Pseudorandom FE and iO with Applications

Shweta Agrawal[*] Simran Kumari[†] Shota Yamada[‡]

Abstract

We propose the abstractions of Functional Encryption (FE) and Indistinguishability Obfuscation (iO) for *pseudorandom* functionalities which are strictly weaker than their general counterparts. Intuitively, a pseudorandom functionality means that the output of the circuit is indistinguishable from uniform for *every* input seen by the adversary. We then leverage weak indistinguishability style security of these tools to obtain the following applications:

1. *Attribute Based Encryption for Unbounded Depth Circuits.* Assuming IND-secure FE for pseudorandom functionalities and LWE, we construct Attribute Based Encryption (ABE) for circuits of unbounded depth. Previously, such ABE required the circular Evasive LWE assumption (Hseih, Lin and Luo, Focs 2023) which has recently been subject to zeroizing attacks.
2. *Attribute Based Encryption for Turing Machines.* Assuming IND-secure FE for pseudorandom functionalities and circular small-secret LWE, we construct Attribute Based Encryption (ABE) for Turing machines. Previously, such ABE required either private coin Evasive LWE (Agrawal, Kumari and Yamada, Crypto 2024) or circular Evasive LWE (Cini and Wee, Eurocrypt 2025), both of which admit attacks in the general case.
3. *Multi Input Predicate Encryption for Polynomial Arity.* Assuming IND-secure multi-input FE for pseudorandom functionalities, we construct Multi Input Predicate Encryption (MIPE) for P for polynomial arity. Previously, MIPE for P was known only for *constant* arity, using private coin evasive LWE (Agrawal, Rossi, Yadav and Yamada, Crypto 2023).
4. *Instantiating the Random Oracle.* We use our IND-secure iO for pseudorandom functionalities to instantiate the random oracle in several applications that previously used iO (Hohenberger, Sahai and Waters, Eurocrypt 2014) such as full-domain hash signature based on trapdoor permutations and more.

We provide heuristic constructions of FE and MIFE for pseudorandom functionalities from private coin evasive LWE and plain LWE, where private coin evasive LWE is suitably parametrized to avoid all know attacks for the functionalities we consider in this work. This implies iO for pseudorandom functionalities from the same assumptions.

[*]IIT Madras, India, `shweta@cse.iitm.ac.in`
[†]NTT Social Informatics Laboratories, Tokyo, Japan, `simran.kumari@ntt.com`
[‡]AIST Tokyo, Japan, `yamada-shota@aist.go.jp`

Contents

1 Introduction

Attribute based encryption. Attribute based encryption (ABE) is a generalization of public key encryption which enables fine grained access control on encrypted data. In ABE, the ciphertext encodes a public *attribute* $\mathbf{x}$ together with a secret message m, the secret key is generated for a public function f, and decryption outputs m if and only if $f(\mathbf{x}) = 1$. Security is formalized in an indistinguishability style game which asks that an adversary should be unable to distinguish between an encryption of $(m_0, \mathbf{x})$ and $(m_1, \mathbf{x})$, even given secret keys for functions f_i so long as $f_i(\mathbf{x}) = 0$ for all i. ABE comes in two avatars – "key-policy" where the function f is encoded in the secret key, or "ciphertext-policy" where it is encoded in the ciphertext. These are denoted by kpABE and cpABE respectively. An interesting strengthening of ABE is the so-called "Predicate Encryption" (PE) [KSW08] where the attribute $\mathbf{x}$ is also hidden but only against an adversary that does not receive any decrypting key, namely $f_i(\mathbf{x}) = 0$ for all f_i queried by the adversary.

There has been significant progress in constructing ABE for circuits over the last several years [GPSW06, GVW13, BGG+14, AY20, Wee22, HLL23] from well-understood assumptions. However, for circuits of *unbounded depth*, the only known solutions rely on full fledged compact FE [JLL23] or a new assumption called *circular evasive LWE* [HLL23]. The elegant construction by [HLL23] (HLL) provides a pathway to ABE for unbounded depth from plausibly weaker assumptions than required for full fledged FE in the lattice regime, but their technique is very specific to algebraic structure of the underlying building blocks and does not lend itself to generalization. Moreover, the circular evasive LWE assumption has recently been shown to have attacks [AMYY25]. Thus, it is important to ask:

Can we construct ABE for unbounded depth circuits from a broader class of assumptions?

ABE for Turing Machines. Another important line of ABE research is to support uniform models of computation [Wat12, GKP+13, AS16, AM18, KNTY19, GWW19, GW20, AMY19a, AMY19b, LL20, AKY24, CW25]. This is because, though powerful, circuits force the size of the input to be fixed ahead of time and also incur *worst case* running time on every input. However, so far the only solutions we have rely on private coin evasive or circular evasive LWE [AKY24, CW25], which have recently been shown to have counter-examples [HJL25, AMYY25, DJM+25, BÜW24]. Again, we ask:

Can we construct ABE for Turing machines from a broader class of assumptions?

Multi-Input Setting. A multi-input predicate encryption (miPE) scheme [AYY22] for n-ary functions $\mathsf{miPE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}_1, \ldots, \mathsf{Enc}_n, \mathsf{Dec})$ generalizes single input predicate encryption [GVW15b] to support multiple encryptors, who each encrypt their data with independently chosen randomness. In miPE, the setup algorithm on input 1^λ, arity 1^n and parameter prm, specifying the parameters of the function class, outputs $(\mathsf{mpk}, \mathsf{msk})$. The key generation algorithm on input msk and a function $f : (\mathcal{X}_{\mathsf{prm}})^n \to \mathcal{Y}_{\mathsf{prm}}$ outputs a functional secret key sk_f. The i-th encryption algorithm on input msk, an attribute $x_i \in \mathcal{X}_{\mathsf{prm}}$ and a message $\mu_i \in \{0, 1\}$ outputs a ciphertext ct_i. The decryption algorithm on input secret key sk_f and n ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_n$ (corresponding to inputs $(x_1, \mu_1) \ldots, (x_n, \mu_n)$ respectively) outputs a string $\mu' \in \{0, 1\}^n \cup \bot$.

Previously, the works of [AYY22, FFMV24] provided the first constructions of multi-input predicate encryption for specific functionalities. The follow-up work of Agrawal et al. [ARYY23] supported the most general functionality – it allowed to compute arbitrary predicates in P on vector $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ where $\mathbf{x}_i$ is encrypted by party $i \in [n]$ and k is a constant. However, the limitation for a constant k seems inherent to their techniques, and they also rely on private coin evasive LWE and a strengthening of Tensor LWE in addition to LWE. Thus, for polynomial arity and function class P, we do not have any constructions of MIPE, even from the evasive LWE family of assumptions. This leads to the question:

Can we construct MIPE for P *supporting polynomial arity (without using iO)?*

In this work, we answer the above questions in the affirmative by leveraging a general new tool – FE for *pseudorandom* functionalities – which we describe next.

1.1 New Tool: Functional Encryption for Pseudorandom Functionalities

We begin by recalling the standard notions of Functional Encryption and Indistinguishability Obfuscation.

FE, MIFE and iO. Functional encryption (FE) [SW05, BSW11] generalizes ABE – in FE, a ciphertext is associated with a vector $\mathbf{x}$, a secret key is associated with a circuit f and decryption enables recovery of $f(\mathbf{x})$ and nothing else. Goldwasser et al. [GGG$^+$14] generalized FE to the multi-input setting – in MIFE, multiple parties can independently encrypt their data and the key generator can provide a function key that jointly decrypts all the ciphertexts. In more detail, now we have n parties, each of who independently computes the ciphertext for its data $\mathbf{x}_i$, for $i \in [n]$, the key generator provides a key for an n-ary function f and decryption allows to recover $f(\mathbf{x}_1, \ldots, \mathbf{x}_n)$. As discussed in the original work of MIFE, the notion is more meaningful in the symmetric rather than public key setting, since the latter allows for too much leakage on the challenge message by dint of legitimate combinations with messages chosen by the adversary. In two concurrent, influential works [AJ15, BV18] it was shown that single input FE, if it supports sufficiently expressive functionality and satisfies an efficiency property called *compactness*, is enough to generically imply multi-input FE. Moreover, multi-input FE is shown [AJ15, BV18] to imply the powerful notion of Indistinguishability Obfuscation (iO) [BGI$^+$01], which seeks to garble circuits while preserving their input-output behaviour. In more detail, given a circuit C, an obfuscation $\tilde{C}$ preserves the correctness of C so that $C(\mathbf{x}) = \tilde{C}(\mathbf{x})$ for *every* input $\mathbf{x}$, but hides everything else about C.

Following a long line of work [JLMS19, JLS21, Agr19, APM20, WW21, GP21, DQV$^+$21], FE was constructed using standard assumptions in the breakthrough work of Jain, Lin and Sahai [JLS21] and improved by [JLS22, RVV24]. However, all these works rely quite crucially on pairings which is dissatisfying. In the realm of conjectured quantum safety, there exist several candidates from lattices but their security is based on non-standard assumptions, many of which have been broken [Agr19, APM20, WW21, GP21, DQV$^+$21, HJL21, AJS23, HJL25].

Evasive LWE. Evasive LWE is a popular new lattice assumption, introduced by Wee and (independently) Tsabary [Wee22, Tsa22]. At a very high level, evasive LWE can be seen as a lattice analog of the generic group model, which is popular in the pairings world, in that it restricts the class of attacks that an adversary can mount. Below, we let $\underline{X}$ denote a noisy version of X where the exact value of noise is not important and $\mathbf{B}^{-1}(\mathbf{P})$ denote a short preimage $\mathbf{K}$ (say) such that $\mathbf{BK} = \mathbf{P} \bmod q$. The evasive LWE assumption roughly says that if

$$\left(\mathbf{A},\ \mathbf{B},\ \mathbf{P},\ \underline{\mathbf{s}^\mathsf{T}\mathbf{A}},\ \ \underline{\mathbf{s}^\mathsf{T}\mathbf{B}},\ \ \underline{\mathbf{s}^\mathsf{T}\mathbf{P}},\ \ \mathsf{aux}\ \right) \approx_c \left(\mathbf{A},\ \mathbf{B},\ \mathbf{P},\ \$,\ \ \$,\ \ \$,\ \ \mathsf{aux}\right)$$

where $\mathbf{A}, \mathbf{B}, \mathbf{P}$ are matrices of appropriate dimensions, $\mathbf{s}$ is a secret vector, aux is some auxiliary information and \$ represents random, then

$$\left(\mathbf{A},\ \mathbf{B},\ \mathbf{P},\ \underline{\mathbf{s}^\mathsf{T}\mathbf{A}},\ \ \underline{\mathbf{s}^\mathsf{T}\mathbf{B}},\ \ \mathbf{B}^{-1}(\mathbf{P}),\ \ \mathsf{aux}\right) \approx_c \left(\mathbf{A},\ \mathbf{B},\ \mathbf{P},\ \$,\ \ \$,\ \ \mathbf{B}^{-1}(\mathbf{P}),\ \ \mathsf{aux}\right)$$

Evasive LWE has proven to be a very meaningful strengthening of LWE in that it has provided several strong new applications that had been elusive from plain LWE, despite significant research effort over decades – optimal broadcast encryption [Wee22], witness encryption [VWW22], multi-input attribute based encryption [ARYY23], optimal broadcast and trace [AKYY23], attribute based encryption (ABE) for unbounded depth circuits [HLL23] and ABE for Turing machines [AKY24], to name a few.

Evasive LWE has been studied in two main regimes, namely "public-coin" and "private-coin", where the former means that the randomness used to sample $\mathbf{P}$ and auxiliary information aux, is made available to the adversary, and the latter means that this information needs to be hidden. The original work of [VWW22] which defined private coin evasive LWE showed contrived counterexamples against the assumption (later formalized and improved in [BÜW24, HHY25]). However, this was not considered too problematic as it relied on highly unnatural auxiliary information which contained obfuscations that would output secrets given $\mathbf{B}^{-1}(\mathbf{P})$ but not otherwise. No attacks were known in the public-coin setting used by Wee's original formulation or its extensions, such as the circular evasive LWE by Hseih, Lin and Luo [HLL23]. Thus, evasive LWE has been seen as a meaningful "middle point" in the land between LWE on one hand, and lattice assumptions used for iO on the other.

Recent Developments. Recently, there have been several new counter-examples against private coin Evasive LWE [BÜW24, BDJ+25, AMYY25, HHY25, DJM+25], which broaden the class of attacks. Moreover, in some cases, there are also attacks in the public coin setting – for instance, the circular evasive LWE assumption which has been considered to be in the public coin [HLL23, CW25] category is now known to broken for arbitrary samplers [AMYY25]. While it is possible to avoid all the counter-examples by suitably restricting the sampler, these attacks have severely shaken the faith of the community on this family of assumptions. Please see Section 1.3 for a detailed discussion.

1.2 Our Results

In this work, we develop a new tool and leverage it in a simple, black box way, to yield new constructions for ABE for unbounded depth circuits, ABE for Turing machines and Multi-Input ABE schemes, from weaker assumptions. We summarize our results below.

1.2.1 New Tools: FE, MIFE and iO for Pseudorandom Functionalities.

We introduce the tools of FE, MIFE and iO for *pseudorandom* functionalities, denoted by prFE, prMIFE, prIO respectively, which are strictly weaker than their counterparts for general circuits. Intuitively, a pseudorandom functionality means that the output of the circuit is indistinguishable from uniform for *every* input seen by the adversary. We define both simulation style as well as indistinguishability style security for our new tools.

Useful Abstraction. We believe these tools provide a clean and useful new abstraction on which to base applications. These tools can already be instantiated using standard assumptions by using constructions of full-fledged compact FE, MIFE and iO [JLS21, JLS22, RVV24]. Additionally, we show that they can be constructed using a suitably restrained version of private coin Evasive LWE, which avoids all known attacks. Note that similar restrictions are required to recover *every* application of private coin evasive or circular evasive LWE to the best of our understanding. Also note that private coin Evasive LWE is currently required even for the weaker "all or nothing" primitive of Witness Encryption in the lattice regime [VWW22] – we show that a similar assumption suffices to build a nontrivial compact Functional Encryption. This significantly expands the capabilities of private coin Evasive LWE.

It is currently unclear how the restricted evasive LWE assumption used here compares with lattice assumptions that have been used to construct iO – we hope that weaker assumptions than both evasive LWE as well as those used for lattice based iO can eventually be used to construct our tools. However, regardless of instantiations, our new tools can be used in a clean, black-box way for applications which previously required full-fledged FE/iO and we are optimistic they will find further applications.

Constructions from (suitably restricted) Evasive LWE. Our constructions are summarized in the following theorems.

Theorem 1.1 (Compact prFE). Assuming LWE and (suitably parametrized) private coin evasive LWE, there exists a secure prFE scheme for function class $\mathcal{F}_{\mathsf{L}(\lambda),\ell(\lambda),\mathsf{dep}(\lambda)} = \{f : \{0,1\}^{\mathsf{L}} \to \{0,1\}^{\ell}\}$ satisfying

$$|\mathsf{mpk}| = \mathsf{L} \cdot \mathrm{poly}(\mathsf{dep}, \lambda), \quad |\mathsf{sk}_f| = \ell \cdot \mathrm{poly}(\mathsf{dep}, \lambda), \quad |\mathsf{ct}| = \mathsf{L} \cdot \mathrm{poly}(\mathsf{dep}, \lambda)$$

where $\mathsf{dep} = \mathrm{poly}(\lambda)$ is the depth bound on the functions supported by the scheme.

We show how to compile a bounded-depth prFE scheme into an unbounded depth scheme but with slightly worse parameters.

Theorem 1.2 (Compact Unbounded-Depth prFE.). Assuming LWE and (suitably parametrized) private coin evasive LWE, there exists a secure prFE scheme for function class $\mathcal{F} = \{f : \{0,1\}^{\mathsf{L}} \to \{0,1\}\}$ of unbounded depth satisfying

$$|\mathsf{mpk}| = \mathsf{L} \cdot \mathrm{poly}(\lambda), \quad |\mathsf{sk}_f| = \mathsf{L} \cdot |f| \cdot \mathrm{poly}(\lambda), \quad |\mathsf{ct}| = \mathsf{L} \cdot \mathrm{poly}(\lambda).$$

We then show how to bootstrap our single input prFE to support multiple inputs. We denote this primitive by prMIFE. Our construction follows the template of [AJ15, BV18], although the proof is significantly different.

Theorem 1.3 (prMIFE **for polynomial arity).** Assume (suitably parametrized) evasive LWE, non-uniform sub-exponential PRF, and non-uniform sub-exponential LWE. Then there exists a prMIFE scheme for arity $n = \mathrm{poly}(\lambda)$, supporting functions with bounded polynomial depth.

We bootstrap our prMIFE to obtain the first iO for pseudorandom functionalities, similar to [GGG+14]. We denote this by prIO. In more detail:

Theorem 1.4 (prIO). Assuming IND-secure prMIFE, there exists a prIO scheme for all polynomial sized circuits.

In light of the new attacks, we treat our constructions as lattice based heuristics, and from these heuristics, we demand only a weak indistinguishability style security [BSW11] which is known to be achievable even for general functions [JLS21]. We justify the heuristic in more detail in Section 2.2.

Comparison with Circular Evasive LWE. Recall that private coin Evasive LWE is a large family of assumptions, parametrized by the choice of matrices, errors, auxiliary information and such. Therefore, we compare our specific version of private coin Evasive LWE with circular evasive LWE, on which the only previous ABE for unbounded depth has been based (from purely lattice assumptions) and which has been widely categorized as a "public-coin" assumption (please see [HLL23] as well as follow-ups, such as [BDJ+25, CW25]).

At a very high level, both assumptions essentially provide an FHE encoding on top of the usual terms provided in Evasive LWE (please see Section 2 for details). The essential difference between circular evasive LWE and the version of evasive LWE used in the present work is that in the former, the value to be hidden within the FHE encoding is the FHE secret key $\mathbf{s}$ (resulting in circularity), while in the latter, it is an arbitrary string $\mathbf{x}$ chosen by the sampler. Moreover, in both constructions, the FHE secret is chosen to be the same as the LWE secret $\mathbf{s}$ in the assumption. Hence, in HLL, $\mathbf{s}$ can be chosen *outside* the sampler, and the FHE encoding is provided as part of the problem instance not as part of the sampler's output. This results in the assumption getting categorized as public-coin (since $\mathbf{s}$ is not known to the sampler). In our work, on the other hand, we assume that $\mathbf{x}$ is chosen by the sampler. Since $\mathbf{x}$ must be hidden inside the FHE encoding, we must have this encoding be output by the sampler itself – this results in our assumption being categorized as private coin. In both cases, there is secret randomness used in generating the FHE encoding which must be kept private. The work of [AMYY25] shows that both assumptions are subject to a very similar attack, due to the presence of the FHE encoding.

1.2.2 Applications of prFE

We then proceed to use our new tool in simple, black box ways to achieve the following results.

Key Policy ABE. For key-policy ABE supporting circuits of unbounded depth, we obtain the following results.

Theorem 1.5 (Unbounded kpABE**, No Circularity).** Assuming LWE and IND-secure prFE (Definition 4.4), there exists a very selectively secure kpABE scheme for circuits of unbounded depth and attribute length ℓ with

$$|\mathsf{mpk}| = \ell \cdot \mathrm{poly}(\lambda), \quad |\mathsf{sk}_C| = |C| \cdot \ell \cdot \mathrm{poly}(\lambda), \quad |\mathsf{ct}| = \ell \cdot \mathrm{poly}(\lambda).$$

By relying on circular small-secret LWE instead of plain LWE, we can improve the parameters as below.

Theorem 1.6. Assuming circular small-secret LWE and IND-secure prFE (Definition 4.4), there exists a very selectively secure kpABE scheme for circuits of unbounded depth and attribute length ℓ with

$$|\mathsf{mpk}| = \mathrm{poly}(\ell, \lambda), \quad |\mathsf{sk}_C| = \mathrm{poly}(\lambda), \quad |\mathsf{ct}| = \mathrm{poly}(\ell, \lambda).$$

Previously, the only other unbounded depth KP-ABE scheme by [HLL23] achieved the same parameters as Theorem 1.6 but relied on the circular evasive LWE assumption, which is now known to have attacks [AMYY25]. Moreover, in Theorem 1.5, we show that circularity can be dispensed with altogether at the cost of worse parameters.

Ciphertext Policy ABE. For ciphertext policy ABE we obtain the following result.

Theorem 1.7. Assuming circular small-secret LWE and IND-secure prFE (Definition 4.4), there exists a very selectively secure cpABE scheme for circuits $\{C : \{0,1\}^\ell \to \{0,1\}\}$ of unbounded depth with

$$|\mathsf{cpABE.mpk}| = \mathrm{poly}(\lambda), \quad |\mathsf{cpABE.sk}_{\mathbf{x}}| = \mathrm{poly}(\ell, \lambda), \quad |\mathsf{cpABE.ct}_C| = \mathrm{poly}(\lambda).$$

Previously, AKY used LWE, private-coin Evasive LWE and circular tensor LWE assumption to construct unbounded depth cpABE. We note that the cpABE scheme instantiated as above replaces the reliance on circular tensor LWE and LWE assumptions used by AKY by simply circular small-secret LWE. It also replaces the private coin evasive LWE used in AKY by IND-secure prFE. We note that this cpABE has a shorter mpk as compared to that of AKY (which is $\mathrm{poly}(\lambda, \ell)$), other parameters being the same.

ABE for Turing Machines. AKY provided a compiler that uses kpABE for bounded depth circuits and cpABE for unbounded depth circuits to achieve kpABE for Turing machines. Plugging our new cpABE into this compiler, we obtain:

Corollary 1.8. Assuming circular small-secret LWE and IND-secure prFE (Definition 4.4), there exists a very selectively secure ABE for TM with

$$|\mathsf{mpk}| = \mathrm{poly}(\lambda), \quad |\mathsf{sk}| = \mathrm{poly}(\lambda, |M|), \quad |\mathsf{ct}| = \mathrm{poly}(\lambda, |\mathbf{x}|, t).$$

[AKY24] uses LWE assumption, private coin evasive LWE assumption and circular tensor assumption for their construction with the same parameters as in Corollary 1.8.

The concurrent work by [CW25] achieves ABE for Turing Machines assuming LWE and circular evasive LWE assumption and better parameters as compared to AKY or Corollary 1.8. Concretely AKY and Corollary 1.8 achieve $|\mathsf{mpk}| = O(1), |\mathsf{ct}| = O(|M|^2)$, $\mathsf{sk} = O(|\mathbf{x}| \cdot t)$ and [CW25] achieves $|\mathsf{mpk}| = O(1), |\mathsf{ct}| = O(|M|)$, $\mathsf{sk} = O(t)$ where $O(\cdot)$ hides $\mathrm{poly}(\lambda)$ factors [1].

1.2.3 Applications of prMIFE and prIO

We use our prMIFE to construct Multi-Input Predicate Encryption (miPE) as below.

Theorem 1.9 (miPE for poly arity). Assuming non-uniform IND-secure prMIFE(Definition 7.4) , subexponentially secure PRF, and sub-exponential LWE, there exists a miPE scheme for polynomial arity, supporting functions of bounded polynomial depth, and satisfying security as per Definition 3.25.

In the special case of constant arity, we can base security of the scheme on weaker assumptions – it suffices to assume polynomial-time security of PRF and LWE.

Previously, the works of [AYY22, FFMV24] defined the notion of multi-input predicate encryption and provided the first constructions for specific functionalities. The follow-up work of Agrawal et al. [ARYY23] supported the most general functionality – it allowed to compute arbitrary predicates in P on vector $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ where $\mathbf{x}_i$ is encrypted by party $i \in [n]$ and k is a constant. Next, we provide some applications of our new tool of prIO.

Instantiating the Random Oracle. Hohenberger, Sahai and Waters [HSW14] used iO to instantiate the random oracle in several applications. In more detail, they showed selective security of the full-domain hash (FDH) signature based on trapdoor permutations (TDP) [BR93], the adaptive security of RSA FDH signatures [Cor00], the selective security of BLS signatures, and the adaptive security of BLS signatures [BLS01] in the standard model. Our prIO can be used to instantiate all these applications.

Theorem 1.10 (Full-Domain Hash Signatures). Assuming IND-secure prIO, sub-exponential secure punctured PRF, and one-way TDP, there exists a selectively secure full-domain hash signatures.

[1] CW25 claims AKY key size to be $O(t^2)$ however it is $O(|\mathbf{x}| \cdot t)$, to the best of our knowledge.

1.3 Recent Attacks on Evasive LWE and Repercussions.

The original work of [VWW22], later formalized by the follow-up [BÜW24, HHY25], showed that for contrived auxiliary information, the private coin evasive LWE assumption is false – all prior work using this assumption avoids this attack by suitably curtailing the auxiliary information. Subsequent to the first online appearance of the present work, there were two important developments that affected our results:

1. *Impossibility of simulation secure PRFE for general circuits.* The concurrent work of [BDJ+25] and the follow-up work of [AMYY25] show that there exists a contrived "self-referential" functionality for which pseudorandom functional encryption or pseudorandom obfuscation satisfying strong simulation style security (called "pseudorandom CT" or prCT security) cannot exist.

2. *Attacks against Evasive and Circular Evasive LWE.* New counter-examples for evasive LWE were developed [BDJ+25, BÜW24, AMYY25, HJL25, DJM+25, HHY25]. Some of these attacks show that the intuition that evasive and circular evasive LWE evade the zeroizing regime is not always true. Of these, the most important attacks as related to our work, are presented in [AMYY25, HJL25] who showed (via essentially the same attack) that by carefully crafting a contrived circuit to implement a PRF which is used (non black-box) in the construction of prFE in the previous version of this work (as well as the construction of prIO in the concurrent work of [BDJ+25]), the attacker can obtain problematic leakage.

The present manuscript takes the following steps to address the situation.

Impossibility of simulation secure PRFE for general circuits. Impossibility results are ubiquitous in cryptography – similar impossibilities are known, for instance, for the random oracle model [CGH04], virtual black box (VBB) obfuscation [BGI+01] and simulation secure FE [BSW11]. The community has addressed these impossibilities either by weakening the security definition or by weakening the functionality.

In general there is no win-win situation around impossibilities – weakening the security definition has the disadvantage that it may admit schemes that are intuitively insecure, as in the case of IND-secure FE [BSW11]. Weakening the functionality has the disadvantage that it is difficult to characterize which functionalities are "safe" and forces one to essentially assume that the attacks do not apply to some "natural" subset. Nevertheless, despite impossibilities known for ROM and VBB obfuscation [CGH04, BGI+01], the meaningfulness of ROM for practical security, and of VBB obfuscation for restricted functionalities [Wee05, CRV10] is accepted widely.

In our setting, both remedies can apply. We have weakened the security definition of prFE, prMIFE and prIO to the IND based versions which is the usual notion of security considered for these primitives and admit constructions from standard assumptions, even for general circuits. Thus, all our applications now rely on IND secure building blocks, which are known to be instantiable via standard assumptions. We also observe that the pseudorandom functionalities that are useful for our applications, such as blind garbled circuits, are quite natural and do not fall prey to known attacks, even for the stronger prCT notion of security.

Attacks against Evasive and Circular Evasive LWE. To handle these attacks, we have restricted the evasive LWE assumption so that it avoids all known counter-examples. To the best of our understanding, similar restrictions need to be placed on all schemes based on circular evasive (including [HLL23, CW25]) or private coin evasive LWE (including [BDJ+25]).

The constructions of prFE, prMIFE and prIO are now viewed as a heuristic satisfying IND security. To further strengthen our heuristic, we modified the construction in the original version of this work (see Appendix A for a high level summary and Section 4 for a formal description)so that the attacks no longer apply, even for a contrived circuit implementation of the PRF. Similar modifications would be required to make the scheme by [BDJ+25] secure against these attacks. We also mention the work by [BÜW24] which presents attacks against classes of evasive LWE such that either $\mathbf{B}$ or $\mathbf{P}$ are not known to the adversary. In our case (even in the original version), both $\mathbf{B}$ and $\mathbf{P}$ are known to the adversary, hence these attacks do not apply.

1.4 Concurrent Work

Pseudorandom Obfuscation. The concurrent work of Branco et al. [BDJ+25] also considered obfuscation for pseudorandom functions and constructed it using private coin evasive LWE, similar to us. Their construction is also subject to the attacks by [HJL25, AMYY25] similarly to the previous version of the present work, if the circuit implementation of the building blocks is allowed to be chosen adversarially. They do not study FE or MIFE for pseudorandom functionalities which is the main focus of our work. Note that while FE implies iO, this is with exponential loss in the reduction, and a large body of work has focused on replacing the usage of iO by FE in applications – please see [GPSZ17] for a discussion. The applications developed in the two works are different.

Additionally, as discussed above, Branco et al. [BDJ+25] also provided an important lower bound by showing that simulation style security for pseudorandom obfuscation is impossible for general functionalities. As mentioned previously, following this impossibility, we have weakened our security to IND based versions – thus, this modification was made subsequent to [BDJ+25]. However, the techniques used to convert the proofs using our original definition to the IND based version are standard and well known in the FE literature (starting with [DCIJ+13]).

ABE for Turing Machines. The concurrent work by Cini and Wee [CW25] achieves ABE for Turing Machines assuming LWE and circular evasive LWE assumption. The previous version of the present work achieved ABE for Turing Machines assuming LWE and private coin evasive LWE. To compare these: (i) [CW25] achieved better parameters, (ii) we relied on private coin evasive LWE assumption without circularity. It was previously believed that circular evasive LWE falls in the public coin category of evasive LWE assumptions, but the recent work of [AMYY25] displays zeroizing attacks even against circular evasive LWE, calling this belief into question. The present version of our work now weakens the requirement of private coin evasive LWE to IND secure prFE.

Comparison with Succinct LWE. We note that the recently introduced succinct LWE assumption [Wee24, Wee25] is a falsifiable variant of LWE that has been used for parameter optimizations in ABE schemes [Wee24, Wee25], distributed broadcast encryption [CW24] and such other applications. It is not currently known to imply stronger primitives such as unbounded ABE or witness encryption, to the best of our knowledge.

2 Technical Overview

In this section, we present the core ideas that we develop in this work. Below $\underline{X}$ denotes a noisy version of X where the exact value of noise is not important.

2.1 Preparations

Evaluation Algorithms by Boneh et al. (BGG+)**.** The seminal work of Boneh et al. [BGG+14] developed algorithms for evaluating arithmetic functions on the ciphertext as well as the public key of an ABE scheme, which form the cornerstone of several subsequent constructions. Their core technique is as follows: given an input $\mathbf{x} \in \{0,1\}^\ell$, and a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times \ell m}$, one can homomorphically evaluate a circuit $f : \{0,1\}^\ell \to \{0,1\}$ on an "input encoding" matrix of form $\mathbf{A} - \mathbf{x} \otimes \mathbf{G}$ by multiplying on the right by a low norm matrix $\mathbf{H}_{\mathbf{A},f,\mathbf{x}}$ to obtain the term $\mathbf{A}_f - f(x)\mathbf{G}$. Here, $\mathbf{G}$ is a special gadget matrix defined as follows. Let $\mathbf{g} = [1, 2, 2^2, \ldots, 2^{\log q}]^\intercal$ and $\mathbf{G} = \mathbf{I} \otimes \mathbf{g}^\intercal$. In key evaluation, one can homomorphically evaluate a circuit $f : \{0,1\}^\ell \to \{0,1\}$ on $\mathbf{A}$ to obtain $\mathbf{A}_f = \mathbf{A} \cdot \mathbf{H}_{\mathbf{A},f}$ for some low norm matrix $\mathbf{H}_{\mathbf{A},f}$. In ciphertext evaluation, given an attribute $\mathbf{x}$ and corresponding attribute encoding of the form $\underline{\mathbf{s}^\intercal(\mathbf{A} - \mathbf{x} \otimes \mathbf{G})}$, which we refer to as BGG+ encoding, right multiplication by $\mathbf{H}_{\mathbf{A},f,\mathbf{x}}$ yields $\underline{\mathbf{s}^\intercal(\mathbf{A}_f - f(\mathbf{x})\mathbf{G})}$ without substantially blowing up the noise in the encoding since $\mathbf{H}_{\mathbf{A},f,\mathbf{x}}$ is low norm. Skipping several details, since the key generator can compute $\mathbf{A}_f = \mathbf{A} \cdot \mathbf{H}_{\mathbf{A},f}$, it can provide a matching key which allows the decryptor to cancel out the masking term $\mathbf{s}^\intercal \mathbf{A}_f$ and proceed with decryption. We refer to $\mathbf{H}_{\mathbf{A},f}$ and $\mathbf{H}_{\mathbf{A},f,\mathbf{x}}$ as the PK and CT evaluation matrices respectively.

Handling Unbounded Depth by Hseih, Lin and Lu (HLL)**.** The essential barrier in supporting circuits of unbounded depth for homomorphic computation is that the norm of the matrix $\mathbf{H}_{\mathbf{A},f,\mathbf{x}}$ grows exponentially with the depth of the circuit being computed, causing the the noise in the ciphertext encoding to blow out of control after some number of evaluations. While it has been long known how to use circular security in the context of unbounded depth FHE [Gen09], its utility in the context of ABE was uncovered only very recently, in an elegant work by Hseih, Lin and Luo [HLL23] (HLL). In order to perform noise reduction while maintaining the required algebraic structure of the encoding, HLL included the following additional advice in their ciphertext:

$$\mathbf{S} = \mathsf{hct}_{\mathbf{s}}(\mathbf{s}), \quad \mathbf{E} = \underline{\mathbf{s}^{\mathsf{T}}(\mathbf{A}_{\mathsf{circ}} - \mathbf{S} \otimes \mathbf{G})}$$

Above, $\mathsf{hct}_{\mathbf{s}}(\cdot)$ is an FHE ciphertext decryptable by secret key $\mathbf{s}$ denoted in the subscript – thus $\mathbf{S}$ is a circular FHE ciphertext, and $\mathbf{E}$ is a BGG^{+} encoding with attribute $\mathbf{S}$ and *re-using* the FHE secret $\mathbf{s}$ as the LWE secret.

The Automatic Decryption Trick. The trick of reusing the FHE secret as the LWE secret in the BGG^{+} encoding of attribute $\mathsf{hct}_{\mathbf{s}}(\cdot)$, was introduced by Brakerski et al. [BTVW17] and can lead to “automatic decryption” of the FHE ciphertext, as described next. Recall that in the GSW FHE scheme [GSW13], the secret key is $\mathbf{s}^{\mathsf{T}}$, a ciphertext for message $\mathbf{y}^{\mathsf{T}}$ is a matrix $\mathbf{C}$ and decryption computes $\mathbf{s}^{\mathsf{T}}\mathbf{C}$ to recover (a noisy version of) $\mathbf{y}^{\mathsf{T}}$. Brakerski et al. [BTVW17] suggested “vectorizing” the BGG^{+} ciphertext evaluation procedure so that homomorphic evaluation on the encoding produces a term of the form $\underline{\mathbf{s}^{\mathsf{T}}(\mathbf{A}_f - \mathsf{hct}_{\mathbf{s}}(\mathbf{y}^{\mathsf{T}}))}$ (i.e. without $\mathbf{G}$). Now, the inner product of $\mathbf{s}$ and $\mathsf{hct}_{\mathbf{s}}(\mathbf{y}^{\mathsf{T}})$ causes FHE decryption to occur automatically and we obtain the encoding $\underline{\mathbf{s}^{\mathsf{T}}\mathbf{A}_f + \mathbf{y}^{\mathsf{T}}}$, where the noise in the encoding is low. This term suffices to proceed with homomorphic computation in HLL but at the cost of incorporating circularity into the evasive LWE assumption.

Randomizing Advice for CP-ABE. Recently, Agrawal, Kumari and Yamada [AKY24] (AKY) built upon the construction by HLL to obtain the first ABE for Turing machines from lattice assumptions. A key technical contribution of the AKY construction is a way to randomize the advice provided in the HLL ciphertext, making it suitable for integration with Wee's bounded depth CP-ABE. These techniques led to the first CP-ABE for unbounded depth circuits, which they further leveraged to construct a KP-ABE for Turing machines. In more detail, AKY transformation requires computation of the following randomized HLL terms:

$$\mathbf{S}_{\mathbf{r}} = \mathsf{hct}_{\mathbf{s}_{\mathbf{r}}}(\mathbf{s}_{\mathbf{r}}), \quad \mathbf{E}_{\mathbf{r}} = \underline{\mathbf{s}_{\mathbf{r}}^{\mathsf{T}}(\mathbf{A}_{\mathsf{circ}} - \mathbf{S}_{\mathbf{r}} \otimes \mathbf{G})}$$

Above, $\mathbf{s}_{\mathbf{r}} = \mathbf{s}^{\mathsf{T}}(\mathbf{I} \otimes \mathbf{r})$ where $\mathbf{r}$ is chosen by the key generator while $\mathbf{s}$ is chosen by the encryptor.

Evidently, neither party can provide the encodings directly, and while randomizing the message inside an FHE ciphertext from $\mathbf{s}$ to $\mathbf{s}_{\mathbf{r}}$ is easy given knowledge of $\mathbf{r}$, randomizing the secret key of FHE ciphertext is much more challenging. To get around this difficulty, AKY suggest that the structure of the advice provided by the encryptor be changed, so that the true power of FHE – which is to transform encoded messages rather than underlying secret keys – be further leveraged. Thus, they provide:

$$\mathbf{T} = \mathsf{hct}_{\mathbf{t}}(\mathbf{s}, \mathsf{sd}), \ \ \mathbf{D} = \underline{\mathbf{t}^{\mathsf{T}}(\mathbf{A}_1 - (1, \mathsf{bits}(\mathbf{T})) \otimes \mathbf{G})}$$

where sd is a PRF seed, $\mathbf{t}$ is the secret of a fresh FHE scheme and $\mathbf{A}_1$ is a public matrix of appropriate dimensions. Now, one can homomorphically evaluate on the encoding $\mathbf{D}$ in *bounded* depth, using knowledge of $\mathbf{T}$, to obtain

$$\underline{\mathbf{t}^{\mathsf{T}}\mathbf{A}'_{\mathbf{r}} + \mathbf{t}^{\mathsf{T}}\mathsf{hct}_{\mathbf{t}}(\mathbf{S}_{\mathbf{r}}, \mathbf{E}_{\mathbf{r}})} = \underline{\mathbf{t}^{\mathsf{T}}\mathbf{A}'_{\mathbf{r}} + (\mathbf{S}_{\mathbf{r}}, \mathbf{E}_{\mathbf{r}})}$$

where $\mathbf{A}'_{\mathbf{r}}$ is some $\mathbf{r}$ dependent matrix and the equality follows by automatic decryption. To get rid of the masking term $\mathbf{t}^{\mathsf{T}}\mathbf{A}'_{\mathbf{r}}$, the encryptor additionally provides $\underline{\mathbf{t}^{\mathsf{T}}\mathbf{C}}$ for some fixed matrix $\mathbf{C}$ and the key generator provides $\mathbf{C}^{-1}(\mathbf{A}'_{\mathbf{r}})$ where $\mathbf{C}^{-1}(\mathbf{A}'_{\mathbf{r}})$ is not a true matrix inverse but rather a low norm matrix so that $\mathbf{C} \cdot \mathbf{C}^{-1}(\mathbf{A}'_{\mathbf{r}}) = \mathbf{A}'_{\mathbf{r}}$. Together these allow the decryptor to compute the term $\underline{\mathbf{t}^{\mathsf{T}}\mathbf{A}'_{\mathbf{r}}}$ and cancel it out from the encoding above, to recover $(\mathbf{S}_{\mathbf{r}}, \mathbf{E}_{\mathbf{r}})$ in the clear.

The security of the above construction, relies on evasive LWE (aside from other assumptions), and depends crucially on the fact that the computed terms $(\mathbf{S}_{\mathbf{r}}, \mathbf{E}_{\mathbf{r}})$ are pseudorandom. Digging deeper into the AKY proof, the term $\underline{\mathbf{s}^{\mathsf{T}}\mathbf{P}}$ in Evasive LWE can be essentially simplified to the advice terms $(\mathbf{S}_{\mathbf{r}}, \mathbf{E}_{\mathbf{r}})$ which, therefore need to be pseudorandom for invoking the assumption.

2.2 Compact Functional Encryption for Pseudorandom Functionalities & Applications.

Our starting point is the observation that the techniques developed in AKY are quite a bit more general and can be leveraged to compute functionalities beyond the randomized HLL advice they were developed for. Perhaps surprisingly, we show that techniques developed to support an "evasive" functionality like ABE (i.e. where the adversary receives no decrypting keys) can be used to construct a full fledged Functional Encryption (FE) for a nontrivial class of functionalities. Note that unlike ABE, the FE functionality is *non-evasive* in that the adversary *can* obtain decryptions of the challenge ciphertext. This core tool – FE for pseudorandom functionalities – can then to be used in a clean, black-box way to make significant advances in the space of ABE schemes. We view the identification of this tool and its multi-input generalization as the core conceptual contribution of this work.

Generalizing AKY techniques beyond ABE. Taking a step back, let us analyze what the AKY technique enables: the encryptor provides an FHE ciphertext $\mathbf{T}$ of a message (say $\mathbf{x}$), and a BGG^+ encoding of attribute $\mathbf{T}$ with the FHE secret doubling up as the encoding randomness. Homomorphic evaluation of any function f coupled with automatic decryption allows to recover a masked version of $f(\mathbf{x})$ and evasive LWE allows to cancel the mask. Thus, this technique seems to enable computation of any function f on the input $\mathbf{x}$, while keeping it hidden! Intuitively, security follows from evasive LWE as long as the output of the functionality is pseudorandom, such as their $(\mathbf{S_r}, \mathbf{E_r})$, but more generally pseudorandom output of any function, such as a PRF.

We show that the above intuition can be formalized to yield the first compact FE for pseudorandom functionalities, namely, functionalities where the output is (pseudo)random for any given input that is seen by the adversary in the security game. We sketch our construction for prFE below. In the following, $f : \{0,1\}^{\mathsf{L}} \to \{0,1\}^{\ell}$ has the property that the output of f is pseudorandom for every input seen by the adversary. We also use a $\mathsf{PRF} : \{0,1\}^{\lambda} \times \{0,1\}^{\lambda} \to [-q/4, q/4]$. The usage of PRF is introduced for the security reasons which we will highlight later.

- The setup algorithm samples matrices $\mathbf{A}_{\mathsf{att}}$ and $(\mathbf{B}, \mathbf{B}^{-1})$ of appropriate dimensions and outputs $\mathsf{mpk} := (\mathbf{A}_{\mathsf{att}}, \mathbf{B})$ and $\mathsf{msk} := \mathbf{B}^{-1}$. Here, $\mathbf{B}^{-1}$ is the trapdoor for $\mathbf{B}$ which allows to compute short preimages $\mathbf{B}^{-1}(\mathbf{U})$ for any target matrix $\mathbf{U}$.
- The encryptor on input $\mathbf{x}$ first samples a GSW secret key $\mathbf{s}$, where $\mathbf{s} = (\bar{\mathbf{s}}^{\mathsf{T}} \quad -1)^{\mathsf{T}}$ and a PRF seed $\mathsf{sd} \leftarrow \{0,1\}^{\lambda}$. It then computes a GSW ciphertext, $\mathbf{X} = \mathsf{hct}_{\mathbf{s}}(\mathbf{x}, \mathsf{sd})$, using public key $\mathbf{A}_{\mathsf{fhe}} = \begin{pmatrix} \bar{\mathbf{A}}_{\mathsf{fhe}} \\ \bar{\mathbf{s}}^{\mathsf{T}} \bar{\mathbf{A}}_{\mathsf{fhe}} + \mathbf{e}_{\mathsf{fhe}}^{\mathsf{T}} \end{pmatrix}$ and randomness $\mathbf{R}$, – followed by a BGG^+ encoding of $\mathbf{X}$ using randomness $\mathbf{s}$ as $\mathbf{c}_{\mathsf{att}}^{\mathsf{T}} := \mathbf{s}^{\mathsf{T}}(\mathbf{A}_{\mathsf{att}} - \mathbf{X} \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{att}}^{\mathsf{T}}$. It additionally computes $\mathbf{c}_{\mathbf{B}}^{\mathsf{T}} := \mathbf{s}^{\mathsf{T}}\mathbf{B} + \mathbf{e}_{\mathbf{B}}^{\mathsf{T}}$ and outputs the ciphertext $\mathsf{ct} = (\mathbf{c}_{\mathbf{B}}, \mathbf{c}_{\mathsf{att}}, \mathbf{X})$.
- The key generator on input $\mathsf{msk} = \mathbf{B}^{-1}$ and function f does the following.
  (a) Samples a nonce $\mathbf{r} \leftarrow \{0,1\}^{\lambda}$ and defines function $\mathrm{F}[f, \mathbf{r}]$, with f and $\mathbf{r}$ hardwired, as
  $$\mathrm{F}[f, \mathbf{r}](\mathbf{x}, \mathsf{sd}) = f(\mathbf{x}) \lfloor q/2 \rceil + \mathsf{PRF}(\mathsf{sd}, \mathbf{r}).$$
  It then computes the FHE evaluation circuit $\mathsf{VEval}_{\mathrm{F}}$ w.r.t. the function $\mathrm{F}[f, \mathbf{r}]$ (this can be computed using the knowledge of $\mathrm{F}[f, \mathbf{r}]$). Note that the circuit $\mathsf{VEval}_{\mathrm{F}}$ can be used to compute on a GSW ciphertext for an input, say $\mathbf{y}$, to recover a GSW ciphertext encoding $\mathrm{F}[f, \mathbf{r}](\mathbf{y})$.
  (b) Next, it computes the matrix $\mathbf{H}_{\mathbf{A}_{\mathsf{att}}}^{\mathrm{F}}$ for the circuit $\mathsf{VEval}_{\mathrm{F}}$ using the public matrix $\mathbf{A}_{\mathsf{att}}$. Recall that the matrix $\mathbf{H}_{\mathbf{A}_{\mathsf{att}}}^{\mathrm{F}}$ and $\mathbf{H}_{\mathbf{A}_{\mathsf{att}}, \mathbf{X}}^{\mathrm{F}}$ (which can be computed given $\mathsf{VEval}_{\mathrm{F}}$, $\mathbf{A}_{\mathsf{att}}$ and $\mathbf{X}$) will satisfy the relation
  $$(\mathbf{A}_{\mathsf{att}} - \mathbf{X} \otimes \mathbf{G})\mathbf{H}_{\mathbf{A}_{\mathsf{att}}, \mathbf{X}}^{\mathrm{F}} = \mathbf{A}_{\mathsf{att}}\mathbf{H}_{\mathbf{A}_{\mathsf{att}}}^{\mathrm{F}} - \mathsf{VEval}_{\mathrm{F}}(\mathbf{X}).$$
  (c) It sets $\mathbf{A}_{\mathrm{F}} = \mathbf{A}_{\mathsf{att}} \cdot \mathbf{H}_{\mathbf{A}_{\mathsf{att}}}^{\mathrm{F}}$, samples $\mathbf{K} \leftarrow \mathbf{B}^{-1}(\mathbf{A}_{\mathrm{F}})$ and outputs $\mathsf{sk}_f = (\mathbf{K}, \mathbf{r})$.
- The decryption on input $\mathsf{sk}_f = (\mathbf{K}, \mathbf{r})$ and $\mathsf{ct} = (\mathbf{c}_{\mathbf{B}}, \mathbf{c}_{\mathsf{att}}, \mathbf{X})$ work as follows.
  (a) It first computes the matrix $\mathbf{H}_{\mathbf{A}_{\mathsf{att}}, \mathbf{X}}^{\mathrm{F}}$ for the circuit $\mathsf{VEval}_{\mathrm{F}}$ using $\mathbf{A}_{\mathsf{att}}$ and $\mathbf{X}$.

(b) Next, it computes $\mathbf{z} := \mathbf{c}_{\mathbf{B}}^{\mathsf{T}} \cdot \mathbf{K} - \mathbf{c}_{\mathsf{att}}^{\mathsf{T}} \cdot \mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}}$, rounds $\mathbf{z}$ co-ordinate wise and output the most significant bits.

To see the correctness of our scheme, we note that

$$\mathbf{c}_{\mathsf{att}}^{\mathsf{T}} \cdot \mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}} \approx \mathbf{s}^{\mathsf{T}} \mathbf{A}_{\mathsf{att}} \mathbf{H}_{\mathbf{A}_{\mathsf{att}}}^{\mathrm{F}} - \mathbf{s}^{\mathsf{T}}(\mathsf{hct}_{\mathbf{s}}(\mathrm{F}(\mathbf{x}, \mathsf{sd}))) \approx \mathbf{s}^{\mathsf{T}} \mathbf{A}_{\mathrm{F}} - \mathrm{F}(\mathbf{x}, \mathsf{sd}), \tag{1}$$

where the second approximate equality follows by automatic decryption. Now to remove the masking term "$\mathbf{s}^{\mathsf{T}}\mathbf{A}_{\mathrm{F}}$" we compute $\mathbf{c}_{\mathbf{B}}^{\mathsf{T}} \cdot \mathbf{K} \approx \mathbf{s}^{\mathsf{T}}\mathbf{A}_{\mathrm{F}}$ and thus $\mathbf{z} \approx \mathbf{s}^{\mathsf{T}}\mathbf{A}_{\mathrm{F}} - \mathbf{s}^{\mathsf{T}}\mathbf{A}_{\mathrm{F}} + \mathrm{F}(\mathbf{x}, \mathsf{sd}) = f(\mathbf{x}) \lfloor q/2 \rceil + \mathsf{PRF}(\mathsf{sd}, \mathbf{r})$. Now, rounding gives us bits of $f(\mathbf{x})$ as long as $|\mathsf{PRF}(\mathsf{sd}, \mathbf{r})| \leq q/4$. The exact decryption error is

$$\mathbf{e}_{\mathbf{B}}^{\mathsf{T}}\mathbf{K} + \mathsf{PRF}(\mathsf{sd}, \mathbf{r}) - (\mathbf{e}_{\mathsf{fhe}}^{\mathsf{T}}\mathbf{R}_{\mathrm{F}} + \mathbf{e}_{\mathsf{att}}^{\mathsf{T}}\mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}}) \tag{2}$$

where $\mathbf{R}_{\mathrm{F}}$ is a matrix with small entries determined by $\mathbf{R}$ and F satisfying $\mathsf{VEval}_{\mathrm{F}}(\mathsf{bits}(\mathbf{X})) = \mathbf{A}_{\mathsf{fhe}}\mathbf{R}_{\mathrm{F}} - (\mathbf{0} \;\; \mathrm{F}[f, \mathbf{r}](\mathbf{x}, \mathsf{sd}))^{\mathsf{T}}$.

Our construction supports functions of bounded polynomial depth $\mathsf{dep} = \mathrm{poly}(\lambda)$ and has the following efficiency

$$|\mathsf{mpk}| = \mathrm{L} \cdot \mathrm{poly}(\mathsf{dep}, \lambda), \quad |\mathsf{sk}_f| = \ell \cdot \mathrm{poly}(\mathsf{dep}, \lambda), \quad |\mathsf{ct}| = \mathrm{L} \cdot \mathrm{poly}(\mathsf{dep}, \lambda).$$

As seen above, our construction achieves compactness.

Malicious Circuit Attack and Fix. In the above construction, if we allow the adversary to choose the circuit implementation of the PRF used in the above construction along with its corresponding homomorphic evaluation, then there is a contrived attack that allows the adversary to learn problematic leakage [AMYY25]. At a very high level, the attack, building upon clever ideas by [HJL21], shows a way to create a correlation between the error term resulting from FHE evaluation (and automatic decryption) with the PRF output by using a contrived circuit to implement the PRF. A similar attack applies to the concurrent work of [BDJ+25].

There are multiple simple ways to prevent such an attack – the simplest one is to leverage the fact that the secret key is computed by the key generator who is an honest party (it holds the master secret key) in the real world, and can ensure that the circuit representation of any function f as well as the PRF can be made canonical by using the universal circuit or a garbled circuit representation. Nevertheless, in this work, we also present an alternate fix to the scheme which uses modulus reduction to "throw away" the accumulated error after FHE evaluation, replacing it with rounding error which is no longer correlated with the PRF seed, even for a contrived circuit chosen by the adversary. Thus, our current scheme is not subject to any attacks for the functionalities considered in our work, even for maliciously chosen circuit specifications, to the best of our knowledge. Please see Appendix A for details.

Security. We first define a strong simulation style security which says that so long as the *output* of the functionality is pseudorandom, the *ciphertext* is pseudorandom, given all the additional information available to the adversary. We denote this security notion as prCT security. While this notion was shown to be impossible for general functionalities [AMYY25] by exhibiting a self referential functionality where it cannot hold, we believe it is still meaningful for natural functionalities and discuss it here. We discuss indistinguishability style security later.

In more detail, let Samp be a PPT algorithm that on input 1^λ, outputs

$$(f_1, \ldots, f_{Q_{\mathsf{key}}}, x_1, \ldots, x_{Q_{\mathsf{msg}}}, \mathsf{aux} \in \{0,1\}^*)$$

where Q_{key} is the number of key queries, Q_{msg} is the number of message queries. We say that a prFE scheme is secure if

$$\begin{pmatrix} \mathsf{mpk},\ \mathsf{aux},\ f_1, \ldots, f_{Q_{\mathsf{key}}}, \\ \{\mathsf{Enc}(\mathsf{mpk}, x_j)\}_{j \in [Q_{\mathsf{msg}}]},\ \mathsf{sk}_{f_1}, \ldots, \mathsf{sk}_{f_{Q_{\mathsf{key}}}} \end{pmatrix} \approx_c \begin{pmatrix} \mathsf{mpk},\ \mathsf{aux},\ f_1, \ldots, f_{Q_{\mathsf{key}}}, \\ \{\delta_j \leftarrow \mathcal{CT}\}_{j \in [Q_{\mathsf{msg}}]},\ \mathsf{sk}_{f_1}, \ldots, \mathsf{sk}_{f_{Q_{\mathsf{key}}}} \end{pmatrix}$$

$$\text{given} \quad \left(\mathsf{aux},\ f_1, \ldots, f_{Q_{\mathsf{key}}},\ \{f_i(x_j)\}_{i \in [Q_{\mathsf{key}}], j \in [Q_{\mathsf{msg}}]}\right) \approx_c \left(\mathsf{aux},\ f_1, \ldots, f_{Q_{\mathsf{key}}},\ \{\Delta_{i,j}\}_{i \in [Q_{\mathsf{key}}], j \in [Q_{\mathsf{msg}}]}\right)$$

where $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$, $\mathsf{sk}_{f_i} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f_i)$ for $i \in [Q_{\mathsf{key}}]$, $\mathcal{CT}$ is the ciphertext space and $\Delta_{i,j} \leftarrow \{0,1\}^\ell$ for $i \in [Q_{\mathsf{key}}], j \in [Q_{\mathsf{msg}}]$.

The careful reader may have noticed that the above definition has a multi-challenge flavour, even though the construction is in the public-key setting. This peculiarity arises because single-challenge security does not generically imply multi-challenge security for our definition. To see this, recall the standard hybrid argument to prove multi-challenge security from single-challenge security: the proof follows a sequence of hybrids, where we simulate some of the ciphertexts honestly, while trying to change a particular honest ciphertext to be random. Now, to generate honest ciphertexts, we need to know the corresponding plaintexts. However, this could ruin the precondition for invoking single-challenge security for the target ciphertext, since knowing some inputs may ruin the pseudorandomness of outputs, if the inputs are correlated to each other.

We provide some high level intuition for our proof of security for prFE. For simplicity, we focus here on the single challenge setting and refer the reader to the main body for the detailed proof in the multi-challenge setting. The proof begins by invoking evasive LWE with an appropriate sampler – this allows to reduce the reasoning to the distribution of the pre-condition, which replaces the term $\mathbf{K} = \mathbf{B}^{-1}(\mathbf{A}_{\mathrm{F}})$ with $\mathbf{c}_{\mathbf{B}}^{\mathsf{T}} \cdot \mathbf{K} = \underline{\mathbf{s}^{\mathsf{T}}\mathbf{A}_{\mathrm{F}}}$. Now, as we see in Equation (1),

$$\mathbf{c}_{\mathsf{att}}^{\mathsf{T}} \cdot \mathbf{H}^{\mathrm{F}}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}} = \underline{\mathbf{s}^{\mathsf{T}}\mathbf{A}_{\mathrm{F}} - \mathrm{F}(\mathbf{x},\mathsf{sd})} = \underline{\mathbf{s}^{\mathsf{T}}\mathbf{A}_{\mathrm{F}} - f(\mathbf{x})\,\lfloor q/2\rceil - \mathsf{PRF}(\mathsf{sd},\mathbf{r})}.$$

This allows to simplify these two terms to $\underline{\mathbf{s}^{\mathsf{T}}\mathbf{A}_{\mathrm{F}}}$ and $\underline{f(\mathbf{x})\,\lfloor q/2\rceil + \mathsf{PRF}(\mathsf{sd},\mathbf{r})}$, where the latter term is pseudorandom, hence simulatable and can be ignored hereafter. The terms that remain can now be handled by relying on LWE using standard techniques [Wee22, HLL23, AKY24]. Please see Section 4.2 for details.

IND-Secure prFE. To avoid the impossibility result of [AMYY25], we define a weaker IND security which is standard in FE literature. The IND security for a prFE scheme is defined as follows. Let Samp be a PPT algorithm that outputs $(f_1,\ldots,f_{Q_{\mathsf{key}}},x_1^0,\ldots,x_{Q_{\mathsf{msg}}}^0,x_1^1,\ldots,x_{Q_{\mathsf{msg}}}^1,\mathsf{aux} \in \{0,1\}^*)$ where Q_{key} is the number of key queries, Q_{msg} is the number of message queries. We say that a prFE scheme satisfies IND-Security if

$$\Big(\mathsf{aux},\ \{f_i,\ f_i(x_j^0)\}_{i\in[Q_{\mathsf{key}}],j\in[Q_{\mathsf{msg}}]}\Big) \approx_c \big(\mathsf{aux},\ \{f_i,\ \Delta_{i,j} \leftarrow \mathcal{Y}_{\mathsf{prm}}\}_{i\in[Q_{\mathsf{key}}],j\in[Q_{\mathsf{msg}}]}\big)$$

and $f_i(x_j^0) = f_i(x_j^1) \qquad \forall i \in [Q_{\mathsf{key}}], \forall j \in [Q_{\mathsf{msg}}]$, then we have

$$(\mathsf{mpk},\ \mathsf{aux},\ \{f_i,\ \mathsf{Enc}(\mathsf{mpk},x_j^0),\ \mathsf{sk}_{f_i}\}_{i,j}) \approx_c (\mathsf{mpk},\ \mathsf{aux},\ \{f_i,\ \mathsf{Enc}(\mathsf{mpk},x_j^1),\ \mathsf{sk}_{f_i}\}_{i,j})$$

where $i \in [Q_{\mathsf{key}}], j \in [Q_{\mathsf{msg}}], (f_1,\ldots,f_{Q_{\mathsf{key}}},x_1^0,\ldots,x_{Q_{\mathsf{msg}}}^0,x_1^1,\ldots,x_{Q_{\mathsf{msg}}}^1,\mathsf{aux}) \leftarrow \mathsf{Samp}(1^\lambda), (\mathsf{mpk},\mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda,\mathsf{prm})$.
We note that this definition is strictly weaker than standard indistinguishability definition for FE [GGH+16], since we require the security to hold only for the case where all the decryption results are pseudorandom. We use this definition for all our applications. We conjecture that our prFE scheme satisfies this notion of security.

Justification for IND security of prFE. We discuss the reasoning behind our conjecture that our prFE satisfies IND security. In terms of basing security from an assumption, we can prove prCT security for prFE under suitably parametrized evasive LWE, to avoid all known attacks for the functionalities considered in this work. Opening up the security proof, the sampler for the evasive LWE that we assume is induced by the sampler used in the prCT definition for prFE. Hence, by suitably restricting the sampler of evasive LWE, one can conjecture prCT security of prFE for natural functionalities, to which lower bounds of [AMYY25, BDJ+25] do not apply.

To the best of our knowledge, there are no attacks or impossibilities against the scheme for suitably restricted samplers and functionalities. Since prCT security can be proven from a plausibly sound version of evasive LWE and since it implies the weaker IND based security which does not suffer from any impossibility, we conjecture that our prFE satisfies IND based security. While our applications only require security to hold for a natural class of functionalities for which even prCT security may plausibly hold, we also conjecture that IND security may be satisfied by our construction for *all* pseudorandom functionalities, even contrived ones for which prCT security cannot hold.

2.2.1 Application to ABE with Unbounded Depth

For ease of exposition, we describe our applications using a (single-challenge) prCT-secure prFE scheme (Definition 4.2). However, we can adapt the construction to be based on an IND-secure prFE by leveraging the *Trojan method* [DCIJ+13] – we can hardwire the decryption result of the prFE ciphertext into the prFE secret key using a SKE scheme having pseudorandom ciphertext. Details can be found in the relevant technical sections.

Removing Circularity from HLL. We demonstrate the utility of prFE by showing that it can be used to bootstrap a very weak kpABE scheme into a full fledged one. This enables us to improve assumptions underlying prior works.

In more detail, our weak kpABE scheme, denoted by $\mathsf{1ABE}$ is a *secret key* scheme which only supports a *single* ciphertext and *single* secret key query – this object is so simple that it can be constructed merely from one way functions. This is lifted using prFE to build a full fledged *public key* ABE scheme supporting *unbounded* ciphertexts and *unbounded* key queries. Our compiler does require $\mathsf{1ABE}$ to satisfy some structural properties:

1. Decomposability: The computation $\mathsf{1ABE.KeyGen}(C)$ can be decomposed into $\{\mathsf{1ABE.KeyGen}_i(C_i)\}_{i\in|C|}$ where C_i denotes the i-th gate of C and has fixed polynomial size. Here, the depth of $\mathsf{ABE.KeyGen}_i$ is fixed and independent of the parameters of C. Moreover, output of $\mathsf{1ABE.Enc}$ should be computable by a circuit of fixed depth, irrespective of the length of $\mathbf{x}$ input to $\mathsf{1ABE.Enc}$.

2. Blindness: The $\mathsf{1ABE}$ ciphertext and secret key should be pseudorandom when decryption is not allowed.

Given the above properties, the core idea is to use prFE to generate randomized versions of bits of $\mathsf{1ABE}$ secret keys and ciphertexts using randomness generated jointly by the encryptor and the key generator. This is supported by prFE of fixed depth because of property 1 and respects the constraints imposed by prFE because of property 2. Thus we obtain a public key scheme which supports unbounded ciphertexts and keys. We outline our compiler below.

- The setup algorithm generates $(\mathsf{prFE.msk}, \mathsf{prFE.mpk})$ using $\mathsf{prFE.Setup}$ and outputs these as msk and mpk respectively.

- The encryption algorithm on input mpk, attribute $\mathbf{x}$ and message μ computes a prFE ciphertext, $\mathsf{prFE.ct}$, encoding input $(\mathbf{x}, \mu, \mathsf{sd})$ where $\mathsf{sd} \leftarrow \{0,1\}^\lambda$ is a PRF seed.

- The keygen algorithm on input $\mathsf{msk} = \mathsf{prFE.msk}$ and circuit C works as follows. It samples nonce $\mathbf{r} \leftarrow \{0,1\}^\lambda$ and defines functions $\mathsf{F}_{\mathsf{key},i}[\mathbf{r}, C_i]$, with $\mathbf{r}$ and i-th gate of C hardwired, for $i \in [|C|]$ and $\mathsf{F}_{\mathsf{ct}}[\mathbf{r}]$ as follows

  (a) $\mathsf{F}_{\mathsf{key},i}[\mathbf{r}, C_i]$ on input $(\mathbf{x}, \mu, \mathsf{sd})$, first computes $\mathsf{1ABE.msk}$ using the randomness $\mathsf{PRF}(\mathsf{sd}, \mathbf{r})$, i.e. $\mathsf{1ABE.msk} \leftarrow \mathsf{1ABE.Setup}(1^\lambda; \mathsf{PRF}(\mathsf{sd}, \mathbf{r}))$ and then outputs $\mathsf{1ABE.sk}_{C_i} \leftarrow \mathsf{1ABE.KeyGen}_i(\mathsf{1ABE.msk}, C_i)$.

  (b) $\mathsf{F}_{\mathsf{ct}}[\mathbf{r}]$ on input $(\mathbf{x}, \mu, \mathsf{sd})$, first computes $\mathsf{1ABE.msk}$ as above and then outputs an $\mathsf{1ABE.ct}$ encoding message μ w.r.t. attribute $\mathbf{x}$.

  It then computes prFE keys $\{\mathsf{prFE.sk}_{\mathsf{key},i}\}_{i\in[|C|]}$ and $\mathsf{prFE.sk}_{\mathsf{ct}}$ corresponding to functions $\{\mathsf{F}_{\mathsf{key}}[\mathbf{r}, i]\}_{i\in[|C|]}$ and $\mathsf{F}_{\mathsf{ct}}[\mathbf{r}]$, respectively. It outputs $\mathsf{sk}_C = (\{\mathsf{prFE.sk}_{\mathsf{key},i}\}_{i\in[|C|]}, \mathsf{prFE.sk}_{\mathsf{ct}})$.

- The decryption algorithm on input $\mathsf{sk}_C = (\{\mathsf{prFE.sk}_{\mathsf{key},i}\}_{i\in[|C|]}, \mathsf{prFE.sk}_{\mathsf{ct}})$ and $\mathsf{ct} = \mathsf{prFE.ct}$ first runs the prFE decryption, using prFE keys and ciphertext $\mathsf{prFE.ct}$, to compute

  $$\mathsf{F}_{\mathsf{key},i}[\mathbf{r}, C_i](\mathbf{x}, \mu, \mathsf{sd}) = \mathsf{1ABE.sk}_{C_i}, \quad \mathsf{F}_{\mathsf{ct}}[\mathbf{r}](\mathbf{x}, \mu, \mathsf{sd}) = \mathsf{1ABE.ct}.$$

  Finally it sets $\mathsf{1ABE.sk}_C = (\mathsf{1ABE.sk}_{C_1}, \ldots, \mathsf{1ABE.sk}_{C_{|C|}})$ and outputs the decryption result as $\mathsf{1ABE.Dec}(\mathsf{1ABE.sk}_C, \mathsf{1ABE.ct})$.

Correctness follows from those of the prFE and $\mathsf{1ABE}$: By correctness of prFE, the decryptor recovers the ciphertext and secret key pair of $\mathsf{1ABE}$ and by the correctness of $\mathsf{1ABE}$, one can recover the message μ when $C(\mathbf{x}) = 1$. To prove the security, it suffices to show that $\mathsf{ct} = \mathsf{prFE.ct}$ is pseudorandom. By the security of prFE, it suffices to show that the decryption results of the ciphertext using the secret keys are jointly pseudorandom. This follows from the security of $\mathsf{1ABE}$, since the decryption results are ciphertext and secret key pairs generated by different randomness, where the decryption is not possible for each pair. We refer to Section 5.2 for details.

Building $\mathsf{1ABE}$. It remains to instantiate $\mathsf{1ABE}$ with the desired properties. Fortunately, these properties are relatively weak and easily satisfied. For instance, we can instantiate $\mathsf{1ABE}$ simply by using blind garbled circuits [BLSV18] (Section 3.2.3)– here *blindness* is precisely the property that the garbled circuit and its labels should be pseudorandom, when the evaluation result of the garbled circuit using the given labels is random. Given a blind garbled circuit, the labels of the garbled circuit form the $\mathsf{1ABE}$ ciphertext, the set of garbled gates form the $\mathsf{1ABE}$ key, and decomposability follows from the structure of a garbled circuit, where we can garble each gate independently. Care needs to be taken to modify the circuit C in the secret key so that when $C(\mathbf{x}) = 0$, it outputs a random string rather than $\perp$ so as to be compatible with our prFE. This yields the following theorem.

Theorem 2.1. Assuming LWE and IND-secure prFE, there exists a very selectively secure kpABE scheme for circuits of unbounded depth and attribute length ℓ with $|\mathsf{mpk}| = \ell \cdot \mathrm{poly}(\lambda)$, $|\mathsf{sk}_C| = |C| \cdot \ell \cdot \mathrm{poly}(\lambda)$, $|\mathsf{ct}| = \ell \cdot \mathrm{poly}(\lambda)$.

Note that HLL relied on a new assumption called circular evasive LWE, which we replace by IND-secure prFE above albeit at the cost of larger parameters – in particular the secret key is large and scales with $|C|$.

Next, we show that by using the abstraction of Attribute Based Laconic Function Evaluation ($\mathsf{AB\text{-}LFE}$) [QWW18], we can match the parameters by HLL. Intuitively $\mathsf{AB\text{-}LFE}$ allows to compress a circuit C into a short digest, which is then used by an encryptor to compute a ciphertext ct for some attribute, message pair $(\mathbf{x}, \mu)$. The decryptor, given C, ct can recover μ if and only if $C(\mathbf{x}) = 1$. For our compiler, we require an $\mathsf{AB\text{-}LFE}$ scheme where the encryption algorithm can be decomposed into an offline and an online phase. The offline encryption algorithm takes as input $(\mathbf{x}, \mu)$ and outputs $\mathsf{ct}_{\mathsf{off}}$ and a private state st. The online encryption algorithm takes as input $(\mathsf{st}, \mathsf{digest})$ and outputs $\mathsf{ct}_{\mathsf{on}}$. This property is satisfied by the construction of [HLL23].

At a high level, our kpABE uses the compression of the $\mathsf{AB\text{-}LFE}$ to shorten the secret key. The key generation computes a digest $\hat{\mathsf{C}}$ for the circuit C, then computes a prFE key for a circuit which outputs the online part of $\mathsf{AB\text{-}LFE}$ ciphertext. The encrypt algorithm computes the offline part of the $\mathsf{AB\text{-}LFE}$ ciphertext and the state st using input $(\mathbf{x}, \mu)$. It then encrypts st using prFE encryption. Now, prFE decryption allows to recover the online part of the ciphertext, and $\mathsf{AB\text{-}LFE}$ decryption allows to recover μ if $C(\mathbf{x}) = 1$.We refer the reader to Section 5.4 for details.We prove the following theorem:

Theorem 2.2. Under the circular LWE assumption and IND-secure prFE, there exists a very selectively secure kpABE scheme for circuits of unbounded depth and attribute length ℓ with

$$|\mathsf{mpk}| = \mathrm{poly}(\ell, \lambda), \quad |\mathsf{sk}_C| = \mathrm{poly}(\lambda), \quad |\mathsf{ct}| = \mathrm{poly}(\ell, \lambda).$$

Constructing kpABE for Turing machines from Weaker Assumptions. Next, we turn our attention to kpABE for Turing machines. We show that using prFE and a single-key kpABE for unbounded depth circuits, we can build a cpABE for unbounded depth circuits and further a kpABE for Turing machines with parameters matching AKY. The construction for cpABE is as follows.

– The setup algorithm generates $(\mathsf{prFE.msk}, \mathsf{prFE.mpk})$ using $\mathsf{prFE.Setup}$ and outputs these as $\mathsf{cpABE.msk}$ and $\mathsf{cpABE.mpk}$ respectively.

– The encryption algorithm, given circuit C and message μ, works as follows: It samples randomness R_{key} and computes $(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk}) = \mathsf{kpABE.Setup}(1^\lambda, 1^\ell; R_{\mathsf{key}})$. Next, it computes a $\mathsf{prFE.ct}$ encoding $(R_{\mathsf{key}}, \mathsf{sd}, \mu)$, where sd is a PRF key, and a kpABE secret key for circuit C as $\mathsf{kpABE.sk}_C \leftarrow \mathsf{kpABE.KeyGen}(\mathsf{kpABE.msk}, C)$.It outputs $\mathsf{cpABE.ct} := (\mathsf{prFE.ct},\ \mathsf{kpABE.mpk}, \mathsf{kpABE.sk}_C)$.

– The key generation algorithm, given msk and attribute $\mathbf{x}$ outputs a prFE key, $\mathsf{prFE.sk}_{\mathrm{F}}$, for the function $\mathrm{F}[\mathbf{x}, \mathbf{r}]$ where $\mathbf{r} \leftarrow \{0,1\}^\lambda$. It outputs $\mathsf{cpABE.sk}_{\mathbf{x}} := \mathsf{prFE.sk}_{\mathrm{F}}$.
  The function $\mathrm{F}[\mathbf{x}, \mathbf{r}]$ on input $(R_{\mathsf{key}}, \mathsf{sd}, \mu)$ first computes $(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk})$ using the randomness R_{key} and then outputs a kpABE ciphertext $\mathsf{kpABE.ct}$ encoding μ w.r.t. attribute $\mathbf{x}$ using randomness $\mathsf{PRF}(\mathsf{sd}, \mathbf{r})$.

– The decryption algorithm on input secret key $\mathsf{prFE.sk}_{\mathrm{F}}$ and ciphertext $\mathsf{cpABE.ct} := (\mathsf{prFE.ct}, \mathsf{kpABE.mpk}, \mathsf{kpABE.sk}_C)$ first runs the prFE decryption to obtain $\mathrm{F}[\mathbf{x}, \mathbf{r}](R_{\mathsf{key}}, \mathsf{sd}, \mu) = \mathsf{kpABE.ct}$ and finally performs kpABE decryption using $\mathsf{kpABE.ct}$ and $\mathsf{kpABE.sk}_C$.

Correctness follows from those of kpABE and prFE: The decryption of prFE ciphertext using the prFE secret key yields kpABE ciphertext encrypted under $\mathbf{x}$. This kpABE ciphertext can be decrypted using $\mathsf{kpABE.sk}_C$ when $C(\mathbf{x}) = 1$. To prove the security, we show prFE.ct is pseudorandom. By the security of prFE, it suffices to show that the decryption results of prFE.ct are pseudorandom. This follows from the security of kpABE, since the decryption results are kpABE ciphertexts which cannot be decrypted by $\mathsf{kpABE.sk}_C$. We refer the reader to Section 6.1 for details. Thus, we obtain the following theorem.

Theorem 2.3. Assuming circular small-secret LWE and IND-secure prFE (Definition 4.4), there exists a very selectively secure cpABE scheme for circuits $\{C : \{0,1\}^\ell \to \{0,1\}\}$ of unbounded depth with

$$|\mathsf{cpABE.mpk}| = \mathrm{poly}(\lambda), \quad |\mathsf{cpABE.sk_x}| = \mathrm{poly}(\ell, \lambda), \quad |\mathsf{cpABE.ct}_C| = \mathrm{poly}(\lambda).$$

AKY provided a compiler that uses kpABE for bounded depth circuits and cpABE for unbounded depth circuits to achieve kpABE for Turing machines. Plugging our new cpABE into this compiler, we obtain:

Corollary 2.4. Assuming circular small-secret LWE and IND-secure prFE (Definition 4.4), there exists a very selectively secure ABE for TM with

$$|\mathsf{mpk}| = \mathrm{poly}(\lambda), \quad |\mathsf{sk}| = \mathrm{poly}(\lambda, |M|), \quad |\mathsf{ct}| = \mathrm{poly}(\lambda, |\mathbf{x}|, t).$$

Unbounded-Depth prFE. In Appendix C , we show how to compile a compact bounded-depth prFE scheme into a compact unbounded depth prFE using blind garbled circuits.

2.3 Multi-Input prFE & Applications

We extend the notion of prFE to multi-input setting and define a *secret-key* multi-input FE for pseudorandom functionalities $\mathsf{prMIFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}_1, \ldots, \mathsf{Enc}_n, \mathsf{Dec})$ for n-ary functions as follows: The setup algorithm on input 1^λ, arity 1^n and parameter prm, specifying the parameters of the function class, outputs $(\mathsf{mpk}, \mathsf{msk})$. The key generation algorithm on input msk and a function $f : (\mathcal{X}_{\mathsf{prm}})^n \to \mathcal{Y}_{\mathsf{prm}}$ outputs a functional secret key sk_f. The i-th encryption algorithm on input msk and an input message $x_i \in \mathcal{X}_{\mathsf{prm}}$ outputs a ciphertext ct_i. The decryption algorithm on input secret key sk_f and n ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_n$ (corresponding to inputs $x_1, \ldots, x_n$ respectively) outputs some $y \in \mathcal{Y}_{\mathsf{prm}}$. As in the single input setting, we define security in both the simulation and indistinguishability styles.

prCT Security. In prCT security, we require that the ct is pseudorandom given the output of the function of encrypted input is pseudorandom– however it requires much care to accommodate the fact that there are exponentially many function evaluations even if only polynomially many input queries per slot are issued. We define it as follows. Let $\kappa = \kappa(\lambda)$ be a function in λ and Samp be a PPT algorithm that on input 1^λ, outputs

$$\Big(\{f_k\}_{k\in[q_0]}, \{x_1^{j_1}\}_{j_1\in[q_1]}, \ldots, \{x_n^{j_n}\}_{j_n\in[q_n]}, \mathsf{aux} \in \{0,1\}^*\Big)$$

where q_0 is the number of key queries and q_i is the number of encryption queries for the i-th slot. We define the following advantage functions:

$$\begin{aligned}\mathsf{Adv}^{\mathrm{PRE}}_{\mathcal{A}_0}(\lambda) \overset{\mathrm{def}}{=} \Pr\Big[&\mathcal{A}_0(1^\kappa, f_1, \ldots, f_{q_0}, \{f_k(x_1^{j_1}, \ldots, x_n^{j_n})\}_{k\in[q_0], j_1\in[q_1], \ldots, j_n\in[q_n]}, \mathsf{aux}) = 1\Big] \\ &- \Pr\Big[\mathcal{A}_0(1^\kappa, f_1, \ldots, f_{q_0}, \{\Delta_{k,j_1,\ldots,j_n} \leftarrow \mathcal{Y}_{\mathsf{prm}}\}_{k\in[q_0], j_1\in[q_1], \ldots, j_n\in[q_n]}, \mathsf{aux}) = 1\Big]\end{aligned}$$

$$\begin{aligned}\mathsf{Adv}^{\mathrm{POST}}_{\mathcal{A}_1}(\lambda) \overset{\mathrm{def}}{=} \Pr\Big[&\mathcal{A}_1(\mathsf{mpk}, f_1, \ldots, f_{q_0}, \{\mathsf{Enc}_i(\mathsf{msk}, x_i^{j_i})\}_{i\in[n], j_i\in[q_i]}, \mathsf{sk}_{f_1}, \ldots, \mathsf{sk}_{f_{q_0}}, \mathsf{aux}) = 1\Big] \\ &- \Pr\Big[\mathcal{A}_1(\mathsf{mpk}, f_1, \ldots, f_{q_0}, \{\delta_i^{j_i} \leftarrow \mathsf{Sim}(\mathsf{msk})\}_{i\in[n], j_i\in[q_i]}, \mathsf{sk}_{f_1}, \ldots, \mathsf{sk}_{f_{q_0}}, \mathsf{aux}) = 1\Big]\end{aligned}$$

where $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, \mathsf{prm})$ and $\mathsf{sk}_{f_k} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f_k)$ for $k \in [q_0]$. We say that a prMIFE scheme is secure if for every PPT Samp, $\mathcal{A}_1$, and Sim there exists another PPT $\mathcal{A}_0$ and a polynomial $p(\cdot)$ such that

$$\mathsf{Adv}^{\mathrm{PRE}}_{\mathcal{A}_0}(\lambda) \geq \mathsf{Adv}^{\mathrm{POST}}_{\mathcal{A}_1}(\lambda)/p(\kappa) - \mathsf{negl}(\kappa), \quad \mathsf{Time}(\mathcal{A}_0) \leq p(\kappa) \cdot \mathsf{Time}(\mathcal{A}_1).$$

The parameter κ above is introduced to adjust the strength of the requirement for the precondition. By default, we require $\kappa \geq \lambda^n$ since the input length to the distinguisher is polynomial in λ^n anyway and this condition should be fulfilled for the above equations to to make sense. If we need κ to be larger, this strengthens the requirement for the precondition, as it means we want the distributions in the pre-condition to be indistinguishable against an adversary with a longer running time.[2] Ideally, we want κ to be as small as λ^n to make the requirement weaker. Looking ahead, for our construction of prMIFE scheme supporting polynomial arity $n = \mathrm{poly}(\lambda)$, we require large κ as an artifact of the security proof techniques. In the special case of n being constant, we can achieve $\kappa = \lambda^n$.

κ-IND Secure prMIFE. Similar to prFE, we introduce the κ-IND-security of prMIFE, a strictly weaker security notion than κ-security. It is defined, for $\kappa = \kappa(\lambda)$, as follows. Let Samp be a PPT algorithm that outputs

$$\left(\{f_k\}_{k\in[q_0]}, \{x_{1,0}^{j_1}, x_{1,1}^{j_1}\}_{j_1\in[q_1]}, \dots, \{x_{n,0}^{j_n}, x_{n,1}^{j_n}\}_{j_n\in[q_n]}, \mathsf{aux} \in \{0,1\}^*\right)$$

where q_0 is the number of key queries, q_i is the number of encryption queries for the i-th slot, $f_1, \dots, f_{q_0} \in \mathcal{F}_{\mathsf{prm}}$ and $x_{i,b}^{j_i} \in \mathcal{X}_{\mathsf{prm}}$ for all $i \in [n], b \in \{0,1\}, j_i \in [q_i]$. We say that the prMIFE scheme satisfies *κ-IND-security* if for every PPT sampler Samp such that

$$f_k(x_{1,0}^{j_1}, \dots, x_{n,0}^{j_n}) = f_k(x_{1,1}^{j_1}, \dots, x_{n,1}^{j_n}) \qquad \forall j_1 \in [q_1], \dots, \forall j_n \in [q_n], \forall k \in [q_0]$$

and

$$\left(1^\kappa, \left\{f_k, f_k(x_1^{j_1}, \dots, x_n^{j_n})\right\}_{k\in[q_0], i\in[n], j_i\in[q_i]}, \mathsf{aux}\right) \approx_c \left(1^\kappa, \left\{f_k, \Delta_{k,j_1,\dots,j_n}\right\}_{k\in[q_0], i\in[n], j_i\in[q_i]}, \mathsf{aux}\right),$$

we have

$$\left(\mathsf{mpk}, \left\{f_k, \mathsf{sk}_{f_k}\right\}_{k\in[q_0]}, \left\{\mathsf{ct}_{i,0}^{j_i}\right\}_{i\in[n], j_i\in[q_i]}, \mathsf{aux}\right) \approx_c \left(\mathsf{mpk}, \left\{f_k, \mathsf{sk}_{f_k}\right\}_{k\in[q_0]}, \left\{\mathsf{ct}_{i,1}^{j_i}\right\}_{i\in[n], j_i\in[q_i]}, \mathsf{aux}\right),$$

where $\kappa \geq \lambda^n$, $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, \mathsf{prm})$, $\mathsf{sk}_{f_k} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f_k)$, $\mathsf{ct}_{i,b}^{j_i} \leftarrow \mathsf{Enc}(\mathsf{msk}, x_{i,b}^{j_i})$ for $i \in [n]$, $j_i \in [q_i]$, $b \in \{0,1\}$, and $k \in [q_0]$. We note that κ-IND security above is weaker than the IND-based security defined for MIFE in [AJ15], since it requires the ciphertext indistinguishability (i.e., Equation (34))to hold only when we have pseudorandom condition for the decryption results (i.e., Equation (33)). Note that the additional parameter κ in our definition is used to parametrize the running time of the distinguisher in the pseudorandomness condition. Since this is an additional condition imposed by the pseudorandomness requirement, the IND-based definition of [AJ15] does not need it.

Construction. Next, we describe our construction for bounded depth prMIFE using a bounded depth prFE and a secret-key encryption scheme. Our construction adapts the key idea from [AJ15], of "unrolling" ciphertexts on the fly via recursive decryption. However, since our security notion is quite different, our proof departs significantly from theirs, as we will discuss below.

Specifically, we use n instances of a single-input prFE scheme $\{\mathsf{prFE}_i\}_{i\in[n]}$, with appropriate input lengths, to build a n arity prMIFE scheme where the i-th encryption algorithm Enc_i outputs the prFE_{i+1} functional secret-key, $\mathsf{prFE}_{i+1}.\mathsf{sk}$, for $i \in [n-1]$ and Enc_n outputs a ciphertext corresponding to prFE_n scheme. Here the $\mathsf{prFE}_{i+1}.\mathsf{sk}$ contains the input $\mathbf{x}_i$ hardcoded within itself, wrapped in an SKE scheme, since prFE does not support function hiding. It computes the ciphertext $\mathsf{prFE}_i.\mathsf{ct}$ for the input $(\mathsf{SKE.sk}, \mathbf{x}_i, \mathbf{x}_{i+1}, \dots, \mathbf{x}_{n-1}, \mathbf{x}_n)$ decryptable by $\mathsf{prFE}_i.\mathsf{sk}$ which in turn computes the ciphertext $\mathsf{prFE}_{i-1}.\mathsf{ct}$ decryptable by $\mathsf{prFE}_{i-1}.\mathsf{sk}$ and so on. Now, note that the decryption of slot n ciphertext with slot $n-1$ functional secret-key will give us a ciphertext decryptable by functional key at slot $n-2$. Unrolling upto slot 1, we get a ciphertext, $\mathsf{prFE}_1.\mathsf{ct}$, corresponding to prFE_1 scheme. Finally, the key generation algorithm outputs a functional secret-key for prFE_1 which together with $\mathsf{prFE}_1.\mathsf{ct}$ will give us the desired output.
In more detail[3],

[2]Recall that $\mathcal{A}_1$ is a PPT algorithm. This means that it runs in polynomial time in its input length. Here, κ serves as a "padding", which artificially makes the input longer and allows $\mathcal{A}_1$ to run in longer time.

[3]We omit substantial notation here for the ease of readability.

1. The setup algorithm generates n instances of prFE scheme $\{\mathsf{prFE}_i.\mathsf{mpk}, \mathsf{prFE}_i.\mathsf{msk}\}$ for appropriate input lengths and a secret key for SKE scheme SKE.sk. It outputs $\mathsf{mpk} = (\{\mathsf{prFE}_i.\mathsf{mpk}\}_{i\in[n]})$ and $\mathsf{msk} = (\mathsf{SKE.sk}, \{\mathsf{prFE}_i.\mathsf{msk}, \mathsf{prFE}_i.\mathsf{mpk}\}_{i\in[n]})$.

2. The keygen algorithm on input msk and a function $f : (\{0,1\}^{\mathsf{L}})^n \to \{0,1\}$, computes $\mathsf{prFE}_1.\mathsf{sk}_f \leftarrow \mathsf{prFE}_1.\mathsf{KeyGen}(\mathsf{prFE}_1.\mathsf{msk}, f)$ and outputs $\mathsf{sk}_f := \mathsf{prFE}_1.\mathsf{sk}_f$.

3. The i-th encryption algorithm on input $(\mathsf{msk}, \mathbf{x}_i \in \{0,1\}^{\mathsf{L}})$, parses $\mathsf{msk} = (\mathsf{SKE.sk}, \{\mathsf{prFE}_i.\mathsf{msk}, \mathsf{prFE}_i.\mathsf{mpk}\}_{i\in[n]})$ and does as follows. If $i \in [n-1]$

   – Compute $\mathsf{SKE.ct}_i \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.sk}, \mathbf{x}_i)$.

   – Define function $\mathrm{F}_i := \mathrm{F}_i[\mathsf{SKE.ct}_i, \mathsf{prFE}_i.\mathsf{mpk}]$ which on input $(\mathsf{SKE.sk}, \mathbf{x}_{i+1}, \ldots, \mathbf{x}_{n-1}, \mathbf{x}_n)$ first computes $\mathbf{x}_i = \mathsf{SKE.Dec}(\mathsf{SKE.sk}, \mathsf{SKE.ct}_i)$ and then computes a $\mathsf{prFE}_i.\mathsf{ct}$ encoding $(\mathsf{SKE.sk}, \mathbf{x}_i, \mathbf{x}_{i+1}, \ldots, \mathbf{x}_{n-1}, \mathbf{x}_n)$ if $i \neq 1$ else it computes $\mathsf{prFE}_1.\mathsf{ct}$ encoding $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{n-1}, \mathbf{x}_n)$[4]. It outputs $\mathsf{prFE}_i.\mathsf{ct}$.

   – It computes a functional key for F_i using the $i+1$-th instance of prFE and outputs it as the i-th ciphertext, i.e., $\mathsf{ct}_i := \mathsf{prFE}_{i+1}.\mathsf{sk} \leftarrow \mathsf{prFE}_{i+1}.\mathsf{KeyGen}(\mathsf{prFE}_{i+1}.\mathsf{msk}, \mathrm{F}_i)$.

   If $i = n$, it outputs $\mathsf{ct}_n := \mathsf{prFE}_n.\mathsf{ct} \leftarrow \mathsf{prFE}_n.\mathsf{Enc}(\mathsf{prFE}_n.\mathsf{mpk}, (\mathsf{SKE.sk}, \mathbf{x}_n))$.

4. The decryption algorithm on input $\mathsf{sk}_f = \mathsf{prFE}_1.\mathsf{sk}_f$, and ciphertexts $\mathsf{ct}_i = \mathsf{prFE}_{i+1}.\mathsf{sk}$ for $i \in [n-1]$, and $\mathsf{ct}_n = \mathsf{prFE}_n.\mathsf{ct}$ does the following: (a) Iteratively compute $\mathsf{prFE}_{i-1}.\mathsf{ct}$ for $i \in [2, n]$ by decrypting $\mathsf{prFE}_i.\mathsf{ct}$ with $\mathsf{prFE}_i.\mathsf{sk}$ starting with $i = n$. (b) Compute and output $\mathbf{y} \leftarrow \mathsf{prFE}_1.\mathsf{Dec}(\mathsf{prFE}_1.\mathsf{mpk}, \mathsf{prFE}_1.\mathsf{sk}_f, f, \mathsf{prFE}_1.\mathsf{ct})$.

Correctness follows from the correctness of underlying ingredients. To see this, note that by the correctness of prFE_n and the definition of F_{n-1}, we have $\mathsf{prFE}_n.\mathsf{Dec}(\mathsf{prFE}_n.\mathsf{mpk}, \mathsf{prFE}_n.\mathsf{sk}, \mathrm{F}_{n-1}, \mathsf{prFE}_n.\mathsf{ct})$ will output $\mathrm{F}_{n-1}(\mathsf{SKE.sk}, \mathbf{x}_n)$ correctly. Next, by the correctness of the SKE scheme, we have $\mathbf{x}_{n-1} = \mathsf{SKE.Dec}(\mathsf{SKE.sk}, \mathsf{SKE}, \mathsf{ct}_{n-1})$ thus $\mathrm{F}_{n-1}(\mathsf{SKE.sk}, \mathbf{x}_n) = \mathsf{prFE}_{n-1}.\mathsf{ct}$ where $\mathsf{prFE}_{n-1}.\mathsf{ct}$ encodes $(\mathsf{SKE.sk}, \mathbf{x}_{n-1}, \mathbf{x}_n)$. Unrolling as in decryption step (a), we get $\mathsf{prFE}_1.\mathsf{ct}$ which encodes $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{n-1}, \mathbf{x}_n)$. Now, from step (b) of decryption and correctness of prFE_1 scheme we have $\mathsf{prFE}_1.\mathsf{Dec}(\mathsf{prFE}_1.\mathsf{mpk}, \mathsf{prFE}_1.\mathsf{sk}_f, f, \mathsf{prFE}_1.\mathsf{ct}) = f(\mathbf{x}_1, \ldots, \mathbf{x}_n)$.

prCT *Security.* While the key-idea of our construction is adapted from [AJ15], our security proof differs significantly from theirs as we elaborate next. Consider the initial view of an adversary $\mathcal{A}$ which outputs q_0 key queries $\{f_1, \ldots, f_{q_0}\}$, q_i input queries for i-th slot $\{x_1^{j_1}\}_{j_1\in[q_1]}, \ldots, \{x_n^{j_n}\}_{j_n\in[q_n]}$ and auxiliary information $\mathsf{aux}_{\mathcal{A}}$

$$\mathcal{D}_{0,0} : \left(\begin{array}{c} \mathsf{aux}_{\mathcal{A}},\ \{\mathsf{prFE}_i.\mathsf{mpk}\}_{i\in[n]},\ \left\{f_k,\ \mathsf{sk}_{f_k} = \mathsf{prFE}_1.\mathsf{sk}_{f_k}\right\}_{k\in[q_0]} \\ \left\{\mathsf{ct}_i^{j_i} = \mathsf{SKE.ct}_i^{j_i},\ \mathsf{prFE}_{i+1}.\mathsf{sk}^{j_i}\right\}_{\substack{i\in[n-1],\\ j_i\in[q_i]}}, \\ \left\{\mathsf{ct}_n^{j_n} = \mathsf{prFE}_n.\mathsf{ct}^{j_n}\right\}_{j_n\in[q_n]} \end{array} \right) \tag{3}$$

To prove security we design a simulator Sim as follows: On input $\mathsf{msk} = (\mathsf{SKE.sk}, \{\mathsf{prFE}_i.\mathsf{msk}, \mathsf{prFE}_i.\mathsf{mpk}\}_{i\in[n]})$

- for $i \in [n-1]$, it first samples a random SKE ciphertext γ_i from the ciphertext space of SKE scheme, defines $\mathrm{F}_i[\gamma_i, \mathsf{prFE}_i.\mathsf{mpk}]$ as in the construction and outputs $\mathsf{ct}_i = \mathsf{prFE}_{i+1}.\mathsf{sk} \leftarrow \mathsf{prFE}_{i+1}.\mathsf{KeyGen}(\mathsf{prFE}_{i+1}.\mathsf{msk}, \mathrm{F}_i[\gamma_i, \mathsf{prFE}_i.\mathsf{mpk}])$.
- for $i = n$, it outputs a randomly sampled ct_n from the ciphertext space of prFE_n scheme.

Given the above simulator it suffices to show that Equation (3) is indistinguishable from the following distribution

$$\mathcal{D}_{0,1} : \left(\begin{array}{c} \mathsf{aux}_{\mathcal{A}},\ \{\mathsf{prFE}_i.\mathsf{mpk}\}_{i\in[n]},\ \left\{f_k,\ \mathsf{sk}_{f_k} = \mathsf{prFE}_1.\mathsf{sk}_{f_k}\right\}_{k\in[q_0]} \\ \left\{\mathsf{ct}_i^{j_i} = \gamma_i^{j_i} \leftarrow \mathcal{CT}_{\mathsf{SKE}},\ \mathsf{prFE}_{i+1}.\mathsf{sk}^{j_i}\right\}_{\substack{i\in[n-1],\\ j_i\in[q_i]}}, \\ \left\{\delta^{j_n} \leftarrow \mathcal{CT}_{\mathsf{prFE}_n}\right\}_{j_n\in[q_n]} \end{array} \right) \tag{4}$$

[4]The randomness for computing the ciphertexts comes from a PRF, which we omit here in the overview.

At a high level, the security proof proceeds as follows. To prove the pseudorandomness of $\{\mathsf{prFE}_n.\mathsf{ct}^{j_n}\}_{j_n}$, we show that the decryption results of these ciphertexts using the secret keys $\{\mathsf{prFE}_n.\mathsf{sk}^{j_{n-1}}\}_{j_{n-1}}$ are all pseudorandom. This allows us to invoke the security of prFE_n. The decryption results of the above ciphertexts using the secret keys are ciphertexts $\{\mathsf{prFE}_{n-1}.\mathsf{ct}^{j_{n-1},j_n}\}_{j_{n-1},j_n}$ of prFE_{n-1} and we would like to prove the pseudorandomness of them. We again consider the decryption results of the ciphertexts using the secret keys $\{\mathsf{prFE}_{n-1}.\mathsf{sk}^{j_{n-2}}\}_{j_{n-2}}$. This process continues until we reach the point where we have to prove the pseudorandomness of $\{\mathsf{prFE}_1.\mathsf{ct}^{j_1,\cdots,j_n}\}_{j_1,\ldots,j_n}$, where each ciphertext encodes $(\mathbf{x}_1^{j_1},\ldots,\mathbf{x}_n^{j_n})$. By invoking the security of prFE_1 once again, we can conclude that it suffices to show that $\{f_k(\mathbf{x}_1^{j_1},\ldots,\mathbf{x}_n^{j_n})\}_{k,j_1,\ldots,j_n}$ are pseudorandom even given SKE ciphertexts encrypting each $\mathbf{x}_i^{j_i}$, where the latter is dealt as auxiliary information throughout the process of the above recursive invocations of prFE security. We then invoke the security of SKE to erase the information of $\mathbf{x}_i^{j_i}$ from the SKE ciphertexts. This allows us to conclude, since the pseudorandomness of $\{f_k(\mathbf{x}_1^{j_1},\ldots,\mathbf{x}_n^{j_n})\}_{k,j_1,\ldots,j_n}$ directly follows from the precondition.

A bit more formally, to prove Equation (3)$\approx$ Equation (4) we begin by invoking the security of prFE_n with sampler that provides inputs $\{(\mathsf{SKE.sk},\mathbf{x}_n^{j_n})\}_{j_n\in[q_n]}$, functions $\{\mathrm{F}_{n-1}^{j_{n-1}}[\mathsf{SKE.ct}_{n-1}^{j_{n-1}},\mathsf{prFE}_{n-1}.\mathsf{mpk}]\}_{j_{n-1}\in[q_{n-1}]}$, and all the remaining components of Equation (3) as auxiliary information. Now, from the security guarantee of prFE_n with the sampler, we know that to prove $\mathsf{prFE.ct}_n^{j_n}$ is pseudorandom it suffices to show function output

$$\mathrm{F}_{n-1}^{j_{n-1}}[\mathsf{SKE.ct}_{n-1}^{j_{n-1}},\mathsf{prFE}_{n-1}.\mathsf{mpk}](\mathsf{SKE.sk},\mathbf{x}_n^{j_n}) = \mathsf{prFE}_{n-1}.\mathsf{Enc}(\mathsf{prFE}_{n-1}.\mathsf{mpk},(\mathsf{SKE.sk},\mathbf{x}_{n-1}^{j_{n-1}},\mathbf{x}_n^{j_n})) = \mathsf{prFE}_{n-1}.\mathsf{ct}^{j_{n-1},j_n}$$

is pseudorandom for all $j_{n-1}\in[q_{n-1}], j_n\in[q_n]$. Thus it suffices to show

$$\mathcal{D}_{1,0}:\left(\begin{array}{c}\mathsf{aux}_{\mathcal{A}},\ \{\mathsf{prFE}_i.\mathsf{mpk}\}_{i\in[n-1]},\ \left\{f_k,\ \mathsf{sk}_{f_k}=\mathsf{prFE}_1.\mathsf{sk}_{f_k}\right\}_{k\in[q_0]},\\ \left\{\mathsf{SKE.ct}_i^{j_i}\right\}_{\substack{i\in[n-1],\\ j_i\in[q_i]}},\left\{\mathsf{prFE}_{i+1}.\mathsf{sk}^{j_i}\right\}_{\substack{i\in[n-2],\\ j_i\in[q_i]}},\left\{\mathsf{prFE}_{n-1}.\mathsf{ct}^{j_{n-1},j_n}\right\}_{\substack{j_{n-1}\in[q_{n-1}],\\ j_n\in[q_n]}}\end{array}\right)$$
$$\approx$$
$$\mathcal{D}_{1,1}:\left(\begin{array}{l}\mathsf{aux}_{\mathcal{A}},\ \{\mathsf{prFE}_i.\mathsf{mpk}\}_{i\in[n-1]},\ \left\{f_k,\ \mathsf{sk}_{f_k}=\mathsf{prFE}_1.\mathsf{sk}_{f_k}\right\}_{k\in[q_0]},\\ \quad\left\{\gamma_i^{j_i}\right\}_{\substack{i\in[n-1],\\ j_i\in[q_i]}},\left\{\mathsf{prFE}_{i+1}.\mathsf{sk}^{j_i}\right\}_{\substack{i\in[n-2],\\ j_i\in[q_i]}},\left\{\delta^{j_{n-1},j_n}\right\}_{\substack{j_{n-1}\in[q_{n-1}],\\ j_n\in[q_n]}}\end{array}\right)$$

where $\gamma_i^{j_i}\leftarrow\mathcal{CT}_{\mathsf{SKE}}$, $\delta^{j_{n-1},j_n}\leftarrow\mathcal{CT}_{\mathsf{prFE}_{n-1}}$, and $\mathcal{CT}_{\mathsf{SKE}}$ and $\mathcal{CT}_{\mathsf{prFE}_{n-1}}$ denotes the ciphertext space of the SKE scheme and prFE_{n-1} scheme, respectively. Recursively invoking the security of prFE_i for $i=n-1,\ldots,2$, it suffices to show the following:

$$\mathcal{D}_{n-1,0}:\left(\begin{array}{l}1^{\kappa},\ \mathsf{aux}_{\mathcal{A}},\ \mathsf{prFE}_1.\mathsf{mpk},\ \left\{f_k,\ \mathsf{sk}_{f_k}=\mathsf{prFE}_1.\mathsf{sk}_{f_k}\right\}_{k\in[q_0]},\\ \quad\left\{\mathsf{SKE.ct}_i^{j_i}\right\}_{\substack{i\in[n-1],\\ j_i\in[q_i]}},\left\{\mathsf{prFE}_1.\mathsf{ct}^{j_1,\ldots,j_n}\right\}_{j_1\in[q_1],\ldots,j_n\in[q_n]}\end{array}\right)$$
$$\approx$$
$$\mathcal{D}_{n-1,1}:\left(\begin{array}{c}1^{\kappa},\mathsf{aux}_{\mathcal{A}},\ \mathsf{prFE}_1.\mathsf{mpk},\ \left\{f_k,\ \mathsf{sk}_{f_k}=\mathsf{prFE}_1.\mathsf{sk}_{f_k}\right\}_{k\in[q_0]},\\ \left\{\gamma_i^{j_i}\leftarrow\mathcal{CT}_{\mathsf{SKE}}\right\}_{\substack{i\in[n-1],\\ j_i\in[q_i]}},\left\{\delta^{j_1,\ldots,j_n}\leftarrow\mathcal{CT}_{\mathsf{prFE}_1}\right\}_{j_1\in[q_1],\ldots,j_n\in[q_n]}\end{array}\right).$$

Finally, applying the security of prFE_1 once again, we can see that it suffices to show the following:

$$
\begin{aligned}
\mathcal{D}_{n,0}: \quad & \left(1^{\kappa}, \mathsf{aux}_{\mathcal{A}}, \left\{f_k,\ f_k(\mathbf{x}_1^{j_1}, \ldots, \mathbf{x}_n^{j_n})\right\}_{k, j_1, \ldots, j_n}, \left\{\mathsf{SKE.ct}_i^{j_i} \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.sk}, \mathbf{x}_i^{j_i})\right\}_{i, j_i}\right) \\
\approx_c & \\
\mathcal{D}_{n,1}: \quad & \left(1^{\kappa}, \mathsf{aux}_{\mathcal{A}}, \left\{\ f_k,\ \Delta_k^{j_1,\ldots,j_n} \leftarrow \{0,1\}\right\}_{k, j_1, \ldots, j_n},\ \left\{\gamma_i^{j_i} \leftarrow \mathcal{CT}_{\mathsf{SKE}}\right\}_{i, j_i}\right) \qquad (5)
\end{aligned}
$$

Here, 1^{κ} appearing in the above distributions is introduced for compensating the blow up of the size of the adversary caused by the multiple invocations of the prFE security– see Section 7 for details. To prove Equation (5), we first invoke the security of SKE scheme to show the pseudorandomness of SKE ciphertexts. This erases the information of $\mathbf{x}_i^{j_i}$ from $\mathsf{SKE.ct}_i^{j_i}$. Next, we use the fact that the functionality supported by our scheme is pseudorandom to argue that the function values $f_k(\mathbf{x}_1^{j_1}, \ldots, \mathbf{x}_n^{j_n})$ are pseudorandom.

Subtleties in the proof. The high level overview presented above hides many important details, and indeed, as stated is not secure. There are two important subtleties that arise when making the formal argument, and these are so significant that they require us to strengthen the underlying evasive LWE assumption. Moreover, to the best of our understanding, these issues also arise in prior work [VWW22] and fixing them there also requires to strengthen the evasive LWE assumption. Please see Section 7.3for details. We obtain the following theorem.

Theorem 2.5 (prMIFE **for poly arity).** Assume suitably parametrized private coin evasive LWE, non-uniform sub-exponential PRF, and non-uniform sub-exponential LWE. Then there exists a prMIFE scheme for arity $n = \mathrm{poly}(\lambda)$, supporting functions with input length L and bounded polynomial depth $d = d(\lambda)$, and satisfying security (Definition 7.2).

For constant arity, we can rely on a weaker variant of private coin evasive LWE , please see Section 7 for details.

2.3.1 Multi-Input Predicate Encryption.

Our construction can bootstrap a single-input PE scheme to a polynomial-input one generically by simply using an prMIFE to generate PE ciphertext using randomness jointly chosen by the encryptors. The PE must have pseudorandom ciphertext so as to be suitable for the compiler but this is a relatively mild property and readily satisfied by known constructions [GVW15b]. In more detail, our (simplified) construction works as follows.

– The setup generates a prMIFE instance $(\mathsf{prMIFE.msk}, \mathsf{prMIFE.mpk})$ and a single-input PE instance $(\mathsf{PE.msk}, \mathsf{PE.mpk})$. It outputs $\mathsf{msk} = (\mathsf{prMIFE.msk}, \mathsf{PE.msk})$ and $\mathsf{mpk} = (\mathsf{prMIFE.mpk}, \mathsf{PE.mpk})$.

– The i-th slot encryption algorithm on input $(\mathsf{msk}, \mathbf{x}_i, \mu_i)$ generates an i-th slot prMIFE ciphertext $\mathsf{prMIFE.ct}_i \leftarrow \mathsf{prMIFE.Enc}_i(\mathsf{prMIFE.msk}, (\mathbf{x}_i, \mu_i))$. It outputs $\mathsf{ct}_i = \mathsf{prMIFE.ct}_i$

– The key generator on input msk and a function f generates a single-input PE functional secret key $\mathsf{PE.sk}_f \leftarrow \mathsf{PE.KeyGen}(\mathsf{PE.msk}, f)$. It also generates a prMIFE key, $\mathsf{prMIFE.sk}_{\mathrm{F}}$, for function $\mathrm{F}[\mathsf{PE.mpk}]$ that, on input n attribute-message pairs $(\mathbf{x}_1, \mu_1), \ldots, (\mathbf{x}_n, \mu_n)$, generates a single-input PE ciphertext w.r.t. attribute $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ and message $\mu = (\mu_1, \ldots, \mu_n)$. It outputs $\mathsf{sk}_f = (\mathsf{PE.sk}_f, \mathsf{prMIFE.sk}_{\mathrm{F}})$.

– The decryption algorithm first runs the prMIFE decryption using $\mathsf{prMIFE.sk}_{\mathrm{F}}$ and $\{\mathsf{ct}_i = \mathsf{prMIFE.ct}_i\}_{i \in [n]}$ to compute the single-input PE ciphertext, PE.ct, encoding message $\mu = (\mu_1, \ldots, \mu_n)$ w.r.t attribute $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$. Finally it performs PE decryption using $\mathsf{PE.sk}_f$ and PE.ct.

Correctness follow readily from those of the underlying building blocks. Intuitively, the security follows since the decryption result obtained by each combination of the prMIFE ciphertexts forms a PE ciphertext, which is pseudorandom. Therefore, we can invoke the security of the prMIFE to prove the security of multi-input PE. The actual construction and proof are more complicated since we only assume κ-IND security for prMIFE and for proving the security we need the machinery of the Trojan method [ABSV15]. Please see Section 8 for details.

2.4 prIO & Applications

We now use our multi-input FE to construct iO, à la [GGG+14] for the same functionality. We begin with the definition of iO for pseudorandom functionalities, which we refer to as prIO. The syntax of prIO is as in regular iO, where we have (i) an obfuscation algorithm which takes as input the security parameter λ and a circuit C and outputs an obfuscated circuit $\tilde{C}$, (ii) an evaluation algorithm takes as input an obfuscated circuit $\tilde{C}$ and an input x. It outputs $y = C(x)$. We also require that the evaluation time of the obfuscated circuit be only polynomially slower than the run time of the circuit C on x.

κ-IND-Secure prIO. Our notion of security is specified as follows. For the security parameter $\lambda = \lambda(\lambda)$, let Samp be a PPT algorithm that on input 1^λ, outputs

$$(C_0, C_1,\ \mathsf{aux} \in \{0,1\}^*)$$

where $C_0 : \{0,1\}^n \to \{0,1\}^m$ and $C_1 : \{0,1\}^n \to \{0,1\}^m$ have the same description size. We then require that

$$\begin{aligned}
&\text{If } \left(1^\kappa, \{C_0(x)\}_{x\in\{0,1\}^n}, \mathsf{aux}\right) \approx_c \left(1^\kappa, \{\Delta_x \leftarrow \{0,1\}^m\}_{x\in\{0,1\}^n}, \mathsf{aux}\right)\\
&\text{and}\quad C_0(x) = C_1(x) \quad \forall x \in \{0,1\}^n,\\
&\text{then}\ \left(\mathsf{iO}(1^\lambda, C_0), \mathsf{aux}\right) \approx_c \left(\mathsf{iO}(1^\lambda, C_1), \mathsf{aux}\right)
\end{aligned}$$

where the parameter κ above is introduced to adjust the strength of the requirement for the precondition, similarly to the case of prMIFE. Roughly speaking, the above security definition says that the obfuscations of two circuits with the same truth tables are indistinguishable, if the (entire) truth table is pseudorandom.

Our construction follows the blueprint of the multi-input FE to iO conversion by [GGG+14]. Briefly, the obfuscation of a circuit $C : \{0,1\}^n \to \{0,1\}^m$ using a $(n+1)$ input prMIFE scheme is

$$\{\mathsf{prMIFE.sk}_U, \mathsf{prMIFE.ct}_{1,0}, \mathsf{prMIFE.ct}_{1,1}, \ldots, \mathsf{prMIFE.ct}_{n,0}, \mathsf{prMIFE.ct}_{n,1}, \mathsf{prMIFE.ct}_{n+1,C}\}$$

where $\mathsf{prMIFE.sk}_U$ is the prMIFE functional key corresponding to the universal circuit U such that $U(x_1, \ldots, x_n, C) = C(x_1, \ldots, x_n)$, $\mathsf{prMIFE.ct}_{i,b}$ for $i \in [n], b \in \{0,1\}$ denotes the i-th slot prMIFE ciphertext encoding bit b, and $\mathsf{prMIFE.ct}_{n+1,C}$ denotes the $(n+1)$-th slot prMIFE ciphertext encoding the circuit C. The evaluation algorithm on input $\mathbf{x} = (x_1, \ldots, x_n)$ runs $\mathsf{prMIFE.Dec}(\mathsf{prMIFE.sk}_U, \mathsf{prMIFE.ct}_{1,x_1}, \ldots, \mathsf{prMIFE.ct}_{n,x_n}, \mathsf{prMIFE.ct}_{n+1,C})$.

Correctness as well as security immediately follow from those of prMIFE. Roughly speaking, by the condition that the truth table of the circuit is pseudorandom, we can invoke the security of the prMIFE to show the security. This leads to the following theorem, shown in Section 9.

Theorem 2.6. Assuming κ-IND-secure prMIFE, we have κ-IND-secure prIO with the same κ.

Instantiating the Random Oracle. In an elegant work [HSW14], Hohenberger, Sahai and Waters posed the following question: "Can we instantiate the random oracle with an actual family of hash functions for existing cryptographic schemes in the random oracle model, such as Full Domain Hash signatures?" They then demonstrated that the selective security of the full-domain hash (FDH) signature based on trapdoor permutations (TDP) [BR93], the adaptive security of RSA FDH signatures [Cor00], the selective security of BLS signatures, and the adaptive security of BLS signatures [BLS01] can be proven in the standard model by carefully instantiating the underlying hash function by iO for each application.

We show in Section 10.1, that the random oracle in the FDH signature can be instantiated using prIO instead of full-fledged iO. Similarly, we can instantiate the random oracle in selectively secure BLS signatures with prIO, following a strategy similar to that in [HSW14]. At a high level, these proofs follow those in the random oracle model (ROM), where we use iO to obfuscate a derandomized version of the simulator for the hash function in ROM-based proofs. In these settings, the truth table of the simulated hash function is pseudorandom, allowing us to follow the same proof strategy using prIO. Please see Section 10 for details.

3 Preliminaries

In this section we define the notation and preliminaries used in our work.

Notation. We use bold letters to denote vectors and the notation $[a, b]$ to denote the set of integers $\{k \in \mathbb{N} \mid a \leq k \leq b\}$. We use $[n]$ to denote the set $[1, n]$. Concatenation is denoted by the symbol $\|$. We say a function $f(n)$ is *negligible* if it is $O(n^{-c})$ for all $c > 0$, and we use $\mathsf{negl}(n)$ to denote a negligible function of n. We say $f(n)$ is *polynomial* if it is $O(n^c)$ for some constant $c > 0$, and we use $\mathrm{poly}(n)$ to denote a polynomial function of n. We use the abbreviation PPT for probabilistic polynomial-time. We say an event occurs with *overwhelming probability* if its probability is $1 - \mathsf{negl}(n)$. For two distributions X_λ and Y_λ, $X_\lambda \approx_c Y_\lambda$ (resp., $X_\lambda \approx_s Y_\lambda$) denotes that they are computationally indistinguishable for any PPT algorithm (resp., statistically indistinguishable). We write $X_\lambda \equiv Y_\lambda$ when these distributions are the same. Note that when we say two distributions are computational indistinguishable, this means that the two distributions cannot be distinguished with non-negligible advantage in the input length of the adversary (rather than the security parameter λ), whose size is polynomial in its input length. For example, if the output length of the distributions is subexponential in λ, this means that the adversary is allowed to run in subexponential time and the advantage should be subexponentially small. For a vector $\mathbf{x}$, we let x_i denote its i-th entry. For a set S, we let $|S|$ denote the number of elements in S. For a binary string x, we let $|x|$ denote the length of x. For a vector $\mathbf{x}$, we let x_i denote its i-th entry. For a set S, we let $|S|$ denote the number of elements in S. For a binary string x, we let $|x|$ denote the length of x.

3.1 Lattice Preliminaries

Here, we recall some facts on lattices that are needed for the exposition of our construction. Throughout this section, n, m, and q are integers such that $n = \mathrm{poly}(\lambda)$ and $m \geq n\lceil \log q \rceil$. In the following, let $\mathsf{SampZ}(\gamma)$ be a sampling algorithm for the truncated discrete Gaussian distribution over $\mathbb{Z}$ with parameter $\gamma > 0$ whose support is restricted to $z \in \mathbb{Z}$ such that $|z| \leq \sqrt{n}\gamma$.

Let

$$\mathbf{g} = (1, 2, \ldots .2^{\lfloor \log q \rfloor})^\intercal\ , \quad \mathbf{G} = \mathbf{I}_n \otimes \mathbf{g}^\intercal$$

be the gadget vector and the gadget matrix. For $\mathbf{p} \in \mathbb{Z}_q^n$, we write $\mathbf{G}^{-1}(\mathbf{p})$ for the m-bit vector $(\mathsf{bits}(\mathbf{p}[1]), \ldots, \mathsf{bits}(\mathbf{p}[n]))^\intercal$, where $\mathsf{bits}(\mathbf{p}[i])$ are m/n bits for each $i \in [n]$. The notation extends column-wise to matrices and it holds that $\mathbf{G}\mathbf{G}^{-1}(\mathbf{P}) = \mathbf{P}$.

Trapdoors. Let us consider a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$. For all $\mathbf{V} \in \mathbb{Z}_q^{n \times m'}$, we let $\mathbf{A}^{-1}(\mathbf{V})$ be an output distribution of $\mathsf{SampZ}(\gamma)^{m \times m'}$ conditioned on $\mathbf{A} \cdot \mathbf{A}^{-1}(\mathbf{V}, \gamma) = \mathbf{V}$. A γ-trapdoor for $\mathbf{A}$ is a trapdoor that enables one to sample from the distribution $\mathbf{A}^{-1}(\mathbf{V}, \gamma)$ in time $\mathrm{poly}(n, m, m', \log q)$ for any $\mathbf{V}$. We slightly overload notation and denote a γ-trapdoor for $\mathbf{A}$ by $\mathbf{A}_\gamma^{-1}$. The following properties had been established in a long sequence of works [GPV08, CHKP10, ABB10a, ABB10b, MP12, BLP+13].

Lemma 3.1 (Properties of Trapdoors). Lattice trapdoors exhibit the following properties.

1. Given $\mathbf{A}_\tau^{-1}$, one can obtain $\mathbf{A}_{\tau'}^{-1}$ for any $\tau' \geq \tau$.
2. Given $\mathbf{A}_\tau^{-1}$, one can obtain $[\mathbf{A}\|\mathbf{B}]_\tau^{-1}$ and $[\mathbf{B}\|\mathbf{A}]_\tau^{-1}$ for any $\mathbf{B}$.
3. There exists an efficient procedure $\mathsf{TrapGen}(1^n, 1^m, q)$ that outputs $(\mathbf{A}, \mathbf{A}_{\tau_0}^{-1})$ where $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ for some $m = O(n \log q)$ and is 2^{-n}-close to uniform, where $\tau_0 = \omega(\sqrt{n \log q \log m})$.

Useful Lemmata.

Lemma 3.2 (tail and truncation of $\mathcal{D}_{\mathbb{Z},\gamma}$). There exists $B_0 \in \Theta(\sqrt{\lambda})$ such that

$$\Pr\Big[x \leftarrow \mathcal{D}_{\mathbb{Z},\gamma} : |x| > \gamma B_0(\lambda)\Big] \leq 2^{-\lambda} \quad \text{for all} \quad \gamma \geq 1 \text{ and } \lambda \in \mathbb{N}.$$

Lemma 3.3 (Smudging Lemma [WWW22]). Let λ be a security parameter. Take any $a \in \mathbb{Z}$ where $|a| \leq B$. Suppose $\gamma \geq B\lambda^{\omega(1)}$. Then the statistical distance between the distributions $\{z : z \leftarrow \mathcal{D}_{\mathbb{Z},\gamma}\}$ and $\{z + a : z \leftarrow \mathcal{D}_{\mathbb{Z},\gamma}\}$ is $\mathsf{negl}(\lambda)$.

Lemma 3.4 (Leftover Hash Lemma). Fix some $n, m, q \in \mathbb{N}$. The leftover hash lemma states that if $m \geq 2n \log q$, then for $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}$, $\mathbf{x} \leftarrow \{0,1\}^m$ and $\mathbf{y} \leftarrow \mathbb{Z}_q^n$ the statistical distance between $(\mathbf{A}, \mathbf{A}\cdot\mathbf{x})$ and $(\mathbf{A}, \mathbf{y})$ is negligible. More concretely, it is bounded by $q^n\sqrt{2^{1-m}}$.

3.1.1 Hardness Assumptions

Assumption 3.5 (The LWE Assumption). Let $n = n(\lambda)$, $m = m(\lambda)$, and $q = q(\lambda) > 2$ be integers and $\chi = \chi(\lambda)$ be a distribution over $\mathbb{Z}_q$. We say that the LWE(n, m, q, χ) hardness assumption holds if for any PPT adversary $\mathcal{A}$ we have

$$|\Pr[\mathcal{A}(\mathbf{A}, \mathbf{s}^\mathsf{T}\mathbf{A} + \mathbf{e}^\mathsf{T}) \to 1] - \Pr[\mathcal{A}(\mathbf{A}, \mathbf{v}^\mathsf{T}) \to 1]| \leq \mathsf{negl}(\lambda)$$

where the probability is taken over the choice of the random coins by the adversary $\mathcal{A}$ and $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}$, $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, $\mathbf{e} \leftarrow \chi^m$, and $\mathbf{v} \leftarrow \mathbb{Z}_q^m$. We also say that LWE$(n, m, q, \chi)$ problem is (non-uniformly and) subexponentially hard if there exists some constant $0 < \delta < 1$ such that the above distinguishing advantage is bounded by 2^{-n^δ} for for all adversaries $\mathcal{A}$ whose running time (resp., size) is 2^{n^δ}.

As shown by previous works [Reg09, BLP$^+$13], if we set $\chi = \mathsf{SampZ}(\gamma)$, the LWE$(n, m, q, \chi)$ problem is as hard as solving worst case lattice problems such as gapSVP and SIVP with approximation factor poly$(n)\cdot(q/\gamma)$ for some poly(n). Since the best known algorithms for 2^k-approximation of gapSVP and SIVP run in time $2^{\tilde{O}(n/k)}$, it follows that the above LWE(n, m, q, χ) with noise-to-modulus ratio 2^{-n^ϵ} is likely to be (subexponentially) hard for some constant ϵ.

Next, we define Evasive LWE assumption, with restricted samplers as described in [AMYY25].

Assumption 3.6 (Evasive LWE). [Wee22, ARYY23, AMYY25] Let $n, m, t, m', q \in \mathbb{N}$ be parameters and λ be a security parameter. Let χ and χ' be parameters for Gaussian distributions. For Samp that outputs

$$\mathbf{S} \in \mathbb{Z}_q^{m'\times n}, \mathbf{P} \in \mathbb{Z}_q^{n\times t}, \mathsf{aux} \in \{0,1\}^*$$

on input 1^λ and for PPT adversaries $\mathcal{A}_0$ and $\mathcal{A}_1$, we define the following advantage functions:

$$\mathsf{Adv}_{\mathcal{A}_0}^{\mathrm{PRE}}(\lambda) \overset{\mathrm{def}}{=} \Pr\big[\mathcal{A}_0(\mathbf{B}, \mathbf{SB} + \mathbf{E}, \mathbf{SP} + \mathbf{E}', \mathsf{aux}) = 1\big] - \Pr\big[\mathcal{A}_0(\mathbf{B}, \mathbf{C}_0, \mathbf{C}', \mathsf{aux}) = 1\big] \tag{6}$$

$$\mathsf{Adv}_{\mathcal{A}_1}^{\mathrm{POST}}(\lambda) \overset{\mathrm{def}}{=} \Pr[\mathcal{A}_1(\mathbf{B}, \mathbf{SB} + \mathbf{E}, \mathbf{K}, \mathsf{aux}) = 1] - \Pr[\mathcal{A}_1(\mathbf{B}, \mathbf{C}_0, \mathbf{K}, \mathsf{aux}) = 1] \tag{7}$$

where

$$(\mathbf{S}, \mathbf{P}, \mathsf{aux}) \leftarrow \mathsf{Samp}(1^\lambda),$$
$$\mathbf{B} \leftarrow \mathbb{Z}_q^{n\times m},$$
$$\mathbf{C}_0 \leftarrow \mathbb{Z}_q^{m'\times m}, \mathbf{C}' \leftarrow \mathbb{Z}_q^{m'\times t},$$
$$\mathbf{E} \leftarrow \mathcal{D}_{\mathbb{Z},\chi}^{m'\times m}, \mathbf{E}' \leftarrow \mathcal{D}_{\mathbb{Z},\chi'}^{m'\times t}$$
$$\mathbf{K} \leftarrow \mathbf{B}^{-1}(\mathbf{P}) \text{ with standard deviation } O(\sqrt{m\log(q)}).$$

We say that the *evasive* LWE (EvLWE) assumption with respect to the sampler class $\mathcal{SC}$ holds if for every PPT $\mathsf{Samp} \in \mathcal{SC}$ and $\mathcal{A}_1$, there exists another PPT $\mathcal{A}_0$ and a polynomial $Q(\cdot)$ such that

$$\mathsf{Adv}_{\mathcal{A}_0}^{\mathrm{PRE}}(\lambda) \geq \mathsf{Adv}_{\mathcal{A}_1}^{\mathrm{POST}}(\lambda)/Q(\lambda) - \mathsf{negl}(\lambda) \quad \text{and} \quad \mathsf{Time}(\mathcal{A}_0) \leq \mathsf{Time}(\mathcal{A}_1)\cdot Q(\lambda). \tag{8}$$

We conjecture that for reasonable class of samplers, the evasive LWE assumption holds. In particular, we conjecture that our sampler $\mathsf{Samp}_{\mathsf{prFE}}(1^\lambda)$ used for the security proof of our prFE for natural class of functions should be in the secure class of samplers $\mathcal{SC}$ for which the evasive LWE holds.

Remark 3.7. In the above definition, all the LWE error terms are chosen from the same distribution $D_{\mathbb{Z},\chi}$. However, in our security proof, we often consider the case where some of LWE error terms are chosen from $D_{\mathbb{Z},\chi}$ and others from $D_{\mathbb{Z},\chi'}$ with different $\chi \gg \chi'$. The evasive LWE assumption with such a mixed noise distribution is implied by the evasive LWE assumption with all LWE error terms being chosen from $D_{\mathbb{Z},\chi}$ as above definition, since if the precondition is satisfied for the latter case, that for the former case is also satisfied. To see this, it suffices to observe that we can convert the distribution from $D_{\mathbb{Z},\chi'}$ into that from $D_{\mathbb{Z},\chi}$ by adding extra Gaussian noise.

In the security proof, we may require the auxiliary information to include terms dependent on $\mathbf{S}$. Furthermore, we may want to prove the pseudorandomness of such auxiliary information. The following lemma from [ARYY23] enables this. In the lemma, we separate the auxiliary information into two parts aux_1 and aux_2, where aux_1 is typically the part dependent on $\mathbf{S}$. The lemma roughly says that aux_1 is pseudorandom in the post condition distribution, if it is pseudorandom in the precondition distribution.

Lemma 3.8 (Lemma 3.4 in [ARYY23]). Let $n, m, t, m', q \in \mathbb{N}$ be parameters and λ be a security parameter. Let χ and χ' be Gaussian parameters. Let Samp be a PPT algorithm that takes as input 1^λ and outputs

$$\mathbf{S} \in \mathbb{Z}_q^{m' \times n}, \mathsf{aux} = (\mathsf{aux}_1, \mathsf{aux}_2) \in \mathcal{S} \times \{0,1\}^* \text{ and } \mathbf{P} \in \mathbb{Z}_q^{n \times t}$$

for some set $\mathcal{S}$. Furthermore, we assume that there exists a public deterministic poly-time algorithm $\mathsf{Reconstruct}$ that allows to derive $\mathbf{P}$ from aux_2, i.e. $\mathbf{P} = \mathsf{Reconstruct}(\mathsf{aux}_2)$.

We introduce the following advantage functions:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{PRE}'}(\lambda) \overset{\mathrm{def}}{=} \Pr[\mathcal{A}(\mathbf{B}, \mathbf{SB}+\mathbf{E}, \mathbf{SP}+\mathbf{E}', \mathsf{aux}_1, \mathsf{aux}_2) = 1] - \Pr[\mathcal{A}(\mathbf{B}, \mathbf{C}_0, \mathbf{C}', \mathbf{c}, \mathsf{aux}_2) = 1] \tag{9}$$

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{POST}'}(\lambda) \overset{\mathrm{def}}{=} \Pr[\mathcal{A}(\mathbf{B}, \mathbf{SB}+\mathbf{E}, \mathbf{K}, \mathsf{aux}_1, \mathsf{aux}_2) = 1] - \Pr[\mathcal{A}(\mathbf{B}, \mathbf{C}_0, \mathbf{K}, \mathbf{c}, \mathsf{aux}_2) = 1] \tag{10}$$

where

$$(\mathbf{S}, \mathsf{aux} = (\mathsf{aux}_1, \mathsf{aux}_2), \mathbf{P}) \leftarrow \mathsf{Samp}(1^\lambda),$$
$$\mathbf{B} \leftarrow \mathbb{Z}_q^{n \times m}$$
$$\mathbf{C}_0 \leftarrow \mathbb{Z}_q^{m' \times m}, \mathbf{C}' \leftarrow \mathbb{Z}_q^{m' \times t}, \mathbf{c} \leftarrow \mathcal{S}$$
$$\mathbf{E} \leftarrow \mathcal{D}_{\mathbb{Z},\chi}^{m' \times m}, \mathbf{E}' \leftarrow \mathcal{D}_{\mathbb{Z},\chi}^{m' \times t}$$
$$\mathbf{K} \leftarrow \mathbf{B}^{-1}(\mathbf{P}) \text{ with standard deviation } O(\sqrt{m \log(q)}).$$

Then, under the Evasive-LWE (cited above in Assumption 3.6) with respect to a sampler $\mathsf{Samp} \in \mathcal{SC}$, for a sampler class $\mathcal{SC}$, if $\mathsf{Adv}_{\mathcal{A}}^{\mathrm{PRE}'}(\lambda)$ is negligible for any PPT adversary $\mathcal{A}$, so is $\mathsf{Adv}_{\mathcal{A}}^{\mathrm{POST}'}(\lambda)$ for any PPT adversary $\mathcal{A}$.

3.1.2 GSW Homomorphic Encryption and Evaluation

We recall the format of the (leveled fully) homomorphic encryption due to [GSW13] and the correctness property. We adapt the syntax from [HLL23].

Lemma 3.9. The leveled FHE scheme works as follows:

- The keys are

$$\text{(public)} \quad \mathbf{A}_{\mathsf{fhe}} = \begin{pmatrix} \bar{\mathbf{A}}_{\mathsf{fhe}} \\ \bar{\mathbf{s}}^{\mathsf{T}} \bar{\mathbf{A}}_{\mathsf{fhe}} + \mathbf{e}_{\mathsf{fhe}}^{\mathsf{T}} \end{pmatrix} \in \mathbb{Z}_q^{(n+1) \times m}, \quad \text{(secret)} \quad \mathbf{s}^{\mathsf{T}} = (\bar{\mathbf{s}}^{\mathsf{T}}, -1),$$

  where $\bar{\mathbf{s}} \in \mathbb{Z}^n$, $\bar{\mathbf{A}}_{\mathsf{fhe}} \in \mathbb{Z}_q^{n \times m}$, and $\mathbf{e}_{\mathsf{fhe}}^{\mathsf{T}} \in \mathbb{Z}^m$.

- A ciphertext of $x \in \{0,1\}$ is $\mathbf{X} = \mathbf{A}_{\mathsf{fhe}}\mathbf{R} - x\mathbf{G} \in \mathbb{Z}_q^{(n+1)\times m}$, where $\mathbf{R} \in \mathbb{Z}^{m\times m}$ is the encryption randomness. The decryption equation is

$$\mathbf{s}^\mathsf{T}\mathbf{X} = -\mathbf{e}_{\mathsf{fhe}}^\mathsf{T}\mathbf{R} - x\mathbf{s}^\mathsf{T}\mathbf{G} \in \mathbb{Z}_q^m,$$

which can be used to extract x via multiplication by $\mathbf{G}^{-1}(\lfloor q/2 \rfloor \, \iota_{n+1})$, where ι_{n+1} is the $n+1$-th unit vector.

Lemma 3.10. (homomorphic evaluation for vector-valued functions [HLL23]) There is an efficient algorithm

$$\mathsf{MakeVEvalCkt}(1^n, 1^m, q, C) = \mathsf{VEval}_C$$

that takes as input n, m, q and a vector-valued circuit $C : \{0,1\}^L \to \mathbb{Z}_q^{1\times m'}$ and outputs a circuit

$$\mathsf{VEval}_C(\mathbf{X}_1, ..., \mathbf{X}_L) = \mathbf{C},$$

taking L ciphertexts as input and outputting a new ciphertext $\mathbf{C}$ of different format.

- The depth of VEval_C is $d \cdot O(\log m \log\log q) + O(\log^2 \log q)$ for C of depth d.
- Suppose $\mathbf{X}_\ell = \mathbf{A}_{\mathsf{fhe}}\mathbf{R}_\ell - \mathbf{x}[\ell]\mathbf{G}$ for $\ell \in [L]$ with $\mathbf{x} \in \{0,1\}^L$, then

$$\mathbf{C} = \mathbf{A}_{\mathsf{fhe}}\mathbf{R}_C - \begin{pmatrix} \mathbf{0}_{n\times m'} \\ C(\mathbf{x}) \end{pmatrix} \in \mathbb{Z}_q^{(n+1)\times m'},$$

where $\left\|\mathbf{R}_C^\mathsf{T}\right\| \le (m+2)^d \lceil \log q \rceil \max_{\ell\in[L]} \left\|\mathbf{R}_\ell^\mathsf{T}\right\|$.
The new decryption equation is

$$\mathbf{s}^\mathsf{T}\mathbf{C} = -\mathbf{e}_{\mathsf{fhe}}^\mathsf{T}\mathbf{R}_C + C(\mathbf{x}) \in \mathbb{Z}_q^{1\times m'}.$$

3.1.3 Homomorphic Evaluation Procedures

In this section we describe the properties of the attribute encoding and its homomorphic evaluation. We adapt the syntax from [HLL23].

- For L-bit input, the public parameter is $\mathbf{A}_{\mathsf{att}} \in \mathbb{Z}_q^{(n+1)\times(L+1)m}$.
- The encoding of $\mathbf{x} \in \{0,1\}^L$ is

$$\mathbf{s}^\mathsf{T}(\mathbf{A}_{\mathsf{att}} - (1, \mathbf{x}^\mathsf{T}) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{att}}^\mathsf{T},$$

where $\mathbf{s}^\mathsf{T} = (\bar{\mathbf{s}}^\mathsf{T}, -1)$ with $\bar{\mathbf{s}} \in \mathbb{Z}^n$ and $\mathbf{e}_{\mathsf{att}}^\mathsf{T} \in \mathbb{Z}^{(L+1)m}$.

- There are efficient deterministic algorithms [BTVW17]

$$\mathsf{MEvalC}(\mathbf{A}_{\mathsf{att}}, C) = \mathbf{H}_C \quad \text{and} \quad \mathsf{MEvalCX}(\mathbf{A}_{\mathsf{att}}, C, \mathbf{x}) = \mathbf{H}_{C,x}$$

that take as input $\mathbf{A}_{\mathsf{att}}$, a matrix-valued circuit $C : \{0,1\}^L \to \mathbb{Z}_q^{(n+1)\times m'}$, and (for $\mathsf{MEvalCX}$) some $\mathbf{x} \in \{0,1\}^L$, and output some matrix in $\mathbb{Z}^{(L+1)m\times m'}$.

  - Suppose C is of depth d, then $\left\|\mathbf{H}_C^\mathsf{T}\right\|, \|\mathbf{H}^\mathsf{T}\|_{C,\mathbf{x}} \le (m+2)^d \lceil \log q \rceil$.
  - The matrix encoding homomorphism is $(\mathbf{A}_{\mathsf{att}} - (1, \mathbf{x}^\mathsf{T}) \otimes \mathbf{G})\mathbf{H}_{C,\mathbf{x}} = \mathbf{A}_{\mathsf{att}}\mathbf{H}_C - C(\mathbf{x})$.

Dual-Use Technique and Extension. In [BTVW17], the attribute encoded with secret $\mathbf{s}^\mathsf{T}$ is FHE ciphertexts under key $\mathbf{s}^\mathsf{T}$ (the same, "dual-use") and the circuit being MEvalCX'ed is some HEval_C. This leads to automatic decryption. Let C be a vector-valued circuit, with codomain $\mathbb{Z}_q^{1\times m'}$, then VEval_C is $\mathbb{Z}_q^{(n+1)\times m'}$-valued and

$$\begin{aligned}
&(\mathbf{s}^\mathsf{T}(\mathbf{A}_{\mathsf{att}} - (1, \mathsf{bits}(\mathbf{X})) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{att}}^\mathsf{T}) \cdot \mathbf{H}_{\mathsf{VEval}_C,\mathbf{X}} \\
&= \mathbf{s}^\mathsf{T}\mathbf{A}_{\mathsf{att}}\mathsf{VEval}_C - \mathbf{s}^\mathsf{T}\mathsf{VEval}_C(\mathbf{X}) + (\mathbf{e}')^\mathsf{T} \quad (\mathsf{MEvalCX}) \\
&= \mathbf{s}^\mathsf{T}\mathbf{A}_{\mathsf{att}}\mathsf{VEval}_C - C(\mathbf{x}) + (\mathbf{e}'')^\mathsf{T}. \qquad (\mathsf{VEval}\text{ decryption })
\end{aligned}$$

3.2 Basic Cryptographic Primitives

3.2.1 Puncturable Pseudorandom Functions

Syntax. A puncturable pseudorandom function $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ with key space $\mathcal{K}$, input space $\mathcal{X}$ and output space $\mathcal{Y}$ has the following syntax.

$\mathsf{Setup}(1^\lambda) \to K$. The setup algorithm takes as input the security parameter λ and outputs a key $K \in \mathcal{K}$.

$\mathsf{Puncture}(K, x) \to K\{x\}$. The puncture algorithm takes as input a PRF key $K \in \mathcal{K}$ and an input $x \in \mathcal{X}$, and outputs a punctured key key $K\{x\}$.

Definition 3.11. (Correctness) A puncturable pseudorandom function scheme is said to be correct if for any $K \in \mathcal{K}$, $x, x' \in \mathcal{X}$ such that $x \neq x'$, we have

$$\Pr\left[F(K\{x\}, x') = F(K, x') \mid K\{x\} \leftarrow \mathsf{Puncture}(K, x)\right] = 1.$$

Definition 3.12. (Security) A puncturable pseudorandom function scheme is said to be (selectively) secure if the advantage of a PPT adversary $\mathcal{A}$ in the following experiment is negligible.

1. $\mathcal{A}$ on input 1^λ outputs the challenge input $x^\star$.
2. The challenger samples a random key $K \leftarrow \mathcal{K}$ and a bit $\beta \leftarrow \{0, 1\}$. Then, it computes $y = F(K, x)$ if $\beta = 0$, else it sample $y \leftarrow \mathcal{Y}$ uniformly at random. It also computes $K\{x^\star\} \leftarrow \mathsf{Puncture}(K, x^\star)$ and sends $K\{x^\star\}, y$ to $\mathcal{A}$.
3. $\mathcal{A}$ outputs a guess bit β'.

$\mathcal{A}$ wins if $\beta = \beta'$.

A puncturable pseudorandom function with the security defined above exists from one-way functions [GGM84, BW13, BGI14, KPTZ13].

3.2.2 Symmetric Key Encryption with Pseudorandom Ciphertext

Syntax. A symmetric key encryption scheme for message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda \in [\mathbb{N}]}$, key space $\mathcal{K} = \{\mathcal{K}_\lambda\}_{\lambda \in [\mathbb{N}]}$ and ciphertext space $\mathcal{CT}_{\mathsf{SKE}} = \{\mathcal{CT}_{\mathsf{SKE},\lambda}\}_{\lambda \in [\mathbb{N}]}$ has the following syntax.

$\mathsf{Setup}(1^\lambda) \to \mathsf{sk}$. The setup algorithm takes as input the security parameter λ and outputs a secret key sk.

$\mathsf{Enc}(\mathsf{sk}, m) \to \mathsf{ct}$. The encryption algorithm takes as input the secret key sk and a message $m \in \mathcal{M}_\lambda$ and outputs a ciphertext ct.

$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to m'$. The decryption algorithm takes as input a secret key sk and a ciphertext ct and outputs a message $m' \in \mathcal{M}_\lambda$.

Definition 3.13. (Correctness) A SKE scheme is said to be correct if there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, for every message $m \in \mathcal{M}_\lambda$, we have

$$\Pr\left[\begin{array}{ll} & \mathsf{sk} \leftarrow \mathsf{Setup}(1^\lambda); \\ m' = m : & \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{sk}, m); \\ & m' = \mathsf{Dec}(\mathsf{sk}, \mathsf{ct}). \end{array}\right] \geq 1 - \mathsf{negl}(\lambda),$$

Definition 3.14. (Security) A SKE scheme is said to have pseudorandom ciphertext if there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, for every message $m \in \mathcal{M}_\lambda$, we have

$$\left|\Pr\left[\begin{array}{ll} \beta' = \beta : & \mathsf{sk} \leftarrow \mathsf{Setup}(1^\lambda); \\ & \beta' \leftarrow \mathcal{A}^{\mathsf{Enc}(\mathsf{sk},\cdot), \mathsf{Enc}^\beta(\mathsf{sk},\cdot)}. \end{array}\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda),$$

where the $\mathsf{Enc}(\mathsf{sk}, \cdot)$ oracle, on input a message m, returns $\mathsf{Enc}(\mathsf{sk}, m)$ and $\mathsf{Enc}^{\beta}(\mathsf{sk}, \cdot)$ oracle, on input a message m, returns ct_{β}, where $\mathsf{ct}_0 \leftarrow \mathsf{Enc}(\mathsf{sk}, m)$ and $\mathsf{ct}_1 \leftarrow \mathcal{CT}_{\mathsf{SKE},\lambda}$. We say that an SKE scheme has (non-uniform) subexponential security if there exists a constant $0 < \delta < 1$ such that the above advantage is at most $2^{-\lambda^{\delta}}$ for any adversary whose running time (resp., size) is $2^{\lambda^{\delta}}$ for sufficiently large λ.

3.2.3 Blind Garbled Circuit

Here we provide the definition of a garbling scheme for circuit class $\mathcal{C} = \{C : \{0,1\}^{\ell_{\mathsf{in}}} \to \{0,1\}^{\ell_{\mathsf{out}}}\}$. A garbling scheme for circuit class $\mathcal{C}$ consists of three algorithms $(\mathsf{Garble}, \mathsf{Eval}, \mathsf{Sim})$ with the following syntax.

$\mathsf{Garble}(1^{\lambda}, 1^{\ell_{\mathsf{in}}}, 1^{\ell_{\mathsf{out}}}, C) \to (\mathsf{lab}, \tilde{C})$. The garbling algorithm takes as input the security parameter λ, the input length ℓ_{in} and output length ℓ_{out} for circuit C, the description of the circuit C, and a random value $\mathsf{st} \in \{0,1\}^{\lambda}$ and outputs the labels for input wire of the garbled circuit $\mathsf{lab} = \{\mathsf{lab}_{j,b}\}_{j \in [\ell_{\mathsf{in}}], b \in \{0,1\}}$ where each $\mathsf{lab}_{j,b} \in \{0,1\}^{\lambda}$ and the garbled circuit $\tilde{C}$.

$\mathsf{Eval}(1^{\lambda}, \tilde{C}, \mathsf{lab}_{\mathbf{x}}) \to \mathbf{y}$. The evaluation algorithm takes as input the garbled circuit $\tilde{C}$ and labels corresponding to an input $\mathbf{x} \in \{0,1\}^{\ell_{\mathsf{in}}}$, $\mathsf{lab}_{\mathbf{x}} = \{\mathsf{lab}_{i,x_i}\}_{i \in [\ell_{\mathsf{in}}]}$ where x_i denotes the i-th bit of $\mathbf{x}$, and it outputs $\mathbf{y} \in \{0,1\}^{\ell_{\mathsf{out}}}$.

$\mathsf{Sim}(1^{\lambda}, 1^{|C|}, 1^{\ell_{\mathsf{in}}}, \mathbf{y})$ is a PPT algorithm that takes as input the security parameter, the description length of C, an input length ℓ_{in} and a string $\mathbf{y} \in \{0,1\}^{\ell_{\mathsf{out}}}$, and outputs a simulated garbled circuit $\bar{C}$ and labels $\bar{\mathsf{lab}}$.

A garbling scheme satisfies the following properties.

Definition 3.15 (Correctness). A garbling scheme is said to be correct if for any circuit $C \in \mathcal{C}$ and any input $x \in \{0,1\}^{\ell_{\mathsf{in}}}$, the following holds

$$\Pr\left[\mathbf{y} = C(\mathbf{x}) : (\mathsf{lab}, \tilde{C}) \leftarrow \mathsf{Garble}(1^{\lambda}, 1^{\ell_{\mathsf{in}}}, 1^{\ell_{\mathsf{out}}}, C); \mathbf{y} \leftarrow \mathsf{Eval}(\tilde{C}, \mathsf{lab}_{\mathbf{x}})\right] = 1.$$

Definition 3.16 (Simulation Security). A garbling scheme is said to satisfy simulation security if for any circuit $C \in \mathcal{C}$ and any input $\mathbf{x} \in \{0,1\}^{\ell_{\mathsf{in}}}$, the following holds

$$\{(\tilde{C}, \mathsf{lab}_{\mathbf{x}}) \mid (\mathsf{lab}, \tilde{C}) \leftarrow \mathsf{Garble}(1^{\lambda}, 1^{\ell_{\mathsf{in}}}, 1^{\ell_{\mathsf{out}}}, C)\} \approx_c \{(\bar{C}, \bar{\mathsf{lab}}) \mid (\bar{C}, \bar{\mathsf{lab}}) \leftarrow \mathsf{Sim}(1^{\lambda}, 1^{|C|}, 1^{\ell_{\mathsf{in}}}, C(\mathbf{x}))\}$$

where $\mathsf{lab} = \{\mathsf{lab}_{j,b}\}_{j \in [\ell_{\mathsf{in}}], b \in \{0,1\}}$ and $\mathsf{lab}_{\mathbf{x}} = \{\mathsf{lab}_{i,x_i}\}_{i \in [\ell_{\mathsf{in}}]}$.

Definition 3.17 (Blindness). [BLSV18] A garbling scheme $(\mathsf{Garble}, \mathsf{Eval}, \mathsf{Sim})$ is called blind if the distribution $\mathsf{Sim}(1^{\lambda}, 1^{|C|}, 1^{\ell_{\mathsf{in}}}, \mathbf{y})$ for $\mathbf{y} \leftarrow \{0,1\}^{\ell_{\mathsf{out}}}$, representing the output of the simulator on a completely uniform output, is indistinguishable from a completely uniform bit string. (Note that the distinguisher must not know the random output value that was used for the simulation.)

Definition 3.18 (Decomposability). We note that the $\mathsf{Garble}(1^{\lambda}, 1^{\ell_{\mathsf{in}}}, 1^{\ell_{\mathsf{out}}}, C)$ algorithm can be decomposed, using shared randomness st, as follows : (i) $\mathsf{Garble}_i(1^{\lambda}, C_i; \mathsf{st})$ for $i \in [|C|]$, where $\mathsf{Garble}_i(1^{\lambda}, C_i)$ outputs the garbling of i-th gate of the circuit C (denoted by C_i) and (ii) $\mathsf{Garble}_{\mathsf{inp}}(1^{\lambda}, 1^{\ell_{\mathsf{in}}}, 1^{\ell_{\mathsf{out}}}; \mathsf{st}) = \mathsf{lab}$ which outputs $2 \cdot \ell_{\mathsf{in}}$ labels.

Note that information of a single gate C_i of C can be represented by a binary string of length at most 4λ for example, since it suffices to encode its index, indices of its two incoming wires, and the truth table of the gate.

Theorem 3.19. [BLSV18] Assume that one-way function exists. Then, there exists a blind garbled circuits scheme.

3.3 Attribute Based and Predicate Encryption

3.3.1 Attribute Based Encryption

We define both ciphertext policy attribute-based encryption (cpABE) and key policy attribute-based encryption (kpABE) in a unified form below.

Let $R = \{R_{\lambda} : A_{\lambda} \times B_{\lambda} \to \{0,1\}\}_{\lambda \in \mathbb{N}}$ be a relation where A_{λ} and B_{λ} denote "ciphertext attribute" and "key attribute" spaces. An attribute-based encryption (ABE) scheme for R and a message space $\mathcal{M} = \{\mathcal{M}_{\lambda}\}_{\lambda \in \mathbb{N}}$ is defined by the following PPT algorithms:

$\mathsf{Setup}(1^\lambda) \to (\mathsf{mpk}, \mathsf{msk})$. The setup algorithm takes as input the unary representation of the security parameter λ and outputs a master public key mpk and a master secret key msk.

$\mathsf{Enc}(\mathsf{mpk}, X, \mu) \to \mathsf{ct}$. The encryption algorithm takes as input a master public key mpk, a ciphertext attribute $X \in A_\lambda$, and a message $\mu \in \mathcal{M}_\lambda$. It outputs a ciphertext ct.

$\mathsf{KeyGen}(\mathsf{msk}, Y) \to \mathsf{sk}_Y$. The key generation algorithm takes as input the master secret key msk and a key attribute $Y \in B_\lambda$. It outputs a private key sk_Y.

$\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_Y, Y, \mathsf{ct}, X) \to \mu$ or $\perp$. The decryption algorithm takes as input the master public key mpk, a private key sk_Y, private key attribute $Y \in B_\lambda$, a ciphertext ct and ciphertext attribute $X \in A_\lambda$. It outputs the message μ or $\perp$ which represents that the ciphertext is not in a valid form.

Definition 3.20 (Correctness). An ABE scheme for relation family R is correct if for all $\lambda \in \mathbb{N}$, $X \in A_\lambda, Y \in B_\lambda$ such that $R(X, Y) = 1$, and for all messages $\mu \in \mathcal{M}_\lambda$,

$$\Pr\left[\begin{array}{l} (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda), \\ \mathsf{sk}_Y \leftarrow \mathsf{KeyGen}(\mathsf{msk}, Y), \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, X, \mu) : \\ \mathsf{Dec}\Big(\mathsf{mpk}, \mathsf{sk}_Y, Y, \mathsf{ct}, X\Big) \neq \mu \end{array}\right] = \mathsf{negl}(\lambda)$$

where the probability is taken over the coins of Setup, KeyGen, and Enc.

Definition 3.21 (VerSel-IND security for ABE). For an ABE scheme $\mathsf{ABE} = \{\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec}\}$ for a relation family $R = \{R_\lambda : A_\lambda \times B_\lambda \to \{0,1\}\}_{\lambda \in [\mathbb{N}]}$ and a message space $\{\mathcal{M}_\lambda\}_{\lambda \in \mathbb{N}}$ and an adversary $\mathcal{A}$, let us define Sel-IND security game as follows.

1. $\mathcal{A}$ outputs the challenge ciphertext attribute $X^\star \in A_\lambda$ and the key queries $Y_1, \ldots, Y_Q$.
2. **Setup phase:** On input 1^λ, the challenger samples $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and gives mpk to $\mathcal{A}$.
3. **Challenge Query :** At some point, $\mathcal{A}$ submits a pair of equal length messages $(\mu_0, \mu_1) \in \mathcal{M}^2$ to the challenger. The challenger samples a random bit $b \leftarrow \{0,1\}$ and replies to $\mathcal{A}$ with $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, X^\star, \mu_b)$. We require that $R(X^\star, Y_i) = 0$ holds for all $i \in [Q]$.
4. **Output phase:** $\mathcal{A}$ outputs a guess bit b' as the output of the experiment.

We define the advantage $\mathsf{Adv}^{\mathsf{VerSel\text{-}IND}}_{\mathsf{ABE},\mathcal{A}}(1^\lambda)$ of $\mathcal{A}$ in the above game as

$$\mathsf{Adv}^{\mathsf{VerSel\text{-}IND}}_{\mathsf{ABE},\mathcal{A}}(1^\lambda) := \Big|\Pr\Big[\mathsf{Exp}_{\mathsf{ABE},\mathcal{A}}(1^\lambda) = 1 | b = 0\Big] - \Pr\Big[\mathsf{Exp}_{\mathsf{ABE},\mathcal{A}}(1^\lambda) = 1 | b = 1\Big]\Big|.$$

The ABE scheme ABE is said to satisfy VerSel-IND security (or simply *very selective security*) if for any stateful PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\mathsf{Adv}^{\mathsf{VerSel\text{-}IND}}_{\mathsf{ABE},\mathcal{A}}(1^\lambda) = \mathsf{negl}(\lambda)$.

We can consider the following stronger version of the security where we require the ciphertext to be pseudorandom.

Definition 3.22 (VerSel-INDr security for ABE). We define VerSel-INDr security game similarly to VerSel-IND security game except that the adversary $\mathcal{A}$ chooses single message μ instead of (μ_0, μ_1) at the challenge phase and the challenger returns $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, X^\star, \mu)$ if $b = 0$ and a random ciphertext $\mathsf{ct} \leftarrow \mathcal{CT}$ from a ciphertext space $\mathcal{CT}$ if $b = 1$. Here, we assume that uniform sampling from the ciphertext space $\mathcal{CT}$ is possible without any parameter other than the security parameter λ. We define the advantage $\mathcal{A}^{\mathsf{VerSel\text{-}INDr}}_{\mathsf{ABE},\mathcal{A}}(1^\lambda)$ of the adversary $\mathcal{A}$ accordingly and say that the scheme satisfies VerSel-INDr security if the quantity is negligible.

In the following, we recall definitions of various ABEs by specifying the relation.

Ciphertext-policy Attribute Based encryption (cpABE)**.** We define cpABE for circuit class $\{\mathcal{C}_{\ell(\lambda),d(\lambda)}\}_\lambda$ by specifying the relation. Here, $\mathcal{C}_{\ell(\lambda),d(\lambda)}$ is a set of circuits with binary output whose input length is $\ell(\lambda)$ and the depth is at most $d(\lambda)$. Note that we do not pose any restriction on the size of the circuits. We define $A_\lambda^{\mathsf{cpABE}} = \mathcal{C}_{\ell(\lambda),d(\lambda)}$ and $B_\lambda^{\mathsf{cpABE}} = \{0,1\}^\ell$. Furthermore, we define the relation $R_\lambda^{\mathsf{cpABE}}$ as $R_\lambda^{\mathsf{cpABE}}(C, \mathbf{x}) = C(\mathbf{x})$.

Key-policy Attribute Based encryption (kpABE)**.** To define kpABE for circuits, we simply swap key and ciphertext attributes in cpABE for circuits. More formally, to define kpABE for circuits, we define $A_\lambda^{\mathsf{kpABE}} = \{0,1\}^\ell$ and $B_\lambda^{\mathsf{kpABE}} = \mathcal{C}_{\ell(\lambda),d(\lambda)}$. We also define $R_\lambda^{\mathsf{kpABE}} : A_\lambda^{\mathsf{kpABE}} \times B_\lambda^{\mathsf{kpABE}} \to \{0,1\}$ as $R_\lambda^{\mathsf{kpABE}}(\mathbf{x}, C) = C(\mathbf{x})$.

The above relations also holds for circuit class $\{\mathcal{C}_{\ell(\lambda)}\}_\lambda$ which is the set of circuits with binary output whose input length is $\ell(\lambda)$ and the depth is unbounded.

3.3.2 Predicate Encryption

In this section we define predicate encryption (PE) scheme. The syntax and correctness is same as that of the ABE scheme in Section 3.3 except that we do not input the ciphertext attribute into the decryption algorithm, i.e $\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_Y, Y, \mathsf{ct}) \to \mu$ or $\perp$. We set $A_\lambda = \{0,1\}^\ell$ and $B_\lambda = \mathcal{C}_{\ell(\lambda),d(\lambda)}$. We also define $R_\lambda : A_\lambda^{\mathsf{PE}} \times B_\lambda^{\mathsf{PE}} \to \{0,1\}$ as $R_\lambda(\mathbf{x}, C) = C(\mathbf{x})$.

Here we define selective INDr security for a PE scheme.

Definition 3.23 (Selective INDr Security). A PE scheme is said to satisfy selective INDr security if there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, we have

$$\left| \Pr\left[\beta' = \beta : \begin{array}{l} \mathbf{x} \leftarrow \mathcal{A}(1^\lambda); \\ (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda); \\ (\mu, \mathsf{st}) \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{mpk}); \\ \mathsf{ct}_0 \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathbf{x}, \mu), \mathsf{ct}_1 \leftarrow \mathcal{CT}; \\ \beta \leftarrow \{0,1\}, \beta' \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{st}, \mathsf{ct}_\beta) \end{array} \right] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda),$$

where $\mathcal{CT}$ is the ciphertext space of the scheme and the adversary $\mathcal{A}$ is admissible in the sense that for all key query f made by $\mathcal{A}$ to the $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$ oracle, it holds that $f(\mathbf{x}) = 0$.

Theorem 3.24 ([GVW15a, GKW17, WZ17]). Assuming LWE, there exists predicate encryption schemes for (bounded depth) circuits satisfying selective INDr security.

3.3.3 Multi-Input Predicate Encryption

In this section we define multi-input Predicate Encryption (mi-PE), adapting the syntax from [AYY22]. Consider a function family $\{\mathcal{F}_{\mathsf{prm}} = \{f : (\mathcal{X}_{\mathsf{prm}})^n \to \{0,1\}\}\}_{\mathsf{prm}}$, for a parameter $\mathsf{prm} = \mathsf{prm}(\lambda)$, where each $\mathcal{F}_{\mathsf{prm}}$ is a finite collection of n-ary functions. Each function $f \in \mathcal{F}_{\mathsf{prm}}$ takes as input strings $x_1, \ldots, x_n$, where each $x_i \in \mathcal{X}_{\mathsf{prm}}$ and outputs $f(x_1, \ldots, x_n) \in \{0,1\}$.

Syntax. A mi-PE scheme miPE_n for n-ary function family $\mathcal{F}_{\mathsf{prm}}$ consists of polynomial time algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}_1, \ldots, \mathsf{Enc}_n, \mathsf{Dec})$ defined as follows.

$\mathsf{Setup}(1^\lambda, 1^n, \mathsf{prm}) \to (\mathsf{mpk}, \mathsf{msk})$. The setup algorithm takes as input the security parameter λ, the function arity n and a parameter prm and outputs a master public key mpk and master secret key msk.

$\mathsf{KeyGen}(\mathsf{msk}, f) \to \mathsf{sk}_f$.The key generation algorithm takes as input the master secret key msk and a function $f \in \mathcal{F}_{\mathsf{prm}}$ and it outputs a functional secret key sk_f.

$\mathsf{Enc}_i(\mathsf{msk}, x_i, \mu_i) \to \mathsf{ct}_i$ for $i \in [n]$. The encryption algorithm for the i^{th} slot takes as input a master secret key msk, an attribute $x_i \in \mathcal{X}_{\mathsf{prm}}$, and message $\mu_i \in \{0,1\}$, and outputs a ciphertext ct_i.

$\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_f, f, \mathsf{ct}_1, \mathsf{ct}_2, \ldots, \mathsf{ct}_n) \to \{0,1\}^n \cup \perp$. The decryption algorithm takes as input the master public key mpk, secret key sk_f, function f and n ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_n$ and outputs a string $\mu' \in \{0,1\}^n \cup \perp$.

Next, we define correctness and security.

Correctness: For every $\lambda \in \mathbb{N}, \mu_1, \ldots, \mu_n \in \{0,1\}, x_1, \ldots, x_n \in \mathcal{X}_{\mathsf{prm}}, f \in \mathcal{F}_{\mathsf{prm}}$, it holds that if $f(x_1, \ldots, x_n) = 1$, then

$$\Pr\left[\mathsf{Dec}\left(\begin{matrix}\mathsf{mpk}, \mathsf{KeyGen}(\mathsf{msk}, f), f, \\ \mathsf{Enc}_1(\mathsf{msk}, x_1, \mu_1), \ldots, \mathsf{Enc}_n(\mathsf{msk}, x_n, \mu_n)\end{matrix}\right) = (\mu_1, \ldots, \mu_n)\right] = 1 - \mathsf{negl}(\lambda)$$

where the probability is over the choice of $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, \mathsf{prm})$ and over the internal randomness of KeyGen and $\mathsf{Enc}_1, \ldots, \mathsf{Enc}_n$.

Definition 3.25 ($\mathsf{VerSel\text{-}IND}$**-Security.).** For a miPE scheme for function family $\{\mathcal{F}_{\mathsf{prm}} = \{f : (\mathcal{X}_{\mathsf{prm}})^n \to \{0,1\}\}\}_{\mathsf{prm}}$, parameter $\mathsf{prm} = \mathsf{prm}(\lambda)$, a stateful adversary $\mathcal{A}$, we define the $\mathsf{VerSel\text{-}IND}$-security game, $\mathsf{Exp}_{\mathsf{miPE},\mathcal{A}}$, as follows.

1. **Query phase:** On input $1^\lambda, 1^n, \mathsf{prm}$, $\mathcal{A}$ outputs the following in an arbitrary order.
   (a) **Key Queries:** $\mathcal{A}$ issues polynomial number of key queries, say $q_0 = q_0(\lambda)$. For each key query $k \in [q_0]$, $\mathcal{A}$ chooses a function $f_k \in \mathcal{F}_{\mathsf{prm}}$.
   (b) **Ciphertext Queries:** $\mathcal{A}$ issues polynomial number of ciphertext queries for each slot, say $q_i = q_i(\lambda)$ for the i^{th} slot. We use $(x_{i,0}^{j_i}, \mu_{i,0}^{j_i}), (x_{i,1}^{j_i}, \mu_{i,1}^{j_i})$ to denote the j_i-th ciphertext query corresponding to the i-th slot, where $j_i \in [q_i]$ and $i \in [n]$.
2. **Setup phase:** On input $1^\lambda, 1^n, \mathsf{prm}, \{f_k\}_{k \in [q_0]}$, the challenger samples $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, \mathsf{prm})$, a bit $\beta \leftarrow \{0,1\}$ and does the following.
   (a) It computes $\mathsf{sk}_{f_k} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f_k)$.
   (b) It computes $\mathsf{ct}_{i,\beta}^{j_i} \leftarrow \mathsf{Enc}_i(\mathsf{msk}, x_{i,\beta}^{j_i}, \mu_{i,\beta}^{j_i})$ for $i \in [n], j_i \in [q_i]$.

   It returns $(\mathsf{mpk}, \{\mathsf{sk}_{f_k}\}_{k \in [q_0]}, \{\mathsf{ct}_{i,\beta}^{j_i}\}_{i \in [n], j_i \in [q_i]})$ to $\mathcal{A}$.
3. **Output phase:** $\mathcal{A}$ outputs a guess bit β' as the output of the experiment.

For the adversary to be *admissible*, we require that it holds that $f_k(x_{1,\beta}^{j_1}, \ldots, x_{n,\beta}^{j_n}) = 0$ for every $i \in [n], j_i \in [q_i], \beta \in \{0,1\}$, and $k \in [q_0]$. We define the advantage $\mathsf{Adv}_{\mathsf{miPE},\mathcal{A}}^{\mathsf{Sel\text{-}IND}}$ of $\mathcal{A}$ in the security game as

$$\mathsf{Adv}_{\mathsf{miPE},\mathcal{A}}^{\mathsf{Sel\text{-}IND}}(1^\lambda) := \left|\Pr\left[\mathsf{Exp}_{\mathsf{miPE},\mathcal{A}}(1^\lambda) = 1|\beta = 0\right] - \Pr\left[\mathsf{Exp}_{\mathsf{miPE},\mathcal{A}}(1^\lambda) = 1|\beta = 1\right]\right|.$$

The miPE scheme is said to satisfy $\mathsf{Sel\text{-}IND}$-security if for any stateful PPT adversary $\mathcal{A}$, $\mathsf{Adv}_{\mathsf{miPE},\mathcal{A}}^{\mathsf{Sel\text{-}IND}}(1^\lambda) = \mathsf{negl}(\lambda)$.

Due to space constraints, we proceed directly to the definition and construction of functional encryption for pseudorandom functionalities. The remaining technical details and the full expansion of our constructions, as outlined in the overview, are deferred to the full version of our work.

4 Functional Encryption for Pseudorandom Functionalities

4.1 Definition

In this section we give the definitions for functional encryption for pseudorandom functionalities. Consider a function family $\{\mathcal{F}_{\mathsf{prm}} = \{f : \mathcal{X}_{\mathsf{prm}} \to \mathcal{Y}_{\mathsf{prm}}\}\}_{\mathsf{prm}}$ for a parameter $\mathsf{prm} = \mathsf{prm}(\lambda)$. Each function $f \in \mathcal{F}_{\mathsf{prm}}$ takes as input a string $x \in \mathcal{X}_{\mathsf{prm}}$ and outputs $f(x) \in \mathcal{Y}_{\mathsf{prm}}$.

Syntax. A functional encryption scheme prFE for pseudorandom functionalities $\mathcal{F}_{\mathsf{prm}}$ consists of four polynomial time algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ defined as follows.

$\mathsf{Setup}(1^\lambda, \mathsf{prm}) \to (\mathsf{mpk}, \mathsf{msk})$. The setup algorithm takes as input the security parameter λ and a parameter prm and outputs a master public key mpk and a master secret key msk[5].

$\mathsf{KeyGen}(\mathsf{msk}, f) \to \mathsf{sk}_f$. The key generation algorithm takes as input the master secret key msk and a function $f \in \mathcal{F}_{\mathsf{prm}}$ and it outputs a functional secret key sk_f.

$\mathsf{Enc}(\mathsf{mpk}, x) \to \mathsf{ct}$. The encryption algorithm takes as input the master public key mpk and an input $x \in \mathcal{X}_{\mathsf{prm}}$ and outputs a ciphertext $\mathsf{ct} \in \mathcal{CT}$, where $\mathcal{CT}$ is the ciphertext space.

$\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_f, f, \mathsf{ct}) \to y$. The decryption algorithm takes as input the master public key mpk, secret key sk_f, function f and a ciphertext ct and outputs $y \in \mathcal{Y}_{\mathsf{prm}}$.

Definition 4.1 (Correctness). A prFE scheme is said to satisfy *perfect* correctness if for all prm, any input $x \in \mathcal{X}_{\mathsf{prm}}$ and function $f \in \mathcal{F}_{\mathsf{prm}}$, we have

$$\Pr\left[\begin{array}{c}(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{prm})\ ,\ \mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f),\\ \mathsf{Dec}\big(\mathsf{mpk}, \mathsf{sk}_f, f, \mathsf{Enc}(\mathsf{mpk}, x)\big) = f(x)\end{array}\right] = 1.$$

We define our security notions next. At a high level, our first notion says that so long as the output of the functionality is pseudorandom, the ciphertext is pseudorandom. For notational brevity, we denote this by prCT security.

Definition 4.2 (prCT **Security).** For a prFE scheme for function family $\{\mathcal{F}_{\mathsf{prm}} = \{f : \mathcal{X}_{\mathsf{prm}} \to \mathcal{Y}_{\mathsf{prm}}\}\}_{\mathsf{prm}}$, parameter $\mathsf{prm} = \mathsf{prm}(\lambda)$, let Samp be a PPT algorithm that on input 1^λ, outputs

$$(f_1, \dots, f_{Q_{\mathsf{key}}}, x_1, \dots, x_{Q_{\mathsf{msg}}}, \mathsf{aux} \in \{0,1\}^*)$$

where Q_{key} is the number of key queries, Q_{msg} is the number of message queries, and $f_i \in \mathcal{F}_{\mathsf{prm}}$, $x_j \in \mathcal{X}_{\mathsf{prm}}$ for all $i \in [Q_{\mathsf{key}}], j \in [Q_{\mathsf{msg}}]$.
We define the following advantage functions:

$$\begin{aligned}\mathsf{Adv}^{\mathrm{PRE}}_{\mathcal{A}_0}(\lambda) \stackrel{\mathrm{def}}{=} &\Pr\Big[\mathcal{A}_0\Big(\ \mathsf{aux},\ \{f_i,\ f_i(x_j)\}_{i\in[Q_{\mathsf{key}}], j\in[Q_{\mathsf{msg}}]}\Big) = 1\Big]\\ &- \Pr\Big[\mathcal{A}_0(\ \mathsf{aux},\ \{f_i,\ \Delta_{i,j} \leftarrow \mathcal{Y}_{\mathsf{prm}}\}_{i\in[Q_{\mathsf{key}}], j\in[Q_{\mathsf{msg}}]}) = 1\Big]\end{aligned}$$

$$\begin{aligned}\mathsf{Adv}^{\mathrm{POST}}_{\mathcal{A}_1}(\lambda) \stackrel{\mathrm{def}}{=} &\Pr\Big[\mathcal{A}_1(\mathsf{mpk},\ \mathsf{aux},\ \{f_i,\ \mathsf{Enc}(\mathsf{mpk}, x_j),\ \mathsf{sk}_{f_i}\}_{i\in[Q_{\mathsf{key}}], j\in[Q_{\mathsf{msg}}]}) = 1\Big]\\ &- \Pr\Big[\mathcal{A}_1(\mathsf{mpk},\ \mathsf{aux},\ \{f_i,\ \delta_j \leftarrow \mathcal{CT},\ \mathsf{sk}_{f_i}\}_{i\in[Q_{\mathsf{key}}], j\in[Q_{\mathsf{msg}}]}) = 1\Big]\end{aligned}$$

where $(f_1, \dots, f_{Q_{\mathsf{key}}}, x_1, \dots, x_{Q_{\mathsf{msg}}}, \mathsf{aux} \in \{0,1\}^*) \leftarrow \mathsf{Samp}(1^\lambda), (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{prm})$ and $\mathcal{CT}$ is the ciphertext space. We say that a prFE scheme for function family $\mathcal{F}_{\mathsf{prm}}$ satisfies prCT security with respect to the sampler class $\mathcal{SC}$ if for every PPT sampler Samp $\in \mathcal{SC}$ there exists a polynomial $Q(\cdot)$ such that for every PPT adversary $\mathcal{A}_1$, there exists another PPT $\mathcal{A}_0$ such that

$$\mathcal{A}^{\mathrm{PRE}}_{\mathcal{A}_0}(\lambda) \geq \mathcal{A}^{\mathrm{POST}}_{\mathcal{A}_1}(\lambda)/Q(\lambda) - \mathsf{negl}(\lambda) \tag{11}$$

and $\mathsf{Time}(\mathcal{A}_0) \leq \mathsf{Time}(\mathcal{A}_1) \cdot Q(\lambda)$.

[5]We assume w.l.o.g that msk includes mpk.

We note that the above security definition is in the multi-challenge flavor. Unlike the standard security notions for public key primitives (e.g., indistinguishability security for functional encryption [GGH+16]), the single-challenge does not imply the multi-challenge security, since the standard hybrid argument fails. It is shown in [AMYY25] that there is no prFE that satisfies prCT security in the multi-challenge setting for all general samplers. This impossibility crucially relies on Q_{msg} being sufficiently large and does not apply to the single-challenge setting. We circumvent this impossibility in two ways.

Remark 4.3 (Single-Challenge Definition). We can consider single-challenge version of the definition where we restrict ourselves to the setting of $Q_{\mathsf{msg}} = 1$. We argue that there is a huge gap between the multi-challenge security and single-challenge security for PRFE, similarly to the case of FE with simulation-based security, where the multi-challenge security is impossible[BSW11], while the single-challenge security is not ruled out in the selective setting, to the best of our knowledge. We conjecture that PRFE for general samplers may plausibly exist for the single-challenge case.

We also introduce an indistinguishability-based security definition. This definition is strictly weaker than standard indistinguishability definition for FE [GGH+16], since we require the security to hold only for the case where all the decryption results are pseudorandom. Since no impossibility result is known to apply to this definition, even for the multi-challenge setting, it is also weaker than Definition 4.2.

Definition 4.4 (IND Security). For a prFE scheme for function family $\{\mathcal{F}_{\mathsf{prm}} = \{f : \mathcal{X}_{\mathsf{prm}} \to \mathcal{Y}_{\mathsf{prm}}\}\}_{\mathsf{prm}}$, parameter $\mathsf{prm} = \mathsf{prm}(\lambda)$, let Samp be a PPT algorithm that on input 1^λ, outputs

$$(f_1, \ldots, f_{Q_{\mathsf{key}}}, x_1^0, \ldots, x_{Q_{\mathsf{msg}}}^0, x_1^1, \ldots, x_{Q_{\mathsf{msg}}}^1, \mathsf{aux} \in \{0,1\}^*)$$

where Q_{key} is the number of key queries, Q_{msg} is the number of message queries, and $f_i \in \mathcal{F}_{\mathsf{prm}}$, $x_j^b \in \mathcal{X}_{\mathsf{prm}}$ for all $i \in [Q_{\mathsf{key}}], j \in [Q_{\mathsf{msg}}], b \in \{0,1\}$. We say that a prFE scheme for function family $\mathcal{F}_{\mathsf{prm}}$ satisfies IND-pr-Security if for every PPT sampler Samp, the following holds: If

$$\Big(\mathsf{aux},\ \{f_i,\ f_i(x_j^0)\}_{i \in [Q_{\mathsf{key}}], j \in [Q_{\mathsf{msg}}]} \Big) \approx_c \big(\mathsf{aux},\ \{f_i,\ \Delta_{i,j} \leftarrow \mathcal{Y}_{\mathsf{prm}}\}_{i \in [Q_{\mathsf{key}}], j \in [Q_{\mathsf{msg}}]} \big)$$

and

$$f_i(x_j^0) = f_i(x_j^1) \qquad \forall i \in [Q_{\mathsf{key}}], \forall j \in [Q_{\mathsf{msg}}],$$

then we have

$$(\mathsf{mpk},\ \mathsf{aux},\ \{f_i,\ \mathsf{Enc}(\mathsf{mpk}, x_j^0),\ \mathsf{sk}_{f_i}\}_{i,j}) \approx_c (\mathsf{mpk},\ \mathsf{aux},\ \{f_i,\ \mathsf{Enc}(\mathsf{mpk}, x_j^1),\ \mathsf{sk}_{f_i}\}_{i,j})$$

where $i \in [Q_{\mathsf{key}}], j \in [Q_{\mathsf{msg}}], (f_1, \ldots, f_{Q_{\mathsf{key}}}, x_1^0, \ldots, x_{Q_{\mathsf{msg}}}^0, x_1^1, \ldots, x_{Q_{\mathsf{msg}}}^1, \mathsf{aux}) \leftarrow \mathsf{Samp}(1^\lambda), (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{prm})$.

Definition 4.5 (Compactness). A prFE scheme is said to be compact if for any input message $x \in \mathcal{X}$, the running time of the encryption algorithm is polynomial in the security parameter and the size of x. In particular, it does not depend on the circuit description size or the output length of any function f supported by the scheme.

4.2 Construction

In this section, we provide our construction of a functional encryption scheme for pseudorandom functionalities for function family $\mathcal{F}_{\mathrm{L}(\lambda), \ell(\lambda), \mathsf{dep}(\lambda)} = \{f : \{0,1\}^{\mathrm{L}} \to \{0,1\}^{\ell}\}$, where the depth of a function $f \in \mathcal{F}$ is at most $\mathsf{dep}(\lambda) = \mathrm{poly}(\lambda)$. We denote the information of the parameters representing the supported class of the circuits by $\mathsf{prm} = (1^{\mathrm{L}(\lambda)}, 1^{\ell(\lambda)}, 1^{\mathsf{dep}(\lambda)})$.

Ingredients. Our construction needs a pseudorandom function $\mathsf{PRF} : \{0,1\}^\lambda \times \{0,1\}^\lambda \to [-q/4 + B, q/4 - B]^{1\times\ell}$ that can be evaluated by a circuit of depth at most $\mathsf{dep}(\lambda) = \mathrm{poly}(\lambda)$. Here B is chosen to be exponentially smaller than $q/4$. We note that for our choice of B the statistical distance between the uniform distribution over $[-q/4, q/4]$ and $[-q/4 + B, q/4 - B]$ is negligible.

$\mathsf{Setup}(1^\lambda, \mathsf{prm}) \to (\mathsf{mpk}, \mathsf{msk})$. The setup algorithm parses $\mathsf{prm} = (1^{\mathrm{L}(\lambda)}, 1^{\ell(\lambda)}, 1^{\mathsf{dep}(\lambda)})$ and does the following.

– Sample appropriate parameters $q, n, m, \tau, \sigma, \sigma_{\mathbf{B}}, B$ and M such that M divides q, as in Equation (12)[6].

– Samples appropriate parameters as in Equation (12).

– Set $\mathrm{L}_{\mathrm{X}} = m(\lambda + \mathrm{L})(n+1)\lceil \log q \rceil$, sample $\mathbf{A}_{\mathsf{att}} \leftarrow \mathbb{Z}_q^{(n+1)\times(\mathrm{L}_{\mathrm{X}}+1)m}$ and $(\mathbf{B}, \mathbf{B}_\tau^{-1}) \leftarrow \mathsf{TrapGen}(1^{n+1}, 1^{mw}, q)$, where $w \in O(\log q)$.

– Output $\mathsf{mpk} := (\mathbf{A}_{\mathsf{att}}, \mathbf{B}, M)$ and $\mathsf{msk} := \mathbf{B}_\tau^{-1}$.

$\mathsf{KeyGen}(\mathsf{msk}, f) \to \mathsf{sk}_f$. The key generation algorithm parses $\mathsf{msk} = \mathbf{B}_\tau^{-1}$ and does the following.

– Sample $\mathbf{r} \leftarrow \{0,1\}^\lambda$ and define function $\mathrm{F} = \mathrm{F}[f, \mathbf{r}]$ with $f, \mathbf{r}$ hardwired as follows[7]:
On input $(\mathbf{x}, \mathsf{sd})$, compute and output $f(\mathbf{x})\lfloor q/2 \rceil + \mathsf{PRF}(\mathsf{sd}, \mathbf{r}) \in \mathbb{Z}_q^{1\times\ell}$.

– Parse $\mathrm{F}[f, \mathbf{r}](\mathbf{x}, \mathsf{sd}) = M \cdot f_{\mathsf{high}}(\mathbf{x}, \mathsf{sd}) + f_{\mathsf{low}}(\mathbf{x}, \mathsf{sd})$, where $f_{\mathsf{high}}(\mathbf{x}, \mathsf{sd}) \in [0, q/M]^\ell$ and $f_{\mathsf{low}}(\mathbf{x}, \mathsf{sd}) \in [0, M-1]^\ell$.Using the fact that the PRF and $f(\mathbf{x})$ can be computed by a circuit of depth at most $\mathsf{dep}(\lambda) = \mathrm{poly}(\lambda)$, the function $\mathrm{F}[f, \mathbf{r}]$ can be computed by a circuit of depth at most $d = \mathrm{poly}(\mathsf{dep})$.

– Define functions $\mathrm{F}_{\mathsf{high}} := M \cdot f_{\mathsf{high}}$ and $\mathrm{F}_{\mathsf{low}} := M \cdot f_{\mathsf{low}}$, which on input $(\mathbf{x}, \mathsf{sd})$ outputs $M \cdot f_{\mathsf{high}}(\mathbf{x}, \mathsf{sd})$ and $M \cdot f_{\mathsf{low}}(\mathbf{x}, \mathsf{sd})$, respectively. We note that these functions can be computed by a circuit of depth at most $d = \mathrm{poly}(\mathsf{dep})$.

– Define $\mathsf{VEval}_{\mathsf{high}} = \mathrm{MakeVEvalCkt}(n, m, q, \mathrm{F}_{\mathsf{high}})$ and $\mathsf{VEval}_{\mathsf{low}} = \mathrm{MakeVEvalCkt}(n, m, q, \mathrm{F}_{\mathsf{low}})$. From Lemma 3.10, the depth of $\mathsf{VEval}_{\mathsf{high}}$ and $\mathsf{VEval}_{\mathsf{low}}$ is bounded by $(dO(\log m \log\log q) + O(\log^2 \log q))$.

– Compute $\mathbf{H}_{\mathbf{A}_{\mathsf{att}}}^{\mathrm{F}_{\mathsf{high}}} = \mathsf{MEvalC}(\mathbf{A}_{\mathsf{att}}, \mathsf{VEval}_{\mathsf{high}})$, $\mathbf{H}_{\mathbf{A}_{\mathsf{att}}}^{\mathrm{F}_{\mathsf{low}}} = \mathsf{MEvalC}(\mathbf{A}_{\mathsf{att}}, \mathsf{VEval}_{\mathsf{low}}) \in \mathbb{Z}_q^{(\mathrm{L}_{\mathrm{X}}+1)m\times\ell}$.

– Compute $\mathbf{A}_{\mathsf{high}} = \mathbf{A}_{\mathsf{att}} \cdot \mathbf{H}_{\mathbf{A}_{\mathsf{att}}}^{\mathrm{F}_{\mathsf{high}}}$ and $\mathbf{A}_{\mathsf{low}} = \mathbf{A}_{\mathsf{att}} \cdot \mathbf{H}_{\mathbf{A}_{\mathsf{att}}}^{\mathrm{F}_{\mathsf{low}}}$.

– Compute
$$\mathbf{A}_{\mathrm{F}} = M \cdot \left\lfloor \frac{\mathbf{A}_{\mathsf{high}}}{M} \right\rfloor + \left\lfloor \frac{\mathbf{A}_{\mathsf{low}}}{M} \right\rfloor$$
and sample $\mathbf{K} \leftarrow \mathbf{B}_\tau^{-1}(\mathbf{A}_{\mathrm{F}})$.

– Output $\mathsf{sk}_f = (\mathbf{K}, \mathbf{r})$.

$\mathsf{Enc}(\mathsf{mpk}, \mathbf{x}) \to \mathsf{ct}$. The encryption parse $\mathsf{mpk} = (\mathbf{A}_{\mathsf{att}}, \mathbf{B}, M)$ algorithm does the following.

– Sample $\bar{\mathbf{s}} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma_{\mathbf{s}}}^n$ and set $\mathbf{s} = (\bar{\mathbf{s}}^\intercal, -1)^\intercal$.

– Sample $\mathbf{e}_{\mathbf{B}} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma_{\mathbf{B}}}^{mw}$ and compute $\mathbf{c}_{\mathbf{B}}^\intercal := \mathbf{s}^\intercal \mathbf{B} + \mathbf{e}_{\mathbf{B}}^\intercal$.

– Sample $\mathsf{sd} \leftarrow \{0,1\}^\lambda$, $\bar{\mathbf{A}}_{\mathsf{fhe}} \leftarrow \mathbb{Z}_q^{n\times m}$, $\mathbf{e}_{\mathsf{fhe}} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}^m$, $\mathbf{R} \leftarrow \{0,1\}^{m\times m(\lambda+\mathrm{L})}$ and compute a GSW encryption as follows.
$$\mathbf{A}_{\mathsf{fhe}} := \begin{pmatrix} \bar{\mathbf{A}}_{\mathsf{fhe}} \\ \bar{\mathbf{s}}^\intercal \bar{\mathbf{A}}_{\mathsf{fhe}} + \mathbf{e}_{\mathsf{fhe}}^\intercal \end{pmatrix}, \quad \mathbf{X} = \mathbf{A}_{\mathsf{fhe}}\mathbf{R} - (\mathbf{x}, \mathsf{sd}) \otimes \mathbf{G} \in \mathbb{Z}_q^{(n+1)\times m(\lambda+\mathrm{L})}.$$
Let $\mathrm{L}_{\mathrm{X}} = m(\lambda + \mathrm{L})(n+1)\lceil \log q \rceil$ be the bit length of $\mathbf{X}$.

[6]We assume these parameters to be part of the mpk.
[7]The circuit representation of the function F is the universal circuit with F hardwired.

– Compute a BGG^+ encoding as follows.

$$\mathbf{e}_{\mathsf{att}} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}^{(\mathrm{L_X}+1)m}, \qquad \mathbf{c}_{\mathsf{att}}^{\mathsf{T}} := \mathbf{s}^{\mathsf{T}}(\mathbf{A}_{\mathsf{att}} - \mathsf{bits}(1,\mathbf{X}) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{att}}^{\mathsf{T}}.$$

– Output $\mathsf{ct} = (\mathbf{c}_{\mathbf{B}}, \mathbf{c}_{\mathsf{att}}, \mathbf{X})$.

$\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_f, f, \mathsf{ct}) \to \mathbf{y}$. The decryption algorithm does the following.

– Parse $\mathsf{mpk} = (\mathbf{A}_{\mathsf{att}}, \mathbf{B}, M)$, $\mathsf{sk}_f = (\mathbf{K}, \mathbf{r})$ and $\mathsf{ct} = (\mathbf{c}_{\mathbf{B}}, \mathbf{c}_{\mathsf{att}}, \mathbf{X})$.

– Compute $\mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}} = \mathsf{MEvalCX}(\mathbf{A}_{\mathsf{att}}, \mathsf{VEval}_{\mathsf{high}}, \mathbf{X})$ and $\mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{low}}} = \mathsf{MEvalCX}(\mathbf{A}_{\mathsf{att}}, \mathsf{VEval}_{\mathsf{low}}, \mathbf{X})$ for circuits $\mathsf{VEval}_{\mathsf{high}}$ and $\mathsf{VEval}_{\mathsf{low}}$ as defined in KeyGen algorithm.

– Compute

$$\mathbf{z} := \mathbf{c}_{\mathbf{B}}^{\mathsf{T}} \cdot \mathbf{K} - \left(M \cdot \left\lfloor \frac{\mathbf{c}_{\mathsf{att}}^{\mathsf{T}} \cdot \mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}}}{M} \right\rfloor + \left\lfloor \frac{\mathbf{c}_{\mathsf{att}}^{\mathsf{T}} \cdot \mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{low}}}}{M} \right\rfloor \right).$$

– For $i \in [\ell]$, set $y_i = 0$ if $z_i \in [-q/4, q/4)$ and $y_i = 1$ otherwise, where z_i is the i-th coordinate of $\mathbf{z}$.

– Output $\mathbf{y} = (y_1, \ldots, y_\ell)$.

Parameters. We set our parameters as follows.

$$\begin{aligned}&\beta = 2^{O(\mathsf{dep}\cdot\log\lambda)},\ q = 2^{12\lambda}\beta,\ M = 2^{4\lambda}\beta,\ n = \mathrm{poly}(\lambda, \mathsf{dep}),\ m = O(n\log q),\ B = 2^{10\lambda}\beta,\\ &\tau = O\left(\sqrt{(n+1)\log q}\right)\ \ \sigma_{\mathbf{s}} = \sigma = 2^{2\lambda},\ \sigma_{\mathbf{B}} = 2^{9\lambda}\beta,\ \sigma_1 = 2^{8\lambda+O(1)}\beta/\mathrm{poly}(\lambda).\end{aligned} \tag{12}$$

Efficiency. Using the above set parameters, we have

$$|\mathsf{mpk}| = \mathrm{L}\cdot\mathrm{poly}(\mathsf{dep},\lambda),\ \ |\mathsf{sk}_f| = \ell\cdot\mathrm{poly}(\mathsf{dep},\lambda),\ \ |\mathsf{ct}| = \mathrm{L}\cdot\mathrm{poly}(\mathsf{dep},\lambda).$$

Correctness We analyze the correctness of our scheme below.

– First, we note that

$$\begin{aligned}
\mathbf{c}_{\mathsf{att}}^{\mathsf{T}} \cdot \mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}} &= (\mathbf{s}^{\mathsf{T}}(\mathbf{A}_{\mathsf{att}} - \mathsf{bits}(1,\mathbf{X}) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{att}}^{\mathsf{T}})\mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}}\\
&= \mathbf{s}^{\mathsf{T}}\mathbf{A}_{\mathsf{att}}\mathbf{H}_{\mathbf{A}_{\mathsf{att}}}^{\mathrm{F}_{\mathsf{high}}} - \mathbf{s}^{\mathsf{T}}\mathsf{VEval}_{\mathsf{high}}(\mathsf{bits}(\mathbf{X})) + \mathbf{e}_{\mathsf{att}}^{\mathsf{T}}\mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}}\\
&= \mathbf{s}^{\mathsf{T}}\mathbf{A}_{\mathsf{high}} - \mathrm{F}_{\mathsf{high}}(\mathbf{x},\mathsf{sd}) + \mathbf{e}_{\mathsf{fhe}}^{\mathsf{T}}\mathbf{R}_{\mathsf{high}} + \mathbf{e}_{\mathsf{att}}^{\mathsf{T}}\mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}}\\
\implies \left\lfloor \frac{\mathbf{c}_{\mathsf{att}}^{\mathsf{T}} \cdot \mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}}}{M} \right\rfloor &= \left\lfloor \frac{\mathbf{s}^{\mathsf{T}}\mathbf{A}_{\mathsf{high}} - M\cdot f_{\mathsf{high}}(\mathbf{x},\mathsf{sd}) + \mathbf{e}_{\mathsf{fhe}}^{\mathsf{T}}\mathbf{R}_{\mathsf{high}} + \mathbf{e}_{\mathsf{att}}^{\mathsf{T}}\mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}}}{M} \right\rfloor\\
&= \left\lfloor \frac{\mathbf{s}^{\mathsf{T}}\mathbf{A}_{\mathsf{high}} + \mathbf{e}_{\mathsf{fhe}}^{\mathsf{T}}\mathbf{R}_{\mathsf{high}} + \mathbf{e}_{\mathsf{att}}^{\mathsf{T}}\mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}}}{M} \right\rfloor - f_{\mathsf{high}}(\mathbf{x},\mathsf{sd})
\end{aligned} \tag{13}$$

where $\mathsf{VEval}_{\mathsf{high}}(\mathsf{bits}(\mathbf{X})) = \mathbf{A}_{\mathsf{fhe}}\mathbf{R}_{\mathsf{high}} - \begin{pmatrix} \mathbf{0}_{n\times\ell} \\ \mathrm{F}_{\mathsf{high}}(\mathbf{x},\mathsf{sd}) \end{pmatrix}$. Using Lemma 3.10, we have

$$\begin{aligned}\left\|\mathbf{R}_{\mathsf{high}}^{\mathsf{T}}\right\| &\leq (m+2)^d \lceil\log q\rceil \cdot m = (m+2)^d \lceil\log q\rceil \cdot 3(n+1)\lceil\log q\rceil\\ &\leq (m+2)^d O(\log q) \leq \beta.\end{aligned}$$

and using the depth bound from Section 3.1.3,

$$\left\|\left(\mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}}\right)^{\mathsf{T}}\right\| \le (m+2)^{d_{\mathsf{VEval}_{\mathsf{high}}}}\lceil\log q\rceil \le 2^{d\cdot O(\log\lambda)} \le \beta$$

where $d_{\mathsf{VEval}_{\mathsf{high}}}$ denotes the depth of the circuit $\mathsf{VEval}_{\mathsf{high}}$. So we have $\left\|\mathbf{e}_{\mathsf{fhe}}^{\mathsf{T}}\mathbf{R}_{\mathsf{high}} + \mathbf{e}_{\mathsf{att}}^{\mathsf{T}}\mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}}\right\| \le 2^{2\lambda+1}\sqrt{\lambda}\beta \le 2^{3\lambda}\beta < M$. Using this in Equation (13),

$$\begin{aligned}
\left\lfloor\frac{\mathbf{c}_{\mathsf{att}}^{\mathsf{T}}\cdot\mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}}}{M}\right\rfloor &= \left\lfloor\frac{\mathbf{s}^{\mathsf{T}}\mathbf{A}_{\mathsf{high}}}{M}\right\rfloor - f_{\mathsf{high}}(\mathbf{x},\mathsf{sd}) + \mathsf{err}_{\mathsf{high}} \\
&= \mathbf{s}^{\mathsf{T}}\left\lfloor\frac{\mathbf{A}_{\mathsf{high}}}{M}\right\rfloor + \mathbf{e}_{\mathbf{s},\mathsf{high}}^{\mathsf{T}} - f_{\mathsf{high}}(\mathbf{x},\mathsf{sd}) + \mathsf{err}_{\mathsf{high}}^{\mathsf{T}} \qquad (14)
\end{aligned}$$

where $\mathsf{err}_{\mathsf{high}}^{\mathsf{T}} \in \{0,1\}^{\ell}$, is the rounding error which is 1 if $\left\|(\mathbf{s}^{\mathsf{T}}\mathbf{A}_{\mathsf{high}})^{\mathsf{T}} + \mathbf{e}_{\mathsf{fhe}}^{\mathsf{T}}\mathbf{R}_{\mathsf{high}} + \mathbf{e}_{\mathsf{att}}^{\mathsf{T}}\mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}}\right\| \ge M$ and 0 otherwise, and $\|\mathbf{e}_{\mathbf{s},\mathsf{high}}\| \le (n+1)\cdot\|\mathbf{s}\|$. To see the latter, we use the fact that $\lfloor\mathbf{s}^{\mathsf{T}}X\rfloor - \mathbf{s}^{\mathsf{T}}\lfloor X\rfloor = \lfloor\mathbf{s}^{\mathsf{T}}X - \mathbf{s}^{\mathsf{T}}\lfloor X\rfloor\rfloor = \lfloor\mathbf{s}^{\mathsf{T}}(X - \lfloor X\rfloor)\rfloor$, where $X - \lfloor X\rfloor < 1$. So $\mathbf{e}_{\mathbf{s},\mathsf{high}}^{\mathsf{T}} = \left\lfloor\mathbf{s}^{\mathsf{T}}\left(\frac{\mathbf{A}_{\mathsf{high}}}{M} - \left\lfloor\frac{\mathbf{A}_{\mathsf{high}}}{M}\right\rfloor\right)\right\rfloor$ and $\|\mathbf{e}_{\mathbf{s},\mathsf{high}}\| \le \|\mathbf{s}\|\left\|\left(\frac{\mathbf{A}_{\mathsf{high}}}{M} - \left\lfloor\frac{\mathbf{A}_{\mathsf{high}}}{M}\right\rfloor\right)^{\mathsf{T}}\right\| < (n+1)\|\mathbf{s}\|$.

Using a similar analysis as to obtain Equation (14), we get

$$\left\lfloor\frac{\mathbf{c}_{\mathsf{att}}^{\mathsf{T}}\cdot\mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{low}}}}{M}\right\rfloor = \mathbf{s}^{\mathsf{T}}\left\lfloor\frac{\mathbf{A}_{\mathsf{low}}}{M}\right\rfloor + \mathbf{e}_{\mathbf{s},\mathsf{low}}^{\mathsf{T}} - f_{\mathsf{low}}(\mathbf{x},\mathsf{sd}) + \mathsf{err}_{\mathsf{low}}^{\mathsf{T}} \qquad (15)$$

where $\mathsf{err}_{\mathsf{low}}^{\mathsf{T}} \in \{0,1\}^{\ell}$ and $\|\mathbf{e}_{\mathbf{s},\mathsf{low}}\| \le (n+1)\cdot\|\mathbf{s}\|$. Using Equation (14) and Equation (15), we get

$$\begin{aligned}
&M\cdot\left\lfloor\frac{\mathbf{c}_{\mathsf{att}}^{\mathsf{T}}\cdot\mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}}}{M}\right\rfloor + \left\lfloor\frac{\mathbf{c}_{\mathsf{att}}^{\mathsf{T}}\cdot\mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{low}}}}{M}\right\rfloor \\
&= \mathbf{s}^{\mathsf{T}}\left(M\cdot\left\lfloor\frac{\mathbf{A}_{\mathsf{high}}}{M}\right\rfloor + \left\lfloor\frac{\mathbf{A}_{\mathsf{low}}}{M}\right\rfloor\right) - (M\cdot f_{\mathsf{high}}(\mathbf{x},\mathsf{sd}) + f_{\mathsf{low}}(\mathbf{x},\mathsf{sd})) + \mathsf{err} \\
&= \mathbf{s}^{\mathsf{T}}\left(M\cdot\left\lfloor\frac{\mathbf{A}_{\mathsf{high}}}{M}\right\rfloor + \left\lfloor\frac{\mathbf{A}_{\mathsf{low}}}{M}\right\rfloor\right) - \mathrm{F}[f,\mathbf{r}](\mathbf{x},\mathsf{sd}) + \mathsf{err} \qquad (16)
\end{aligned}$$

where

$$\begin{aligned}
\mathsf{err} &= M\cdot\mathbf{e}_{\mathbf{s},\mathsf{high}}^{\mathsf{T}} + \mathbf{e}_{\mathbf{s},\mathsf{low}}^{\mathsf{T}} + M\cdot\mathsf{err}_{\mathsf{high}} + \mathsf{err}_{\mathsf{low}} \\
&= M\cdot\left(\mathbf{s}^{\mathsf{T}}\left\lfloor\frac{\mathbf{A}_{\mathsf{high}}}{M}\right\rfloor - \left\lfloor\mathbf{s}^{\mathsf{T}}\frac{\mathbf{A}_{\mathsf{high}}}{M}\right\rfloor\right) + \left(\mathbf{s}^{\mathsf{T}}\left\lfloor\frac{\mathbf{A}_{\mathsf{low}}}{M}\right\rfloor - \left\lfloor\mathbf{s}^{\mathsf{T}}\frac{\mathbf{A}_{\mathsf{low}}}{M}\right\rfloor\right) + M\cdot\mathsf{err}_{\mathsf{high}} + \mathsf{err}_{\mathsf{low}} \qquad (17)
\end{aligned}$$

where $\mathsf{err}_{\mathsf{high}}, \mathsf{err}_{\mathsf{low}} \in \{0,1\}^{\ell}$ are rounding errors and matrices $\mathbf{A}_{\mathsf{high}}, \mathbf{A}_{\mathsf{low}}$ are publicly computable matrices and

$$\begin{aligned}
\|\mathsf{err}\| \le& M\cdot((n+1)\cdot\|\mathbf{s}\|+1) + (n+1)\cdot\|\mathbf{s}\| + 1 \le 2M\cdot((n+1)\cdot\|\mathbf{s}\|+1) \\
=& 2^{4\lambda+1}\beta\left((n+1)\cdot 2^{2\lambda+1}\sqrt{\lambda}\right) \le 2^{7\lambda}\beta
\end{aligned}$$

– Next, we note that

$$\mathbf{c}_{\mathbf{B}}^{\mathsf{T}}\cdot\mathbf{K} = \mathbf{s}^{\mathsf{T}}\left(M\cdot\left\lfloor\frac{\mathbf{A}_{\mathsf{high}}}{M}\right\rfloor + \left\lfloor\frac{\mathbf{A}_{\mathsf{low}}}{M}\right\rfloor\right) + \mathbf{c}_{\mathbf{B}}^{\mathsf{T}}\cdot\mathbf{K}. \qquad (18)$$

where $\|(\mathbf{c}_{\mathbf{B}}^{\mathsf{T}}\cdot\mathbf{K})^{\mathsf{T}}\| \le 2^{9\lambda}\beta\sqrt{\lambda}\cdot\tau$ from our parameter setting.

– Using Equations (16) and (18), we get

$$\begin{aligned}\mathbf{z} = \mathbf{c}_{\mathbf{B}}^{\mathsf{T}} \cdot \mathbf{K} - \left(M \cdot \left\lfloor \frac{\mathbf{c}_{\mathsf{att}}^{\mathsf{T}} \cdot \mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}}}{M} \right\rceil + \left\lfloor \frac{\mathbf{c}_{\mathsf{att}}^{\mathsf{T}} \cdot \mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{low}}}}{M} \right\rceil \right) \\ = \mathrm{F}[f, \mathbf{r}](\mathbf{x}, \mathsf{sd}) + \mathbf{e}_{\mathbf{B}}^{\mathsf{T}} \cdot \mathbf{K} - \mathsf{err} \\ = f(\mathbf{x}) \lfloor q/2 \rceil + \mathsf{PRF}(\mathsf{sd}, \mathbf{r}) + \mathbf{e}_{\mathbf{B}}^{\mathsf{T}} \cdot \mathbf{K} - \mathsf{err}\end{aligned}$$

where

$$\begin{aligned}\left\|\mathsf{PRF}(\mathsf{sd}, \mathbf{r}) + \mathbf{e}_{\mathbf{B}}^{\mathsf{T}} \cdot \mathbf{K} - \mathsf{err}\right\| &\leq \|\mathsf{PRF}(\mathsf{sd}, \mathbf{r})\| + 2^{9\lambda}\beta\sqrt{\lambda} \cdot \tau + 2^{7\lambda}\beta \\ &\leq \|\mathsf{PRF}(\mathsf{sd}, \mathbf{r})\| + 2^{9\lambda+1}\beta\sqrt{\lambda} \cdot \tau < q/4 - B + B < q/4\end{aligned}$$

Hence the last step of decryption outputs $\mathbf{y}$ correctly with probability 1.

4.3 Security Proof for Pseudorandom Functionalities

Theorem 4.6. Let $\mathcal{SC}_{\mathsf{prFE}}$ be a sampler class for prFE. Assuming LWE (Assumption 3.5) and private coin Evasive LWE (Assumption 3.6) with respect to the sampler class that contains all $\mathsf{Samp}_{\mathsf{evs}}(1^{\lambda})$ induced by $\mathsf{Samp}_{\mathsf{prFE}} \in \mathcal{SC}_{\mathsf{prFE}}$ as defined in Figure 1, our prFE scheme satisfies prCT security with respect to $\mathcal{SC}_{\mathsf{prFE}}$ as defined in Definition 4.2.

Proof. Consider a sampler $\mathsf{Samp}_{\mathsf{prFE}}$ that generates the following:

1. **Key Queries.** It issues Q_{key} number of functions $f_1, \ldots, f_{Q_{\mathsf{key}}}$ for key queries.
2. **Ciphertext Queries.** It issues Q_{msg} ciphertext queries $\mathbf{x}_1, \ldots, \mathbf{x}_{Q_{\mathsf{msg}}}$.
3. **Auxiliary Information.** It outputs the auxiliary information $\mathsf{aux}_{\mathcal{A}}$.

To prove the prCT security as per Definition 4.2, we show

$$\begin{pmatrix} \mathsf{mpk} = (\mathbf{A}_{\mathsf{att}}, \mathbf{B}, M),\ \mathsf{aux}_{\mathcal{A}},\ \ \mathbf{C}_{\mathbf{B}} = \mathbf{S}\mathbf{B} + \mathbf{E}_{\mathbf{B}}, \\ \{\mathbf{X}_j = \mathbf{A}_{\mathsf{fhe},j}\mathbf{R}_j - (\mathbf{x}_j, \mathsf{sd}_j) \otimes \mathbf{G}\}_{j \in [Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}_{\mathsf{att},j}^{\mathsf{T}} = \mathbf{s}_j^{\mathsf{T}}(\mathbf{A}_{\mathsf{att}} - \mathsf{bits}(1, \mathbf{X}_j) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{att},j}^{\mathsf{T}}\}_{j \in [Q_{\mathsf{msg}}]}, \\ \{\mathbf{K}_k, \mathbf{r}_k\}_{k \in [Q_{\mathsf{key}}]} \end{pmatrix} \approx_c \begin{pmatrix} \mathsf{mpk} = (\mathbf{A}_{\mathsf{att}}, \mathbf{B}, M),\ \mathsf{aux}_{\mathcal{A}},\ \ \mathbf{C}_{\mathbf{B}} \leftarrow \mathbb{Z}_q^{Q_{\mathsf{msg}} \times mw}, \\ \{\mathbf{X}_j \leftarrow \mathbb{Z}_q^{(n+1) \times m(\lambda + \mathrm{L})}\}_{j \in [Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}_{\mathsf{att},j} \leftarrow \mathbb{Z}_q^{(\mathrm{L}_{\mathrm{X}}+1)m}\}_{j \in [Q_{\mathsf{msg}}]}, \\ \{\mathbf{K}_k, \mathbf{r}_k\}_{k \in [Q_{\mathsf{key}}]} \end{pmatrix} \tag{19}$$

where

$$\mathbf{S} = \begin{pmatrix} \mathbf{s}_1^{\mathsf{T}} \\ \vdots \\ \mathbf{s}_{Q_{\mathsf{msg}}}^{\mathsf{T}} \end{pmatrix},\ \mathbf{E}_{\mathbf{B}} = \begin{pmatrix} \mathbf{e}_{\mathbf{B},1}^{\mathsf{T}} \\ \vdots \\ \mathbf{e}_{\mathbf{B},Q_{\mathsf{msg}}}^{\mathsf{T}} \end{pmatrix},$$

$$(\mathsf{aux}_{\mathcal{A}}, \{f_k\}_{k \in [Q_{\mathsf{key}}]}, \{\mathbf{x}_j\}_{j \in [Q_{\mathsf{msg}}]}) \leftarrow \mathsf{Samp}_{\mathsf{prFE}}(1^{\lambda})$$

and for $j \in [Q_{\mathsf{msg}}]$, $\mathbf{s}_j, \mathbf{e}_{\mathbf{B},j}, \mathbf{A}_{\mathsf{fhe},j}, \mathbf{R}_j, \mathsf{sd}_j, \mathbf{e}_{\mathsf{att},j}$ are sampled as in the construction, for $k \in [Q_{\mathsf{key}}]$, we have $\mathbf{r}_k \leftarrow \{0,1\}^{\lambda}$, $\mathrm{F}_k = \mathrm{F}[f_k, \mathbf{r}_k]$ and $\mathbf{A}_{\mathrm{F}_k}$ is as defined in the construction, and $\mathbf{K}_k = \mathbf{B}_{\tau}^{-1}(\mathbf{A}_{\mathrm{F}_k})$

assuming we have

$$(1^{\lambda},\ \mathsf{aux}_{\mathcal{A}},\ \{f_k,\ f_k(x_j)\}_{j \in [Q_{\mathsf{msg}}], k \in [Q_{\mathsf{key}}]}) \approx_c (1^{\lambda},\ \mathsf{aux}_{\mathcal{A}},\ \{f_k,\ \Delta_{j,k} \leftarrow \{0,1\}^{\ell}\}_{j \in [Q_{\mathsf{msg}}], k \in [Q_{\mathsf{key}}]}). \tag{20}$$

$\mathsf{Samp}_{\mathsf{evs}}(1^\lambda)$

The sampler does the following.

- Runs the prFE sampler $\mathsf{Samp}_{\mathsf{prFE}}$ to obtain $(\{f_k\}_{k\in Q_{\mathsf{key}}}, \{\mathbf{x}_j\}_{j\in[Q_{\mathsf{msg}}]}, \mathsf{aux}_{\mathcal{A}})$ where $f_k : \{0,1\}^{\mathrm{L}} \to \{0,1\}^\ell$, $\mathbf{x}_j \in \{0,1\}^{\mathrm{L}}$ and $\mathsf{aux}_{\mathcal{A}} \in \{0,1\}^\star$.
- Set appropriate parameters as in Equation (12)[a].
- Samples $\mathsf{sd}_j \leftarrow \{0,1\}^\lambda$, $\bar{\mathbf{A}}_{\mathsf{fhe},j} \leftarrow \mathbb{Z}_q^{n\times m}$, $\mathbf{e}_{\mathsf{fhe},j} \leftarrow \mathcal{D}^m_{\mathbb{Z},\sigma}$, $\mathbf{R}_j \leftarrow \{0,1\}^{m\times m(\lambda+\mathrm{L})}$ and computes $\mathbf{X}_j = \mathbf{A}_{\mathsf{fhe},j}\mathbf{R}_j - (\mathbf{x}_j, \mathsf{sd}_j)\otimes\mathbf{G}$ for $j\in[Q_{\mathsf{msg}}]$ where $\mathbf{A}_{\mathsf{fhe},j} = \begin{pmatrix} \bar{\mathbf{A}}_{\mathsf{fhe},j} \\ \bar{\mathbf{s}}^\mathsf{T}\bar{\mathbf{A}}_{\mathsf{fhe},j} + \mathbf{e}^\mathsf{T}_{\mathsf{fhe},j}\end{pmatrix}$ $\forall\, j\in[Q_{\mathsf{msg}}]$.
- Samples $\bar{\mathbf{s}}_j \leftarrow \mathcal{D}^n_{\mathbb{Z},\sigma_s}$, $\mathbf{e}_{\mathsf{att},j} \leftarrow \mathcal{D}^{(\mathrm{L_X}+1)m}_{\mathbb{Z},\sigma}$, sets $\mathbf{s}_j = (\bar{\mathbf{s}}_j^\mathsf{T}, -1)^\mathsf{T}$ and computes $\mathbf{c}^\mathsf{T}_{\mathsf{att},j} = \mathbf{s}_j^\mathsf{T}(\mathbf{A}_{\mathsf{att}} - \mathsf{bits}(1,\mathbf{X}_j)\otimes\mathbf{G}) + \mathbf{e}^\mathsf{T}_{\mathsf{att},j}$ $\forall\, j\in[Q_{\mathsf{msg}}]$.
- Samples $\mathbf{r}_k \leftarrow \{0,1\}^\lambda$, defines $\mathrm{F}[f_k,\mathbf{r}_k]$ and computes $\mathbf{A}_{\mathrm{F}_k}$, for $k\in[Q_{\mathsf{key}}]$, as in the key generation algorithm.
- It outputs

$$\mathbf{S} = \begin{pmatrix} \mathbf{s}_1^\mathsf{T} \\ \vdots \\ \mathbf{s}^\mathsf{T}_{Q_{\mathsf{msg}}}\end{pmatrix}, \quad \mathbf{P} = [\mathbf{A}_{\mathrm{F}_1}||\ldots||\mathbf{A}_{\mathrm{F}_{Q_{\mathsf{key}}}}]$$

$$\mathsf{aux}_1 = \Big(\{\mathbf{X}_j = \mathbf{A}_{\mathsf{fhe},j}\mathbf{R}_j - (\mathbf{x}_j,\mathsf{sd}_j)\otimes\mathbf{G}\}_{j\in[Q_{\mathsf{msg}}]}, \{\mathbf{c}^\mathsf{T}_{\mathsf{att},j} = \mathbf{s}_j^\mathsf{T}(\mathbf{A}_{\mathsf{att}} - \mathsf{bits}(1,\mathbf{X}_j)\otimes\mathbf{G}) + \mathbf{e}^\mathsf{T}_{\mathsf{att},j}\}_{j\in[Q_{\mathsf{msg}}]}\Big),$$

$$\mathsf{aux}_2 = (f_1,\ldots,f_{Q_{\mathsf{key}}}, \mathsf{aux}_{\mathcal{A}}, \mathbf{r}_1,\ldots,\mathbf{r}_{Q_{\mathsf{key}}}, \mathbf{A}_{\mathsf{att}}, M).$$

[a] We assume the parameters to be output as a part of aux_2, even though we do not explicitly write so.

Figure 1: Description of the Sampler for Evasive LWE

We invoke evasive LWE assumption for a matrix $\mathbf{B}$ with the private coin sampler $\mathsf{Samp}_{\mathsf{evs}}$ that outputs $(\mathbf{S},\mathbf{P},\mathsf{aux} = (\mathsf{aux}_1,\mathsf{aux}_2))$ with private coin $\mathsf{coins}^{\mathsf{Samp}_{\mathsf{evs}}}_{\mathsf{priv}} = \{\mathsf{sd}_j, \mathbf{R}_j, \mathbf{e}_{\mathsf{att},j}, \mathbf{A}_{\mathsf{fhe},j}\}_{j\in[Q_{\mathsf{msg}}]}$, defined as follows.

By Lemma 3.8, to prove Equation (19) assuming evasive LWE, it suffices to show

$$\begin{pmatrix} \mathsf{aux}_2,\ \mathbf{B},\ \mathbf{C}_\mathbf{B} = \mathbf{S}\mathbf{B} + \mathbf{E}_\mathbf{B}, \\ \{\mathbf{X}_j = \mathbf{A}_{\mathsf{fhe},j}\mathbf{R}_j - (\mathbf{x}_j,\mathsf{sd}_j)\otimes\mathbf{G}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}^\mathsf{T}_{\mathsf{att},j} = \mathbf{s}_j^\mathsf{T}(\mathbf{A}_{\mathsf{att}} - \mathsf{bits}(1,\mathbf{X}_j)\otimes\mathbf{G}) + \mathbf{e}^\mathsf{T}_{\mathsf{att},j}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \mathbf{C}_\mathbf{P} = \mathbf{S}\mathbf{P} + \mathbf{E}_\mathbf{P} \end{pmatrix} \approx_c \begin{pmatrix} \mathsf{aux}_2,\ \mathbf{B},\ \mathbf{C}_\mathbf{B} \leftarrow \mathbb{Z}_q^{Q_{\mathsf{msg}}\times mw}, \\ \{\mathbf{X}_j \leftarrow \mathbb{Z}_q^{(n+1)\times m(\lambda+\mathrm{L})}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}_{\mathsf{att},j} \leftarrow \mathbb{Z}_q^{(\mathrm{L_X}+1)m}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \mathbf{C}_\mathbf{P} \leftarrow \mathbb{Z}_q^{Q_{\mathsf{msg}}\times\ell\cdot Q_{\mathsf{key}}} \end{pmatrix} \tag{21}$$

where $\mathbf{E}_\mathbf{P} \leftarrow \mathcal{D}^{Q_{\mathsf{msg}}\times\ell\cdot Q_{\mathsf{key}}}_{\mathbb{Z},\sigma_1}$. Using the representation

$$\mathbf{C}_\mathbf{B} = \begin{pmatrix} \mathbf{c}^\mathsf{T}_{\mathbf{B},1} = \mathbf{s}_1^\mathsf{T}\mathbf{B} + \mathbf{e}^\mathsf{T}_{\mathbf{B},1} \\ \vdots \\ \mathbf{c}^\mathsf{T}_{\mathbf{B},Q_{\mathsf{msg}}} = \mathbf{s}^\mathsf{T}_{Q_{\mathsf{msg}}}\mathbf{B} + \mathbf{e}^\mathsf{T}_{\mathbf{B},Q_{\mathsf{msg}}}\end{pmatrix} = \{\mathbf{c}^\mathsf{T}_{\mathbf{B},j}\}_{j\in[Q_{\mathsf{msg}}]},$$

$$\mathbf{C}_\mathbf{P} = \begin{pmatrix} \mathbf{c}^\mathsf{T}_{\mathbf{P},1} = \mathbf{s}_1^\mathsf{T}\mathbf{A}_{\mathrm{F}_1} + \mathbf{e}^\mathsf{T}_{\mathbf{P},1,1}||\ldots||\mathbf{s}_1^\mathsf{T}\mathbf{A}_{\mathrm{F}_{Q_{\mathsf{key}}}} + \mathbf{e}^\mathsf{T}_{\mathbf{P},1,Q_{\mathsf{key}}} \\ \vdots \\ \mathbf{c}^\mathsf{T}_{\mathbf{P},Q_{\mathsf{msg}}} = \mathbf{s}^\mathsf{T}_{Q_{\mathsf{msg}}}\mathbf{A}_{\mathrm{F}_1} + \mathbf{e}^\mathsf{T}_{\mathbf{P},Q_{\mathsf{msg}},1}||\ldots||\mathbf{s}^\mathsf{T}_{Q_{\mathsf{msg}}}\mathbf{A}_{\mathrm{F}_{Q_{\mathsf{key}}}} + \mathbf{e}^\mathsf{T}_{\mathbf{P},Q_{\mathsf{msg}},Q_{\mathsf{key}}}\end{pmatrix} = \{\mathbf{c}^\mathsf{T}_{\mathbf{P},j,k}\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]},$$

we rewrite Equation (21) as follows.

$$\begin{pmatrix} \mathsf{aux}_2,\ \mathbf{B}, \{\mathbf{c}_{\mathbf{B},j}^{\mathsf{T}} = \mathbf{s}_j^{\mathsf{T}}\mathbf{B} + \mathbf{e}_{\mathbf{B},j}^{\mathsf{T}}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{X}_j = \mathbf{A}_{\mathsf{fhe},j}\mathbf{R}_j - (\mathbf{x}_j, \mathsf{sd}_j)\otimes\mathbf{G}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}_{\mathsf{att},j}^{\mathsf{T}} = \mathbf{s}_j^{\mathsf{T}}(\mathbf{A}_{\mathsf{att}} - \mathsf{bits}(1,\mathbf{X}_j)\otimes\mathbf{G}) + \mathbf{e}_{\mathsf{att},j}^{\mathsf{T}}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}_{\mathbf{P},j,k}^{\mathsf{T}} = \mathbf{s}_j^{\mathsf{T}}\mathbf{A}_{\mathrm{F}_k} + \mathbf{e}_{\mathbf{P},j,k}^{\mathsf{T}}\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]} \end{pmatrix} \approx_c \begin{pmatrix} \mathsf{aux}_2,\ \mathbf{B}, \{\mathbf{c}_{\mathbf{B},j} \leftarrow \mathbb{Z}_q^{mw}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{X}_j \leftarrow \mathbb{Z}_q^{(n+1)\times m(\lambda+\mathrm{L})}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}_{\mathsf{att},j} \leftarrow \mathbb{Z}_q^{(\mathrm{L_X}+1)m}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}_{\mathbf{P},j,k} \leftarrow \mathbb{Z}_q^{\ell}\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]} \end{pmatrix} \quad (22)$$

where $\mathbf{e}_{\mathbf{P},j,k} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma_1}^{\ell}$. Now, to prove Equation (19) it suffices to show Equation (22).
We prove Equation (22) via the following sequence of hybrids.

Hyb_0. This is L.H.S distribution of Equation (22).

Hyb_1. This hybrid is same as Hyb_0, except we compute $\mathbf{c}_{\mathbf{P},j,k}^{\mathsf{T}}$ as

$$\mathbf{c}_{\mathbf{P},j,k}^{\mathsf{T}} = M\cdot\left\lfloor \frac{\mathbf{c}_{\mathsf{att},j}^{\mathsf{T}}\cdot\mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}_j}^{\mathrm{F}_{\mathsf{high},k}}}{M}\right\rfloor + \left\lfloor \frac{\mathbf{c}_{\mathsf{att},j}^{\mathsf{T}}\cdot\mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}_j}^{\mathrm{F}_{\mathsf{low},k}}}{M}\right\rfloor + f_k(\mathbf{x}_j)\lfloor q/2\rceil + \mathsf{PRF}(\mathsf{sd}_j,\mathbf{r}_k) + \mathbf{e}_{\mathbf{P},j,k}^{\mathsf{T}}$$

where $\mathbf{e}_{\mathbf{P},j,k} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma_1}^{\ell}$. We claim that Hyb_0 and Hyb_1 are statistically indistinguishable. To see this, we observe the following.

– From Equation (16) we note that

$$\begin{aligned} M\cdot\left\lfloor \frac{\mathbf{c}_{\mathsf{att},j}^{\mathsf{T}}\cdot\mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}_j}^{\mathrm{F}_{\mathsf{high},k}}}{M}\right\rfloor + \left\lfloor \frac{\mathbf{c}_{\mathsf{att},j}^{\mathsf{T}}\cdot\mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}_j}^{\mathrm{F}_{\mathsf{low},k}}}{M}\right\rfloor + f_k(\mathbf{x}_j)\lfloor q/2\rceil + \mathsf{PRF}(\mathsf{sd}_j,\mathbf{r}_k) + \mathbf{e}_{\mathbf{P},j,k}^{\mathsf{T}} \\ = \mathbf{s}_j^{\mathsf{T}}\left(M\cdot\left\lfloor\frac{\mathbf{A}_{\mathsf{high},k}}{M}\right\rfloor + \left\lfloor\frac{\mathbf{A}_{\mathsf{low},k}}{M}\right\rfloor\right) + \mathsf{err}_{j,k} + \mathbf{e}_{\mathbf{P},j,k}^{\mathsf{T}} \\ = \mathbf{s}_j^{\mathsf{T}}\mathbf{A}_{\mathrm{F}_k} + \mathsf{err}_{j,k} + \mathbf{e}_{\mathbf{P},j,k}^{\mathsf{T}} \end{aligned}$$

where $\|\mathsf{err}_{j,k}\| \le 2^{7\lambda}\beta$.

– Next, we note that $\|\mathsf{err}_{j,k}\| \le 2^{8\lambda+O(1)}\beta/\mathrm{poly}(\lambda) = \chi_1 = \|\mathbf{e}_{\mathbf{P},j,k}\|$. Thus by noise flooding (Lemma 3.3) we have $\mathbf{e}_{\mathbf{P},j,k}^{\mathsf{T}} \approx_s \mathsf{err}_{j,k} + \mathbf{e}_{\mathbf{P},j,k}^{\mathsf{T}}$ with a statistical distance of $\mathrm{poly}(\lambda)2^{-\lambda}$.

From the above, we have

$$\Delta(\mathsf{Hyb}_0,\mathsf{Hyb}_1) = \frac{Q_{\mathsf{key}}\cdot Q_{\mathsf{msg}}\cdot\mathrm{poly}(\lambda)}{2^{\lambda}}.$$

Thus, it suffices to show pseudorandomness of the following distribution given aux_2

$$\begin{pmatrix} \mathbf{B},\ \{\mathbf{c}_{\mathbf{B},j}^{\mathsf{T}} = \mathbf{s}_j^{\mathsf{T}}\mathbf{B} + \mathbf{e}_{\mathbf{B},j}^{\mathsf{T}}\}_{j\in[Q_{\mathsf{msg}}]} \\ \{\mathbf{X}_j = \mathbf{A}_{\mathsf{fhe},j}\mathbf{R}_j - (\mathbf{x}_j,\mathsf{sd}_j)\otimes\mathbf{G}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}_{\mathsf{att},j}^{\mathsf{T}} = \mathbf{s}_j^{\mathsf{T}}(\mathbf{A}_{\mathsf{att}} - \mathsf{bits}(1,\mathbf{X}_j)\otimes\mathbf{G}) + \mathbf{e}_{\mathsf{att},j}^{\mathsf{T}}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\tilde{\mathrm{F}}_{j,k} = f_k(\mathbf{x}_j)\lfloor q/2\rceil + \mathsf{PRF}(\mathsf{sd}_j,\mathbf{r}_k) + \mathbf{e}_{\mathbf{P},j,k}^{\mathsf{T}}\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]}. \end{pmatrix}$$

Hyb_2. This hybrid is same as Hyb_1 except that for all $j\in[Q_{\mathsf{msg}}]$ we sample $\mathbf{c}_{\mathbf{B},j} \leftarrow \mathbb{Z}_q^{mw}$, $\mathbf{c}_{\mathsf{att},j} \leftarrow \mathbb{Z}_q^{(\mathrm{L_X}+1)m}$ and $\mathbf{A}_{\mathsf{fhe},j} \leftarrow \mathbb{Z}_q^{(n+1)\times m}$, where $\mathbf{A}_{\mathsf{fhe},j}$ is the fhe public key used to compute $\mathbf{X}_j$. We have $\mathsf{Hyb}_1 \approx_c \mathsf{Hyb}_2$ using LWE. To prove this we consider sub-hybrids $\mathsf{Hyb}_{1.i}$ for $i\in[Q_{\mathsf{msg}}]$, where in $\mathsf{Hyb}_{1.i}$ we sample $\mathbf{c}_{\mathbf{B},j} \leftarrow \mathbb{Z}_q^{mw}$, $\mathbf{c}_{\mathsf{att},j} \leftarrow \mathbb{Z}_q^{(\mathrm{L_X}+1)m}$ and $\mathbf{A}_{\mathsf{fhe},j} \leftarrow \mathbb{Z}_q^{(n+1)\times m}$ for $1\le j\le i$. We set $\mathsf{Hyb}_1 = \mathsf{Hyb}_{1.0}$ and $\mathsf{Hyb}_2 = \mathsf{Hyb}_{1.Q_{\mathsf{msg}}}$. Next, we prove that for all $i\in[Q_{\mathsf{msg}}]$, $\mathsf{Hyb}_{1.i-1} \approx_c \mathsf{Hyb}_{1.i}$ via the following claim.

Claim 4.7. $\mathsf{Hyb}_{1.i-1} \approx_c \mathsf{Hyb}_{1.i}$, for $i \in [Q_{\mathsf{msg}}]$, assuming the security of LWE.

Proof. We show that if there exists an adversary $\mathcal{A}$ who can distinguish between the two hybrids with non-negligible advantage, then there is a reduction $\mathcal{B}$ that breaks LWE security with non-negligible advantage. The reduction is as follows.

1. The adversary $\mathcal{A}$ sends the function queries $f_1, \ldots, f_{Q_{\mathsf{key}}}$, message queries $\mathbf{x}_1, \ldots, \mathbf{x}_{Q_{\mathsf{msg}}}$ and auxiliary input $\mathsf{aux}_{\mathcal{A}}$ to the reduction.
2. $\mathcal{B}$ initiates the LWE security game with the LWE challenger. The challenger sends $\mathbf{A}_{\mathsf{LWE}} \in \mathbb{Z}_q^{n \times mw+m+(\mathrm{L_X}+1)m}$ and $\mathbf{b} \in \mathbb{Z}_q^{mw+m+(\mathrm{L_X}+1)m}$ to $\mathcal{B}$.
3. $\mathcal{B}$ parses $\mathbf{A}_{\mathsf{LWE}} = (\mathbf{B}',\ \hat{\mathbf{A}}_{\mathsf{fhe}},\ \mathbf{A}'_{\mathsf{att}})$, where $\mathbf{B}' \in \mathbb{Z}_q^{n \times mw}, \hat{\mathbf{A}}_{\mathsf{fhe}} \in \mathbb{Z}_q^{n \times m}, \mathbf{A}'_{\mathsf{att}} \in \mathbb{Z}_q^{n \times (\mathrm{L_X}+1)m}$ and $\mathbf{b}^{\mathsf{T}} = (\mathbf{b}_{\mathbf{B}}^{\mathsf{T}},\ \mathbf{b}_{\mathsf{fhe}}^{\mathsf{T}},\ \mathbf{b}_{\mathsf{att}}^{\mathsf{T}})$. For $j \in [Q_{\mathsf{msg}}]$, it computes $\mathbf{c}_{\mathbf{B},j},\ \mathbf{c}_{\mathsf{att},j}$ and $\mathbf{A}_{\mathsf{fhe},j}$ as follows.
   - For $1 \le j < i$: $\mathcal{B}$ samples $\mathbf{c}_{\mathbf{B},j} \leftarrow \mathbb{Z}_q^{mw},\ \mathbf{c}_{\mathsf{att},j} \leftarrow \mathbb{Z}_q^{(\mathrm{L_X}+1)m}$ and $\mathbf{A}_{\mathsf{fhe},j} \leftarrow \mathbb{Z}_q^{(n+1)\times m}$.
   - For $j = i$: $\mathcal{B}$ does the following.
     - Samples $\underline{\mathbf{b}} \leftarrow \mathbb{Z}_q^{mw}$ and sets $\mathbf{B} = \begin{pmatrix} \mathbf{B}' \\ \underline{\mathbf{b}}^{\mathsf{T}} \end{pmatrix}$ and $\mathbf{c}_{\mathbf{B},i}^{\mathsf{T}} := \mathbf{b}_{\mathbf{B}}^{\mathsf{T}} - \underline{\mathbf{b}}^{\mathsf{T}}$.
     - Sets $\mathbf{A}_{\mathsf{fhe},i} := \begin{pmatrix} \hat{\mathbf{A}}_{\mathsf{fhe}} \\ \mathbf{b}_{\mathsf{fhe}}^{\mathsf{T}} \end{pmatrix}$ and computes $\mathbf{X}_i = \mathbf{A}_{\mathsf{fhe},i}\mathbf{R}_i - (\mathbf{x}_i, \mathsf{sd}_i) \otimes \mathbf{G}$ as in the construction.
     - Sets $\bar{\mathbf{A}}_{\mathsf{att}} = \mathbf{A}'_{\mathsf{att}} + \mathsf{bits}(1, \mathbf{X}_i) \otimes \bar{\mathbf{G}}$, $\mathbf{A}_{\mathsf{att}} = \begin{pmatrix} \bar{\mathbf{A}}_{\mathsf{att}} \\ \underline{\mathbf{a}}_{\mathsf{att}}^{\mathsf{T}} \end{pmatrix}$, where $\underline{\mathbf{a}}_{\mathsf{att}} \leftarrow \mathbb{Z}_q^{(\mathrm{L_X}+1)m}$, and $\mathbf{c}_{\mathsf{att},i}^{\mathsf{T}} = \mathbf{b}_{\mathsf{att}}^{\mathsf{T}} - (\underline{\mathbf{a}}_{\mathsf{att}}^{\mathsf{T}} - \mathsf{bits}(1, \mathbf{X}_i) \otimes \underline{\mathbf{G}})$, where $\bar{\mathbf{G}}$ and $\underline{\mathbf{G}}$ denotes the first n rows and $n+1$-th row of the gadget matrix $\mathbf{G} \in \mathbb{Z}_q^{(n+1)\times m}$, respectively.
   - For $j > i$: $\mathcal{B}$ computes $\mathbf{c}_{\mathbf{B},j}^{\mathsf{T}}, \mathbf{X}_j$ and $\mathbf{c}_{\mathsf{att},j}^{\mathsf{T}}$ as in the construction, where $\mathbf{c}_{\mathsf{att},j}^{\mathsf{T}}$ is computed using $\mathbf{A}_{\mathsf{att}} = \begin{pmatrix} \bar{\mathbf{A}}_{\mathsf{att}} \\ \underline{\mathbf{a}}_{\mathsf{att}}^{\mathsf{T}} \end{pmatrix}$.
4. $\mathcal{B}$ sets $\mathsf{aux}_2 = (f_1, \ldots, f_{Q_{\mathsf{key}}}, \mathsf{aux}_{\mathcal{A}}, \mathbf{r}_1, \ldots, \mathbf{r}_{Q_{\mathsf{key}}}, \mathbf{A}_{\mathsf{att}}, M)$ where $\mathbf{r}_k \leftarrow \{0,1\}^{\lambda}$ and computes $\tilde{\mathrm{F}}_{j,k}$ as in Hyb_1. It sends $(\mathsf{aux}_2, \{\mathbf{c}_{\mathbf{B},j}^{\mathsf{T}}, \mathbf{X}_j, \mathbf{c}_{\mathsf{att},j}^{\mathsf{T}}, \tilde{\mathrm{F}}_{j,k}\})$ to the adversary.
5. $\mathcal{A}$ outputs a bit β'. $\mathcal{B}$ forwards the bit β' to the LWE challenger.

We note that if the LWE challenger sent $\mathbf{b} = \bar{\mathbf{s}}\mathbf{A}_{\mathsf{LWE}} + \mathbf{e}_{\mathsf{LWE}}$, then $\mathcal{B}$ simulated $\mathsf{Hyb}_{1,i-1}$ with $\mathcal{A}$ else if LWE challenger sent random $\mathbf{b} \leftarrow \mathbb{Z}_q^{mw+m+(\mathrm{L_X}+1)m}$ then $\mathcal{B}$ simulated $\mathsf{Hyb}_{1,i}$ with $\mathcal{A}$.

To see the former case, we note that if $\mathbf{b} = \bar{\mathbf{s}}\mathbf{A}_{\mathsf{LWE}} + \mathbf{e}_{\mathsf{LWE}}^{\mathsf{T}} = \bar{\mathbf{s}}(\mathbf{B}',\ \hat{\mathbf{A}}_{\mathsf{fhe}},\ \mathbf{A}'_{\mathsf{att}}) + (\mathbf{e}_{\mathbf{B}}^{\mathsf{T}},\ \mathbf{e}_{\mathsf{fhe}}^{\mathsf{T}},\ \mathbf{e}_{\mathsf{att}}^{\mathsf{T}})$, then $\mathbf{b}_{\mathbf{B}} = \bar{\mathbf{s}}\mathbf{B}' + \mathbf{e}_{\mathbf{B}}^{\mathsf{T}},\ \mathbf{b}_{\mathsf{fhe}} := \bar{\mathbf{s}}\hat{\mathbf{A}}_{\mathsf{fhe}} + \mathbf{e}_{\mathsf{fhe}}^{\mathsf{T}}$, and $\mathbf{b}_{\mathsf{att}} := \bar{\mathbf{s}}\mathbf{A}'_{\mathsf{att}} + \mathbf{e}_{\mathsf{att}}^{\mathsf{T}}$. Thus we have

$$\mathbf{c}_{\mathbf{B},i}^{\mathsf{T}} = (\bar{\mathbf{s}}, -1)\begin{pmatrix} \mathbf{B}' \\ \underline{\mathbf{b}}^{\mathsf{T}} \end{pmatrix} + \mathbf{e}_{\mathbf{B}}^{\mathsf{T}},\ \ \mathbf{A}_{\mathsf{fhe},i} = \begin{pmatrix} \hat{\mathbf{A}}_{\mathsf{fhe}} \\ \bar{\mathbf{s}}\hat{\mathbf{A}}_{\mathsf{fhe}} + \mathbf{e}_{\mathsf{fhe}}^{\mathsf{T}} \end{pmatrix},\ \ \mathbf{c}_{\mathsf{att},i}^{\mathsf{T}} = (\bar{\mathbf{s}}, -1)\left(\begin{pmatrix} \bar{\mathbf{A}}_{\mathsf{att}} \\ \underline{\mathbf{a}}_{\mathsf{att}}^{\mathsf{T}} \end{pmatrix} - \mathsf{bits}(1, \mathbf{X}_i) \otimes \mathbf{G}\right) + \mathbf{e}_{\mathsf{att}}^{\mathsf{T}}$$

To see the latter case, we note that if $\mathbf{b} \leftarrow \mathbb{Z}_q^{mw+m+(\mathrm{L_X}+1)m}$ then it implies $\mathbf{b}_{\mathbf{B}} \leftarrow \mathbb{Z}_q^{mw},\ \mathbf{b}_{\mathsf{fhe}} \leftarrow \mathbb{Z}_q^{m},\ \mathbf{b}_{\mathsf{att}} \leftarrow \mathbb{Z}_q^{(\mathrm{L_X}+1)m}$. This implies the following.

- Randomness of $\mathbf{b}_{\mathbf{B}}$ implies the randomness of $\mathbf{c}_{\mathbf{B},i}^{\mathsf{T}} := \mathbf{b}_{\mathbf{B}}^{\mathsf{T}} - \underline{\mathbf{b}}^{\mathsf{T}}$.
- Randomness of $\mathbf{b}_{\mathsf{fhe}}$ implies $\mathbf{A}_{\mathsf{fhe},i} \leftarrow \mathbb{Z}_q^{(n+1)\times m}$.
- Randomness of $\mathbf{b}_{\mathsf{att}}$ implies randomness of $\mathbf{c}_{\mathsf{att},i}^{\mathsf{T}} = \mathbf{b}_{\mathsf{att}}^{\mathsf{T}} - (\underline{\mathbf{a}}_{\mathsf{att}}^{\mathsf{T}} - \mathsf{bits}(1, \mathbf{X}_i) \otimes \underline{\mathbf{G}})$.

□

Thus, it suffices to show pseudorandomness of the following distribution given aux_2

$$\left(\begin{array}{c} \mathbf{B}, \ \{\mathbf{c}_{\mathbf{B},j} \leftarrow \mathbb{Z}_q^{mw}\}_{j\in[Q_{\mathsf{msg}}]}, \ \{\mathbf{X}_j = \mathbf{A}_{\mathsf{fhe},j}\mathbf{R}_j - (\mathbf{x}_j, \mathsf{sd}_j) \otimes \mathbf{G}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}_{\mathsf{att},j} \leftarrow \mathbb{Z}_q^{(\mathrm{L_X}+1)m}\}_{j\in[Q_{\mathsf{msg}}]}, \ \{\tilde{\mathsf{F}}_{j,k} = f_k(\mathbf{x}_j)\lfloor q/2 \rceil + \mathsf{PRF}(\mathsf{sd}_j, \mathbf{r}_k) + \mathbf{e}^{\mathsf{T}}_{\mathbf{P},j,k}\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]}. \end{array} \right)$$

where $\mathbf{A}_{\mathsf{fhe},j} \leftarrow \mathbb{Z}_q^{(n+1)\times m}$.

Hyb_3. This hybrid is same as Hyb_2 except that for $j \in [Q_{\mathsf{msg}}]$ we sample $\mathbf{X}_j \leftarrow \mathbb{Z}_q^{(n+1)\times m(\lambda+\mathrm{L})}$. We have $\mathsf{Hyb}_2 \approx_s \mathsf{Hyb}_3$ using leftover hash lemma. By leftover hash lemma (Lemma 3.4) we have that the statistical distance between $\mathbf{A}_{\mathsf{fhe},j}\mathbf{R}_j$ and a uniform matrix $U \leftarrow \mathbb{Z}_q^{(n+1)\times m(\lambda+\mathrm{L})}$ is $m(\lambda+\mathrm{L})/2^n$. This implies that the statistical distance between $\mathbf{X}_j = \mathbf{A}_{\mathsf{fhe},j}\mathbf{R}_j - (\mathbf{x}_j, \mathsf{sd}_j)\otimes\mathbf{G}$ and $\mathbf{X}_j \leftarrow \mathbb{Z}_q^{(n+1)\times m(\lambda+\mathrm{L})}$ is $m(\lambda+\mathrm{L})/2^n$ and we have

$$\Delta(\mathsf{Hyb}_2, \mathsf{Hyb}_3) \leq \frac{Q_{\mathsf{msg}} \cdot m(\lambda+\mathrm{L})}{2^n} \leq \frac{Q_{\mathsf{msg}} \cdot \mathrm{poly}(\lambda)}{2^\lambda}.$$

Thus, it suffices to show pseudorandomness of the following distribution given aux_2

$$\left(\begin{array}{c} \mathbf{B}, \ \{\mathbf{c}_{\mathbf{B},j} \leftarrow \mathbb{Z}_q^{mw}, \mathbf{X}_j \leftarrow \mathbb{Z}_q^{(n+1)\times m(\lambda+\mathrm{L})}, \mathbf{c}_{\mathsf{att},j} \leftarrow \mathbb{Z}_q^{(\mathrm{L_X}+1)m}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\tilde{\mathsf{F}}_{j,k} = f_k(\mathbf{x}_j)\lfloor q/2 \rceil + \mathsf{PRF}(\mathsf{sd}_j, \mathbf{r}_k) + \mathbf{e}^{\mathsf{T}}_{\mathbf{P},j,k}\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]}. \end{array} \right)$$

Hyb_4. This hybrid is the same as the previous one except that we replace $\mathsf{PRF}(\mathsf{sd}_j, \cdot)$ with the real random function $\mathsf{R}^j(\cdot)$ for each $j \in [q_{\mathsf{msg}}]$. Since sd_j is not used anywhere else, we can use the security of PRF to conclude that this hybrid is computationally indistinguishable from the previous one.

Hyb_5. This hybrid is same as the previous one except that we output a failure symbol if the set $\{\mathbf{r}_k\}_{k\in[Q_{\mathsf{key}}]}$, in aux_2, contains a collision. We prove that the probability with which there occurs a collision is negligible in λ. To prove this it suffices to show that there is no $k, k' \in [Q_{\mathsf{key}}]$ such that $k \neq k'$ and $\mathbf{r}_k = \mathbf{r}_{k'}$. The probability of this happening can be bounded by $Q^2_{\mathsf{key}}/2^\lambda$ by taking the union bound with respect to all the combinations of k, k'. Thus the probability of outputting the failure symbol is $Q^2_{\mathsf{key}}/2^\lambda$ which is $\mathsf{negl}(\lambda)$.

Hyb_6. In this hybrid we compute $\tilde{\mathsf{F}}_{j,k}$ as

$$\tilde{\mathsf{F}}_{j,k} = f_k(\mathbf{x}_j)\lfloor q/2 \rceil + R_{j,k} + \mathbf{e}^{\mathsf{T}}_{\mathbf{P},j,k}$$

for all $j \in [Q_{\mathsf{msg}}], k \in [Q_{\mathsf{key}}]$. Namely, we use fresh randomness $R_{j,k} \leftarrow [-q/4 + B, q/4 - B]^{1\times\ell}$ instead of deriving the randomness by $\mathsf{R}^j(\mathbf{r}_k)$. We claim that this change is only conceptual. To see this, we observe that unless the failure condition introduced in Hyb_5 is satisfied, every invocation of the function R^j is with respect to a fresh input and thus the output can be replaced with a fresh randomness.
Thus, it suffices to show pseudorandomness of the following distribution given aux_2

$$\left(\begin{array}{c} \mathbf{B}, \ \{\mathbf{c}_{\mathbf{B},j} \leftarrow \mathbb{Z}_q^{mw}, \mathbf{X}_j \leftarrow \mathbb{Z}_q^{(n+1)\times m(\lambda+\mathrm{L})}, \mathbf{c}_{\mathsf{att},j} \leftarrow \mathbb{Z}_q^{(\mathrm{L_X}+1)m}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\tilde{\mathsf{F}}_{j,k} = f_k(\mathbf{x}_j)\lfloor q/2 \rceil + R_{j,k} + \mathbf{e}^{\mathsf{T}}_{\mathbf{P},j,k}\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]}. \end{array} \right)$$

Hyb_7. This hybrid is same as the previous one except we sample $R_{j,k} \leftarrow [-q/4, q/4]^{1\times\ell}$. We note that $\mathsf{Hyb}_6 \approx_s \mathsf{Hyb}_7$. To see this note that the statistical distance between the uniform distributions $U_1 = [-q/4 + B, q/4 - B]$ and $U_2 = [-q/4, q/4]$ is

$$\Delta(U_1, U_2) = \frac{1}{2}\left|\frac{2}{q-4B} - \frac{2}{q}\right| \leq \frac{4B}{q} \leq \frac{\mathrm{poly}(\lambda)}{2^\lambda}$$

by our parameter setting. Therefore,

$$\Delta(\mathsf{Hyb}_2, \mathsf{Hyb}_3) \le \frac{Q_{\mathsf{key}} \cdot Q_{\mathsf{msg}} \cdot \mathrm{poly}(\lambda)}{2^\lambda}.$$

Hyb_8. This hybrid is same as the previous one except we sample $\tilde{\mathsf{F}}_{j,k} \leftarrow \mathbb{Z}_q^\ell$. This follows from the pseudorandomness of $\{f_k(x_j)\}_{j,k}$. To see this note that we have

$$(1^\lambda,\ \mathsf{aux}_{\mathcal{A}},\ \{f_k,\ f_k(x_j)\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]}) \approx_c (1^\lambda,\ \mathsf{aux}_{\mathcal{A}},\ \{f_k,\ \Delta_{j,k} \leftarrow \{0,1\}^\ell\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]})$$

which implies

$$\begin{aligned}(1^\lambda,\ \mathsf{aux}_{\mathcal{A}},\ \{f_k,\ \tilde{\mathsf{F}}_{j,k} = f_k(x_j)\ \lfloor q/2 \rceil + R_{j,k} + \mathbf{e}^{\mathsf{T}}_{\mathbf{P},j,k}\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]})\\ \approx_c (1^\lambda,\ \mathsf{aux}_{\mathcal{A}},\ \{f_k,\ \tilde{\mathsf{F}}_{j,k} \leftarrow \mathbb{Z}_q^\ell\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]})\end{aligned} \tag{23}$$

where $R_{j,k} \leftarrow [-q/4, q/4]^{1\times\ell}$ and $\mathbf{e}_{\mathbf{P},j,k} \leftarrow \mathcal{D}^\ell_{\mathbb{Z},\sigma_1}$.
Thus, using Equation (23) and noting that adding random strings does not make the task of distinguishing the two distributions any easier, we achieve the following distribution

$$\left(\begin{array}{c}\mathsf{aux}_{\mathcal{A}},\ \ \mathbf{B},\ \ \{\mathbf{c}_{\mathbf{B},j} \leftarrow \mathbb{Z}_q^{mw}, \mathbf{X}_j \leftarrow \mathbb{Z}_q^{(n+1)\times m(\lambda+\mathrm{L})}, \mathbf{c}_{\mathsf{att},j} \leftarrow \mathbb{Z}_q^{(\mathrm{L_X}+1)m}\}_{j\in[Q_{\mathsf{msg}}]},\\ \{\tilde{\mathsf{F}}_{j,k} \leftarrow \mathbb{Z}_q^\ell\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]}.\end{array}\right)$$

which is the R.H.S distribution of Equation (22), hence the proof.

□

5 KP-ABE for unbounded depth circuits using prFE

In this section, we provide a construction of kpABE for unbounded depth circuits.

5.1 Preparations

Here we define attribute based laconic function evaluation scheme and its properties.

Syntax. An attribute based laconic function evaluation (AB-LFE) scheme for a circuit class $\{\mathcal{C}_{\mathsf{prm}} = \{C : \mathcal{X}_{\mathsf{prm}} \to \{0,1\}\}\}_{\mathsf{prm}}$ for a parameter $\mathsf{prm} = \mathsf{prm}(\lambda)$ and a message space $\mathcal{M}$ consists of four algorithms $(\mathsf{crsGen}, \mathsf{Compress}, \mathsf{Enc}, \mathsf{Dec})$ defined as follows.

$\mathsf{crsGen}(1^\lambda, \mathsf{prm}) \to \mathsf{crs}$. The generation algorithm takes as input the security parameter 1^λ and circuit parameters prm and outputs a uniformly sampled common reference string crs.

$\mathsf{Compress}(\mathsf{crs}, C) \to \mathsf{digest}$. The compress algorithm takes as input the common random string crs and a circuit $C \in \mathcal{C}$ and outputs a digest digest.

$\mathsf{Enc}(\mathsf{crs}, \mathsf{digest}, (\mathbf{x}, \mu)) \to \mathsf{ct}$. The encryption algorithm takes as input the common random string crs, a digest digest, an attribute $\mathbf{x} \in \mathcal{X}_{\mathsf{prm}}$ and a message $\mu \in \mathcal{M}$ and outputs a ciphertext ct.

$\mathsf{Dec}(\mathsf{crs}, C, \mathsf{ct}) \to \mu/\perp$. The decryption algorithm takes as input the common random string crs, a circuit C, digest and a ciphertext ct and outputs a message $\mu \in \mathcal{M}$ or $\perp$.

Definition 5.1 (Correctness). An $\mathsf{AB\text{-}LFE}$ scheme for circuit family $\mathcal{C}_{\mathsf{prm}}$ is correct if for all prm, $C \in \mathcal{C}_{\mathsf{prm}}, \mathbf{x} \in \mathcal{X}_{\mathsf{prm}}$ such that $C(\mathbf{x}) = 1$, and for all messages $\mu \in \mathcal{M}$,

$$\Pr\left[\begin{array}{l}\mathsf{crs} \leftarrow \mathsf{crsGen}(1^\lambda, \mathsf{prm}),\\ \mathsf{digest} = \mathsf{Compress}(\mathsf{crs}, C),\\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{crs}, \mathsf{digest}, (\mathbf{x}, \mu)) :\\ \mathsf{Dec}(\mathsf{crs}, C, \mathsf{ct}) \neq \mu\end{array}\right] = \mathsf{negl}(\lambda)$$

where the probability is taken over the coins of Setup, KeyGen, and Enc.

Definition 5.2 (Pseudorandom Ciphertext Security). For a $\mathsf{AB\text{-}LFE}$ scheme and an adversary $\mathcal{A}$, we define the experiment for security $\mathsf{Expt}_{\beta,\mathcal{A}}^{\mathsf{AB\text{-}LFE}}(1^\lambda)$ as follows.

1. Run $\mathcal{A}$ to receive circuit parameters prm. Run $\mathsf{crs} \leftarrow \mathsf{crsGen}(1^\lambda, \mathsf{prm})$ and send crs to $\mathcal{A}$.
2. $\mathcal{A}$ chooses $C \in \mathcal{C}_{\mathsf{prm}}$, $\mathbf{x} \in \mathcal{X}_{\mathsf{prm}}$ and $\mu \in \mathcal{M}$. Run $\mathsf{digest} = \mathsf{Compress}(\mathsf{crs}, C)$, sample $\beta \leftarrow \{0,1\}$. If $\beta = 0$, it computes $\mathsf{ct}_0 \leftarrow \mathsf{Enc}(\mathsf{crs}, \mathsf{digest}, (\mathbf{x}, \mu))$ else if $\beta = 1$, it computes $\mathsf{ct}_1 \leftarrow \mathcal{CT}_{\mathsf{AB\text{-}LFE}}$, where $\mathcal{CT}_{\mathsf{AB\text{-}LFE}}$ is the ciphertext space of $\mathsf{AB\text{-}LFE}$. It sends $\mathsf{digest}, \mathsf{ct}_\beta$ to $\mathcal{A}$.
3. $\mathcal{A}$ outputs a guess bit β' as the output of the experiment.

We define the advantage $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{AB\text{-}LFE}}(\lambda)$ of $\mathcal{A}$ in the above game as

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{AB\text{-}LFE}}(\lambda) := \left|\Pr\left[\mathsf{Expt}_{0,\mathcal{A}}^{\mathsf{AB\text{-}LFE}}(1^\lambda) = 1\right] - \Pr\left[\mathsf{Expt}_{1,\mathcal{A}}^{\mathsf{AB\text{-}LFE}}(1^\lambda) = 1\right]\right|.$$

We say that a $\mathsf{AB\text{-}LFE}$ scheme is *adaptive* pseudorandom ciphertext secure if for every *admissible* PPT adversary $\mathcal{A}$, we have $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{AB\text{-}LFE}}(\lambda) \leq \mathsf{negl}(\lambda)$, where $\mathcal{A}$ is said to be admissible if $C(\mathbf{x}) = 0$.
The *selective* (resp. *very selective*) notion of the security requires the adversary $\mathcal{A}$ to choose $\mathbf{x}$ (resp. $\mathbf{x}, C$) along with prm before it receives crs.

Definition 5.3 (Decomposability). We say that a $\mathsf{AB\text{-}LFE}$ scheme for a circuit class $\{\mathcal{C}_{\mathsf{prm}} = \{C : \mathcal{X}_{\mathsf{prm}} \to \{0,1\}\}\}_{\mathsf{prm}}$ satisfies decomposability if for any $\mathsf{crs} \leftarrow \mathsf{crsGen}(1^\lambda, \mathsf{prm})$ and $\mathsf{digest} \leftarrow \mathsf{Compress}(\mathsf{crs}, C)$, we have $\mathsf{digest} = \{\mathsf{digest}_i\}_{i\in[q_C]}$ for some polynomial q_C, which may depend on C, and $\mathsf{size}(\mathsf{digest}_i) \leq \mathrm{poly}(\lambda)$. Furthermore, we have that $\mathsf{Enc}(\mathsf{crs}, \mathsf{digest}, (\mathbf{x}, \mu)) = \{\mathsf{Enc}_i(\mathsf{crs}, \mathsf{digest}_i, (\mathbf{x}, \mu))\}_{i\in[q_C]}$ where size of the encryption circuit $\mathsf{size}(\mathsf{Enc}_i(\cdot, \cdot, (\cdot, \cdot))) \leq \mathrm{poly}(\lambda, |\mathbf{x}|)$.

Remark 5.4. Here, we do not require the digest to be much smaller than the circuit description C, unlike the usual convention in the context of $\mathsf{AB\text{-}LFE}$. This relaxation allows us to instantiate $\mathsf{AB\text{-}LFE}$ using blind garbled circuits, which do not have compact digests.

5.2 Construction of kpABE with Unbounded Depth

Building Blocks. We require the following building blocks for our construction.

1. An attribute-based laconic function evaluation scheme $\mathsf{AB\text{-}LFE} = \mathsf{AB\text{-}LFE}.(\mathsf{crsGen}, \mathsf{Compress}, \mathsf{Enc}, \mathsf{Dec})$ for circuit class $\mathcal{C}_{\ell(\lambda)}$, consisting of circuits with input length $\ell(\lambda)$ and with unbounded depth and size. We let $\mathcal{CT}_{\mathsf{AB\text{-}LFE}}$ denote the ciphertext space of the scheme. We assume that the $\mathsf{AB\text{-}LFE}$ scheme is decomposable (Definition 5.3), i.e., we have $\mathsf{AB\text{-}LFE}.\mathsf{Enc} = \{\mathsf{AB\text{-}LFE}.\mathsf{Enc}_i\}_{i\in[q_C]}$ for some q_C and use $d_{\mathsf{AB\text{-}LFE}}$ to denote the maximum depth of a circuit required to compute $\{\mathsf{AB\text{-}LFE}.\mathsf{Enc}_i\}_{i\in[q_C]}$. We assume that the output length of $\mathsf{AB\text{-}LFE}.\mathsf{Enc}_i$ does not depend on i and denote its length by $\ell_{\mathsf{ct}}^{\mathsf{AB\text{-}LFE}}$.
2. A FE scheme for pseudorandom functionality $\mathsf{prFE} = (\mathsf{prFE}.\mathsf{Setup}, \mathsf{prFE}.\mathsf{KeyGen}, \mathsf{prFE}.\mathsf{Enc}, \mathsf{prFE}.\mathsf{Dec})$ for circuit class $\mathcal{C}_{\mathrm{L}(\lambda), d_{\mathsf{prFE}}(\lambda), \ell_{\mathsf{ct}}^{\mathsf{AB\text{-}LFE}}}$ consisting of circuits with input length $\mathrm{L}(\ell, \lambda)$, maximum depth $d_{\mathsf{prFE}}(\lambda)$ and output length $\ell_{\mathsf{ct}}^{\mathsf{AB\text{-}LFE}}$. We denote by prm the parameters $(1^{\mathrm{L}(\lambda)}, 1^{d_{\mathsf{prFE}}(\lambda)}, 1^{\ell_{\mathsf{ct}}^{\mathsf{AB\text{-}LFE}}})$ that specifies the function class being supported. We also denote the ciphertext space of the scheme by $\mathcal{CT}_{\mathsf{prFE}}$.

3. A PRF function PRF : $\{0,1\}^\lambda \times \{0,1\}^\lambda \rightarrow \{0,1\}^{\mathsf{R_{len}}}$ where $\mathsf{R_{len}}$ is the length of randomness used in $\mathsf{AB\text{-}LFE.Enc}$. We assume that PRF can be computed by a circuit of depth at most d_{prFE}.
4. A secret key encryption scheme $\mathsf{SKE} = (\mathsf{SKE.Setup}, \mathsf{SKE.Enc}, \mathsf{SKE.Dec})$ with the message space being $\{0,1\}^{\ell_{\mathsf{ct}}^{\mathsf{AB\text{-}LFE}}}$. We denote the ciphertext space of the scheme by $\mathcal{CT}_{\mathsf{SKE}}$ and the key space of the scheme by $\mathcal{K}_{\mathsf{SKE}}$. We assume that SKE has pseudorandom ciphertext as per Definition 3.14.

We assume that uniform sampling from $\mathcal{CT}_{\mathsf{SKE}}$ are possible without any parameter other than the security parameter.

Now, we describe our compiler for constructing a key-policy ABE scheme $\mathsf{kpABE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ for circuits of unbounded depth with attribute length $\ell(\lambda)$. We denote the ciphertext space of the scheme by $\mathcal{CT}_{\mathsf{kpABE}}$. For our construction, we have $\mathcal{CT}_{\mathsf{kpABE}} = \mathcal{CT}_{\mathsf{prFE}}$.

$\mathsf{Setup}(1^\lambda, 1^\ell) \rightarrow (\mathsf{mpk}, \mathsf{msk})$. The setup algorithm does the following.

- Run $(\mathsf{prFE.msk}, \mathsf{prFE.mpk}) \leftarrow \mathsf{prFE.Setup}(1^\lambda, \mathsf{prm})$ and $\mathsf{crs} \leftarrow \mathsf{AB\text{-}LFE.crsGen}(1^\lambda)$.
- Set $\mathsf{msk} = \mathsf{prFE.msk}$[8] and $\mathsf{mpk} = (\mathsf{prFE.mpk}, \mathsf{crs})$. Output $(\mathsf{msk}, \mathsf{mpk})$.

$\mathsf{KeyGen}(\mathsf{msk}, C) \rightarrow \mathsf{sk}_C$. The key generation algorithm does the following.

- Parse $\mathsf{msk} = \mathsf{prFE.msk}$.
- Compute $\mathsf{digest} = \mathsf{AB\text{-}LFE.Compress}(\mathsf{crs}, C)$. Parse $\mathsf{digest} = \{\mathsf{digest}_i\}_{i \in [q_C]}$.
- Sample $\mathbf{r} \leftarrow \{0,1\}^\lambda$ and $\mathsf{SKE.ct}_i \leftarrow \mathcal{CT}_{\mathsf{SKE}}$ for $i \in [q_C]$.
- For $i \in [q_C]$, define circuit $\mathrm{F}[\mathsf{crs}, \mathsf{digest}_i, \mathbf{r}, \mathsf{SKE.ct}_i]$, with $\mathsf{crs}, \mathsf{digest}_i, \mathbf{r}, \mathsf{SKE.ct}_i$ hardwired, as follows:

Circuit $\mathbf{F}[\mathsf{crs}, \mathsf{digest}_i, \mathbf{r}, \mathsf{SKE.ct}_i]$
Input: $(\mathsf{sd}, \mathbf{x}, \mu, \mathsf{flag}, \mathsf{SKE.sk})$

1. It computes $\mathsf{AB\text{-}LFE.ct}_i$ as follows:

$$\mathsf{AB\text{-}LFE.ct}_i = \begin{cases} \mathsf{AB\text{-}LFE.Enc}_i(\mathsf{crs}, \mathsf{digest}_i, (\mathbf{x}, \mu); \mathsf{PRF}(\mathsf{sd}, \mathbf{r})) & \text{if } \mathsf{flag} = 0 \\ \mathsf{SKE.Dec}(\mathsf{SKE.sk}, \mathsf{SKE.ct}_i) & \text{if } \mathsf{flag} = 1 \end{cases}$$

2. Output $\mathsf{AB\text{-}LFE.ct}_i$.

- For $i \in [q_C]$ compute $\mathsf{prFE.sk}_i \leftarrow \mathsf{prFE.KeyGen}(\mathsf{prFE.msk}, \mathrm{F}[\mathsf{crs}, \mathsf{digest}_i, \mathbf{r}])$.
- Output $\mathsf{sk}_C := \left(\{\mathsf{digest}_i,\ \mathsf{prFE.sk}_i, \mathsf{SKE.ct}_i\}_{i \in [q_C]}, \mathbf{r}\right)$.

$\mathsf{Enc}(\mathsf{mpk}, \mathbf{x}, \mu) \rightarrow \mathsf{ct}$. The encryption algorithm does the following.

- Parse $\mathsf{mpk} = (\mathsf{prFE.mpk}, \mathsf{crs})$ and sample a PRF key $\mathsf{sd} \leftarrow \{0,1\}^\lambda$.
- Compute $\mathsf{prFE.ct} \leftarrow \mathsf{prFE.Enc}(\mathsf{prFE.mpk}, (\mathsf{sd}, \mathbf{x}, \mu, 0, \perp))$.
- Output $\mathsf{ct} := \mathsf{prFE.ct}$.

$\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_C, C, \mathsf{ct}, \mathbf{x}) \rightarrow \mathbf{y}$. The decryption algorithm does the following.

- Parse $\mathsf{mpk} = (\mathsf{prFE.mpk}, \mathsf{crs})$, $\mathsf{sk}_C = \left(\{\mathsf{digest}_i,\ \mathsf{prFE.sk}_i, \mathsf{SKE.ct}_i\}_{i \in [q_C]}, \mathbf{r}\right)$ and $\mathsf{ct} = \mathsf{prFE.ct}$.
- For all $i \in [q_C]$, compute $y_i = \mathsf{prFE.Dec}(\mathsf{prFE.mpk}, \mathsf{prFE.sk}_i, \mathrm{F}[\mathsf{crs}, \mathsf{digest}_i, \mathbf{r}, \mathsf{SKE.ct}_i], \mathsf{prFE.ct})$.
- Set $\mathbf{y} = (y_1, \ldots, y_{q_C})$ and output $\mathsf{AB\text{-}LFE.Dec}(\mathsf{crs}, C, \mathbf{y})$.

[8]W.L.O.G we assume that msk contains mpk.

Correctness. We prove the correctness of our scheme using the following theorem.

Theorem 5.5. Assume $\mathsf{AB\text{-}LFE}$ and prFE schemes are correct. Then the above construction is correct.

Proof. From the correctness of prFE scheme we have

$$\begin{aligned} y_i =& \mathrm{F}[\mathsf{crs}, \mathsf{digest}_i, \mathbf{r}, \mathsf{SKE.ct}_i](\mathsf{sd}, \mathbf{x}, \mu, 0, \perp) \\ =& \mathsf{AB\text{-}LFE.ct}_i = \mathsf{AB\text{-}LFE.Enc}_i(\mathsf{crs}, \mathsf{digest}_i, (\mathbf{x}, \mu); \mathsf{PRF}(\mathsf{sd}, \mathbf{r})) \end{aligned}$$

for $i \in [q_C]$ with probability 1. Thus we have $\mathbf{y} = \{\mathsf{AB\text{-}LFE.ct}_i\}_{i\in[q_C]} = \mathsf{AB\text{-}LFE.ct}$. Next, by the correctness of $\mathsf{AB\text{-}LFE}$ scheme it follows that, if $C(\mathbf{x}) = 1$,

$$\mathsf{AB\text{-}LFE.Dec}(\mathsf{crs}, C, \mathbf{y}) = \mathsf{AB\text{-}LFE.Dec}(\mathsf{crs}, C, \mathsf{AB\text{-}LFE.ct}) = \mu$$

with all but negligible probability. □

Security. We prove the security of our scheme via the following theorem.

Theorem 5.6. Assume that the prFE scheme is IND secure (Definition 4.4), $\mathsf{AB\text{-}LFE}$ scheme satisfies very selective pseudorandom ciphertext security (Definition 5.2). Then our construction of kpABE scheme satisfies $\mathsf{VerSel\text{-}IND}$ security (Definition 3.22).

Proof. We prove the security via the following sequence of hybrids.

Hyb_0. This is the $\mathsf{VerSel\text{-}IND}$ game. Namely, the adversary sees

$$\left(\begin{array}{c} \mathsf{coins}_{\mathcal{A}},\ \mathsf{mpk} = (\mathsf{crs},\ \mathsf{prFE.mpk}),\ \mathsf{sk}_{C^k} = \{\mathsf{digest}_i^k, \mathbf{r}^k, \mathsf{SKE.ct}_i^k,\ \mathsf{prFE.sk}_i^k\}_{k\in[Q], i\in[q_{C_k}]} \\ \mathsf{prFE.ct} \leftarrow \mathsf{prFE.Enc}(\mathsf{prFE.mpk}, (\mathsf{sd}, \mathbf{x}, \mu_\beta, 0, \perp)) \end{array} \right)$$

where the adversary $\mathcal{A}$ with randomness $\mathsf{coins}_{\mathcal{A}}$ queries for $(\mathbf{x}, (\mu_0, \mu_1), C^1, \ldots, C^Q)$, $\mathsf{sk}_{C^k} = \{\mathsf{digest}_i^k, \mathbf{r}^k, \mathsf{SKE.ct}_i^k,$ $\mathsf{prFE.sk}_i^k\}_{i\in[q_{C^k}]}$ denotes the secret key corresponding to the k-th key query C^k as defined in the KeyGen algorithm, and β is the challenge bit chosen by the game. In particular, we choose $\mathsf{SKE.ct}_i^k \leftarrow \mathcal{CT}_{\mathsf{SKE}}$ for all k and i.

Hyb_1. This hybrid is the same as Hyb_0 except that $\mathsf{SKE.sk} \leftarrow \mathcal{K}_{\mathsf{SKE}}$ is chosen and $\mathsf{SKE.ct}_i^k$ is computed as

$$\mathsf{SKE.ct}_i^k = \mathsf{SKE.Enc}(\mathsf{SKE.sk}, \mathsf{AB\text{-}LFE.ct}_i^k) \tag{24}$$

where

$$\mathsf{AB\text{-}LFE.ct}_i^k = \mathsf{AB\text{-}LFE.Enc}_i(\mathsf{crs}, \mathsf{digest}_i^k, (\mathbf{x}, \mu_0); \mathsf{PRF}(\mathsf{sd}, \mathbf{r}^k)) \tag{25}$$

for $k \in [Q]$ and $i \in [q_{C^k}]$. By the pseudorandom ciphertext security of SKE, this hybrid is computationally indistinguishable from the previous one.

Hyb_2. This hybrid is the same as Hyb_1 except that $\mathsf{prFE.ct}$ is computed as

$$\mathsf{prFE.ct} \leftarrow \mathsf{prFE.Enc}(\mathsf{prFE.mpk}, (\perp, \perp, \perp, 1, \mathsf{SKE.sk})).$$

We claim that this game is computationally indistinguishable from the previous hybrid by the IND-pr-Security of prFE. To see this, we first observe that we have

$$\begin{aligned} \mathrm{F}[\mathsf{crs}, \mathsf{digest}_i^k, \mathbf{r}^k, \mathsf{SKE.ct}_i^k](\mathsf{sd}, \mathbf{x}, \mu_0, 0, \perp) &= \mathsf{AB\text{-}LFE.ct}_i^k \\ &= \mathsf{SKE.Dec}(\mathsf{SKE.sk}, \mathsf{SKE.ct}_i^k) \\ &= \mathrm{F}[\mathsf{crs}, \mathsf{digest}_i^k, \mathbf{r}^k, \mathsf{SKE.ct}_i^k](\perp, \perp, \perp, 1, \mathsf{SKE.sk}) \end{aligned}$$

for all $k \in [Q]$ and $i \in [q_{C^k}]$ by the correctness of prFE and SKE, where $\mathsf{AB\text{-}LFE.ct}_i^k$ is defined as Equation (25). As we will show later, we have

$$\begin{aligned}&\left(\mathsf{coins}_{\mathcal{A}}, \{\mathrm{F}[\mathsf{crs}, \mathsf{digest}_i^k, \mathbf{r}^k, \mathsf{SKE.ct}_i^k],\ \mathsf{AB\text{-}LFE.ct}_i^k\}_{k\in[Q],i\in[q_{C^k}]}\right)\\ \approx_c &\left(\mathsf{coins}_{\mathcal{A}}, \{\mathrm{F}[\mathsf{crs}, \mathsf{digest}_i^k, \mathbf{r}^k, \mathsf{SKE.ct}_i^k],\ \mathsf{AB\text{-}LFE.ct}_i^k \leftarrow \{0,1\}^{\ell_{\mathsf{ct}}^{\mathsf{AB\text{-}LFE}}}\}_{k\in[Q],i\in[q_{C^k}]}\right) \qquad (26)\end{aligned}$$

where $\mathsf{AB\text{-}LFE.ct}_i^k$ is defined as Equation (25) in the LHS and $\mathsf{SKE.ct}_i^k$ is defined as Equation (24) on both sides. Therefore, we can invoke the security of prFE.

Hyb_3. This hybrid is the same as Hyb_2 except that $\mathsf{AB\text{-}LFE.ct}_i^k$ is replaced with $\mathsf{AB\text{-}LFE.ct}_i^k \leftarrow \{0,1\}^{\ell_{\mathsf{ct}}^{\mathsf{AB\text{-}LFE}}}$. By Equation (26), this hybrid is computationally indistinguishable from the previous one. Notice that in this hybrid, the view of the adversary is independent from the challenge bit β.

The fact that Hyb_0 and Hyb_3 are computationally indistinguishable means that the adversary has negligible advantage in Hyb_0, since its advantage in Hyb_3 is 0. It remains to prove Equation (26). We prove Equation (26) via the following sequence of hybrids.

Hyb_0'. This is the LHS distribution of Equation (26) except that we give crs, $\{\mathsf{digest}_i^k, \mathbf{r}^k\}_{k,i}$ instead of $\{\mathrm{F}[\mathsf{crs}, \mathsf{digest}_i^k, \mathbf{r}^k, \mathsf{SKE.ct}_i^k]\}_{k,i}$ to the distinguisher:

$$\left(\mathsf{coins}_{\mathcal{A}}, \mathsf{crs}, \left\{\mathsf{digest}_i^k, \mathbf{r}^k, \mathsf{AB\text{-}LFE.ct}_i^k = \mathsf{AB\text{-}LFE.Enc}_i(\mathsf{crs}, \mathsf{digest}_i^k, (\mathbf{x}, \mu_\beta); \mathsf{PRF}(\mathsf{sd}, \mathbf{r}^k))\right\}_{k\in[Q],i\in[q_{C^k}]}\right).$$

We can rewrite the above distribution as

$$\left(\mathsf{coins}_{\mathcal{A}}, \mathsf{crs}, \left\{\mathsf{digest}^k, \mathbf{r}^k,\ \mathsf{AB\text{-}LFE.ct}^k = \mathsf{AB\text{-}LFE.Enc}(\mathsf{crs}, \mathsf{digest}^k, (\mathbf{x}, \mu_\beta); \mathsf{PRF}(\mathsf{sd}, \mathbf{r}^k))\right\}_{k\in[Q]}\right).$$

where we group the terms $\{\mathsf{digest}_i^k\}_{i\in[q_{C^k}]}$ and $\{\mathsf{AB\text{-}LFE.ct}_i^k\}_{i\in[q_{C^k}]}$ into digest^k and $\mathsf{AB\text{-}LFE.ct}^k$ for all $k \in [Q]$.

Hyb_1'. This hybrid is same as the previous one except that we output a failure symbol if the set $\{\mathbf{r}^k\}_{k\in[Q]}$ contains a collision. We prove that the probability with which there occurs a collision is negligible in λ. To prove this it suffices to show that there is no $k, k' \in [Q]$ such that $k \neq k'$ and $\mathbf{r}^k = \mathbf{r}^{k'}$. The probability of this happening can be bounded by $Q^2/2^\lambda$ by taking the union bound with respect to all the combinations of k, k'. Thus the probability of outputting the failure symbol is $Q^2/2^\lambda$ which is $\mathsf{negl}(\lambda)$.

Hyb_2'. In this hybrid we change all the PRF values computed using sd to random. Namely, we replace $\mathsf{PRF}(\mathsf{sd}, \mathbf{r}^k)$ with true randomness. Since PRF is invoked for fresh input for each $k \in [Q]$, this hybrid is indistinguishable from the previous hybrid. We now consider the following distribution:

$$\left(\mathsf{coins}_{\mathcal{A}}, \mathsf{crs}, \{\mathsf{digest}^k,\ \mathbf{r}^k,\ \mathsf{AB\text{-}LFE.ct}^k \leftarrow \mathsf{AB\text{-}LFE.Enc}(\mathsf{crs}, \mathsf{digest}^k, (\mathbf{x}, \mu_\beta))\}_{k\in[Q]}\right)$$

Hyb_3'. In this hybrid we invoke the security of $\mathsf{AB\text{-}LFE}$ scheme to switch $\mathsf{AB\text{-}LFE.ct}^k$ to random for all $k \in [Q]$. Namely, the distribution is now:

$$\left(\mathsf{coins}_{\mathcal{A}}, \mathsf{crs}, \{\mathsf{digest}^k,\ \mathbf{r}^k,\ \mathsf{AB\text{-}LFE.ct}^k \leftarrow \mathcal{CT}_{\mathsf{AB\text{-}LFE}}\}_{k\in[Q]}\right)$$

By the admissibility of the adversary and very selective pseudorandom ciphertext security of $\mathsf{AB\text{-}LFE}$, this hybrid is indistinguishable from the previous one. Notice that this hybrid is the same as the RHS distribution of Equation (26) except that we give crs, $\{\mathsf{digest}_i^k, \mathbf{r}^k\}_{k,i}$ instead of $\{\mathrm{F}[\mathsf{crs}, \mathsf{digest}_i^k, \mathbf{r}^k, \mathsf{SKE.ct}_i^k]\}_{k,i}$.

We therefore have $\mathsf{Hyb}_0' \approx_c \mathsf{Hyb}_3'$, which implies Equation (26), since $\mathsf{SKE.ct}_i^k$ can be simulated by sampling $\mathsf{SKE.sk}$ and encrypting $\mathsf{AB\text{-}LFE.ct}_i^k$. This completes the proof. □

5.3 $\mathsf{AB\text{-}LFE}$ from Blind Garbled circuits.

In this section we give a construction of $\mathsf{AB\text{-}LFE} = (\mathsf{crsGen}, \mathsf{Compress}, \mathsf{Enc}, \mathsf{Dec})$ (or $\mathsf{1ABE}$) for a circuit class $\mathcal{C} = \{C : \{0,1\}^L \to \{0,1\}\}$ and message space $\{0,1\}$ from blind garbled circuits. Our construction supports circuits of unbounded depth and input length. Our construction can equivalently be seen as constructing a $\mathsf{1ABE}$ scheme.

Building Blocks. A blind garbled circuit scheme $\mathsf{bGC} = (\mathsf{bGC.Eval}, \mathsf{bGC.Garble}, \mathsf{bGC.SIM})$ for circuit class $\mathcal{C}$.

Construction. We describe our construction for AB-LFE scheme $\mathsf{AB\text{-}LFE} = (\mathsf{crsGen}, \mathsf{Compress}, \mathsf{Enc}, \mathsf{Dec})$ below.

$\mathsf{crsGen}(1^\lambda) \to \mathsf{crs}$. The crs generation algorithm outputs $\mathsf{crs} := \perp$.

$\mathsf{Compress}(\mathsf{crs}, C) \to \mathsf{digest}$. The Compress algorithm outputs $\mathsf{digest} = C$.

$\mathsf{Enc}(\mathsf{crs}, \mathsf{digest}, \mathbf{x}, \mu) \to \mathsf{ct}$. The encryption algorithm does the following.

- Parse $\mathsf{digest} = C$, sample $R \leftarrow \{0,1\}$ and define circuit $C[R]$, with R hardwired, as follows
$$C[R](\mathbf{x}, \mu) = \begin{cases} \mu \text{ if } C(\mathbf{x}) = 1, \\ R \text{ if } C(\mathbf{x}) = 0. \end{cases} .$$
- Compute $(\{\mathsf{lab}_{j,b}\}_{j\in[L+1], b\in\{0,1\}}, \widetilde{C[R]}) \leftarrow \mathsf{bGC.Garble}(1^\lambda, 1^{L+1}, C[R], 1^1)$.
- Output $\mathsf{ct} = \left(\widetilde{C[R]}, \mathsf{lab}_{\mathbf{x},\mu}\right)$ where $\mathsf{lab}_{\mathbf{x},\mu} = (\mathsf{lab}_{1,x_1}, \ldots, \mathsf{lab}_{L,x_L}, \mathsf{lab}_{L+1,\mu})$.

$\mathsf{Dec}(\mathsf{crs}, C, \mathsf{ct}) \to \mu/\perp$. The decryption algorithm does the following.

- Parse $\mathsf{ct} = \left(\widetilde{C[R]}, \mathsf{lab}_{\mathbf{x},\mu}\right)$.
- Output $\mu' = \mathsf{bGC.Eval}(\widetilde{C[R]}, \mathsf{lab}_{\mathbf{x},\mu})$.

Correctness. The correctness of the scheme follows directly from the correctness of the underlying bGC scheme. We prove it using the following theorem.

Theorem 5.7. Assume that the blind bGC is correct (Definition 3.15). Then the $\mathsf{AB\text{-}LFE}$ scheme is correct (Definition 5.1).

Proof. For $\mathsf{ct} = \left(\widetilde{C[R]}, \mathsf{lab}_{\mathbf{x},\mu}\right)$ where $\mathsf{lab}_{\mathbf{x},\mu} = (\mathsf{lab}_{1,x_1}, \ldots, \mathsf{lab}_{L,x_L}, \mathsf{lab}_{L+1,\mu})$, we have

$$\mathsf{bGC.Eval}(\widetilde{C[R]}, \mathsf{lab}_{\mathbf{x},\mu}) = \mu$$

if $C(\mathbf{x}) = 1$ from the correctness of bGC scheme with probability 1. This implies the correctness. □

Efficiency. The $\mathsf{AB\text{-}LFE}$ scheme has the following efficiency properties.

$$|\mathsf{digest}| = |C|, \; |\mathsf{ct}| = O(|C|, L, \lambda).$$

Decomposability. The decomposability follows from the decomposability of the blind garbled circuits. In particular we can parse $\mathsf{digest} = \{\mathsf{digest}_i\}_{i\in[|C|]}$, where $\mathsf{digest}_i = C_i$ for $i \in [|C|]$ and C_i denotes the i-th gate of C in topological order. We can also parse $\mathsf{ct} = \{\mathsf{ct}_i\}_{i\in[|C|]}$ where we can set $\mathsf{ct}_1 = (\widetilde{C_1[R]}, \mathsf{lab}_{\mathbf{x},\mu})$ and $\mathsf{ct}_i = \widetilde{C_i[R]}$ for $i \in [2, |C|]$ where $C_i[R]$ denotes the i-th gate of $C_i[R]$ and $\widetilde{C_i[R]}$ is the corresponding garbling. Note that here $|\mathsf{digest}_i| = \mathrm{poly}(\lambda)$ and $|\mathsf{ct}_i| \le \mathrm{poly}(\lambda, L)$.

Security. The security of the scheme follows from the simulation security and the blindness of the underlying bGC scheme. We prove this using the following theorem.

Theorem 5.8. Assume that the bGC scheme satisfies simulation security (Definition 3.16) and blindness (Definition 3.17). Then the $\mathsf{AB\text{-}LFE}$ scheme satisfies adaptive pseudorandom ciphertext security (Definition 5.2).

Proof. Suppose the adversary after receiving $\mathsf{crs} = \perp$ from the challenger outputs the challenge circuit C and the challenge inputs $(\mathbf{x}, \mu)$. To prove the security of the $\mathsf{AB\text{-}LFE}$ scheme, we consider the following sequence of hybrids.

Hyb_0. This hybrid corresponds to the real-world game where the ciphertext is computed honestly using the Enc algorithm.

Hyb_1. This hybrid is same as the previous hybrid except that the challenger computes $\left(\widetilde{C}, \widetilde{\mathsf{lab}}\right) \leftarrow \mathsf{bGC.SIM}(1^\lambda, 1^{|C[R]|}, 1^{L+1}, R)$ where $R \leftarrow \{0,1\}$.
Noting that $C[R](\mathbf{x}, \mu) = R$ by the admissibility of the adversary, $\mathsf{Hyb}_0 \approx_c \mathsf{Hyb}_1$ follows by the simulation security of the bGC scheme.

Hyb_2. This hybrid is same as the previous hybrid except that the challenger samples uniformly random string $(\widetilde{C}, \widetilde{\mathsf{lab}})$ such that $|(\widetilde{C}, \widetilde{\mathsf{lab}})| = |\mathsf{bGC.SIM}(1^\lambda, 1^{L+1}, 1^{|C[R]|}, R)|$ and returns digest $= C$, $\mathsf{ct} = (\widetilde{C}, \widetilde{\mathsf{lab}})$ to the adversary. $\mathsf{Hyb}_1 \approx_c \mathsf{Hyb}_2$ using the blindness of the bGC scheme.

Note that in Hyb_2, the adversary is given a random string. Therefore, the adaptive pseudorandom ciphertext security follows. □

We get the following theorem.

Theorem 5.9. Assume that one-way function exists. Then, there exists a $\mathsf{AB\text{-}LFE}$ scheme for the circuit class $\mathcal{C} = \{C : \{0,1\}^L \to \{0,1\}\}$ and message space $\{0,1\}$ with $|\mathsf{digest}| = |C|$, $|\mathsf{ct}| = O(|C|, L, \lambda)$.

Instantiating the $\mathsf{AB\text{-}LFE}$ scheme as above, and using a prFE scheme supporting $d_{\mathsf{prFE}} = \mathrm{poly}(\lambda)$ depth circuits with input length $\mathrm{L} = \ell + \mathrm{poly}(\lambda)$, we obtain the following theorem.

Theorem 5.10. Assuming LWE and IND secure prFE (Definition 4.4), there exists a very selectively secure kpABE scheme for circuits of unbounded depth and attribute length ℓ with

$$|\mathsf{mpk}| = \ell \cdot \mathrm{poly}(\lambda), \quad |\mathsf{sk}_C| = |C| \cdot \ell \cdot \mathrm{poly}(\lambda), \quad |\mathsf{ct}| = \ell \cdot \mathrm{poly}(\lambda).$$

We note that the kpABE scheme instantiated as above has longer secret keys but is not based on any circular assumptions.

Equivalence of $\mathsf{1ABE}$ and $\mathsf{AB\text{-}LFE}$ from Blind Garbled Circuits. We show that the $\mathsf{AB\text{-}LFE}$ scheme instantiated from blind garbled circuits is in fact equivalent to the instantiation of $\mathsf{1ABE}$ using BGC– which we bootstrap to a full fledged KP-ABE using prFE in the technical overview (Section 2).

First, we roughly outline the instantiation of $\mathsf{1ABE}$ using BGC : 1) Setup: set $\mathsf{msk} = R_{\mathsf{bGC}}$, where R_{bGC} is the randomness required to compute bGC components. 2) Encrypt: To encrypt $(\mathbf{x}, \mu)$ using $\mathsf{msk} = R_{\mathsf{bGC}}$, we simply generate bGC labels corresponding to $(\mathbf{x}, \mu)$ using randomness R_{bGC}. 3) Keygen : To generate a key for circuit C using msk we first sample some randomness $R \leftarrow \{0,1\}$ and generate bGC garbled circuit corresponding to $C[R]$, where $C[R]$ is defined exactly as in the above construction, using randomness R_{bGC}. This secret-key $\mathsf{1ABE}$ is (1-key,1-ct) secure and has *random keys* and *random ciphertexts*.

Next, we note that this $\mathsf{1ABE}$ is equivalent to $\mathsf{AB\text{-}LFE}$ from bGC except for minor syntactical differences. Both primitives essentially generate the components of a bGC scheme, the only difference being that $\mathsf{1ABE}$ generates the garbed circuit and garbled labels in KeyGen and Enc separately using the same randomness, while $\mathsf{AB\text{-}LFE}$ generates both in $\mathsf{Enc}(\mathsf{crs}, \mathsf{digest}, (\mathbf{x}, \mu))$ since digest $= C$ is provided as input. So, when bootstrapping a $\mathsf{1ABE}$ to kpABE, we let the prFE output both $\mathsf{1ABE.sk}$ and $\mathsf{1ABE.ct}$ as described in technical overview. This is the only difference from the construction of kpABE using $\mathsf{AB\text{-}LFE}$.

5.4 Using the $\mathsf{AB\text{-}LFE}$ from HLL

We can directly instantiate the $\mathsf{AB\text{-}LFE}$ in Section 5.2 with the construction shown by [HLL23] (HLL henceforth). For this instantiation, we do not assume decomposability of $\mathsf{AB\text{-}LFE}$ since the HLL construction has succinct digest, i.e. $q_C = 1$ for any C and $|\mathsf{digest}| = \mathrm{poly}(\lambda)$. This instantiation leads to the parameter size of $|\mathsf{mpk}| = \mathrm{poly}(\ell, \lambda)$, $|\mathsf{sk}_C| = \mathrm{poly}(\ell, \lambda)$, and $|\mathsf{ct}| = \mathrm{poly}(\ell, \lambda)$, which is already better than the previous instantiation.

By adding a twist to the construction in Section 5.2 exploiting the structural property of HLL, we can improve the secret key size so that its dependency on ℓ can be removed. To do so, we exploit the online-offline structure of the $\mathsf{AB\text{-}LFE.Enc}$ algorithm which can be split as $\mathsf{AB\text{-}LFE.Enc} = (\mathsf{AB\text{-}LFE.EncX}, \mathsf{AB\text{-}LFE.EncD})$. Here, $\mathsf{AB\text{-}LFE.EncX}$ takes as input crs and $\mathbf{x}$ and outputs the offline part of the ciphertext $\mathsf{AB\text{-}LFE.ct_{off}}$ and short state st and $\mathsf{AB\text{-}LFE.EncD}$ takes as input digest and st and outputs online part of the ciphertext $\mathsf{AB\text{-}LFE.ct_{on}}$. For the $\mathsf{AB\text{-}LFE}$ with online-offline structure, we consider the following security notion.

Definition 5.11 (Online-Offline Pseudorandom Ciphertext Security). For a $\mathsf{AB\text{-}LFE}$ scheme with the online-offline structure and an adversary $\mathcal{A}$, we define the experiment for security $\mathsf{Expt}_{\beta,\mathcal{A}}^{\mathsf{AB\text{-}LFE_{on-off}}}(1^\lambda)$ as follows.

1. Run $\mathcal{A}$ to receive circuit parameters prm. Run $\mathsf{crs} \leftarrow \mathsf{crsGen}(1^\lambda, \mathsf{prm})$ and send crs to $\mathcal{A}$.
2. $\mathcal{A}$ chooses $C^1, \ldots, C^Q \in \mathcal{C}_{\mathsf{prm}}$, $\mathbf{x} \in \mathcal{X}_{\mathsf{prm}}$ and $\mu \in \mathcal{M}$. Run $(\mathsf{AB\text{-}LFE.ct_{off}}, \mathsf{st}) \leftarrow \mathsf{AB\text{-}LFE.EncX}(\mathsf{crs}, (\mathbf{x}, \mu))$ and sample $\beta \leftarrow \{0,1\}$. It then computes $\mathsf{AB\text{-}LFE.ct}_{\mathsf{on}}^k$ for $k \in [Q]$ as
$$\mathsf{AB\text{-}LFE.ct}_{\mathsf{on}}^k \leftarrow \begin{cases} \mathsf{Enc}(\mathsf{st}, \mathsf{digest}^k, (\mathbf{x}, \mu)) & \text{if } \beta = 0 \\ \mathcal{CT}_{\mathsf{AB\text{-}LFE}} & \text{if } \beta = 1 \end{cases}, \quad \text{where} \quad \mathsf{digest}^k = \mathsf{AB\text{-}LFE.Compress}(\mathsf{crs}, C^k)$$
where $\mathcal{CT}_{\mathsf{AB\text{-}LFE}}$ is the ciphertext space of $\mathsf{AB\text{-}LFE}$. It sends $\mathsf{AB\text{-}LFE.ct_{off}}, \{\mathsf{digest}^k, \mathsf{AB\text{-}LFE.ct}_{\mathsf{on}}^k\}_{k\in[Q]}$ to $\mathcal{A}$.
3. $\mathcal{A}$ outputs a guess bit β' as the output of the experiment.

We define the advantage $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{AB\text{-}LFE_{on-off}}}(\lambda)$ of $\mathcal{A}$ in the above game as
$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{AB\text{-}LFE_{on-off}}}(\lambda) := \left|\Pr\left[\mathsf{Expt}_{0,\mathcal{A}}^{\mathsf{AB\text{-}LFE_{on-off}}}(1^\lambda) = 1\right] - \Pr\left[\mathsf{Expt}_{1,\mathcal{A}}^{\mathsf{AB\text{-}LFE_{on-off}}}(1^\lambda) = 1\right]\right|.$$

We say that a $\mathsf{AB\text{-}LFE}$ scheme is very selective pseudorandom ciphertext secure if for every *admissible* PPT adversary $\mathcal{A}$, we have $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{AB\text{-}LFE_{on-off}}}(\lambda) \leq \mathsf{negl}(\lambda)$, where $\mathcal{A}$ is said to be admissible if $C^k(\mathbf{x}) = 0$ for all $k \in [Q]$.

Formally, HLL proved the following theorem:

Theorem 5.12 ([HLL23]). Under the circular LWE assumption, there exists a very selectively secure $\mathsf{AB\text{-}LFE}$ scheme for circuit class $\mathcal{C} = \{C : \{0,1\}^\ell \to \{0,1\}\}$ with online-offline pseudorandom ciphertext security (as per Definition 5.11) satisfying
$$|\mathsf{crs}| = O(\ell, \lambda),\ |\mathsf{digest}| = O(\lambda),\ |\mathsf{st}| = O(\lambda),\ |\mathsf{ct_{off}}| = O(\ell, \lambda),\ |\mathsf{ct_{on}}| = O(\lambda)$$

We make slight modifications to our construction of kpABE scheme to optimize the secret key size. The high level idea is very simple. Instead of letting the prFE decryption recover the entire $\mathsf{AB\text{-}LFE}$ ciphertext, we recover only the online part of it. We then put the offline part of $\mathsf{AB\text{-}LFE}$ ciphertext into the ciphertext of kpABE so that the decryptor can recover the entire $\mathsf{AB\text{-}LFE}$ ciphertext during the decryption. This eliminates the necessity of hardwiring crs to the prFE secret key, since the online part can be computed only from the short state and digest. This leads to the improvement on the efficiency, since the state and digest are of fixed polynomial size, while crs is of size $O(\ell)$. Concretely, we modify the $\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}$ algorithm of Section 5.2 as follows.

$\mathsf{KeyGen}(\mathsf{msk}, C)$. This is same as the KeyGen algorithm in Section 5.2 except the following.

- Define circuit $\mathrm{F}[\mathsf{digest}, \mathbf{r}, \mathsf{SKE.ct}]$ (instead of $\mathrm{F}[\mathsf{crs}, \mathsf{digest}_i, \mathbf{r}, \mathsf{SKE.ct}]$) , with $\mathsf{digest}, \mathbf{r}$ hardwired, as follows On input $(\mathsf{sd}, \mathsf{st}, \mathsf{flag}, \mathsf{SKE.sk})$:

- Compute $\mathsf{AB\text{-}LFE.ct_{on}} := \begin{cases} \mathsf{AB\text{-}LFE.EncD}(\mathsf{st}, \mathsf{digest}; \mathsf{PRF}(\mathsf{sd}, \mathbf{r})) & \text{if flag} = 0 \\ \mathsf{SKE.Dec}(\mathsf{SKE.sk}, \mathsf{SKE.ct}) & \text{if flag} = 1 \end{cases}$.
- Output $\mathsf{AB\text{-}LFE.ct_{on}}$.

- Compute $\mathsf{prFE.sk} \leftarrow \mathsf{prFE.KeyGen}(\mathsf{prFE.msk}, \mathrm{F}[\mathsf{digest}, \mathbf{r}, \mathsf{SKE.ct}])$.
- Output $\mathsf{sk}_C = (\mathsf{digest}, \mathsf{prFE.sk}, \mathbf{r}, \mathsf{SKE.ct})$.

$\mathsf{Enc}(\mathsf{mpk}, \mathbf{x}, \mu)$. The encryption algorithm does the following.

- Parse $\mathsf{mpk} = (\mathsf{prFE.mpk}, \mathsf{crs})$ and sample a PRF key $\mathsf{sd} \leftarrow \{0,1\}^\lambda$.
- Compute $(\mathsf{AB\text{-}LFE.ct_{off}}, \mathsf{st}) \leftarrow \mathsf{AB\text{-}LFE.EncX}(\mathsf{crs}, (\mathbf{x}, \mu))$.
- Compute $\mathsf{prFE.ct} \leftarrow \mathsf{prFE.Enc}(\mathsf{prFE.mpk}, (\mathsf{sd}, \mathsf{st}))$.
- Output $\mathsf{ct} := (\mathsf{AB\text{-}LFE.ct_{off}}, \mathsf{prFE.ct})$.

$\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_C, C, \mathsf{ct}, \mathbf{x})$. The decryption algorithm does the following.

- Parse $\mathsf{mpk} = (\mathsf{prFE.mpk}, \mathsf{crs})$, $\mathsf{sk}_C = (\mathsf{digest}, \mathsf{prFE.sk}, \mathbf{r}, \mathsf{SKE.ct})$ and $\mathsf{ct} = (\mathsf{AB\text{-}LFE.ct_{off}}, \mathsf{prFE.ct})$.
- Compute $\mathsf{AB\text{-}LFE.ct_{on}} = \mathsf{prFE.Dec}(\mathsf{prFE.mpk}, \mathsf{prFE.sk}, \mathrm{F}[\mathsf{digest}, \mathbf{r}, \mathsf{SKE.ct}], \mathsf{prFE.ct})$.
- Set $\mathbf{y} = (\mathsf{AB\text{-}LFE.ct_{off}}, \mathsf{AB\text{-}LFE.ct_{on}})$ and output $\mathsf{AB\text{-}LFE.Dec}(\mathsf{crs}, C, \mathbf{y})$.

We note that even with the above changes the correctness and security arguments are almost the same as that of Section 5.2. We skip the proof for correctness since it is straightforward and focus on the security proof, where we highlight the difference from that for Theorem 5.6. We consider similar sequence of hybrids Hyb_0 to Hyb_3 and Hyb'_0 to Hyb'_3, where the latter hybrids are introduced to show that the online parts of the AB-LFE ciphertexts obtained by prFE decryption are pseudorandom. The main difference in the former hybrids is that $\mathsf{SKE.ct}^k$ encrypts the online part of the AB-LFE ciphertext $\mathsf{AB\text{-}LFE.ct}^k_{\mathsf{on}}$ instead of the entire ciphertext $\mathsf{AB\text{-}LFE.ct}^k$. For proving $\mathsf{Hyb}'_0 \approx_c \mathsf{Hyb}'_3$, we use online-offline pseudorandom ciphertext security of AB-LFE (as per Definition 5.11).

For this instantiation, we have $d^{\mathsf{EncD}}_{\mathsf{AB\text{-}LFE}} = \mathrm{poly}(\lambda)$ where $d^{\mathsf{EncD}}_{\mathsf{AB\text{-}LFE}}$ is the maximum depth of a circuit required to compute AB-LFE.EncD and hence we use a prFE scheme supporting $d_{\mathsf{prFE}} = \mathrm{poly}(\lambda)$ depth circuits with input length $\mathrm{L} = \mathrm{poly}(\lambda)$. We formalise this using the following theorem.

Theorem 5.13. Assuming the circular small-secret LWE and IND secure prFE (Definition 4.4), there exists a very selectively secure kpABE scheme for circuits of unbounded depth and attribute length ℓ with

$$|\mathsf{mpk}| = \mathrm{poly}(\ell, \lambda), \quad |\mathsf{sk}_C| = \mathrm{poly}(\lambda), \quad |\mathsf{ct}| = \mathrm{poly}(\ell, \lambda).$$

We note that the kpABE scheme instantiated as above has succinct keys and ciphertexts. Our scheme achieves the same parameters as the unbounded depth KP-ABE scheme by [HLL23] but does not make use of the circular evasive LWE assumption as they do.

6 Compiling KP-ABE to CP-ABE using prFE

In this section, we give a compiler that upgrades a single-key secure kpABE scheme for circuits of unbounded depth to a collusion resistant cpABE scheme for circuits of unbounded depth using a prFE scheme with IND security.

6.1 Construction

Building Blocks. We require the following building blocks for our construction.

1. A single-key secure key-policy ABE scheme $\mathsf{kpABE} = (\mathsf{kpABE.Setup}, \mathsf{kpABE.KeyGen}, \mathsf{kpABE.Enc}, \mathsf{kpABE.Dec})$ for circuit class $\mathcal{C}_{\ell(\lambda)} = \{C : \{0,1\}^\ell \to \{0,1\}\}$ consisting of circuits with input length $\ell(\lambda)$ and unbounded depth. We denote the ciphertext space of kpABE scheme by $\mathcal{CT}_{\mathsf{kpABE}} := \{0,1\}^{\ell_{\mathsf{ct}}^{\mathsf{kpABE}}}$, ciphertext size by $\ell_{\mathsf{ct}}^{\mathsf{kpABE}}$ and master public key size by $\ell_{\mathsf{mpk}}^{\mathsf{kpABE}}$. We note that the length $\ell_{\mathsf{ct}}^{\mathsf{kpABE}}$ of the ciphertext only depends on ℓ and λ. We also require that the the depth $d_{\mathsf{key}}^{\mathsf{kpABE}}$ of the setup algorithm represented as a circuit and the depth $d_{\mathsf{ct}}^{\mathsf{kpABE}}$ of the encryption algorithm represented as a circuit are of fixed polynomial $\mathrm{poly}(\lambda)$ independent of ℓ.

2. A FE scheme for pseudorandom functionality $\mathsf{prFE} = (\mathsf{prFE.Setup}, \mathsf{prFE.KeyGen}, \mathsf{prFE.Enc}, \mathsf{prFE.Dec})$ for circuit class $\mathcal{C}_{\mathrm{L}(\lambda), d_{\mathsf{prFE}}(\lambda), \ell_{\mathsf{ct}}^{\mathsf{kpABE}}}$ consisting of circuits with input length $\mathrm{L}(\lambda)$, maximum depth $d_{\mathsf{prFE}}(\lambda)$ and output length $\ell_{\mathsf{ct}}^{\mathsf{kpABE}}$. For our compiler, we should set d_{prFE} large enough so that the scheme can support a circuit of the form $\mathrm{F}[\mathbf{x}, \mathbf{r}, \mathsf{SKE.ct}]$ defined below. The depth of the circuit can be bounded by a fixed polynomial as we discuss below. Therefore, we set $d_{\mathsf{prFE}} = \mathrm{poly}(\lambda)$, where $\mathrm{poly}(\lambda)$ is a sufficiently large fixed polynomial.

3. A PRF function $\mathsf{PRF} : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^{\mathsf{R_{len}}}$ where $\mathsf{R_{len}}$ is the length of randomness used in $\mathsf{kpABE.Enc}$. We assume that PRF can be computed by a circuit of depth at most d_{prFE}.

4. A secret key encryption scheme $\mathsf{SKE} = (\mathsf{SKE.Setup}, \mathsf{SKE.Enc}, \mathsf{SKE.Dec})$ with the message space being $\{0,1\}^{\ell_{\mathsf{ct}}^{\mathsf{kpABE}}}$. We denote the ciphertext space of the scheme by $\mathcal{CT}_{\mathsf{SKE}}$ and the key space of the scheme by $\mathcal{K}_{\mathsf{SKE}}$. We assume that SKE has pseudorandom ciphertext as per Definition 3.14.

Now, we describe our compiler for constructing a ciphertext-policy ABE scheme $\mathsf{cpABE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ for circuits of unbounded depth with attribute length ℓ.

$\mathsf{Setup}(1^\lambda, 1^\ell) \to (\mathsf{cpABE.mpk}, \mathsf{cpABE.msk})$. The setup algorithm does the following.

- Run $(\mathsf{prFE.msk}, \mathsf{prFE.mpk}) \leftarrow \mathsf{prFE.Setup}(1^\lambda, 1^{\mathrm{L}})$.
- Set $\mathsf{cpABE.mpk} = \mathsf{prFE.mpk}$ and $\mathsf{cpABE.msk} = \mathsf{prFE.msk}$. Output $(\mathsf{cpABE.mpk}, \mathsf{cpABE.msk})$.

$\mathsf{KeyGen}(\mathsf{cpABE.msk}, \mathbf{x}) \to \mathsf{cpABE.sk}_{\mathbf{x}}$. The key generation algorithm does the following.

- Parse $\mathsf{cpABE.msk} = \mathsf{prFE.msk}$ and sample $\mathbf{r} \leftarrow \{0,1\}^\lambda$ and $\mathsf{SKE.ct} \leftarrow \mathcal{CT}_{\mathsf{SKE}}$.
- Define circuit $\mathrm{F}[\mathbf{x}, \mathbf{r}, \mathsf{SKE.ct}]$, with $\mathbf{x}, \mathbf{r}$ and $\mathsf{SKE.ct}$ hardwired, as follows.

  Circuit F$[\mathbf{x}, \mathbf{r}, \mathsf{SKE.ct}]$
  Input: $(R_{\mathsf{key}}, \mathsf{sd}, \mu, \mathsf{flag}, \mathsf{SKE.sk})$

  1. It computes $(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk}) = \mathsf{kpABE.Setup}(1^\lambda, 1^\ell; R_{\mathsf{key}})$.
  2. It computes kpABE.ct as follows:

  $$\mathsf{kpABE.ct} = \begin{cases} \mathsf{kpABE.ct} = \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathbf{x}, \mu; \mathsf{PRF}(\mathsf{sd}, \mathbf{r})) & \text{if } \mathsf{flag} = 0 \\ \mathsf{SKE.Dec}(\mathsf{SKE.sk}, \mathsf{SKE.ct}) & \text{if } \mathsf{flag} = 1 \end{cases}$$

  3. Output kpABE.ct.

  Note that since the depths of the setup algorithm kpABE.Setup and the encryption algorithm kpABE.Enc represented as circuits are bounded by $\mathrm{poly}(\lambda)$ by our assumption, the depth of the above circuit $\mathrm{F}[\mathbf{x}, \mathbf{r}, \mathsf{SKE.ct}]$ can be bounded by a fixed polynomial $\mathrm{poly}(\lambda)$.
- Compute $\mathsf{prFE.sk} \leftarrow \mathsf{prFE.KeyGen}(\mathsf{prFE.msk}, \mathrm{F}[\mathbf{x}, \mathbf{r}])$.
- Output $\mathsf{cpABE.sk}_{\mathbf{x}} := (\mathsf{prFE.sk}, \mathsf{SKE.ct}, \mathbf{r})$.

$\mathsf{Enc}(\mathsf{cpABE.mpk}, C, \mu) \to \mathsf{cpABE.ct}$. The encryption algorithm does the following.

- Parse $\mathsf{cpABE.mpk} = \mathsf{prFE.mpk}$ and sample a PRF key $\mathsf{sd} \leftarrow \{0,1\}^\lambda$.

- Sample $R_{\mathsf{key}} \leftarrow \{0,1\}^\lambda$ and generate $(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk}) = \mathsf{kpABE.Setup}(1^\lambda, 1^\ell; R_{\mathsf{key}})$.
- Compute $\mathsf{prFE.ct} \leftarrow \mathsf{prFE.Enc}(\mathsf{prFE.mpk}, (R_{\mathsf{key}}, \mathsf{sd}, \mu, 0, \perp))$.
- Compute $\mathsf{kpABE.sk}_C \leftarrow \mathsf{kpABE.KeyGen}(\mathsf{kpABE.msk}, C)$.
- Output $\mathsf{cpABE.ct} := (\mathsf{prFE.ct},\ \mathsf{kpABE.mpk},\ \mathsf{kpABE.sk}_C)$.

$\mathsf{Dec}(\mathsf{cpABE.mpk}, \mathsf{cpABE.sk}_{\mathbf{x}}, \mathbf{x}, \mathsf{cpABE.ct}, C)$. The decryption algorithm does the following.

- Parse $\mathsf{cpABE.mpk} = \mathsf{prFE.mpk}, \mathsf{cpABE.sk}_{\mathbf{x}} = (\mathsf{prFE.sk}, \mathsf{SKE.ct}, \mathbf{r})$ and $\mathsf{cpABE.ct} = (\mathsf{prFE.ct},\ \mathsf{kpABE.sk}_C)$.
- Compute $\mathbf{y} = \mathsf{prFE.Dec}(\mathsf{prFE.mpk}, \mathsf{prFE.sk}, \mathrm{F}[\mathbf{x}, \mathbf{r}, \mathsf{SKE.ct}], \mathsf{prFE.ct})$.
- Compute and output $\mathsf{kpABE.Dec}(\mathsf{kpABE.mpk}, \mathsf{kpABE.sk}_C, C, \mathbf{y}, \mathbf{x})$.

Correctness. We prove the correctness of our scheme using the following theorem.

Theorem 6.1. Assume kpABE is correct and PRF is secure. Then the above construction of cpABE scheme is correct.

Proof. From the correctness of prFE scheme, with probability 1 we have

$$\mathbf{y} = \mathrm{F}[\mathbf{x}, \mathbf{r}, \mathsf{SKE.ct}](R_{\mathsf{key}}, \mathsf{sd}, \mu, 0, \perp) = \mathsf{kpABE.ct}$$

where $\mathsf{kpABE.ct} = \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathbf{x}, \mu; \mathsf{PRF}(\mathsf{sd}, \mathbf{r}))$ and $(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk}) = \mathsf{kpABE.Setup}(1^\lambda, 1^\ell; R_{\mathsf{key}})$. Next, using the security of PRF, $\mathsf{PRF}(\mathsf{sd}, \mathbf{r})$ is indistinguishable from $R \leftarrow \{0,1\}^{\mathsf{R_{len}}}$. Furthermore, by the correctness of kpABE, we have

$$\mathsf{kpABE.Dec}(\mathsf{kpABE.sk}_C, C, \mathsf{kpABE.ct}, \mathbf{x}) = \mathsf{kpABE.Dec}(\mathsf{kpABE.sk}_C, \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathbf{x}, \mu; \mathsf{PRF}(\mathsf{sd}, \mathbf{r}))) = \mu.$$

if $C(\mathbf{x}) = 1$, with all but negl probability. Therefore, by the security of PRF, the correctness follows. □

Remark 6.2. For the above conversion to work, it is important that the ciphertext length $\ell_{\mathsf{ct}}^{\mathsf{kpABE}}$ of kpABE is independent from the depth of the circuits being supported by the scheme. Otherwise, the key generator, who does not know the depth of the circuit associated with the ciphertext, cannot choose sufficiently long output length for the circuit $\mathrm{F}[\mathbf{x}, \mathbf{r}, \mathsf{SKE.ct}]$ and we cannot achieve correctness.

6.2 Security

We prove the security of our scheme via the following theorem.

Theorem 6.3. Assume that the prFE scheme is IND-secure (Definition 4.4), kpABE scheme satisfies VerSel-INDr security (Definition 3.22) then the construction of cpABE satisfies VerSel-IND security.

Proof. Suppose the adversary $\mathcal{A}$ with randomness $\mathsf{coins}_{\mathcal{A}}$ queries for $C, (\mu_0, \mu_1), \mathbf{x}^1, \ldots \mathbf{x}^Q$. We prove the security via the following sequence of hybrids.

Hyb_0. This is the VerSel-IND game. The adversary sees

$$\begin{pmatrix} \mathsf{coins}_{\mathcal{A}},\ \mathsf{cpABE.mpk} = \mathsf{prFE.mpk},\ \mathsf{cpABE.sk}_{\mathbf{x}^k} = \{\mathsf{prFE.sk}^k, \mathsf{SKE.ct}^k, \mathbf{r}^k\}_{k \in [Q]}, \\ \mathsf{cpABE.ct} = (\mathsf{prFE.ct} \leftarrow \mathsf{prFE.Enc}(\mathsf{prFE.mpk}, (R_{\mathsf{key}}, \mathsf{sd}, \mu_\beta, 0, \perp)), \mathsf{kpABE.mpk}, \mathsf{kpABE.sk}_C) \end{pmatrix}$$

where $\mathsf{prFE.sk}^k \leftarrow \mathsf{prFE.KeyGen}(\mathsf{prFE.msk}, \mathrm{F}[\mathbf{x}^k, \mathbf{r}^k, \mathsf{SKE.ct}^k])$, $\mathsf{SKE.ct}^k \leftarrow \mathcal{CT}_{\mathsf{SKE}}, \mathbf{r}^k \leftarrow \{0,1\}^\lambda$ for all $k \in [Q]$, $R_{\mathsf{key}} \leftarrow \{0,1\}^\lambda, \mathsf{sd} \leftarrow \{0,1\}^\lambda$ and $\mathsf{kpABE.sk}_C \leftarrow \mathsf{kpABE.KeyGen}(\mathsf{kpABE.msk}, C)$. Also for all the key queries $\mathbf{x}_1, \ldots, \mathbf{x}_Q$ and the challenge circuit C issued by the adversary, we have $C[\mathbf{x}^k] = 0$.

Hyb_1. This hybrid is the same as Hyb_0 except that $\mathsf{SKE.sk} \leftarrow \mathcal{K}_{\mathsf{SKE}}$ is chosen and $\mathsf{SKE.ct}^k$ is computed as

$$\mathsf{SKE.ct}^k = \mathsf{SKE.Enc}(\mathsf{SKE.sk}, \mathsf{kpABE.ct}^k) \tag{27}$$

where

$$\mathsf{kpABE.ct}^k = \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathbf{x}^k, \mu_\beta; \mathsf{PRF}(\mathsf{sd}, \mathbf{r}^k)) \tag{28}$$

for $k \in [Q]$. By the pseudorandom ciphertext security of SKE, this hybrid is computationally indistinguishable from the previous one.

Hyb_2. This hybrid is the same as Hyb_1 except that $\mathsf{prFE.ct}$ is computed as

$$\mathsf{prFE.ct} \leftarrow \mathsf{prFE.Enc}(\mathsf{prFE.mpk}, (\bot, \bot, \bot, 1, \mathsf{SKE.sk})).$$

We claim that this game is computationally indistinguishable from the previous hybrid by the IND-pr-Security of prFE. To see this, we first observe that we have

$$\begin{aligned}
\mathrm{F}[\mathbf{x}^k, \mathbf{r}^k, \mathsf{SKE.ct}^k](R_{\mathsf{key}}, \mathsf{sd}, \mu_\beta, 0, \bot) &= \mathsf{kpABE.ct}^k \\
&= \mathsf{SKE.Dec}(\mathsf{SKE.sk}, \mathsf{SKE.ct}^k) \\
&= \mathrm{F}[\mathbf{x}^k, \mathbf{r}^k, \mathsf{SKE.ct}^k](\bot, \bot, \bot, 1, \mathsf{SKE.sk})
\end{aligned}$$

for all $k \in [Q]$ and $i \in [q_{C^k}]$ by the correctness of prFE and SKE, where $\mathsf{kpABE.ct}^k$ is defined as Equation (27). As we will show later, we have

$$\begin{aligned}
&\left(\mathsf{coins}_{\mathcal{A}}, \mathsf{kpABE.mpk}, \{\mathrm{F}[\mathbf{x}^k, \mathbf{r}^k, \mathsf{SKE.ct}^k],\ \mathsf{kpABE.ct}^k\}_{k \in [Q]}\right) \\
\approx_c &\left(\mathsf{coins}_{\mathcal{A}}, \mathsf{kpABE.mpk}, \{\mathrm{F}[\mathbf{x}^k, \mathbf{r}^k, \mathsf{SKE.ct}^k],\ \mathsf{kpABE.ct}^k \leftarrow \{0,1\}^{\ell_{\mathsf{ct}}^{\mathsf{kpABE}}}\}_{k \in [Q]}\right)
\end{aligned} \tag{29}$$

where $\mathsf{kpABE.ct}^k$ is defined as Equation (28) and $\mathsf{SKE.ct}_i^k$ is defined as Equation (27). Therefore, we can invoke the security of prFE.

Hyb_3. This hybrid is the same as Hyb_2 except that $\mathsf{kpABE.ct}_i^k$ is replaced with $\mathsf{kpABE.ct}^k \leftarrow \{0,1\}^{\ell_{\mathsf{ct}}^{\mathsf{kpABE}}}$. By Equation (29), this hybrid is computationally indistinguishable from the previous one. Notice that in this hybrid, the view of the adversary is independent from the challenge bit β.

The fact that Hyb_0 and Hyb_3 are computationally indistinguishable means that the adversary has negligible advantage in Hyb_0, since its advantage in Hyb_3 is 0. It remains to prove Equation (29). We prove Equation (29) via the following sequence of hybrids.

Hyb_0'. This is the LHS distribution of Equation (29) except that we give $\mathbf{x}^k, \mathbf{r}^k\}_{k \in [Q]}$ instead of $\{\mathrm{F}[\mathbf{x}^k, \mathbf{r}^k, \mathsf{SKE.ct}^k]\}_{k \in [Q]}$ to the distinguisher:

$$\left(\mathsf{coins}_{\mathcal{A}}, \mathsf{kpABE.mpk}, \left\{\mathbf{x}^k, \mathbf{r}^k, \mathsf{kpABE.ct}^k = \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathbf{x}^k, \mu_\beta; \mathsf{PRF}(\mathsf{sd}, \mathbf{r}^k))\right\}_{k \in [Q]}\right).$$

Hyb_1'. This hybrid is same as the previous one except that we output a failure symbol if the set $\{\mathbf{r}^k\}_{k \in [Q]}$ contains a collision. We prove that the probability with which there occurs a collision is negligible in λ. To prove this it suffices to show that there is no $k, k' \in [Q]$ such that $k \neq k'$ and $\mathbf{r}^k = \mathbf{r}^{k'}$. The probability of this happening can be bounded by $Q^2/2^\lambda$ by taking the union bound with respect to all the combinations of k, k'. Thus the probability of outputting the failure symbol is $Q^2/2^\lambda$ which is $\mathsf{negl}(\lambda)$.

Hyb'_2. In this hybrid we change all the PRF values computed using sd to random. Namely, we replace $\mathsf{PRF}(\mathsf{sd}, \mathbf{r}^k)$ with true randomness. Since PRF is invoked for fresh input for each $k \in [Q]$, this hybrid is indistinguishable from the previous hybrid. We now consider the following distribution:

$$\left(\mathsf{coins}_{\mathcal{A}}, \mathsf{kpABE.mpk}, \{\mathbf{x}^k, \mathbf{r}^k, \mathsf{kpABE.ct}^k = \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathbf{x}^k, \mu_\beta)\}_{k \in [Q]}\right)$$

Hyb'_3. In this hybrid we invoke the security of kpABE scheme to switch $\mathsf{kpABE.ct}^k$ to random for all $k \in [Q]$. Namely, the distribution is now:

$$\left(\mathsf{coins}_{\mathcal{A}}, \mathsf{kpABE.mpk}, \{\mathbf{x}^k,\ \mathbf{r}^k,\ \mathsf{kpABE.ct}^k \leftarrow \mathcal{CT}_{\mathsf{kpABE}}\}_{k \in [Q]}\right)$$

By the admissibility of the adversary and very selective pseudorandom ciphertext security of kpABE, this hybrid is indistinguishable from the previous one.

We therefore have $\mathsf{Hyb}'_0 \approx_c \mathsf{Hyb}'_3$, which implies Equation (29), since $\mathsf{SKE.ct}^k$ can be simulated by sampling SKE.sk and encrypting $\mathsf{kpABE.ct}^k$. This completes the proof. □

Instantiations of CP-ABE Instantiating the building block kpABE by the single-key secure KP-ABE from circular small-secret LWE [HLL23], we have $|\mathsf{kpABE.mpk}| = \mathrm{poly}(\ell, \lambda), |\mathsf{kpABE.sk}_C| = \mathrm{poly}(\lambda),\ |\mathsf{kpABE.ct}| = \mathrm{poly}(\ell, \lambda)$. Furthermore, it is easy to see that the depth of the circuits implementing the setup and the encryption algorithms can be independent of ℓ in their scheme by appropriate parallelization. Namely, we have $d_{\mathsf{key}}^{\mathsf{kpABE}} = \mathrm{poly}(\lambda)$ and $d_{\mathsf{ct}}^{\mathsf{kpABE}} = \mathrm{poly}(\lambda)$. Hence we use a prFE scheme supporting $d_{\mathsf{prFE}} = \mathrm{poly}(\lambda)$ depth circuits with input length $\mathrm{L} = \mathrm{poly}(\lambda)$ and output length $\ell_{\mathsf{ct}}^{\mathsf{kpABE}} = \mathrm{poly}(\ell, \lambda)$ for this instantiation.

Theorem 6.4. Assuming circular small-secret LWE and IND-secure prFE (Definition 4.4), there exists a very selectively secure cpABE scheme for circuits $\{\mathsf{C} : \{0,1\}^\ell \to \{0,1\}\}$ of unbounded depth with

$$|\mathsf{cpABE.mpk}| = \mathrm{poly}(\lambda), \quad |\mathsf{cpABE.sk}_{\mathbf{x}}| = \mathrm{poly}(\ell, \lambda), \quad |\mathsf{cpABE.ct}_{\mathsf{C}}| = \mathrm{poly}(\lambda).$$

Implications to ABE for Turing Machines. We note that our unbounded CP-ABE scheme in Theorem 6.4 can be used to instantiate ABE for Turing Machines ([AKY24]). We get the following corollary.

Corollary 6.5. Assuming circular small-secret LWE and IND-secure prFE (Definition 4.4), there exists a very selectively secure ABE for TM with

$$|\mathsf{mpk}| = \mathrm{poly}(\lambda), \quad |\mathsf{sk}| = \mathrm{poly}(\lambda, |M|), \quad |\mathsf{ct}| = \mathrm{poly}(\lambda, |\mathbf{x}|, t).$$

[AKY24] uses the LWE assumption, evasive LWE assumption and circular tensor assumption for their construction with the same parameters as above. Since our prFE can be instantiated from the LWE and the evasive LWE, we can remove the circular tensor assumption from [AKY24] to base the security on the evasive LWE and the LWE.

7 Multi-Input FE for Pseudorandom Functionalities

In this section, we construct our main tool – multi-input functional encryption for pseudorandom functionalities.

7.1 Definition

In this section we give the definitions for multi-input functional encryption for pseudorandom functionalities (prMIFE). Consider a function family $\{\mathcal{F}_{\mathsf{prm}} = \{f : (\mathcal{X}_{\mathsf{prm}})^n \to \mathcal{Y}_{\mathsf{prm}}\}\}_{\mathsf{prm}}$, for a parameter $\mathsf{prm} = \mathsf{prm}(\lambda)$, where each $\mathcal{F}_{\mathsf{prm}}$ is a finite collection of n-ary functions. Each function $f \in \mathcal{F}_{\mathsf{prm}}$ takes as input strings $x_1, \ldots, x_n$, where each $x_i \in \mathcal{X}_{\mathsf{prm}}$ and outputs $f(x_1, \ldots, x_n) \in \mathcal{Y}_{\mathsf{prm}}$.

Syntax. A miprfe scheme prMIFE_n for n-ary function family $\mathcal{F}_{\mathsf{prm}}$ consists of polynomial time algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}_1, \ldots, \mathsf{Enc}_n, \mathsf{Dec})$ defined as follows.

$\mathsf{Setup}(1^\lambda, 1^n, \mathsf{prm}) \to (\mathsf{mpk}, \mathsf{msk})$. The setup algorithm takes as input the security parameter λ, the function arity n and a parameter prm and outputs a master public key mpk and master secret key msk[9].

$\mathsf{KeyGen}(\mathsf{msk}, f) \to \mathsf{sk}_f$. The key generation algorithm takes as input the master secret key msk and a function $f \in \mathcal{F}_{\mathsf{prm}}$ and it outputs a functional secret key sk_f.

$\mathsf{Enc}_i(\mathsf{msk}, x) \to \mathsf{ct}$. The encryption algorithm for the i-th slot takes as input the master secret key msk and an input $x \in \mathcal{X}_{\mathsf{prm}}$ and outputs a ciphertext $\mathsf{ct}_i \in \mathcal{CT}$, where $\mathcal{CT}$ is the ciphertext space.

$\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_f, f, \mathsf{ct}_1, \ldots, \mathsf{ct}_n) \to y$. The decryption algorithm takes as input the master public key mpk, secret key sk_f, function f and n ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_n$, and outputs $y \in \mathcal{Y}_{\mathsf{prm}}$.

Definition 7.1 (Correctness). A prMIFE scheme is said to be correct if for every prm, n-ary function $f \in \mathcal{F}_{\mathsf{prm}}$ and input tuple $(x_1, \ldots, x_n) \in \mathcal{X}^n_{\mathsf{prm}}$ we have

$$\Pr\left[\begin{array}{c}(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, \mathsf{prm})\ ,\ \mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f), \\ \mathsf{Dec}\big(\mathsf{mpk}, \mathsf{sk}_f, f, \mathsf{Enc}_1(\mathsf{msk}, x_1), \ldots \mathsf{Enc}_n(\mathsf{msk}, x_n)\big) = f(x_1, \ldots, x_n)\end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

Definition 7.2 (κ-Security and (κ, ϵ)-Security). Let $\kappa = \kappa(\lambda)$ be a function in λ. For a prMIFE scheme for function family $\{\mathcal{F}_{\mathsf{prm}} = \{f : (\mathcal{X}_{\mathsf{prm}})^n \to \mathcal{Y}_{\mathsf{prm}}\}\}_{\mathsf{prm}}$, parameter $\mathsf{prm} = \mathsf{prm}(\lambda)$, let Samp be a PPT algorithm that on input 1^λ, outputs

$$\Big(\{f_k\}_{k \in [q_0]}, \{x_1^{j_1}\}_{j_1 \in [q_1]}, \ldots, \{x_n^{j_n}\}_{j_n \in [q_n]}, \mathsf{aux} \in \{0,1\}^*\Big)$$

where q_0 is the number of key queries, q_i is the number of encryption queries for the i-th slot, $f_1, \ldots, f_{q_0} \in \mathcal{F}_{\mathsf{prm}}$ and $x_i^{j_i} \in \mathcal{X}_{\mathsf{prm}}$ for all $i \in [n], j_i \in [q_i]$. We say that the prMIFE scheme satisfies κ-security with respect to the sampler class $\mathcal{SC}$ if for every PPT sampler Samp $\in \mathcal{SC}$ there exists a PPT simulator algorithm $\{\mathsf{Sim}_i\}_{i \in [n]}$ such that

$$\text{If}\ \left(1^\kappa, \left\{f_k, f_k(x_1^{j_1}, \ldots, x_n^{j_n})\right\}_{k \in [q_0], j_1 \in [q_1] \ldots, j_n \in [q_n]}, \mathsf{aux}\right) \approx_c \left(1^\kappa, \left\{f_k, \Delta_{k, j_1, \ldots, j_n}\right\}_{k \in [q_0], j_1 \in [q_1], \ldots, j_n \in [q_n]}, \mathsf{aux}\right) \quad (30)$$

$$\text{then}\ \left(\mathsf{mpk}, \left\{f_k, \mathsf{sk}_{f_k}\right\}_{k \in [q_0]}, \left\{\mathsf{ct}_i^{j_i}\right\}_{i \in [n], j_i \in [q_i]}, \mathsf{aux}\right) \approx_c \left(\mathsf{mpk}, \left\{f_k, \mathsf{sk}_{f_k}\right\}_{k \in [q_0]}, \left\{\delta_i^{j_i}\right\}_{i \in [n], j_i \in [q_i]}, \mathsf{aux}\right), \quad (31)$$

where $\kappa \geq \lambda^n$, $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, \mathsf{prm})$, $\mathsf{sk}_{f_k} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f_k)$, $\mathsf{ct}_i^{j_i} \leftarrow \mathsf{Enc}(\mathsf{msk}, x_i^{j_i})$, $\delta_i^{j_i} \leftarrow \mathsf{Sim}_i(\mathsf{msk})$, and $\Delta_{k, j_1, \ldots, j_n} \leftarrow \mathcal{Y}_{\mathsf{prm}}$ for $i \in [n]$, $j_i \in [q_i]$, and $k \in [q_0]$.

We also define the parametrized version of the definition that we call (κ, ϵ)-*security*, where we require that when Equation (30) holds, the two distributions in Equation (31) should not be distinguishable for any PPT adversary with advantage more than ϵ. Therefore, (κ, ϵ)-security for all inverse polynomial ϵ implies κ security. When we consider (κ, ϵ)-security, we typically consider sub-exponentially small ϵ.

Remark 7.3. Note that 1^κ in Equation (30) is introduced for the purpose of padding, allowing the distinguisher for the distributions to run in time polynomial in κ and requiring the distinguishing advantage to be negligible in κ.[10] The reason why we require $\kappa \geq \lambda^n$ is that the input length to the distinguisher is polynomial in λ^n anyway and in order for the padding to make sense, κ should satisfy this condition. If we need κ to be larger, this doubly strengthens the requirement for the precondition, as it means we want the distributions in Equation (30) to be indistinguishable against an adversary with a longer running time and smaller advantage. Ideally, we want κ to be as small as λ^n to make the requirement weaker. However, the security proof for our construction in Section 7.2 for general n requires large κ as an artifact of the proof technique. In the special case of n being constant, we can achieve $\kappa = \lambda^n$.

[9]We assume w.l.o.g that msk includes mpk.

[10]This is due to our convention, where the running time of the distinguisher should be polynomial in its input length and the distinguishing advantage should be negligible in its input length. Please refer to Section 3 for the details.

It is shown in [AMYY25, BDJ+25] that there is no prMIFE that satisfies the above style security for all general samplers. We can only hope that the prMIFE satisfies the above security with respect to some class of samplers. To get around the impossibility, we also define the following indistinguishability based variant of the security definition for prMIFE. Our definition is a strict weakening of the standard indistinguishability security notion for multi-input FE, since we require that the security should hold only for the case where the decryption results obtained by all combinations of the ciphertexts and secret keys are jointly pseudorandom. It is known that multi-input FE with standard IND-based security is achievable assuming the existence of FE [AJ15], so our definition below is achievable in principle, though we do not know an instantiation of the latter from standard post-quantum assumptions.

Definition 7.4 (κ-IND-Security and (κ,ϵ)-IND-Security). Let $\kappa = \kappa(\lambda)$ be a function in λ. For a prMIFE scheme for function family $\{\mathcal{F}_{\mathsf{prm}} = \{f : (\mathcal{X}_{\mathsf{prm}})^n \to \mathcal{Y}_{\mathsf{prm}}\}\}_{\mathsf{prm}}$, parameter $\mathsf{prm} = \mathsf{prm}(\lambda)$, let Samp be a PPT algorithm that on input 1^λ, outputs

$$\left(\{f_k\}_{k\in[q_0]}, \{x^{j_1}_{1,0}, x^{j_1}_{1,1}\}_{j_1\in[q_1]}, \ldots, \{x^{j_n}_{n,0}, x^{j_n}_{n,1}\}_{j_n\in[q_n]}, \mathsf{aux} \in \{0,1\}^*\right)$$

where q_0 is the number of key queries, q_i is the number of encryption queries for the i-th slot, $f_1, \ldots, f_{q_0} \in \mathcal{F}_{\mathsf{prm}}$ and $x^{j_i}_{i,b} \in \mathcal{X}_{\mathsf{prm}}$ for all $i \in [n], b \in \{0,1\}, j_i \in [q_i]$. We say that the prMIFE scheme satisfies *κ-IND-security* if for every PPT sampler Samp such that

$$f_k(x^{j_1}_{1,0}, \ldots, x^{j_n}_{n,0}) = f_k(x^{j_1}_{1,1}, \ldots, x^{j_n}_{n,1}) \qquad \forall j_1 \in [q_1], \ldots, \forall j_n \in [q_n], \forall k \in [q_0] \tag{32}$$

and

$$\left(1^\kappa, \left\{f_k, f_k(x^{j_1}_1, \ldots, x^{j_n}_n)\right\}_{k\in[q_0], j_1\in[q_1]\ldots,j_n\in[q_n]}, \mathsf{aux}\right) \approx_c \left(1^\kappa, \left\{f_k, \Delta_{k,j_1,\ldots,j_n}\right\}_{k\in[q_0], j_1\in[q_1],\ldots,j_n\in[q_n]}, \mathsf{aux}\right), \tag{33}$$

we have

$$\left(\mathsf{mpk}, \left\{f_k, \mathsf{sk}_{f_k}\right\}_{k\in[q_0]}, \left\{\mathsf{ct}^{j_i}_{i,0}\right\}_{i\in[n], j_i\in[q_i]}, \mathsf{aux}\right) \approx_c \left(\mathsf{mpk}, \left\{f_k, \mathsf{sk}_{f_k}\right\}_{k\in[q_0]}, \left\{\mathsf{ct}^{j_i}_{i,1}\right\}_{i\in[n], j_i\in[q_i]}, \mathsf{aux}\right), \tag{34}$$

where $\kappa \geq \lambda^n$, $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, \mathsf{prm})$, $\mathsf{sk}_{f_k} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f_k)$, $\mathsf{ct}^{j_i}_{i,b} \leftarrow \mathsf{Enc}(\mathsf{msk}, x^{j_i}_{i,b})$, $\Delta_{k,j_1,\ldots,j_n} \leftarrow \mathcal{Y}_{\mathsf{prm}}$ for $i \in [n]$, $j_i \in [q_i]$, $b \in \{0,1\}$, and $k \in [q_0]$.

We also define the parametrized version of the definition that we call (κ,ϵ)*-IND-security*, where we require that when Equation (32) and (33) hold, the two distributions in Equation (34) should not be distinguishable for any PPT adversary with advantage more than ϵ. Therefore, (κ,ϵ)-IND security for all inverse polynomial ϵ implies κ-IND security. When we consider (κ,ϵ)-IND security, we typically consider sub-exponentially small ϵ.

We note that κ-IND security above is weaker than the IND-based security defined for MIFE in [AJ15], since it requires the ciphertext should hide the message (i.e., Equation (34)) only when the decryption results are pseudorandom (i.e., Equation (33)). Furthermore, it is weaker in that it requires the adversary to submit the key queries in addition to the encryption queries at the beginning of the game.

7.2 Construction for n-input prFE

In this section we provide our construction of a multi-input functional encryption scheme for pseudorandom functionalities for function family $\mathcal{F}_{nL(\lambda),d(\lambda)} = \{f : \{\{0,1\}^L\}^n \to \{0,1\}\}$, where the depth of a function $f \in \mathcal{F}$ is at most $d(\lambda) = \mathrm{poly}(\lambda)$. Each function $f \in \mathcal{F}$ takes as input strings $x_1, \ldots, x_n \in \{0,1\}^L$ and outputs $f(x_1, \ldots, x_n) \in \{0,1\}$. We consider the case of arity n being constant and the general case of n being arbitrary polynomial in λ. While we provide separate security proofs for these cases, we have unified description of the construction. The reason why we consider the proofs separately is that we can base the security of the scheme on a weaker assumption when n is constant than the general case.

Building Blocks. Our construction uses the following building blocks.

1. n single-input FE scheme for pseudorandom functionality $\mathsf{prFE}_1, \ldots, \mathsf{prFE}_n$. For $i \in [n]$, $\mathsf{prFE}_i = (\mathsf{prFE}_i.\mathsf{Setup}, \mathsf{prFE}_i.\mathsf{KeyGen}, \mathsf{prFE}_i.\mathsf{Enc}, \mathsf{prFE}_i.\mathsf{Dec})$ for circuit class $\mathcal{C}_{\mathsf{inp}_i(\lambda),\mathsf{dep}_i(\lambda),\mathsf{out}_i(\lambda)}$ consisting of circuits with input length $\mathsf{inp}_i(\lambda)$, maximum depth $\mathsf{dep}_i(\lambda)$ and output length $\mathsf{out}_i(\lambda)$. We denote the ciphertext space of the prFE_i scheme by $\mathcal{CT}_{\mathsf{prFE}_i}$.
   For our construction, we set the following parameters
   - $\mathsf{inp}_1 = n \cdot L$, $\mathsf{dep}_1 = d$, and $\mathsf{out}_1 = 1$.
   - $\mathsf{inp}_i = |\mathsf{SKE.key}| + (n-i)L + n\Lambda$, $\mathsf{dep}_i = \mathrm{poly}(d, \lambda)$, and $\mathsf{out}_i = |\mathsf{prFE}_{i-1}.\mathsf{ct}|$ for $i \in [2, n]$, where $\mathsf{SKE.key} \in \mathcal{CT}_{\mathsf{SKE}}$ and $\mathsf{prFE}_{i-1}.\mathsf{ct} \in \mathcal{CT}_{\mathsf{prFE}_{i-1}}$.
2. We also use $n-1$ pseudorandom functions $\mathsf{PRF}_1, \ldots, \mathsf{PRF}_{n-1}$. Similarly to the case of SKE, we use Λ to setup these instances of PRF. We specify the domain and codomain of the functions as $\mathsf{PRF}_i : \{0,1\}^\Lambda \times \{\{0,1\}^\Lambda\}^{n-i} \to \{0,1\}^{\mathsf{len}_i}$ where len_i is the length of randomness used in $\mathsf{prFE}_i.\mathsf{Enc}$ for $i \in [n-1]$.

We describe our construction of $\mathsf{prMIFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}_1, \ldots, \mathsf{Enc}_n, \mathsf{Dec})$ in the following.

$\mathsf{Setup}(1^\lambda, 1^n, \mathsf{prm}) \to (\mathsf{mpk}, \mathsf{msk})$. The setup algorithm does the following.

- For all $i \in [n]$, generate $(\mathsf{prFE}_i.\mathsf{mpk}, \mathsf{prFE}_i.\mathsf{msk}) \leftarrow \mathsf{prFE}_i.\mathsf{Setup}(1^\lambda, 1^{\mathsf{prm}_i})$.
- Generate $\mathsf{SKE.sk} \leftarrow \mathsf{SKE.Setup}(1^\Lambda)$.
- Output $\mathsf{mpk} := (\{\mathsf{prFE}_i.\mathsf{mpk}\}_{i\in[n]})$ and $\mathsf{msk} := (\mathsf{SKE.sk}, \{\mathsf{prFE}_i.\mathsf{msk}, \mathsf{prFE}_i.\mathsf{mpk}\}_{i\in[n]})$.

$\mathsf{KeyGen}(\mathsf{msk}, f) \to \mathsf{sk}_f$. The key generation algorithm does the following.

- Parse $\mathsf{msk} = (\mathsf{SKE.sk}, \{\mathsf{prFE}_i.\mathsf{msk}, \mathsf{prFE}_i.\mathsf{mpk}\}_{i\in[n]})$.
- Compute $\mathsf{prFE}_1.\mathsf{sk}_f \leftarrow \mathsf{prFE}_1.\mathsf{KeyGen}(\mathsf{prFE}_1.\mathsf{msk}, f)$.
- Output $\mathsf{sk}_f := \mathsf{prFE}_1.\mathsf{sk}_f$.

$\mathsf{Enc}_i(\mathsf{msk}, \mathbf{x}_i) \to \mathsf{ct}_i$. For $i \in [n-1]$, the Enc_i algorithm outputs a function secret key corresponding to prFE_{i+1}-th instance in the following way.

- Parse $\mathsf{msk} = (\mathsf{SKE.sk}, \{\mathsf{prFE}_i.\mathsf{msk}, \mathsf{prFE}_i.\mathsf{mpk}\}_{i\in[n]})$.
- Sample $\mathbf{r}_i \leftarrow \{0,1\}^\Lambda$.
- Compute $\mathsf{SKE.ct}_i \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.sk}, \mathbf{x}_i)$.
- Define $\mathrm{F}_i := \mathrm{F}_i[\mathsf{SKE.ct}_i, \mathbf{r}_i, \mathsf{prFE}_i.\mathsf{mpk}]$ as in Figure 2.[11]
- Compute $\mathsf{prFE}_{i+1}.\mathsf{sk} \leftarrow \mathsf{prFE}_{i+1}.\mathsf{KeyGen}(\mathsf{prFE}_{i+1}.\mathsf{msk}, \mathrm{F}_i)$.
- Output $\mathsf{ct}_i := \mathsf{prFE}_{i+1}.\mathsf{sk}$.

$\mathsf{Enc}_n(\mathsf{msk}, \mathbf{x}_n) \to \mathsf{ct}_n$. The Enc_n algorithm does the following.

- Parse $\mathsf{msk} = (\mathsf{SKE.sk}, \{\mathsf{prFE}_i.\mathsf{msk}, \mathsf{prFE}_i.\mathsf{mpk}\}_{i\in[n]})$.
- For $i \in [n-1]$, sample $K_i \leftarrow \{0,1\}^\Lambda$.
- Compute $\mathsf{prFE}_n.\mathsf{ct} \leftarrow \mathsf{prFE}_n.\mathsf{Enc}(\mathsf{prFE}_n.\mathsf{mpk}, (\mathsf{SKE.sk}, \mathbf{x}_n, K_1, \ldots, K_{n-1}))$.
- Output $\mathsf{ct}_n := \mathsf{prFE}_n.\mathsf{ct}$.

$\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_f, f, \mathsf{ct}_1, \ldots, \mathsf{ct}_n) \to y \in \{0,1\}$. The decryption algorithm does the following.

- Parse $\mathsf{mpk} = (\{\mathsf{prFE}_i.\mathsf{mpk}\}_{i\in[n]})$, $\mathsf{sk}_f = \mathsf{prFE}_1.\mathsf{sk}_f$, $\mathsf{ct}_i = \mathsf{prFE}_{i+1}.\mathsf{sk}$ for $i \in [n-1]$, and $\mathsf{ct}_n = \mathsf{prFE}_n.\mathsf{ct}$.

[11]The hardwired values are not hidden, even if we don't output them explicitly.

Function $\mathrm{F}_i[\mathsf{SKE.ct}_i, \mathbf{r}_i, \mathsf{prFE}_i.\mathsf{mpk}]$

Hardwired constants: A SKE ciphertext $\mathsf{SKE.ct}_i$, $\mathbf{r}_i \in \{0,1\}^{\Lambda}$, and a prFE master public key $\mathsf{prFE}_i.\mathsf{mpk}$.
On input $(\mathsf{SKE.sk}, (\mathbf{x}_{i+1}, \mathbf{r}_{i+1}), \ldots, (\mathbf{x}_{n-1}, \mathbf{r}_{n-1}), \mathbf{x}_n, K_1, \ldots K_i)$, proceed as follows:

1. Compute $\mathbf{x}_i := \mathsf{SKE.Dec}(\mathsf{SKE.sk}, \mathsf{SKE.ct}_i)$.
2. Compute $\mathsf{prFE}_i.\mathsf{ct}$ as
   – $\mathsf{prFE}_1.\mathsf{Enc}(\mathsf{prFE}_1.\mathsf{mpk}, (\mathbf{x}_1, \ldots, \mathbf{x}_n); \mathsf{PRF}_1(K_1, (\mathbf{r}_1, \ldots, \mathbf{r}_{n-1})))$ if $i = 1$.
   – $\mathsf{prFE}_i.\mathsf{Enc}(\mathsf{prFE}_i.\mathsf{mpk}, (\mathsf{SKE.sk}, (\mathbf{x}_i, \mathbf{r}_i), \ldots, (\mathbf{x}_{n-1}, \mathbf{r}_{n-1}), \mathbf{x}_n, K_1, \ldots K_{i-1}); \mathsf{PRF}_i(K_i, (\mathbf{r}_i, \ldots, \mathbf{r}_{n-1})))$, if $i \neq 1$.
3. Output $\mathsf{prFE}_i.\mathsf{ct}$.

Figure 2: Function F_i

– For $i = n, \ldots, 2$ and do the following.
  1. Compute $\mathsf{prFE}_{i-1}.\mathsf{ct} := \mathsf{prFE}_i.\mathsf{Dec}(\mathsf{prFE}_i.\mathsf{mpk}, \mathsf{prFE}_i.\mathsf{sk}, \mathrm{F}_{i-1}, \mathsf{prFE}_i.\mathsf{ct})$.
  2. If $i = 2$ output $\mathsf{prFE}_1.\mathsf{ct}$, else set $i := i - 1$ and go to Step 1.

– Output $y := \mathsf{prFE}_1.\mathsf{Dec}(\mathsf{prFE}_1.\mathsf{mpk}, \mathsf{prFE}_1.\mathsf{sk}_f, f, \mathsf{prFE}_1.\mathsf{ct})$.

Remark 7.5. We consider two cases of parameter settings for the construction. One is the case of n being constant. In this case, we simply set $\Lambda = \lambda$. In the general case of $n = \mathrm{poly}(\lambda)$, we do something more complex. In this case, we assume that PRF and SKE have subexponential security. This means that there exists $0 < \delta < 1$ such that there is no adversary with size $2^{\lambda^{\delta}}$ and distinguishing advantage $2^{-\lambda^{\delta}}$ against SKE and PRF for all sufficiently large λ. In the security proof, we require PRF and SKE to be secure even against an adversary that takes 1^{κ} as an input and thus runs in polynomial time in κ. To satisfy this requirement, we run SKE and PRF with respect to a larger security parameter Λ that satisfies $2^{\Lambda^{\delta}} \geq \kappa^{\omega(1)}$. An example choice would be to take $\Lambda := (n^2\lambda)^{1/\delta}$.

Efficiency. The scheme satisfies

$$|\mathsf{mpk}| = \mathrm{poly}(n, L, d, \Lambda, \lambda),\ |\mathsf{sk}_f| = \mathrm{poly}(d, \lambda),\ |\mathsf{ct}_1| = nL\mathrm{poly}(\mathsf{dep}, \lambda),\ |\mathsf{ct}_i| = \mathrm{poly}(n, L, d, \Lambda, \lambda) \text{ for } i \in [2, n].$$

Correctness. We prove the correctness of our scheme via the following theorem.

Theorem 7.6. Suppose prFE_i for $i \in [n]$ and SKE are correct, then the above construction of prMIFE satisfies correctness as defined in Definition 7.1.

Proof. To prove the theorem, we first prove the following statement.

Claim 7.7. For $i = n, \ldots, 2$, we have

$$\Pr[\mathsf{prFE}_i.\mathsf{Dec}(\mathsf{prFE}_i.\mathsf{mpk}, \mathsf{prFE}_i.\mathsf{sk}, \mathrm{F}_{i-1}, \mathsf{prFE}_i.\mathsf{ct}) = \mathsf{prFE}_{i-1}.\mathsf{ct}] = 1 \tag{35}$$

where

$$\mathsf{prFE}_{i-1}.\mathsf{ct} = \begin{cases} \mathsf{prFE}_{i-1}.\mathsf{Enc}\begin{pmatrix} \mathsf{prFE}_{i-1}.\mathsf{mpk}, (\mathsf{SKE.sk}, (\mathbf{x}_{i-1}, \mathbf{r}_{i-1}), \ldots, (\mathbf{x}_{n-1}, \mathbf{r}_{n-1}), \mathbf{x}_n, K_1, \ldots K_{i-2}) \\ ; \mathsf{PRF}_{i-1}(K_{i-1}, (\mathbf{r}_{i-1}, \ldots, \mathbf{r}_{n-1})) \end{pmatrix} & \text{if } i \neq 2 \\ \mathsf{prFE}_1.\mathsf{Enc}(\mathsf{prFE}_1.\mathsf{mpk}, (\mathbf{x}_1, \ldots, \mathbf{x}_n); \mathsf{PRF}_1(K_1, (\mathbf{r}_1, \ldots, \mathbf{r}_{n-1}))) & \text{if } i = 2. \end{cases}$$

Proof. We prove this by induction.
Base Case: For $i = n$, we show that

$$\Pr[\mathsf{prFE}_n.\mathsf{Dec}(\mathsf{prFE}_n.\mathsf{mpk}, \mathsf{prFE}_n.\mathsf{sk}, \mathrm{F}_{n-1}, \mathsf{prFE}_n.\mathsf{ct}) = \mathsf{prFE}_{n-1}.\mathsf{ct}] = 1.$$

From the correctness of prFE_n scheme, we have with probability 1

$$\begin{aligned}&\mathsf{prFE}_n.\mathsf{Dec}(\mathsf{prFE}_n.\mathsf{mpk}, \mathsf{prFE}_n.\mathsf{sk}, \mathrm{F}_{n-1}, \mathsf{prFE}_n.\mathsf{ct})\\ &\quad = \mathrm{F}_{n-1}[\mathsf{SKE.ct}_{n-1}, \mathbf{r}_{n-1}, \mathsf{prFE}_{n-1}.\mathsf{mpk}](\mathsf{SKE.sk}, \mathbf{x}_n, K_1, \ldots K_{n-1}).\end{aligned}$$

Next, by the definition of F_{n-1} and the correctness of the SKE scheme, we have

$$\begin{aligned}&\mathsf{prFE}_n.\mathsf{Dec}(\mathsf{prFE}_n.\mathsf{mpk}, \mathsf{prFE}_n.\mathsf{sk}, \mathrm{F}_{n-1}, \mathsf{prFE}_n.\mathsf{ct})\\ &\quad =\mathsf{prFE}_{n-1}.\mathsf{Enc}\begin{pmatrix}\mathsf{prFE}_{n-1}.\mathsf{mpk}, (\mathsf{SKE.sk}, (\mathbf{x}_{n-1}, \mathbf{r}_{n-1}), \mathbf{x}_n, K_1, \ldots K_{n-2}); \\ \hfill \mathsf{PRF}_{n-1}(K_{n-1}, \mathbf{r}_{n-1})\end{pmatrix}\end{aligned}$$

which proves the base case.
Inductive Step: For the inductive step, suppose Equation (35) holds for some $i \in [3, n]$ then we prove the same statement for $i - 1$. Consider

$$\begin{aligned}&\mathsf{prFE}_{i-1}.\mathsf{Dec}(\mathsf{prFE}_{i-1}.\mathsf{mpk}, \mathsf{prFE}_{i-1}.\mathsf{sk}, \mathrm{F}_{i-2}, \mathsf{prFE}_{i-1}.\mathsf{ct})\\ &= \mathrm{F}_{i-2}[\mathsf{SKE.ct}_{i-2}, \mathbf{r}_{i-2}, \mathsf{prFE}_{i-2}.\mathsf{mpk}](\mathsf{SKE.sk}, (\mathbf{x}_{i-1}, \mathbf{r}_{i-1}), \ldots, (\mathbf{x}_{n-1}, \mathbf{r}_{n-1}), \mathbf{x}_n, K_1, \ldots K_{i-2})\\ &= \begin{cases}\mathsf{prFE}_{i-2}.\mathsf{Enc}\begin{pmatrix}\mathsf{prFE}_{i-2}.\mathsf{mpk}, (\mathsf{SKE.sk}, (\mathbf{x}_{i-2}, \mathbf{r}_{i-2}), \ldots, \mathbf{x}_n, K_1, \ldots K_{i-3}); \\ \hfill \mathsf{PRF}_{i-2}(K_{i-2}, (\mathbf{r}_{i-2}, \ldots, \mathbf{r}_{n-1}))\end{pmatrix} & \text{if } i \neq 3 \\ \mathsf{prFE}_1.\mathsf{Enc}(\mathsf{prFE}_1.\mathsf{mpk}, (\mathbf{x}_1, \ldots, \mathbf{x}_n); \mathsf{PRF}_1(K_1, (\mathbf{r}_1, \ldots, \mathbf{r}_{n-1}))) & \text{if } i = 3.\end{cases}\end{aligned}$$

where in the first equality we use $\mathsf{prFE}_{i-1}.\mathsf{sk} = \mathsf{prFE}_{i-1}.\mathsf{KeyGen}(\mathsf{prFE}_{i-1}.\mathsf{msk}, \mathrm{F}_{i-2})$ and $\mathsf{prFE}_{i-1}.\mathsf{ct} = \mathsf{prFE}_{i-1}.\mathsf{Enc}$ $(\mathsf{prFE}_{i-1}.\mathsf{mpk}, (\mathsf{SKE.sk}, (\mathbf{x}_{i-1}, \mathbf{r}_{i-1}), \ldots, (\mathbf{x}_{n-1}, \mathbf{r}_{n-1}), \mathbf{x}_n, K_1, \ldots K_{i-2}); \mathsf{PRF}_{i-1}(K_{i-1}, (\mathbf{r}_{i-1}, \ldots, \mathbf{r}_{n-1})))$ which follows from the assumption for i. The second equality follows from the definition of F_{i-2} and the correctness of the SKE scheme.
This completes the proof of the inductive step. ☐

Using the above claim we get $\mathsf{prFE.ct}_1 = \mathsf{prFE}_1.\mathsf{Enc}(\mathsf{prFE}_1.\mathsf{mpk}, (\mathbf{x}_1, \ldots, \mathbf{x}_n); \mathsf{PRF}_1(K_1, (\mathbf{r}_1, \ldots, \mathbf{r}_{n-1})))$ from Step 7.2 of the decryption algorithm with probability 1. From the correctness of prFE_1 scheme, the decryption Step 7.2 outputs

$$\begin{aligned}y =&\mathsf{prFE}_1.\mathsf{Dec}(\mathsf{prFE}_1.\mathsf{mpk}, \mathsf{prFE}_1.\mathsf{sk}_f, f, \mathsf{prFE}_1.\mathsf{ct})\\ =&\mathsf{prFE}_1.\mathsf{Dec}(\mathsf{prFE}_1.\mathsf{mpk}, \mathsf{prFE}_1.\mathsf{sk}_f, f, \mathsf{prFE}_1.\mathsf{Enc}(\mathsf{prFE}_1.\mathsf{mpk}, (\mathbf{x}_1, \ldots, \mathbf{x}_n); \mathsf{PRF}_1(K_1, (\mathbf{r}_1, \ldots, \mathbf{r}_{n-1}))))\\ =&f(\mathbf{x}_1, \ldots, \mathbf{x}_n)\end{aligned}$$

with probability 1. ☐

7.3 Security Proof for General n

Theorem 7.8. Let $\mathcal{SC}_{\mathsf{prMIFE}}$ be a sampler class for prMIFE. Suppose prFE_i scheme satisfies non-uniform κ-prCT security as per Definition B.1 for $\kappa = \lambda^{n^2 \log \lambda}$ with respect to the sampler class that contains all $\mathsf{Samp}_{\mathsf{prFE}}(1^\lambda)$, induced by $\mathsf{Samp}_{\mathsf{prMIFE}} \in \mathcal{SC}_{\mathsf{prMIFE}}$, as in Equation (44), SKE satisfies sub-exponential INDr security and PRF_i is sub-exponentially secure, then prMIFE constructed above satisfies κ-security for $\kappa = \lambda^{n^2 \log \lambda}$ as per Definition 7.2. Furthermore, under the same assumptions, we have that prMIFE satsifies (κ, ϵ)-security with $\kappa = \lambda^{n^2 \log \lambda}$ and $\epsilon = \lambda^{-n \log \lambda / 2}$.

Proof. We first prove the former part of the theorem (i.e., κ-security). Consider a sampler $\mathsf{Samp}_{\mathsf{prMIFE}}$ that generates the following:

1. **Key Queries.** It issues q_0 number of functions $f_1, \ldots, f_{q_0}$ for key queries.

2. **Ciphertext Queries.** It issues q_i number of messages for ciphertext queries for slot i. We use $\mathbf{x}_i^{j_i}$ to denote the j_i-th ciphertext query corresponding to the i-th slot, where $j_i \in [q_i]$ and $i \in [n]$.

3. **Auxiliary Information.** It outputs the auxiliary information $\mathsf{aux}_{\mathcal{A}}$.

To prove the security of prMIFE, we first define $\{\mathsf{Sim}_i\}_{i\in[n]}$ as follows.

$\mathsf{Sim}_i(\mathsf{msk}) \to \mathsf{ct}_i$ for $i \in [n-1]$.

- Parse $\mathsf{msk} = (\mathsf{SKE.sk}, \{\mathsf{prFE}_i.\mathsf{msk}, \mathsf{prFE}_i.\mathsf{mpk}\}_{i\in[n]})$.
- Sample $\mathbf{r}_i \leftarrow \{0,1\}^{\Lambda}$ and $\gamma_i \leftarrow \mathcal{CT}_{\mathsf{SKE}}$.
- Compute $\mathsf{prFE}_{i+1}.\mathsf{sk} \leftarrow \mathsf{prFE}_{i+1}.\mathsf{KeyGen}(\mathsf{prFE}_{i+1}.\mathsf{msk}, \mathrm{F}_i[\gamma_i, \mathbf{r}_i, \mathsf{prFE}_i.\mathsf{mpk}])$.
- Output $\mathsf{ct}_i := \mathsf{prFE}_{i+1}.\mathsf{sk}$.

$\mathsf{Sim}_n(\mathsf{msk}) \to \mathsf{ct}_n$. Sample $\delta_n \leftarrow \mathcal{CT}_{\mathsf{prFE}_n}$ and output $\mathsf{ct}_n := \delta_n$.

Then, it suffices to show

$$\begin{pmatrix} \mathsf{aux}_{\mathcal{A}},\ \{\mathsf{prFE}_i.\mathsf{mpk}\}_{i\in[n]},\ \{\mathsf{prFE}_n.\mathsf{ct}^{j_n}\}_{j_n\in[q_n]}, \\ \left\{f_k,\ \mathsf{sk}_{f_k} = \mathsf{prFE}_1.\mathsf{sk}_{f_k}\right\}_{k\in[q_0]}, \left\{\mathsf{SKE.ct}_i^{j_i},\ \mathbf{r}_i^{j_i},\ \mathsf{prFE}_{i+1}.\mathsf{sk}^{j_i}\right\}_{\substack{i\in[n-1],\\ j_i\in[q_i]}}, \end{pmatrix}$$
$$\approx_c \begin{pmatrix} \mathsf{aux}_{\mathcal{A}},\ \{\mathsf{prFE}_i.\mathsf{mpk}\}_{i\in[n]},\ \left\{\delta_n^{j_n}\right\}_{j_n\in[q_n]} \\ \left\{f_k,\ \mathsf{sk}_{f_k} = \mathsf{prFE}_1.\mathsf{sk}_{f_k}\right\}_{k\in[q_0]}, \left\{\gamma_i^{j_i},\ \mathbf{r}_i^{j_i},\ \mathsf{prFE}_{i+1}.\mathsf{sk}^{j_i}\right\}_{\substack{i\in[n-1],\\ j_i\in[q_i]}} \end{pmatrix} \quad (36)$$

where $(\mathsf{aux}_{\mathcal{A}}, \{f_k\}_k, \{\mathbf{x}_i^{j_i}\}_{i,j_i}) \leftarrow \mathsf{Samp}_{\mathsf{prMIFE}}(1^\lambda)$,

$$\begin{aligned}
&\Big(\mathsf{mpk} = \{\mathsf{prFE}_i.\mathsf{mpk}\}_i, \mathsf{msk} = (\mathsf{SKE.sk}, \{\mathsf{prFE}_i.\mathsf{msk}, \mathsf{prFE}_i.\mathsf{mpk}\}_i)\Big) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, \mathsf{prm}),\\
&\mathsf{prFE}_1.\mathsf{sk}_{f_k} \leftarrow \mathsf{prFE}_1.\mathsf{KeyGen}(\mathsf{prFE}_1.\mathsf{msk}, f_k) \quad \text{for } k \in [q_0],\\
&\mathsf{SKE.ct}_i^{j_i} \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.sk}, \mathbf{x}_i^{j_i}), \quad \gamma_i^{j_i} \leftarrow \mathcal{CT}_{\mathsf{SKE}}, \quad \mathbf{r}_i^{j_i} \leftarrow \{0,1\}^{\Lambda},\\
&\mathsf{prFE}_{i+1}.\mathsf{sk}^{j_i} \leftarrow \begin{cases} \mathsf{prFE}_{i+1}.\mathsf{KeyGen}(\mathsf{prFE}_{i+1}.\mathsf{msk}, \mathrm{F}_i[\mathsf{SKE.ct}_i^{j_i}, \mathbf{r}_i^{j_i}, \mathsf{prFE}_i.\mathsf{mpk}]) & \text{in LHS of Eq. (36)}\\ \mathsf{prFE}_{i+1}.\mathsf{KeyGen}(\mathsf{prFE}_{i+1}.\mathsf{msk}, \mathrm{F}_i[\gamma_i^{j_i}, \mathbf{r}_i^{j_i}, \mathsf{prFE}_i.\mathsf{mpk}]) & \text{in RHS of Eq. (36)} \end{cases}\\
&\mathsf{prFE}_n.\mathsf{ct}^{j_n} \leftarrow \mathsf{prFE}_n.\mathsf{Enc}(\mathsf{prFE}_n.\mathsf{mpk}, (\mathsf{SKE.sk}, \mathbf{x}_n^{j_n}, K_1^{j_n}, \ldots, K_{n-1}^{j_n})),\ \delta_n^{j_n} \leftarrow \mathcal{CT}_{\mathsf{prFE}_n},\\
&K_i^{j_n} \leftarrow \{0,1\}^{\Lambda}, \quad \text{for } i \in [n-1],\ j_i \in [q_i], \text{ and } j_n \in [q_n]
\end{aligned}$$

assuming we have

$$\left(1^\kappa, \mathsf{aux}_{\mathcal{A}}, \left\{ f_k,\ f_k(\mathbf{x}_1^{j_1}, \ldots \mathbf{x}_n^{j_n}) \right\}_{k\in[q_0], j_1\in[q_1], \ldots, j_n\in[q_n]}\right)$$
$$\approx_c \left(1^\kappa, \mathsf{aux}_{\mathcal{A}}, \left\{ f_k,\ \Delta_k^{j_1,\ldots,j_n} \right\}_{k\in[q_0], j_1\in[q_1], \ldots, j_n\in[q_n]}\right) \quad (37)$$

where $(\mathsf{aux}_{\mathcal{A}}, \{f_k\}_k, \{\mathbf{x}_i^{j_i}\}_{i,j_i}) \leftarrow \mathsf{Samp}_{\mathsf{prMIFE}}(1^\lambda)$, and $\Delta_k^{j_1,\ldots,j_n} \leftarrow \{0,1\}$. We prove this in the following two steps.

- **Step 1.** We first show that Equation (37) implies

$$\begin{pmatrix} 1^\kappa, \quad \mathsf{aux}_{\mathcal{A}}, \quad \mathsf{prFE.mpk}_1 \\ \{\mathsf{SKE.ct}_i^{j_i}\}_{i\in[n-1],j_i\in[q_i]} \quad \{\mathsf{prFE}_1.\mathsf{ct}^{\mathbf{j}}\}_{\mathbf{j}\in[q_1]\times\cdots\times[q_n]} \\ \left\{f_k,\ \mathsf{sk}_{f_k} = \mathsf{prFE}_1.\mathsf{sk}_{f_k}\right\}_{k\in[q_0]} \end{pmatrix} \approx_c \begin{pmatrix} 1^\kappa, \quad \mathsf{aux}_{\mathcal{A}}, \quad \mathsf{prFE.mpk}_1 \\ \{\mathsf{SKE.ct}_i^{j_i}\}_{i\in[n-1],j_i\in[q_{\mathsf{msg}}]}, \quad \{\Delta^{\mathbf{j}}\}_{\mathbf{j}\in[q_1]\times\cdots\times[q_n]} \\ \left\{f_k,\ \mathsf{sk}_{f_k} = \mathsf{prFE}_1.\mathsf{sk}_{f_k}\right\}_{k\in[q_0]} \end{pmatrix} \tag{38}$$

where $\mathbf{j} = (j_1, \ldots, j_n) \in [q_1] \times \cdots \times [q_n]$, $\Delta^{\mathbf{j}} \leftarrow \mathcal{CT}_{\mathsf{prFE}_1}$, and

$$\mathsf{prFE}_1.\mathsf{ct}^{\mathbf{j}} \leftarrow \mathsf{prFE}_1.\mathsf{Enc}\left(\mathsf{prFE}_1.\mathsf{mpk}, (\mathbf{x}_1^{j_1}, \ldots \mathbf{x}_n^{j_n})\right).$$

- **Step 2.** We prove that Equation (38) implies Equation (36).

Step 1. We show the following lemma.

Lemma 7.9. If SKE satisfies subexponential INDr security, Equation (37) implies Equation (38).

Proof. We first prove the following:

$$\begin{aligned} &\left(1^\kappa, \mathsf{aux}_{\mathcal{A}}, \left\{ f_k,\ f_k(\mathbf{x}_1^{j_1}, \ldots \mathbf{x}_n^{j_n})\right\}_{k\in[q_0],j_1\in[q_1],\ldots,j_n\in[q_n]}, \left\{\mathsf{SKE.ct}_i^{j_i} \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.sk}, \mathbf{x}_i^{j_i})\right\}_{i\in[n-1],j_i\in[q_i]}\right) \\ \approx_c &\left(1^\kappa, \mathsf{aux}_{\mathcal{A}}, \left\{ f_k,\ \Delta_k^{j_1,\ldots,j_n}\right\}_{k\in[q_0],j_1\in[q_1],\ldots,j_n\in[q_n]}, \left\{\mathsf{SKE.ct}_i^{j_i} \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.sk}, \mathbf{x}_i^{j_i})\right\}_{i\in[n-1],j_i\in[q_i]}\right), \end{aligned} \tag{39}$$

where $(\mathsf{aux}_{\mathcal{A}}, \{f_k\}_k, \{\mathbf{x}_i^{j_i}\}_{i,j_i}) \leftarrow \mathsf{Samp}(1^\lambda)$, $\mathsf{SKE.sk} \leftarrow \mathsf{SKE.Setup}(1^\Lambda)$, and $\Delta_k^{j_1,\ldots,j_n} \leftarrow \{0,1\}$. To prove this, we observe

$$\begin{aligned} &\left(1^\kappa, \mathsf{aux}_{\mathcal{A}}, \left\{ f_k,\ f_k(\mathbf{x}_1^{j_1}, \ldots, \mathbf{x}_n^{j_n})\right\}_{k,j_1,\ldots,j_n}, \left\{\mathsf{SKE.ct}_i^{j_i} \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.sk}, \mathbf{x}_i^{j_i})\right\}_{i,j_i}\right) \\ \approx_c &\left(1^\kappa, \mathsf{aux}_{\mathcal{A}}, \left\{ f_k,\ f_k(\mathbf{x}_1^{j_1}, \ldots, \mathbf{x}_n^{j_n})\right\}_{k,j_1,\ldots,j_n}, \left\{\gamma_i^{j_i} \leftarrow \mathcal{CT}_{\mathsf{SKE}}\right\}_{i,j_i}\right) && (40) \\ \approx_c &\left(1^\kappa, \mathsf{aux}_{\mathcal{A}}, \left\{ f_k,\ \Delta_k^{j_1,\ldots,j_n}\right\}_{k,j_1,\ldots,j_n}, \left\{\gamma_i^{j_i} \leftarrow \mathcal{CT}_{\mathsf{SKE}}\right\}_{i,j_i}\right) && (41) \\ \approx_c &\left(1^\kappa, \mathsf{aux}_{\mathcal{A}}, \left\{ f_k,\ \Delta_k^{j_1,\ldots,j_n}\right\}_{k,j_1,\ldots,j_n}, \left\{\mathsf{SKE.ct}_i^{j_i} \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.sk}, \mathbf{x}_i^{j_i})\right\}_{i,j_i}\right). && (42) \end{aligned}$$

Here, we justify each step of the equations above. We can see that Equation (40) follows from subexponential INDr security of SKE, since SKE.sk is used only for computing $\{\mathsf{SKE.ct}_i^{j_i}\}_{i,j_i}$ and not used anywhere else. Note that by our choice of the parameter $2^{\Lambda^\delta} > \kappa^{\omega(1)}$, we can use the security of SKE even for an adversary who runs in time polynomial in κ. We can see that Equation (41) follows from Equation (37) by noting that adding random strings does not make the task of disginguishing the two distributions any easier. Finally, Equation (42) follows from INDr security of SKE again. We then consider a sampler Samp_1 that on input 1^κ outputs

$$\left(f_1, \ldots, f_{q_0}, \{(\mathbf{x}_1^{j_1}, \ldots, \mathbf{x}_n^{j_n})\}_{j_1\in[q_1],\ldots,j_n\in[q_n]}, \mathsf{aux}_1 \overset{\text{def}}{=} \left(1^\kappa, \mathsf{aux}_{\mathcal{A}}, \{\mathsf{SKE.ct}_i^{j_i}\}_{i\in[n-1],j_i\in[q_i]}\right)\right).$$

By the security guarantee of prFE_1 with sampler Samp_1 and Equation (39), we obtain Equation (38). □

Step 2. To prove that Equation (38) implies Equation (36), we prove the following statement.

Lemma 7.10. For $h \in [n]$ and an adversary $\mathcal{A}$, let us consider the following distinguishing advantage:

$$\mathsf{Adv}^{h}_{\mathcal{A}}(\lambda) \overset{\text{def}}{=} \left| \mathcal{A}\begin{pmatrix} \mathsf{aux}_{\mathcal{A}},\ \{\mathsf{prFE.mpk}_i\}_{i\in[h]},\ \{\mathsf{SKE.ct}_i^{j_i}, \mathbf{r}_i^{j_i}\}_{i\in[n-1],j_i\in[q_i]} \\ \{f_k, \mathsf{sk}_{f_k} = \mathsf{prFE}_1.\mathsf{sk}_{f_k}\}_{k\in[q_0]}, \\ \{\mathsf{prFE}_{i+1}.\mathsf{sk}^{j_i}\}_{i\in[h-1],j_i\in[q_i]},\ \{\mathsf{prFE}_h.\mathsf{ct}^{\mathbf{j}}\}_{\mathbf{j}\in[q_h]\times\cdots\times[q_n]}, \end{pmatrix} - \mathcal{A}\begin{pmatrix} \mathsf{aux}_{\mathcal{A}},\ \{\mathsf{prFE.mpk}_i\}_{i\in[h]},\ \{\mathsf{SKE.ct}_i^{j_i}, \mathbf{r}_i^{j_i}\}_{i\in[n-1],j_i\in[q_i]} \\ \{f_k, \mathsf{sk}_{f_k} = \mathsf{prFE}_1.\mathsf{sk}_{f_k}\}_{k\in[q_0]}, \\ \{\mathsf{prFE}_{i+1}.\mathsf{sk}^{j_i}\}_{i\in[h-1],j_i\in[q_i]},\ \{\Delta^{\mathbf{j}}\}_{\mathbf{j}\in[q_h]\times\cdots\times[q_n]} \end{pmatrix} \right| \quad (43)$$

where $\mathbf{j} = (j_h, \ldots, j_n)$, $\Delta^{\mathbf{j}} \leftarrow \mathcal{CT}_{\mathsf{prFE}_h}$, $\mathsf{prFE}_{i+1}.\mathsf{sk}^{j_i} \leftarrow \mathsf{prFE}_{i+1}.\mathsf{KeyGen}(\mathsf{prFE}_{i+1}.\mathsf{msk}, \mathrm{F}_i[\mathsf{SKE.ct}_i^{j_i}, \mathbf{r}_i, \mathsf{prFE}_i.\mathsf{mpk}])$, and

$$\mathsf{prFE}_h.\mathsf{ct}^{\mathbf{j}} \leftarrow \mathsf{prFE}_h.\mathsf{Enc}\left(\mathsf{prFE}_h.\mathsf{mpk}, \left(\mathsf{SKE.sk}, (\mathbf{x}_h^{j_h}, \mathbf{r}_h^{j_h}), \ldots, (\mathbf{x}_{n-1}^{j_{n-1}}, \mathbf{r}_{n-1}^{j_{n-1}}), (\mathbf{x}_n^{j_n}, K_1^{j_n}, \ldots, K_{h-1}^{j_n})\right)\right).$$

Then, for every $h^* := \{h^*_\lambda \in [2, n(\lambda)]\}_\lambda$ and every non-uniform adversary $\mathcal{A} = \{\mathcal{A}_\lambda\}_\lambda$ such that $\mathsf{Size}(\mathcal{A}) < \mathrm{poly}(\kappa)$,[12] there exists another non-uniform adversary $\mathcal{B} = \{\mathcal{B}_\lambda\}_\lambda$ and a polynomial Q such that

$$\mathsf{Adv}^{h^*-1}_{\mathcal{B}}(\lambda) \geq \mathsf{Adv}^{h^*}_{\mathcal{A}}(\lambda)/Q(\lambda') - \mathsf{negl}(\kappa) \qquad \text{and} \qquad \mathsf{Size}(\mathcal{B}) \leq Q(\lambda')\cdot\mathsf{Size}(\mathcal{A})$$

assuming the security of prFE as per Definition B.1 with respect to κ and the subexponential security of PRF, where $\lambda' := \lambda^n$.

Proof. We invoke the security of prFE_{h^*} with non-uniform sampler Samp_{h^*} that takes as input the security parameter 1^λ and outputs

$$\begin{pmatrix} \text{Functions:} & \left\{\mathrm{F}_{h^*-1}^{j_{h^*-1}} = \mathrm{F}_{h^*-1}^{j_{h^*-1}}[\mathsf{SKE.ct}_{h^*-1}^{j_{h^*-1}}, \mathbf{r}_{h^*-1}^{j_{h^*-1}}, \mathsf{prFE}_{h^*-1}.\mathsf{mpk}]\right\}_{j_{h^*-1}\in[q_{h^*-1}]}, \\ \text{Inputs:} & \left\{\mathbf{x}^{j_{h^*},\ldots,j_n} \overset{\text{def}}{=} \left(\mathsf{SKE.sk}, (\mathbf{x}_{h^*}^{j_{h^*}}, \mathbf{r}_{h^*}^{j_{h^*}}), \ldots (\mathbf{x}_n^{j_n}, K_1^{j_n}, \ldots, K_{h^*-1}^{j_n})\right)\right\}_{j_{h^*}\in[q_{h^*}],\ldots,j_n\in[q_n]} \\ \text{Auxiliary Information:} & \mathsf{aux}_{h^*} \overset{\text{def}}{=} \begin{pmatrix} \mathsf{aux}_{\mathcal{A}}, \{\mathsf{prFE}_i.\mathsf{mpk}\}_{i\in[h^*-1]}, \{f_k, \mathsf{sk}_{f_k} = \mathsf{prFE}_1.\mathsf{sk}_{f_k}\}_{k\in[q_0]}, \\ \{\mathsf{SKE.ct}_i^{j_i}, \mathbf{r}_i^{j_i}\}_{i\in[n-1],j_i\in[q_i]}, \{\mathsf{prFE}_{i+1}.\mathsf{sk}^{j_i}\}_{i\in[h^*-2],j_i\in[q_i]} \end{pmatrix} \end{pmatrix}. \quad (44)$$

We can see that the size of Samp is $\mathrm{poly}(\lambda^n) = \mathrm{poly}(\lambda')$, since $q_0 q_1 \ldots q_n = \mathrm{poly}(\lambda^n)$. We also consider the following distributions:

$$\left(\mathsf{prFE}_{h^*}.\mathsf{mpk}, \left\{\mathrm{F}_{h^*-1}^{j_{h^*-1}}, \mathsf{prFE}_{h^*}.\mathsf{sk}^{j_{h^*-1}}\right\}_{j_{h^*-1}}, \left\{\mathsf{prFE}_{h^*}.\mathsf{Enc}(\mathbf{x}^{j_{h^*},\ldots,j_n})\right\}_{j_{h^*},\ldots,j_n}, \mathsf{aux}_{h^*}\right)$$
$$\text{and} \quad \left(\mathsf{prFE}_{h^*}.\mathsf{mpk}, \left\{\mathrm{F}_{h^*-1}^{j_{h^*-1}}, \mathsf{prFE}_{h^*}.\mathsf{sk}^{j_{h^*-1}}\right\}_{j_{h^*-1}}, \left\{\Delta^{j_{h^*},\ldots,j_n} \leftarrow \mathcal{CT}_{\mathsf{prFE}_{h^*}}\right\}_{j_{h^*},\ldots,j_n}, \mathsf{aux}_{h^*}\right). \quad (45)$$

By the security guarantee of prFE_{h^*} with respect to Samp_{h^*}, we can see that an adversary $\mathcal{A}$ that can distinguish the distributions in Equation (45) with advantage more than ϵ can be converted into another adversary $\mathcal{B}$ that can distinguish the following distributions with advantage more than $\epsilon' \overset{\text{def}}{=} \epsilon/Q(\lambda') - \mathsf{negl}(\kappa)$ satisfying $\mathsf{Size}(\mathcal{B}) \leq Q(\lambda')\mathsf{Size}(\mathcal{A})$ for some polynomial Q:

$$\left(\left\{\mathrm{F}_{h^*-1}^{j_{h^*-1}}\right\}_{j_{h^*-1}}, \left\{\mathrm{F}_{h^*-1}^{j_{h^*-1}}(\mathbf{x}^{j_{h^*},\ldots,j_n})\right\}_{j_{h^*-1},\ldots,j_n}, \mathsf{aux}_{h^*}\right)$$
$$\text{and} \quad \left(\left\{\mathrm{F}_{h^*-1}^{j_{h^*-1}}\right\}_{j_{h^*-1}}, \left\{\Delta^{j_{h^*-1},\ldots,j_n} \leftarrow \mathcal{CT}_{\mathsf{prFE}_{h^*}}\right\}_{j_{h^*-1},\ldots,j_n}, \mathsf{aux}_{h^*}\right). \quad (46)$$

[12]Here, we deviate from our convention that the adversary runs in polynomial time in its input length. Note that κ here may be super-polynomial in the input length to $\mathcal{A}$.

By inspection, one can see that the distributions in Equation (45) are equivalent to those in Equation (43) with $h = h^*$.[13] Therefore, to complete the proof, it suffices to show that $\mathcal{B}$ can be used as a distinguisher against the distributions in Equation (43) with $h = h^* - 1$ whose advantage is at least $\epsilon' - \mathsf{negl}(\kappa)$. To show this, we first observe that the second distribution in Equation (46) is equivalent to that in Equation (43) with $h = h^* - 1$. Therefore, it suffices to prove that $\mathcal{A}$ cannot distinguish the first distribution in Equation (46) from the first distribution in Equation (43) with $h = h^* - 1$ with more than negligible advantage in κ. To show this, we consider the following sequence of hybrids.

Hyb_1. This is the first distribution of Equation (46). Recall that we have

$$\mathsf{F}^{j_{h^*-1}}_{h^*-1}(\mathbf{x}^{j_{h^*},\ldots,j_n}) = \mathsf{prFE}_{h^*-1}.\mathsf{Enc}\left(\begin{array}{r}\mathsf{prFE}_{h^*-1}.\mathsf{mpk}, \left(\mathsf{SKE.sk}, (\mathbf{x}^{j_{h^*-1}}_{h^*-1}, \mathbf{r}^{j_{h^*-1}}_{h^*-1}), \ldots (\mathbf{x}^{j_n}_n, K^{j_n}_1, \ldots, K^{j_n}_{h^*-2})\right); \\ \mathsf{PRF}_{h^*-1}(K^{j_n}_{h^*-1}, (\mathbf{r}^{j_{h^*-1}}_{h^*-1}, \ldots, \mathbf{r}^{j_{n-1}}_{n-1}))\end{array}\right).$$

Hyb_2. This hybrid is the same as the previous one except that we replace $\mathsf{PRF}_{h^*-1}(K^{j_n}_{h^*-1}, \cdot)$ with the real random function $\mathsf{R}^{j_n}(\cdot)$ for each $j_n \in [q_{\mathsf{msg}}]$. Since $K^{j_n}_{h^*-1}$ is not used anywhere else, we can use the subexponential security of PRF to conclude that this hybrid is computationally indistinguishable from the previous one. In particular, by our choice of the parameter $2^{\Lambda^\delta} > \kappa^{\omega(1)}$, we can conclude that the adversary cannot distinguish this hybrid from the previous one with advantage more than $\mathsf{negl}(\kappa)$.

Hyb_3. This hybrid is the same as the previous one except that we output a failure symbol when there exist $(j_{h^*-1}, \ldots, j_{n-1}) \neq (j'_{h^*-1}, \ldots, j'_{n-1})$ such that $(\mathbf{r}^{j_{h^*-1}}_{h^*-1}, \ldots, \mathbf{r}^{j_{n-1}}_{n-1}) = (\mathbf{r}^{j'_{h^*-1}}_{h^*-1}, \ldots, \mathbf{r}^{j'_{n-1}}_{n-1})$. We show that the probability of this happening is negligible in κ. To prove this, it suffices to show that there are no $i \in [h^* - 1, n - 1]$, $j, j' \in [q_i]$ satisfying $j \neq j'$ and $\mathbf{r}^j_i = \mathbf{r}^{j'}_i$. The probability of this happening can be bounded by $(q_1^2 + \cdots q_{n-1}^2)/2^{2\Lambda}$ by taking the union bound with respect to all the combinations of i, j, j'. By our choice of Λ, this is bounded by $\mathsf{negl}(\kappa)$.

Hyb_4. In this hybrid, we replace $\mathsf{F}^{j_{h^*-1}}_{h^*-1}(\mathbf{x}^{j_{h^*},\ldots,j_n})$ with

$$\mathsf{prFE}_{h^*-1}.\mathsf{ct}^{j_{h^*-1},\ldots,j_n} = \mathsf{prFE}_{h^*-1}.\mathsf{Enc}\left(\mathsf{prFE}_{h^*-1}.\mathsf{mpk}, \left(\mathsf{SKE.sk}, (\mathbf{x}^{j_{h^*-1}}_{h^*-1}, \mathbf{r}^{j_{h^*-1}}_{h^*-1}), \ldots (\mathbf{x}^{j_n}_n, K^{j_n}_1, \ldots, K^{j_n}_{h^*-2})\right)\right).$$

Namely, we use fresh randomness for each encryption instead of deriving the randomness by $\mathsf{R}^{j_n}(\mathbf{r}^{j_{h^*-1}}_{h^*-1}, \ldots, \mathbf{r}^{j_{n-1}}_{n-1})$. We claim that this change is only conceptual. To see this, we observe that unless the failure condition introduced in Hyb_3 is satisfied, every invocation of the function R^{j_n} is with respect to a fresh input and thus the output can be replaced with a fresh randomness.

Hyb_5. In this hybrid, we remove the failure event. Namely, we always outputs $\mathsf{prFE}_{h^*-1}.\mathsf{ct}_{j_{h^*-1},\ldots,j_n}$ regardless of whether the failure event happens or not. Since the failure event happens with probability only $\mathsf{negl}(\kappa)$ probability, we conclude that the adversary is not able to distinguish this hybrid from the previous one with more than $\mathsf{negl}(\kappa)$ probability.

Noting that the final hybrid is equivalent to the first distribution in Equation (43) with $h = h^* - 1$, we complete the proof. $\square$

Lemma 7.11. Assuming that $\mathsf{Adv}^1_{\mathcal{A}}(\lambda)$ (defined in Equation (43)) is negligible in κ for all non-uniform adversary $\mathcal{A}$ such that $\mathsf{Size}(\mathcal{A}) = \mathrm{poly}(\kappa)$, we have $\mathsf{Adv}^n_{\mathcal{B}}(\lambda) = \mathsf{negl}(\lambda)$ for all non-uniform adversary $\mathcal{B}$ such that $\mathsf{Size}(\mathcal{B}) = \mathrm{poly}(\lambda)$.

Proof. For the sake of contradiction, suppose that there exists an adversary $\mathcal{B} = \{\mathcal{B}_\lambda\}_\lambda$ such that $\epsilon_n(\lambda) \overset{\text{def}}{=} \mathsf{Adv}^n_{\mathcal{B}}(\lambda)$ is non-negligible and $t_n(\lambda) \overset{\text{def}}{=} \mathsf{Size}(\mathcal{B})$ is polynomial in λ. In particular, this implies that there exists an infinite

[13]Equation (45) includes additional terms $\{\mathsf{F}^{j_{h^*-1}}_{h^*-1}\}_{j_{h^*-1}}$ while Equation (43) does not. We ignore this difference, since theses terms can be efficiently computed from aux_{h^*} and does not affect the indistinguishability.

set $\mathcal{L} \subseteq \mathbb{N}$ and some polynomial p such that $\epsilon_n(\lambda) \geq 1/p(\lambda)$ and $t_n(\lambda) \leq p(\lambda)$ for all $\lambda \in \mathcal{L}$. We then define $t_i(\lambda) \stackrel{\text{def}}{=} p(\lambda)\lambda^{n(n-i)\log\lambda}$ and $\epsilon_i(\lambda) \stackrel{\text{def}}{=} 1/p(\lambda)\lambda^{n(n-i)\log\lambda}$ for $i = 1, \ldots, n-1$. We can also see that $t_1(\lambda) < \kappa$ and $\epsilon_1(\lambda) > 1/\kappa$ for sufficiently large λ. This implies that there exists $\lambda_0 \in \mathbb{N}$ such that for all $\lambda > \lambda_0$, there is no adversary $\mathcal{A}_\lambda$ such that $\mathsf{Size}(\mathcal{A}_\lambda) \leq t_1(\lambda)$ and $\mathsf{Adv}^1_{\mathcal{A}_\lambda}(\lambda) \geq \epsilon_1(\lambda)$. We then consider the following statement that is parameterized by λ and $h \in [n]$:

$$\mathsf{Statement}_{\lambda,h}\text{: There exists an adversary } \mathcal{A}_\lambda \text{ such that } \mathsf{Size}(\mathcal{A}_\lambda) \leq t_h(\lambda) \text{ and } \mathsf{Adv}^h_{\mathcal{A}_\lambda}(\lambda) \geq \epsilon_h(\lambda).$$

For each $\lambda \in \mathcal{L} \cap \mathbb{N}_{>\lambda_0}$, there exists $h^*_\lambda \in [2, n]$ such that $\mathsf{Statement}_{\lambda,h^*_\lambda-1}$ is false and $\mathsf{Statement}_{\lambda,h^*_\lambda}$ is true, since $\mathsf{Statement}_{\lambda,1}$ is false and $\mathsf{Statement}_{\lambda,n}$ is true. However, by applying Lemma 7.10 to the adversary guaranteed by $\mathsf{Statement}_{\lambda,h^*_\lambda}$ being true for the sequence $\{h^*_\lambda\}_\lambda$, we obtain another adversary $\mathcal{A}' = \{\mathcal{A}'_\lambda\}_\lambda$ such that $\mathsf{Size}(\mathcal{A}'_\lambda) \leq t_{h^*}Q(\lambda^n)$ and $\mathsf{Adv}^{h^*-1}_{\mathcal{A}'_\lambda}(\lambda) \geq \epsilon_{h^*}/Q(\lambda^n) - \mathsf{negl}(\kappa) \geq \epsilon_{h^*}/2Q(\lambda^n)$ for some polynomial Q. In particular, $\mathsf{Size}(\mathcal{A}'_\lambda) \leq t_{h^*-1}(\lambda)$ and $\mathsf{Adv}^{h^*-1}_{\mathcal{A}'_\lambda}(\lambda) \geq \epsilon_{h^*-1}(\lambda)$ for all sufficiently large λ, since we have $\lambda^{n\log\lambda} > Q(\lambda^n)$ for all sufficiently large λ. However, this contradicts the above assertion that $\mathsf{Statement}_{\lambda,h^*_\lambda-1}$ is false. This concludes the proof. $\square$

The following lemma completes Step 2 of the proof of Theorem 7.8.

Lemma 7.12. If SKE is INDr secure, Equation (38) implies Equation (36).

Proof. We first observe that Equation (38) is equivalent to saying $\mathsf{Adv}^1_{\mathcal{A}}(\lambda) = \mathsf{negl}(\kappa)$ for all $\mathcal{A}$ with $\mathsf{Size}(\mathcal{A}) = \mathrm{poly}(\kappa)$. By Lemma 7.11, this implies that $\mathsf{Adv}^n_{\mathcal{A}}(\lambda) = \mathsf{negl}(\lambda)$ for all $\mathcal{A}$ with $\mathsf{Size}(\mathcal{A}) = \mathrm{poly}(\lambda)$. Namely, we have

$$\begin{pmatrix} \mathsf{aux}_{\mathcal{A}},\ \{\mathsf{prFE}_i.\mathsf{mpk}\}_{i\in[n]},\ \left\{f_k,\ \mathsf{sk}_{f_k} = \mathsf{prFE}_1.\mathsf{sk}_{f_k}\right\}_{k\in[q_0]} \\ \left\{\mathsf{SKE}.\mathsf{ct}^{j_i}_i,\ \mathbf{r}^{j_i}_i,\ \mathsf{prFE}_{i+1}.\mathsf{sk}^{j_i}\right\}_{\substack{i\in[n-1],\\ j_i\in[q_i]}},\ \left\{\mathsf{prFE}_n.\mathsf{ct}^{j_n}\right\}_{j_n\in[q_n]} \end{pmatrix} \tag{47}$$

$$\approx_c \begin{pmatrix} \mathsf{aux}_{\mathcal{A}},\ \{\mathsf{prFE}_i.\mathsf{mpk}\}_{i\in[n]},\ \left\{f_k,\ \mathsf{sk}_{f_k} = \mathsf{prFE}_1.\mathsf{sk}_{f_k}\right\}_{k\in[q_0]} \\ \left\{\mathsf{SKE}.\mathsf{ct}^{j_i}_i,\ \mathbf{r}^{j_i}_i,\ \mathsf{prFE}_{i+1}.\mathsf{sk}^{j_i}\right\}_{\substack{i\in[n-1],\\ j_i\in[q_i]}},\ \left\{\delta^{j_n}_n\right\}_{j_n\in[q_n]} \end{pmatrix}.$$

By INDr security of SKE, we have

$$\begin{pmatrix} \mathsf{aux}_{\mathcal{A}},\ \{\mathsf{prFE}_i.\mathsf{mpk}\}_{i\in[n]},\ \left\{f_k,\ \mathsf{sk}_{f_k} = \mathsf{prFE}_1.\mathsf{sk}_{f_k}\right\}_{k\in[q_0]} \\ \left\{\mathsf{SKE}.\mathsf{ct}^{j_i}_i,\ \mathbf{r}^{j_i}_i,\ \mathsf{prFE}_{i+1}.\mathsf{sk}^{j_i}\right\}_{\substack{i\in[n-1],\\ j_i\in[q_i]}},\ \left\{\delta^{j_n}_n\right\}_{j_n\in[q_n]} \end{pmatrix} \tag{48}$$

$$\approx_c \begin{pmatrix} \mathsf{aux}_{\mathcal{A}},\ \{\mathsf{prFE}_i.\mathsf{mpk}\}_{i\in[n]},\ \left\{f_k,\ \mathsf{sk}_{f_k} = \mathsf{prFE}_1.\mathsf{sk}_{f_k}\right\}_{k\in[q_0]} \\ \left\{\gamma^{j_i}_i,\ \mathbf{r}^{j_i}_i,\ \mathsf{prFE}_{i+1}.\mathsf{sk}^{j_i}\right\}_{\substack{i\in[n-1],\\ j_i\in[q_i]}},\ \left\{\delta^{j_n}_n\right\}_{j_n\in[q_n]} \end{pmatrix}.$$

Combining the above equations, Equation (36) readily follows. $\square$

The above concludes the proof of the former part of Theorem 7.8. We then move to the proof of the latter part of the theorem (i.e., (κ, ϵ)-security). The proof for the latter part is almost the same as the former part. In more detail, Step 1 of the proof is exactly the same. In Step 2, we prove that Equation (38) implies that the two distributions in Equation (36) is not distinguishable for a PPT adversary with advantage more than $\lambda^{-n\log\lambda/2}$. For doing so, we strengthen Lemma 7.11 so that we have $\mathsf{Adv}^n_{\mathcal{B}}(\lambda) = \lambda^{-2n\log\lambda/3}$ instead of $\mathsf{Adv}^n_{\mathcal{B}}(\lambda) = \mathsf{negl}(\lambda)$. This is proven in almost the same manner by setting $p(\lambda)$ to be $p(\lambda) = \lambda^{2n\log\lambda/3}$. Then, we have that the distributions in Equation (47) are not distinguishable for any PPT adversary with advantage more than $\lambda^{-2n\log\lambda/3}$. By sub-exponential INDr security of SKE, we have that the distributions in Equation (48) are not distinguishable for any PPT adversary with advantage more than $\mathsf{negl}(\kappa)$. Combining these, we the two distributions in Equation (36) are not distinguishable for any PPT adversary with advantage more than $\lambda^{-n\log\lambda/2}$, since we have $\mathsf{negl}(\kappa) + \lambda^{-2n\log\lambda/3} < \lambda^{-n\log\lambda/2}$. This completes the proof of the theorem. $\square$

Remark 7.13 (Comparison with [VWW22]). We note that in high level, overall structure of our security proof above is similar to that of witness encryption in [VWW22]. In both proofs, the main step considers parameterized distributions $\{\mathcal{D}_{h,b}\}_{h\in[n],b\in\{0,1\}}$ and shows that $\mathcal{D}_{1,0} \approx_c \mathcal{D}_{1,1}$ holds if $\mathcal{D}_{n,0}$ and $\mathcal{D}_{n,1}$ are indistinguishable even with subexponentially small advantage against subexponential time adversary. To show this claim, [VWW22] uses the evasive LWE assumption, while we use the security of prFE, which in turn is reduced to the evasive LWE assumption. While this difference stems simply from the fact that we introduce the intermediate primitive of prFE to construct prMIFE instead of directly constructing it from evasive LWE, there are more fundamental differences as well. In particular, we identify certain subtle issues in the proof by [VWW22] and fix these by strengthening the assumptions. We elaborate on this in the following.

On the multiplicative invocation of evasive LWE. To prove $\mathcal{D}_{1,0} \approx_c \mathcal{D}_{1,1}$, [VWW22] assumes that there exists an adversary $\mathcal{A}_1$ that distinguishes them with non-negligible advantage ϵ and polynomial time t for the sake of contradiction. They then invoke the evasive LWE assumption with respect to an appropriately defined sampler Samp_1 to conclude that there exists a distinguishing adversary $\mathcal{A}_2$ against $\mathcal{D}_{2,0}$ and $\mathcal{D}_{2,1}$. This process continues multiple times, where they invoke evasive LWE with respect to the security parameter $\lambda_j := 2^j\lambda$ and an adversary $\mathcal{A}_j$ for the j-th invocation to obtain another adversary $\mathcal{A}_{j+1}$, where $\mathcal{A}_k$ is a distinguisher against $\mathcal{D}_{k,0}$ and $\mathcal{D}_{k,1}$. Denoting the distinguishing advantage against $\mathcal{D}_{j,0}$ and $\mathcal{D}_{j,1}$ of $\mathcal{A}_j$ by ϵ_j, we have $\epsilon_{j+1} \geq \epsilon_j/\mathrm{poly}_j(\lambda_j)$, where poly_j is a polynomial that is determined by the sampler Samp_j. Finally, they obtain a distinguisher $\mathcal{A}_n$ against $\mathcal{D}_{n,0}$ and $\mathcal{D}_{n,1}$, where $\epsilon_n = \epsilon/\mathrm{poly}_1(\lambda_1)\mathrm{poly}_2(\lambda_2)\cdots\mathrm{poly}_n(\lambda_n)$ and the running time being $\mathrm{poly}_1(\lambda_1)\mathrm{poly}_2(\lambda_2)\cdots\mathrm{poly}_n(\lambda_n)$. They derive the conclusion by saying

$$\mathrm{poly}_1(\lambda_1)\mathrm{poly}_2(\lambda_2)\cdots\mathrm{poly}_n(\lambda_n) = \mathrm{poly}(\lambda_1\cdots\lambda_n) = \mathrm{poly}(2^{n^2}\lambda^n) \tag{49}$$

and setting the parameter so that there is no adversary of this running time and distinguishing advantage against $\mathcal{D}_{n,0}$ and $\mathcal{D}_{n,1}$. However, a subtlety is that $\mathrm{poly}_1(\lambda_1)\mathrm{poly}_2(\lambda_2)\cdots\mathrm{poly}_n(\lambda_n) = \mathrm{poly}(\lambda_1\cdots\lambda_n)$ is not necessarily true. For example, one can consider the setting where we have $\mathrm{poly}_j(\lambda) = \lambda^{2^j}$. This example may look a bit artificial, but it does not contradict the evasive LWE assumption, since j is treated as a constant asymptotically. In words, the issue arises from the fact that even if each polynomial has a constant exponent, the maximum of the exponents can be arbitrarily large function in λ, when we consider non-constant number of polynomials. In this setting, $\mathcal{A}_n$'s distinguishing advantage is too small to derive the contradiction.

The above issue occurs due to the invocations of evasive LWE super-constant times. To resolve the problem, we consider non-uniform sampler $\{\mathsf{Samp}_{h^*}\}_{h^*}$ that hardwires the "best" index h^* and invoke the evasive LWE only with respect to this sampler (See Lemma 7.10 and 7.11). We avoid the above problem, since we invoke the evasive LWE only once in the proof. However, this solution entails the strengthening of the assumption where we consider non-uniform samplers. We believe that the same strengthening of the assumption is required for the proof in [VWW22] as well.

On the additive term of evasive LWE. Here, we also discuss the other subtlety that arises in the proof by [VWW22]. To focus on the issue, we ignore the first issue discussed above and assume $\mathrm{poly}_j(\lambda) = \lambda^c$ holds for some fixed $c \in \mathbb{N}$ that does not depend on j, which makes Equation (49) correct. In the proof of [VWW22] (and in our explanation above), we implicitly ignore the negligible additive term when applying evasive LWE. Namely, when we apply the assumption with respect to $\mathcal{A}_j$, the lower bound for the advantage ϵ_{j+1} of $\mathcal{A}_{j+1}$ should be $\epsilon_{j+1} \geq \epsilon_j/\mathrm{poly}(\lambda_j) - \mathsf{negl}(\lambda_j)$ rather than $\epsilon_{j+1} \geq \epsilon_j/\mathrm{poly}(\lambda_j)$. This does not cause any difference when we consider the setting where ϵ_j is non-negligible in λ_j. However, for larger j, ϵ_j is negligible function in the security parameter λ_j. Concretely, the lower bound on ϵ_{n-1} obtained by ignoring the additive term is $\epsilon/\mathrm{poly}(2^{(n-1)^2}\lambda^{n-1})$.[14] If we apply evasive LWE once more with respect to $\lambda_n = 2^n\lambda^n$ to complete the proof,

[14]Namely, the actual value of ϵ_{n-1} may be even smaller.

we have $\epsilon_n \geq \epsilon_{n-1}/\mathrm{poly}(\lambda_n) - \mathsf{negl}(\lambda_n)$. The RHS of the inequality may be negative, since $\epsilon_{n-1}/\mathrm{poly}(\lambda_n)$ is some specific negligible function in λ_n and this may be smaller than the second term $\mathsf{negl}(\lambda_n)$. Therefore, what we can derive here is the trivial bound $\epsilon_n \geq 0$, which is not enough for our purpose. To fix this issue, we introduce additional parameter κ and then modify the assumption so that the additive term is negligibly small in κ, which is set much larger than λ. Again, we believe that the same strengthening of the assumption is required for the proof in [VWW22] as well. Finally, we note that the above problem does not occur when n is constant. This is because in that case, we have ϵ_n is non-negligible and thus the problem of ϵ_n may be smaller than negl_n does not occur.

7.4 Security Proof for Constant n (with Weaker Assumption)

Here, we prove the security of our construction in the case of n being a constant. The reason why we consider the security proof separately for this special case is that we can give a proof from better assumptions than the general case. In more detail, the security is proven assuming the standard security notion for prFE, rather than the non-uniform and κ version of it. As a result, the security of the prMIFE is reduced to the the (plain) evasive LWE instead of non-uniform κ-evasive LWE. The reason why we can achieve this is that in the case of n being constant, we can avoid all the subtleties that arise in the general case. We refer to Remark 7.13 for more discussions.

Theorem 7.14. Let $\mathcal{SC}_{\mathsf{prMIFE}}$ be a sampler class for prMIFE. Suppose prFE_i scheme satisfies prCT security as per Definition 4.2 with respect to the sampler class that contains all $\mathsf{Samp}_{\mathsf{prFE}}(1^\lambda)$, induced by $\mathsf{Samp}_{\mathsf{prMIFE}} \in \mathcal{SC}_{\mathsf{prMIFE}}$, as in Equation (44), SKE satisfies INDr security and PRF_i is secure, then prMIFE constructed in Section 7.2 for constant n satisfies security for $\kappa = \lambda^n$ as in Definition 7.2.

Proof. The proof of this theorem largely follows that of Theorem 7.8. The crucial difference is that to get the equivalent of Lemma 7.11, we simply invoke the (non-κ, uniform) security of prFE n-times. We sketch the proof below while highlighting the difference. We consider the same simulator as in the proof of Theorem 7.8 and divide the proof steps into Step 1 and Step 2 in the same manner.

- We start with Step 1, which consists of proving that Equation (37) implies Equation (38). This is proven in the same manner as Lemma 7.9. However here, since we set $\kappa = \lambda^n = \mathrm{poly}(\lambda)$, we do not need subexponential INDr security and only (polynomial) INDr security suffices for SKE.
- We then move to prove Step 2. The goal here is to prove Equation (38) implies Equation (36).
  - We first observe that the uniform version of Lemma 7.10 holds by the same proof, where we only consider constant h^* rather than arbitrary sequence $h^* = \{h^*_\lambda\}_\lambda$ and uniform PPT adversaries. Notice that then the sampler is now uniform, since it no longer has to hardwire the sequence $\{h^*_\lambda\}_\lambda$. In this setting, non-uniform κ-prCT security collapses to the (plain) prCT security, since $\kappa = \lambda^n = \mathrm{poly}(\lambda)$ and the sampler is uniform. Therefore, plain prCT security is sufficient for the proof. Furthermore, since we set $\kappa = \lambda^n = \mathrm{poly}(\lambda)$, we do not need subexponential security for PRF and standard security suffices.
  - We then consider an analogue of Lemma 7.11, which asserts that if $\mathsf{Adv}^1_{\mathcal{A}}(\lambda)$ (defined in Equation (43)) is negligible for all (uniform) PPT adversary that runs in polynomial time in λ, then so is $\mathsf{Adv}^n_{\mathcal{A}}(\lambda)$. This is proven by observing that the indistinguishability of the distributions in Equation (43) for $h = h^* - 1$ implies that for h^* by the analogue of Lemma 7.10 explained in the previous item. By applying this n-times, we obtain the conclusion.
  - We finally conclude the proof by the same argument as Lemma 7.12.

This completes the proof of Theorem 7.14. □

8 Multi-Input Predicate Encryption for Polynomial Arity for P

In this section, we provide our construction of multi-input predicate encryption (miPE) for all circuits as an application of prMIFE. Similarly to Section 7, we consider two settings where the arity is either constant or arbitrary polynomial.

8.1 Construction

In this section, we give a construction of a miPE scheme using prMIFE and PE. The construction will support functions with arity $n = n(\lambda)$ where input string for each arity is in $\{0,1\}^L$ and the output is $\{0,1\}$. We further restrict the depth of the circuits that implement the function by a parameter dep. We denote this function class by $\mathcal{F}_{\mathsf{prm}}$, where prm is the set of the parameters n, L, dep. Namely, $\mathcal{F}_{\mathsf{prm}}$ consists of n-ary functions that takes as input strings $x_1, \ldots, x_n$, where each $x_i \in \{0,1\}^L$ and outputs $f(x_1, \ldots, x_n) \in \{0,1\}$. The message space for each slot is $\{0,1\}$. Namely, we have $\mu_1, \ldots, \mu_n \in \{0,1\}$ in the following.

Building Blocks. Below, we list the building blocks required for our construction.

1. A single input predicate encryption scheme $\mathsf{PE} = \mathsf{PE}.(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ for function family supporting functions $f : \{0,1\}^{nL} \to \{0,1\}$ that can be represented as circuits with depth at most dep. We denote by this function class by $\mathcal{F}_{\mathsf{prm}_{\mathsf{PE}}}$, where $\mathsf{prm}_{\mathsf{PE}} = (1^{nL}, 1^{\mathsf{dep}})$ is the parameter that specifies the circuit class. We also assume that the scheme has message space $\{0,1\}^n$ and satisfies sub-exponential INDr security (Definition 3.23). We can instantiate such a PE by using the construction by [GVW15b] or by the combination of lockable obfuscation [WZ17] (a.k.a compute-and-compare obfuscation [WZ17]) and ABE for circuits by [BGG$^+$14], for example. We will generate the PE instance with respect to possibly scaled security parameter Λ. We will discuss how to set Λ in Remark 8.1.

   We use $\mathcal{CT}_{\mathsf{PE}}$ to denote the ciphertext space, $\ell_{\mathsf{ct}}^{\mathsf{PE}}$ to denote the ciphertext length and $d_{\mathsf{Enc}}^{\mathsf{PE}}$ to denote the depth of the circuit required to compute the PE.Enc algorithm.

2. A $n+1$-input FE for pseudorandom functionalities $\mathsf{prMIFE} = \mathsf{prMIFE}.(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}_0, \mathsf{Enc}_1, \ldots, \mathsf{Enc}_n, \mathsf{Dec})$ for function family $\mathcal{F}_{L'(\lambda), d_{\mathsf{Enc}}^{\mathsf{PE}}}$ consisting of circuits with input space $\{0,1\}^{L'}$ and output space $\{0,1\}$ where we set $L' = \max\{\ell_{\mathsf{ct}}^{\mathsf{PE}} + n\lambda + \Lambda + 2, 2\ell + 1 + \lambda\}$ for our construction and with maximum depth $d_{\mathsf{Enc}}^{\mathsf{PE}}$. We denote the parameters that specify $\mathcal{F}_{L'(\lambda), d_{\mathsf{Enc}}^{\mathsf{PE}}}$ by $\mathsf{prm}_{\mathsf{prMIFE}}$. We can instantiate it by the construction we obtained in Section 7.

3. A puncturable pseudorandom function $\mathsf{PRF} : \{0,1\}^{\Lambda} \times \{0,1\}^{n\lambda} \to \{0,1\}^{\mathsf{R_{len}}}$, where $\{0,1\}^{\Lambda}$ and $\{0,1\}^{n\lambda}$ are the key space and input space respectively and $\mathsf{R_{len}}$ is the length of randomness used in the PE.Enc algorithm. We require the puncturable PRF to satsify the sub-exponential security.

Next, we describe our construction.

$\mathsf{Setup}(1^\lambda, 1^n, \mathsf{prm}) \to (\mathsf{mpk}, \mathsf{msk})$. The setup algorithm does the following.

- Generate $(\mathsf{prMIFE.mpk}, \mathsf{prMIFE.msk}) \leftarrow \mathsf{prMIFE.Setup}(1^\lambda, 1^n, \mathsf{prm}_{\mathsf{prMIFE}})$.
- Generate $(\mathsf{PE.mpk}, \mathsf{PE.msk}) \leftarrow \mathsf{PE.Setup}(1^\Lambda, \mathsf{prm}_{\mathsf{PE}})$.
- Sample $K \leftarrow \{0,1\}^\Lambda$.
- Generate $\mathsf{prMIFE.ct}_{n+1} \leftarrow \mathsf{prMIFE.Enc}_{n+1}(\mathsf{prMIFE.msk}, (K, 0, 0, \bot))$.
- Compute $\mathsf{prMIFE.sk_F} \leftarrow \mathsf{prMIFE.KeyGen}(\mathsf{prMIFE.msk}, \mathrm{F}[\mathsf{PE.mpk}])$ where $\mathrm{F}[\mathsf{PE.mpk}]$ is defined as follows:

Circuit F[PE.mpk]
Input: $(K, \mathsf{cnt}, \mathsf{flag}, \mathsf{PE.ct}^*), (\mathbf{x}_1, \mu_1, \mathbf{r}_1, \mathbf{x}'_1, \mu'_1), \ldots, (\mathbf{x}_n, \mu_n, \mathbf{r}_n, \mathbf{x}'_n, \mu'_n)$

1. Define $R \overset{\text{def}}{=} (\mathbf{r}_1, \ldots, \mathbf{r}_n)$.
2. It computes PE.ct as follows:

$$\mathsf{PE.ct} = \begin{cases} \mathsf{PE.Enc}(\mathsf{PE.mpk}, (\mathbf{x}_1, \ldots, \mathbf{x}_n), (\mu_1, \ldots, \mu_n); \mathsf{PRF}(K, R)) & \text{if } \mathsf{cnt} < R \\ \mathsf{PE.ct}^* & \text{if } (\mathsf{cnt} = R) \wedge (\mathsf{flag} = 1) \\ \mathsf{PE.Enc}(\mathsf{PE.mpk}, (\mathbf{x}'_1, \ldots, \mathbf{x}'_n), (\mu'_1, \ldots, \mu'_n); \mathsf{PRF}(K, R)) & \text{else} \end{cases}$$

   where R is interpreted as an integer in $[2^{n\lambda}]$ by some bijection between $\{0,1\}^{n\lambda}$ and $[2^{n\lambda}]$ here.
3. Output PE.ct.

– Output $\mathsf{mpk} = (\mathsf{prMIFE.mpk}, \mathsf{prMIFE.sk_F}, \mathsf{prMIFE.ct}_{n+1}, \mathsf{PE.mpk})$ and $\mathsf{msk} = (\mathsf{prMIFE.msk}, \mathsf{PE.msk})$.

$\mathsf{KeyGen}(\mathsf{mpk}, \mathsf{msk}, f) \to \mathsf{sk}_f$.The key generation algorithm does the following.

- Parse $\mathsf{mpk} = (\mathsf{prMIFE.mpk}, \mathsf{prMIFE.sk_F}, \mathsf{prMIFE.ct}_0, \mathsf{PE.mpk})$ and $\mathsf{msk} = (\mathsf{prMIFE.msk}, \mathsf{PE.msk})$.
- Compute $\mathsf{PE.sk}_f \leftarrow \mathsf{PE.KeyGen}(\mathsf{PE.msk}, f)$.
- Output $\mathsf{sk}_f = \mathsf{PE.sk}_f$.

$\mathsf{Enc}_i(\mathsf{msk}, \mathbf{x}_i, \mu_i) \to \mathsf{ct}_i$ for $1 \le i \le n$. The slot i encryption algorithm does the following.

- Parse $\mathsf{msk} = (\mathsf{prMIFE.msk}, \mathsf{PE.msk})$.
- Sample $\mathbf{r}_i \leftarrow \{0,1\}^\lambda$ and compute $\mathsf{prMIFE.ct}_i \leftarrow \mathsf{prMIFE.Enc}_i(\mathsf{prMIFE.msk}, (\mathbf{x}_i, \mu_i, \mathbf{r}_i, \perp, \perp))$.
- Output $\mathsf{ct}_i := \mathsf{prMIFE.ct}_i$.

$\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_f, f, \mathsf{ct}_1, \ldots, \mathsf{ct}_n) \to y \in \{0,1\}^n \cup \{\perp\}$. The decryption algorithm does the following.

- Parse $\mathsf{mpk} = (\mathsf{prMIFE.mpk}, \mathsf{prMIFE.sk_F}, \mathsf{prMIFE.ct}_{n+1}, \mathsf{PE.mpk})$, $\mathsf{sk}_f = \mathsf{PE.sk}_f$ and $\{\mathsf{ct}_i = \mathsf{prMIFE.ct}_i\}_{i\in[n]}$.
- Compute $\mathsf{PE.ct} = \mathsf{prMIFE.Dec}(\mathsf{prMIFE.mpk}, \mathsf{prMIFE.sk_F}, \mathrm{F}, \mathsf{prMIFE.ct}_1, \ldots, \mathsf{prMIFE.ct}_n, \mathsf{prMIFE.ct}_{n+1})$.
- Compute $y = \mathsf{PE.Dec}(\mathsf{PE.mpk}, \mathsf{PE.sk}_f, f, \mathsf{PE.ct})$.
- Output y.

Remark 8.1. Here, we discuss how we set Λ. Similarly to the case of prMIFE in Section 7, we consider two cases of parameter settings for the construction. One is the case of n being constant. In this case, we simply set $\Lambda = \lambda$. In the general case of $n = \mathrm{poly}(\lambda)$, we assume that PRF and SKE are subexponentially secure. This means that there exists $0 < \delta < 1$ such that there is no adversary with size 2^{λ^δ} and distinguishing advantage $2^{-\lambda^\delta}$. Similarly to the case of prMIFE (See Remark 7.5 for further discussion), we set Λ so that it satisfies $2^{\Lambda^\delta} \ge \kappa^{\omega(1)}$. An example choice would be to take $\Lambda := (n^2\lambda)^{1/\delta}$.

Correctness. We prove the correctness of our scheme using the following theorem.

Theorem 8.2. Assume PE and prMIFE schemes are correct, and PRF is secure. Then the above construction of miPE scheme is correct.

Proof. From the correctness of the prMIFE scheme and definition of function $\mathrm{F}[\mathsf{PE.mpk}]$, we have

$$\mathsf{prMIFE.Dec}(\mathsf{prMIFE.mpk}, \mathsf{prMIFE.sk_F}, \mathrm{F}, \mathsf{prMIFE.ct}_1, \ldots, \mathsf{prMIFE.ct}_{n+1}) = \mathsf{PE.ct}$$

with probability 1, where

$$\mathsf{PE.ct} = \mathsf{PE.Enc}\left(\mathsf{PE.mpk}, (\mathbf{x}_1, \ldots, \mathbf{x}_n), (\mu_1, \ldots, \mu_n); \mathsf{PRF}(K, R)\right).$$

If we replace $\mathsf{PRF}(K,R)$ with truely random R', we have

$$\mathsf{PE.Dec}(\mathsf{PE.mpk}, \mathsf{PE.sk}_f, f, \mathsf{PE.Enc}\left(\mathsf{PE.mpk}, (\mathbf{x}_1, \ldots, \mathbf{x}_n), (\mu_1, \ldots, \mu_n); R'\right)) = (\mu_1, \ldots, \mu_n)$$

by the correctness of the PE. The decryption succeeds with all but negligible probability even if the randomness is replaced with $\mathsf{PRF}(K,R)$, by the security of PRF.

□

8.2 Security

Here, we prove the security of our scheme. The following theorem asserts the security of the scheme for the case of n being arbitrary polynomial.

Theorem 8.3. Assume prMIFE scheme is (κ,ϵ)-IND-secure (as per Definition 7.4) with respect to $\kappa = \lambda^{(n+1)^2\log\lambda}$ and $\epsilon = \lambda^{-(n+1)\log\lambda/2}$, PE scheme is sub-exponentially secure (Definition 3.23), and the puncturable PRF is sub-exponentially secure. Then the construction of miPE is secure as per Definition 3.25.

Proof. Suppose the PPT adversary $\mathcal{A}$ outputs the following:

1. **Key Queries.** It issues q_0 number of functions $f_1,\ldots,f_{q_0}$ for key queries.
2. **Ciphertext Queries.** It issues q_i number of messages for ciphertext queries for slot i. We use $(\mathbf{x}_{i,0}^{j_i},\mu_{i,0}^{j_i}),(\mathbf{x}_{i,1}^{j_i},\mu_{i,1}^{j_i})$ to denote the j_i-th ciphertext query corresponding to the i-th slot, where $j_i\in[q_i]$ and $i\in[n]$.
3. **Auxiliary Information.** It outputs an auxiliary information $\mathsf{aux}_{\mathcal{A}}$.

We have to prove

$$\begin{pmatrix} \mathsf{mpk} = (\mathsf{prMIFE.mpk}, \mathsf{prMIFE.sk_F}, \mathsf{prMIFE.ct}_{n+1}, \mathsf{PE.mpk}), \\ \mathsf{aux}_{\mathcal{A}},\ \left\{f_k, \mathsf{sk}_{f_k} = \mathsf{PE.sk}_{f_k}\right\}_{k\in[q_0]}, \\ \left\{\mathsf{ct}_i^{j_i} = \mathsf{prMIFE.Enc}_i(\mathsf{prMIFE.msk}, (\mathbf{x}_{i,0}^{j_i}, \mu_{i,0}^{j_i}, \mathbf{r}_i^{j_i}, \bot, \bot))\right\}_{i\in[n], j_i\in[q_i]} \end{pmatrix}.$$
$$\approx_c \begin{pmatrix} \mathsf{mpk} = (\mathsf{prMIFE.mpk}, \mathsf{prMIFE.sk_F}, \mathsf{prMIFE.ct}_{n+1}, \mathsf{PE.mpk}), \\ \mathsf{aux}_{\mathcal{A}},\ \left\{f_k, \mathsf{sk}_{f_k} = \mathsf{PE.sk}_{f_k}\right\}_{k\in[q_0]}, \\ \left\{\mathsf{ct}_i^{j_i} = \mathsf{prMIFE.Enc}_i(\mathsf{prMIFE.msk}, (\mathbf{x}_{i,1}^{j_i}, \mu_{i,1}^{j_i}, \mathbf{r}_i^{j_i}, \bot, \bot))\right\}_{i\in[n], j_i\in[q_i]} \end{pmatrix} \tag{50}$$

where

$(\mathsf{aux}_{\mathcal{A}}, \{f_k\}_k, \{(\mathbf{x}_{i,0}^{j_i},\mu_{i,0}^{j_i}),(\mathbf{x}_{i,1}^{j_i},\mu_{i,1}^{j_i})\}_{i,j_i}) \leftarrow \mathcal{A}(1^\lambda)$,
$(\mathsf{mpk} = (\mathsf{prMIFE.mpk}, \mathsf{prMIFE.sk_F}, \mathsf{prMIFE.ct}_{n+1}, \mathsf{PE.mpk}),\ \mathsf{msk} = (\mathsf{prMIFE.msk}, \mathsf{PE.msk})) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, \mathsf{prm})$,
$\mathsf{PE.sk}_{f_k} \leftarrow \mathsf{PE.KeyGen}(\mathsf{PE.msk}, f_k)$ for $k\in[q_0]$,
$\mathbf{r}_i^{j_i} \leftarrow \{0,1\}^\lambda$ for $i\in[n], j_i\in[q_i]$.

We prove this via the following sequence of hybrids.

Hyb_1. This is the LHS of Equation (50). Namely,

$$\begin{pmatrix} \mathsf{aux}_{\mathcal{A}}, \mathsf{prMIFE.mpk},\ \mathsf{prMIFE.sk_F},\ \mathsf{PE.mpk},\ \left\{f_k, \mathsf{sk}_{f_k} = \mathsf{PE.sk}_{f_k}\right\}_{k\in[q_0]}, \\ \mathsf{prMIFE.ct}_{n+1} = \mathsf{prMIFE.Enc}_{n+1}(\mathsf{prMIFE.msk}, (K, 0, 0, \bot)), \\ \left\{\mathsf{ct}_i^{j_i} = \mathsf{prMIFE.Enc}_i(\mathsf{prMIFE.msk}, (\mathbf{x}_{i,0}^{j_i}, \mu_{i,0}^{j_i}, \mathbf{r}_i^{j_i}, \bot, \bot))\right\}_{i\in[n], j_i\in[q_i]} \end{pmatrix}$$

where we arrange the terms.

Hyb_2. This hybrid is the same as the previous one except that we change how we compute $\mathsf{ct}_i^{j_i}$ for all $i\in[n]$ and $j_i\in[q_i]$. Namely, we compute

$$\mathsf{ct}_i^{j_i} = \mathsf{prMIFE.Enc}_i(\mathsf{prMIFE.msk}, (\mathbf{x}_{i,0}^{j_i}, \mu_{i,0}^{j_i}, \mathbf{r}_i^{j_i}, \mathbf{x}_{i,1}^{j_i}, \mu_{i,1}^{j_i})).$$

We can see that for all combinations of $j_1, \ldots, j_n$, the decryption result of $\mathsf{prMIFE.ct}_1^{j_1}, \ldots, \mathsf{prMIFE.ct}_n^{j_n}, \mathsf{prMIFE.ct}_{n+1}$ using $\mathsf{prMIFE.sk}_F$ is unchanged from the previous game by the definition of $F[\mathsf{PE.mpk}]$. In particular, $F[\mathsf{PE.mpk}]$ simply ignores the newly added terms $(\mathbf{x}_{i,1}^{j_i}, \mu_{i,1}^{j_i})$, since the relevant branch is not triggered. Furthermore, these decryption results are pseudorandom even for an adversary who runs in polynomial time in κ. We show this in Lemma 8.4 in a generalized form, which gives what we need here as a special case of $J^* = 0$. Therefore, this game is computationally indistinguishable from the previous game by the (κ, ϵ)-IND-security of prMIFE.

Hyb_3. This hybrid is the same as the previous one except that we output a failure symbol when there exist $(j_1, \ldots, j_n) \neq (j_1', \ldots, j_n')$ such that $(\mathbf{r}_1^{j_1}, \ldots, \mathbf{r}_n^{j_n}) = (\mathbf{r}_1^{j_1'}, \ldots, \mathbf{r}_n^{j_n'})$. We show that the probability of this happening is negligible in λ. To prove this, it suffices to show that there are no $i \in [n]$, $j, j' \in [q_i]$ satisfying $j \neq j'$ and $\mathbf{r}_i^j = \mathbf{r}_i^{j'}$. The probability of this happening can be bounded by $(q_1^2 + \cdots q_{n-1}^2)/2^\lambda$ by taking the union bound with respect to all the combinations of i, j, j', which can be bounded by $\mathsf{negl}(\lambda)$.

Here, we introduce several notations before describing further hybrids. Each combination of indices $j_1 \in [q_1], \ldots, j_n \in [q_n]$ specifies an element $R^{j_1,\ldots,j_n} = (\mathbf{r}_1^{j_1}, \ldots, \mathbf{r}_n^{j_n})$. There are $q_1 \cdots q_n$ such elements, and due to the change introduced in Hyb_3, we know that they are all distinct. We regard each $R^{j_1,\ldots,j_n}$ as a number in $[2^{n\lambda}]$ via the bijection with $\{0,1\}^{n\lambda}$, and we sort them in ascending order. Let $Q \overset{\text{def}}{=} q_1 \cdots q_n$ and $\{R^J\}_{J \in [Q]}$ denote the sorted sequence corresponding to the set $\{R^{j_1,\ldots,j_n}\}_{j_1 \in [q_1], \ldots, j_n \in [q_n]}$. We also define $R^0 = 0$, which implies $R^0 < R^J$ for any $J \in [Q]$. For each $J \in [Q]$ and $b \in \{0,1\}$, we also define $X_b^J = (\mathbf{x}_{1,b}^{j_1}, \ldots, \mathbf{x}_{n,b}^{j_n})$ and $M_b^J = (\mu_{1,b}^{j_1}, \ldots, \mu_{n,b}^{j_n}))$, where $(j_1, \ldots, j_n)$ is the unique combination of indices such that $R^J = (\mathbf{r}_1^{j_1}, \ldots, \mathbf{r}_n^{j_n})$.

We then define $\mathsf{Hyb}_{3,J}$ for $J \in [0, Q]$ as follows.

$\mathsf{Hyb}_{3,J}$. This hybrid is the same as Hyb_3 except that we change how we compute $\mathsf{prMIFE.ct}_{n+1}$. Namely, we compute

$$\mathsf{prMIFE.ct}_{n+1} = \mathsf{prMIFE.Enc}_{n+1}(\mathsf{prMIFE.msk}, (K, R^J, 0, \perp)).$$

$\mathsf{Hyb}_{3,0}$ is exactly the same as Hyb_3, since we defined $R^0 = 0$. We then want to prove that $\mathsf{Hyb}_{3,0}$ is computationally indistinguishable from $\mathsf{Hyb}_{3,Q}$. Towards this goal, we show that $\mathsf{Hyb}_{3,J-1}$ is indistinguishable from $\mathsf{Hyb}_{3,J}$ for all $J \in [Q]$ with subexponentially small advantage, since there are exponentially many hybrids (i.e., $Q+1$ hybrids). To establish this, we introduce the following additional sub-hybrids.

$\mathsf{Hyb}_{3,J,0}$. This is the same as $\mathsf{Hyb}_{3,J}$.

$\mathsf{Hyb}_{3,J,1}$. This is the same as $\mathsf{Hyb}_{3,J,0}$ except that we compute $\mathsf{prMIFE.ct}_{n+1}$ as

$$\mathsf{prMIFE.ct}_{n+1} = \mathsf{prMIFE.Enc}_{n+1}(\mathsf{prMIFE.msk}, (K\{R^J\}, R^J, 1, \mathsf{PE.ct}^*)),$$

where

$$\mathsf{PE.ct}^* = \mathsf{PE.Enc}(\mathsf{PE.mpk}, X_0^J, M_0^J; \mathsf{PRF}(K, R^J)) \quad \text{and} \quad K\{R^J\} = \mathsf{Puncture}(K, R^J).$$

We can see that for all combinations of $j_1, \ldots, j_n$, the decryption result of $\mathsf{prMIFE.ct}_1^{j_1}, \ldots, \mathsf{prMIFE.ct}_n^{j_n}, \mathsf{prMIFE.ct}_{n+1}$ using $\mathsf{prMIFE.sk}_F$ is unchanged from the previous game by the correctness of the puncturable PRF and by the definition of $F[\mathsf{PE.mpk}]$. In particular, for the particular combination of $j_1, \ldots, j_n$ such that $R^J = (\mathbf{r}_1^{j_1}, \ldots, \mathbf{r}_n^{j_n})$, the decryption of $\mathsf{prMIFE.ct}_1^{j_1}, \ldots, \mathsf{prMIFE.ct}_n^{j_n}$ using $\mathsf{prMIFE.sk}_F$ triggers the branch of the function that outputs $\mathsf{PE.ct}^*$ encrypted under $\mathsf{prMIFE}_{n+1}.\mathsf{ct}$. However, this PE ciphertext is exactly the same as the one output by the decryption of the same combination of the ciphertexts and the secret key in the previous game. Furthermore, these decryption results are pseudorandom even for an adversary who runs in polynomial time in κ. To see this, we apply Lemma 8.4 for the case of $J^* = J$. Therefore, we can conclude that the adversary cannot distinguish this hybrid from the previous one with advantage more than $\epsilon = \lambda^{-(n+1)\log\lambda/2}$ by the (κ, ϵ)-IND-security of prMIFE.

$\mathsf{Hyb}_{3,J,2}$. This is the same as $\mathsf{Hyb}_{3,J,1}$ except that we replace $\mathsf{PRF}(K, R^J)$ with real randomness. Namely, we set

$$\mathsf{PE.ct}^* \leftarrow \mathsf{PE.Enc}(\mathsf{PE.mpk}, X_0^J, M_0^J).$$

By the subexponential security of the pucturable PRF PRF, this hybrid is computationally indistinguishable from the previous one. In particular, by our choice of the parameter $2^{\Lambda^\delta} > \kappa^{\omega(1)}$, we can conclude that the adversary cannot distinguish this hybrid from the previous one with advantage more than $\kappa^{-\omega(1)} < \lambda^{-(n+1)\log\lambda/2}$.

$\mathsf{Hyb}_{3,J,3}$. This is the same as $\mathsf{Hyb}_{3,J,2}$ except that $\mathsf{PE.ct}^*$ is computed as

$$\mathsf{PE.ct}^* \leftarrow \mathsf{PE.Enc}(\mathsf{PE.mpk}, X_1^J, M_1^J).$$

By the subexponential security of PE, this hybrid is computationally indistinguishable from the previous one, since we have $f_k(X_0^J) = f_k(X_1^J) = 0$. In particular, by our choice of the parameter $2^{\Lambda^\delta} > \kappa^{\omega(1)}$, we can conclude that the adversary cannot distinguish this hybrid from the previous one with advantage more than $\kappa^{-\omega(1)} < \lambda^{-(n+1)\log\lambda/2}$.

$\mathsf{Hyb}_{3,J,4}$. This is the same as $\mathsf{Hyb}_{3,J,3}$ except that $\mathsf{PE.ct}^*$ is computed as

$$\mathsf{PE.ct}^* \leftarrow \mathsf{PE.Enc}(\mathsf{PE.mpk}, X_1^J, M_1^J; \mathsf{PRF}(K, R^J)).$$

Similarly to the difference between $\mathsf{Hyb}_{3,J,1}$ and $\mathsf{Hyb}_{3,J,2}$, by the security of the puncturable PRF, this game is computationally indistinguishable from the previous one with advantage more than $\lambda^{-(n+1)\log\lambda/2}$.

$\mathsf{Hyb}_{3,J,5}$. This game is the same as $\mathsf{Hyb}_{3,J+1,0}$. Namely, we unpuncture K and compute $\mathsf{prMIFE.ct}_{n+1}$ as

$$\mathsf{prMIFE.ct}_{n+1} = \mathsf{prMIFE.Enc}_{n+1}(\mathsf{prMIFE.msk}, (K, R^{J+1}, 0, \perp)).$$

Similarly to the difference between $\mathsf{Hyb}_{3,J,0}$ and $\mathsf{Hyb}_{3,J,1}$, by the correctness of the puncturable PRF and the (κ, ϵ)-IND security of prMIFE, this game is computationally indistinguishable from the previous one with advantage more than $\lambda^{-(n+1)\log\lambda/2}$. To invoke the security of prMIFE, we use Lemma 8.4 with $J^* = J + 1$.

The above shows that the adversary cannot distinguish $\mathsf{Hyb}_{3,J-1}$ from $\mathsf{Hyb}_{3,J}$ with advantage more than $5\lambda^{-(n+1)\log\lambda/2}$. This implies that Hyb_3 and $\mathsf{Hyb}_{3,Q}$ is computationally indistinguishable, since we have

$$Q \cdot (5\lambda^{-(n+1)\log\lambda/2}) \le 5\left(\prod_{i=1}^{n} q_i\right)\lambda^{-(n+1)\log\lambda/2} = 5\left(\prod_{i=1}^{n} q_i/\lambda^{\log\lambda/2}\right)\lambda^{-\log\lambda/2} = \mathsf{negl}(\lambda).$$

Hyb_4. This is the RHS of Equation (50). Namely,

$$\left(\begin{array}{c} \mathsf{aux}_{\mathcal{A}}, \mathsf{prMIFE.mpk},\ \mathsf{prMIFE.sk}_{\mathsf{F}},\ \mathsf{PE.mpk},\ \left\{f_k, \mathsf{sk}_{f_k} = \mathsf{PE.sk}_{f_k}\right\}_{k\in[q_0]}, \\ \mathsf{prMIFE.ct}_{n+1} = \mathsf{prMIFE.Enc}_{n+1}(\mathsf{prMIFE.msk}, (K, 0, 0, \perp)), \\ \left\{\mathsf{ct}_i^{j_i} = \mathsf{prMIFE.Enc}_i(\mathsf{prMIFE.msk}, (\mathbf{x}_{i,1}^{j_i}, \mu_{i,1}^{j_i}, \mathbf{r}_i^{j_i}, \perp, \perp))\right\}_{i\in[n], j_i\in[q_i]} \end{array}\right).$$

We claim that this hybrid is computationally indistinguishable from $\mathsf{Hyb}_{3,Q}$. To see this, We claim that for all combinations of $j_1, \ldots, j_n$, the decryption result of $\mathsf{prMIFE.ct}_1^{j_1}, \ldots, \mathsf{prMIFE.ct}_n^{j_n}, \mathsf{prMIFE.ct}_{n+1}$ using $\mathsf{prMIFE.sk}_F$ is unchanged from $\mathsf{Hyb}_{3,Q}$ by the correctness of the puncturable PRF and by the definition of $F[\mathsf{PE.mpk}]$. In particular, $F[\mathsf{PE.mpk}]$ always outputs PE ciphertext for $\{(\mathbf{x}_{i,1}^{j_i}, \mu_{i,1}^{j_i})\}_i$ in both games. Furthermore, these decryption results are pseudorandom even for an adversary who runs in polynomial time in κ. To see this, we apply Lemma 8.4 for the case of $J^* = Q$. Therefore, we can conclude that this hybrid is computationally indistinguishable from $\mathsf{Hyb}_{3,Q}$ by the (κ, ϵ)-IND-security of prMIFE.

To complete the proof of the theorem, it remains to prove Lemma 8.4.

Lemma 8.4. For any $J^* \in [0, Q]$, the following two distributions are computationally indistinguishable

$$\left(\begin{array}{r} 1^\kappa,\ \mathsf{aux}_{\mathcal{A}},\ \mathsf{PE.mpk}, \\ \left\{f_k, \mathsf{sk}_{f_k} = \mathsf{PE.sk}_{f_k}\right\}_{k\in[q_0]}, \\ \left\{\mathsf{PE.ct}^J = \mathsf{PE.Enc}(\mathsf{PE.mpk}, X_1^J, M_1^J; \mathsf{PRF}(K, R^J))\right\}_{J\in[J^*]}, \\ \left\{\mathsf{PE.ct}^J = \mathsf{PE.Enc}(\mathsf{PE.mpk}, X_0^J, M_0^J; \mathsf{PRF}(K, R^J))\right\}_{J\in[J^*+1,Q]} \end{array}\right) \approx_c \left(\begin{array}{r} 1^\kappa,\ \mathsf{aux}_{\mathcal{A}},\ \mathsf{PE.mpk}, \\ \left\{f_k, \mathsf{sk}_{f_k} = \mathsf{PE.sk}_{f_k}\right\}_{k\in[Q]}, \\ \left\{\mathsf{PE.ct}^J \leftarrow \mathcal{CT}_{\mathsf{PE}}\right\}_{J\in[J^*]}, \\ \left\{\mathsf{PE.ct}^J \leftarrow \mathcal{CT}_{\mathsf{PE}}\right\}_{J\in[J^*+1,Q]} \end{array}\right)$$

where

$$\begin{aligned} &(\mathsf{aux}_{\mathcal{A}}, \{f_k\}_k, \{(\mathbf{x}_{i,0}^{j_i}, \mu_{i,0}^{j_i}), (\mathbf{x}_{i,1}^{j_i}, \mu_{i,1}^{j_i})\}_{i,j_i}) \leftarrow \mathcal{A}(1^\lambda), \\ &(\mathsf{PE.mpk}, \mathsf{PE.msk}) \leftarrow \mathsf{PE.Setup}(1^\Lambda, \mathsf{prm}_{\mathsf{PE}}), \\ &\mathsf{PE.sk}_{f_k} \leftarrow \mathsf{PE.KeyGen}(\mathsf{PE.msk}, f_k) \text{ for } k \in [q_0]. \end{aligned}$$

Proof. We start from the LHS of the above distributions and gradually change it to that of the RHS. Our first step is to change all the PRF values $\{\mathsf{PRF}(K, R^J)\}_{J\in[Q]}$ used inside the PE encryption to be truly random. This change is unnoticed by the distinguisher by the sub-exponential security of the PRF and by our choice of the parameter $2^{\Lambda^\delta} > \kappa^{\omega(1)}$. In particular, the distinguishing advantage of the adversary is less than $\kappa^{-\omega(1)}$ even if it runs in time polynomial in κ. We then change all the PE ciphertexts to be chosen from $\mathcal{CT}_{\mathsf{PE}}$. This change is unnoticed by the distinguisher by the sub-exponential security of the PE and by our choice of the parameter $2^{\Lambda^\delta} > \kappa^{\omega(1)}$. In more detail, we may change all the ciphertexts to be random one-by-one, using the security of the PE. The distinguishing advantage of the adversary for each change is less than $\kappa^{-\omega(1)}$ even if it runs in time polynomial in κ, by the sub-exponential security of the PE. There are $Q < \mathrm{poly}(\kappa)$ number of ciphertexts and the overall distinguishing advantage is bounded by $\mathrm{poly}(\kappa) \cdot \kappa^{-\omega(1)} = \kappa^{-\omega(1)}$. This completes the proof of the lemma. □

This completes the proof of the theorem. □

Constant arity case. In the special case of n being a constant, we can base the security of the scheme on weaker security requirements for the underlying ingredients. Concretely, we have the following theorem.

Theorem 8.5. Assume prMIFE scheme is κ-IND secure (as per Definition 7.4) with respect to $\kappa = \lambda^n$, PE scheme is secure (Definition 3.23) and PRF is secure. Then the construction of miPE is secure as per Definition 3.25.

The proof of the above theorem is exactly the same as Theorem 8.3 except that here $\kappa = \lambda^n$. Since $\kappa = \mathrm{poly}(\lambda)$, we do not need subexponential security for PRF and PE. In addition, since $Q = q_1 \cdots q_n = \mathrm{poly}(\lambda)$ for constant n, the number of hybrids that appear in the proof is only polynomial in λ. Therefore, we do not need (κ, ϵ)-IND security for prMIFE for subexponentially small ϵ here. Since prMIFE for constant arity with $\kappa = \lambda^n$ can be constructed from (plain) evasive LWE, we can base the security of prMIFE on the same assumption with suitably defined sampler. This is weaker assumption than non-uniform κ-version of it that is necessary for the general case.

9 Indistinguishability Obfuscation for Pseudorandom Functionalities

9.1 Definition

Syntax. An indistinguishability obfuscator for pseudorandom functionalities (prIO) consists of the following algorithms.

$\mathsf{iO}(1^\lambda, C) \to \tilde{C}$. The obfuscation algorithm takes as input the security parameter λ and a circuit $C : \{0,1\}^n \to \{0,1\}^m$ with arbitrary n and m. It outputs an obfuscated circuit $\tilde{C}$.

$\mathsf{Eval}(\tilde{C}, x) \to y$. The evaluation algorithm takes as input an obfuscated circuit $\tilde{C}$ and an input $x \in \{0,1\}^n$. It outputs y.

A uniform PPT machine iO is an indistinguishability obfuscator for pseudorandom functionalities w.r.t parameter $\kappa = \kappa(\lambda)$ if it satisfies the following properties.

Definition 9.1 (Polynomial Slowdown). For all security parameters $\lambda \in \mathbb{N}$, for any circuit C and every input x, the evaluation time of $\mathsf{iO}(1^\lambda, C)$ on x is at most polynomially slower than the run time of the circuit C on x.

Definition 9.2 (Correctness). For all security parameters $\lambda \in \mathbb{N}$, for all integers n, m, all circuits $C : \{0,1\}^n \to \{0,1\}^m$, and all input $x \in \{0,1\}^n$, we have that:

$$\Pr\left[C' \leftarrow \mathsf{iO}(1^\lambda, C) : C'(x) = C(x)\right] = 1$$

where the probability is taken over the coin-tosses of the obfuscator iO.

Definition 9.3 (Indistinguishability for Pseudorandom Functionality). For the security parameter $\lambda = \lambda(\lambda)$, let Samp be a PPT algorithm that on input 1^λ, outputs

$$(C_0, C_1,\ \mathsf{aux} \in \{0,1\}^*)$$

where $C_0 : \{0,1\}^n \to \{0,1\}^m$ and $C_1 : \{0,1\}^n \to \{0,1\}^m$ have the same description size. We say that a prIO scheme is secure if for every PPT sampler Samp the following holds.

$$\begin{aligned} &\text{If} \quad \left(1^\kappa, \{C_0(x)\}_{x \in \{0,1\}^n}, \mathsf{aux}\right) \approx_c \left(1^\kappa, \{\Delta_x \leftarrow \{0,1\}^m\}_{x \in \{0,1\}^n}, \mathsf{aux}\right) \\ &\text{and} \quad C_0(x) = C_1(x) \quad \forall x \in \{0,1\}^n, \\ &\text{then} \ \left(\mathsf{iO}(1^\lambda, C_0), \mathsf{aux}\right) \approx_c \left(\mathsf{iO}(1^\lambda, C_1), \mathsf{aux}\right) \end{aligned}$$

where $\kappa \geq 2^n$.

Remark 9.4. Note that 1^κ in the precondition is introduced for the purpose of padding, allowing the distinguisher for the distributions to run in time polynomial in κ. The reason why we require $\kappa \geq 2^n$ is that the input length to the distinguisher is polynomial in 2^n anyway and in order for the padding to make sense, κ should satisfy this condition.

Remark 9.5. Note that the above security definition is strictly weaker than the standard security notion for IO [GGH+16], since we require that the obfuscation of the circuits to be indistinguishable only when their truth tables are pseudorandom.

9.2 Construction

Our construction follows the multi-input FE to iO conversion by [GGG+14]. To obfuscate a circuit with input domain $\{0,1\}^n$, we generate a prMIFE instance for arity $n+1$. We then let C be encrypted in position $n+1$, and the two inputs 0 and 1 be encrypted in position i for $i \in [n]$. The $2n+1$ ciphertexts together with a secret key for the universal circuit and the public parameters would form the iO.

Building Blocks. We use a $(n+1)$-input prFE scheme $\mathsf{prMIFE} = \mathsf{prMIFE}.(\mathsf{Setup}, \mathsf{KeyGen}, \{\mathsf{Enc}_i\}_{i \in [n+1]}, \mathsf{Dec})$ for the circuit class with fixed input length, bounded depth, and binary output. We require prMIFE to satisfy κ-IND-security defined as per Definition 7.4. We can instantiate the scheme by our construction in Section 7.2 with $\kappa = \lambda^{n^2 \log \lambda}$.

Next, we describe the construction of $\mathsf{prIO} = (\mathsf{iO}, \mathsf{Eval})$ for all circuits.

$\mathsf{iO}(1^\lambda, C)$. Given as input the security parameter 1^λ and a circuit $C : \{0,1\}^n \to \{0,1\}^m$ for arbitrary input length n, output length m, and description size L, do the following:

- Run $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{prMIFE.Setup}(1^\lambda, 1^{n+1}, \mathsf{prm})$, where prm specifies message length L and the maximum depth d of the circuits supported by the prMIFE instance. We set d to be the depth of the universal circuit U that, upon input an n-ary circuit C and vector $\mathbf{x} \in \{0,1\}^n$, outputs $U(C, \mathbf{x}) = C(\mathbf{x})$.

- Compute $\mathsf{ct}_{n+1} \leftarrow \mathsf{prMIFE.Enc}_{n+1}(\mathsf{msk}, C)$.
- For $i \in [n]$ and $b \in \{0,1\}$, compute $\mathsf{ct}_{i,b} = \mathsf{prMIFE.Enc}_i(\mathsf{msk}, b)$.
- Compute $\mathsf{sk}_U \leftarrow \mathsf{prMIFE.KeyGen}(\mathsf{msk}, U)$, where U is defined in the first item above.
- Output $\tilde{C} = \left(\{\mathsf{ct}_{i,b}\}_{i\in[n],b\in\{0,1\}}, \mathsf{ct}_{n+1}, \mathsf{sk}_U, \mathsf{mpk} \right)$

$\mathsf{Eval}(\tilde{C}, \mathbf{x})$**.** Given as input an obfuscated circuit $\tilde{C}$ and an input $\mathbf{x} \in \{0,1\}^n$, do the following:

1. Parse $\tilde{C} = \left(\{\mathsf{ct}_{i,b}\}_{i\in[n],b\in\{0,1\}}, \mathsf{ct}_{n+1}, \mathsf{sk}_U, \mathsf{mpk} \right)$.
2. Output $\mathsf{prMIFE.Dec}(\mathsf{mpk}, \mathsf{sk}_U, U, \mathsf{ct}_{x_1}, \ldots, \mathsf{ct}_{x_n}, \mathsf{ct}_{n+1})$, where x_i is the i-th bit of $\mathbf{x}$.

Remark 9.6. We note that the input size of prMIFE scheme varies for slot 0, where we encrypt C, and slot i, where we encrypt a bit b, for $i \in [n]$. To make the input size consistent throughout the slots, we can pad the bit b with $\mathbf{0}$ (say) such that $|b\mathbf{0}| = |C|$ and give a prMIFE key for a circuit U which on input $(C, b_1\mathbf{0}, \ldots, b_n\mathbf{0})$, where $b_i \in \{0,1\}$, simply discards the padding and outputs $C(b_1, \ldots, b_n)$.

Correctness. The correctness of the scheme follows in a straightforward manner from the correctness of the underlying prMIFE scheme and the definition of the universal circuit U.

9.3 Security

Theorem 9.7. Suppose prMIFE scheme is κ-IND-secure (Definition 7.4). Then the prIO scheme satisfies security as defined in Definition 9.3 with $\kappa = \kappa(\lambda)$.

Proof. Consider a sampler $\mathsf{Samp}_{\mathsf{prIO}}$ that generates the following:

1. **Obfuscation Query.** It issues $C_0, C_1 : \{0,1\}^n \to \{0,1\}^m$ with the same size L and the same truth table (i.e., we have $C_0(x) = C_1(x)$ for all $x \in \{0,1\}^n$) as an obfuscation query.
2. **Auxiliary Information.** It outputs the auxiliary information $\mathsf{aux}_{\mathcal{A}}$.

To prove the security as per Definition 9.3, we show that

$$\left(\{\mathsf{ct}_{i,b}\}_{i\in[n],b\in\{0,1\}}, \mathsf{ct}^0_{n+1}, \mathsf{sk}_U, \mathsf{mpk}, \mathsf{aux}_{\mathcal{A}}\right) \approx_c \left(\{\mathsf{ct}_{i,b}\}_{i\in[n],b\in\{0,1\}}, \mathsf{ct}^1_{n+1}, \mathsf{sk}_U, \mathsf{mpk}, \mathsf{aux}_{\mathcal{A}}\right) \tag{51}$$

$$\text{if } \left(1^\kappa, \{C_0(x)\}_{x\in\mathcal{X}_\lambda}, \mathsf{aux}_{\mathcal{A}}\right) \approx_c \left(1^\kappa, \{\Delta_x \leftarrow \mathcal{Y}_\lambda\}_{x\in\mathcal{X}_\lambda}, \mathsf{aux}_{\mathcal{A}}\right) \tag{52}$$

where

$$\begin{aligned}
&(C_0, C_1, \mathsf{aux}_{\mathcal{A}}) \leftarrow \mathsf{Samp}_{\mathsf{prIO}}(1^\lambda),\\
&(\mathsf{msk}, \mathsf{mpk}) \leftarrow \mathsf{prMIFE.Setup}(1^\lambda, 1^{n+1}, \mathsf{prm}),\\
&\mathsf{sk}_U \leftarrow \mathsf{prMIFE.KeyGen}(\mathsf{msk}, U),\\
&\mathsf{ct}^0_{n+1} \leftarrow \mathsf{prMIFE.Enc}_{n+1}(\mathsf{msk}, C_0), \mathsf{ct}^1_{n+1} \leftarrow \mathsf{prMIFE.Enc}_{n+1}(\mathsf{msk}, C_1),\\
&\mathsf{ct}_{i,b} = \mathsf{prMIFE.Enc}_i(\mathsf{msk}, b), \text{ for } i \in [n], b \in \{0,1\}.
\end{aligned}$$

Equation (51) immediately follows from prMIFE security by considering a sampler $\mathsf{Samp}_{\mathsf{prMIFE}}$ that outputs

$$\left(\begin{array}{cc}
\text{Function:} & \text{Universal Circuit } U\\
\text{Inputs:} & \{x^{j_1}_{1,0} = x^{j_1}_{1,1} = j_1, \ldots, x^{j_n}_{n,0} = x^{j_n}_{n,1} = j_n\}_{j_1\in\{0,1\},\ldots,j_n\in\{0,1\}}, x_{n+1,0} = C_0, x_{n+1,1} = C_1\\
\text{Auxiliary Information:} & \mathsf{aux}_{\mathcal{A}}
\end{array}\right),$$

since the pseudorandom condition directly follows from Equation (52).

□

Full Domain Hash H

Constants: PRF key K, TDP key PK
Input: Message m.

- Output $g_{\mathsf{PK}}(F(K,m))$.

Figure 3: Full Domain Hash

10 Instantiating the Random Oracle Using prIO

Hohenberger, Sahai, and Waters [HSW14] show that some applications of random oracles can be made secure in the standard model by instantiating the hash functions using iO in a specific manner. In this section, we show that we can replace full-fledged iO with prIO in these applications. As a concrete example, we show that full-domian hash (FDH) signatures can be proven secure in the standard model if we instantiate the hash function using prIO in place of iO.

10.1 Full-Domain Hash Signatures (Selectively Secure) from prIO

Ingredients. We make use of the following ingredients.

1. A one-way trapdoor permutation family (TDP).
2. Punctured PRFs
3. A prIO scheme.

The construction follows.

1. $\mathsf{Setup}(1^\lambda)$: The setup algorithm takes as input the security parameter and does the following:
    - It runs the setup of the TDP to obtain a public index PK along with a trapdoor SK, yielding the map $g_{\mathsf{PK}} : \{0,1\}^n \to \{0,1\}^n$ together with its inverse g^{-1}_{SK}.
    - It chooses a puncturable PRF key K for F where $F(K,\cdot) : \{0,1\}^\lambda \to \{0,1\}^n$. Then, it creates a prIO obfuscation of the program **Full Domain Hash** in Figure 3. We refer to the obfuscated program as the function $H : \{0,1\}^\lambda \to \{0,1\}^n$. We need the truth table of the PRF to be pseudorandom against an adversary whose size is polynomial in κ, where κ is the parameter specified by our prIO. This can be achieved assuming the subexponential security for the PRF.
    - It outputs the verification key VK as the trapdoor index PK as well as the hash function $H(\cdot)$. The secret key is the trapdoor SK.
2. $\mathsf{Sign}(\mathsf{SK},\mathsf{m})$: Output $\sigma = g^{-1}_{\mathsf{SK}}(H(m))$.
3. $\mathsf{Verify}(\mathsf{VK},m,\sigma)$: Check if $g_{\mathsf{PK}}(\sigma) = H(m)$ and output "Accept" if and only if this is true.

Correctness follows from the correctness of the TDP. We sketch security next.

Security The security proof closely resembles that of [HSW14], except that we have to make sure that the truth table of the obfuscated circuit is pseudorandom when we apply the security of prIO.

Theorem 10.1. If the prIO is secure as per Definition 9.3 with respect to a parameter κ, F is subexponentially secure punctured PRF, and the trapdoor permutation scheme TDP is one-way, then the above signature scheme is selectively secure.

Full Domain Hash* H

Constants: Punctured PRF key $K(m^*)$, m^*, z^*, TDP key PK
Input: Message m.

- If $m = m^*$, output z^* and exit.
- Else output $g_{\mathsf{PK}}(F(K, m))$.

Figure 4: Full Domain Hash*

Proof. The proof follows the same sequence of hybrids as [HSW14]. In the first hybrid, we move to using the obfuscation of the circuit **Full Domain Hash*** in Figure 4 with $z^* = g_{\mathsf{PK}}(F(K, m^*))$, where m^* is chosen by the adversary at the beginning of the game. We claim that this change is unnoticed by the adversary by the security of prIO. To see this, we observe that the truth tables of the circuits in Figure 3 and Figure 4 are the same. Furthermore, the truth table is pseudorandom even against an adversary who runs in time $\mathsf{poly}(\kappa)$ by the sub-exponential security of PRF. In the next hybrid, we replace z^* hardwired into the obfuscated circuit with truly random point in $\{0, 1\}^n$. This change is unnoticed by the security of the puncturable PRF. This allows to reduce the security to that of the TDP, exactly as in [HSW14] since a valid signature would imply a preimage to z^* and other signatures can be simulated using the punctured PRF key. □

10.2 Discussion about Other Applications.

Hohenberger, Sahai, and Waters [HSW14] demonstrate that the selective security of the full-domain hash (FDH) signature based on trapdoor permutations (TDP), the adaptive security of RSA FDH signatures [Cor00], the selective security of BLS signatures, and the adaptive security of BLS signatures [BLS01] can be proven in the standard model by carefully instantiating the underlying hash function by iO for each application. As shown in Section 10.1, the random oracle in the FDH signature can be instantiated using prIO instead of full-fledged iO. Similarly, we can instantiate the random oracle in selectively secure BLS signatures with prIO, following a strategy similar to that in [HSW14]. At a high level, these proofs follow those in the random oracle model (ROM), where we use iO to obfuscate a derandomized version of the simulator for the hash function in ROM-based proofs. In these settings, the truth table of the simulated hash function is pseudorandom, allowing us to follow the same proof strategy using prIO.

For adaptively secure RSA FDH and BLS signatures, the situation is different. In these cases, Hohenberger et al. adopt an alternative proof strategy that deviates from the high level strategy of obfuscating the simulator for the proof in the ROM. This is due to the fact that the original proofs [BLS01, Cor00] are incompatible with the conditions required for using iO, where the truth table of the hash functions must remain unchanged across game hops. To be compatible with iO, they introduce a structure for the hash function, making its truth table no longer pseudorandom. This prevents us from replacing the hash function with prIO following their approach. To instantiate the hash function with prIO, we may first consider the selectively secure variant of the signature schemes and then use the complexity leveraging technique. We omit the details.

Acknowledgement. The first author is supported by the CyStar center of excellence, the VHAR faculty chair and the C3iHub fellowship. The third author is partly supported by JST CREST Grant Number JPMJCR22M1.

References

[ABB10a] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 553–572. Springer, Heidelberg, May / June 2010. (Cited on page 22.)

[ABB10b] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical IBE. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 98–115. Springer, Heidelberg, August 2010. (Cited on page 22.)

[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, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 657–677. Springer, Heidelberg, August 2015. (Cited on page 20.)

[Agr19] Shweta Agrawal. Indistinguishability obfuscation without multilinear maps: New techniques for bootstrapping and instantiation. In *Eurocrypt*, 2019. (Cited on page 4.)

[AJ15] Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 308–326. Springer, Heidelberg, August 2015. (Cited on page 4, 5, 17, 18, 55.)

[AJS23] Paul Lou Aayush Jain, Huijia Lin and Amit Sahai. Polynomial-time cryptanalysis of the subspace flooding assumption for post-quantum io. In *Eurocrypt*, 2023. (Cited on page 4.)

[AKY24] Shweta Agrawal, Simran Kumari, and Shota Yamada. Attribute Based Encryption for Turing Machines from Lattices. In *Crypto*, 2024. (Cited on page 3, 4, 7, 10, 13, 53.)

[AKYY23] Shweta Agrawal, Simran Kumari, Anshu Yadav, and Shota Yamada. Broadcast, trace and revoke with optimal parameters from polynomial hardness. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 605–636. Springer, 2023. (Cited on page 4.)

[AM18] Shweta Agrawal and Monosij Maitra. FE and iO for Turing machines from minimal assumptions. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part II*, volume 11240 of *LNCS*, pages 473–512. Springer, Heidelberg, November 2018. (Cited on page 3.)

[AMY19a] Shweta Agrawal, Monosij Maitra, and Shota Yamada. Attribute based encryption (and more) for nondeterministic finite automata from LWE. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 765–797. Springer, Heidelberg, August 2019. (Cited on page 3.)

[AMY19b] Shweta Agrawal, Monosij Maitra, and Shota Yamada. Attribute based encryption for deterministic finite automata from DLIN. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part II*, volume 11892 of *LNCS*, pages 91–117. Springer, Heidelberg, December 2019. (Cited on page 3.)

[AMYY25] Shweta Agrawal, Anuja Modi, Anshu Yadav, and Shota Yamada. Evasive lwe: Attacks, variants & obfustopia, 2025. (Cited on page 3, 5, 6, 8, 9, 12, 13, 23, 32, 55.)

[APM20] Shweta Agrawal and Alice Pellet-Mary. Indistinguishability obfuscation without maps: Attacks and fixes for noisy linear fe. In *Advances in Cryptology–EUROCRYPT 2020: 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10–14, 2020, Proceedings, Part I*, pages 110–140. Springer, 2020. (Cited on page 4.)

[ARYY23] Shweta Agrawal, Mélissa Rossi, Anshu Yadav, and Shota Yamada. Constant input attribute based (and predicate) encryption from evasive and tensor LWE. In *CRYPTO 2023, Part IV*, LNCS, pages 532–564. Springer, Heidelberg, August 2023. (Cited on page 3, 4, 7, 23, 24.)

[AS16] Prabhanjan Vijendra Ananth and Amit Sahai. Functional encryption for Turing machines. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part I*, volume 9562 of *LNCS*, pages 125–153. Springer, Heidelberg, January 2016. (Cited on page 3.)

[AY20] Shweta Agrawal and Shota Yamada. CP-ABE for circuits (and more) in the symmetric key setting. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part I*, volume 12550 of *LNCS*, pages 117–148. Springer, Heidelberg, November 2020. (Cited on page 3.)

[AYY22] Shweta Agrawal, Anshu Yadav, and Shota Yamada. Multi-input attribute based encryption and predicate encryption. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part I*, volume 13507 of *LNCS*, pages 590–621. Springer, Heidelberg, August 2022. (Cited on page 3, 7, 29.)

[BDJ$^+$25] Pedro Branco, Nico Döttling, Abhishek Jain, Giulio Malavolta, Surya Mathialagan, Spencer Peters, and Vinod Vaikuntanathan. Pseudorandom obfuscation and applications. In *CRYPTO*, 2025. Available from `https://eprint.iacr.org/2024/1742`. (Cited on page 5, 6, 8, 9, 12, 13, 55.)

[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, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 533–556. Springer, Heidelberg, May 2014. (Cited on page 3, 9, 66.)

[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, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 1–18. Springer, Heidelberg, August 2001. (Cited on page 4, 8.)

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 501–519. Springer, Heidelberg, March 2014. (Cited on page 26.)

[BLP$^+$13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 575–584. ACM Press, June 2013. (Cited on page 22, 23.)

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532. Springer, Heidelberg, December 2001. (Cited on page 7, 21, 75.)

[BLSV18] Zvika Brakerski, Alex Lombardi, Gil Segev, and Vinod Vaikuntanathan. Anonymous IBE, leakage resilience and circular security from new assumptions. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 535–564. Springer, Heidelberg, April / May 2018. (Cited on page 15, 27.)

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 62–73, 1993. (Cited on page 7, 21.)

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *Theory of Cryptography: 8th Theory of Cryptography Conference, TCC 2011, Providence, RI, USA, March 28-30, 2011. Proceedings 8*, pages 253–273. Springer, 2011. (Cited on page 4, 6, 8, 32.)

[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, Part I*, volume 10677 of *LNCS*, pages 264–302. Springer, Heidelberg, November 2017. (Cited on page 10, 25.)

[BÜW24] Chris Brzuska, Akin Ünal, and Ivy KY Woo. Evasive lwe assumptions: Definitions, classes, and counterexamples. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 418–449. Springer, 2024. (Cited on page 3, 4, 5, 8.)

[BV18] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. *Journal of the ACM*, 65(6):39:1–39:37, 2018. (Cited on page 4, 5.)

[BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 280–300. Springer, Heidelberg, December 2013. (Cited on page 26.)

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, 2004. (Cited on page 8.)

[CHKP10] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 523–552. Springer, Heidelberg, May / June 2010. (Cited on page 22.)

[Cor00] Jean-Sébastien Coron. On the exact security of full domain hash. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 229–235. Springer, Heidelberg, August 2000. (Cited on page 7, 21, 75.)

[CRV10] Ran Canetti, Guy N. Rothblum, and Mayank Varia. Obfuscation of hyperplane membership. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 72–89. Springer, Heidelberg, February 2010. (Cited on page 8.)

[CW24] Jeffrey Champion and David J Wu. Distributed broadcast encryption from lattices. In *Theory of Cryptography Conference*, pages 156–189. Springer, 2024. (Cited on page 9.)

[CW25] Valerio Cini and Hoeteck Wee. Faster abe for turing machines from circular evasive lwe. Springer-Verlag, 2025. (Cited on page 3, 5, 6, 7, 8, 9.)

[DCIJ$^+$13] Angelo De Caro, Vincenzo Iovino, Abhishek Jain, Adam O'Neill, Omer Paneth, and Giuseppe Persiano. On the achievability of simulation-based security for functional encryption. In *Advances in Cryptology–CRYPTO 2013: 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, pages 519–535. Springer, 2013. (Cited on page 9, 14.)

[DJM$^+$25] Nico Döttling, Abhishek Jain, Giulio Malavolta, Surya Mathialagan, and Vinod Vaikuntanathan. Simple and general counterexamples for private-coin evasive lwe. In *CRYPTO*, 2025. Available from `https://eprint.iacr.org/2025/374`. (Cited on page 3, 5, 8.)

[DQV$^+$21] Lalita Devadas, Willy Quach, Vinod Vaikuntanathan, Hoeteck Wee, and Daniel Wichs. Succinct lwe sampling, random polynomials, and obfuscation. In *Theory of Cryptography: 19th International Conference, TCC 2021, Raleigh, NC, USA, November 8–11, 2021, Proceedings, Part II 19*, pages 256–287. Springer, 2021. (Cited on page 4.)

[FFMV24] Danilo Francati, Daniele Friolo, Giulio Malavolta, and Daniele Venturi. Multi-key and multi-input predicate encryption (for conjunctions) from learning with errors. *Journal of Cryptology*, 37(3):24, 2024. (Cited on page 3, 7.)

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 169–178, 2009. (Cited on page 10.)

[GGG$^+$14] Shafi Goldwasser, S. Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. Multi-input functional encryption. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 578–602. Springer, Heidelberg, May 2014. (Cited on page 4, 6, 21, 72.)

[GGH$^+$16] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. *SIAM Journal on Computing*, 45(3):882–929, 2016. (Cited on page 13, 32, 72.)

[GGM84] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions (extended abstract). In *25th FOCS*, pages 464–479. IEEE Computer Society Press, October 1984. (Cited on page 26.)

[GKP$^+$13] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. How to run Turing machines on encrypted data. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 536–553. Springer, Heidelberg, August 2013. (Cited on page 3.)

[GKW17] Rishab Goyal, Venkata Koppula, and Brent Waters. Lockable obfuscation. In Chris Umans, editor, *58th FOCS*, pages 612–621. IEEE Computer Society Press, October 2017. (Cited on page 29.)

[GP21] Romain Gay and Rafael Pass. Indistinguishability obfuscation from circular security. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 736–749, 2021. (Cited on page 4.)

[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*, pages 89–98. ACM Press, October / November 2006. Available as Cryptology ePrint Archive Report 2006/309. (Cited on page 3.)

[GPSZ17] Sanjam Garg, Omkant Pandey, Akshayaram Srinivasan, and Mark Zhandry. Breaking the sub-exponential barrier in obfustopia. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 156–181. Springer, 2017. (Cited on page 9.)

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008. (Cited on page 22.)

[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, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 75–92. Springer, Heidelberg, August 2013. (Cited on page 10, 24.)

[GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In *STOC*, 2013. (Cited on page 3.)

[GVW15a] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits from LWE. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 503–523. Springer, Heidelberg, August 2015. (Cited on page 29.)

[GVW15b] Sergey Gorbunov, Vinod Vaikuntanathan, and Daniel Wichs. Leveled fully homomorphic signatures from standard lattices. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 469–477, 2015. (Cited on page 3, 20, 66.)

[GW20] Junqing Gong and Hoeteck Wee. Adaptively secure abe for dfa from k-lin and more. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 278–308. Springer, 2020. (Cited on page 3.)

[GWW19] Junqing Gong, Brent Waters, and Hoeteck Wee. Abe for dfa from k-lin. In *Advances in Cryptology–CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2019, Proceedings, Part II 39*, pages 732–764. Springer, 2019. (Cited on page 3.)

[HHY25] Tzu-Hsiang Huang, Wei-Hsiang Hung, and Shota Yamada. A note on obfuscation-based attacks on private-coin evasive LWE. Cryptology ePrint Archive, Paper 2025/421, 2025. (Cited on page 4, 5, 8.)

[HJL21] Samuel B. Hopkins, Aayush Jain, and Huijia Lin. Counterexamples to new circular security assumptions underlying iO. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part II*, volume 12826 of *LNCS*, pages 673–700, Virtual Event, August 2021. Springer, Heidelberg. (Cited on page 4, 12.)

[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*, 2025. (Cited on page 3, 4, 8, 9.)

[HLL23] Yao-Ching Hsieh, Huijia Lin, and Ji Luo. Attribute-Based Encryption for Circuits of Unbounded Depth from Lattices. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 415–434. IEEE, 2023. (Cited on page 3, 4, 5, 6, 8, 10, 13, 15, 24, 25, 48, 49, 53.)

[HSW14] Susan Hohenberger, Amit Sahai, and Brent Waters. Replacing a random oracle: Full domain hash from indistinguishability obfuscation. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 201–220. Springer, Heidelberg, May 2014. (Cited on page 7, 21, 74, 75.)

[JLL23] Aayush Jain, Huijia Lin, and Ji Luo. On the optimal succinctness and efficiency of functional encryption and attribute-based encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 479–510. Springer, 2023. (Cited on page 3.)

[JLMS19] Aayush Jain, Huijia Lin, Christian Matt, and Amit Sahai. How to leverage hardness of constant-degree expanding polynomials over r to build io. In *EUROCRYPT*, 2019. (Cited on page 4.)

[JLS21] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In Samir Khuller and Virginia Vassilevska Williams, editors, *53rd ACM STOC*, pages 60–73. ACM Press, June 2021. (Cited on page 4, 5, 6.)

[JLS22] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from LPN over $\mathbb{F}_p$, DLIN, and PRGs in NC^0. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part I*, volume 13275 of *LNCS*, pages 670–699. Springer, Heidelberg, May / June 2022. (Cited on page 4, 5.)

[KNTY19] Fuyuki Kitagawa, Ryo Nishimaki, Keisuke Tanaka, and Takashi Yamakawa. Adaptively secure and succinct functional encryption: Improving security and efficiency, simultaneously. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 521–551. Springer, Heidelberg, August 2019. (Cited on page 3.)

[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*, pages 669–684. ACM Press, November 2013. (Cited on page 26.)

[KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *Advances in Cryptology–EUROCRYPT 2008: 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings 27*, pages 146–162. Springer, 2008. (Cited on page 3.)

[LL20] Huijia Lin and Ji Luo. Compact adaptively secure ABE from k-Lin: Beyond NC^1 and towards NL. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part III*, volume 12107 of *LNCS*, pages 247–277. Springer, Heidelberg, May 2020. (Cited on page 3.)

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 700–718. Springer, Heidelberg, April 2012. (Cited on page 22.)

[QWW18] Willy Quach, Hoeteck Wee, and Daniel Wichs. Laconic function evaluation and applications. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 859–870. IEEE, 2018. (Cited on page 15.)

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM*, 56(6):34:1–34:40, 2009. (Cited on page 23.)

[RVV24] Seyoon Ragavan, Neekon Vafa, and Vinod Vaikuntanathan. Indistinguishability obfuscation from bilinear maps and lpn variants. *Cryptology ePrint Archive*, 2024. (Cited on page 4, 5.)

[SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, 2005. (Cited on page 4.)

[Tsa22] Rotem Tsabary. Candidate witness encryption from lattice techniques. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part I*, volume 13507 of *LNCS*, pages 535–559. Springer, Heidelberg, August 2022. (Cited on page 4.)

[VWW22] Vinod Vaikuntanathan, Hoeteck Wee, and Daniel Wichs. Witness encryption and null-IO from evasive LWE. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part I*, volume 13791 of *LNCS*, pages 195–221. Springer, Heidelberg, December 2022. (Cited on page 4, 5, 8, 20, 64, 65.)

[Wat12] Brent Waters. Functional encryption for regular languages. In *Annual Cryptology Conference*, pages 218–235. Springer, 2012. (Cited on page 3.)

[Wee05] Hoeteck Wee. On obfuscating point functions. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 523–532. ACM Press, May 2005. (Cited on page 8.)

[Wee22] Hoeteck Wee. Optimal broadcast encryption and CP-ABE from evasive lattice assumptions. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 217–241. Springer, Heidelberg, May / June 2022. (Cited on page 3, 4, 13, 23.)

[Wee24] Hoeteck Wee. Circuit abe with poly(depth,λ)-sized ciphertexts and keys from lattices. In *Advances in Cryptology – CRYPTO 2024: 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2024, Proceedings, Part III*, page 178–209. Springer-Verlag, 2024. (Cited on page 9.)

[Wee25] Hoeteck Wee. Almost optimal kp and cp-abe for circuits from succinct lwe. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 34–62. Springer, 2025. (Cited on page 9.)

[WW21] Hoeteck Wee and Daniel Wichs. Candidate obfuscation via oblivious LWE sampling. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part III*, volume 12698 of *LNCS*, pages 127–156. Springer, Heidelberg, October 2021. (Cited on page 4.)

[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, Part I*, volume 13747 of *LNCS*, pages 651–679. Springer, Heidelberg, November 2022. (Cited on page 23.)

[WZ17] Daniel Wichs and Giorgos Zirdelis. Obfuscating compute-and-compare programs under LWE. In Chris Umans, editor, *58th FOCS*, pages 600–611. IEEE Computer Society Press, October 2017. (Cited on page 29, 66.)

A Discussion on the Attack and the Fix for prFE construction (Section 2.2)

First we recall the construction.

- The setup algorithm samples matrices $\mathbf{A}_{\mathsf{att}}$ and $(\mathbf{B}, \mathbf{B}^{-1})$ of appropriate dimensions and outputs $\mathsf{mpk} := (\mathbf{A}_{\mathsf{att}}, \mathbf{B})$ and $\mathsf{msk} := \mathbf{B}^{-1}$. Here, $\mathbf{B}^{-1}$ is the trapdoor for $\mathbf{B}$ which allows to compute short preimages $\mathbf{B}^{-1}(\mathbf{U})$ for any target matrix $\mathbf{U}$.
- The encryptor on input $\mathbf{x}$ first samples a GSW secret key $\mathbf{s}$, where $\mathbf{s} = (\bar{\mathbf{s}}^{\mathsf{T}} \quad -1)^{\mathsf{T}}$ and a PRF seed $\mathsf{sd} \leftarrow \{0,1\}^{\lambda}$. It then computes a GSW ciphertext, $\mathbf{X} = \mathsf{hct}_{\mathbf{s}}(\mathbf{x}, \mathsf{sd})$, using public key $\mathbf{A}_{\mathsf{fhe}} = (\bar{\mathbf{A}}_{\mathsf{fhe}} \quad \bar{\mathbf{s}}^{\mathsf{T}}\bar{\mathbf{A}}_{\mathsf{fhe}} + \mathbf{e}_{\mathsf{fhe}}^{\mathsf{T}})^{\mathsf{T}}$ and randomness $\mathbf{R}$, – followed by a BGG^{+} encoding of $\mathbf{X}$ using randomness $\mathbf{s}$ as $\mathbf{c}_{\mathsf{att}}^{\mathsf{T}} := \mathbf{s}^{\mathsf{T}}(\mathbf{A}_{\mathsf{att}} - \mathbf{X} \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{att}}^{\mathsf{T}}$. It additionally computes $\mathbf{c}_{\mathbf{B}}^{\mathsf{T}} := \mathbf{s}^{\mathsf{T}}\mathbf{B} + \mathbf{e}_{\mathbf{B}}^{\mathsf{T}}$ and outputs the ciphertext $\mathsf{ct} = (\mathbf{c}_{\mathbf{B}}, \mathbf{c}_{\mathsf{att}}, \mathbf{X})$.
- The key generator on input $\mathsf{msk} = \mathbf{B}^{-1}$ and function f does the following.
  (a) Samples a nonce $\mathbf{r} \leftarrow \{0,1\}^{\lambda}$ and defines function $\mathrm{F}[f, \mathbf{r}]$, with f and $\mathbf{r}$ hardwired, as
  $$\mathrm{F}[f, \mathbf{r}](\mathbf{x}, \mathsf{sd}) = f(\mathbf{x}) \lfloor q/2 \rceil + \mathsf{PRF}(\mathsf{sd}, \mathbf{r}).$$
  It then computes the FHE evaluation circuit $\mathsf{VEval}_{\mathrm{F}}$ w.r.t. the function $\mathrm{F}[f, \mathbf{r}]$ (this can be computed using the knowledge of $\mathrm{F}[f, \mathbf{r}]$). Note that the circuit $\mathsf{VEval}_{\mathrm{F}}$ can be used to compute on a GSW ciphertext encoding an input, say $\mathbf{y}$, to recover a GSW ciphertext encoding $\mathrm{F}[f, \mathbf{r}](\mathbf{y})$.
  (b) Next, it computes the matrix $\mathbf{H}_{\mathbf{A}_{\mathsf{att}}}^{\mathrm{F}}$ for the circuit $\mathsf{VEval}_{\mathrm{F}}$ using the public matrix $\mathbf{A}_{\mathsf{att}}$. Recall that the matrix $\mathbf{H}_{\mathbf{A}_{\mathsf{att}}}^{\mathrm{F}}$ and $\mathbf{H}_{\mathbf{A}_{\mathsf{att}}, \mathbf{X}}^{\mathrm{F}}$ (which can be computed given $\mathsf{VEval}_{\mathrm{F}}$, $\mathbf{A}_{\mathsf{att}}$ and $\mathbf{X}$) will satisfy the relation
  $$(\mathbf{A}_{\mathsf{att}} - \mathbf{X} \otimes \mathbf{G})\mathbf{H}_{\mathbf{A}_{\mathsf{att}}, \mathbf{X}}^{\mathrm{F}} = \mathbf{A}_{\mathsf{att}}\mathbf{H}_{\mathbf{A}_{\mathsf{att}}}^{\mathrm{F}} - \mathsf{VEval}_{\mathrm{F}}(\mathbf{X}).$$
  (c) It sets $\mathbf{A}_{\mathrm{F}} = \mathbf{A}_{\mathsf{att}} \cdot \mathbf{H}_{\mathbf{A}_{\mathsf{att}}}^{\mathrm{F}}$, samples $\mathbf{K} \leftarrow \mathbf{B}^{-1}(\mathbf{A}_{\mathrm{F}})$ and outputs $\mathsf{sk}_f = (\mathbf{K}, \mathbf{r})$.
- The decryption on input $\mathsf{sk}_f = (\mathbf{K}, \mathbf{r})$ and $\mathsf{ct} = (\mathbf{c}_{\mathbf{B}}, \mathbf{c}_{\mathsf{att}}, \mathbf{X})$ work as follows.
  (a) It first computes the matrix $\mathbf{H}_{\mathbf{A}_{\mathsf{att}}, \mathbf{X}}^{\mathrm{F}}$ for the circuit $\mathsf{VEval}_{\mathrm{F}}$ using $\mathbf{A}_{\mathsf{att}}$ and $\mathbf{X}$.
  (b) Next, it computes $\mathbf{z} := \mathbf{c}_{\mathbf{B}}^{\mathsf{T}} \cdot \mathbf{K} - \mathbf{c}_{\mathsf{att}}^{\mathsf{T}} \cdot \mathbf{H}_{\mathbf{A}_{\mathsf{att}}, \mathbf{X}}^{\mathrm{F}}$, rounds $\mathbf{z}$ co-ordinate wise and output the most significant bits.

Next, we provide a high level outline of the attack.

The Attack. The adversary, given $\mathbf{c}_{\mathbf{B}}$, $\mathbf{c}_{\mathsf{att}}$, $\mathbf{X}$, and $\mathbf{K}$, computes $\mathbf{c}_{\mathbf{B}}^{\mathsf{T}} \cdot \mathbf{K} - \mathbf{c}_{\mathsf{att}}^{\mathsf{T}} \cdot \mathbf{H}_{\mathbf{A}_{\mathsf{att}}, \mathbf{X}}^{\mathrm{F}}$. Simplifying, she obtains
$$f(\mathbf{x}) \lfloor q/2 \rceil + \mathbf{e}_{\mathbf{B}}^{\mathsf{T}}\mathbf{K} + \mathsf{PRF}(\mathsf{sd}, \mathbf{r}) - (\mathbf{e}_{\mathsf{fhe}}^{\mathsf{T}}\mathbf{R}_{\mathrm{F}} + \mathbf{e}_{\mathsf{att}}^{\mathsf{T}}\mathbf{H}_{\mathbf{A}_{\mathsf{att}}, \mathbf{X}}^{\mathrm{F}})$$

By correctness, the adversary recovers $f(\mathbf{x})$ and can therefore strip it away to obtain $\mathsf{PRF}(\mathsf{sd}, \mathbf{r}) + \mathbf{e}_{\mathbf{B}}^{\top}\mathbf{K} - \mathbf{e}_{\mathsf{fhe}}^{\top}\mathbf{R}_{\mathrm{F}} - \mathbf{e}_{\mathsf{att}}^{\top}\mathbf{H}_{\mathbf{A}_{\mathsf{att}}, \mathbf{X}}^{\mathrm{F}}$, as described in Equation (2). Now, in the proof, the error term $\mathbf{e}_{\mathbf{B}}^{\top}\mathbf{K}$ is replaced by i.i.d error $\mathbf{e}_{\mathbf{P}}$ and is used to break any correlation between $\mathsf{PRF}(\mathsf{sd}, \mathbf{r})$ and $\mathbf{e}_{\mathsf{fhe}}^{\top}\mathbf{R}_{\mathrm{F}} - \mathbf{e}_{\mathsf{att}}^{\top}\mathbf{H}_{\mathbf{A}_{\mathsf{att}}, \mathbf{X}}^{\mathrm{F}}$. This allows us to prove the pre-condition based on just plain LWE.

However, in the real world, $\mathbf{e}_{\mathbf{B}}^{\top}\mathbf{K}$ cannot be used to break the dependence between the above two terms. Then, by choosing the circuit implementing F in some contrived way, the authors set it up so that $\mathsf{PRF}(\mathsf{sd}, \mathbf{r})$ and $\mathbf{e}_{\mathsf{fhe}}^{\top}\mathbf{R}_{\mathrm{F}}$ are correlated, and in particular cancel each other modulo 2. Note that these terms are small, and do not wraparound modulo q, so computing mod 2 is well defined. Now we are left with $\mathbf{e}_{\mathbf{B}}^{\mathsf{T}}\mathbf{K} + \mathbf{e}_{\mathsf{att}}^{\mathsf{T}}\mathbf{H}_{\mathbf{A}_{\mathsf{att}}, \mathbf{X}}^{\mathrm{F}}$ – but these are linear equations with known coefficients, in the error terms of the original encodings. Using sufficiently many equations, the adversary can easily recover the error terms. On the other hand, had the term $\mathbf{e}_{\mathbf{B}}^{\mathsf{T}}\mathbf{K}$ been truly random, such a system of equations would not admit any solution. This leads to a distinguishing strategy.

Modifying the Construction. We describe an approach that helps us break the problematic correlation even if the circuit implementation is chosen in a contrived manner – our idea is to use modulus reduction get rid of the problematic error terms involving $\mathbf{e}_{\mathsf{fhe}}^{\mathsf{T}}\mathbf{R}_{\mathrm{F}}$ so that the correlation is destroyed. Informally, we fix a rounding constant $M \in \mathbb{Z}$ such that $\left\|\mathbf{e}_{\mathsf{fhe}}^{\mathsf{T}}\mathbf{R}_{\mathrm{F}}\right\| < M$ which implies $\left\lfloor(\mathbf{e}_{\mathsf{fhe}}^{\mathsf{T}}\mathbf{R}_{\mathrm{F}})/M\right\rceil = 0$. This gets rid of the problematic error, replacing it with rounding error which is uncorrelated with the PRF seed. For concreteness, we elaborate the changes that must be made to our prFE construction to incorporate the above fix.

1. In setup algorithm, we output M as a part of mpk. The encryption algorithm remains the same.
2. The key generation algorithm has the following changes.
   - We parse $\mathrm{F}[f, \mathbf{r}](\mathbf{x}, \mathsf{sd}) = f(\mathbf{x}) \lfloor q/2 \rceil + \mathsf{PRF}(\mathsf{sd}, \mathbf{r}) = M \cdot f_{\mathsf{high}}(\mathbf{x}, \mathsf{sd}) + f_{\mathsf{low}}(\mathbf{x}, \mathsf{sd})$, where $f_{\mathsf{high}}(\mathbf{x}, \mathsf{sd}) \in [0, q/M]^{\ell}$ and $f_{\mathsf{low}}(\mathbf{x}, \mathsf{sd}) \in [0, M-1]^{\ell}$. Next we define functions $\mathrm{F}_{\mathsf{high}} := M \cdot f_{\mathsf{high}}$ and $\mathrm{F}_{\mathsf{low}} := M \cdot f_{\mathsf{low}}$, which on input $(\mathbf{x}, \mathsf{sd})$ outputs $M \cdot f_{\mathsf{high}}(\mathbf{x}, \mathsf{sd})$ and $M \cdot f_{\mathsf{low}}(\mathbf{x}, \mathsf{sd})$, respectively.
   - Next, it computes circuits $\mathsf{VEval}_{\mathsf{high}}$ and $\mathsf{VEval}_{\mathsf{low}}$ for the functions $\mathrm{F}_{\mathsf{high}}$ and $\mathrm{F}_{\mathsf{low}}$, respectively and then uses these circuits to compute the matrices $\mathbf{H}_{\mathbf{A}_{\mathsf{att}}}^{\mathrm{F}_{\mathsf{high}}}$ and $\mathbf{H}_{\mathbf{A}_{\mathsf{att}}}^{\mathrm{F}_{\mathsf{high}}}$ (as described in the previous sketch).
   - It sets
   $$\mathbf{A}_{\mathrm{F}} = M \cdot \left\lfloor \frac{\mathbf{A}_{\mathsf{att}} \cdot \mathbf{H}_{\mathbf{A}_{\mathsf{att}}}^{\mathrm{F}_{\mathsf{high}}}}{M} \right\rceil + \left\lfloor \frac{\mathbf{A}_{\mathsf{att}} \cdot \mathbf{H}_{\mathbf{A}_{\mathsf{att}}}^{\mathrm{F}_{\mathsf{low}}}}{M} \right\rceil$$
   and outputs $\mathsf{sk}_f = (\mathbf{K}, \mathbf{r})$ where $\mathbf{K} = \mathbf{B}^{-1}(\mathbf{A}_{\mathrm{F}})$.
3. The decryption algorithm is the same except that we compute $\mathbf{z}$ differently as
$$\mathbf{z} := \mathbf{c}_{\mathbf{B}}^{\mathsf{T}} \cdot \mathbf{K} - \left(M \cdot \left\lfloor \frac{\mathbf{c}_{\mathsf{att}}^{\mathsf{T}} \cdot \mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}}}{M} \right\rceil + \left\lfloor \frac{\mathbf{c}_{\mathsf{att}}^{\mathsf{T}} \cdot \mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{low}}}}{M} \right\rceil \right)$$

We expand the correctness of the scheme to see how the above changes helps us get rid of the problematic noise terms while decryption. Observe that

$$\begin{aligned}
\mathbf{c}_{\mathsf{att}}^{\mathsf{T}} \cdot \mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}} &= (\mathbf{s}^{\mathsf{T}}(\mathbf{A}_{\mathsf{att}} - \mathsf{bits}(1, \mathbf{X}) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{att}}^{\mathsf{T}})\mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}} \\
&= \mathbf{s}^{\mathsf{T}}\mathbf{A}_{\mathsf{high}} - \mathrm{F}_{\mathsf{high}}(\mathbf{x}, \mathsf{sd}) + \mathbf{e}_{\mathsf{fhe}}^{\mathsf{T}}\mathbf{R}_{\mathsf{high}} + \mathbf{e}_{\mathsf{att}}^{\mathsf{T}}\mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}} \\
\implies \left\lfloor \frac{\mathbf{c}_{\mathsf{att}}^{\mathsf{T}} \cdot \mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}}}{M} \right\rceil &= \left\lfloor \frac{\mathbf{s}^{\mathsf{T}}\mathbf{A}_{\mathsf{high}} - M \cdot f_{\mathsf{high}}(\mathbf{x}, \mathsf{sd}) + \mathbf{e}_{\mathsf{fhe}}^{\mathsf{T}}\mathbf{R}_{\mathsf{high}} + \mathbf{e}_{\mathsf{att}}^{\mathsf{T}}\mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}}}{M} \right\rceil \\
&= \left\lfloor \frac{\mathbf{s}^{\mathsf{T}}\mathbf{A}_{\mathsf{high}} + \mathbf{e}_{\mathsf{fhe}}^{\mathsf{T}}\mathbf{R}_{\mathsf{high}} + \mathbf{e}_{\mathsf{att}}^{\mathsf{T}}\mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}}}{M} \right\rceil - f_{\mathsf{high}}(\mathbf{x}, \mathsf{sd}) \\
&= \mathbf{s}^{\mathsf{T}} \left\lfloor \frac{\mathbf{A}_{\mathsf{high}}}{M} \right\rceil - f_{\mathsf{high}}(\mathbf{x}, \mathsf{sd}) \quad \text{w.h.p.}
\end{aligned}$$

where we set $\left\|\mathbf{e}_{\mathsf{fhe}}^{\mathsf{T}}\mathbf{R}_{\mathrm{F}} + \mathbf{e}_{\mathsf{att}}^{\mathsf{T}}\mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}}\right\| \ll M$ such that the last equation in the above will hold with high probability. Similarly, we get $\left\lfloor \frac{\mathbf{c}_{\mathsf{att}}^{\mathsf{T}} \cdot \mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{low}}}}{M} \right\rceil = \mathbf{s}^{\mathsf{T}} \left\lfloor \frac{\mathbf{A}_{\mathsf{low}}}{M} \right\rceil - f_{\mathsf{low}}(\mathbf{x}, \mathsf{sd})$ w.h.p. The rest of correctness follows from the same argument as in the previous sketch. Note that the final error obtained now is $\mathsf{PRF}(\mathsf{sd}) + \mathsf{err}$ where (please see Equation (17))

$$\begin{aligned}
\mathsf{err} &= M \cdot \mathbf{e}_{\mathbf{s},\mathsf{high}}^{\mathsf{T}} + \mathbf{e}_{\mathbf{s},\mathsf{low}}^{\mathsf{T}} + M \cdot \mathsf{err}_{\mathsf{high}} + \mathsf{err}_{\mathsf{low}} \\
&= M \cdot \left(\mathbf{s}^{\mathsf{T}} \left\lfloor \frac{\mathbf{A}_{\mathsf{high}}}{M} \right\rceil - \left\lfloor \mathbf{s}^{\mathsf{T}} \frac{\mathbf{A}_{\mathsf{high}}}{M} \right\rceil \right) + \left(\mathbf{s}^{\mathsf{T}} \left\lfloor \frac{\mathbf{A}_{\mathsf{low}}}{M} \right\rceil - \left\lfloor \mathbf{s}^{\mathsf{T}} \frac{\mathbf{A}_{\mathsf{low}}}{M} \right\rceil \right) + M \cdot \mathsf{err}_{\mathsf{high}} + \mathsf{err}_{\mathsf{low}}
\end{aligned}$$

where $\mathsf{err}_{\mathsf{high}}, \mathsf{err}_{\mathsf{low}} \in \{0,1\}^{\ell}$ are rounding errors and matrices $\mathbf{A}_{\mathsf{high}}, \mathbf{A}_{\mathsf{low}}$ are publicly computable matrices. Notably, the FHE error $\mathbf{e}_{\mathsf{fhe}}$, which is the problematic term that enabled the attack, does not appear here. We conjecture flooding for appropriately defined sizes.

B Pseudorandom FE with Stronger Security

In this section, we consider a fine-grained variant of Definition 4.2, which we term as κ-prCT security, which is useful in Section 7.

Definition B.1 (Non-uniform κ-prCT Security). For a prFE scheme for function family $\{\mathcal{F}_{\mathsf{prm}} = \{f : \mathcal{X}_{\mathsf{prm}} \to \mathcal{Y}_{\mathsf{prm}}\}\}_{\mathsf{prm}}$, parameter $\mathsf{prm} = \mathsf{prm}(\lambda)$, and function $\kappa \stackrel{\text{def}}{=} \kappa(\lambda)$ of λ, let $\mathsf{Samp} = \{\mathsf{Samp}_\lambda\}_\lambda$ be a non-uniform polynomial-time algorithm that on input 1^λ, outputs

$$(f_1, \ldots, f_{Q_{\mathsf{key}}}, x_1, \ldots, x_{Q_{\mathsf{msg}}}, \mathsf{aux} \in \{0,1\}^*)$$

where Q_{key} is the number of key queries, Q_{msg} is the number of message queries, and $f_i \in \mathcal{F}_{\mathsf{prm}}$ $x_j \in \mathcal{X}_{\mathsf{prm}}$ for all $i \in [Q_{\mathsf{key}}], j \in [Q_{\mathsf{msg}}]$.
For non-uniform adversaries $\mathcal{A}_0 := \{\mathcal{A}_{0,\lambda}\}_\lambda$ and $\mathcal{A}_1 := \{\mathcal{A}_{1,\lambda}\}_\lambda$, we define the following advantage functions:

$$\begin{aligned}\mathsf{Adv}^{\mathrm{PRE}}_{\mathcal{A}_0}(\lambda) \stackrel{\text{def}}{=} &\Pr\Big[\mathcal{A}_0\Big(\ \mathsf{aux},\ \{f_i,\ f_i(x_j)\}_{i\in[Q_{\mathsf{key}}], j\in[Q_{\mathsf{msg}}]}\Big) = 1\Big] \\ &- \Pr\Big[\mathcal{A}_0(\ \mathsf{aux},\ \{f_i,\ \Delta_{i,j} \leftarrow \mathcal{Y}_{\mathsf{prm}}\}_{i\in[Q_{\mathsf{key}}], j\in[Q_{\mathsf{msg}}]}) = 1\Big]\end{aligned}$$

$$\begin{aligned}\mathsf{Adv}^{\mathrm{POST}}_{\mathcal{A}_1}(\lambda) \stackrel{\text{def}}{=} &\Pr\Big[\mathcal{A}_1(\mathsf{mpk},\ \mathsf{aux},\ \{f_i,\ \mathsf{ct}_j \leftarrow \mathsf{Enc}(\mathsf{mpk}, x_j),\ \mathsf{sk}_{f_i}\}_{i\in[Q_{\mathsf{key}}], j\in[Q_{\mathsf{msg}}]}) = 1\Big] \\ &- \Pr\Big[\mathcal{A}_1(\mathsf{mpk},\ \mathsf{aux},\ \{f_i,\ \delta_j \leftarrow \mathcal{CT},\ \mathsf{sk}_{f_i}\}_{i\in[Q_{\mathsf{key}}], j\in[Q_{\mathsf{msg}}]}) = 1\Big]\end{aligned}$$

where $(f_1, \ldots, f_{Q_{\mathsf{key}}}, x_1, \ldots, x_{Q_{\mathsf{msg}}}, \mathsf{aux} \in \{0,1\}^*) \leftarrow \mathsf{Samp}(1^\lambda), (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{prm})$ and $\mathcal{CT}$ is the ciphertext space. We say that a prFE scheme for function family $\mathcal{F}_{\mathsf{prm}}$ is secure in the non-uniform κ setting with respect to the sampler class $\mathcal{SC}$ if for every sampler $\mathsf{Samp} \in \mathcal{SC}$ and an adversary $\mathcal{A}_1$ such that $\mathrm{Size}(\mathsf{Samp}) \le \mathrm{poly}(\lambda')$ and $\mathrm{Size}(\mathcal{A}_1) \le \mathrm{poly}(\kappa)$ for $\lambda' \le \kappa$, there exists another adversary $\mathcal{A}_0$ such that

$$\mathsf{Adv}^{\mathrm{PRE}}_{\mathcal{A}_0}(\lambda) \ge \mathsf{Adv}^{\mathrm{POST}}_{\mathcal{A}_1}(\lambda)/Q(\lambda') - \mathsf{negl}(\kappa) \tag{53}$$

and $\mathrm{Size}(\mathcal{A}_0) \le \mathrm{Size}(\mathcal{A}_1) \cdot Q(\lambda')$ for some polynomial $Q(\cdot)$.

Remark B.2 (Comparison between Definition B.1 and Definition 4.2). We remark that Definition B.1 strengthens Definition 4.2 in two aspects. First of all, it considers non-uniform adversaries instead of uniform adversaries. Secondly, it is parameterized by κ and the additive term $\mathsf{negl}(\lambda)$ that appears in Equation (11) is replaced by $\mathsf{negl}(\kappa)$ in Equation (53). By taking κ asymptotically larger than λ (e.g., $\kappa := \lambda^\lambda$), we can make the additive term $\mathsf{negl}(\kappa)$ much smaller than $\mathsf{negl}(\lambda)$. We note that these changes are introduced to prove the security of our prMIFE in Section 7. We refer to Remark 7.13 for the discussion on why these changes are necessary for the security proof there.

Next, we propose a variant of the evasive LWE assumption which will be used to prove security of our prFE scheme as per Definition B.1. This variant strengthens Assumption 3.6 in that it considers non-uniform samplers and replaces the negligible term in Equation (8) with negligible function in another parameter κ, which can be much larger than λ. The reason why we need this strengthened version of the assumption is that we need prFE to satisfy stronger security notion than prCT security that we call non-uniform κ-prCT security for the application to prMIFE.

Assumption B.3 (Non-Uniform κ-Evasive LWE). Let $n, m, t, m', q, \lambda \in \mathbb{N}$ be parameters defined as in Assumption 3.6 and $\mathsf{Samp} = \{\mathsf{Samp}_\lambda\}_\lambda$ be a non-uniform sampler that takes as input 1^λ and outputs $\mathbf{S}, \mathbf{P}, \mathsf{aux}$ as in Assumption 3.6. For non-uniform adversaries $\mathcal{A}_0 = \{\mathcal{A}_{0,\lambda}\}_\lambda$ and $\mathcal{A}_1 = \{\mathcal{A}_{1,\lambda}\}_\lambda$, we define the advantage functions $\mathsf{Adv}^{\mathrm{PRE}}_{\mathcal{A}_0}(\lambda)$ and

$\mathsf{Adv}_{\mathcal{A}_1}^{\mathrm{POST}}(\lambda)$ as in Equation (6) and Equation (7), respectively. For a function $\kappa := \kappa(\lambda)$ of the security parameter λ, we say that the non-uniform κ-evasive LWE assumption with respect to the sampler class $\mathcal{SC}$ holds if for every non-uniform sampler $\mathsf{Samp} \in \mathcal{SC}$ and a non-uniform adversary $\mathcal{A}_1$ such that $\mathrm{Size}(\mathsf{Samp}) \leq \mathrm{poly}(\lambda')$ and $\mathrm{Size}(\mathcal{A}_1) \leq \mathrm{poly}(\kappa)$ for $\lambda'(\lambda) \leq \kappa(\lambda)$, there exists another non-uniform adversary $\mathcal{A}_0$ and a polynomial $Q(\cdot)$ such that

$$\mathsf{Adv}_{\mathcal{A}_0}^{\mathrm{PRE}}(\lambda) \geq \mathsf{Adv}_{\mathcal{A}_1}^{\mathrm{POST}}(\lambda)/Q(\lambda') - \mathsf{negl}(\kappa) \quad \text{and} \quad \mathrm{Size}(\mathcal{A}_0) \leq Q(\lambda') \cdot \mathrm{Size}(\mathcal{A}_1).$$

Note that in the case λ' is superpolynomial in λ, $\mathtt{Samp}(1^\lambda)$ outputs $\mathbf{S}$, $\mathbf{P}$, and aux whose sizes are polynomial in λ' and thus superpolynomial in λ. We require the above assumption with $\kappa = 2^{\mathrm{poly}(\lambda)}$ in our construction.

The following lemma is an adaptation of Lemma 3.8 for the stronger version of evasive LWE assumption defined in Assumption B.3. The proof is almost the same as that for Lemma 3.8. We provide it here for completeness.

Lemma B.4. Let $n, m, t, m', q, \lambda \in \mathbb{N}$ be parameters defined as in Assumption 3.6 and $\mathsf{Samp} = \{\mathsf{Samp}_\lambda\}_\lambda$ be a non-uniform sampler that takes as input 1^λ and outputs $\mathbf{S}, \mathsf{aux} = (\mathsf{aux}_1, \mathsf{aux}_2)$, and $\mathbf{P}$ as in Lemma 3.8. For a non-uniform adversaries $\mathcal{A}$, we define the advantage functions $\mathsf{Adv}_{\mathcal{A}}^{\mathrm{PRE}'}(\lambda)$ and $\mathsf{Adv}_{\mathcal{A}}^{\mathrm{POST}'}(\lambda)$ as in Equation (9) and Equation (10), respectively. Then, for a function $\kappa := \kappa(\lambda)$ of the security parameter λ, under the non-uniform κ-evasive LWE assumption (Assumption B.3), if $\mathrm{Size}(\mathsf{Samp}) \leq \mathrm{poly}(\lambda')$ and $\mathrm{Size}(\mathcal{A}_1) \leq \mathrm{poly}(\kappa)$ for $\lambda'(\lambda) \leq \kappa(\lambda)$, there exists another non-uniform adversary $\mathcal{A}_0$ and a polynomial $Q(\cdot)$ such that

$$\mathsf{Adv}_{\mathcal{A}_0}^{\mathrm{PRE}'}(\lambda) \geq \mathsf{Adv}_{\mathcal{A}_1}^{\mathrm{POST}'}(\lambda)/Q(\lambda') - \mathsf{negl}(\kappa) \quad \text{and} \quad \mathrm{Size}(\mathcal{A}_0) \leq Q(\lambda') \cdot \mathrm{Size}(\mathcal{A}_1).$$

Proof. Let us consider an adversary $\mathcal{A}_1$ and a sampler Samp with size being polynomial in κ and λ' respectively and $\epsilon = \mathsf{Adv}_{\mathcal{A}_1}^{\mathrm{POST}'}$. Then, the same adversary is able to distinguish either (1) $(\mathbf{B}, \mathbf{SB} + \mathbf{E}, \mathbf{K}, \mathsf{aux}_1, \mathsf{aux}_2)$ from $(\mathbf{B}, \mathbf{C}_0, \mathbf{K}, \mathsf{aux}_1, \mathsf{aux}_2)$ with advantage at least $\epsilon/2$ or (2) $(\mathbf{B}, \mathbf{C}_0, \mathbf{K}, \mathsf{aux}_1, \mathsf{aux}_2)$ from $(\mathbf{B}, \mathbf{C}_0, \mathbf{K}, \mathbf{c}, \mathsf{aux}_2)$ with advantage at least $\epsilon/2$. If the latter is the case, then we can obtain an adversary $\mathcal{A}_0$ that distinguishes $(\mathbf{B}, \mathbf{SB} + \mathbf{E}, \mathbf{SP} + \mathbf{E}', \mathsf{aux}_1, \mathsf{aux}_2)$ from $(\mathbf{B}, \mathbf{C}_0, \mathbf{C}', \mathbf{c}, \mathsf{aux}_2)$ with advantage $\epsilon/2$. This can be seen by observing that $\mathcal{A}_1$ can be turned into an adversary that distinguishes $(\mathsf{aux}_1, \mathsf{aux}_2)$ from $(\mathbf{c}, \mathsf{aux}_2)$ and then turned into an adversary that distinguishes $(\mathbf{B}, \mathbf{C}_0, \mathbf{K}, \mathsf{aux}_1, \mathsf{aux}_2)$ from $(\mathbf{B}, \mathbf{C}_0, \mathbf{K}, \mathbf{c}, \mathsf{aux}_2)$ by sampling $(\mathbf{B}, \mathbf{C}_0, \mathbf{K})$ by itself, where we sample $\mathbf{B}$ with the corresponding trapdoor and then sample $\mathbf{K}$ using it. We therefore assume that the former is the case. Then, by invoking the non-uniform κ-evasive LWE with respect to the sampler Samp, we obtain another adversary $\mathcal{A}_0$ whose size is bounded by $Q(\lambda') \cdot \mathrm{Size}(\mathcal{A}_1)$ and distinguishing advantage against $(\mathbf{B}, \mathbf{SB} + \mathbf{E}, \mathbf{SP} + \mathbf{E}', \mathsf{aux}_1, \mathsf{aux}_2)$ and $(\mathbf{B}, \mathbf{C}_0, \mathbf{C}', \mathsf{aux}_1, \mathsf{aux}_2)$ is at least $\epsilon/2Q(\lambda')$. Then, $\mathcal{A}_0$ is able to distinguish either (1) $(\mathbf{B}, \mathbf{SB} + \mathbf{E}, \mathbf{SP} + \mathbf{E}', \mathsf{aux}_1, \mathsf{aux}_2)$ from $(\mathbf{B}, \mathbf{C}_0, \mathbf{C}', \mathbf{c}, \mathsf{aux}_2)$ with advantage at least $\epsilon/4Q(\lambda')$ or (2) $(\mathbf{B}, \mathbf{C}_0, \mathbf{C}', \mathbf{c}, \mathsf{aux}_2)$ from $(\mathbf{B}, \mathbf{C}_0, \mathbf{C}', \mathsf{aux}_1, \mathsf{aux}_2)$ with advantage at least $\epsilon/4Q(\lambda')$. If the former is the case, we are done. If the latter is the case, we are still able to convert it into a distinguisher against $(\mathbf{B}, \mathbf{SB} + \mathbf{E}, \mathbf{SP} + \mathbf{E}', \mathsf{aux}_1, \mathsf{aux}_2)$ and $(\mathbf{B}, \mathbf{C}_0, \mathbf{C}', \mathbf{c}, \mathsf{aux}_2)$ by the similar argument to the above. $\square$

Now we explain how to extend construction in Section 4 to achieve strengthened security notion for pseudorandom FE that we call non-uniform κ-prCT security (Definition B.1), which is required for many of our applications.

B.1 Proof for Non-Uniform κ-prCT Security

Theorem B.5. Let $\kappa = 2^{\lambda^c}$ for some constant c. Assuming non-uniform κ-evasive LWE (Assumption B.3), subexponentially secure PRF against non-uniform adversary, and non-uniform sub-exponential LWE (Assumption 3.5), there exists a prFE scheme satisfying κ-prCT security as per Definition B.1.

Proof. We prove that the construction in Section 4.2 with λ being replaced by appropriately chosen $\Lambda = \mathrm{poly}(\lambda)$ satisfies the security notion. The reason why we need this scaled version of the security parameter is that we have to consider an adversary whose running time can be κ, which is exponential in λ. In particular, in the security proof, we require LWE and PRF to be secure even against an adversary that runs polynomial time in κ. To handle such an adversary, we rely on the subexponential security of LWE and PRF. By our assumption, there exists $0 < \delta < 1$ such that there is no adversary with size 2^{λ^δ} and distinguishing advantage $2^{-\lambda^\delta}$ against LWE and PRF for all sufficiently large λ. To satisfy the requirement, we generate PRF and LWE instances with respect to a larger security parameter Λ that satisfies $2^{\Lambda^\delta} \geq \kappa^{\omega(1)}$. An example choice of the parameter would be $\Lambda := \lambda^{(c+1)/\delta}$.

The overall structure of the proof is the same as that of Theorem 4.6.

We start with a sampler $\mathsf{Samp}_{\mathsf{prFE}}$ and an adversary $\mathcal{A}_1$ satisfying $\mathrm{Size}(\mathsf{Samp}_{\mathsf{prFE}}) \leq \mathrm{poly}(\lambda')$ and $\mathrm{Size}(\mathcal{A}_1) \leq \mathrm{poly}(\kappa)$ for $\lambda' < \kappa$.[15] We denote the size of $\mathcal{A}_1$ by t and the distinguishing advantage for the distributions in Equation (19) by ϵ. Assuming non-uniform κ-evasive LWE with respect to Samp defined from $\mathsf{Samp}_{\mathsf{prFE}}$ as in the proof of Theorem 4.6, we obtain an adversary $\mathcal{A}_0$ whose size is $Q(\lambda')t$ and the distinguishing advantage against the distributions in Equation (22) is $\epsilon/Q(\lambda') - \mathsf{negl}(\kappa)$ for some polynomial Q by applying Lemma B.4. We then consider the same sequence of hybrids as that for the proof of Theorem 4.6. Note that here, the security parameter for the construction λ is replaced by Λ and Q_{msg} and Q_{key} are bounded by $\mathrm{poly}(\lambda')$, since the size of the sampler is $\mathrm{poly}(\lambda')$. By the definition of the hybrids, the adversary has the distinguishing advantage $\epsilon/Q(\lambda') - \mathsf{negl}(\kappa)$ for Hyb_0 and Hyb_8. Furthermore, we argue that the distinguishing advantage between Hyb_0 and Hyb_7 is only $\mathsf{negl}(\kappa)$. We inspect this in the following:

- The changes from Hyb_0 to Hyb_1, from Hyb_2 to Hyb_3, from Hyb_4 to Hyb_5, and from Hyb_6 to Hyb_7 are statistical, where each statistical difference is bounded by $\mathrm{poly}(\lambda')/2^{-\Lambda}$. We have $\mathrm{poly}(\lambda')/2^{-\Lambda} \leq \mathrm{poly}(\kappa)/2^{-\Lambda} = \mathsf{negl}(\kappa)$ by our choice of Λ.

- The change from Hyb_1 to Hyb_2 is computational, which is dependent on the hardness of LWE. Since the size of $\mathcal{A}_0$ is bounded by $\mathrm{poly}(\kappa)$, the distinguishing advantage between Hyb_1 and Hyb_2 should be bounded by $\mathsf{negl}(\kappa)$ by the subexponential hardness of LWE and by our choice of Λ.

- The change from Hyb_3 to Hyb_4 is computational, which is dependent on the security of PRF. Since the size of $\mathcal{A}_0$ is bounded by $\mathrm{poly}(\kappa)$, the distinguishing advantage between Hyb_3 and Hyb_4 should be bounded by $\mathsf{negl}(\kappa)$ by the subexponential security of PRF.

- The changes from Hyb_5 to Hyb_6 is conceptual and thus they are equivalent.

We therefore conclude that the distinguishing advantage of $\mathcal{A}_0$ against Hyb_7 and Hyb_8 should be $\epsilon/Q(\lambda') - \mathsf{negl}(\kappa)$. Then, from $\mathcal{A}_0$, it is straightforward to extract a distinguisher $\mathcal{A}_0'$ against the distributions in Equation (20) with the same advantage and almost the same size. This concludes the proof of the theorem. □

Theorem B.6. Let $\kappa = 2^{\lambda^c}$ for some constant c. Assuming non-uniform κ-evasive LWE (Assumption B.3), subexponentially secure PRF against non-uniform adversary, and non-uniform sub-exponential LWE (Assumption 3.5), there exists a prFE scheme for function class $\mathcal{F}_{\mathrm{L}(\lambda),\ell(\lambda),\mathsf{dep}(\lambda)} = \{f : \{0,1\}^{\mathrm{L}} \to \{0,1\}^{\ell}\}$ satisfying κ-prCT security as per Definition B.1 with efficiency

$$|\mathsf{mpk}| = \mathrm{L} \cdot \mathrm{poly}(\mathsf{dep}, \lambda), \quad |\mathsf{sk}_f| = \ell \cdot \mathrm{poly}(\mathsf{dep}, \lambda), \quad |\mathsf{ct}| = \mathrm{L} \cdot \mathrm{poly}(\mathsf{dep}, \lambda).$$

where $\mathsf{dep} = \mathrm{poly}(\lambda)$ is the depth bound on the functions supported by the scheme.

C FE for Pseudorandom Functionalities with Unbounded Depth

In this section we provide our construction of a functional encryption scheme for pseudorandom functionalities for circuit class $\mathcal{C}_{\mathrm{L}(\lambda)} = \{C : \{0,1\}^{\mathrm{L}} \to \{0,1\}\}$, which consists of circuits with unbounded polynomial depth, using our prFE scheme for bounded depth (Section 4.2) and a blind garbling scheme.

[15]Here, we deviate from our convention that the adversary runs in time polynomial in its input length. The input length of $\mathcal{A}_1$ is $\mathrm{poly}(\lambda')$, but its running time is $\mathrm{poly}(\kappa)$, which may be super-polynomial in λ'.

C.1 Construction.

Ingredients. Below, we list the ingredients for our construction.

1. A blind garbled circuit scheme $\mathsf{bGC} = (\mathsf{bGC.Eval}, \mathsf{bGC.Garble}, \mathsf{bGC.SIM})$ scheme for circuit class $\mathcal{C}_{\mathrm{L}(\lambda)} = \{C : \{0,1\}^{\mathrm{L}} \rightarrow \{0,1\}\}$. We use the decomposability property (Definition 3.18) of the bGC scheme for our construction, i.e., for $C \in \mathcal{C}_{\mathrm{L}}$ we write $\mathsf{bGC.Garble} = (\{\mathsf{bGC.Garble}_i\}_{i\in[|C|]}, \mathsf{bGC.Garble}_{\mathsf{inp}})$.
   We assume that if $\widetilde{C}_i \leftarrow \mathsf{bGC.Garble}_i(1^\lambda, C_i; \mathsf{st})$ and $\{\mathsf{lab}_{j,b}\}_{j\in[\mathrm{L}],b\in\{0,1\}} \leftarrow \mathsf{bGC.Garble}_{\mathsf{inp}}(1^\lambda, 1^{\mathrm{L}}; \mathsf{st})$, then $|\widetilde{C}_i| = |\mathsf{lab}_{\mathbf{x}}| = \{0,1\}^{\ell_{\mathsf{bGC}}}$ where $\mathsf{lab}_{\mathbf{x}} = (\mathsf{lab}_{1,x_1}, \ldots, \mathsf{lab}_{\mathrm{L},x_{\mathrm{L}}})$ and x_j denotes the j-th bit of $\mathbf{x}$ for $j \in [\mathrm{L}]$. This can be ensured using appropriate padding. We use $\mathcal{CT}_{\mathsf{SIM}}$ to denote the output space of bGC.SIM. Here $\mathcal{CT}_{\mathsf{SIM}} = \{0,1\}^{(|C|+1)\ell_{\mathsf{bGC}}}$
2. A bounded depth FE scheme for pseudorandom functionality $\mathsf{BD\text{-}prFE} = (\mathsf{BD.Setup}, \mathsf{BD.KeyGen}, \mathsf{BD.Enc}, \mathsf{BD.Dec})$. for circuit class $\mathcal{C}_{\mathrm{L}'(\lambda), d_{\mathsf{BD}}(\lambda), \ell'(\lambda)}$ consisting of circuits with input length $\mathrm{L}'(\lambda) = \mathrm{L} + \lambda$, maximum depth $d_{\mathsf{BD}}(\lambda) = O(\lambda)$ and output length $\ell'(\lambda) = \ell_{\mathsf{bGC}}$. We denote the ciphertext space of the scheme by $\mathcal{CT}_{\mathsf{BD}} = \{0,1\}^{\ell_{\mathsf{bGC}}}$.
3. A PRF function $\mathsf{PRF} : \{0,1\}^\lambda \times \{0,1\}^\lambda \rightarrow \{0,1\}^{\mathsf{R}_{\mathsf{bGC}}}$ where $\mathsf{R}_{\mathsf{bGC}}$ is the length of randomness used in bGC.Garble. We assume that PRF can be computed by a circuit of depth at most $d_{\mathsf{BD}} = O(\lambda)$.

Next, we describe our construction for unbounded depth prFE scheme $\mathsf{prFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ with ciphertext space $\mathcal{CT}_{\mathsf{prFE}}$.

$\mathsf{Setup}(1^\lambda, 1^{\mathrm{L}}) \rightarrow (\mathsf{mpk}, \mathsf{msk})$. The setup algorithm does the following.

- Run $(\mathsf{BD.msk}, \mathsf{BD.mpk}) \leftarrow \mathsf{BD.Setup}(1^\lambda, 1^{\mathrm{L}'})$.
- Set $\mathsf{msk} = \mathsf{BD.msk}$[16] and $\mathsf{mpk} = \mathsf{BD.mpk}$. Output $(\mathsf{msk}, \mathsf{mpk})$.

$\mathsf{KeyGen}(\mathsf{msk}, C) \rightarrow \mathsf{sk}_C$. The key generation algorithm does the following.

1. Parse $\mathsf{msk} = \mathsf{BD.msk}$ and denote C_i as the gates of circuit C for $i \in [|C|]$.
2. Sample $\mathbf{r} \leftarrow \{0,1\}^\lambda$.
3. For $i \in [|C|]$ define circuit $U[C_i, \mathbf{r}]$ as: on input (sd, x)
   $$U[C_i, \mathbf{r}](\mathsf{sd}, x) = \widetilde{C}_i = \mathsf{bGC.Garble}_i(1^\lambda, C_i; \mathsf{PRF}(\mathsf{sd}, \mathbf{r}))$$
   For a dummy gate C_0, define $U[C_0, \mathbf{r}]$ as follows. On input (sd, x)
   (a) Compute $\{\mathsf{lab}_{j,b}\}_{j\in[\mathrm{L}],b\in\{0,1\}} = \mathsf{bGC.Garble}_{\mathsf{inp}}(1^\lambda, 1^{\mathrm{L}}; \mathsf{PRF}(\mathsf{sd}, \mathbf{r}))$.
   (b) Output $\mathsf{lab}_{\mathbf{x}} = (\mathsf{lab}_{1,x_1}, \ldots, \mathsf{lab}_{\mathrm{L},x_{\mathrm{L}}})$, where x_j denotes the j-th bit of $\mathbf{x}$ for $j \in [\mathrm{L}]$.
4. Compute $\mathsf{BD.sk}_i \leftarrow \mathsf{BD.KeyGen}(\mathsf{BD.msk}, U[C_i, \mathbf{r}])$ for $i \in [0, |C|]$.
5. Output $\mathsf{sk}_C = \{\mathsf{BD.sk}_i\}_{i\in[0,|C|]}$.

$\mathsf{Enc}(\mathsf{mpk}, \mathbf{x}) \rightarrow \mathsf{ct}$. The encryption algorithm does the following.

- Parse $\mathsf{mpk} = \mathsf{BD.mpk}$.
- Sample a PRF key $\mathsf{sd} \leftarrow \{0,1\}^\lambda$ and compute $\mathsf{BD.ct} \leftarrow \mathsf{BD.Enc}(\mathsf{BD.mpk}, (\mathsf{sd}, \mathbf{x}))$.
- Output $\mathsf{ct} = \mathsf{BD.ct}$.

$\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_C, \mathsf{ct})$. The decryption algorithm does the following.

- Parse $\mathsf{mpk} = \mathsf{BD.mpk}$, $\mathsf{sk}_C = \{\mathsf{BD.sk}_i\}_{i\in[0,|C|]}$ and $\mathsf{ct} = \mathsf{BD.ct}$.
- For $i \in [|C|]$, compute $\widetilde{C}_i = \mathsf{BD.Dec}(\mathsf{BD.mpk}, \mathsf{BD.sk}_i, \mathsf{BD.ct})$. Set $\widetilde{C} = \{\widetilde{C}_i\}_{i\in[|C|]}$.
- Compute $\mathsf{lab}_{\mathbf{x}} = \mathsf{BD.Dec}(\mathsf{BD.mpk}, \mathsf{BD.sk}_0, \mathsf{BD.ct})$.
- Compute and output $C(\mathbf{x}) = \mathsf{bGC.Eval}(1^\lambda, \widetilde{C}, \mathsf{lab}_{\mathbf{x}})$.

[16] W.L.O.G we assume that msk contains mpk.

Correctness. We prove the correctness of our scheme using the following theorem.

Theorem C.1. Assume that bGC and BD-prFE schemes are correct. Then the above construction of prFE scheme is correct.

Proof. For $\mathsf{sk}_C = \{\mathsf{BD.sk}_i\}_{i\in[0,|C|]}$, where $\mathsf{BD.sk}_i \leftarrow \mathsf{BD.KeyGen}(\mathsf{BD.msk}, U[C_i, \mathbf{r}])$ and $\mathsf{ct} = \mathsf{BD.ct}$ where $\mathsf{BD.ct} \leftarrow \mathsf{BD.Enc}(\mathsf{BD.mpk}, (\mathsf{sd}, \mathbf{x}))$, with probability 1, we have

$$\mathsf{BD.Dec}(\mathsf{BD.mpk}, \mathsf{BD.sk}_i, \mathsf{BD.ct}) = \begin{cases} \widetilde{C}_i = \mathsf{bGC.Garble}_i(1^\lambda, C_i; \mathsf{PRF}(\mathsf{sd}, \mathbf{r})) \text{ if } i \in [|C|] \\ \mathsf{lab}_{\mathbf{x}} \text{ if } i = 0 \end{cases}$$

from the correctness of prFE scheme and definition of $U[C_i, \mathbf{r}]$.
Next, setting $\widetilde{C} = \{\widetilde{C}_i\}_{i\in[|C|]}$ and using the correctness of the bGC scheme, with probability 1, we have

$$\mathsf{bGC.Eval}(1^\lambda, \widetilde{C}, \mathsf{lab}_{\mathbf{x}}) = C(x)$$

hence the correctness. □

C.2 Security.

We prove the security of our scheme via the following theorem.

Theorem C.2. Assume that the BD-prFE scheme is secure (Definition 4.2) and the bGC scheme satisfies simulation security (Definition 3.16) and blindness (Definition 3.17). Then the construction of unbounded depth prFE is secure (Definition 4.2).

Proof. Consider a sampler $\mathsf{Samp}_{\mathsf{prFE}}$ that generates the following:

1. **Key Queries.** It issues Q_{key} key queries $C_1, \ldots, C_{Q_{\mathsf{key}}}$. We use $|C_k|$ to denote the size of circuit C_k for $k \in [Q_{\mathsf{key}}]$.
2. **Ciphertext Queries.** It issues Q_{msg} ciphertext queries $\mathbf{x}_1, \ldots, \mathbf{x}_{Q_{\mathsf{msg}}}$, where $|\mathbf{x}_1| = \cdots = |\mathbf{x}_{Q_{\mathsf{msg}}}|$.
3. **Auxiliary Information.** It outputs the auxiliary information $\mathsf{aux}_{\mathcal{A}}$.

To prove the multi-challenge security as per Definition 4.2, we show

$$\begin{pmatrix} \mathsf{mpk} = \mathsf{BD.mpk}, \mathsf{aux}_{\mathcal{A}}, \\ \{U[C_{k,i}, \mathbf{r}_k]\}_{k\in[Q_{\mathsf{key}}], i\in[0,|C_k|]}, \\ \{\mathsf{sk}_{C_k} = \{\mathsf{BD.sk}_{k,i}\}_{i\in[0,|C_k|]}\}_{k\in[Q_{\mathsf{key}}]}, \\ \{\mathsf{ct}_j = \mathsf{BD.ct}_j\}_{j\in[Q_{\mathsf{msg}}]} \end{pmatrix} \approx_c \begin{pmatrix} \mathsf{mpk} = \mathsf{BD.mpk}, \mathsf{aux}_{\mathcal{A}}, \\ \{U[C_{k,i}, \mathbf{r}_k]\}_{k\in[Q_{\mathsf{key}}], i\in[0,|C_k|]}, \\ \{\mathsf{sk}_{C_k} = \{\mathsf{BD.sk}_{k,i}\}_{i\in[0,|C_k|]}\}_{k\in[Q_{\mathsf{key}}]}, \\ \{\mathsf{ct}_j \leftarrow \mathcal{CT}_{\mathsf{prFE}}\}_{j\in[Q_{\mathsf{msg}}]} \end{pmatrix}$$

assuming we have

$$(1^\lambda,\ \mathsf{aux}_{\mathcal{A}},\ \{C_k,\ C_k(x_j)\}_{j\in[Q_{\mathsf{msg}}], k\in[Q_{\mathsf{key}}]}) \approx_c (1^\lambda,\ \mathsf{aux}_{\mathcal{A}},\ \{C_k,\ \Delta_{j,k} \leftarrow \{0,1\}\}_{j\in[Q_{\mathsf{msg}}], k\in[Q_{\mathsf{key}}]}) \quad (54)$$

where for $i \in [|C_k|]$, $U[C_{k,i}, \mathbf{r}_k]$ denotes the function corresponding to i-th gate of the k-th key query, $C_{k,i}$, and $U[C_{k,0}, \mathbf{r}_k]$ denotes the function corresponding to the dummy gate for the k-th key query, as defined in the KeyGen algorithm. Also, $\mathsf{BD.sk}_{k,i} \leftarrow \mathsf{BD.KeyGen}(\mathsf{BD.msk}, U[C_{k,i}, \mathbf{r}_k])$ and $\mathsf{BD.ct}_j \leftarrow \mathsf{BD.Enc}(\mathsf{BD.mpk}, (\mathsf{sd}, \mathbf{x}))$ for $j \in [Q_{\mathsf{msg}}]$, $k \in [Q_{\mathsf{key}}]$, $i \in [0, |C_k|]$.

We invoke the multi-challenge security of BD-prFE with sampler $\mathsf{Samp}_{\mathsf{BD}}$ that outputs

$$\begin{pmatrix} \text{Functions:} & \{U[C_{k,i}, \mathbf{r}_k]\}_{k\in[Q_{\mathsf{key}}], i\in[0,|C_k|]}, \\ \text{Inputs:} & \{\mathsf{sd}_j, \mathbf{x}_j\}_{j\in[Q_{\mathsf{msg}}]}, \\ \text{Auxiliary Information:} & \mathsf{aux} = \left(\mathsf{aux}_{\mathcal{A}}, \{C_{k,i}, \mathbf{r}_k\}_{k\in[Q_{\mathsf{key}}], i\in[|C_k|]}\right) \end{pmatrix}$$

By the security guarantee of BD-prFE with sampler $\mathsf{Samp}_{\mathsf{BD}}$ we have

$$\begin{pmatrix} \mathsf{BD.mpk},\ \mathsf{aux}, \{U[C_{k,i},\mathbf{r}_k],\ \mathsf{BD.sk}_{k,i}\}_{k\in[Q_{\mathsf{key}}],i\in[0,|C_k|]} \\ \{\mathsf{BD.ct}_j \leftarrow \mathsf{BD.Enc}(\mathsf{BD.mpk},(\mathsf{sd}_j,\mathbf{x}_j))\}_{j\in[Q_{\mathsf{msg}}]} \end{pmatrix}$$
$$\approx_c \begin{pmatrix} \mathsf{BD.mpk},\ \mathsf{aux}, \{U[C_{k,i},\mathbf{r}_k],\ \mathsf{BD.sk}_{k,i}\}_{k\in[Q_{\mathsf{key}}],i\in[0,|C_k|]} \\ \{\Delta_j \leftarrow \mathcal{CT}_{\mathsf{BD}}\}_{j\in[Q_{\mathsf{msg}}]} \end{pmatrix}$$

if

$$\begin{pmatrix} \mathsf{aux},\ \left\{U[C_{k,i},\mathbf{r}_k],\ \tilde{C}^j_{k,i} = \mathsf{bGC.Garble}_i(1^\lambda, C_{k,i}; \mathsf{PRF}(\mathsf{sd}_j,\mathbf{r}_k))\right\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}],i\in[|C_k|]} \\ \left\{\mathsf{lab}^k_{\mathbf{x}_j} = (\mathsf{lab}^k_{1,x_{j,1}},\ldots,\mathsf{lab}^k_{\mathrm{L},x_{j,\mathrm{L}}}) \mid \{\mathsf{lab}^k_{j',b}\}_{j'\in[\mathrm{L}],b\in\{0,1\}} \leftarrow \mathsf{bGC.Garble}_{\mathsf{inp}}(1^\lambda,1^{\mathrm{L}};\mathsf{PRF}(\mathsf{sd}_j,\mathbf{r}_k))\right\}_{j,k} \end{pmatrix} \tag{55}$$
$$\approx_c \left(\mathsf{aux}, \{U[C_{k,i},\mathbf{r}_k],\ \tilde{C}^j_{k,i} \leftarrow \{0,1\}^{\ell_{\mathsf{bGC}}}, \mathsf{lab}^k_{\mathbf{x}_j} \leftarrow \{0,1\}^{\ell_{\mathsf{bGC}}}\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}],i\in[|C_k|]}\right)$$

where $\tilde{C}^j_{k,i} = U[C_{k,i},\mathbf{r}_k](\mathsf{sd}_j,\mathbf{x}_j)$ for $i\in[|C_k|]$ and $\mathsf{lab}^k_{\mathbf{x}_j} = U[C_{k,0},\mathbf{r}_k](\mathsf{sd}_j,\mathbf{x}_j)$. In the above, $C_{k,i}$ denotes the i-th gate of circuit C_k and $x_{j,j'}$ denotes the j'-th bit of $\mathbf{x}_j$ for $j'\in[\mathrm{L}], j\in[Q_{\mathsf{msg}}]$.

Thus to prove the security of the prFE scheme it suffices to prove Equation (55).
We prove Equation (55) via the following sequence of hybrids.

Hyb_0. This is the LHS distribution of Equation (55). We re-write the distribution as

$$\begin{pmatrix} \mathsf{aux},\ \left\{U[C_{k,i},\mathbf{r}_k],\ \tilde{C}^j_{k} = \mathsf{bGC.Garble}_1(1^\lambda, C_{k}; \mathsf{PRF}(\mathsf{sd}_j,\mathbf{r}_k))\right\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]} \\ \left\{\mathsf{lab}^k_{\mathbf{x}_j} = (\mathsf{lab}^k_{1,x_{j,1}},\ldots,\mathsf{lab}^k_{\mathrm{L},x_{j,\mathrm{L}}}) \mid \{\mathsf{lab}^k_{j',b}\}_{j'\in[\mathrm{L}],b\in\{0,1\}} \leftarrow \mathsf{bGC.Garble}_{\mathsf{inp}}(1^\lambda,1^{\mathrm{L}};\mathsf{PRF}(\mathsf{sd}_j,\mathbf{r}_k))\right\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]} \end{pmatrix}$$

where $\mathsf{bGC.Garble}_1(1^\lambda,C_k) = \{\mathsf{bGC.Garble}_i(1^\lambda,C_{k,i})\}_{i\in[|C_k|]}$ for a circuit $C_k\in\mathcal{C}_{\mathrm{L}}$.

Hyb_1. This hybrid is the same as the previous one except that we replace $\mathsf{PRF}(\mathsf{sd}_j,\cdot)$, used to compute $\tilde{C}^j_k$ and $\mathsf{lab}^k_{\mathbf{x}_j}$, with the real random function $\mathsf{R}^j(\cdot)$ for each $j\in[q_{\mathsf{msg}}]$. Since sd_j is not used anywhere else, we can use the security of PRF to conclude that this hybrid is computationally indistinguishable from the previous one.

Hyb_2. This hybrid is same as the previous one except that we output a failure symbol if the set $\{\mathbf{r}_k\}_{k\in[Q_{\mathsf{key}}]}$, in aux, contains a collision. We prove that the probability with which there occurs a collision is negligible in λ. To prove this it suffices to show that there is no $k,k'\in[Q_{\mathsf{key}}]$ such that $k\neq k'$ and $\mathbf{r}_k=\mathbf{r}_{k'}$. The probability of this happening can be bounded by $Q^2_{\mathsf{key}}/2^\lambda$ by taking the union bound with respect to all the combinations of k,k'. Thus the probability of outputting the failure symbol is $Q^2_{\mathsf{key}}/2^\lambda$ which is $\mathsf{negl}(\lambda)$.

Hyb_3. In this hybrid we compute $\tilde{C}^j_k$ and $\mathsf{lab}^k_{\mathbf{x}_j}$ using fresh randomness $R_{k,j}\leftarrow\{0,1\}^{\mathsf{R}_{\mathsf{bGC}}}$ instead of deriving the randomness by $\mathsf{R}^j(\mathbf{r}_k)$. We claim that this change is only conceptual. To see this, we observe that unless the failure condition introduced in Hyb_2 is satisfied, every invocation of the function R^j is with respect to a fresh input and thus the output can be replaced with a fresh randomness.
In this hybrid, the view of the adversary is

$$\begin{pmatrix} \mathsf{aux},\ \left\{U[C_{k,i},\mathbf{r}_k],\ \tilde{C}^j_{k} = \mathsf{bGC.Garble}_1(1^\lambda, C_{k}; R_{k,j}), \mathsf{lab}^k_{\mathbf{x}_j}\right\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]} \\ \left\{\mathsf{lab}^k_{\mathbf{x}_j} = (\mathsf{lab}^k_{1,x_{j,1}},\ldots,\mathsf{lab}^k_{\mathrm{L},x_{j,\mathrm{L}}}) \mid \{\mathsf{lab}^k_{j',b}\}_{j'\in[\mathrm{L}],b\in\{0,1\}} \leftarrow \mathsf{bGC.Garble}_{\mathsf{inp}}(1^\lambda,1^{\mathrm{L}};R_{k,j})\right\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]} \end{pmatrix}$$

Hyb_4. This hybrid is same as the previous one except that we compute $(\tilde{C}_k^j, \mathsf{lab}_{\mathbf{x}_j}^k) \leftarrow \mathsf{bGC.Sim}(1^\lambda, 1^{\mathrm{L}}, \mathsf{C}_k(\mathbf{x}_j))$ for all $j \in [Q_{\mathsf{msg}}], k \in [Q_{\mathsf{key}}]$. $\mathsf{Hyb}_3 \approx_c \mathsf{Hyb}_4$ using the simulation security of the bGC scheme.
In this hybrid, the view of the adversary is

$$\left(\mathsf{aux},\ \left\{U[\mathsf{C}_{k,i}, \mathbf{r}_k],\ (\tilde{C}_k^j, \mathsf{lab}_{\mathbf{x}_j}^k) \leftarrow \mathsf{bGC.Sim}\left(1^\lambda, 1^{|C|}, 1^{\mathrm{L}}, \mathsf{C}_k(\mathbf{x}_j)\right)\right\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]}\right)$$

where $|C|$ is the maximum size of a circuit in the circuit class $\mathcal{C}_{\mathrm{L}}$.

Hyb_5. This hybrid is same as the previous one except that we compute $(\tilde{C}_k^j, \mathsf{lab}_{\mathbf{x}_j}^k) \leftarrow \mathsf{bGC.Sim}(1^\lambda, 1^{\mathrm{L}}, \Delta_{j,k})$, where $\Delta_{j,k} \leftarrow \{0,1\}$, for all $j \in [Q_{\mathsf{msg}}], k \in [Q_{\mathsf{key}}]$. $\mathsf{Hyb}_4 \approx_c \mathsf{Hyb}_5$ follows from the pseudorandomness of $\mathsf{C}_k(x_j)$ (Equation (54)).
In this hybrid, the view of the adversary is

$$\left(\mathsf{aux},\ \left\{U[\mathsf{C}_{k,i}, \mathbf{r}_k],\ (\tilde{C}_k^j, \mathsf{lab}_{\mathbf{x}_j}^k) \leftarrow \mathsf{bGC.Sim}\left(1^\lambda, 1^{\mathrm{L}}, \Delta_{j,k}\right)\right\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]}\right)$$

where $\Delta_{j,k} \leftarrow \{0,1\}$.

Hyb_6. This hybrid is same as the previous one except that we sample $(\tilde{C}_k^j, \mathsf{lab}_{\mathbf{x}_j}^k) \leftarrow \mathcal{CT}_{\mathsf{SIM}}$ uniformly at random. $\mathsf{Hyb}_5 \approx_c \mathsf{Hyb}_6$ using the blindness of the bGC scheme. In this hybrid, the view of the adversary is

$$\left(\mathsf{aux},\ \left\{U[\mathsf{C}_{k,i}, \mathbf{r}_k],\ (\tilde{C}_k^j, \mathsf{lab}_{\mathbf{x}_j}^k) \leftarrow \mathcal{CT}_{\mathsf{SIM}}\right\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]}\right)$$

which is the RHS distribution of Equation (55).

Hence, the proof. □

Instantiation. Instantiating the construction using the bounded depth prFE from Section 4.2, we achieve the following theorem.

Theorem C.3. Under the LWE assumption and the Evasive LWE assumption, there exists a very selectively secure prFE scheme for circuits of unbounded depth and input length L with

$$|\mathsf{mpk}| = \mathrm{L} \cdot \mathrm{poly}(\lambda), \quad |\mathsf{sk}_C| = \mathrm{L} \cdot |C| \cdot \mathrm{poly}(\lambda), \quad |\mathsf{ct}| = \mathrm{L} \cdot \mathrm{poly}(\lambda).$$