $\mathbf{0}$. Next, it generates $(\text{cprf.pp}, \text{cprf.msk}) \leftarrow \text{CPRF.Setup}(1^\lambda, 1^\ell, 1^{\ell'}, 1^L, 1^d)$. It then sends cprf.pp to $\mathcal{A}$. Next, during each secret key query, $\mathcal{A}$ outputs $\mathbf{y}$. If $\mathbf{y} \in \mathcal{Y}$, the attacker fails and the game aborts. Additionally, if the challenge query has already been made, and $\mathbf{y} = f(\mathbf{x}_0)$ or $\mathbf{y} = f(\mathbf{x}_1)$, but $f(\mathbf{x}_0) \neq f(\mathbf{x}_1)$ or $\mathbf{m}_0 \neq \mathbf{m}_1$, the attacker fails and the game aborts. Otherwise, if sk_y does not exist, $\mathcal{B}$ samples $\mathbf{y} \leftarrow \{0, 1\}^{\ell'}$ and generates $\text{sk}_y \leftarrow \text{CPRF.Constrain}(\text{cprf.msk}, C_{\text{CPRF}.f, \mathbf{y}})$. Let $\text{PE}.f$ be the function that takes as input $\text{ske.ct}, \tau$, and has the above cprf.sk_y hardwired. It outputs 1 if and only if $\text{CPRF.Eval}(\text{cprf.sk}_y, \text{ske.ct}) = \tau$. $\mathcal{B}$ queries the PE challenger with pe.msk and the challenger responds using sk . $\mathcal{B}$ sends such sk to $\mathcal{A}$. $\mathcal{B}$ also adds $\mathbf{y}$ to $\mathcal{Y}$. In each message query, $\mathcal{A}$ queries $\mathbf{m}, \mathbf{x}$. Let $\mathbf{y}$ be $f(\mathbf{m}_b)$. If sk_y does not exist yet, $\mathcal{B}$ samples $\mathbf{y}' \leftarrow \{0, 1\}^{\ell'}$ generates $\text{sk}_y \leftarrow \text{CPRF.Constrain}(\text{cprf.msk}, \text{CPRF}.f, \mathbf{y}')$. $\mathcal{B}$ then generates $\text{ske.ct} \leftarrow \text{SKE.Enc}(\text{ske.sk}, \mathbf{x})$, and $\tau := \text{CPRF.CEval}(\text{sk}_y, \text{ske.ct})$. $\mathcal{B}$ queries the PE challenger using attribute $(\text{ske.ct}, \tau)$ and message $\mathbf{m}$, the challenger responds with ct . $\mathcal{B}$ then sends ct to $\mathcal{A}$. During the challenge query, $\mathcal{A}$ outputs message $\mathbf{m}_0, \mathbf{m}_1$ and attributes $\mathbf{x}_0, \mathbf{x}_1$. If $f(\mathbf{x}_0) \in \mathcal{Y}$ or $f(\mathbf{x}_1) \in \mathcal{Y}$, but $f(\mathbf{x}_0) \neq f(\mathbf{x}_1)$ or $\mathbf{m}_0 \neq \mathbf{m}_1$, then $\mathcal{A}$ fails and the game aborts. Let $\mathbf{y}_b$ be $f(\mathbf{x}_b)$. If $\text{cprf.sk}_{\mathbf{y}_b}$ does not exist, $\mathcal{B}$ samples $\mathbf{y}' \leftarrow \{0, 1\}^{\ell'}$ and generates $\text{sk}_{\mathbf{y}_b} \leftarrow \text{CPRF.Constrain}(\text{cprf.msk}, C_{\text{CPRF}.f, \mathbf{y}'})$. $\mathcal{B}$ then generates $\text{ske.ct} \leftarrow \text{SKE.Enc}(\text{ske.sk}, \mathbf{0})$. $\mathcal{B}$ then sets $\tau := \text{CPRF.CEval}(\text{cprf.sk}_y, \text{ske.ct})$. $\mathcal{B}$ sends $\mathbf{m}_0, \mathbf{m}_1$ to the PE challenger and the challenger responds using ct . After all queries, $\mathcal{A}$ outputs b' and $\mathcal{B}$ sends b' to the challenger.

Observe that if the challenger outputs the encryption of $\mathbf{m}_b$, then the experiment is identical to Hybrid 3 under coin b . Thus if $\text{Adv}_{\mathcal{A}}^3 - 1/2$ is non-negligible, $\mathcal{B}$ breaks the PE semantic security with non-negligible advantage. $\square$

Combining these lemmas, our theorem holds. $\square$

9.2 Purified Predicate Privacy

The purified predicate privacy property is identical to Theorem 4.5, except that attacker $\mathcal{A}$ selectively specifies function f_0, f_1 before setting up the master secret key:

Definition 9.9 (Purified Predicate Privacy). An SK-PE scheme PE is said satisfy predicate privacy for compute-&-compare class, if for every stateful admissible PPT adversary $\mathcal{A}$, there exists a

negligible function $\text{negl}(\cdot)$, such that for every $\lambda \in \mathbb{N}$ the following holds:

$$\Pr \left[\mathcal{A}^{O_b(\text{msk}, \cdot, \cdot), \text{Enc}(\text{msk}, \cdot, \cdot)} = b : \begin{array}{l} (f_0, f_1, \ell_m, \ell_x, \ell', L_f, d, f) \leftarrow \mathcal{A}(1^\lambda); b \leftarrow \{0, 1\} \\ \text{msk} \leftarrow \text{Setup}(1^\lambda, 1^{\ell_m}, 1^{\ell_x}, 1^{\ell'}, 1^{L_f}, 1^d) \end{array} \right] \leq \frac{1}{2} + \text{negl}(\lambda)$$

where $\mathcal{A}$ sends two circuits $C_{i,0}, C_{i,1}$ in each query to oracle O_b , and O_b answers it as $\text{KeyGen}(\text{msk}, C_{i,b})$. Moreover, $\mathcal{A}$ is an admissible adversary if for each query to oracle O_b , (i) $C_{i,0}, C_{i,1}$ are in the form of $C_{f_0, y_0}, C_{f_1, y_1}$, (ii) y_0 must be distinct from all other y_0 values in the rest oracle queries and y_1 is distinct from all other y_1 , and (iii) $C_{i,0}(x) = C_{i,1}(x)$ for every attribute x that was queried by $\mathcal{A}$ to the Enc oracle.

Theorem 9.10. Assuming that the underlying compute-&-compare constrained PRF CPRF satisfies constraint-hiding, then the above secret key predicate encryption satisfies predicate privacy for compute-&-compare class.

Proof. We start by presenting the following hybrid experiments:

Hybrid 0: This is the original purified predicate privacy game.

1. Attacker $\mathcal{A}$ outputs f_0, f_1 .
2. Challenger tosses coin $b \leftarrow \{0, 1\}$, generates $\text{ske.sk} \leftarrow \text{SKE.Setup}(1^\lambda)$. Let $\text{CPRF}.f$ be the function that has ske.sk hardwired and takes as input ct . Function $\text{CPRF}.f$ outputs $f(\text{Dec}(\text{ske.sk}, \text{ct}))$ and outputs $\mathbf{0}$ if $\text{Dec}(\text{ske.sk}, \text{ct}) = \perp$. Next, it generates $(\text{cprf.pp}, \text{cprf.msk}) \leftarrow \text{CPRF.Setup}(1^\lambda, 1^\ell, 1^{\ell'}, 1^L, 1^d)$, $\text{pe.msk} \leftarrow \text{PE.Setup}(1^\lambda)$. It sets cprf.pp as pp , and $(\text{cprf.msk}, \text{ske.sk}, \text{pe.msk})$ as master secret key msk . It sends pp to $\mathcal{A}$.
3. Attacker $\mathcal{A}$ is allowed polynomial secret key and message queries.
 - In each secret key query, $\mathcal{A}$ outputs $\mathbf{y}_0, \mathbf{y}_1$. If $\mathbf{y}_0 \in \mathcal{Y}_0$, or $\mathbf{y}_1 \in \mathcal{Y}_1$, or there exists $\mathbf{x} \in \mathcal{X}$ such that $f(\mathbf{x}) = \mathbf{y}_0$ or $f(\mathbf{x}) = \mathbf{y}_1$, the attacker fails and the game aborts. Otherwise, the challenger generates $\text{cprf.sk}_{\mathbf{y}_b} \leftarrow \text{Constrain}(\text{cprf.msk}, C_{\text{CPRF}.f, \mathbf{y}_b})$. Next, let $\text{PE}.f$ be the function that takes as input $\text{ske.ct}, \tau$, and has the above $\text{cprf.sk}_{\mathbf{y}_b}$ hardwired. It outputs 1 if and only if $\text{CPRF.Eval}(\text{cprf.sk}_{\mathbf{y}_b}, \text{ske.ct}) = \tau$. The challenger outputs $\text{sk} \leftarrow \text{PE.KeyGen}(\text{pe.msk}, \text{PE}.f)$. It adds $\mathbf{y}_0$ to $\mathcal{Y}_0$ and $\mathbf{y}_1$ to $\mathcal{Y}_1$.
 - In each message query, $\mathcal{A}$ outputs message $\mathbf{m}$ and attribute $\mathbf{x}$. If $f(\mathbf{x}) \in \mathcal{Y}_0$ or $f(\mathbf{x}) \in \mathcal{Y}_1$, $\mathcal{A}$ fails and the game aborts. Otherwise, the challenger then generates $\text{ske.ct} \leftarrow \text{SKE.Enc}(\text{ske.sk}, \mathbf{x})$, and $\tau := \text{CPRF.Eval}(\text{cprf.msk}, \text{ske.ct})$. It outputs $\text{ct} \leftarrow \text{PE.Enc}(\text{pe.msk}, (\text{ske.ct}, \tau), \mathbf{m})$. It adds $\mathbf{x}$ to $\mathcal{X}$.
4. The attacker outputs b' . It wins if and only if $b' = b$.

Hybrid 1: It is the same as Hybrid 0, except that during the message queries and computes the pseudorandom function output using the constrained keys.

3. Attacker $\mathcal{A}$ is allowed polynomial secret key and message queries.
 - This is the same as Hybrid 0.

- In each message query, $\mathcal{A}$ outputs message $\mathbf{m}$ and attribute $\mathbf{x}$. If $f(\mathbf{x}) \in \mathcal{Y}_0$ or $f(\mathbf{x}) \in \mathcal{Y}_1$, $\mathcal{A}$ fails and the game aborts. Let $\mathbf{y}$ be $f(\mathbf{x})$. If $\text{cprf.sk}_{\mathbf{y}}$ does not exist, the challenger generates $\text{cprf.sk}_{\mathbf{y}} \leftarrow \text{Constrain}(\text{cprf.msk}, C_{\text{CPRF},f,\mathbf{y}})$. The challenger then generates $\text{ske.ct} \leftarrow \text{SKE.Enc}(\text{ske.sk}, \mathbf{x})$, and $\tau := \text{CPRF.CEval}(\text{cprf.sk}_{\mathbf{y}}, \text{ske.ct})$. It outputs $\text{ct} \leftarrow \text{PE.Enc}(\text{pe.msk}, (\text{ske.ct}, \tau), \mathbf{m})$. It adds $\mathbf{x}$ to $\mathcal{X}$.

Hybrid 2: It is identical to Hybrid 1, except that $\text{CPRF}.f$ is set as a function that always outputs $\mathbf{0}$. And the constrained keys are generated under random function outputs $\mathbf{y}'$.

2. Challenger tosses coin $b \leftarrow \{0, 1\}$, generates $\text{ske.sk} \leftarrow \text{SKE.Setup}(1^\lambda)$. Let $\text{CPRF}.f$ be the function that has input length ℓ and output length ℓ' : It takes as input in $\{0, 1\}^\ell$ and always outputs $\mathbf{0}$. Next, it generates $(\text{cprf.pp}, \text{cprf.msk}) \leftarrow \text{CPRF.Setup}(1^\lambda, 1^\ell, 1^{\ell'}, 1^L, 1^d)$, $\text{pe.msk} \leftarrow \text{PE.Setup}(1^\lambda)$. It sets cprf.pp as pp , and $(\text{cprf.msk}, \text{ske.sk}, \text{pe.msk})$ as master secret key msk . It sends pp to $\mathcal{A}$.
3. Attacker $\mathcal{A}$ is allowed polynomial secret key and message queries.
 - In each secret key query, $\mathcal{A}$ outputs $\mathbf{y}_0, \mathbf{y}_1$. If $\mathbf{y}_0 \in \mathcal{Y}_0$, or $\mathbf{y}_1 \in \mathcal{Y}_1$, or there exists $\mathbf{x} \in \mathcal{X}$ such that $f(\mathbf{x}) = \mathbf{y}_0$ or $f(\mathbf{x}) = \mathbf{y}_1$, the attacker fails and the game aborts. Otherwise, the challenger samples $\mathbf{y}' \leftarrow \{0, 1\}^{\ell'}$ and generates $\text{cprf.sk}_{\mathbf{y}_b} \leftarrow \text{Constrain}(\text{cprf.msk}, C_{\text{CPRF},f,\mathbf{y}'})$. Next, let $\text{PE}.f$ be the function that takes as input $\text{ske.ct}, \tau$, and has the above $\text{cprf.sk}_{\mathbf{y}_b}$ hardwired. It outputs 1 if and only if $\text{CPRF.Eval}(\text{cprf.sk}_{\mathbf{y}_b}, \text{ske.ct}) = \tau$. The challenger outputs $\text{sk} \leftarrow \text{PE.KeyGen}(\text{pe.msk}, \text{PE}.f)$. It adds $\mathbf{y}_0$ to $\mathcal{Y}_0$, $\mathbf{y}_1$ to $\mathcal{Y}_1$.
 - In each message query, $\mathcal{A}$ outputs message $\mathbf{m}$ and attribute $\mathbf{x}$. Let $\mathbf{y}$ be $f(\mathbf{x})$. If $\text{cprf.sk}_{\mathbf{y}}$ does not exist, the challenger samples $\mathbf{y}' \leftarrow \{0, 1\}^{\ell'}$ and generates $\text{cprf.sk}_{\mathbf{y}} \leftarrow \text{Constrain}(\text{cprf.msk}, C_{\text{CPRF},f,\mathbf{y}'})$. The challenger then generates $\text{ske.ct} \leftarrow \text{SKE.Enc}(\text{ske.sk}, \mathbf{x})$, and $\tau := \text{CPRF.CEval}(\text{cprf.sk}_{\mathbf{y}}, \text{ske.ct})$. It outputs $\text{ct} \leftarrow \text{PE.Enc}(\text{pe.msk}, (\text{ske.ct}, \tau), \mathbf{m})$. It adds $\mathbf{x}$ to $\mathcal{X}$.

Lemma 9.11. Assuming that CPRF satisfies completeness, then for all polynomial time adversary $\mathcal{A}$, there exists a negligible function $\text{negl}(\cdot)$ such that $\text{Adv}_{\mathcal{A}}^0 - \text{Adv}_{\mathcal{A}}^1 \leq \text{negl}(\lambda)$.

Proof. The proof is omitted. $\square$

Lemma 9.12. Assuming that CPRF satisfies constraint-hiding, then for all polynomial time adversary $\mathcal{A}$, there exists a negligible function $\text{negl}(\cdot)$ such that $\text{Adv}_{\mathcal{A}}^1 - \text{Adv}_{\mathcal{A}}^2 \leq \text{negl}(\lambda)$.

Proof. Omitted as it is nearly identical to Theorem 9.6. $\square$

We observe that the attacker has only 1/2 advantage in Hybrid 2. By the above lemmas, our theorem holds. $\square$

10 CPRFs in the Static Constraint Model

As we explained in the introduction, designing CPRFs is a far easier task if the Setup algorithm gets the set of constraints as an input to begin with. We refer this as the ‘static constraint model’. In particular, the CPRF Setup algorithm now takes, as an additional input, a sequence of constraints $C_1, \dots, C_n$. In this section, we provide a simple generic construction for CPRFs in the static model

from standard PRFs. Our design supports any $n = O(\log \lambda)$ constraints, where each constraint could be represented as any polynomial-sized circuits. The construction is as follows:

Setup($1^\lambda, C_1, \dots, C_n$): It takes as input security parameter λ and constraining circuits $C_1, \dots, C_n$. For all $\mathbf{y} \in \{0, 1\}^n$, it sets $K_{\mathbf{y}} \leftarrow \text{PRF.Setup}(1^\lambda)$. It sets $\text{msk} := (\{K_{\mathbf{y}}\}_{\mathbf{y} \in \{0, 1\}^n}, \{C_i\}_{i \in [n]})$.

Eval($\text{msk}, \mathbf{x}$): It takes as input master secret key msk and input $\mathbf{x}$. Let $\text{msk} = (\{K_{\mathbf{y}}\}_{\mathbf{y} \in \{0, 1\}^n}, \{C_i\}_{i \in [n]})$. Set $\mathbf{y} \in \{0, 1\}^n$ as the vector such that $\mathbf{y}[i] = 1$ if and only if $C_i[\mathbf{x}] = 1$. It outputs $\text{PRF.Eval}(K_{\mathbf{y}}, \mathbf{x})$.

Constrain(msk, C): It takes as input msk and circuit C . It again parses msk as $(\{K_{\mathbf{y}}\}_{\mathbf{y} \in \{0, 1\}^n}, C_1, \dots, C_n)$. Let i be the index such that $C_i = C$, it outputs $\text{sk} := (\{K_{\mathbf{y}}\}_{\mathbf{y}[i]=1}, \{C_i\}_{i \in [n]})$.

CEval($\text{sk}, \mathbf{x}$): It parses sk as $(\{K_{\mathbf{y}}\}_{\mathbf{y}[i]=1}, \{C_i\}_{i \in [n]})$. Let $\mathbf{y} \in \{0, 1\}^n$ be the vector where $\mathbf{y}[i] = 1$ if and only if $C_i(\mathbf{x}) = 1$. It outputs $\text{PRF.Eval}(K_{\mathbf{y}}, \mathbf{x})$.

Correctness. The correctness property directly follows from the correctness of the underlying PRF.

Security. The security of this naive constrained PRF follows directly from the security of the underlying PRF. Below we provide a short sketch.

Proof sketch. Assume that there exists a PPT adversary $\mathcal{A}$ that has non-negligible advantage, we design a reduction algorithm $\mathcal{B}$ which breaks the underlying PRF security game. During the **Setup** step, $\mathcal{A}$ outputs $C_1, \dots, C_n$. $\mathcal{B}$ then samples $\mathbf{y}^* \leftarrow \{0, 1\}^n$. It generates $K_{\mathbf{y}} \leftarrow \text{PRF.Setup}(1^\lambda)$ for all $\mathbf{y} \in \{0, 1\}^n$ such that $\mathbf{y} \neq \mathbf{y}^*$. During each secret key query, $\mathcal{A}$ outputs C_i . If $\mathbf{y}^*[i] = 1$, $\mathcal{B}$ aborts and outputs a random guess. Otherwise, it outputs $\text{sk} := (\{K_{\mathbf{y}}\}_{\mathbf{y}[i]=1}, \{C_i\}_{i \in [n]})$. During each evaluation query, $\mathcal{A}$ outputs $\mathbf{x}$. $\mathcal{B}$ sets $\mathbf{y} \in \{0, 1\}^n$ as the vector such that $\mathbf{y}[i] = 1$ if and only if $C_i(\mathbf{x}) = 1$. If $\mathbf{y}^* = \mathbf{y}$, $\mathcal{B}$ queries the PRF challenger with $\mathbf{x}$ and forwards the PRF challenger's response to $\mathcal{A}$. Otherwise, $\mathcal{B}$ outputs $\text{PRF.Eval}(K_{\mathbf{y}}, \mathbf{x})$. During the challenge query, $\mathcal{A}$ outputs $\mathbf{x}^*$, $\mathcal{B}$ computes $\mathbf{y}$ where $\mathbf{y}[i] = 1$ if and only if $C_i(\mathbf{x}^*) = 1$. If $\mathbf{y} \neq \mathbf{y}^*$, $\mathcal{B}$ aborts and outputs a random guess. Otherwise, $\mathcal{B}$ sends $\mathbf{x}^*$ to the PRF challenger. It then forwards the challenger's output to $\mathcal{A}$. Additionally, if $\mathcal{A}$ queries $\mathbf{x}^*$ during any evaluation query or if $\mathcal{A}$ queries any C_i during secret key queries such that $C(\mathbf{x}) = 1$, $\mathcal{A}$ fails and $\mathcal{B}$ makes a random guess. Note that in the view of $\mathcal{A}$, $\mathcal{B}$ simulates the constrained PRF security game with non-negligible advantage. If $\mathcal{A}$ has non-negligible advantage against the CPRF security, $\mathcal{B}$ breaks the pseudorandomness of the underlying PRF with non-negligible advantage.

11 Bootstrapping purified circuits to general (impure) circuits

In this section, we discuss generic approaches to bootstrap constrained PRFs for purified circuit constraints to more general constraint classes. We start by introducing a new notion of *weakly impure circuit classes*.

Definition 11.1 (weakly impure circuit class). Let $\{\mathcal{C}_\ell\}_{\ell \in \mathbb{N}}$ be an ensemble of circuit classes, where $\mathcal{C}_\ell$ is a set of circuits that take ℓ -bits of input. Consider some subset $\mathcal{C} \subseteq \{\mathcal{C}_\ell\}_{\ell \in \mathbb{N}}$. We say $\mathcal{C}$ is a

“*weakly-impure*” *constrain class* if there exists a purified circuit class $\mathcal{C}'$ (see Eq. (2)) such that:

$$\forall C \in \mathcal{C}, \exists k \in \text{poly}(\ell), C_1, \dots, C_k \in \mathcal{C}' \text{ such that } \forall x \in \{0, 1\}^\ell : C(x) = \sum_{i=1}^k C_i(x)$$

In words, we say an ensemble of circuit class is a weakly impure class if every circuit in that class can be decomposed into a polynomial number of purified circuits. Moreover, we assume that the corresponding purified circuit class as well as the per-circuit decomposition can be found in polynomial time.

Bootstrapping to weakly impure circuits. Next, we show that constrained PRFs for purified predicates imply the existence of constrained PRFs for weakly impure predicates. Our construction leverages the constrained PRFs for the purified circuit class (P.Setup, P.Eval, P.Constrain, P.CEval).

Setup($1^\lambda, 1^\ell$): It takes as input security parameter λ and input length ℓ . It generates $\text{msk} \leftarrow \text{P.Setup}(1^\lambda, 1^\ell)$.

Eval(msk, x): It takes as input master secret key msk and input x . It outputs $\text{Eval}(\text{msk}, x)$.

Constrain(msk, C): It takes as input msk and a circuit C . The constrain algorithm tries to decompose C into a polynomial number of purified circuits. If it is successful, then let circuits $C_1, \dots, C_k \in \mathcal{C}'$ be such that $\forall x : C(x) = \sum_{i=1}^k C_i(x)$, where $\mathcal{C}'$ is a purified circuit class. It generates $\text{sk}_i \leftarrow \text{P.Constrain}(\text{msk}, C_i)$ for all $i \in [k]$. It outputs secret key sk as $(\text{sk}_1, \dots, \text{sk}_k)$. If it is unsuccessful (i.e., C cannot be decomposed into a polynomial number of purified circuits), then it simply output $\text{sk} \leftarrow \text{P.Constrain}(\text{msk}, C)$. (We also require that the constrained key additionally contains the full/decomposed circuit, but do not write it explicitly for clarity.)

CEval(sk, x): Parse sk as $(\text{sk}_1, \dots, \text{sk}_k)$ for some $k \geq 1$. (Recall we consider each constrained key also contains the associated circuit C_i .) The constrained evaluator finds the first index i such that $C_i(x) = 1$, then outputs $\text{P.CEval}(\text{sk}_i, x)$.

Note that by correctness of the underlying CPRF as well as the decomposition property as described in Theorem 11.1, the correctness of above scheme follows in a straightforward way. Next, we provide a short proof sketch arguing it satisfies constrained pseudorandomness as long as the attacker only makes constrained key queries that correspond to a *weakly impure constraint class*.

Theorem 11.2. Assuming that the underlying constrained PRF satisfies purified constrained pseudorandomness (Theorem 5.1), then the above constrained PRF satisfies constrained pseudorandomness for weakly impure constraint classes.

Proof. Suppose there exists a PPT adversary $\mathcal{A}$ that breaks the weakly impure constrained pseudorandomness experiment of the above construction, then one could design a reduction algorithm $\mathcal{B}$ that breaks the purified pseudorandomness of the underlying constrained PRF.

The purified challenger starts by generating $\text{msk} \leftarrow \text{P.Setup}(1^\lambda, 1^\ell)$, and samples a coin $b \leftarrow \{0, 1\}$. Next, $\mathcal{A}$ makes constrained key and evaluation queries, and a single challenge query. During each constrained key query, $\mathcal{A}$ outputs circuit C . $\mathcal{B}$ then generates purified circuits $C_1, \dots, C_k$ accordingly, such that $C(x) = \sum_{i=1}^k C_i(x)$ for any x . $\mathcal{B}$ queries the challenger with all these circuits

and the challenger outputs $sk_1, \dots, sk_k$. $\mathcal{B}$ sets $msk := (sk_1, \dots, sk_k)$ and sends msk to $\mathcal{A}$. For each evaluation query where $\mathcal{A}$ outputs x , $\mathcal{B}$ forwards x to the challenger. The challenger responds such query using r , and $\mathcal{B}$ sends r to $\mathcal{A}$. Next, during the challenge query, $\mathcal{A}$ outputs x^* . $\mathcal{B}$ again sends x^* to the challenger and the challenger outputs r^* . $\mathcal{B}$ then sends r^* to $\mathcal{A}$. At the end of the game, $\mathcal{A}$ makes a guess b' . $\mathcal{B}$ will output the same b' as its guess.

By definition of purified and weakly impure class, if x^* does not lie in the support of any circuit of $\mathcal{A}$'s constrained key queries, then x^* is also an admissible challenge query for reduction algorithm $\mathcal{B}$. If $\text{Adv}_{\mathcal{A}} - 1/2 \geq 1/2 + \epsilon(\lambda)$ for some non-negligible $\epsilon(\cdot)$, $\mathcal{B}$ also breaks the pseudorandomness of the underlying purified constrained PRF. $\square$

11.1 Further discussion: beyond CPRFs, constraint hiding, and more

The above principle of bootstrapping cryptographic primitives to beyond purified classes is extremely general, thus could be applied in the context of predicate encryption, functional encryption, and more. However, such generic compilers do not carry forward the predicate privacy property (such as constraint privacy, function hiding, etc). The reason is that during the above compilation process we must give out the constraint circuit in the clear (for ensuring correctness). Thus, even when the underlying CPRFs satisfy constraint privacy, our resulting CPRFs for weakly impure circuits do not satisfy constraint privacy. It is an interesting question whether we can either generically, or even in a non-black-box way, get around this limitation. This is an interesting question for future research.

Additionally, our current bootstrapping approach clearly relies on decomposing a given impure circuit into a sequence of pure circuits, and then doing an independent key generation for each purified circuit. We believe one needs to take advantage of the fact that the key generation algorithm gets complete access to circuit and not just each purified sub-component to ensure we could bootstrap to more general circuits.

Finally, it is an interesting question whether our approach of starting from purified circuits can be further used in other areas of cryptography to derive more meaningful results.

12 Candidate CPRFs for Purified NC¹

12.1 Preliminaries

LWE with Short Secrets. In this work, we will also be using a variant of the LWE problem called *LWE with Short Secrets*. In this variant, introduced by Applebaum et al. [ACPS09], the secret vector is also chosen from the noise distribution χ . They showed that this variant is as hard as LWE for sufficiently large number of samples m .

Assumption 12.1 (LWE with Short Secrets). Let n, m and q be positive integers and χ a noise distribution on $\mathbb{Z}$. The LWE with Short Secrets assumption $\text{LWESS}_{n,m,q,\chi}$, parameterized by n, m, q, χ , states that the following distributions are computationally indistinguishable⁶:

$$\left\{ (\mathbf{A}, \mathbf{S} \cdot \mathbf{A} + \mathbf{E}) : \begin{array}{l} \mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}, \\ \mathbf{S} \leftarrow \chi^{n \times n}, \mathbf{E} \leftarrow \chi^{n \times m} \end{array} \right\} \approx_c \left\{ (\mathbf{A}, \mathbf{U}) : \begin{array}{l} \mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}, \\ \mathbf{U} \leftarrow \mathbb{Z}_q^{n \times m} \end{array} \right\}.$$

⁶Applebaum et al. showed that $\{(\mathbf{A}, \mathbf{s}^\top \cdot \mathbf{A} + \mathbf{e}) : \mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}, \mathbf{s} \leftarrow \chi^n, \mathbf{e} \leftarrow \chi^m\} \approx_c \{(\mathbf{A}, \mathbf{u}) : \mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}, \mathbf{u} \leftarrow \mathbb{Z}_q^m\}$, assuming LWE is hard. However, by a simple hybrid argument, we can replace vectors $\mathbf{s}, \mathbf{e}, \mathbf{u}$ with matrices $\mathbf{S}, \mathbf{E}, \mathbf{U}$ of appropriate dimensions.

Lattices with Trapdoors. Lattices with trapdoors are lattices that are statistically indistinguishable from randomly chosen lattices, but have certain ‘trapdoors’ that allow efficient solutions to hard lattice problems.

Definition 12.2. A trapdoor lattice sampler consists of algorithms TrapGen and SamplePre with the following syntax and properties:

- $\text{TrapGen}(1^n, 1^m, q) \rightarrow (\mathbf{A}, T_{\mathbf{A}})$: The lattice generation algorithm is a randomized algorithm that takes as input the matrix dimensions n, m , modulus q , and outputs a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ together with a trapdoor $T_{\mathbf{A}}$.
- $\text{SamplePre}(\mathbf{A}, \mathbf{u}, T_{\mathbf{A}}, \sigma) \rightarrow \mathbf{s}$: The presampling algorithm takes as input a matrix $\mathbf{A}$, trapdoor $T_{\mathbf{A}}$, a vector $\mathbf{u} \in \mathbb{Z}_q^n$ and a parameter $\sigma \in \mathcal{R}$ (which determines the length of the output vectors). It outputs a vector $\mathbf{s} \in \mathbb{Z}_q^m$.

These algorithms must satisfy the following properties:

1. *Correct Presampling*: For all vectors $\mathbf{u}$, parameters σ , $(\mathbf{A}, T_{\mathbf{A}}) \leftarrow \text{TrapGen}(1^n, 1^m, q)$, and $\mathbf{S} \leftarrow \text{SamplePre}(\mathbf{A}, \mathbf{u}, T_{\mathbf{A}}, \sigma)$, $\mathbf{A} \cdot \mathbf{s} = \mathbf{u}$ and $\|\mathbf{s}\|_{\infty} \leq \sqrt{m} \cdot \sigma$.
2. *Well Distributedness of Matrix*: The following distributions are statistically indistinguishable:

$$\{\mathbf{A} : (\mathbf{A}, T_{\mathbf{A}}) \leftarrow \text{TrapGen}(1^n, 1^m, q)\} \approx_s \{\mathbf{A} : \mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}\}.$$

3. *Well Distributedness of Preimage*: For all $(\mathbf{A}, T_{\mathbf{A}}) \leftarrow \text{TrapGen}(1^n, 1^m, q)$, if $\sigma = \omega(\sqrt{n \cdot \log q \cdot \log m})$, then the following distributions are statistically indistinguishable:

$$\{\mathbf{S} : \mathbf{U} \leftarrow \mathbb{Z}_q^n, \mathbf{S} \leftarrow \text{SamplePre}(\mathbf{A}, T_{\mathbf{A}}, \mathbf{U}, \sigma)\} \approx_s \mathcal{D}_{\mathbb{Z}^m, \sigma}.$$

These properties are satisfied by the gadget-based trapdoor lattice sampler of [MP12] for parameters m such that $m = \Omega(n \cdot \log q)$.

Lemma 12.3 ([CVW18b, GKW18]). Let $n, m, k, q \in \mathbb{N}$ and $\sigma \in \mathbb{R}$ such that $n, m, k \in \text{poly}(\lambda)$, $m = \Omega(n \cdot \log q)$, and $\sigma = \omega(\sqrt{n \cdot \log q \cdot \log m})$. For an arbitrary matrix $\mathbf{Z} \in \mathbb{Z}_q^{n \times k}$, the following two distributions are computationally indistinguishable, assuming $\text{LWE}_{m, k, q, \sigma}$:

$$\left\{ \begin{array}{l} (\mathbf{A}, \tau) \leftarrow \text{TrapGen}(1^n, 1^m, q), \\ \mathbf{D} : \mathbf{E} \leftarrow D_{\mathbb{Z}, \sigma}^{n \times k}, \\ \mathbf{D} \leftarrow \text{SamplePre}(\mathbf{A}, \mathbf{Z} + \mathbf{E}, \tau, \sigma_2) \text{ using } \tau \end{array} \right\} \approx_c \left\{ \mathbf{D} : \mathbf{D} \leftarrow D_{\mathbb{Z}, \sigma}^{m \times k} \right\}.$$

This property was introduced as the computational lemma (Lemma 4.4) in [CVW18b] and was also implied by the target switching property (Definition 7.2) of [GKW18].

12.1.1 Branching Programs

Branching programs are a model of computation used to capture space-bounded computations [BDFP83, Bar86].

Definition 12.4 (Branching Program). A branching program of length L , width w and input space $\{0,1\}^n$ consists of a sequence of $2L$ functions $\sigma_{i,b} : [w] \rightarrow [w]$ for $1 \leq i \leq L, b \in \{0,1\}$, an input selection function $\iota : [L] \rightarrow [n]$, an accepting state $\text{acc} \in [w]$ and a rejection state $\text{rej} \in [w]$. The starting state st_0 is set to be 1 without loss of generality. The branching program evaluation on input $\mathbf{x} \in \{0,1\}^n$ proceeds as follows:

- For $i = 1$ to L ,
 - Let $b = \mathbf{x}[\iota(i)]$. Compute $\text{st}_i = \sigma_{i,b}(\text{st}_{i-1})$.
- If $\text{st}_L = \text{acc}$, output 1. If $\text{st}_L = \text{rej}$, output 0, else output $\perp$.

A permutation branching program follows the above definition, except that it additionally requires $\sigma_{i,b}$ to be a permutation function. In a remarkable result, Barrington [Bar86] showed that any circuit of depth d can be simulated by a permutation branching program of width 5 and length 4^d .

Definition 12.5 (Matrix Branching Programs). A matrix branching program of length L , width w , and input space $\{0,1\}^L$ consists of an input selection function $\iota : [L] \rightarrow [n]$, a row vector $\mathbf{B}_0 \in \{0,1\}^w$, a sequence of transition matrices $\mathbf{B}_{i,b} \in \{0,1\}^{w \times w}$: $\text{BP} = (\mathbf{B}_0, \{\mathbf{B}_{i,b}\}_{i \in [L], b \in \{0,1\}})$. The program computes a function $f_{\text{BP}} : \{0,1\}^L \rightarrow \{0,1\}^w$, where

$$f_{\text{BP}}(\mathbf{x}) = \mathbf{B}_0 \cdot \prod_{i=1}^L \mathbf{B}_{i,\mathbf{x}[\iota(i)]}.$$

Next, we explain how to use the above matrix branching program to represent any branching program and t -CNF formulas.

Matrix representation of any branching program A branching program can be represented by matrices $(\mathbf{B}_0, \{\mathbf{B}_{i,b}\}_{i \in [L], b \in \{0,1\}})$. It assigns a starting matrix $\mathbf{B}_0 = [1, 0, \dots, 0] \in \{0,1\}^{1 \times w}$ representing the initial state. An intermediate state matrix $\mathbf{B} \in \{0,1\}^{1 \times w}$ represents state ψ by having a 1 in the ψ -th entry and 0 in all other entries. At the i -th step, the transition on input bit b updates the intermediate state from $\mathbf{B}$ to $\mathbf{B}'$, computed as $\mathbf{B}' := \mathbf{B} \cdot \mathbf{B}_{i,b}$.

Matrix branching programs for t -CNF Previous works [GLW14, CVW18b, VWW22] described a read-once matrix BP representations of CNFs, which will be used in our constructions. Given a CNF formula of w clauses and L literals, one could obtain a branching program where the starting matrix $\mathbf{B}_0 = [1, \dots, 1] \in \{0,1\}^{1 \times w}$. Let $\mathbf{B}_{i,b} \in \{0,1\}^{w \times w}$ be a diagonal matrix where the bit at the j -th row and j' -th column is always zero if $j \neq j'$. Each diagonal entry of the transition matrix $\mathbf{B}_{i,b}$ is determined by whether the i -th literal satisfies the j -th clause, such that for all $j \in [w]$:

$$\mathbf{B}_{i,b}[j][j] = \begin{cases} 0, & \text{if the } i\text{-th literal satisfies the } j\text{-th clause when set to bit } b, \\ 1, & \text{otherwise.} \end{cases}$$

As a result, $\mathbf{B}_0 \cdot \prod_{i=1}^L \mathbf{B}_{i,\mathbf{x}[i]} = \mathbf{0}$ if and only if $\mathbf{x}$ satisfies the CNF formula.

t -CNF for input validity We now describe how to use a t -CNF formula to verify the input validity of a read-multiple-times branching program. Given an input selection function $\iota : [L] \rightarrow [n]$, the formula takes $\mathbf{x}' \in \{0, 1\}^L$ as input and checks that $\mathbf{x}'[i] = \mathbf{x}'[j]$ whenever $\iota[i] = \iota[j]$. The t -CNF formula consists of L literals, and equality between $\mathbf{x}'[i]$ and $\mathbf{x}'[j]$ is enforced by two clauses:

$$(\neg \mathbf{x}'[i] \vee \mathbf{x}'[j]) \wedge (\mathbf{x}'[i] \vee \neg \mathbf{x}'[j]).$$

To ensure that $\mathbf{x}'$ is consistent with ι , at most $2(L - 1)$ clauses are needed. As a result, the corresponding matrix branching program has width $w \leq 2(L - 1)$ and length L .

12.2 Candidate Construction

We rely on the LWE assumption with sub-exponential hardness, where for some $\epsilon > 0$, indistinguishability holds against adversaries running in time 2^{n^ϵ} with advantage at most 2^{-n^ϵ} , as long as $q \leq 2^{n^\epsilon}$. Let χ_0, χ_1 , and χ_2 be truncated discrete gaussian distributions set as $\tilde{D}_{\mathbb{Z}, \sigma_0}, \tilde{D}_{\mathbb{Z}, \sigma_1}$, and $\tilde{D}_{\mathbb{Z}, \sigma_2}$ respectively. Fix any $\epsilon < 1/2$. Let $n, m, q, p, \sigma_0, \sigma_1, \sigma_2, L, Q$, where Q denotes the total number of constrained key queries, under the following constraints:

- $n = \text{poly}(\lambda)$, $\max\{q, 2^{(LQ)^{2\lambda}}\} \leq 2^{n^\epsilon}$, $m = \Omega(n \log q)$ (for LWE security)
- $\sigma_1 \geq (\sigma_0 w n m)^{L+1} \cdot 2^{\Omega(L+\lambda)}$ (for smudging)
- $\frac{q}{p} \geq (L + 1) \cdot (\sigma_0 \sigma_2 w n \cdot m^2)^{L+1} \cdot 2^{\Omega(L+\lambda)}$ (for correctness/smudging)

One possible parameter setting that satisfies the above requirement is as follows:

$$n = (LQ)^3 \lambda^2 / \epsilon, m = n \log q, p = 2, q = 2^{(LQ)^{2\lambda}}, \sigma_0 = n, \sigma_1 = 2^{L+\lambda}.$$

Setup($1^\lambda, 1^L$): It takes as input security parameter λ and branching program size L . It samples

$$\mathbf{A}_{L+1,1} \leftarrow \mathbb{Z}_q^{n \times m}, \mathbf{S}_0 \leftarrow \chi_0^{n \times n}, \{\mathbf{S}_{i,b} \leftarrow \chi_0^{n \times n}\}_{i \in [L], b \in \{0,1\}}, \text{ and outputs } \text{msk} := (\mathbf{S}_0, \{\mathbf{S}_{i,b}\}_{i \in [L], b \in \{0,1\}}, \mathbf{A}_{L+1,1}).$$

Eval($\text{msk}, \mathbf{x}$): It takes as input master secret key msk and input $\mathbf{x}$. It parses msk as $(\mathbf{S}_0, \{\mathbf{S}_{i,b}\}_{i \in [L], b \in \{0,1\}}, \mathbf{A}_{L+1,1})$ and outputs

$$\left[\mathbf{S}_0 \cdot \left(\prod_{i=1}^L \mathbf{S}_{i, \mathbf{x}[\iota(i)]} \right) \cdot \mathbf{A}_{L+1,1} \right].$$

Constrain(msk, C): It takes as input msk and a NC^1 circuit C . Let circulating branching program

$\text{BP} = (\mathbf{B}_0, \{\mathbf{B}_{i,b}\}_{i \in [L], b \in \{0,1\}})$ be the branching program that represents such NC^1 circuit. It

parses msk as $(\mathbf{S}_0, \{\mathbf{S}_{i,b}\}_{i \in [L], b \in \{0,1\}}, \mathbf{A}_{L+1,1})$. Let $(\mathbf{B}_0, \{\mathbf{B}_i^b\}_{i \in [L], b \in \{0,1\}})$. Let $\text{BP}' = (\mathbf{B}'_0, \{\mathbf{B}'_{i,b}\}_{i \in [L], b' \in \{0,1\}})$

be the t -CNF branching program that verifies the input validity for BP. Next, it sets

$$\hat{\mathbf{B}}_0 := [\mathbf{B}_0 \mid \mathbf{B}'_0] \in \{0, 1\}^{1 \times w}, \hat{\mathbf{S}}_0 := \hat{\mathbf{B}}_0 \otimes \mathbf{S}_0, \text{ and for all } i \in [L], \hat{\mathbf{B}}_{i,b} := \begin{bmatrix} \mathbf{B}_{i,b} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}'_{i,b} \end{bmatrix} \in$$

$\{0, 1\}^{w \times w}$, $\hat{\mathbf{S}}_{i,b} := \hat{\mathbf{B}}_{i,b} \otimes \mathbf{S}_{i,b}$. Then for all $i \in [L]$, it generates lattice trapdoor $(\mathbf{A}_i, \tau_i) \leftarrow$

$\text{TrapGen}(1^{wn}, 1^m, q)$. It additionally samples $\mathbf{A}_{L+1,\psi} \leftarrow \mathbb{Z}_q^{n \times m}$ for all $\psi \in \{2, \dots, w\}$, and

sets $\mathbf{A}_{L+1}$ as $\begin{bmatrix} \mathbf{A}_{L+1,1} \\ \dots \\ \mathbf{A}_{L+1,w} \end{bmatrix}$. It then samples LWE error $\mathbf{E}_1 \leftarrow \chi_0^{n \times m}$ and $\mathbf{E}_{i,b} \leftarrow \chi_0^{wn \times m}$ for all

$i \in \{2, \dots, L+1\}, b \in \{0, 1\}$. Next, it sets $\mathbf{Y}_0 := \hat{\mathbf{S}}_0 \cdot \mathbf{A}_1 + \mathbf{E}_1$, and $\mathbf{Y}_{i,b} := \hat{\mathbf{S}}_{i,b} \cdot \mathbf{A}_{i+1} + \mathbf{E}_{i+1,b}$

for all $i \in [L], b \in \{0, 1\}$. It then generates preimages $\mathbf{D}_{i,b} \leftarrow \text{SamplePre}(\mathbf{A}_i, \mathbf{Y}_{i,b}, \tau_i, \sigma_2)$ for

all $i \in [L], b \in \{0, 1\}$. It outputs $\text{sk} := (\mathbf{Y}_0, \{\mathbf{D}_{i,b}\}_{i \in [L], b \in \{0,1\}})$.

$\text{CEval}(\text{sk}, \mathbf{x})$: It parses sk as $(\mathbf{Y}_0, \{\mathbf{D}_{i,b}\}_{i \in [L], b \in \{0,1\}})$. It outputs $\left[\mathbf{Y}_0 \cdot \prod_{i=1}^L \mathbf{D}_{i,\mathbf{x}[\iota(i)]} \right]$.

12.3 Correctness

We show that if $\text{BP}(\mathbf{x}) = 1$, then with high probability it holds that $\text{Eval}(\text{msk}, \mathbf{x}) = \text{CEval}(\text{sk}, \mathbf{x})$ for $\text{msk} \leftarrow \text{Setup}(1^\lambda, 1^L)$, $\text{sk} \leftarrow \text{Constrain}(\text{msk}, f, \mathbf{y})$. If $\text{BP}(\mathbf{x}) = 1$, then the output equation $\left[\mathbf{Y}_0 \cdot \prod_{i=1}^L \mathbf{D}_{i,\mathbf{x}[\iota(i)]} \right]$ of $\text{CEval}(\text{sk}, \mathbf{x})$ can be expanded as

$$\left[\hat{\mathbf{S}}_0 \cdot \prod_{i=1}^L \hat{\mathbf{S}}_{i,\mathbf{x}[\iota(i)]} \cdot \mathbf{A}_{L+1} + \mathbf{E}_1 \cdot \prod_{i=1}^L \mathbf{D}_{i,\mathbf{x}[\iota(i)]} + \sum_{j=1}^L \left(\hat{\mathbf{S}}_0 \prod_{i=1}^{j-1} \hat{\mathbf{S}}_{i,\mathbf{x}[\iota(i)]} \right) \cdot \mathbf{E}_{j+1,\mathbf{x}[\iota(j)]} \cdot \left(\prod_{i=j+1}^L \mathbf{D}_{i,\mathbf{x}[\iota(i)]} \right) \right].$$

Recall that we assume $\text{BP}(\mathbf{x}) = 1$ and the fact that $\{\mathbf{x}[\iota(i)]\}_{i \in [L]}$ is a valid input, thus the corresponding t -CNF branching program BP' will always output $\mathbf{0}$. It implies that

$$\hat{\mathbf{S}}_0 \cdot \prod_{i=1}^L \hat{\mathbf{S}}_{i,\mathbf{x}[\iota(i)]} \cdot \mathbf{A}_{L+1} = \mathbf{S}_0 \cdot \prod_{i=1}^L \mathbf{S}_{i,\mathbf{x}[\iota(i)]} \cdot \mathbf{A}_{L+1,1}.$$

As a result, the output of CEval differs from the equation of $\text{Eval}(\text{msk}, \mathbf{x})$ by the highlighted part, and we analyze the norm bound of this expression:

$$\begin{aligned} & \left\| \mathbf{E}_1 \cdot \prod_{i=1}^L \mathbf{D}_{i,\mathbf{x}[\iota(i)]} + \sum_{j=1}^L \left(\left(\hat{\mathbf{S}}_0 \prod_{i=1}^{j-1} \hat{\mathbf{S}}_{i,\mathbf{x}[\iota(i)]} \right) \cdot \mathbf{E}_{j+1,\mathbf{x}[\iota(j)]} \cdot \left(\prod_{i=j+1}^L \mathbf{D}_{i,\mathbf{x}[\iota(i)]} \right) \right) \right\|_\infty \\ & \leq \sigma_0 \cdot \sigma_2^L \cdot m^{2L} + \sum_{j=1}^L \left(\sigma_0^{j+1} \cdot (wnm)^j \cdot \sigma_2^{(L-j)} \cdot m^{2(L-j)} \right) \\ & \leq (L+1) \cdot (\sigma_0 \sigma_2 w n \cdot m^2)^{L+1} \ll q/p. \end{aligned}$$

Thus, the completeness of our design holds.

12.4 Purified Pseudorandomness and Constraint Hiding

In this section, we show that if private coin evasive LWE and standard LWE assumption holds, then our CPRF satisfies pseudorandomness security as in Theorem 5.1, except we additionally require each input (sent by $\mathcal{A}$ during evaluation query) is not a satisfying input for any constraint that $\mathcal{A}$ gets a constrained key for. The reason that we require this restriction is to avoid potential zeroizing attacks so that evasive LWE could be applied to our analysis.

Assumption 12.6 (Evasive LWE). Let Samp be a PPT algorithm that on input 1^λ , outputs $\mathbf{S} \in \mathbb{Z}_q^{n' \times n}$, $\mathbf{P} \in \mathbb{Z}_q^{n \times t}$, $\text{aux} \in \{0,1\}^*$. Consider the following advantage functions:

$$\text{Adv}_{\mathcal{A}_0}^{\text{PRE}}(\lambda) := \Pr[\mathcal{A}_0(\mathbf{S}\mathbf{B} + \mathbf{E}, \mathbf{S}\mathbf{P} + \mathbf{E}', \text{aux}) = 1] - \Pr[\mathcal{A}_0(\mathbf{C}, \mathbf{C}', \text{aux}) = 1], \quad (3)$$

$$\text{Adv}_{\mathcal{A}_1}^{\text{POST}}(\lambda) := \Pr[\mathcal{A}_1(\mathbf{S}\mathbf{B} + \mathbf{E}, \mathbf{D}, \text{aux}) = 1] - \Pr[\mathcal{A}_1(\mathbf{C}, \mathbf{D}, \text{aux}) = 1]. \quad (4)$$

where

$$(\mathbf{S}, \mathbf{P}, \text{aux}) \leftarrow \text{Samp}(1^\lambda), \mathbf{B} \leftarrow \mathbb{Z}_q^{n \times m}, \mathbf{E} \leftarrow D_{\mathbb{Z}, \sigma}^{n' \times m}, \mathbf{E}' \leftarrow D_{\mathbb{Z}, \sigma'}^{n' \times t}, \\ \mathbf{C} \leftarrow \mathbb{Z}_q^{n' \times m}, \mathbf{C}' \leftarrow \mathbb{Z}_q^{n' \times t}, \mathbf{D} \leftarrow \mathbf{B}^{-1}(\mathbf{P}, \sigma).$$

The evasive LWE assumption holds if for every PPT algorithm Samp , there exists some polynomial $\text{poly}(\cdot)$ such that for every PPT adversary $\mathcal{A}_1$, there exists another PPT adversary $\mathcal{A}_0$ and a negligible function $\text{negl}(\cdot)$ satisfying $\text{Adv}_{\mathcal{A}_0}^{\text{PRE}}(\lambda) \geq \frac{\text{Adv}_{\mathcal{A}_1}^{\text{POST}}(\lambda)}{\text{poly}(\lambda)} - \text{negl}(\lambda)$, and the running time $\text{Time}(\mathcal{A}_0) \leq \text{Time}(\mathcal{A}_1) \cdot \text{poly}(\lambda)$.

Definition 12.7 (Purified Constrained Pseudorandomness). A CPRF $\mathcal{F}$ satisfies (collusion-resistant) semi-adaptive constrained pseudorandomness for NC^1 circuits if the advantage of any PPT adversary in the experiment described in Theorem 5.1 is always negligible, where the adversary must also adhere to the restriction that each of $\mathcal{A}$'s constraining key query is always in the NC^1 circuit class and evaluation query $\mathbf{x}$ (and the challenge query $\mathbf{x}^*$) must not satisfy the NC^1 circuit for any constrained key query (i.e., $C_i(\mathbf{x}) = 0$ for all i).

Next, we prove the following theorem:

Theorem 12.8. Assuming that evasive LWE (Theorem 12.6) and LWESS $_{2^{k-k'}m, n, q, \chi_0}$ (Theorem 12.1) hold, then our construction satisfies pseudorandomness and constraint hiding for purified constraints.

12.4.1 Sequence of Hybrid Games

Hybrid 0 : This is identical to the original pseudorandomness experiment.

1. The challenger tosses coin $\beta \leftarrow \{0, 1\}$, samples $\mathbf{A}_{L+1,1} \leftarrow \mathbb{Z}_q^{n \times m}$, $\mathbf{S}_0 \leftarrow \chi_0^{n \times n}$, $\{\mathbf{S}_{i,b} \leftarrow \chi_0^{n \times n}\}_{i \in [L], b \in \{0,1\}}$.
2. The adversary $\mathcal{A}$ is allowed polynomial evaluation and constrained key queries:

- During the j -th query, if it is a constrained key query, $\mathcal{A}$ outputs a branching program $\text{BP} = (\mathbf{B}_0, \{\mathbf{B}_{j,i,b}\}_{i \in [L], b \in \{0,1\}})$. It adds BP to $\mathcal{B}$. If $\mathcal{B}$ is no longer a purified class, or if there exists $\mathbf{x} \in \mathcal{X}$ such that $\text{BP}(\mathbf{x}) = 1$, the attacker fails and the game aborts. Let $\text{BP}' = (\mathbf{B}'_0, \{\mathbf{B}'_{i,b}\}_{i \in [L], b \in \{0,1\}})$ be the t -CNF branching program that verifies input validity. The challenger sets $\hat{\mathbf{B}}_0 := [\mathbf{B}_0 \mid \mathbf{B}'_0] \in \{0, 1\}^{1 \times w}$, $\hat{\mathbf{S}}_0 := \hat{\mathbf{B}}_0 \otimes \mathbf{S}_0$, and for all $i \in [L]$, $\hat{\mathbf{B}}_{j,i,b} := \begin{bmatrix} \mathbf{B}_{j,i,b} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}'_{j,i,b} \end{bmatrix} \in \{0, 1\}^{w \times w}$, $\hat{\mathbf{S}}_{j,i,b} := \hat{\mathbf{B}}_{j,i,b} \otimes \mathbf{S}_{i,b}$. Next the challenger samples $\{(\mathbf{A}_{j,i}, \tau_{j,i}) \leftarrow \text{TrapGen}(1^{wn}, 1^m, q)\}_{i \in [L]}, \mathbf{E}_{j,i,b} \leftarrow \chi_0^{m \times m}$ and

$\{\mathbf{A}_{j,L+1,\psi} \leftarrow \mathbb{Z}_q^{n \times m}\}_{\psi \in \{2, \dots, \omega\}}$, and sets $\mathbf{A}_{j,L+1}$ as $\begin{bmatrix} \mathbf{A}_{L+1,1} \\ \mathbf{A}_{j,L+1,2} \\ \dots \\ \mathbf{A}_{j,L+1,w} \end{bmatrix}$. Next, it samples LWE

errors $\mathbf{E}_{j,1} \leftarrow \chi_0^{n \times m}$ and $\{\mathbf{E}_{j,i,b} \leftarrow \chi_0^{wn \times m}\}_{i \in [L], b \in \{0,1\}}$. It sets $\mathbf{Y}_{j,0} := \hat{\mathbf{S}}_0 \cdot \mathbf{A}_{j,1} + \mathbf{E}_{j,1}$, $\{\mathbf{Y}_{j,i,b} := \hat{\mathbf{S}}_{j,i,b} \cdot \mathbf{A}_{i+1} + \mathbf{E}_{j,i+1,b}, \mathbf{D}_{j,i,b} \leftarrow \mathbf{A}_{j,i}^{-1}(\mathbf{Y}_{j,i,b})\}_{i \in [L], b \in \{0,1\}}$. It outputs $\text{sk}_j := (\mathbf{Y}_{j,0}, \{\mathbf{D}_{j,i,b}\}_{i \in [L], b \in \{0,1\}})$.

- During each evaluation query, $\mathcal{A}$ outputs $\mathbf{x}$. If there exists any secret key query such that $\mathbf{BP} \in \mathcal{B}$ such that $\mathbf{BP}(\mathbf{x}) = 1$, the game aborts and the attacker fails. Otherwise, the challenger outputs $\left[\mathbf{S}_0 \cdot \left(\prod_{i=1}^L \mathbf{S}_{i,\mathbf{x}[\iota(i)]} \right) \cdot \mathbf{A}_{L+1,1} \right]$.
- 3. During the challenge query, $\mathcal{A}$ outputs $\mathbf{x}^*$. If there exists any secret key query such that $\mathbf{BP} \in \mathcal{B}$ such that $\mathbf{BP}(\mathbf{x}^*) = 1$, the game aborts and the attacker fails. If $\beta = 0$, the challenger outputs $\left[\mathbf{S}_0 \cdot \left(\prod_{i=1}^L \mathbf{S}_{i,\mathbf{x}[\iota(i)]} \right) \cdot \mathbf{A}_{L+1,1} \right]$. Otherwise, the challenger samples and outputs uniformly random $\mathbf{R} \leftarrow \mathbb{Z}_q^{n \times m}$.
- 4. It is the same step 4, except that $\mathbf{x}^*$ must not be queried during the evaluation queries and $\mathbf{BP}(\mathbf{x}^*) = 0$ for each C during the constrained key queries.
- 5. The challenger outputs β' and wins if and only if $\beta' = \beta$.

Hybrid 1 : This is the same as Hybrid 0, except that the evaluation and challenge queries are answered by constrained keys.

2. The adversary $\mathcal{A}$ is allowed polynomial evaluation and constrained key queries:
 - Constrained key queries remain the same as the previous hybrid.
 - During the j -th query where it is a evaluation query, $\mathcal{A}$ outputs $\mathbf{x}$. If there exists any secret key query such that $\mathbf{BP} \in \mathcal{B}$ such that $\mathbf{BP}(\mathbf{x}) = 1$, the game aborts and the attacker fails. Next, Let $\mathbf{BP} = (\mathbf{B}_0, \{\mathbf{B}_{j,i,b}\}_{i \in [L], b \in \{0,1\}})$ be the matrix branching program that accepts if and only if the input is equal to $\mathbf{x}$. We remark that that such branching program for equality testing can be constructed within constant width. Next, the challenger generates constrained key $(\mathbf{Y}_{j,0}, \{\mathbf{D}_{j,i,b}\}_{i \in [L], b \in \{0,1\}})$ according to $\mathbf{BP}$ following from the same procedure as the challenge query. It outputs $\left[\mathbf{Y}_{j,0} \cdot \prod_{i=1}^L \mathbf{D}_{j,i,\mathbf{x}[\iota(i)]} \right]$.
3. During the challenge query, $\mathcal{A}$ outputs $\mathbf{x}^*$. If there exists any secret key query such that $\mathbf{BP} \in \mathcal{B}$ where $\mathbf{BP}(\mathbf{x}^*) = 1$, the game aborts and the attacker fails. Otherwise if $\beta = 0$, the challenger generates the constrained key, the challenger generates $\mathbf{Y}_0, \{\mathbf{D}_{j,i,b}\}_{i \in [L], b \in \{0,1\}}$ identically as the above evaluation query (j is the index of the challenge query). It outputs $\left[\mathbf{Y}_{j,0} \cdot \prod_{i=1}^L \mathbf{D}_{j,i,\mathbf{x}^*[\iota(i)]} \right]$. Otherwise, the challenger samples and outputs uniformly random $\mathbf{R} \leftarrow \mathbb{Z}_q^{n \times m}$.

Hybrid 2: This is the same as Hybrid 1, except that $\mathbf{Y}_{j,0}$ are uniformly randomly sampled.

2. The constrained key and evaluation queries are the same as the previous hybrid, except that $\mathbf{Y}_{j,0}$ is sampled from $\mathbb{Z}_q^{n \times m}$ and $\mathbf{D}_{j,i,b} \leftarrow \chi_2^{m \times m}$ for all $j \in [Q], i \in [L], b \in \{0,1\}$ for both constrained key and evaluation queries.
3. It is the same as the challenge query of Hybrid 1, except that $\mathbf{Y}_{j,0} \leftarrow \mathbb{Z}_q^{n \times m}$ and $\mathbf{D}_{j,i,b} \leftarrow \chi_2^{m \times m}$ when $\beta = 1$.

Lemma 12.9. Assume that LWE and evasive LWE assumptions hold, then for any PPT adversary $\mathcal{A}$, there exists a negligible function $\text{negl}(\cdot)$ such that $\text{Adv}_{\mathcal{A}}^0 - \text{Adv}_{\mathcal{A}}^1 \leq \text{negl}(\lambda)$.

Proof. The lemma holds by the completeness of our design. $\square$

Lemma 12.10. Let $\mathbf{S}_{i,b} \leftarrow \chi_0^{n \times n}$ for $i \in [L], b \in \{0,1\}$, $\mathbf{E}_{j,\mathbf{x}} \leftarrow \chi_1^{n \times m}$ for $j \in [Q], \mathbf{x} \in \{0,1\}^k$. Assuming that $\text{LWESS}_{2^{k-k'}m,n,q,\chi_0}$ assumption holds, then for all $k \in [L]$, $Q = \text{poly}(\lambda)$, it satisfies that

$$\{\mathbf{S}_{\mathbf{x}} \cdot \mathbf{A}_{j,k+1} + \mathbf{E}_{j,\mathbf{x}}\}_{j \in [Q], \mathbf{x} \in \{0,1\}^k}, \mathbf{S}_0, \{\mathbf{S}_{i,b}\}_{i \in [L], b \in \{0,1\}} \approx_c \{U(\mathbb{Z}_q^{n \times m})\}_{j \in [Q], \mathbf{x} \in \{0,1\}^k}, \mathbf{S}_0, \{\mathbf{S}_{i,b}\}_{i \in [L], b \in \{0,1\}}$$

where in the above equation

$$\mathbf{S}_{\mathbf{x}} := \mathbf{S}_0 \cdot \prod_{i=1}^k \mathbf{S}_{i,\mathbf{x}[i]}, \{\mathbf{E}_{j,\mathbf{x}} \leftarrow \chi_1^{n \times m}, \mathbf{A}_{j,k+1} \leftarrow \mathbb{Z}_q^{n \times m}\}_{j \in [Q]}.$$

Proof. First, we prove the following intermediate steps:

Claim 12.11. Fix any $0 \leq k' \leq k$, and given $\mathbf{S}_0 \leftarrow \chi_0^{n \times n}$ and $\{\mathbf{S}_{i,b}\} \leftarrow \chi_0^{n \times n}$ for $i \in [L], b \in \{0,1\}$, $\mathbf{E}_{j,\mathbf{x}} \leftarrow \chi_1^{n \times 2^{k-k'}m}$ for $j \in [Q], \mathbf{x} \in \{0,1\}^{k'}$, $\mathbf{E}'_{j,b} \leftarrow \chi_0^{n \times 2^{k-k'}m}$ for $j \in [Q], b \in \{0,1\}$, and $\mathbf{A}_{j,k'+1} \leftarrow \mathbb{Z}_q^{n \times 2^{k-k'}m}$ for $j \in [Q]$, it holds that

$$\{\mathbf{S}_{\mathbf{x}} \cdot \mathbf{A}_{j,k'+1} + \mathbf{E}_{j,\mathbf{x}}\}_{j \in [Q], \mathbf{x} \in \{0,1\}^{k'}}, \mathbf{S}_0, \{\mathbf{S}_{i,b}\}_{i \in [L], b \in \{0,1\}} \quad (5)$$

$$\approx_s \{\mathbf{S}_{\mathbf{x}'} \cdot (\mathbf{S}_{k',b} \cdot \mathbf{A}_{j,k'+1} + \mathbf{E}'_{j,b}) + \mathbf{E}_{j,\mathbf{x}}\}_{j \in [Q], \mathbf{x}' \in \{0,1\}^{k'-1}, b \in \{0,1\}}, \mathbf{S}_0, \{\mathbf{S}_{i,b}\}_{i \in [L], b \in \{0,1\}}. \quad (6)$$

Proof. By truncated discrete gaussian, $\|\mathbf{E}'_{j,b}\|_{\infty} \leq \sigma_0 \cdot \sqrt{m}$. As a result, $\|\mathbf{S}_{\mathbf{x}'} \cdot \mathbf{E}'_{j,b}\|_{\infty} \leq \sigma_0 \cdot m \cdot (\sigma_0 \omega n m)^L$. By our parameter setting, $\sigma_1 \geq (\sigma_0 \omega n m)^{L+1} \cdot 2^{\Omega(L+\lambda)}$. Thus by Theorem 3.4, for $\mathbf{E}_{j,\mathbf{x}} \leftarrow \chi_1^{n \times 2^{k-k'}m}$, the distribution of $\mathbf{S}_{\mathbf{x}'} \cdot \mathbf{E}'_{j,b} + \mathbf{E}_{j,\mathbf{x}}$ and $\mathbf{E}_{j,\mathbf{x}}$ are statistically close. $\square$

Claim 12.12. Assume that $\text{LWESS}_{2^{k-k'}m,n,q,\chi_0}$ assumption holds, then given any $0 \leq k' \leq k$, $\mathbf{S}_0 \leftarrow \chi_0^{n \times n}$ and $\{\mathbf{S}_{i,b}\} \leftarrow \chi_0^{n \times n}$ for $i \in [L], b \in \{0,1\}$, $\mathbf{E}_{j,\mathbf{x}} \leftarrow \chi_1^{n \times 2^{k+1-k'}m}$, $\mathbf{E}'_{j,\mathbf{x}} \leftarrow \chi_0^{n \times 2^{k-k'}m}$ for $j \in [Q], \mathbf{x} \in \{0,1\}^{k'}$, and $\mathbf{A}_{j,k'+1} \leftarrow \mathbb{Z}_q^{n \times 2^{k-k'}m}$, $\mathbf{A}_{j,k'} \leftarrow \mathbb{Z}_q^{n \times 2^{k+1-k'}m}$ for $j \in [Q]$, it satisfies that

$$\{\mathbf{S}_{\mathbf{x}'} \cdot [\mathbf{S}_{k',0} \cdot \mathbf{A}_{j,k'+1} + \mathbf{E}'_{j,0} \mid \mathbf{S}_{k',1} \cdot \mathbf{A}_{j,k'+1} + \mathbf{E}'_{j,1}] + \mathbf{E}_{j,\mathbf{x}}\}_{j \in [Q], \mathbf{x}' \in \{0,1\}^{k'-1}}, \mathbf{S}_0, \{\mathbf{S}_{i,b}\}_{i \in [L], b \in \{0,1\}} \quad (7)$$

$$\approx_c \{\mathbf{S}_{\mathbf{x}'} \cdot \mathbf{A}_{j,k'} + \mathbf{E}_{j,\mathbf{x}}\}_{j \in [Q], \mathbf{x}' \in \{0,1\}^{k'-1}}, \mathbf{S}_0, \{\mathbf{S}_{i,b}\}_{i \in [L], b \in \{0,1\}}. \quad (8)$$

Proof. It directly follows from LWE with short secret. $\square$

We observe that Eq. (6) is identically distributed as Eq. (7) and just reshaped. Thus, by reshaping Eq. (5), it is computationally indistinguishable from Eq. (8). Furthermore, Eq. (8) is identically distributed as Eq. (5) when k' is decreased by 1. Thus by applying Theorem 12.11 and Theorem 12.12 from $k' = k$ to $k' = 0$, our lemma follows. $\square$

Lemma 12.13. Let $\{\hat{\mathbf{S}}_{i,b}\}_{i \in [L], b \in \{0,1\}}$ be generated following the sampling procedure in Hybrid 1. Assuming that $\text{LWESS}_{2^{k-k'}m,n,q,\chi_0}$ assumption holds, and all of the branching programs BP that the adversary queries are within a purified class, then for all $k \in [L]$, $Q = \text{poly}(\lambda)$ it satisfies that

$$\left\{ \hat{\mathbf{S}}_{\mathbf{x}} \cdot \mathbf{A}_{j,k+1} + \mathbf{E}_{j,\mathbf{x}} \right\}_{j \in [Q], \mathbf{x} \in \{0,1\}^k}, \hat{\mathbf{S}}_0, \left\{ \hat{\mathbf{S}}_{i,b} \right\}_{i \in [L], b \in \{0,1\}} \approx_c \left\{ U(\mathbb{Z}_q^{n \times m}) \right\}_{j \in [Q], \mathbf{x} \in \{0,1\}^k}, \hat{\mathbf{S}}_0, \left\{ \hat{\mathbf{S}}_{i,b} \right\}_{i \in [L], b \in \{0,1\}}$$

where in the above equation

$$\hat{\mathbf{S}}_{\mathbf{x}} := \hat{\mathbf{S}}_0 \cdot \prod_{i=1}^k \hat{\mathbf{S}}_{i,\mathbf{x}[i]}, \{\mathbf{E}_{j,\mathbf{x}} \leftarrow \chi_1^{n \times m}\}_{j \in [Q]},$$

and if $k \leq L-1$ then $\{\mathbf{A}_{j,k+1} \leftarrow \mathbb{Z}_q^{wn \times m}\}_{j \in [Q]}$, otherwise if $k = L$ then $\mathbf{A}_{k+1,1} \leftarrow \mathbb{Z}_q^{n \times m}$, $\{\mathbf{A}_{j,k+1,\psi} \leftarrow \mathbb{Z}_q^{n \times m}\}_{\psi \in \{2, \dots, w\}, j \in [Q]}$, and $\mathbf{A}_{j,L+1} := \begin{bmatrix} \mathbf{A}_{L+1,1} \\ \mathbf{A}_{j,L+1,2} \\ \vdots \\ \mathbf{A}_{j,L+1,w} \end{bmatrix}$.

Proof. We start by expanding the equation introduced in the lemma:

$$\begin{aligned} & \left\{ \hat{\mathbf{S}}_{\mathbf{x}} \cdot \mathbf{A}_{j,k+1} + \mathbf{E}_{j,\mathbf{x}} \right\}_{j \in [Q], \mathbf{x} \in \{0,1\}^k} \\ &= \left\{ \sum_{l=1}^w \left(\mathbf{S}_0 \cdot \prod_{i=1}^k \mathbf{S}_{i,\mathbf{x}[i]} \cdot \mathbf{A}_{j,k+1,l} \cdot \psi_j(k, \mathbf{x}, l) \right) + \mathbf{E}_{j,\mathbf{x}} \right\}_{j \in [Q], \mathbf{x} \in \{0,1\}^k} \end{aligned}$$

where $\mathbf{A}_{j,k+1} = \begin{bmatrix} \mathbf{A}_{j,k+1,1} \\ \vdots \\ \mathbf{A}_{j,k+1,w} \end{bmatrix}$ and $\psi_j(k, \mathbf{x}, l) = 1$ if and only if the l -th state of the j -th matrix

branching program is equal to one after k steps when taking $\mathbf{x}$ as input (to be more specific, if and only if the l -th entry of $\mathbf{B}_0 \cdot \prod_{i=1}^k \mathbf{B}_{j,i,\mathbf{x}[i]}$ is equal to one). Let w_{BP} be the width of the constraining branching program BP, and $w_{\text{BP}'}$ be the width of the t -CNF branching program that checks input validity, such that $\mathbf{B}_{j,i,b} \in \{0,1\}^{w_{\text{BP}} \times w_{\text{BP}}}$, $\mathbf{B}'_{j,i,b} \in \{0,1\}^{w_{\text{BP}'} \times w_{\text{BP}'}}$ and $w_{\text{BP}} + w_{\text{BP}'} = w$. For $k = L$, and for each $\mathbf{x} \in \{0,1\}^k$, there are two cases:

1. $\mathbf{x}$ satisfies the t -CNF branching program BP' such that $\mathbf{x}$ is a valid input, then it must be the case that $\psi_j(k, \mathbf{x}, l) = 0$ for all $l \in \{w - w_{\text{BP}'} + 1, w\}$ by definition of the t -CNF branching program. Recall that in our construction of the constrained PRF, $\mathbf{A}_{j,k+1,l}$ is set equivalent to $\mathbf{A}_{j',k+1,l}$ only when $l = 1$ and $k = L$. In all other cases, it is generated at random. However, since all the branching programs that the adversary queries is within a purified class, there exists at most one $j \in [Q]$ such that $\psi_j(k, \mathbf{x}, 1) = 1$.
2. $\mathbf{x}$ is not a valid input and does not satisfy BP' . Then, there exists at least one $l \in \{w - w_{\text{BP}'} + 1, w\}$ such that $\psi_j(k, \mathbf{x}, l) = 1$.

Combining the above two cases, when $k = L$, then for each $\mathbf{x} \in \{0,1\}^k$ and $j \in [Q]$, there exists at least one $l \in [w]$, such that (i) $\psi_j(k, \mathbf{x}, l) = 1$, and (ii) for all $j' \neq j$, either $\mathbf{A}_{j',k+1,l}$ is independently

sampling of $\mathbf{A}_{j,k+1,l}$, or $\psi_j(k, \mathbf{x}, l) = 0$. Let $l_{j,\mathbf{x}}$ be such l for each $\mathbf{x} \in \{0,1\}^k$ and $j \in [Q]$. As a result, it follows from Theorem 12.10 that

$$\left\{ \mathbf{S}_0 \cdot \prod_{i=1}^k \mathbf{S}_{i,\mathbf{x}[i]} \cdot \mathbf{A}_{j,k+1,l_{j,\mathbf{x}}} \cdot \psi_j(k, \mathbf{x}, l_{j,\mathbf{x}}) + \mathbf{E}_{j,\mathbf{x}} \right\}_{j \in [Q], \mathbf{x} \in \{0,1\}^k}, \mathbf{S}_0, \{\mathbf{S}_{i,b}\}_{i \in [L], b \in \{0,1\}} \\ \approx_c \left\{ U(\mathbb{Z}_q^{n \times m}) \right\}_{j \in [Q], \mathbf{x} \in \{0,1\}^k}, \mathbf{S}_0, \{\mathbf{S}_{i,b}\}_{i \in [L], b \in \{0,1\}},$$

which in turn implies that

$$\left\{ \sum_{l=1}^w \left(\mathbf{S}_0 \cdot \prod_{i=1}^k \mathbf{S}_{i,\mathbf{x}[i]} \cdot \mathbf{A}_{j,k+1,l} \cdot \psi_j(k, \mathbf{x}, l) \right) + \mathbf{E}_{j,\mathbf{x}} \right\}_{j \in [Q], \mathbf{x} \in \{0,1\}^k}, \hat{\mathbf{S}}_0, \{\hat{\mathbf{S}}_{i,b}\}_{i \in [L], b \in \{0,1\}} \\ \approx_c \left\{ U(\mathbb{Z}_q^{n \times m}) \right\}_{j \in [Q], \mathbf{x} \in \{0,1\}^k}, \hat{\mathbf{S}}_0, \{\hat{\mathbf{S}}_{i,b}\}_{i \in [L], b \in \{0,1\}}.$$

The same also holds for $k < L$, since in this case $\mathbf{A}_{j',k+1,l}$ will always be independently sampled from $\mathbf{A}_{j,k+1,l}$ for all $j \neq j', l \in [w]$. Thus, our lemma follows from Theorem 12.10. $\square$

Lemma 12.14. Assume that LWE and evasive LWE assumptions hold, then for any PPT adversary $\mathcal{A}$, there exists a negligible function $\text{negl}(\cdot)$ such that $\text{Adv}_{\mathcal{A}}^1 - \text{Adv}_{\mathcal{A}}^2 \leq \text{negl}(\lambda)$.

Proof. We proceed by induction on (k, j') , proceeding in the order of $(L, 0), (L, 1), \dots, (L, Q), (L-1, 0), \dots, (L-1, Q), \dots, (1, Q)$ to prove that the following holds:

$$\left\{ \underbrace{\left\{ \hat{\mathbf{S}}_{\mathbf{x}} \cdot \mathbf{A}_{j,k+1} + \mathbf{E}_{j,\mathbf{x}} \right\}_{\mathbf{x} \in \{0,1\}^k}}_{\text{for } j \in \{j'+1, \dots, Q\}}, \left\{ \mathbf{D}_{j,i,b} \right\}_{i \geq k+1, b \in \{0,1\}} \right\}, \left\{ \underbrace{\left\{ \hat{\mathbf{S}}_{\mathbf{x}} \cdot \mathbf{A}_{j,k} + \mathbf{E}_{j,\mathbf{x}} \right\}_{\mathbf{x} \in \{0,1\}^{k-1}}}_{\text{for } j \in [j']}, \left\{ \mathbf{D}_{j,i,b} \right\}_{i \geq k, b \in \{0,1\}} \right\}, \\ 0, \left\{ \hat{\mathbf{S}}_{i,b} \right\}_{i \in [L], b \in \{0,1\}} \\ \approx_c \left\{ \underbrace{\left\{ U(\mathbb{Z}_q^{n \times m}) \right\}_{\mathbf{x} \in \{0,1\}^k}, \left\{ \chi_2^{m \times m} \right\}_{i \geq k+1, b \in \{0,1\}}}_{\text{for } j \in \{j'+1, \dots, Q\}} \right\}, \\ \left\{ \underbrace{\left\{ U(\mathbb{Z}_q^{n \times m}) \right\}_{\mathbf{x} \in \{0,1\}^{k-1}}, \left\{ \chi_2^{m \times m} \right\}_{i \geq k, b \in \{0,1\}}}_{\text{for } j \in [j']} \right\}, \hat{\mathbf{S}}_0, \left\{ \hat{\mathbf{S}}_{i,b} \right\}_{i \in [L], b \in \{0,1\}},$$

where $\hat{\mathbf{S}}_0, \left\{ \hat{\mathbf{S}}_{i,b} \right\}_{i \in [L], b \in \{0,1\}}, \left\{ \mathbf{D}_{j,i,b} \right\}_{j \in [Q], i \geq k+1, b \in \{0,1\}}, \left\{ \mathbf{A}_{j,k+1} \right\}_{j \in [j'+1, \dots, Q]}, \left\{ \mathbf{A}_{j,k} \right\}_{j \in [j']}$ are generated as in Hybrid 1, and $\hat{\mathbf{S}}_{\mathbf{x}} := \hat{\mathbf{S}}_0 \cdot \prod_{i=1}^k \hat{\mathbf{S}}_{i,\mathbf{x}[i]}, \mathbf{E}_{j,\mathbf{x}} \leftarrow \chi_1^{n \times m}$ for $\mathbf{x} \in \{0,1\}^k, j \in [Q]$.

Base case: The base case where $k = L$ and $j' = 0$ holds by Theorem 12.13.

Inductive step: In the inductive step, we show that if the inductive hypothesis holds for k and $j' = j' - 1$, then the hypothesis also holds for k and j' . We start by setting the variables for evasive LWE:

$$\begin{aligned} \mathbf{S} &:= \left\{ \hat{\mathbf{S}}_{\mathbf{x}} \right\}_{\mathbf{x} \in \{0,1\}^{k-1}} \in \mathbb{Z}_q^{2^{k-1}n \times wn}, \mathbf{B} := \mathbf{A}_{j',k}, \\ \mathbf{E}_{k,0}, \mathbf{E}_{k,1} &\leftarrow \chi_0^{n \times m}, \mathbf{E}_{k+1} \leftarrow \chi_1^{2^{k-1}n \times m}, \mathbf{E}_k \leftarrow \chi_1^{2^{k-1}n \times 2m}, \\ \mathbf{P} &:= \left[\hat{\mathbf{S}}_{k,0} \cdot \mathbf{A}_{j',k+1} + \mathbf{E}_{k,0} \mid \hat{\mathbf{S}}_{k,1} \cdot \mathbf{A}_{j',k+1} + \mathbf{E}_{k,1} \right] \in \mathbb{Z}_q^{wn \times 2m}, \\ \mathbf{aux} &:= \left(\begin{array}{c} \hat{\mathbf{S}}_0, \left\{ \hat{\mathbf{S}}_{i,b} \right\}_{i \in [L], b \in \{0,1\}}, \\ \left\{ \left\{ \hat{\mathbf{S}}_{\mathbf{x}} \cdot \mathbf{A}_{j,k+1} + \mathbf{E}_{j,\mathbf{x}} \right\}_{j \in \{j'+1, \dots, Q\}}, \left\{ \mathbf{D}_{j,i,b} \right\}_{i \geq k+1, b \in \{0,1\}} \right\}_{j \in \{j'+1, \dots, Q\}}, \\ \left\{ \left\{ \hat{\mathbf{S}}_{\mathbf{x}} \cdot \mathbf{A}_{j,k} + \mathbf{E}_{j,\mathbf{x}} \right\}_{\mathbf{x} \in \{0,1\}^{k-1}}, \left\{ \mathbf{D}_{j,i,b} \right\}_{i \geq k, b \in \{0,1\}} \right\}_{j \in [j'-1]} \end{array} \right), \end{aligned}$$

where $\hat{\mathbf{S}}_0, \left\{ \hat{\mathbf{S}}_{i,b} \right\}_{i \in [L], b \in \{0,1\}}, \left\{ \mathbf{D}_{j,i,b} \right\}_{j \in \{j'+1, \dots, L\}, i \geq k+1, b \in \{0,1\}}$ follows from Hybrid 1, $\left\{ \mathbf{E}_{j,\mathbf{x}} \right\}_{\mathbf{x} \in \{0,1\}^k, j \in \{j'+1, \dots, Q\}}$ are sampled identically as our inductive claim. We first consider the inductive case from $k, j' = 0$ to $k, j' = 1$.

Claim 12.15. Under the distribution of $\mathbf{E}_{k+1}$, we have that $\mathbf{S} \cdot \mathbf{P} + \mathbf{E}_{k+1}$ is statistically close to the distribution of $\left[\mathbf{S} \cdot \hat{\mathbf{S}}_{k,0} \cdot \mathbf{A}_{j',k+1} \mid \mathbf{S} \cdot \hat{\mathbf{S}}_{k,1} \cdot \mathbf{A}_{j',k+1} \right] + \mathbf{E}_{k+1}$.

Proof. We omit the analysis here, since it follows from smudging and is nearly identical to the analysis of Theorem 12.11. $\square$

Claim 12.16. Assume that the inductive hypothesis holds, then under the distribution of the variables in the inductive step, the following holds:

$$\begin{aligned} &\left(\mathbf{S} \cdot \mathbf{B} + \mathbf{E}_k, \left[\mathbf{S} \cdot \hat{\mathbf{S}}_{k,0} \cdot \mathbf{A}_{j',k+1} \mid \mathbf{S} \cdot \hat{\mathbf{S}}_{k,1} \cdot \mathbf{A}_{j',k+1} \right] + \mathbf{E}_{k+1}, \mathbf{aux} \right) \\ &\approx_c \left(\mathbf{S} \cdot \mathbf{B} + \mathbf{E}_k, U \left(\mathbb{Z}_q^{2^{j-1}n \times 2m} \right), \mathbf{aux} \right). \end{aligned}$$

Proof. The claim directly follows from our inductive hypothesis, because $\mathbf{S} \cdot \mathbf{B} + \mathbf{E}_k = \left\{ \hat{\mathbf{S}}_{\mathbf{x}} \right\}_{\mathbf{x} \in \{0,1\}^{k-1}} \cdot \mathbf{A}_{j',k} + \mathbf{E}_k$ can be sampled with the knowledge of $\left\{ \hat{\mathbf{S}}_{i,b} \right\}_{i \in [L], b \in \{0,1\}}$ by picking $\mathbf{A}_{j',k}$ and $\mathbf{E}_k$ from appropriate distributions. $\square$

Next, by the assumption explained in our lemma statement, we have the following:

$$\left(\mathbf{S} \cdot \mathbf{B} + \mathbf{E}_k, U \left(\mathbb{Z}_q^{2^{j-1}n \times 2m} \right), \mathbf{aux} \right) \approx_c \left(U \left(\mathbb{Z}_q^{2^{j-1}n \times m} \right), U \left(\mathbb{Z}_q^{2^{j-1}n \times 2m} \right), \mathbf{aux} \right).$$

Then, combining the above, the precondition of evasive LWE holds:

$$(\mathbf{S} \cdot \mathbf{B} + \mathbf{E}_k, \mathbf{S} \cdot \mathbf{P} + \mathbf{E}_{k+1}, \mathbf{aux}) \approx_c \left(U \left(\mathbb{Z}_q^{2^{j-1}n \times m} \right), U \left(\mathbb{Z}_q^{2^{j-1}n \times 2m} \right), \mathbf{aux} \right).$$

As a result, it follows from evasive LWE that

$$(\mathbf{S} \cdot \mathbf{B} + \mathbf{E}_k, \mathbf{B}^{-1}(\mathbf{P}), \text{aux}) \approx_c \left(U \left(\mathbb{Z}_q^{2^{j-1}n \times m} \right), \mathbf{B}^{-1}(\mathbf{P}), \text{aux} \right).$$

Again since $\mathbf{A}_{j',k}$ is independent from auxiliary information aux , then by our setup of $\mathbf{B}$ and $\mathbf{P}$, it follows from Theorem 12.3 that

$$(\mathbf{S} \cdot \mathbf{B} + \mathbf{E}_k, \mathbf{B}^{-1}(\mathbf{P}), \text{aux}) \approx_c \left(U \left(\mathbb{Z}_q^{2^{j-1}n \times m} \right), \chi_2^{m \times 2m}, \text{aux} \right).$$

By expanding the above equation, we have proved that:

$$\begin{aligned} & \left\{ \underbrace{\left\{ \hat{\mathbf{S}}_{\mathbf{x}} \cdot \mathbf{A}_{j,k+1} + \mathbf{E}_{j,\mathbf{x}} \right\}_{\mathbf{x} \in \{0,1\}^k}, \{ \mathbf{D}_{j,i,b} \}_{i \geq k+1, b \in \{0,1\}}}_{\text{for } j \in \{j'+1, \dots, Q\}} \right\}, \\ & \left\{ \hat{\mathbf{S}}_{\mathbf{x}} \cdot \mathbf{A}_{j',k} + \mathbf{E}_{j',\mathbf{x}} \right\}_{\mathbf{x} \in \{0,1\}^{k-1}}, \{ \mathbf{D}_{j',i,b} \}_{i \geq k, b \in \{0,1\}}, \\ & \left\{ \underbrace{\left\{ \hat{\mathbf{S}}_{\mathbf{x}} \cdot \mathbf{A}_{j,k} + \mathbf{E}_{j,\mathbf{x}} \right\}_{\mathbf{x} \in \{0,1\}^{k-1}}, \{ \mathbf{D}_{j,i,b} \}_{i \geq k, b \in \{0,1\}}}_{\text{for } j \in [j'-1]} \right\}, \hat{\mathbf{S}}_0, \{ \hat{\mathbf{S}}_{i,b} \}_{i \in [L], b \in \{0,1\}} \\ & \approx_c \left\{ \underbrace{\left\{ \hat{\mathbf{S}}_{\mathbf{x}} \cdot \mathbf{A}_{j,k+1} + \mathbf{E}_{j,\mathbf{x}} \right\}_{\mathbf{x} \in \{0,1\}^k}, \{ \mathbf{D}_{j,i,b} \}_{i \geq k+1, b \in \{0,1\}}}_{\text{for } j \in \{j'+1, \dots, Q\}} \right\}, \\ & \left\{ U(\mathbb{Z}_q^{n \times m}) \right\}_{\mathbf{x} \in \{0,1\}^{k-1}}, \{ \chi_2^{m \times m} \}_{i \geq k, b \in \{0,1\}}, \\ & \left\{ \underbrace{\left\{ \hat{\mathbf{S}}_{\mathbf{x}} \cdot \mathbf{A}_{j,k} + \mathbf{E}_{j,\mathbf{x}} \right\}_{\mathbf{x} \in \{0,1\}^{k-1}}, \{ \mathbf{D}_{j,i,b} \}_{i \geq k, b \in \{0,1\}}}_{\text{for } j \in [j'-1]} \right\}, \hat{\mathbf{S}}_0, \{ \hat{\mathbf{S}}_{i,b} \}_{i \in [L], b \in \{0,1\}}. \end{aligned}$$

To prove the inductive step, we still need to prove that the auxiliary information looks random.

Recall that the inductive hypothesis for k and $j' = j' - 1$ where we have:

$$\begin{aligned}
& \left\{ \underbrace{\left\{ \hat{\mathbf{S}}_{\mathbf{x}} \cdot \mathbf{A}_{j,k+1} + \mathbf{E}_{j,\mathbf{x}} \right\}_{\mathbf{x} \in \{0,1\}^k}, \left\{ \mathbf{D}_{j,i,b} \right\}_{i \geq k+1, b \in \{0,1\}}}_{\text{for } j \in \{j', \dots, Q\}} \right\}, \\
& \left\{ \underbrace{\left\{ \hat{\mathbf{S}}_{\mathbf{x}} \cdot \mathbf{A}_{j,k} + \mathbf{E}_{j,\mathbf{x}} \right\}_{\mathbf{x} \in \{0,1\}^{k-1}}, \left\{ \mathbf{D}_{j,i,b} \right\}_{i \geq k, b \in \{0,1\}}}_{\text{for } j \in [j'-1]} \right\}, \hat{\mathbf{S}}_0, \left\{ \hat{\mathbf{S}}_{i,b} \right\}_{i \in [L], b \in \{0,1\}} \\
& \approx_c \left\{ \underbrace{\left\{ U(\mathbb{Z}_q^{n \times m}) \right\}_{\mathbf{x} \in \{0,1\}^k}, \left\{ \chi_2^{m \times m} \right\}_{i \geq k+1, b \in \{0,1\}}}_{\text{for } j \in \{j', \dots, Q\}} \right\}, \\
& \left\{ \underbrace{\left\{ U(\mathbb{Z}_q^{n \times m}) \right\}_{\mathbf{x} \in \{0,1\}^{k-1}}, \left\{ \chi_2^{m \times m} \right\}_{i \geq k, b \in \{0,1\}}}_{\text{for } j \in [j'-1]} \right\}, \hat{\mathbf{S}}_0, \left\{ \hat{\mathbf{S}}_{i,b} \right\}_{i \in [L], b \in \{0,1\}}.
\end{aligned}$$

Thus by assuming the inductive hypothesis holds for k and $j' = j' - 1$, and by combining the above equations, the following holds:

$$\begin{aligned}
& \left\{ \underbrace{\left\{ \hat{\mathbf{S}}_{\mathbf{x}} \cdot \mathbf{A}_{j,k+1} + \mathbf{E}_{j,\mathbf{x}} \right\}_{\mathbf{x} \in \{0,1\}^k}, \left\{ \mathbf{D}_{j,i,b} \right\}_{i \geq k+1, b \in \{0,1\}}}_{\text{for } j \in \{j'+1, \dots, Q\}} \right\}, \\
& \left\{ \hat{\mathbf{S}}_{\mathbf{x}} \cdot \mathbf{A}_{j',k} + \mathbf{E}_{j',\mathbf{x}} \right\}_{\mathbf{x} \in \{0,1\}^{k-1}}, \left\{ \mathbf{D}_{j',i,b} \right\}_{i \geq k, b \in \{0,1\}}, \\
& \left\{ \underbrace{\left\{ \hat{\mathbf{S}}_{\mathbf{x}} \cdot \mathbf{A}_{j,k} + \mathbf{E}_{j,\mathbf{x}} \right\}_{\mathbf{x} \in \{0,1\}^{k-1}}, \left\{ \mathbf{D}_{j,i,b} \right\}_{i \geq k, b \in \{0,1\}}}_{\text{for } j \in [j'-1]} \right\}, \hat{\mathbf{S}}_0, \left\{ \hat{\mathbf{S}}_{i,b} \right\}_{i \in [L], b \in \{0,1\}} \\
& \approx_c \left\{ \underbrace{\left\{ U(\mathbb{Z}_q^{n \times m}) \right\}_{\mathbf{x} \in \{0,1\}^k}, \left\{ \chi_2^{m \times m} \right\}_{i \geq k+1, b \in \{0,1\}}}_{\text{for } j \in \{j'+1, \dots, Q\}} \right\}, \\
& \left\{ U(\mathbb{Z}_q^{n \times m}) \right\}_{\mathbf{x} \in \{0,1\}^{k-1}}, \left\{ \chi_2^{m \times m} \right\}_{i \geq k, b \in \{0,1\}}, \\
& \left\{ \underbrace{\left\{ U(\mathbb{Z}_q^{n \times m}) \right\}_{\mathbf{x} \in \{0,1\}^{k-1}}, \left\{ \chi_2^{m \times m} \right\}_{i \geq k, b \in \{0,1\}}}_{\text{for } j \in [j'-1]} \right\}, \hat{\mathbf{S}}_0, \left\{ \hat{\mathbf{S}}_{i,b} \right\}_{i \in [L], b \in \{0,1\}}.
\end{aligned}$$

The above equation says that inductive hypothesis also holds for k and j' . Following from this

analysis, we move from inductive step indexed by $(k, j' - 1)$ to (k, j') . We also note that the step k, Q is identical to step $k - 1, 0$.

We will invoke the inductive hypothesis a number of LQ times. Let $\mathcal{A}_{k,j'}$ denote an adversary that breaks the inductive statement indexed by (k, j') . Let $\lambda' = 2^{LQ}\lambda$ be the security parameter of the underlying evasive LWE assumption, then the dimensions matrices of $\mathbf{S}, \mathbf{B}, \mathbf{P}$ are bounded by some polynomial of λ' . It follows from evasive LWE that $\text{Adv}_{\mathcal{A}_{k,j'-1}} \geq \text{Adv}_{\mathcal{A}_{k,j'}} / \text{poly}(2^{LQ}\lambda)$ and $\text{Time}(\mathcal{A}_{k,j'-1}) \leq \text{Time}(\mathcal{A}_{k,j'}) \cdot \text{poly}(2^{LQ}\lambda)$, which implies that $\text{Adv}_{\mathcal{A}_{L,0}} \geq \text{Adv}_{\mathcal{A}_{1,Q}} / \text{poly}(2^{(LQ)^2}\lambda^{LQ})$ and $\text{Time}(\mathcal{A}_{L,0}) \leq \text{Time}(\mathcal{A}_{1,Q}) \cdot \text{poly}(2^{(LQ)^2}\lambda^{LQ})$. Thus we require $\text{poly}(2^{(LQ)^2}\lambda^{LQ})$ hardness from LWE, which follows from our parameter settings.

Thus, the inductive proof holds. Next, observe that the inductive step of $L, 0$ corresponds to Hybrid 1, and $1, Q$ corresponds to Hybrid 2, so that our lemma follows. $\square$

Note that in Hybrid 2, the output by the challenger during the challenge query should look uniformly random, which follows from Theorem 12.13. Then, combining Theorem 12.9 and Theorem 12.14, the pseudorandomness property follows. Next, we observe that in Hybrid 2, the constrained keys are fully simulated, so that constraint privacy follows. We conclude that Theorem 12.8 holds.

References

- [AAB⁺15] Shashank Agrawal, Shweta Agrawal, Saikrishna Badrinarayanan, Abishek Kumarasubramanian, Manoj Prabhakaran, and Amit Sahai. On the practical security of inner product functional encryption. In Jonathan Katz, editor, *PKC 2015: 18th International Conference on Theory and Practice of Public Key Cryptography*, volume 9020 of *Lecture Notes in Computer Science*, pages 777–798, Gaithersburg, MD, USA, March 30 – April 1, 2015. Springer Berlin Heidelberg, Germany.
- [ABSV15] Prabhanjan Ananth, Zvika Brakerski, Gil Segev, and Vinod Vaikuntanathan. From selective to adaptive security in functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 657–677, Santa Barbara, CA, USA, August 16–20, 2015. Springer Berlin Heidelberg, Germany.
- [ACPS09] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 595–618, Santa Barbara, CA, USA, August 16–20, 2009. Springer Berlin Heidelberg, Germany.
- [AJL⁺12] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 483–501, Cambridge, UK, April 15–19, 2012. Springer Berlin Heidelberg, Germany.

- [AKSX04] Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, and Yirong Xu. Order preserving encryption for numeric data. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 563–574, 2004.
- [AKY24] Shweta Agrawal, Simran Kumari, and Shota Yamada. Compact pseudorandom functional encryption from evasive LWE. Cryptology ePrint Archive, Report 2024/1719, 2024.
- [AMN⁺18] Nuttapong Attrapadung, Takahiro Matsuda, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Constrained prfs for nc1 in traditional groups (from crypto 2018). *IEICE Technical Report; IEICE Tech. Rep.*, 118(212):61–61, 2018.
- [AMN⁺19] Nuttapong Attrapadung, Takahiro Matsuda, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Adaptively single-key secure constrained PRFs for NC¹. In Dongdai Lin and Kazue Sako, editors, *PKC 2019: 22nd International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 11443 of *Lecture Notes in Computer Science*, pages 223–253, Beijing, China, April 14–17, 2019. Springer, Cham, Switzerland.
- [AMYY25] Shweta Agrawal, Anuja Modi, Anshu Yadav, and Shota Yamada. Evasive LWE: Attacks, variants & obfustopia. Cryptology ePrint Archive, Report 2025/375, 2025.
- [AP20] Shweta Agrawal and Alice Pellet-Mary. Indistinguishability obfuscation without maps: Attacks and fixes for noisy linear FE. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020, Part I*, volume 12105 of *Lecture Notes in Computer Science*, pages 110–140, Zagreb, Croatia, May 10–14, 2020. Springer, Cham, Switzerland.
- [ARYY23] Shweta Agrawal, Mélissa Rossi, Anshu Yadav, and Shota Yamada. Constant input attribute based (and predicate) encryption from evasive and tensor LWE. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023, Part IV*, volume 14084 of *Lecture Notes in Computer Science*, pages 532–564, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Cham, Switzerland.
- [Bar86] David A. Mix Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC¹. In *18th Annual ACM Symposium on Theory of Computing*, pages 1–5, Berkeley, CA, USA, May 28–30, 1986. ACM Press.
- [BBKK18] Boaz Barak, Zvika Brakerski, Ilan Komargodski, and Praves K. Kothari. Limits on low-degree pseudorandom generators (or: Sum-of-squares meets program obfuscation). In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 649–679, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Cham, Switzerland.
- [BCLO09] Alexandra Boldyreva, Nathan Chenette, Younho Lee, and Adam O’Neill. Order-preserving symmetric encryption. In Antoine Joux, editor, *Advances in Cryptology – EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 224–241, Cologne, Germany, April 26–30, 2009. Springer Berlin Heidelberg, Germany.

- [BCO11] Alexandra Boldyreva, Nathan Chenette, and Adam O’Neill. Order-preserving encryption revisited: Improved security analysis and alternative solutions. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 578–595, Santa Barbara, CA, USA, August 14–18, 2011. Springer Berlin Heidelberg, Germany.
- [BDFP83] Allan Borodin, Danny Dolev, Faith E. Fich, and Wolfgang J. Paul. Bounds for width two branching programs. In *15th Annual ACM Symposium on Theory of Computing*, pages 87–93, Boston, MA, USA, April 25–27, 1983. ACM Press.
- [BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229, Santa Barbara, CA, USA, August 19–23, 2001. Springer Berlin Heidelberg, Germany.
- [BFP⁺15] Abhishek Banerjee, Georg Fuchsbauer, Chris Peikert, Krzysztof Pietrzak, and Sophie Stevens. Key-homomorphic constrained pseudorandom functions. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part II*, volume 9015 of *Lecture Notes in Computer Science*, pages 31–60, Warsaw, Poland, March 23–25, 2015. Springer Berlin Heidelberg, Germany.
- [BGG⁺14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EURO-CRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 533–556, Copenhagen, Denmark, May 11–15, 2014. Springer Berlin Heidelberg, Germany.
- [BGH⁺15] Zvika Brakerski, Craig Gentry, Shai Halevi, Tancrede Lepoint, Amit Sahai, and Mehdi Tibouchi. Cryptanalysis of the quadratic zero-testing of GGH. Cryptology ePrint Archive, Report 2015/845, 2015.
- [BGI⁺01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 1–18, August 2001.
- [BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In Hugo Krawczyk, editor, *PKC 2014: 17th International Conference on Theory and Practice of Public Key Cryptography*, volume 8383 of *Lecture Notes in Computer Science*, pages 501–519, Buenos Aires, Argentina, March 26–28, 2014. Springer Berlin Heidelberg, Germany.
- [Bit17] Nir Bitansky. Verifiable random functions from non-interactive witness-indistinguishable proofs. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017: 15th Theory of Cryptography Conference, Part II*, volume 10678 of *Lecture Notes in Computer Science*, pages 567–594, Baltimore, MD, USA, November 12–15, 2017. Springer, Cham, Switzerland.

- [BKM17] Dan Boneh, Sam Kim, and Hart William Montgomery. Private puncturable PRFs from standard lattice assumptions. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017, Part I*, volume 10210 of *Lecture Notes in Computer Science*, pages 415–445, Paris, France, April 30 – May 4, 2017. Springer, Cham, Switzerland.
- [BKW17] Dan Boneh, Sam Kim, and David J. Wu. Constrained keys for invertible pseudorandom functions. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017: 15th Theory of Cryptography Conference, Part I*, volume 10677 of *Lecture Notes in Computer Science*, pages 237–263, Baltimore, MD, USA, November 12–15, 2017. Springer, Cham, Switzerland.
- [BLMR13] Dan Boneh, Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. Key homomorphic PRFs and their applications. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 410–428, Santa Barbara, CA, USA, August 18–22, 2013. Springer Berlin Heidelberg, Germany.
- [BLR⁺15] Dan Boneh, Kevin Lewi, Mariana Raykova, Amit Sahai, Mark Zhandry, and Joe Zimmerman. Semantically secure order-revealing encryption: Multi-input functional encryption without obfuscation. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 563–594, Sofia, Bulgaria, April 26–30, 2015. Springer Berlin Heidelberg, Germany.
- [BLW17] Dan Boneh, Kevin Lewi, and David J. Wu. Constraining pseudorandom functions privately. In Serge Fehr, editor, *PKC 2017: 20th International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 10175 of *Lecture Notes in Computer Science*, pages 494–524, Amsterdam, The Netherlands, March 28–31, 2017. Springer Berlin Heidelberg, Germany.
- [BMO⁺25] Amik Raj Behera, Pierre Meyer, Claudio Orlandi, Lawrence Roy, and Peter Scholl. Privately constrained PRFs from DCR: Puncturing and bounded waring rank. Cryptology ePrint Archive, Report 2025/230, 2025.
- [BP14] Abhishek Banerjee and Chris Peikert. New and improved key-homomorphic pseudorandom functions. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 353–370, Santa Barbara, CA, USA, August 17–21, 2014. Springer Berlin Heidelberg, Germany.
- [BS02] Dan Boneh and Alice Silverberg. Applications of multilinear forms to cryptography. Cryptology ePrint Archive, Report 2002/080, 2002.
- [BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Yuval Ishai, editor, *TCC 2011: 8th Theory of Cryptography Conference*, volume 6597 of *Lecture Notes in Computer Science*, pages 253–273, Providence, RI, USA, March 28–30, 2011. Springer Berlin Heidelberg, Germany.

- [BTVW17] Zvika Brakerski, Rotem Tsabary, Vinod Vaikuntanathan, and Hoeteck Wee. Private constrained PRFs (and more) from LWE. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017: 15th Theory of Cryptography Conference, Part I*, volume 10677 of *Lecture Notes in Computer Science*, pages 264–302, Baltimore, MD, USA, November 12–15, 2017. Springer, Cham, Switzerland.
- [BÜW24] Chris Brzuska, Akin Ünal, and Ivy K. Y. Woo. Evasive LWE assumptions: Definitions, classes, and counterexamples. In Kai-Min Chung and Yu Sasaki, editors, *Advances in Cryptology – ASIACRYPT 2024, Part IV*, volume 15487 of *Lecture Notes in Computer Science*, pages 418–449, Kolkata, India, December 9–13, 2024. Springer, Singapore, Singapore.
- [BV15] Zvika Brakerski and Vinod Vaikuntanathan. Constrained key-homomorphic PRFs from standard lattice assumptions - or: How to secretly embed a circuit in your PRF. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part II*, volume 9015 of *Lecture Notes in Computer Science*, pages 1–30, Warsaw, Poland, March 23–25, 2015. Springer Berlin Heidelberg, Germany.
- [BW07] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In Salil P. Vadhan, editor, *TCC 2007: 4th Theory of Cryptography Conference*, volume 4392 of *Lecture Notes in Computer Science*, pages 535–554, Amsterdam, The Netherlands, February 21–24, 2007. Springer Berlin Heidelberg, Germany.
- [BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology – ASIACRYPT 2013, Part II*, volume 8270 of *Lecture Notes in Computer Science*, pages 280–300, Bangalore, India, December 1–5, 2013. Springer Berlin Heidelberg, Germany.
- [CC17] Ran Canetti and Yilei Chen. Constraint-hiding constrained PRFs for NC^1 from LWE. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017, Part I*, volume 10210 of *Lecture Notes in Computer Science*, pages 446–476, Paris, France, April 30 – May 4, 2017. Springer, Cham, Switzerland.
- [CDD⁺24] Geoffroy Couteau, Lalita Devadas, Srinivas Devadas, Alexander Koch, and Sacha Servan-Schreiber. QuietOT: Lightweight oblivious transfer with a public-key setup. In Kai-Min Chung and Yu Sasaki, editors, *Advances in Cryptology – ASIACRYPT 2024, Part II*, volume 15485 of *Lecture Notes in Computer Science*, pages 197–231, Kolkata, India, December 9–13, 2024. Springer, Singapore, Singapore.
- [CFN94] Benny Chor, Amos Fiat, and Moni Naor. Tracing traitors. In Yvo Desmedt, editor, *Advances in Cryptology – CRYPTO’94*, volume 839 of *Lecture Notes in Computer Science*, pages 257–270, Santa Barbara, CA, USA, August 21–25, 1994. Springer Berlin Heidelberg, Germany.
- [CGH⁺15] Jean-Sébastien Coron, Craig Gentry, Shai Halevi, Tancrede Lepoint, Hemanta K. Maji, Eric Miles, Mariana Raykova, Amit Sahai, and Mehdi Tibouchi. Zeroizing without low-level zeroes: New MMAP attacks and their limitations. In Rosario Genaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015*,

- Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 247–266, Santa Barbara, CA, USA, August 16–20, 2015. Springer Berlin Heidelberg, Germany.
- [CGKO06] Reza Curtmola, Juan A. Garay, Seny Kamara, and Rafail Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006: 13th Conference on Computer and Communications Security*, pages 79–88, Alexandria, Virginia, USA, October 30 – November 3, 2006. ACM Press.
 - [CHL⁺14] Jung Hee Cheon, Kyoohyung Han, Changmin Lee, Hansol Ryu, and Damien Stehlé. Cryptanalysis on the multilinear map over the integers and its related problems. *Cryptology ePrint Archive*, 2014.
 - [CHN⁺16] Aloni Cohen, Justin Holmgren, Ryo Nishimaki, Vinod Vaikuntanathan, and Daniel Wichs. Watermarking cryptographic capabilities. In Daniel Wichs and Yishay Mansour, editors, *48th Annual ACM Symposium on Theory of Computing*, pages 1115–1127, Cambridge, MA, USA, June 18–21, 2016. ACM Press.
 - [CK10] Melissa Chase and Seny Kamara. Structured encryption and controlled disclosure. In Masayuki Abe, editor, *Advances in Cryptology – ASIACRYPT 2010*, volume 6477 of *Lecture Notes in Computer Science*, pages 577–594, Singapore, December 5–9, 2010. Springer Berlin Heidelberg, Germany.
 - [CLT14] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Cryptanalysis of two candidate fixes of multilinear maps over the integers. *Cryptology ePrint Archive*, Report 2014/975, 2014.
 - [CLT15] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. New multilinear maps over the integers. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 267–286, Santa Barbara, CA, USA, August 16–20, 2015. Springer Berlin Heidelberg, Germany.
 - [CMPR23] Geoffroy Couteau, Pierre Meyer, Alain Passelègue, and Mahshid Riahinia. Constrained pseudorandom functions from homomorphic secret sharing. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023, Part III*, volume 14006 of *Lecture Notes in Computer Science*, pages 194–224, Lyon, France, April 23–27, 2023. Springer, Cham, Switzerland.
 - [Coc01] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In Bahram Honary, editor, *8th IMA International Conference on Cryptography and Coding*, volume 2260 of *Lecture Notes in Computer Science*, pages 360–363, Cirencester, UK, December 17–19, 2001. Springer Berlin Heidelberg, Germany.
 - [CVW⁺18a] Yilei Chen, Vinod Vaikuntanathan, Brent Waters, Hoeteck Wee, and Daniel Wichs. Traitor-tracing from LWE made simple and attribute-based. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018: 16th Theory of Cryptography Conference, Part II*, volume 11240 of *Lecture Notes in Computer Science*, pages 341–369, Panaji, India, November 11–14, 2018. Springer, Cham, Switzerland.

- [CVW18b] Yilei Chen, Vinod Vaikuntanathan, and Hoeteck Wee. GGH15 beyond permutation branching programs: Proofs, attacks, and candidates. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part II*, volume 10992 of *Lecture Notes in Computer Science*, pages 577–607, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Cham, Switzerland.
- [DJM⁺25] Nico Döttling, Abhishek Jain, Giulio Malavolta, Surya Mathialagan, and Vinod Vaikuntanathan. Simple and general counterexamples for private-coin evasive IWE. In *Annual International Cryptology Conference*, pages 73–92. Springer, 2025.
- [DKN⁺20] Alex Davidson, Shuichi Katsumata, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Adaptively secure constrained pseudorandom functions in the standard model. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020, Part I*, volume 12170 of *Lecture Notes in Computer Science*, pages 559–589, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Cham, Switzerland.
- [DKW16] Apoorva Deshpande, Venkata Koppula, and Brent Waters. Constrained pseudorandom functions for unconstrained inputs. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 124–153, Vienna, Austria, May 8–12, 2016. Springer Berlin Heidelberg, Germany.
- [DKXY02] Yevgeniy Dodis, Jonathan Katz, Shouhuai Xu, and Moti Yung. Key-insulated public key cryptosystems. In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 65–82, Amsterdam, The Netherlands, April 28 – May 2, 2002. Springer Berlin Heidelberg, Germany.
- [DORS08] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam D. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139, 2008.
- [DRS04] Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 523–540, Interlaken, Switzerland, May 2–6, 2004. Springer Berlin Heidelberg, Germany.
- [GGH13a] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 1–17, Athens, Greece, May 26–30, 2013. Springer Berlin Heidelberg, Germany.
- [GGH⁺13b] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th Annual Symposium on Foundations of Computer Science*, pages 40–49, Berkeley, CA, USA, October 26–29, 2013. IEEE Computer Society Press.

- [GGH15] Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graph-induced multilinear maps from lattices. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part II*, volume 9015 of *Lecture Notes in Computer Science*, pages 498–527, Warsaw, Poland, March 23–25, 2015. Springer Berlin Heidelberg, Germany.
- [GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, October 1986.
- [GHKW17] Rishab Goyal, Susan Hohenberger, Venkata Koppula, and Brent Waters. A generic approach to constructing and proving verifiable random functions. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017: 15th Theory of Cryptography Conference, Part II*, volume 10678 of *Lecture Notes in Computer Science*, pages 537–566, Baltimore, MD, USA, November 12–15, 2017. Springer, Cham, Switzerland.
- [GKP⁺13] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nikolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th Annual ACM Symposium on Theory of Computing*, pages 555–564, Palo Alto, CA, USA, June 1–4, 2013. ACM Press.
- [GKR25] Rishab Goyal, Venkata Koppula, and Mahesh Sreekumar Rajasree. A note on adaptive security in hierarchical identity-based encryption. In *Annual International Cryptology Conference*, pages 101–136. Springer, 2025.
- [GKW17a] Rishab Goyal, Venkata Koppula, and Brent Waters. Lockable obfuscation. In Chris Umans, editor, *58th Annual Symposium on Foundations of Computer Science*, pages 612–621, Berkeley, CA, USA, October 15–17, 2017. IEEE Computer Society Press.
- [GKW17b] Rishab Goyal, Venkata Koppula, and Brent Waters. Separating semantic and circular security for symmetric-key bit encryption from the learning with errors assumption. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017, Part II*, volume 10211 of *Lecture Notes in Computer Science*, pages 528–557, Paris, France, April 30 – May 4, 2017. Springer, Cham, Switzerland.
- [GKW18] Rishab Goyal, Venkata Koppula, and Brent Waters. Collusion resistant traitor tracing from learning with errors. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *50th Annual ACM Symposium on Theory of Computing*, pages 660–670, Los Angeles, CA, USA, June 25–29, 2018. ACM Press.
- [GKWW21] Rishab Goyal, Sam Kim, Brent Waters, and David J. Wu. Beyond software watermarking: Traitor-tracing for pseudorandom functions. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2021, Part III*, volume 13092 of *Lecture Notes in Computer Science*, pages 250–280, Singapore, December 6–10, 2021. Springer, Cham, Switzerland.
- [GLW12] Shafi Goldwasser, Allison B. Lewko, and David A. Wilson. Bounded-collusion IBE from key homomorphism. In Ronald Cramer, editor, *TCC 2012: 9th Theory of Cryptography Conference*, volume 7194 of *Lecture Notes in Computer Science*, pages 564–581, Taormina, Sicily, Italy, March 19–21, 2012. Springer Berlin Heidelberg, Germany.

- [GLW14] Craig Gentry, Allison B. Lewko, and Brent Waters. Witness encryption from instance independent assumptions. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 426–443, Santa Barbara, CA, USA, August 17–21, 2014. Springer Berlin Heidelberg, Germany.
- [Goh03] Eu-Jin Goh. Secure indexes. Cryptology ePrint Archive, Report 2003/216, 2003.
- [GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006: 13th Conference on Computer and Communications Security*, pages 89–98, Alexandria, Virginia, USA, October 30 – November 3, 2006. ACM Press. Available as Cryptology ePrint Archive Report 2006/309.
- [GQWW19] Rishab Goyal, Willy Quach, Brent Waters, and Daniel Wichs. Broadcast and trace with N^ϵ ciphertext size from standard assumptions. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 826–855, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Cham, Switzerland.
- [GS02] Craig Gentry and Alice Silverberg. Hierarchical ID-based cryptography. In Yuliang Zheng, editor, *Advances in Cryptology – ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 548–566, Queenstown, New Zealand, December 1–5, 2002. Springer Berlin Heidelberg, Germany.
- [GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 75–92, Santa Barbara, CA, USA, August 18–22, 2013. Springer Berlin Heidelberg, Germany.
- [GVW12] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 162–179, Santa Barbara, CA, USA, August 19–23, 2012. Springer Berlin Heidelberg, Germany.
- [GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th Annual ACM Symposium on Theory of Computing*, pages 545–554, Palo Alto, CA, USA, June 1–4, 2013. ACM Press.
- [GVW15] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits from LWE. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 503–523, Santa Barbara, CA, USA, August 16–20, 2015. Springer Berlin Heidelberg, Germany.

- [HHY25] Tzu-Hsiang Huang, Wei-Hsiang Hung, and Shota Yamada. A note on obfuscation-based attacks on private-coin evasive LWE. *Cryptology ePrint Archive*, Report 2025/421, 2025.
- [HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.
- [HJL21] Samuel B. Hopkins, Aayush Jain, and Huijia Lin. Counterexamples to new circular security assumptions underlying iO. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part II*, volume 12826 of *Lecture Notes in Computer Science*, pages 673–700, Virtual Event, August 16–20, 2021. Springer, Cham, Switzerland.
- [HJL25] Yao-Ching Hsieh, Aayush Jain, and Huijia Lin. Lattice-based post-quantum iO from circular security with random opening assumption (part II: zeroizing attacks against private-coin evasive LWE assumptions). *Cryptology ePrint Archive*, Report 2025/390, 2025.
- [HKKW19] Dennis Hofheinz, Akshay Kamath, Venkata Koppula, and Brent Waters. Adaptively secure constrained pseudorandom functions. In Ian Goldberg and Tyler Moore, editors, *FC 2019: 23rd International Conference on Financial Cryptography and Data Security*, volume 11598 of *Lecture Notes in Computer Science*, pages 357–376, Frigate Bay, St. Kitts and Nevis, February 18–22, 2019. Springer, Cham, Switzerland.
- [HL02] Jeremy Horwitz and Ben Lynn. Toward hierarchical identity-based encryption. In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 466–481, Amsterdam, The Netherlands, April 28 – May 2, 2002. Springer Berlin Heidelberg, Germany.
- [HLL23] Yao-Ching Hsieh, Huijia Lin, and Ji Luo. Attribute-based encryption for circuits of unbounded depth from lattices. In *64th Annual Symposium on Foundations of Computer Science*, pages 415–434, Santa Cruz, CA, USA, November 6–9, 2023. IEEE Computer Society Press.
- [HLL24] Yao-Ching Hsieh, Huijia Lin, and Ji Luo. A general framework for lattice-based ABE using evasive inner-product functional encryption. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology – EUROCRYPT 2024, Part II*, volume 14652 of *Lecture Notes in Computer Science*, pages 433–464, Zurich, Switzerland, May 26–30, 2024. Springer, Cham, Switzerland.
- [HLL25] Yao-Ching Hsieh, Huijia Lin, and Ji Luo. Attribute-based encryption for circuits of unbounded depth from lattices: garbled circuits of optimal size, laconic functional evaluation, and more. *SIAM Journal on Computing*, pages FOCS23–90, 2025.
- [HMW07] Nicholas Hopper, David Molnar, and David Wagner. From weak to strong watermarking. In Salil P. Vadhan, editor, *TCC 2007: 4th Theory of Cryptography Conference*, volume 4392 of *Lecture Notes in Computer Science*, pages 362–382, Amsterdam, The Netherlands, February 21–24, 2007. Springer Berlin Heidelberg, Germany.

- [ILL24] Yuval Ishai, Hanjun Li, and Huijia Lin. Succinct homomorphic macs from groups and applications. *Cryptology ePrint Archive*, 2024.
- [JLLS23] Aayush Jain, Huijia Lin, Paul Lou, and Amit Sahai. Polynomial-time cryptanalysis of the subspace flooding assumption for post-quantum $i\mathcal{O}$. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023, Part I*, volume 14004 of *Lecture Notes in Computer Science*, pages 205–235, Lyon, France, April 23–27, 2023. Springer, Cham, Switzerland.
- [KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013: 20th Conference on Computer and Communications Security*, pages 669–684, Berlin, Germany, November 4–8, 2013. ACM Press.
- [KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 146–162, Istanbul, Turkey, April 13–17, 2008. Springer Berlin Heidelberg, Germany.
- [KWZ22] Venkata Koppula, Brent Waters, and Mark Zhandry. Adaptive multiparty NIKE. In Eike Kiltz and Vinod Vaikuntanathan, editors, *TCC 2022: 20th Theory of Cryptography Conference, Part II*, volume 13748 of *Lecture Notes in Computer Science*, pages 244–273, Chicago, IL, USA, November 7–10, 2022. Springer, Cham, Switzerland.
- [MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 700–718, Cambridge, UK, April 15–19, 2012. Springer Berlin Heidelberg, Germany.
- [NP01] Moni Naor and Benny Pinkas. Efficient trace and revoke schemes. In Yair Frankel, editor, *FC 2000: 4th International Conference on Financial Cryptography*, volume 1962 of *Lecture Notes in Computer Science*, pages 1–20, Anguilla, British West Indies, February 20–24, 2001. Springer Berlin Heidelberg, Germany.
- [NWZ16] Ryo Nishimaki, Daniel Wichs, and Mark Zhandry. Anonymous traitor tracing: How to embed arbitrary information in a key. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 388–419, Vienna, Austria, May 8–12, 2016. Springer Berlin Heidelberg, Germany.
- [PS18] Chris Peikert and Sina Shiehian. Privately constraining and programming PRFs, the LWE way. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018: 21st International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 10770 of *Lecture Notes in Computer Science*, pages 675–701, Rio de Janeiro, Brazil, March 25–29, 2018. Springer, Cham, Switzerland.

- [Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 84–93, Baltimore, MA, USA, May 22–24, 2005. ACM Press.
- [Ria24] Mahshid Riahinia. *Constrained Pseudorandom Functions: New Constructions and Connections with Secure Computation*. PhD thesis, Ecole normale supérieure de lyon-ENS LYON, 2024.
- [SBC⁺07] Elaine Shi, John Bethencourt, Hubert T.-H. Chan, Dawn Xiaodong Song, and Adrian Perrig. Multi-dimensional range query over encrypted data. In *2007 IEEE Symposium on Security and Privacy*, pages 350–364, Oakland, CA, USA, May 20–23, 2007. IEEE Computer Society Press.
- [Sha84] Adi Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology – CRYPTO’84*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53, Santa Barbara, CA, USA, August 19–23, 1984. Springer Berlin Heidelberg, Germany.
- [SS10] Amit Sahai and Hakan Seyalioglu. Worry-free encryption: functional encryption with public keys. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *ACM CCS 2010: 17th Conference on Computer and Communications Security*, pages 463–472, Chicago, Illinois, USA, October 4–8, 2010. ACM Press.
- [SS24] Sacha Servan-Schreiber. Constrained pseudorandom functions for inner-product predicates from weaker assumptions. In Kai-Min Chung and Yu Sasaki, editors, *Advances in Cryptology – ASIACRYPT 2024, Part II*, volume 15485 of *Lecture Notes in Computer Science*, pages 232–265, Kolkata, India, December 9–13, 2024. Springer, Singapore, Singapore.
- [SSW09] Emily Shen, Elaine Shi, and Brent Waters. Predicate privacy in encryption systems. In Omer Reingold, editor, *TCC 2009: 6th Theory of Cryptography Conference*, volume 5444 of *Lecture Notes in Computer Science*, pages 457–473. Springer Berlin Heidelberg, Germany, March 15–17, 2009.
- [SW05] Amit Sahai and Brent R. Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 457–473, Aarhus, Denmark, May 22–26, 2005. Springer Berlin Heidelberg, Germany.
- [SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 475–484, 2014.
- [SWP00] Dawn Xiaodong Song, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In *2000 IEEE Symposium on Security and Privacy*, pages 44–55, Oakland, CA, USA, May 2000. IEEE Computer Society Press.

- [Tsa19] Rotem Tsabary. Fully secure attribute-based encryption for t-CNF from LWE. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part I*, volume 11692 of *Lecture Notes in Computer Science*, pages 62–85, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Cham, Switzerland.
- [Tsa22] Rotem Tsabary. Candidate witness encryption from lattice techniques. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022, Part I*, volume 13507 of *Lecture Notes in Computer Science*, pages 535–559, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Cham, Switzerland.
- [VWW22] Vinod Vaikuntanathan, Hoeteck Wee, and Daniel Wichs. Witness encryption and null-IO from evasive LWE. In Shweta Agrawal and Dongdai Lin, editors, *Advances in Cryptology – ASIACRYPT 2022, Part I*, volume 13791 of *Lecture Notes in Computer Science*, pages 195–221, Taipei, Taiwan, December 5–9, 2022. Springer, Cham, Switzerland.
- [Wee22] Hoeteck Wee. Optimal broadcast encryption and CP-ABE from evasive lattice assumptions. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022, Part II*, volume 13276 of *Lecture Notes in Computer Science*, pages 217–241, Trondheim, Norway, May 30 – June 3, 2022. Springer, Cham, Switzerland.
- [Wee24] Hoeteck Wee. Circuit ABE with $\text{poly}(\text{depth}, \lambda)$ -sized ciphertexts and keys from lattices. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology – CRYPTO 2024, Part III*, volume 14922 of *Lecture Notes in Computer Science*, pages 178–209, Santa Barbara, CA, USA, August 18–22, 2024. Springer, Cham, Switzerland.
- [Wee25] Hoeteck Wee. Almost optimal KP and CP-ABE for circuits from succinct LWE. In Serge Fehr and Pierre-Alain Fouque, editors, *Advances in Cryptology – EUROCRYPT 2025, Part III*, volume 15603 of *Lecture Notes in Computer Science*, pages 34–62, Madrid, Spain, May 4–8, 2025. Springer, Cham, Switzerland.
- [WWW22] Brent Waters, Hoeteck Wee, and David J. Wu. Multi-authority ABE from lattices without random oracles. In Eike Kiltz and Vinod Vaikuntanathan, editors, *TCC 2022: 20th Theory of Cryptography Conference, Part I*, volume 13747 of *Lecture Notes in Computer Science*, pages 651–679, Chicago, IL, USA, November 7–10, 2022. Springer, Cham, Switzerland.
- [WZ17] Daniel Wichs and Giorgos Zirdelis. Obfuscating compute-and-compare programs under LWE. In Chris Umans, editor, *58th Annual Symposium on Foundations of Computer Science*, pages 600–611, Berkeley, CA, USA, October 15–17, 2017. IEEE Computer Society Press.
- [Yan23] Rupeng Yang. Privately puncturing PRFs from lattices: Adaptive security and collusion resistant pseudorandomness. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023, Part III*, volume 14006 of *Lecture Notes in Computer Science*, pages 163–193, Lyon, France, April 23–27, 2023. Springer, Cham, Switzerland.

A Matrix Evaluation of Fully Homomorphic Encryption

Let us first recall the [GSW13] fully homomorphic encryption scheme which consists of algorithms (Setup, Enc, Dec, Eval):

Setup(1^λ) : It chooses $n = \text{poly}(\lambda)$ for some polynomial $\text{poly}(\cdot)$, and $m = O(n \log q)$. The setup algorithm samples $\mathbf{B} \leftarrow \mathbb{Z}_q^{(n-1) \times m}$, $\mathbf{s} \leftarrow \mathbb{Z}_q^{n-1}$, $\mathbf{e} \leftarrow \chi^m$ for some error distribution χ . It outputs public key $\text{pk} := \begin{bmatrix} \mathbf{B} \\ \mathbf{s}^T \mathbf{B} + \mathbf{e}^T \end{bmatrix}$, and secret key $\text{sk} := \mathbf{s}$.

Enc(pk, b) : It takes as input pk and a bit $b \in \{0, 1\}$ to encrypt. Let $\mathbf{G}$ be the gadget matrix of dimension $n \times m$. It samples $\mathbf{R} \leftarrow \{0, 1\}^{m \times m}$ and outputs $\begin{bmatrix} \mathbf{B} \\ \mathbf{s}^T \mathbf{B} + \mathbf{e}^T \end{bmatrix} \cdot \mathbf{R} + b \cdot \mathbf{G}$.

Eval($f, \text{ct}_1, \dots, \text{ct}_\ell$) : It takes as input f which is a boolean function that takes of ℓ bits of input. It homomorphically evaluates function f over ciphertext $\text{ct}_1, \dots, \text{ct}_\ell$ and outputs ciphertext ct .

Dec(sk, ct) : The correctly encrypted or evaluated ciphertext is of the form $\begin{bmatrix} \mathbf{B} \\ \mathbf{s}^T \mathbf{B} + \mathbf{e}^T \end{bmatrix} \cdot \mathbf{R} + b \cdot \mathbf{G}$ for some low norm matrix $\mathbf{R}$ and bit b is the message that it encrypts. Given that

$$[\mathbf{s}^T] - 1 \cdot \left(\begin{bmatrix} \mathbf{B} \\ \mathbf{s}^T \mathbf{B} + \mathbf{e}^T \end{bmatrix} \cdot \mathbf{R} + b \cdot \mathbf{G} \right) \approx b \cdot [\mathbf{s}^T] - 1 \cdot \mathbf{G},$$

it computes $[\mathbf{s}^T] - 1 \cdot \text{ct}$ and outputs 0 if it has low norm and 1 otherwise.

The above design satisfies semantic security. The completeness property says that given correctly computed public/secret key $\text{pk}, \text{sk} = \mathbf{s}$, $\text{ct}_i = \text{Enc}(\text{pk}, x_i)$ for $i \in [\ell]$, binary function $f : x_1, \dots, x_\ell \rightarrow \{0, 1\}$, let ct be the evaluated ciphertext by $\text{Eval}(f, \text{ct}_1, \dots, \text{ct}_\ell)$, then it satisfies that $[\mathbf{s}^T] - 1 \cdot \text{ct} \approx y \cdot \mathbf{G}$, where $y = f(x_1, \dots, x_\ell)$. Furthermore, by [GSW13], let β be the error bound of $\mathbf{e}$ such that $\|\mathbf{e}\|_\infty \leq \beta$, and let d be the depth of boolean circuit f , it holds that

$$\|[\mathbf{s}^T] - 1 \cdot \text{ct} - y \cdot [\mathbf{s}^T] - 1 \cdot \mathbf{G}\|_\infty \leq \beta \cdot m^{O(d)}.$$

One could extend the above binary function into a matrix function. Consider function $f : x_1, \dots, x_k \rightarrow \mathbf{X}_f$, such that f outputs a matrix $\mathbf{X}_f \in \mathbb{Z}_q^{n \times m}$. Then, we define the function **MtrxEval**:

MtrxEval($f, \text{ct}_1, \dots, \text{ct}_k$): Let $f_{t,\ell} : x_1, \dots, x_k \rightarrow \{0, 1\}$ denote the function that outputs the ℓ -th bit of the entry t of $\mathbf{X}_f$. Let $\mathbf{E}_{t,\ell}$ be the matrix such that it has the same dimension as $\mathbf{X}_f$, has all-zero entries except that the for entry t is $2^{\ell-1}$. **MtrxEval** computes $\text{ct}_{t,\ell} = \text{Eval}(f_{t,\ell}, x_1, \dots, x_k)$. Next, it outputs

$$\text{ct} = \sum_{t,\ell} \text{ct}_{t,\ell} \cdot \mathbf{G}^{-1}(\mathbf{E}_{t,\ell}).$$

Then, we have $[\mathbf{s}^T] - 1 \cdot \text{ct} \approx [\mathbf{s}^T] - 1 \cdot \mathbf{X}_f$, such that for $m = \Omega(n, \log q)$,

$$\|[\mathbf{s}^T] - 1 \cdot \text{ct} - [\mathbf{s}^T] - 1 \cdot \mathbf{X}_f\|_\infty \leq \beta \cdot m^{O(d)}.$$