Pseudorandom Multi-Input Functional Encryption and Applications

Shweta Agrawal* Simran Kumari† Shota Yamada‡

Abstract

We construct the first multi-input functional encryption (MIFE) and indistinguishability obfuscation (iO) schemes for pseudorandom functionalities, where the output of the functionality is pseudorandom for every input seen by the adversary. Our MIFE scheme relies on LWE and (private-coin) evasive LWE (Wee, Eurocrypt 2022 and Tsabary, Crypto 2022) for constant arity functions, and a strengthening of evasive LWE for polynomial arity. Thus, we obtain the first MIFE and iO schemes for a nontrivial functionality from conjectured post-quantum assumptions.

Along the way, we identify subtle issues in the proof of witness encryption from evasive LWE by prior work and believe that a similar strengthening of evasive LWE should also be required for their proof, for the same reasons as ours. We demonstrate the power of our new tools via the following applications:

1. *Multi Input Predicate Encryption for Constant Arity.* Assuming evasive LWE and LWE, we construct a multi-input predicate encryption scheme (MIPE) for P, supporting constant arity. The only prior work to support MIPE for P with constant arity by Agrawal et al. (Crypto, 2023) relies on a strengthening of Tensor LWE in addition to LWE and evasive LWE.

2. *Multi Input Predicate Encryption for Polynomial Arity.* Assuming a stronger variant of evasive LWE and LWE, we construct MIPE for P for polynomial arity. MIPE for polynomial arity supporting P was not known before, to the best of our knowledge.

3. *Two Party ID Based Key Exchange.* Assuming a stronger variant of evasive LWE and LWE, along with Decision Bilinear Diffie-Hellman, we provide the first two-party ID based Non-Interactive Key Exchange (ID-NIKE) scheme in the standard model. This leads to the first ID-NIKE in the standard model without using multilinear maps or indistinguishability obfuscation.

4. *Instantiating the Random Oracle.* We use our pseudorandom iO 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.

Our tools of MIFE and iO for pseudorandom functionalities appear quite powerful and yield extremely simple constructions when used in applications. We are optimistic they will find other applications.

*IIT Madras, India, shweta@cse.iitm.ac.in
†IIT Madras, India, sim78608@gmail.com
‡AIST Tokyo, Japan, yamada-shota@aist.go.jp

Contents

1 Introduction

Functional Encryption (FE) [BSW11, SW05] is a popular strengthening of public key encryption where decryption corresponds to functionalities rather than users. In more detail, in FE, the ciphertext encodes some private data $\mathbf{x}$, the secret key encodes some public function f and decryption allows to recover $f(\mathbf{x})$ while hiding everything else about $\mathbf{x}$. Since FE supports computing on encrypted data together with learning outputs of some authorized computation in the clear, it holds the promise of enabling futuristic applications such as privacy preserving machine learning. FE also generalizes popular primitives [BSW11] such as Identity Based Encryption [BF01], Attribute Based Encryption [SW05] and Predicate Encryption [KSW08]. Hence, FE attracted a lot of attention right from its inception, and has been widely studied in the community.

Multi-Input FE and iO. A compelling extension of functional encryption, introduced by Goldwasser et al. is multi-input FE (MIFE) [GGG+14] where 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 $f(\mathbf{x}_1, \ldots, \mathbf{x}_n)$. Evidently for real world applications of computing on encrypted data, it is highly desirable to be able to compute on data generated by different parties independently – this can enable important statistical functionalities such as running medical research algorithms on encrypted genomic or medical data. 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 powerful enough to generically imply multi-input FE. We remark that for restricted functionalities, the compiler does not apply and multi-input FE for interesting restricted classes have been studied extensively [AGRW17, DOT18, ACF+18, CDG+18, Tom19, ABKW19, ABG19, LT19, AGT21b, AGT21a, AGT22, ATY23].

Aside from the real world applications of MIFE, the work of [AJ15, BV18] showed that sufficiently expressive MIFE can be used to construct the powerful notion of *Indistinguishability Obfuscation* (iO), 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. This security property can be formalized in various ways, and one popular formalization asks that an adversary, given the obfuscation of C_b where b is a random bit and C_0 is functionally *equivalent* to C_1, cannot predict the value of b with non-negligible advantage.

While non-obvious in the beginning what such an object is useful for, a series of works has shown that iO can be used to instantiate a large number of advanced cryptographic primitives [GGH+13, SW14, BFM14, GGG+14, HSW13, KLW15, BPR15, CHN+16, GPS16, HJK+16, HY17, KS17]. Following this, there ensued a quest for constructing iO from reasonable cryptographic assumptions, with a large number of works coming closer and closer to the goal [GGH+16, Agr19, APM20, JLMS19, GJLS21, GP21, WW21, BDGM23, BDGM20], until the beautiful work of Jain, Lin and Sahai [JLS21] finally accomplished the goal.

Constructions from Lattice Assumptions. So far, the only constructions of compact FE/MIFE/iO from standard assumptions rely on pairings (together with LPN variants and/or low depth PRGs) [JLS21, JLS22, RVV24] and are hence quantum insecure. Perhaps even more importantly, it is dissatisfying to have only a single pathway to constructing such a central object in the theory of cryptography. Casting around for other assumptions, hopefully with (conjectured) quantum security – the most obvious candidate is some assumption based on lattices. In particular, the Learning With Errors (LWE) assumption has proven to be amazingly versatile and provides elegant solutions to encrypted computation primitives such as fully homomorphic encryption or attribute based encryption [BV14, Bra12, BGV14, GSW13, GVW13, BGG+14]. Thus the hope basing iO on LWE is natural, and unsurprisingly, has received a lot of attention. Several exciting candidates for compact FE and iO have been constructed from strengthenings of the LWE assumption [Agr19, APM20, WW21, GP21, DQV+21] – unfortunately, these either rely on some heuristic, or their underlying assumptions have been broken [HJL21, JLLS23].

Our Approach. Since full fledged FE/iO from lattice based assumptions has been elusive despite significant effort, a principled approach for making progress would be to restrict the functionality supported by FE/iO so that it is still meaningful for applications while also admitting a construction from some reasonable lattice assumption. A promising candidate assumption that interpolates the safety of LWE and the insecurity of iO or multilinear map assumptions is the recently introduced *evasive* LWE [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

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

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

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

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 [AKY24a], to name a few. Evasive LWE is considered plausible in "safe" regimes (discussed in detail in Section 1.2). Continuing this agenda, we ask:

Can we construct MIFE/iO *for a nontrivial functionality from evasive* LWE*?*

1.1 Our Results

We answer the above question in the affirmative and construct the first MIFE and iO for pseudorandom functionalities, namely functions where the output is pseudorandom for every input *seen by the adversary*. In the context of iO this means that the entire truth table must be pseudorandom, since the adversary can compute the functionality for any input. But for MIFE, it suffices to only restrict the output for the functions queried by the adversary. We denote such an MIFE by prMIFE.

In more detail, we obtain the following results:

Theorem 1.1 (prMIFE **for constant arity).** Assume evasive LWE and LWE. Then there exists a prMIFE scheme for arity $n = O(1)$, supporting functions with (fixed) input length L and bounded polynomial depth.[1]

For polynomial arity, we require a strengthening of evasive LWE, which we call non-uniform κ-evasive LWE. This is necessary to handle some delicate technical issues, which also appear in the proof of witness encryption from evasive LWE by [VWW22], to the best of our understanding. We believe that a similar strengthening of evasive LWE should also be required for their proof, for the same reasons as ours. We then show:

Theorem 1.2 (prMIFE **for polynomial arity).** Assume non-uniform κ-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 [AJ15, BV18], albeit via a different proof of security. We denote this by prIO. In more detail:

Theorem 1.3 (prIO). Assuming non-uniform κ-evasive LWE, non-uniform sub-exponential PRF, and non-uniform sub-exponential LWE, there exists a prIO scheme for all polynomial sized circuits.

[1] As is shown in [AMYY25], evasive LWE does not hold for the general samplers. When we invoke the assumption, we always invoke it with respect to a specific sampler class induced by the respective application. However, for ease of exposition, we will sometimes omit the reference to the specific sampler in the overview. Similar remarks can be applied to prFE and prMIFE.

We also define a variant of prIO which only supports polynomial sized domains – we denote this variant by pPRIO. The advantage of considering this restricted variant is that we can base its security on (plain) evasive LWE, rather than it's strengthening.

Theorem 1.4. Assuming evasive LWE and LWE, there exists a secure pPRIO scheme supporting circuits of bounded size.

Applications.

We obtain several applications from our new tools of prMIFE and prIO:

1. *Multi Input Predicate Encryption (*miPE*) for Constant Arity.* Assuming evasive LWE and LWE, there exists a prMIFE scheme for arity $n = O(1)$, supporting functions of bounded polynomial depth. The only prior work to support miPE with constant arity for function class P is by Agrawal et al. [ARYY23] and uses a strengthening of (non-standard) tensor LWE [Wee21] together with evasive LWE.

2. *Multi Input Predicate Encryption for Polynomial Arity.* Assuming non-uniform κ-evasive LWE, non-uniform sub-exponential PRF, and non-uniform sub-exponential LWE, there exists a miPE scheme for arity $n = \text{poly}(\lambda)$, supporting functions of bounded polynomial depth. MIPE for polynomial arity supporting P was not known before, to the best of our knowledge.

3. *Two Party ID Based Key Exchange.* Assuming non-uniform κ-evasive LWE, non-uniform sub-exponential PRF, non-uniform sub-exponential LWE, and the DBDH assumption, there exists a secure two-party ID-NIKE scheme. This leads to the first ID-NIKE in the standard model without using multilinear maps or indistinguishability obfuscation.

4. *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.

Additional Prior Work. The notion of iO for pseudorandom functionalities was considered implicitly by the work of Mathialagan, Peters and Vaikuntanathan [MPV24a] where they used subexponential LWE and evasive LWE to construct adaptively sound zero-knowledge SNARKs for UP. In a previous version of their work [MPV24b], which was in private circulation and shared with us, this notion was defined explicitly and leveraged to obtain unlevelled fully homomorphic encryption. However, an explicit construction of iO for pseudorandom functionalities was not provided.

Our Companion Paper. In our companion work [AKY24b], we provided the first construction of compact FE for pseudorandom functionalities, namely, functionalities where the output is (pseudo)random for any input seen by the adversary. We then use our pseudorandom FE to improve the state of the art in single input attribute based encryption (ABE) schemes, supporting circuits of unbounded depth. To achieve *optimal* parameters in [AKY24b], we use (black box) the building block of "Polynomial Domain IO for Pseudorandom Functionalities" (pPRIO), which is constructed only in the present work (Section 6), creating a dependence between the two works.

The rationale for our choice is primarily to have the most concise presentation and avoid duplication of content. We chose to split the papers across the axis of single input and multi-input since this seems most natural. The notion of pPRIO uses MIFE for constant arity, so fits naturally in the present work. By *not* using it in the companion work, we would achieve ABE schemes with sub-optimal parameters in [AKY24b] which nevertheless outperform the state of the art. We would then improve these schemes in the present work, which was undesirable. Another alternative was to provide the multi-input compiler for constant arity in [AKY24b] and generalize this to polynomial arity (from a stronger assumption) in the present work. However this would lead to duplicating the MIFE construction in both works. Therefore, we believe that using the pPRIO black-box as a tool in the ABE application of [AKY24b] is the best option.

1.2 Recent Attacks on Evasive LWE and Repercussions.

As discussed in our companion paper [AKY24b], subsequent to the first online appearance of the present work, new counter-examples for evasive LWE were developed [AMYY25, HJL25, DJM+25]. These attacks show that the intuition that evasive LWE evades 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 [AKY24b], the attacker can obtain problematic leakage. However, as discussed by [AMYY25], this intuition can be recovered by placing appropriate restrictions on the sampler. Additionally, subsequent to the development of the counter-examples, we modified the construction in [AKY24b] so that the attacks no longer apply even when the sampler is malicious. We refer the reader to the companion work [AKY24b] for a detailed discussion.

Additionally, the work of [BDJ+24] and [AMYY25] show that there exists a contrived "self-referential" functionality for which pseudorandom functional encryption or pseudorandom obfuscation cannot exist. As discussed in [AMYY25], this result may be seen as analogous to the impossibilities known for the random oracle model [CGH04] or virtual black box (VBB) obfuscation [BGI+01]. In more detail, 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. The pseudorandom functionalities that are useful for our applications, such as computing FE ciphertexts, are quite natural and do not fall prey to such attacks. Finally, we mention the elegant 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, both $\mathbf{B}$ and $\mathbf{P}$ are known to the adversary, hence these attacks do not apply.

We view these attacks as an important step forward in our understanding of evasive LWE. Note that evasive LWE should be seen as a family of assumptions parametrized by the description of the sampler and choice of error distributions, which, if invoked in full generality, is now known to be false, even in the public-coin setting. However, as discussed in [AMYY25], the original intuition by Wee [Wee22] and Tsabary [Tsa22] about the security of evasive LWE can be recovered by taking precautions to identify and respect a "safe zone" for evasive LWE. In our opinion, constructions from disciplined new assumptions are important to make meaningful progress on long-standing problems that have resisted solutions from standard assumptions despite much effort.

1.3 Technical Overview

Compact FE for Pseudorandom Functionalities. Our starting point is our companion work which provides FE for pseudorandom functionalities in the single input setting [AKY24b]. We recall the syntax here: The setup algorithm takes as input the security parameter λ and parameter prm, specifying the parameters of the function class, and outputs $(\mathsf{mpk}, \mathsf{msk})$. The key generation algorithm on input msk and a function $f : \{0,1\}^{\mathrm{L}} \to \{0,1\}^{\ell}$ outputs a functional secret key sk_f. The encryption algorithm on input mpk and an input message $\mathbf{x} \in \{0,1\}^{\mathrm{L}}$ outputs a ciphertext ct. The decryption algorithm takes the functional secret key sk_f and ciphertext ct as input and outputs some $\mathbf{y} \in \{0,1\}^{\ell}$. Correctness requires that decryption should output $f(x)$ if the setup, encrypt and key generation algorithms were run honestly.

The definition of security, termed prCT security, says that so long as the *output* of the functionality is pseudorandom, the *ciphertext* is pseudorandom. In more detail, 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.

We define the following advantage functions:

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

$$\begin{aligned} \mathsf{Adv}^{\mathrm{POST}}_{\mathcal{A}_1}(\lambda) \stackrel{\mathrm{def}}{=} \Pr\Big[\mathcal{A}_1(\mathsf{mpk},\ \mathsf{aux},\ \{f_i,\ \mathsf{ct}_j \leftarrow \mathsf{Enc}(\mathsf{mpk}, x_j),\ \mathsf{sk}_{f_i}\}_{i\in[Q_{\mathsf{key}}],j\in[Q_{\mathsf{msg}}]}) = 1\Big] \\ - \Pr\Big[\mathcal{A}_1(\mathsf{mpk},\ \mathsf{aux},\ \{f_i,\ \delta_j \leftarrow \mathcal{CT},\ \mathsf{sk}_{f_i}\}_{i\in[Q_{\mathsf{key}}],j\in[Q_{\mathsf{msg}}]}) = 1\Big] \end{aligned} \tag{2}$$

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 $\mathcal{F}_{\mathsf{prm}}$ satisfies prCT security if for every PPT Samp and $\mathcal{A}_1$, there exists another PPT $\mathcal{A}_0$ such that

$$\mathcal{A}_{\mathcal{A}_0}^{\mathrm{PRE}}(\lambda) \geq \mathcal{A}_{\mathcal{A}_1}^{\mathrm{POST}}(\lambda)/Q(\lambda) - \mathsf{negl}(\lambda) \tag{3}$$

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

The work of [AKY24b] obtains the following result:

Theorem 1.5. Assuming the LWE and evasive LWE assumptions, there exists a secure 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

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

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

We remark that their result is actually significantly more general – they construct the broader notion of *partially hiding* FE, where the ciphertext can additionally have a public attribute in addition to a private payload[2], with optimal parameters and can support circuits of unbounded depth. But bounded depth prFE is sufficient for the applications considered in the present work, so we restrict our attention to this.

Extending to the Multi-Input Setting. 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 $y \in \mathcal{Y}_{\mathsf{prm}}$. The security has a similar flavor to the single-input setting, where 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 the security as follows.

Let $\kappa = \kappa(\lambda)$ be a function in λ and Samp be a PPT algorithm that on input 1^λ, outputs

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

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:

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

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

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

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

[2]The astute reader may wonder why regular FE does not imply partially hiding FE (PHFE) just by outputting the public attribute as part of the ciphertext – such an approach would make the ciphertext size grow with the length of public attribute which can be avoided in PHFE.

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.[3] 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 $n = \text{poly}(\lambda)$, we require large κ as an artifact of the security proof techniques. In the special case of n being constant, we can achieve $\kappa = \lambda^n$.

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 $(\mathsf{SKE.sk}, \mathbf{x}_i, \mathbf{x}_{i+1}, \ldots, \mathbf{x}_{n-1}, \mathbf{x}_n)$ 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[4],

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 corresponding to 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 key generation algorithm on input msk and a function $f : (\{0,1\}^{\mathrm{L}})^n \to \{0,1\}$, 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 $(\mathsf{msk}, \mathbf{x}_i \in \{0,1\}^{\mathrm{L}})$, 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 $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{n-1}, \mathbf{x}_n)$[5]. 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

[3] 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.

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

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

$\mathrm{F}_{n-1}(\mathsf{SKE.sk}, \mathbf{x}_n) = \mathsf{prFE}_{n-1}.\mathsf{ct}$ 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 $\mathsf{prFE}_1.\mathsf{Dec}(\mathsf{prFE}_1.\mathsf{mpk}, \mathsf{prFE}_1.\mathsf{sk}_f, f, \mathsf{prFE}_1.\mathsf{ct}) = f(\mathbf{x}_1, \ldots, \mathbf{x}_n)$.

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 $\mathsf{aux}_{\mathcal{A}}$

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

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 (4) is indistinguishable from the following distribution

$$\mathcal{D}_{0,1} : \left(\begin{array}{c} \mathsf{aux}_{\mathcal{A}},\ \{\mathsf{prFE}_i.\mathsf{mpk}\}_{i \in [n]},\ \left\{ f_k,\ \mathsf{sk}_{f_k} = \mathsf{prFE}_1.\mathsf{sk}_{f_k} \right\}_{k \in [q_0]} \\ \left\{ \mathsf{ct}_i^{j_i} = \gamma_i^{j_i} \leftarrow \mathcal{CT}_{\mathsf{SKE}},\ \mathsf{prFE}_{i+1}.\mathsf{sk}^{j_i} \right\}_{\substack{i \in [n-1], \\ j_i \in [q_i]}}, \\ \left\{ \delta^{j_n} \leftarrow \mathcal{CT}_{\mathsf{prFE}_n} \right\}_{j_n \in [q_n]} \end{array} \right) \tag{5}$$

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 (4)$\approx$ Equation (5) 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 (4) as auxiliary information. Now, from the security guarantee of prFE_n with the sampler, we know that to prove $\mathsf{prFE}.\mathsf{ct}_n^{j_n}$ 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

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

$$\approx$$

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

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:

$$\mathcal{D}_{n-1,0}: \begin{pmatrix} 1^\kappa,\ \mathsf{aux}_{\mathcal{A}},\ \mathsf{prFE}_1.\mathsf{mpk},\ \left\{f_k,\ \mathsf{sk}_{f_k} = \mathsf{prFE}_1.\mathsf{sk}_{f_k}\right\}_{k\in[q_0]}, \\ \left\{\mathsf{SKE.ct}_i^{j_i}\right\}_{\substack{i\in[n-1],\\ j_i\in[q_i]}}, \left\{\mathsf{prFE}_1.\mathsf{ct}^{j_1,\ldots,j_n}\right\}_{j_1\in[q_1],\ldots,j_n\in[q_n]} \end{pmatrix}$$

$$\approx$$

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

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

$$\mathcal{D}_{n,0}: \left(1^\kappa, \mathsf{aux}_{\mathcal{A}}, \left\{f_k,\ f_k(\mathbf{x}_1^{j_1}, \ldots, \mathbf{x}_n^{j_n})\right\}_{k,j_1,\ldots,j_n}, \left\{\mathsf{SKE.ct}_i^{j_i} \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.sk}, \mathbf{x}_i^{j_i})\right\}_{i,j_i}\right)$$

$$\approx_c$$

$$\mathcal{D}_{n,1}: \left(1^\kappa, \mathsf{aux}_{\mathcal{A}}, \left\{f_k,\ \Delta_k^{j_1,\ldots,j_n} \leftarrow \{0,1\}\right\}_{k,j_1,\ldots,j_n},\ \left\{\gamma_i^{j_i} \leftarrow \mathcal{CT}_{\mathsf{SKE}}\right\}_{i,j_i}\right) \tag{6}$$

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. The detail is not important here and we refer to Section 3 for the detail.

To prove Equation (6), 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 as we do below.

We note that at a high level, the 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 $\{\mathcal{D}_{h,b}\}_{h\in[n,0],b\in\{0,1\}}$ and shows that $\mathcal{D}_{0,0}$ and $\mathcal{D}_{0,1}$ are indistinguishable if $\mathcal{D}_{n,0} \approx_c \mathcal{D}_{n,1}$ 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 the prFE instances, 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 below. To begin, we formally define the evasive LWE assumption.

Evasive LWE. Let Samp be a PPT algorithm that outputs $(\mathbf{S}, \mathbf{P}, \mathsf{aux})$ on input 1^λ. For PPT adversaries $\mathcal{A}_0$ and $\mathcal{A}_1$, we define the following advantage functions:

$$\mathsf{Adv}^{\mathrm{PRE}}_{\mathcal{A}_0}(\lambda) \stackrel{\mathrm{def}}{=} \Pr\big[\mathcal{A}_0(\mathbf{B}, \mathbf{SB}+\mathbf{E}, \mathbf{SP}+\mathbf{E}', \mathsf{aux}) = 1\big] - \Pr\big[\mathcal{A}_0(\mathbf{B}, \mathbf{C}_0, \mathbf{C}', \mathsf{aux}) = 1\big] \tag{7}$$

$$\mathsf{Adv}^{\mathrm{POST}}_{\mathcal{A}_1}(\lambda) \stackrel{\mathrm{def}}{=} \Pr[\mathcal{A}_1(\mathbf{B}, \mathbf{SB}+\mathbf{E}, \mathbf{K}, \mathsf{aux}) = 1] - \Pr[\mathcal{A}_1(\mathbf{B}, \mathbf{C}_0, \mathbf{K}, \mathsf{aux}) = 1] \tag{8}$$

where $(\mathbf{S}, \mathbf{P}, \mathsf{aux}) \leftarrow \mathsf{Samp}(1^\lambda)$, $\mathbf{B}, \mathbf{C}_0, \mathbf{C}'$ are uniform matrices of appropriate dimensions, $\mathbf{E}, \mathbf{E}'$ are low norm Gaussian error matrices, and $\mathbf{K} \leftarrow \mathbf{B}^{-1}(\mathbf{P})$.

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

$$\mathsf{Adv}^{\mathrm{PRE}}_{\mathcal{A}_0}(\lambda) \geq \mathsf{Adv}^{\mathrm{POST}}_{\mathcal{A}_1}(\lambda)/Q(\lambda) - \mathsf{negl}(\lambda) \ \text{ and } \ \mathsf{Time}(\mathcal{A}_0) \leq Q(\lambda) \cdot \mathsf{Time}(\mathcal{A}_1). \tag{9}$$

Issue 1: On the multiplicative invocation of evasive LWE. To prove $\mathcal{D}_{0,0} \approx_c \mathcal{D}_{0,1}$, [VWW22] assumes that there exists an adversary $\mathcal{A}_0$ 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 $\mathcal{A}_1$ against $\mathcal{D}_{1,0}$ and $\mathcal{D}_{1,1}$. 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

$$\mathrm{poly}_1(\lambda_1)\mathrm{poly}_2(\lambda_2)\cdots\mathrm{poly}_n(\lambda_n) = \mathrm{poly}(\lambda_1\cdots\lambda_n) = \mathrm{poly}(2^{n^2}\lambda^n) \tag{10}$$

and setting the parameters 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[6] only with respect to this sampler. In more detail, to argue that the final distributions $\mathcal{D}_{n,0}$ and $\mathcal{D}_{n,1}$ are indistinguishable, it is required that *all* pairs of distributions $\mathcal{D}_{j,0}$ and $\mathcal{D}_{j,1}$ be indistinguishable. Then, to arrive at a contradiction, it suffices to find even one intermediate pair (for index h^*) which is distinguishable – we invoke evasive LWE with respect to that. We avoid the problem of incurring security loss of polynomial with arbitrarily large exponent, since we invoke the evasive LWE only once w.r.t a single sampler in the proof. However, this solution entails the strengthening of the

[6] To be precise, in our context, we invoke the security of prFE rather than evasive LWE. However, since the same issue arises both in our context and the context of witness encryption [VWW22] and the security of prFE is eventually reduced to the evasive LWE in our context, we intentionally do not distinguish the invocation of evasive LWE and prFE security here.

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.

Issue 2: On the additive term of evasive LWE. We believe there is another subtle issue 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 (10) 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 $\epsilon/\mathrm{poly}(2^{(n-1)^2}\lambda^{n-1})$. If we apply evasive LWE once more with respect to $\lambda_n = 2^n\lambda^n$ to complete the proof, 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. We define the strengthened version of evasive LWE next.

Non-Uniform κ-Evasive LWE. Let $\mathsf{Samp} = \{\mathsf{Samp}_\lambda\}_\lambda$ be a non-uniform sampler that takes as input 1^λ and outputs $(\mathbf{S}, \mathbf{P}, \mathsf{aux})$ as above. 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 $\mathsf{Adv}^{\mathrm{POST}}_{\mathcal{A}_1}(\lambda)$ as in Equation (7) and Equation (8), respectively. For a function $\kappa := \kappa(\lambda)$ of the security parameter λ, we say that the non-uniform κ-evasive LWE assumption holds if for every non-uniform sampler Samp and a non-uniform adversary $\mathcal{A}_1$ whose sizes are polynomial in λ' and κ respectively for $\lambda' := \lambda'(\lambda) < \kappa(\lambda)$, there exists another non-uniform adversary $\mathcal{A}_0$ and a polynomial $Q(\cdot)$ such that

$$\mathsf{Adv}^{\mathrm{PRE}}_{\mathcal{A}_0}(\lambda) \geq \mathsf{Adv}^{\mathrm{POST}}_{\mathcal{A}_1}(\lambda)/Q(\lambda') - \mathsf{negl}(\kappa) \quad \text{and} \quad \mathrm{Size}(\mathcal{A}_0) \leq Q(\lambda') \cdot \mathrm{Size}(\mathcal{A}_1). \tag{11}$$

There are two main differences from the standard EvLWE assumption: (i) we consider non-uniform sampler and adversaries, (ii) it is parameterized by κ and the additive term $\mathsf{negl}(\lambda)$ that appears in Equation (9) is replaced by $\mathsf{negl}(\kappa)$ in Equation (11). These changes are introduced so that our prFE can satisfy a stronger notion of security, as discussed below. Note that in the case λ' is superpolynomial in λ, $\mathsf{Samp}(1^\lambda)$ 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.

prFE with Non-Uniform κ Security. We now describe a stronger notion of security from prFE, which we need for our prMIFE compiler.

We say that a prFE scheme is secure in the non-uniform κ setting if for every Samp and an adversary $\mathcal{A}_1$ whose sizes are polynomial in λ' and κ respectively for $\lambda' := \lambda'(\lambda) < \kappa(\lambda)$, there exists another adversary $\mathcal{A}_0$ such that

$$\mathsf{Adv}^{\mathrm{PRE}}_{\mathcal{A}_0}(\lambda) \geq \mathsf{Adv}^{\mathrm{POST}}_{\mathcal{A}_1}(\lambda)/Q(\lambda') - \mathsf{negl}(\kappa) \tag{12}$$

where the advantage functions $\mathsf{Adv}^{\mathrm{PRE}}_{\mathcal{A}_0}(\lambda)$ and $\mathsf{Adv}^{\mathrm{POST}}_{\mathcal{A}_1}(\lambda)$ are defined as in Equation (1) and Equation (2) respectively, and $\mathrm{Size}(\mathcal{A}_0) \leq \mathrm{Size}(\mathcal{A}_1) \cdot Q(\lambda')$ for some polynomial $Q(\cdot)$.

The new security notion differs from the previous one in two ways: (i) it considers non-uniform adversaries instead of uniform adversaries, (ii) it is parameterized by κ and the additive term $\mathsf{negl}(\lambda)$ that appears in Equation (3) is replaced by $\mathsf{negl}(\kappa)$ in Equation (12). By taking κ asymptotically larger than λ (e.g., $\kappa := \lambda^\lambda$), we can make the additive term $\mathsf{negl}(\kappa)$ much smaller than $\mathsf{negl}(\lambda)$.

Fortunately, we can show, with some careful adjustments, that the prFE scheme from [AKY24b] also satisfies this stronger notion of security. In Appendix B.3, we prove the following theorem.

Theorem 1.6. Let $\kappa = 2^{\lambda^c}$ for some constant c. Assuming non-uniform κ-evasive LWE, subexponentially secure PRF against non-uniform adversary, and non-uniform sub-exponential LWE, there exists a 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 κ-prCT security as per Definition 2.15 with efficiency

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

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

With the above modifications, we are ready to state our final theorem:

Theorem 1.7 (prMIFE **for poly arity).** Let $\kappa = \lambda^{n^2 \log \lambda}$. Assume non-uniform κ-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 κ-security (Definition 3.2) with efficiency

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

where $\Lambda := (n^2\lambda)^{1/\delta}$.

Since the aforementioned issues in the proof only arise when evasive LWE is applied super-constant number of times, we do not need the stronger version of evasive LWE to support constant arity. Hence we get:

Theorem 1.8 (prMIFE **for constant arity).** Let $\kappa = \lambda^n$. Assume evasive LWE and LWE. Then there exists a prMIFE scheme for arity $n = O(1)$, supporting functions with input length L and bounded polynomial depth $d = d(\lambda)$, and satisfying κ-security (Definition 3.2) with efficiency

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

iO **for Pseudorandom Functionalities.** We now use our multi-input FE to construct iO, à la AJ/BV [AJ15, BV18] for the same functionality. We begin with the definition of iO for pseudorandom functionalities, which we refer to as prIO. Recall that this notion was first defined by Mathialagan, Peters and Vaikuntanathan [MPV24a]. 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.

Our notion of security however, differs from the standard notion considered in the literature of iO and 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

$$\text{If } \left(1^\kappa, \{C_0(x)\}_{x \in \{0,1\}^n}, \mathsf{aux}\right) \approx_c \left(1^\kappa, \{\Delta_x \leftarrow \{0,1\}^m\}_{x \in \{0,1\}^n}, \mathsf{aux}\right) \approx_c \left(1^\kappa, \{C_1(x)\}_{x \in \{0,1\}^n}, \mathsf{aux}\right)$$
$$\text{then } \left(\mathsf{iO}(1^\lambda, C_0), \mathsf{aux}\right) \approx_c \left(\mathsf{iO}(1^\lambda, C_1), \mathsf{aux}\right)$$

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 pseudorandom truth tables are indistinguishable.

Our construction follows the blueprint of the multi-input FE to iO conversion by Ananth and Jain [AJ15]. 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 follow from those of prMIFE. This leads to the following theorem, shown in Section 5.

Theorem 1.9. Let $\kappa = \lambda^{n^2 \log \lambda}$. Assuming non-uniform κ-evasive LWE, non-uniform sub-exponential PRF,, and non-uniform sub-exponential LWE, there exists a prIO scheme for circuits.

iO for Pseudorandom Functionalities with Polynomial Domains. We also define a variant of prIO which only supports polynomial sized domains. We denote this variant by pPRIO. The advantage of considering this restricted variant is that we can base the security of the construction on (plain) evasive LWE, rather than non-uniform κ-variant of it. For applications we consider stronger variant of the primitive where we decompose the obfuscation algorithm into online-offline parts and consequently define a reusable security for this variant. In more detail, we show the following:

Theorem 1.10. Assuming evasive LWE and LWE, there exists a secure pPRIO scheme supporting circuits of bounded size $L = \mathrm{poly}(\lambda)$ with

$$|\mathsf{obf}_{\mathsf{off}}| = \mathrm{poly}(L, \lambda), \quad |\mathsf{obf}_{\mathsf{on}}| = \mathrm{poly}(L, \lambda),$$

where $\mathsf{obf}_{\mathsf{off}}$ and $\mathsf{obf}_{\mathsf{on}}$ refer to the offline and online part of the obfuscated program, respectively.

We refer the reader to Section 6 for details. As discussed in Section 1, we use this construction in our companion paper [AKY24b] to construct ABE for optimal parameters.

Applications. We show that our new tool of MIFE/iO for pseudorandom functionalities is quite powerful and yields several interesting applications.

Multi-Input Predicate Encryption for Circuits. 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 $\mu' \in \{0,1\}^n \cup \perp$.

We prove the following security guarantee for miPE: Consider an adversary $\mathcal{A}$ which outputs q_0 key queries $\{f_0, \ldots, f_{q_0}\}$, q_i pairs of attribute-message queries for i-th slot $\{(x_1^{j_1}, \mu_1^{j_1})\}_{j_1 \in [q_1]}, \ldots, \{(x_n^{j_n}, \mu_n^{j_n})\}_{j_n \in [q_n]}$ and auxiliary information $\mathsf{aux}_\mathcal{A}$. We say that a miPE scheme is secure if the following holds for the adversary $\mathcal{A}$

$$\left(\mathsf{mpk}, \left\{f_k, \mathsf{sk}_{f_k}\right\}_{k \in [q_0]}, \left\{\mathsf{ct}_i^{j_i} \leftarrow \mathsf{Enc}_i(\mathsf{msk}, x_i^{j_i}, \mu_i^{j_i})\right\}_{i \in [n], j_i \in [q_i]}, \mathsf{aux}_\mathcal{A}\right)$$
$$\approx_c \left(\mathsf{mpk}, \left\{f_k, \mathsf{sk}_{f_k}\right\}_{k \in [q_0]}, \left\{\delta_i^{j_i} \leftarrow \mathsf{Sim}(\mathsf{msk})\right\}_{i \in [n], j_i \in [q_i]}, \mathsf{aux}_\mathcal{A}\right)$$

given $f_k(x_1^{j_1}, \ldots, x_n^{j_n}) = 0$ for every $i \in [n], j_i \in [q_i]$, and $k \in [q_0]$, where $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, \mathsf{prm})$ and $\mathsf{sk}_{f_k} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f_k)$.

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. The miPE of this work is obtained by first building a multi-input ABE, which is then compiled into miPE using a generic compiler by [AYY22]. Their multi-input ABE for constant arity is quite complex and leverages intricate algebraic properties of the underlying building blocks. Moreover, the limitation for a constant k seems inherent to their techniques, since the parameters grow exponentially in n. In contrast, our construction is extremely simple and 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 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 and security follow readily from those of the underlying building blocks. Please see Section 4 for details. In summary, we obtain the following theorems.

Theorem 1.11 (miPE **for poly arity).** Let $\kappa = \lambda^{n^2 \log \lambda}$. Assuming non-uniform κ-evasive LWE, non-uniform sub-exponential PRF, and non-uniform sub-exponential LWE, there exists a miPE scheme for arity $n = \mathrm{poly}(\lambda)$, supporting functions of bounded polynomial depth.

Using prMIFE scheme for constant arity allows to relax the assumption to normal evasive LWE.

Theorem 1.12 (miPE **for constant arity).** Assuming evasive LWE and LWE, there exists a prMIFE scheme for arity $n = O(1)$, supporting functions of bounded polynomial depth.

Two Party ID based Non-Interactive Key Exchange. Next, we provide a construction of two party ID based non-interactive key exchange (ID-NIKE) scheme. The construction is the same as the ID-based NIKE system by Sakai, Ohgishi, and Kasahara [SOK00] except that the hash function is replaced with an obfuscation of a PRF, which can be supported by our prIO. In more detail, we show:

Theorem 1.13. Let $\kappa = \lambda^{n^2 \log \lambda}$. Assuming non-uniform κ-evasive LWE (Assumption 2.4), non-uniform sub-exponential PRF, non-uniform sub-exponential LWE, and the DBDH assumption, there exists a secure ID-NIKE scheme.

This leads to the first construction of ID-NIKE without multilinear maps [FHPS13] or indistinguishability obfuscation in the standard model. Please see Section 8 for details.

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 7.1, 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.

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 revert to the original ROM-based proof strategy [BLS01, Cor00]. Unlike the iO-based approach, prIO-based proof does not require the truth table of the hash function to remain unchanged across game hops; it only requires the truth table to be pseudorandom. This relaxed condition enables the use of the original ROM security proofs. Please see Section 7 for details.

1.4 Organization of the Paper

We provide the preliminaries used in this work in Section 2 and Appendix A. In Section 3, we define the notion of multi-input FE for pseudorandom functionalities (prMIFE) and construct a (bounded-depth) prMIFE using a single-input FE scheme for pseudorandom functionalities (prFE) – a tool from our companion paper [AKY24b]. In Appendix B, we recall the construction of prFE from [AKY24b] and prove that it achieves strengthened security notion of non-uniform κ-prCT security which is required for the prMIFE compiler in Section 3. In Section 4, we give the construction for multi-input predicate encryption scheme for polynomial arity. In Section 5, we give the definitions for indistinguishability obfuscation for pseudorandom functionalities (prIO) for circuits and construct this using a prMIFE scheme supporting polynomial arity. In Section 6, we define and construct indistinguishability obfuscation for pseudorandom functionalities with polynomial size domain (pPRIO) using a prMIFE scheme supporting constant arity. In Section 7 and 8 we show how to instantiate the random oracle using a prIO scheme via various applications.

2 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 $\mathsf{poly}(n)$ 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.

2.1 Assumptions

In this section, we define the assumptions that we use in this work.

Assumption 2.1 (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:

$$\mathsf{Adv}_{\mathcal{A}_0}^{\mathrm{PRE}}(\lambda) \stackrel{\text{def}}{=} \Pr\left[\mathcal{A}_0(\mathbf{B}, \mathbf{SB}+\mathbf{E}, \mathbf{SP}+\mathbf{E}', \mathsf{aux}) = 1\right] - \Pr\left[\mathcal{A}_0(\mathbf{B}, \mathbf{C}_0, \mathbf{C}', \mathsf{aux}) = 1\right] \tag{13}$$

$$\mathsf{Adv}_{\mathcal{A}_1}^{\mathrm{POST}}(\lambda) \stackrel{\text{def}}{=} \Pr[\mathcal{A}_1(\mathbf{B}, \mathbf{SB}+\mathbf{E}, \mathbf{K}, \mathsf{aux}) = 1] - \Pr[\mathcal{A}_1(\mathbf{B}, \mathbf{C}_0, \mathbf{K}, \mathsf{aux}) = 1] \tag{14}$$

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

$$\mathsf{Adv}_{\mathcal{A}_0}^{\mathrm{PRE}}(\lambda) \geq \mathsf{Adv}_{\mathcal{A}_1}^{\mathrm{POST}}(\lambda)/Q(\lambda) - \mathsf{negl}(\lambda) \quad \text{and} \quad \mathsf{Time}(\mathcal{A}_0) \leq \mathsf{Time}(\mathcal{A}_1) \cdot Q(\lambda). \tag{15}$$

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

Remark 2.2. 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 2.3 (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:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{PRE}'}(\lambda) \stackrel{\text{def}}{=} \Pr\left[\mathcal{A}(\mathbf{B}, \mathbf{SB}+\mathbf{E}, \mathbf{SP}+\mathbf{E}', \mathsf{aux}_1, \mathsf{aux}_2) = 1\right] - \Pr\left[\mathcal{A}(\mathbf{B}, \mathbf{C}_0, \mathbf{C}', \mathbf{c}, \mathsf{aux}_2) = 1\right] \tag{16}$$

$$\mathsf{Adv}^{\mathrm{POST}'}_{\mathcal{A}}(\lambda) \stackrel{\mathrm{def}}{=} \Pr[\mathcal{A}(\mathbf{B}, \mathbf{SB}+\mathbf{E}, \mathbf{K}, \mathsf{aux}_1, \mathsf{aux}_2) = 1] - \Pr[\mathcal{A}(\mathbf{B}, \mathbf{C}_0, \mathbf{K}, \mathbf{c}, \mathsf{aux}_2) = 1] \tag{17}$$

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 2.1) with respect to $\mathsf{Samp} \in \mathcal{SC}$ for a sampler class $\mathcal{SC}$, if $\mathsf{Adv}^{\mathrm{PRE}'}_{\mathcal{A}}(\lambda)$ is negligible for any PPT adversary $\mathcal{A}$, so is $\mathsf{Adv}^{\mathrm{POST}'}_{\mathcal{A}}(\lambda)$ for any PPT adversary $\mathcal{A}$.

The following variant of the evasive LWE assumption will be used in Section 3. This strengthens Assumption 2.1 in that it considers non-uniform samplers and replaces the negligible term in Equation (15) 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. We refer to Remark 3.14 for the detailed discussion on why we need the stronger security definition for prFE for the application to prMIFE.

Assumption 2.4 (Non-Uniform κ-Evasive LWE). Let $n, m, t, m', q, \lambda \in \mathbb{N}$ be parameters defined as in Assumption 2.1 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 2.1. 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 $\mathsf{Adv}^{\mathrm{POST}}_{\mathcal{A}_1}(\lambda)$ as in Equation (13) and Equation (14), 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 $\mathrm{Size}(\mathsf{Samp}) \le \mathrm{poly}(\lambda')$ and $\mathrm{Size}(\mathcal{A}_1) \le \mathrm{poly}(\kappa)$ for $\lambda'(\lambda) \le \kappa(\lambda)$, there exists another non-uniform adversary $\mathcal{A}_0$ and a polynomial $Q(\cdot)$ such that

$$\mathsf{Adv}^{\mathrm{PRE}}_{\mathcal{A}_0}(\lambda) \ge \mathsf{Adv}^{\mathrm{POST}}_{\mathcal{A}_1}(\lambda)/Q(\lambda') - \mathsf{negl}(\kappa) \quad \text{and} \quad \mathrm{Size}(\mathcal{A}_0) \le Q(\lambda') \cdot \mathrm{Size}(\mathcal{A}_1).$$

Note that in the case λ' is superpolynomial in λ, $\mathsf{Samp}(1^\lambda)$ 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 2.3 for the stronger version of evasive LWE assumption defined in Assumption 2.4. The proof is almost the same as that for Lemma 2.3. We provide it here for completeness.

Lemma 2.5. Let $n, m, t, m', q, \lambda \in \mathbb{N}$ be parameters defined as in Assumption 2.1 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 2.3. For a non-uniform adversaries $\mathcal{A}$, we define the advantage functions $\mathsf{Adv}^{\mathrm{PRE}'}_{\mathcal{A}}(\lambda)$ and $\mathsf{Adv}^{\mathrm{POST}'}_{\mathcal{A}}(\lambda)$ as in Equation (16) and Equation (17), respectively. Then, for a function $\kappa := \kappa(\lambda)$ of the security parameter λ, under the non-uniform κ-evasive LWE assumption (Assumption 2.4), if $\mathrm{Size}(\mathsf{Samp}) \le \mathrm{poly}(\lambda')$ and $\mathrm{Size}(\mathcal{A}_1) \le \mathrm{poly}(\kappa)$ for $\lambda'(\lambda) \le \kappa(\lambda)$, there exists another non-uniform adversary $\mathcal{A}_0$ and a polynomial $Q(\cdot)$ such that

$$\mathsf{Adv}^{\mathrm{PRE}'}_{\mathcal{A}_0}(\lambda) \ge \mathsf{Adv}^{\mathrm{POST}'}_{\mathcal{A}_1}(\lambda)/Q(\lambda') - \mathsf{negl}(\kappa) \quad \text{and} \quad \mathrm{Size}(\mathcal{A}_0) \le Q(\lambda') \cdot \mathrm{Size}(\mathcal{A}_1).$$

Proof. Let us consider an adversary $\mathcal{A}_1$ and a sampler Samp with size being polynomial in κ and λ' respectively and $\epsilon = \mathsf{Adv}^{\mathrm{POST}'}_{\mathcal{A}_1}$. 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 $Q(\lambda') \cdot \mathsf{Size}(\mathcal{A}_1)$ 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. □

Assumption 2.6 (Decisional Bilinear Diffie-Hellman). Let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a bilinear map, where $e(g^a, g^b) = e(g, g)^{ab}$ and $a, b \in \mathbb{Z}_p$, for a cyclic group $\mathbb{G}$ of order p with a generator g. The Decisional Bilinear Diffie-Hellman assumption (DBDH) states that for any PPT adversary

$$\left(g^\alpha, g^\beta, g^\gamma, e(g,g)^{\alpha\beta\gamma} \mid \alpha, \beta, \gamma \leftarrow \mathbb{Z}_p\right) \approx_c \left(g^\alpha, g^\beta, g^\gamma, T \mid \alpha, \beta, \gamma \leftarrow \mathbb{Z}_p, T \leftarrow \mathbb{G}_T\right)$$

2.2 Puncturable Pseudorandom Functions

Syntax. A puncturable pseudorandom function (PPRF) $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ 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.

$\mathsf{Eval}(K_x, x') \to y$. The evaluation algorithm takes as input a punctured key K_x an input $x' \in \mathcal{X}$, such that $x \neq x'$ and outputs $y \in \mathcal{Y}$. It outputs $\perp$ if $x = x'$.

Definition 2.7. (Correctness) A PPRF 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

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

Definition 2.8. (Security) A PPRF 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 $\beta = \beta'$.

We know that PPRF with the security defined above from one-way functions [GGM84, BW13, BGI14, KPTZ13].

2.3 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 2.9. (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

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

Definition 2.10. (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

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

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^{λ^δ} for sufficiently large λ.

2.4 Pseudorandom Functional Encryption

In this section we give definitions for functional encryption for pseudorandom functionalities, adapted from [AKY24b].

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[7].

$\mathsf{KeyGen}(\mathsf{msk}, f) \to \mathsf{sk}_f$. 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, a functional secret key sk_f, function f and a ciphertext ct, and outputs $y \in \mathcal{Y}_{\mathsf{prm}}$.

Definition 2.11 (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 notion next. At a high level, our 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.

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

Definition 2.12 (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

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

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 $i \in [Q_{\mathsf{key}}], j \in [Q_{\mathsf{msg}}]$.
We define the following advantage functions:

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

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

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 $\mathcal{F}_{\mathsf{prm}}$ satisfies prCT security with respect to the sampler class $\mathcal{SC}$ if for every PPT sampler $\mathsf{Samp} \in \mathcal{SC}$ there exists a polynomial $Q(\cdot)$ such that for every PPT adversary $\mathcal{A}_1$, there exists another PPT $\mathcal{A}_0$ such that

$$\mathcal{A}^{\mathrm{PRE}}_{\mathcal{A}_0}(\lambda) \geq \mathcal{A}^{\mathrm{POST}}_{\mathcal{A}_1}(\lambda)/Q(\lambda) - \mathsf{negl}(\lambda) \tag{18}$$

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

Remark 2.13 ([AKY24b]). It is shown in [AMYY25] that there is no prFE that satisfies prCT security for all general samplers. Therefore, when we use the security of prFE, we invoke the security *with respect to a specific sampler class* that is induced by the respective applications. For simplicity, we sometimes will treat as if there was prFE that is secure for all the samplers.

Theorem 2.14 ([AKY24b]). Assuming LWE and evasive LWE assumptions, there exists a secure (Definition 2.12) prFE scheme, with respect to a specific sampler class, for function class $\mathcal{F}_{\mathrm{L}(\lambda),\ell(\lambda),\mathsf{dep}(\lambda)} = \{f : \{0,1\}^{\mathrm{L}} \to \{0,1\}^\ell\}$ satisfying

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

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

Definition 2.15 (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 $\kappa \stackrel{\mathrm{def}}{=} \kappa(\lambda)$ 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:

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

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

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 $\mathrm{Size}(\mathsf{Samp}) \leq \mathrm{poly}(\lambda')$ and $\mathrm{Size}(\mathcal{A}_1) \leq \mathrm{poly}(\kappa)$ for $\lambda' \leq \kappa$, there exists another adversary $\mathcal{A}_0$ such that

$$\mathsf{Adv}^{\mathrm{PRE}}_{\mathcal{A}_0}(\lambda) \geq \mathsf{Adv}^{\mathrm{POST}}_{\mathcal{A}_1}(\lambda)/Q(\lambda') - \mathsf{negl}(\kappa) \tag{19}$$

and $\mathrm{Size}(\mathcal{A}_0) \leq \mathrm{Size}(\mathcal{A}_1) \cdot Q(\lambda')$ for some polynomial $Q(\cdot)$.

In Appendix B.3, we prove the following theorem.

Theorem 2.16. Let $\kappa = 2^{\lambda^c}$ for some constant c. Assuming non-uniform κ-evasive LWE (Assumption 2.4), subexponentially secure PRF against non-uniform adversary, and non-uniform sub-exponential LWE (Assumption A.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 2.15, with respect to a specific sampler class, 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.

Remark 2.17 (Comparison between Definition 2.15 and Definition 2.12). We remark that Definition 2.15 strengthens Definition 2.12 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 (18) is replaced by $\mathsf{negl}(\kappa)$ in Equation (19). 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 3. We refer to Remark 3.14 for the discussion on why these changes are necessary for the security proof there.

2.5 Predicate Encryption

Consider a function family $\{\mathcal{F}_{\mathsf{prm}} = \{f : \mathcal{X}_{\mathsf{prm}} \to \{0,1\}\}\}_{\mathsf{prm}}$, for a parameter $\mathsf{prm} = \mathsf{prm}(\lambda)$.

Syntax. A PE scheme PE for function family $\mathcal{F}_{\mathsf{prm}}$ consists of 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 master secret key msk[8].

$\mathsf{KeyGen}(\mathsf{msk}, f) \to \mathsf{sk}_f$.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.

$\mathsf{Enc}(\mathsf{mpk}, x, \mu) \to \mathsf{ct}$. The encryption algorithm takes as input a master secret key msk, an attribute $x \in \mathcal{X}_{\mathsf{prm}}$, and message $\mu \in \{0,1\}$, and outputs a ciphertext ct.

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

Definition 2.18 (Correctness.). For every $\lambda \in \mathbb{N}, \mu \in \{0,1\}, x \in \mathcal{X}_{\mathsf{prm}}, f \in \mathcal{F}_{\mathsf{prm}}$, if $f(x) = 1$, then

$$\Pr\left[\mathsf{Dec}\left(\mathsf{mpk}, \mathsf{KeyGen}(\mathsf{msk}, f), f, \mathsf{Enc}(\mathsf{mpk}, x, \mu)\right) = \mu\right] = 1 - \mathsf{negl}(\lambda)$$

where $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{prm})$.

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

Definition 2.19 (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

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

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 $\mathcal{A}$, it holds that $f(\mathbf{x}) = 0$.

Predicate encryption schemes for (bounded depth) circuits satisfying the above security notion are known from LWE [GVW15a, GKW17, WZ17].

2.6 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

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

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 2.20 (Sim**-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 $\mathcal{A}$, and a simulator algorithm $\{\mathsf{Sim}_i\}_{i \in [n]}$, we define the Sim-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 $(x_i^{j_i}, \mu_i^{j_i})$ to denote the j_i-th ciphertext query corresponding to the i-th slot, where $j_i \in [q_i]$ and $i \in [n]$.

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) If $\beta = 0$, it computes $\mathsf{ct}_i^{j_i} \leftarrow \mathsf{Enc}_i(\mathsf{msk}, x_i^{j_i}, \mu_i^{j_i})$, else if $\beta = 1$, it computes $\mathsf{ct}_i^{j_i} \leftarrow \mathsf{Sim}_i(\mathsf{msk})$ for $i \in [n], j_i \in [q_i]$.

   It returns $(\mathsf{mpk}, \{\mathsf{sk}_{f_k}\}_{k\in[q_0]}, \{\mathsf{ct}_i^{j_i}\}_{i\in[n], j_i\in[q_i]})$ 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 $f_k(x_1^{j_1}, \ldots, x_n^{j_n}) = 0$ for every $i \in [n], j_i \in [q_i]$, and $k \in [q_0]$. We define the advantage $\mathsf{Adv}^{\mathsf{Sim}}_{\mathsf{miPE},\mathcal{A}}$ of $\mathcal{A}$ in the security game as

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

The miPE scheme is said to satisfy Sim-security if for any stateful PPT adversary $\mathcal{A}$, there exists a PPT simulator algorithm $\{\mathsf{Sim}_i\}_{i\in[n]}$ such that $\mathsf{Adv}^{\mathsf{Sim}}_{\mathsf{miPE},\mathcal{A}}(1^\lambda) = \mathsf{negl}(\lambda)$.

2.7 ID-Based Non-Interactive Key Exchange

In this section, we give the definitions for an identity-based non-interactive key exchange scheme, for two parties, adapted from [FHPS13].

Syntax. An identity-based non-interactive key exchange (ID-NIKE) scheme for identity space $\mathcal{ID}$ has the following syntax.

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

$\mathsf{Extract}(\mathsf{mpk}, \mathsf{msk}, \mathsf{id}) \rightarrow \mathsf{usk}_{\mathsf{id}}$. The key extraction algorithm takes as input the master public key mpk, the master secret key msk and an identity $\mathsf{id} \in \mathcal{ID}$. It outputs a user secret key $\mathsf{usk}_{\mathsf{id}}$ for id.

$\mathsf{Share}(\mathsf{mpk}, \mathsf{usk}_{\mathsf{id}_1}, \mathsf{id}_2) \rightarrow K$. The share algorithm takes as input the master public key mpk, a user secret key $\mathsf{usk}_{\mathsf{id}_1}$ for an identity $\mathsf{id}_1 \in \mathcal{ID}$ and an identity $\mathsf{id}_2 \in \mathcal{ID}$. It outputs a shared key K.

Definition 2.21 (Correctness). An ID-NIKE scheme for an identity space $\mathcal{ID}$ is correct if for any $\lambda \in \mathbb{N}$, $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$, any $\mathsf{id}_1, \mathsf{id}_2 \in \mathcal{ID}$ we have

$$\mathsf{Share}(\mathsf{mpk}, \mathsf{usk}_{\mathsf{id}_1}, \mathsf{id}_2) = \mathsf{Share}(\mathsf{mpk}, \mathsf{usk}_{\mathsf{id}_2}, \mathsf{id}_1)$$

where $\mathsf{usk}_{\mathsf{id}_1} \leftarrow \mathsf{Extract}(\mathsf{mpk}, \mathsf{msk}, \mathsf{id}_1)$ and $\mathsf{usk}_{\mathsf{id}_2} \leftarrow \mathsf{Extract}(\mathsf{mpk}, \mathsf{msk}, \mathsf{id}_2)$.

Definition 2.22 (Security). We say that an ID-NIKE scheme for an identity space $\mathcal{ID}$ is secure if the advantage function

$$\mathrm{Adv}_{\text{ID-NIKE},\mathcal{A}}(\lambda) = \Pr\left[\mathrm{Exp}_{\text{ID-NIKE},\mathcal{A}}(\lambda) = 1\right] - \frac{1}{2}$$

is negligible for all PPT adversaries $\mathcal{A}$. Here, experiment $\mathrm{Exp}_{\text{ID-NIKE},\mathcal{A}}$ is defined as follows:

1. **Setup Phase.** The experiment samples $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$, and gives $\mathcal{A}$ the mpk.

2. **Query Phase.** The adversary can make the following queries in an arbitrary order.

   – Extraction Query: $\mathcal{A}$ sends an extraction query for an identity $\mathsf{id} \in \mathcal{ID}$. The experiment returns $\mathsf{usk}_{\mathsf{id}}$ to the adversary, where $\mathsf{usk}_{\mathsf{id}} \leftarrow \mathsf{Extract}(\mathsf{mpk}, \mathsf{msk}, \mathsf{id})$.

   – Challenge Query : $\mathcal{A}$ sends a pair $(\mathsf{id}_1^\star, \mathsf{id}_2^\star) \in \mathcal{ID} \times \mathcal{ID}$ as the challenge query. The experiment samples a bit $\beta \leftarrow \{0,1\}$ and returns $K^\star \leftarrow \mathsf{Share}(\mathsf{mpk}, \mathsf{usk}_{\mathsf{id}_1^\star}, \mathsf{id}_2^\star)$, where $\mathsf{usk}_{\mathsf{id}_1^\star} \leftarrow \mathsf{Extract}(\mathsf{mpk}, \mathsf{msk}, \mathsf{id}_1^\star)$, if $\beta = 0$ and returns $K^\star \leftarrow \mathbb{G}_T$ if $\beta = 1$.

3. **Output Phase.** $\mathcal{A}$ outputs a guess bit β', and the experiment outputs 1 if $\beta = \beta'$.

We only quantify over $\mathcal{A}$ that guarantees that does not make extraction queries for $\mathsf{id}_1^\star$ and $\mathsf{id}_2^\star$.

3 Multi-Input FE for Pseudorandom Functionalities

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

3.1 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 3.1 (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 $(x_1, \ldots, x_n) \in \mathcal{X}_{\mathsf{prm}}^n$ 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 3.2 (κ-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

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

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

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

$$\text{If} \quad \left(1^\kappa, \left\{f_k, f_k(x_1^{j_1}, \ldots, x_n^{j_n})\right\}_{k\in[q_0], j_1\in[q_1]\ldots, j_n\in[q_n]}, \mathsf{aux}\right) \approx_c \left(1^\kappa, \left\{f_k, \Delta_{k,j_1,\ldots,j_n}\right\}_{k\in[q_0], j_1\in[q_1],\ldots,j_n\in[q_n]}, \mathsf{aux}\right) \quad (20)$$

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

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 $k \in [q_0]$.

Remark 3.3. Note that 1^κ in Equation (20) 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 (20) 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 3.2 for general n requires large κ as an artifact of the proof technique. In the special case of n being constant, we can achieve $\kappa = \lambda^n$.

The following variant of security will be necessary in Section 6.

Definition 3.4 (Pseudorandomness of the Last Slot Ciphertext). We say that a prMIFE scheme satisfies κ-pseudorandomness of the last slot ciphertext property if it satisfies κ-security as per defined in Definition 3.2 where the simulator Sim_n corresponding to the last slot ciphertext outputs a random string of the same length as $\mathsf{ct}_n^{j_n}$.

Remark 3.5. It is shown in [AMYY25, BDJ+24] that there is no prMIFE that satisfies the above style security for all general samplers. Therefore, when we use the security of prMIFE, we invoke the security *with respect to a specific sampler class* that is induced by the respective applications. For simplicity, we sometimes will treat as if there was prMIFE that is secure for all the samplers.

3.2 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. A secret key encryption scheme $\mathsf{SKE} = (\mathsf{SKE.Setup}, \mathsf{SKE.Enc}, \mathsf{SKE.Dec})$. When we set up the scheme SKE, we run it on a scaled version of the security parameter Λ, instead of the usual security parameter λ. We will explain how to set Λ in Remark 3.6. We denote the ciphertext space of the scheme by $\mathcal{CT}_{\mathsf{SKE},\Lambda}$ and the key space of the scheme by $\mathcal{K}_{\mathsf{SKE},\Lambda}$. In the following, we drop Λ and denote them as $\mathcal{CT}_{\mathsf{SKE}}$ and $\mathcal{K}_{\mathsf{SKE}}$ respectively.

[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 2 for the details.

2. 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 $\mathsf{prFE}_{i-1}.\mathsf{ct} \in \mathcal{CT}_{\mathsf{prFE}_{i-1}}$.
3. 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 1.[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 $\mathsf{ct}_n = \mathsf{prFE}_n.\mathsf{ct}$.
- For $i = n, \ldots, 2$ and do the following.
  1. Compute $\mathsf{prFE}_{i-1}.\mathsf{ct} := \mathsf{prFE}_i.\mathsf{Dec}(\mathsf{prFE}_i.\mathsf{mpk}, \mathsf{prFE}_i.\mathsf{sk}, \mathrm{F}_{i-1}, \mathsf{prFE}_i.\mathsf{ct})$.

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 1: Function F_i

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 3.6. 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^{λ^δ} 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 3.7. Suppose prFE_i for $i \in [n]$ and SKE are correct, then the above construction of prMIFE satisfies correctness as defined in Definition 3.1.

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

Claim 3.8. For $i = n, \ldots, 2$, we have

$$\Pr[\mathsf{prFE}_i.\mathsf{Dec}(\mathsf{prFE}_i.\mathsf{mpk}, \mathsf{prFE}_i.\mathsf{sk}, \mathrm{F}_{i-1}, \mathsf{prFE}_i.\mathsf{ct}) = \mathsf{prFE}_{i-1}.\mathsf{ct}] = 1 \tag{21}$$

where

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

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

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

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

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

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

which proves the base case.
Inductive Step: For the inductive step, suppose Equation (21) holds for some $i \in [3, n]$ then we prove the same statement for $i - 1$. Consider

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

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 $\mathsf{prFE}_{i-1}.\mathsf{ct} = \mathsf{prFE}_{i-1}.\mathsf{Enc}$ $(\mathsf{prFE}_{i-1}.\mathsf{mpk}, (\mathsf{SKE}.\mathsf{sk}, (\mathbf{x}_{i-1}, \mathbf{r}_{i-1}), \ldots, (\mathbf{x}_{n-1}, \mathbf{r}_{n-1}), \mathbf{x}_n, K_1, \ldots K_{i-2}); \mathsf{PRF}_{i-1}(K_{i-1}, (\mathbf{r}_{i-1}, \ldots, \mathbf{r}_{n-1})))$ 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 $\mathsf{prFE}.\mathsf{ct}_1 = \mathsf{prFE}_1.\mathsf{Enc}(\mathsf{prFE}_1.\mathsf{mpk}, (\mathbf{x}_1, \ldots, \mathbf{x}_n); \mathsf{PRF}_1(K_1, (\mathbf{r}_1, \ldots, \mathbf{r}_{n-1})))$ from Step 3.2 of the decryption algorithm with probability 1. From the correctness of prFE_1 scheme, the decryption Step 3.2 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. □

3.3 Security Proof for General n

Theorem 3.9. Let $\mathcal{SC}_{\mathsf{prMIFE}}$ be a sampler class for prMIFE. Suppose prFE_i scheme satisfies non-uniform κ-prCT security as per Definition 2.15 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 (30), SKE satisfies sub-exponential INDr security and PRF_i is sub-exponentially secure, then prMIFE constructed above satisfies security for $\kappa = \lambda^{n^2 \log \lambda}$ as in Definition 3.4. Note that this in particular implies the κ-security defined in Definition 3.2.

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

1. **Key Queries.** It issues q_0 number of functions $f_1, \dots, f_{q_0}$ 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 as per Definition 3.4, we first define $\{\mathsf{Sim}_i\}_{i\in[n]}$ as follows. Observe that Sim_n outputs random string as is required by Definition 3.4.

$\mathsf{Sim}_i(\mathsf{msk}) \to \mathsf{ct}_i$ 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

$$\begin{pmatrix} \mathsf{aux}_{\mathcal{A}},\ \{\mathsf{prFE}_i.\mathsf{mpk}\}_{i\in[n]},\ \left\{\mathsf{prFE}_n.\mathsf{ct}^{j_n}\right\}_{j_n\in[q_n]}, \\ \left\{f_k,\ \mathsf{sk}_{f_k} = \mathsf{prFE}_1.\mathsf{sk}_{f_k}\right\}_{k\in[q_0]}, \left\{\mathsf{SKE.ct}_i^{j_i},\ \mathbf{r}_i^{j_i},\ \mathsf{prFE}_{i+1}.\mathsf{sk}^{j_i}\right\}_{\substack{i\in[n-1],\\ j_i\in[q_i]}}, \end{pmatrix}$$
$$\approx_c \begin{pmatrix} \mathsf{aux}_{\mathcal{A}},\ \{\mathsf{prFE}_i.\mathsf{mpk}\}_{i\in[n]},\ \left\{\delta_n^{j_n}\right\}_{j_n\in[q_n]} \\ \left\{f_k,\ \mathsf{sk}_{f_k} = \mathsf{prFE}_1.\mathsf{sk}_{f_k}\right\}_{k\in[q_0]}, \left\{\gamma_i^{j_i},\ \mathbf{r}_i^{j_i},\ \mathsf{prFE}_{i+1}.\mathsf{sk}^{j_i}\right\}_{\substack{i\in[n-1],\\ j_i\in[q_i]}} \end{pmatrix} \tag{22}$$

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

$$\begin{aligned}
&\left(\mathsf{mpk} = \{\mathsf{prFE}_i.\mathsf{mpk}\}_i, \mathsf{msk} = (\mathsf{SKE.sk}, \{\mathsf{prFE}_i.\mathsf{msk}, \mathsf{prFE}_i.\mathsf{mpk}\}_i)\right) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, \mathsf{prm}),\\
&\mathsf{prFE}_1.\mathsf{sk}_{f_k} \leftarrow \mathsf{prFE}_1.\mathsf{KeyGen}(\mathsf{prFE}_1.\mathsf{msk}, f_k) \quad \text{for } k \in [q_0],\\
&\mathsf{SKE.ct}_i^{j_i} \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.sk}, \mathbf{x}_i^{j_i}), \quad \gamma_i^{j_i} \leftarrow \mathcal{CT}_{\mathsf{SKE}}, \quad \mathbf{r}_i^{j_i} \leftarrow \{0,1\}^\Lambda,\\
&\mathsf{prFE}_{i+1}.\mathsf{sk}^{j_i} \leftarrow \begin{cases} \mathsf{prFE}_{i+1}.\mathsf{KeyGen}(\mathsf{prFE}_{i+1}.\mathsf{msk}, \mathrm{F}_i[\mathsf{SKE.ct}_i^{j_i}, \mathbf{r}_i^{j_i}, \mathsf{prFE}_i.\mathsf{mpk}]) & \text{in LHS of Eq. (22)}\\ \mathsf{prFE}_{i+1}.\mathsf{KeyGen}(\mathsf{prFE}_{i+1}.\mathsf{msk}, \mathrm{F}_i[\gamma_i^{j_i}, \mathbf{r}_i^{j_i}, \mathsf{prFE}_i.\mathsf{mpk}]) & \text{in RHS of Eq. (22)} \end{cases}\\
&\mathsf{prFE}_n.\mathsf{ct}^{j_n} \leftarrow \mathsf{prFE}_n.\mathsf{Enc}(\mathsf{prFE}_n.\mathsf{mpk}, (\mathsf{SKE.sk}, \mathbf{x}_n^{j_n}, K_1^{j_n}, \dots, K_{n-1}^{j_n})),\ \delta_n^{j_n} \leftarrow \mathcal{CT}_{\mathsf{prFE}_n},\\
&K_i^{j_n} \leftarrow \{0,1\}^\Lambda, \quad \text{for } i \in [n-1],\ j_i \in [q_i],\ \text{and}\ \ j_n \in [q_n]
\end{aligned}$$

assuming we have

$$\left(1^\kappa, \mathsf{aux}_{\mathcal{A}}, \left\{ f_k,\ f_k(\mathbf{x}_1^{j_1}, \dots \mathbf{x}_n^{j_n}) \right\}_{k\in[q_0], j_1\in[q_1],\dots,j_n\in[q_n]}\right)$$
$$\approx_c \left(1^\kappa, \mathsf{aux}_{\mathcal{A}}, \left\{ f_k,\ \Delta_k^{j_1,\dots,j_n} \right\}_{k\in[q_0], j_1\in[q_1],\dots,j_n\in[q_n]}\right) \tag{23}$$

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

- **Step 1.** We first show that Equation (23) implies

$$\begin{pmatrix} 1^\kappa, \quad \mathsf{aux}_\mathcal{A}, \quad \mathsf{prFE.mpk}_1 \\ \{\mathsf{SKE.ct}_i^{j_i}\}_{i\in[n-1],j_i\in[q_i]} \quad \{\mathsf{prFE}_1.\mathsf{ct}^{\mathbf{j}}\}_{\mathbf{j}\in[q_1]\times\cdots\times[q_n]} \\ \left\{f_k,\ \mathsf{sk}_{f_k} = \mathsf{prFE}_1.\mathsf{sk}_{f_k}\right\}_{k\in[q_0]} \end{pmatrix} \approx_c \begin{pmatrix} 1^\kappa, \quad \mathsf{aux}_\mathcal{A}, \quad \mathsf{prFE.mpk}_1 \\ \{\mathsf{SKE.ct}_i^{j_i}\}_{i\in[n-1],j_i\in[q_{\mathsf{msg}}]}, \quad \{\Delta^{\mathbf{j}}\}_{\mathbf{j}\in[q_1]\times\cdots\times[q_n]} \\ \left\{f_k,\ \mathsf{sk}_{f_k} = \mathsf{prFE}_1.\mathsf{sk}_{f_k}\right\}_{k\in[q_0]} \end{pmatrix} \tag{24}$$

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 (24) implies Equation (22).

Step 1. We show the following lemma.

Lemma 3.10. If SKE satisfies subexponential INDr security, Equation (23) implies Equation (24).

Proof. We first prove the following:

$$\begin{aligned} &\left(1^\kappa, \mathsf{aux}_\mathcal{A}, \left\{ f_k,\ f_k(\mathbf{x}_1^{j_1}, \ldots \mathbf{x}_n^{j_n})\right\}_{k\in[q_0],j_1\in[q_1],\ldots,j_n\in[q_n]}, \ \left\{\mathsf{SKE.ct}_i^{j_i} \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.sk}, \mathbf{x}_i^{j_i})\right\}_{i\in[n-1],j_i\in[q_i]}\right) \\ \approx_c &\left(1^\kappa, \mathsf{aux}_\mathcal{A}, \left\{ f_k,\ \Delta_k^{j_1,\ldots,j_n}\right\}_{k\in[q_0],j_1\in[q_1],\ldots,j_n\in[q_n]}, \ \left\{\mathsf{SKE.ct}_i^{j_i} \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.sk}, \mathbf{x}_i^{j_i})\right\}_{i\in[n-1],j_i\in[q_i]}\right), \end{aligned} \tag{25}$$

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

$$\left(1^\kappa, \mathsf{aux}_\mathcal{A}, \left\{ f_k,\ f_k(\mathbf{x}_1^{j_1}, \ldots, \mathbf{x}_n^{j_n})\right\}_{k,j_1,\ldots,j_n}, \ \left\{\mathsf{SKE.ct}_i^{j_i} \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.sk}, \mathbf{x}_i^{j_i})\right\}_{i,j_i}\right)$$

$$\approx_c \left(1^\kappa, \mathsf{aux}_\mathcal{A}, \left\{ f_k,\ f_k(\mathbf{x}_1^{j_1}, \ldots, \mathbf{x}_n^{j_n})\right\}_{k,j_1,\ldots,j_n}, \ \left\{\gamma_i^{j_i} \leftarrow \mathcal{CT}_{\mathsf{SKE}}\right\}_{i,j_i}\right) \tag{26}$$

$$\approx_c \left(1^\kappa, \mathsf{aux}_\mathcal{A}, \left\{ f_k,\ \Delta_k^{j_1,\ldots,j_n}\right\}_{k,j_1,\ldots,j_n}, \ \left\{\gamma_i^{j_i} \leftarrow \mathcal{CT}_{\mathsf{SKE}}\right\}_{i,j_i}\right) \tag{27}$$

$$\approx_c \left(1^\kappa, \mathsf{aux}_\mathcal{A}, \left\{ f_k,\ \Delta_k^{j_1,\ldots,j_n}\right\}_{k,j_1,\ldots,j_n}, \ \left\{\mathsf{SKE.ct}_i^{j_i} \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.sk}, \mathbf{x}_i^{j_i})\right\}_{i,j_i}\right). \tag{28}$$

Here, we justify each step of the equations above. We can see that Equation (26) follows from subexponential INDr security of SKE, since $\mathsf{SKE.sk}$ 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 (27) follows from Equation (23) by noting that adding random strings does not make the task of disginguishing the two distributions any easier. Finally, Equation (28) follows from INDr security of SKE again. We then consider a sampler Samp_1 that on input 1^κ outputs

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

By the security guarantee of prFE_1 with sampler Samp_1 and Equation (25), we obtain Equation (24). □

Step 2. To prove that Equation (24) implies Equation (22), we prove the following statement.

Lemma 3.11. For $h \in [n]$ and an adversary $\mathcal{A}$, let us consider the following distinguishing advantage:

$$\mathsf{Adv}^{h}_{\mathcal{A}}(\lambda) \stackrel{\text{def}}{=} \left| \mathcal{A}\begin{pmatrix} \mathsf{aux}_{\mathcal{A}},\ \{\mathsf{prFE.mpk}_i\}_{i\in[h]},\ \{\mathsf{SKE.ct}_i^{j_i}, \mathbf{r}_i^{j_i}\}_{i\in[n-1],j_i\in[q_i]} \\ \{f_k, \mathsf{sk}_{f_k} = \mathsf{prFE}_1.\mathsf{sk}_{f_k}\}_{k\in[q_0]}, \\ \{\mathsf{prFE}_{i+1}.\mathsf{sk}^{j_i}\}_{i\in[h-1],j_i\in[q_i]},\ \{\mathsf{prFE}_h.\mathsf{ct}^{\mathbf{j}}\}_{\mathbf{j}\in[q_h]\times\cdots\times[q_n]}, \end{pmatrix} \right.$$
$$\left. - \mathcal{A}\begin{pmatrix} \mathsf{aux}_{\mathcal{A}},\ \{\mathsf{prFE.mpk}_i\}_{i\in[h]},\ \{\mathsf{SKE.ct}_i^{j_i}, \mathbf{r}_i^{j_i}\}_{i\in[n-1],j_i\in[q_i]} \\ \{f_k, \mathsf{sk}_{f_k} = \mathsf{prFE}_1.\mathsf{sk}_{f_k}\}_{k\in[q_0]}, \\ \{\mathsf{prFE}_{i+1}.\mathsf{sk}^{j_i}\}_{i\in[h-1],j_i\in[q_i]},\ \{\Delta^{\mathbf{j}}\}_{\mathbf{j}\in[q_h]\times\cdots\times[q_n]} \end{pmatrix} \right| \tag{29}$$

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 $\mathrm{Size}(\mathcal{A}) < \mathrm{poly}(\kappa)$,[12] there exists another non-uniform adversary $\mathcal{B} = \{\mathcal{B}_\lambda\}_\lambda$ and a polynomial Q such that

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

assuming the security of prFE as per Definition 2.15 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

$$\begin{pmatrix} \text{Functions:} & \left\{\mathrm{F}_{h^*-1}^{j_{h^*-1}} = \mathrm{F}_{h^*-1}^{j_{h^*-1}}[\mathsf{SKE.ct}_{h^*-1}^{j_{h^*-1}}, \mathbf{r}_{h^*-1}^{j_{h^*-1}}, \mathsf{prFE}_{h^*-1}.\mathsf{mpk}]\right\}_{j_{h^*-1}\in[q_{h^*-1}]}, \\ \text{Inputs:} & \left\{\mathbf{x}^{j_{h^*},\ldots,j_n} \stackrel{\text{def}}{=} \left(\mathsf{SKE.sk}, (\mathbf{x}_{h^*}^{j_{h^*}}, \mathbf{r}_{h^*}^{j_{h^*}}), \ldots (\mathbf{x}_n^{j_n}, K_1^{j_n}, \ldots, K_{h^*-1}^{j_n})\right)\right\}_{j_{h^*}\in[q_{h^*}],\ldots,j_n\in[q_n]} \\ \text{Auxiliary Information:} & \mathsf{aux}_{h^*} \stackrel{\text{def}}{=} \begin{pmatrix} \mathsf{aux}_{\mathcal{A}}, \{\mathsf{prFE}_i.\mathsf{mpk}\}_{i\in[h^*-1]}, \{f_k, \mathsf{sk}_{f_k} = \mathsf{prFE}_1.\mathsf{sk}_{f_k}\}_{k\in[q_0]}, \\ \{\mathsf{SKE.ct}_i^{j_i}, \mathbf{r}_i^{j_i}\}_{i\in[n-1],j_i\in[q_i]}, \{\mathsf{prFE}_{i+1}.\mathsf{sk}^{j_i}\}_{i\in[h^*-2],j_i\in[q_i]} \end{pmatrix} \end{pmatrix}. \tag{30}$$

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:

$$\left(\mathsf{prFE}_{h^*}.\mathsf{mpk}, \left\{\mathrm{F}_{h^*-1}^{j_{h^*-1}}, \mathsf{prFE}_{h^*}.\mathsf{sk}^{j_{h^*-1}}\right\}_{j_{h^*-1}}, \left\{\mathsf{prFE}_{h^*}.\mathsf{Enc}(\mathbf{x}^{j_{h^*},\ldots,j_n})\right\}_{j_{h^*},\ldots,j_n}, \mathsf{aux}_{h^*}\right)$$
$$\text{and} \quad \left(\mathsf{prFE}_{h^*}.\mathsf{mpk}, \left\{\mathrm{F}_{h^*-1}^{j_{h^*-1}}, \mathsf{prFE}_{h^*}.\mathsf{sk}^{j_{h^*-1}}\right\}_{j_{h^*-1}}, \left\{\Delta^{j_{h^*},\ldots,j_n} \leftarrow \mathcal{CT}_{\mathsf{prFE}_{h^*}}\right\}_{j_{h^*},\ldots,j_n}, \mathsf{aux}_{h^*}\right). \tag{31}$$

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 (31) with advantage more than ϵ can be converted into another adversary $\mathcal{B}$ that can distinguish the following distributions with advantage more than $\epsilon' \stackrel{\text{def}}{=} \epsilon/Q(\lambda') - \mathsf{negl}(\kappa)$ satisfying $\mathrm{Size}(\mathcal{B}) \leq Q(\lambda')\mathrm{Size}(\mathcal{A})$ for some polynomial Q:

$$\left(\left\{\mathrm{F}_{h^*-1}^{j_{h^*-1}}\right\}_{j_{h^*-1}}, \left\{\mathrm{F}_{h^*-1}^{j_{h^*-1}}(\mathbf{x}^{j_{h^*},\ldots,j_n})\right\}_{j_{h^*-1},\ldots,j_n}, \mathsf{aux}_{h^*}\right)$$
$$\text{and} \quad \left(\left\{\mathrm{F}_{h^*-1}^{j_{h^*-1}}\right\}_{j_{h^*-1}}, \left\{\Delta^{j_{h^*-1},\ldots,j_n} \leftarrow \mathcal{CT}_{\mathsf{prFE}_{h^*}}\right\}_{j_{h^*-1},\ldots,j_n}, \mathsf{aux}_{h^*}\right). \tag{32}$$

[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 (31) are equivalent to those in Equation (29) 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 (29) 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 (32) is equivalent to that in Equation (29) with $h = h^* - 1$. Therefore, it suffices to prove that $\mathcal{A}$ cannot distinguish the first distribution in Equation (32) from the first distribution in Equation (29) 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 (32). Recall that we have

$$\mathsf{F}_{h^*-1}^{j_{h^*-1}}(\mathbf{x}^{j_{h^*},\ldots,j_n}) = \mathsf{prFE}_{h^*-1}.\mathsf{Enc}\left(\begin{array}{r}\mathsf{prFE}_{h^*-1}.\mathsf{mpk}, \left(\mathsf{SKE.sk}, (\mathbf{x}_{h^*-1}^{j_{h^*-1}}, \mathbf{r}_{h^*-1}^{j_{h^*-1}}), \ldots (\mathbf{x}_n^{j_n}, K_1^{j_n}, \ldots, K_{h^*-2}^{j_n})\right);\\ \mathsf{PRF}_{h^*-1}(K_{h^*-1}^{j_n}, (\mathbf{r}_{h^*-1}^{j_{h^*-1}}, \ldots, \mathbf{r}_{n-1}^{j_{n-1}}))\end{array}\right).$$

Hyb_2. This hybrid is the same as the previous one except that we replace $\mathsf{PRF}_{h^*-1}(K_{h^*-1}^{j_n}, \cdot)$ with the real random function $\mathsf{R}^{j_n}(\cdot)$ for each $j_n \in [q_{\mathsf{msg}}]$. Since $K_{h^*-1}^{j_n}$ 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 $(\mathbf{r}_{h^*-1}^{j_{h^*-1}}, \ldots, \mathbf{r}_{n-1}^{j_{n-1}}) = (\mathbf{r}_{h^*-1}^{j'_{h^*-1}}, \ldots, \mathbf{r}_{n-1}^{j'_{n-1}})$. 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 $\mathbf{r}_i^j = \mathbf{r}_i^{j'}$. 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 $\mathsf{F}_{h^*-1}^{j_{h^*-1}}(\mathbf{x}^{j_{h^*},\ldots,j_n})$ with

$$\mathsf{prFE}_{h^*-1}.\mathsf{ct}^{j_{h^*-1},\ldots,j_n} = \mathsf{prFE}_{h^*-1}.\mathsf{Enc}\left(\mathsf{prFE}_{h^*-1}.\mathsf{mpk}, \left(\mathsf{SKE.sk}, (\mathbf{x}_{h^*-1}^{j_{h^*-1}}, \mathbf{r}_{h^*-1}^{j_{h^*-1}}), \ldots (\mathbf{x}_n^{j_n}, K_1^{j_n}, \ldots, K_{h^*-2}^{j_n})\right)\right).$$

Namely, we use fresh randomness for each encryption instead of deriving the randomness by $\mathsf{R}^{j_n}(\mathbf{r}_{h^*-1}^{j_{h^*-1}}, \ldots, \mathbf{r}_{n-1}^{j_{n-1}})$. 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 (29) with $h = h^* - 1$, we complete the proof. □

Lemma 3.12. Assuming that $\mathsf{Adv}^1_{\mathcal{A}}(\lambda)$ (defined in Equation (29)) is negligible in κ for all non-uniform adversary $\mathcal{A}$ such that $\mathrm{Size}(\mathcal{A}) = \mathrm{poly}(\kappa)$, we have $\mathsf{Adv}^n_{\mathcal{B}}(\lambda) = \mathsf{negl}(\lambda)$ for all non-uniform adversary $\mathcal{B}$ such that $\mathrm{Size}(\mathcal{B}) = \mathrm{poly}(\lambda)$.

Proof. For the sake of contradiction, suppose that there exists an adversary $\mathcal{B} = \{\mathcal{B}_\lambda\}_\lambda$ such that $\epsilon_n(\lambda) \stackrel{\text{def}}{=} \mathsf{Adv}^n_{\mathcal{B}}(\lambda)$ is non-negligible and $t_n(\lambda) \stackrel{\text{def}}{=} \mathrm{Size}(\mathcal{B})$ is polynomial in λ. In particular, this implies that there exists an infinite

[13] Equation (31) includes additional terms $\{\mathsf{F}_{h^*-1}^{j_{h^*-1}}\}_{j_{h^*-1}}$ while Equation (29) 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 3.11 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. □

The following lemma completes Step 2 of the proof of Theorem 3.9.

Lemma 3.13. If SKE is INDr secure, Equation (24) implies Equation (22).

Proof. We first observe that Equation (24) 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 3.12, 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

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

By INDr security of SKE, we have

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

Combining the above equations, Equation (22) readily follows. □

We conclude the proof of Theorem 3.9. □

Remark 3.14 (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

$$\mathrm{poly}_1(\lambda_1)\mathrm{poly}_2(\lambda_2)\cdots\mathrm{poly}_n(\lambda_n) = \mathrm{poly}(\lambda_1\cdots\lambda_n) = \mathrm{poly}(2^{n^2}\lambda^n) \tag{33}$$

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 3.11 and 3.12). 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 (33) 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, 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.

[14] Namely, the actual value of ϵ_{n-1} may be even smaller.

3.4 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 3.14 for more discussions.

Theorem 3.15. Let $\mathcal{SC}_{\mathsf{prMIFE}}$ be a samples class for prMIFE. Suppose prFE_i scheme satisfies prCT security as per Definition 2.12 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 (30), SKE satisfies INDr security and PRF_i is secure, then prMIFE constructed in Section 3.2 for constant n satisfies security for $\kappa = \lambda^n$ as in Definition 3.4. Note that this in particular implies the κ-security defined in Definition 3.2.

Proof. The proof of this theorem largely follows that of Theorem 3.9. The crucial difference is that to get the equivalent of Lemma 3.12, 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 3.9 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 (23) implies Equation (24). This is proven in the same manner as Lemma 3.10. 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 (24) implies Equation (22).
  - We first observe that the uniform version of Lemma 3.11 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 3.12, which asserts that if $\mathsf{Adv}^1_{\mathcal{A}}(\lambda)$ (defined in Equation (29)) 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 (29) for $h = h^* - 1$ implies that for h^* by the analogue of Lemma 3.11 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 3.13.

This completes the proof of Theorem 3.15. □

We encapsulate the results of this section using the following theorems.

Theorem 3.16 (prMIFE **for poly arity).** Let $\kappa = \lambda^{n^2 \log \lambda}$. Assume non-uniform κ-evasive LWE (Assumption 2.4), subexponentially secure PRF against non-uniform adversary, and non-uniform sub-exponential LWE (Assumption A.5). 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 κ-security (Definition 3.2) with efficiency

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

where $\Lambda := (n^2\lambda)^{1/\delta}$.

Theorem 3.17 (prMIFE **for constant arity).** Let $\kappa = \lambda^n$. Assume evasive LWE (Assumption 2.1) and LWE (Assumption A.5). Then there exists a prMIFE scheme for arity $n = O(1)$, supporting functions with input length L and bounded polynomial depth $d = d(\lambda)$, and satisfying κ-security (Definition 3.2) with efficiency

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

4 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 3, we consider two settings where the arity is either constant or arbitrary polynomial. The former setting leads to a construction with weaker assumption where we only require (plain) evasive LWE and LWE. This improves the construction by [ARYY23], which additionally requires non-standard tensor LWE assumption. The latter setting leads to a construction with polynomial arity and for all circuits under the stronger non-uniform κ-evasive LWE assumption. This resolves the open problem posed by [ARYY23].

4.1 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 INDr security (Definition 2.19). 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 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-input FE for pseudorandom functionalities $\mathsf{prMIFE} = \mathsf{prMIFE}.(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}_1, \ldots, \mathsf{Enc}_n, \mathsf{Dec})$ 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 $L' = L + \lambda + 1$ 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 $\mathsf{prm}_{\mathsf{prMIFE}}$.

3. A pseudorandom function $\mathsf{PRF} : \{0,1\}^{\Lambda} \times \{0,1\}^{(n-1)\Lambda} \to \{0,1\}^{\mathsf{R}_{\mathsf{len}}}$, where $\{0,1\}^{\Lambda}$ and $\{0,1\}^{(n-1)\Lambda}$ are the key space and input space respectively and $\mathsf{R}_{\mathsf{len}}$ is the length of randomness used in the PE.Enc algorithm. We will discuss how to set Λ in Remark 4.1.

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 $(\mathsf{PE.mpk}, \mathsf{PE.msk}) \leftarrow \mathsf{PE.Setup}(1^{\Lambda}, \mathsf{prm}_{\mathsf{PE}})$.
- Output $\mathsf{mpk} = (\mathsf{prMIFE.mpk}, \mathsf{PE.mpk})$ 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 $\mathsf{mpk} = (\mathsf{prMIFE.mpk}, \mathsf{PE.mpk})$ and $\mathsf{msk} = (\mathsf{prMIFE.msk}, \mathsf{PE.msk})$.
- Compute $\mathsf{PE.sk}_f \leftarrow \mathsf{PE.KeyGen}(\mathsf{PE.msk}, f)$.

– 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:

$$\mathrm{F}[\mathsf{PE.mpk}]\left((\mathbf{x}_1, \mu_1, \mathbf{r}_1), \ldots, (\mathbf{x}_{(n-1)}, \mu_{(n-1)}, \mathbf{r}_{(n-1)}), (\mathbf{x}_n, \mu_n, \mathbf{r}_n)\right)$$
$$= \mathsf{PE.Enc}\left(\mathsf{PE.mpk}, (\mathbf{x}_1, \ldots, \mathbf{x}_n), (\mu_1, \ldots, \mu_n); \mathsf{PRF}(\mathbf{r}_n, (\mathbf{r}_1, \ldots, \mathbf{r}_{(n-1)}))\right)$$

– Output $\mathsf{sk}_f = (\mathsf{PE.sk}_f, \mathsf{prMIFE.sk_F})$[15].

$\mathsf{Enc}_i(\mathsf{msk}, \mathbf{x}_i, \mu_i) \to \mathsf{ct}_i$ for $1 \le i \le n$.The slot i encryption algorithm does the following.

– Parse $\mathsf{msk} = (\mathsf{prMIFE.msk}, \mathsf{PE.msk})$.
– For $1 \le i \le n$, sample $\mathbf{r}_i \leftarrow \{0,1\}^\Lambda$ and compute $\mathsf{prMIFE.ct}_i \leftarrow \mathsf{prMIFE.Enc}_i(\mathsf{prMIFE.msk}, (\mathbf{x}_i, \mu_i, \mathbf{r}_i))$.
– 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 $\mathsf{mpk} = (\mathsf{prMIFE.mpk}, \mathsf{PE.mpk})$, $\mathsf{sk}_f = (\mathsf{PE.sk}_f, \mathsf{prMIFE.sk_F})$ and $\{\mathsf{ct}_i = \mathsf{prMIFE.ct}_i\}_{i \in [n]}$.
– Compute $\mathsf{PE.ct} = \mathsf{prMIFE.Dec}(\mathsf{prMIFE.mpk}, \mathsf{prMIFE.sk_F}, \mathrm{F}, \mathsf{prMIFE.ct}_1, \ldots, \mathsf{prMIFE.ct}_n)$.
– Compute $y = \mathsf{PE.Dec}(\mathsf{PE.mpk}, \mathsf{PE.sk}_f, f, \mathsf{PE.ct})$.
– Output y.

Remark 4.1. Here, we discuss how we set Λ. Similarly to the case of prMIFE in Section 3, 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 3.6 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 4.2. 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 F, we have

$$\mathsf{prMIFE.Dec}(\mathsf{prMIFE.mpk}, \mathsf{prMIFE.sk_F}, \mathrm{F}, \mathsf{prMIFE.ct}_1, \ldots, \mathsf{prMIFE.ct}_n) = \mathsf{PE.ct}$$

with probability 1, where

$$\mathsf{PE.ct} = \mathsf{PE.Enc}\left(\mathsf{PE.mpk}, (\mathbf{x}_1, \ldots, \mathbf{x}_n), (\mu_1, \ldots, \mu_n); \mathsf{PRF}(\mathbf{r}_n, (\mathbf{r}_1, \ldots, \mathbf{r}_{(n-1)}))\right).$$

Next, using the security of PRF, with all but negl advantage, $\mathsf{PRF}(\mathbf{r}_n, (\mathbf{r}_1, \ldots, \mathbf{r}_{(n-1)}))$ is indistinguishable from $R \leftarrow \{0,1\}^{\mathsf{R_{len}}}$. Since

$$\mathsf{PE.Dec}(\mathsf{PE.mpk}, \mathsf{PE.sk}_f, f, \mathsf{PE.Enc}\,(\mathsf{PE.mpk}, (\mathbf{x}_1, \ldots, \mathbf{x}_n), (\mu_1, \ldots, \mu_n); R)) = (\mu_1, \ldots, \mu_n)$$

holds for randomly chosen R with all but negl probability if $f(\mathbf{x}_1, \ldots, \mathbf{x}_n) = 1$ by the correctness of PE scheme, the above holds even for $R = \mathsf{PRF}(\mathbf{r}_n, (\mathbf{r}_1, \ldots, \mathbf{r}_{(n-1)}))$ with all but negl probability. Hence, the correctness. □

[15]Note that one can compute $\mathsf{prMIFE.sk_F}$ in the Setup algorithm and output it as a part of public key rather than the secret key as it does not require the knowledge of f. We output this here for notational convinience.

4.2 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 4.3. Assume prMIFE scheme is secure (as per Definition 3.2) with respect to $\kappa = \lambda^{n^2 \log \lambda}$ and the sampler class containing the sampler $\mathsf{Samp}_{\mathsf{prMIFE}}$ as defined in Eq. 35, PE scheme is sub-exponentially secure (Definition 2.19) and PRF is sub-exponentially secure. Then the construction of miPE is secure as per Definition 2.20.

Proof. Suppose the 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 $(\mathbf{x}_i^{j_i}, \mu_i^{j_i})$ 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 as per Definition 2.20 we first define $\{\mathsf{Sim}_i\}_{i \in [n]}$ as follows.

$\mathsf{Sim}_i(\mathsf{msk}) \to \mathsf{ct}_i$ for $i \in [n]$. Parse $\mathsf{msk} = (\mathsf{prMIFE.msk}, \mathsf{PE.msk})$. Set $\mathsf{Sim}_i(\mathsf{msk}) = \mathsf{prMIFE.Sim}_i(\mathsf{prMIFE.msk})$, where $\{\mathsf{prMIFE.Sim}_i\}_{i \in [n]}$ is the simulator whose existence is guaranteed by the security of prMIFE (See Definition 3.2).

Then, it suffices to show

$$\begin{pmatrix} \mathsf{aux}_{\mathcal{A}},\ \mathsf{mpk} = (\mathsf{prMIFE.mpk}, \mathsf{PE.mpk}), \\ \left\{\mathrm{F}[\mathsf{PE.mpk}], f_k, \mathsf{sk}_{f_k} = (\mathsf{PE.sk}_{f_k}, \mathsf{prMIFE.sk}_{\mathrm{F}})\right\}_{k \in [q_0]}, \\ \left\{\mathsf{ct}_i^{j_i} = \mathsf{prMIFE.ct}_i^{j_i}\right\}_{i \in [n], j_i \in [q_i]} \end{pmatrix} \approx_c \begin{pmatrix} \mathsf{aux}_{\mathcal{A}},\ \mathsf{mpk} = (\mathsf{prMIFE.mpk}, \mathsf{PE.mpk}), \\ \left\{\mathrm{F}[\mathsf{PE.mpk}], f_k, \mathsf{sk}_{f_k} = (\mathsf{PE.sk}_{f_k}, \mathsf{prMIFE.sk}_{\mathrm{F}})\right\}_{k \in [q_0]}, \\ \left\{\delta_i^{j_i} \leftarrow \mathsf{prMIFE.Sim}_i(\mathsf{msk})\right\}_{i \in [n], j_i \in [q_i]} \end{pmatrix} \tag{34}$$

where

$$\begin{aligned} &(\mathsf{aux}_{\mathcal{A}}, \{f_k\}_k, \{(\mathbf{x}_i^{j_i}, \mu_i^{j_i})\}_{i,j_i}) \leftarrow \mathcal{A}(1^\lambda), \\ &(\mathsf{mpk} = (\mathsf{prMIFE.mpk}, \mathsf{PE.mpk}),\ \mathsf{msk} = (\mathsf{prMIFE.msk}, \mathsf{PE.msk})) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, \mathsf{prm}), \\ &\mathsf{prMIFE.sk}_{\mathrm{F}} \leftarrow \mathsf{prMIFE.KeyGen}(\mathsf{prMIFE.msk}, \mathrm{F}[\mathsf{PE.mpk}]), \\ &\mathsf{PE.sk}_{f_k} \leftarrow \mathsf{PE.KeyGen}(\mathsf{PE.msk}, f_k) \text{ for } k \in [q_0], \\ &\mathsf{prMIFE.ct}_i^{j_i} \leftarrow \mathsf{prMIFE.Enc}_i(\mathsf{prMIFE.msk}, (\mathbf{x}_i^{j_i}, \mu_i^{j_i}, \mathbf{r}_i^{j_i})),\ \mathbf{r}_i^{j_i} \leftarrow \{0,1\}^{\Lambda} \text{ for } i \in [n], j_i \in [q_i]. \end{aligned}$$

We invoke the security of prMIFE with sampler $\mathsf{Samp}_{\mathsf{prMIFE}}$ that outputs

$$\begin{pmatrix} \text{Function:} & \mathrm{F}[\mathsf{PE.mpk}], \\ \text{Inputs:} & \left\{\left((\mathbf{x}_1^{j_1}, \mu_1^{j_1}, \mathbf{r}_1^{j_1}), \ldots, (\mathbf{x}_n^{j_n}, \mu_n^{j_n}, \mathbf{r}_n^{j_n})\right)\right\}_{i \in [n], j_i \in [q_i]}, \\ \text{Auxiliary Information:} & \mathsf{aux} = \left(\mathsf{aux}_{\mathcal{A}}, \mathsf{PE.mpk}, \{f_k, \mathsf{PE.sk}_{f_k}\}_{k \in [q_0]}\right) \end{pmatrix} \tag{35}$$

From the security guarantee of the prMIFE scheme with sampler $\mathsf{Samp}_{\mathsf{prMIFE}}$ and simulator $\{\mathsf{prMIFE.Sim}_i\}_{i \in [n]}$, we have that

$$\begin{aligned} &\begin{pmatrix} \mathsf{aux},\ \mathsf{prMIFE.mpk},\ \mathrm{F}[\mathsf{PE.mpk}], \mathsf{prMIFE.sk}_{\mathrm{F}}, \\ \{\mathsf{prMIFE.ct}_i^{j_i} \leftarrow \mathsf{prMIFE.Enc}_i(\mathsf{prMIFE.msk}, (\mathbf{x}_i^{j_i}, \mu_i^{j_i}, \mathbf{r}_i^{j_i}))\}_{i \in [n], j_i \in [q_i]} \end{pmatrix} \\ \approx_c &\begin{pmatrix} \mathsf{aux},\ \mathsf{prMIFE.mpk},\ \mathrm{F}[\mathsf{PE.mpk}], \mathsf{prMIFE.sk}_{\mathrm{F}}, \\ \{\mathsf{prMIFE.ct}_i^{j_i} \leftarrow \mathsf{prMIFE.Sim}_i(\mathsf{prMIFE.msk})\}_{i \in [n], j_i \in [q_i]} \end{pmatrix} \end{aligned}$$

if $\left(1^\kappa,\ \mathsf{aux},\ \{\mathsf{F}[\mathsf{PE.mpk}], \mathsf{PE.ct}_{j_1,\ldots,j_n}\}_{i\in[n],j_i\in[q_i]}\right) \approx_c \left(1^\kappa,\ \mathsf{aux},\ \{\mathsf{F}[\mathsf{PE.mpk}], \mathsf{PE.}\Delta_{j_1,\ldots,j_n}\}_{i\in[n],j_i\in[q_i]}\right)$

where $\mathsf{PE.}\Delta_{j_1,\ldots,j_n} \leftarrow \mathcal{CT}_{\mathsf{PE}}$ for $i \in [n], j_i \in [q_i]$ and

$$\begin{aligned}\mathsf{PE.ct}_{j_1,\ldots,j_n} =& \mathsf{F}[\mathsf{PE.mpk}]\left(\left(\mathbf{x}_1^{j_1}, \mu_1^{j_1}, \mathbf{r}_1^{j_1}\right), \ldots, \left(\mathbf{x}_{(n-1)}^{j_{(n-1)}}, \mu_{(n-1)}^{j_{(n-1)}}, \mathbf{r}_{(n-1)}^{j_{(n-1)}}\right), \left(\mathbf{x}_n^{j_n}, \mu_n^{j_n}, \mathbf{r}_n^{j_n}\right)\right)\\ =& \mathsf{PE.Enc}\left(\mathsf{PE.mpk}, (\mathbf{x}_1^{j_1}, \ldots, \mathbf{x}_n^{j_n}), (\mu_1^{j_1}, \ldots, \mu_n^{j_n}); \mathsf{PRF}\left(\mathbf{r}_n^{j_n}, (\mathbf{r}_1^{j_1}, \ldots, \mathbf{r}_{(n-1)}^{j_{(n-1)}})\right)\right)\end{aligned}$$

and $\mathsf{PE.}\Delta_{j_1,\ldots,j_n} \leftarrow \mathcal{CT}_{\mathsf{PE}}$ for $i \in [n], j_i \in [q_i]$.

Thus to prove Equation (34), it suffices to show

$$\begin{pmatrix} 1^\kappa,\ \mathsf{aux}_{\mathcal{A}},\ \mathsf{F}[\mathsf{PE.mpk}],\ \mathsf{PE.mpk},\ \{f_k, \mathsf{PE.sk}_{f_k}\}_{k\in[q_0]}, \\ \left\{\mathsf{PE.ct}_{j_1,\ldots,j_n} = \mathsf{PE.Enc}\left(\mathsf{PE.mpk}, (\mathbf{x}_1^{j_1}, \ldots, \mathbf{x}_n^{j_n}), (\mu_1^{j_1}, \ldots, \mu_n^{j_n}); \mathsf{PRF}(\mathbf{r}_n^{j_n}, (\mathbf{r}_1^{j_1}, \ldots, \mathbf{r}_{(n-1)}^{j_{(n-1)}}))\right)\right\}_{i\in[n],j_i\in[q_i]} \end{pmatrix} \tag{36}$$

$$\approx_c \begin{pmatrix} 1^\kappa,\ \mathsf{aux}_{\mathcal{A}},\ \mathsf{F}[\mathsf{PE.mpk}],\ \mathsf{PE.mpk},\ \{f_k, \mathsf{PE.sk}_{f_k}\}_{k\in[q_0]}, \\ \left\{\mathsf{PE.}\Delta_{j_1,\ldots,j_n} \leftarrow \mathcal{CT}_{\mathsf{PE}}\right\}_{i\in[n],j_i\in[q_i]} \end{pmatrix}$$

We prove the above via the following sequence of hybrids.

Hyb_1. This is Equation (36).

$$\begin{pmatrix} 1^\kappa,\ \mathsf{aux}_{\mathcal{A}},\ \mathsf{F}[\mathsf{PE.mpk}],\ \mathsf{PE.mpk},\ \{f_k, \mathsf{PE.sk}_{f_k}\}_{k\in[q_0]}, \\ \left\{\mathsf{PE.ct}_{j_1,\ldots,j_n} = \mathsf{PE.Enc}\left(\mathsf{PE.mpk}, (\mathbf{x}_1^{j_1}, \ldots, \mathbf{x}_n^{j_n}), (\mu_1^{j_1}, \ldots, \mu_n^{j_n}); \mathsf{PRF}(\mathbf{r}_n^{j_n}, (\mathbf{r}_1^{j_1}, \ldots, \mathbf{r}_{(n-1)}^{j_{(n-1)}}))\right)\right\}_{i\in[n],j_i\in[q_i]} \end{pmatrix}$$

Hyb_2. This hybrid is the same as the previous one except that we replace $\mathsf{PRF}(\mathbf{r}_n^{j_n}, \cdot)$ with the real random function $\mathsf{R}^{j_n}(\cdot)$ for each $j_n \in [q_n]$. Since $\mathbf{r}_n^{j_n}$ 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_4. This hybrid is the same as the previous one except that we output a failure symbol when there exist $(j_1, \ldots, j_{n-1}) \neq (j_1', \ldots, j_{n-1}')$ such that $(\mathbf{r}_1^{j_1}, \ldots, \mathbf{r}_{n-1}^{j_{n-1}}) = (\mathbf{r}_1^{j_1'}, \ldots, \mathbf{r}_{n-1}^{j_{n-1}'})$. We show that the probability of this happening is negligible in κ. To prove this, it suffices to show that there are no $i \in [n]$, $j, j' \in [q_i]$ satisfying $j \neq j'$ and $\mathbf{r}_i^j = \mathbf{r}_i^{j'}$. The probability of this happening can be bounded by $(q_1^2 + \cdots q_{n-1}^2)/2^{2\Lambda}$ by taking the union bound with respect to all the combinations of i, j, j'. By our choice that $2^{\Lambda^\delta} > \kappa^{\omega(1)}$, this probability is bounded by $\mathsf{negl}(\kappa)$.

Hyb_5. This hybrid is the same as the previous one except that we compute

$$\mathsf{PE.ct}_{j_1,\ldots,j_n} \leftarrow \mathsf{PE.Enc}\left(\mathsf{PE.mpk}, (\mathbf{x}_1^{j_1}, \ldots, \mathbf{x}_n^{j_n}), (\mu_1^{j_1}, \ldots, \mu_n^{j_n})\right).$$

Namely, we use fresh randomness for each encryption instead of deriving the randomness from $\mathsf{R}^{j_n}(\cdot)$. This change is only conceptual.
In this hybrid, the view of the adversary is

$$\begin{pmatrix} 1^\kappa,\ \mathsf{aux}_{\mathcal{A}},\ \mathsf{F}[\mathsf{PE.mpk}],\ \mathsf{PE.mpk},\ \{f_k, \mathsf{PE.sk}_{f_k}\}_{k\in[q_0]}, \\ \left\{\mathsf{PE.ct}_{j_1,\ldots,j_n} \leftarrow \mathsf{PE.Enc}\left(\mathsf{PE.mpk}, (\mathbf{x}_1^{j_1}, \ldots, \mathbf{x}_n^{j_n}), (\mu_1^{j_1}, \ldots, \mu_n^{j_n})\right)\right\}_{i\in[n],j_i\in[q_i]} \end{pmatrix}$$

Hyb_6. This hybrid is the same as the previous one except that we use sub-exponential security of underlying PE scheme to replace $\mathsf{PE.Enc}(\mathsf{PE.mpk}, (\mathbf{x}_1^{j_1}, \ldots, \mathbf{x}_n^{j_n}), (\mu_1^{j_1}, \ldots, \mu_n^{j_n}))$ with $\mathsf{PE}.\delta_{j_1,\ldots,j_n} \leftarrow \mathcal{CT}_{\mathsf{PE}}$ for all $i \in [n]$ and $j_i \in [q_i]$. By admissibility of $\mathcal{A}$ in the miPE security, we have $f(\mathbf{x}_1^{j_1}, \ldots, \mathbf{x}_n^{j_n}) = 0$ which satisfies the admissibility of the single input PE security game. We therefore replace each of $\mathsf{PE.ct}_{j_1,\ldots,j_n}$ with a random string by the PE security. By our choice of the parameter $2^{\Lambda^\delta} > \kappa^{\omega(1)}$ and subexponential security of PE, the distinguishing advantage of the adversary when we replace single ciphertext is $\mathsf{negl}(\kappa)$. Since there are $q_1 \cdots q_n = \mathrm{poly}(\kappa)$ ciphertexts, we can conclude that the distinguishing advantage between this hybrid from the previous one is at most $\mathrm{poly}(\kappa)\mathsf{negl}(\kappa) = \mathsf{negl}(\kappa)$.
In this hybrid, the view of the adversary is

$$\begin{pmatrix} 1^\kappa, \ \mathsf{aux}_\mathcal{A}, \ \mathsf{F}[\mathsf{PE.mpk}], \ \mathsf{PE.mpk}, \ \{f_k, \mathsf{PE.sk}_{f_k}\}_{k \in [q_0]}, \\ \{\mathsf{PE}.\delta_{j_1,\ldots,j_n} \leftarrow \mathcal{CT}_{\mathsf{PE}}\}_{i \in [n], j_i \in [q_i]} \end{pmatrix}$$

which is the distribution on RHS in Equation (36).

Hence, the proof. □

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 4.4. Assume prMIFE scheme is secure (as per Definition 3.2) with respect to $\kappa = \lambda^n$ and the sampler class containing the sampler $\mathsf{Samp}_{\mathsf{prMIFE}}$ as defined in Eq. 35, PE scheme is secure (Definition 2.19) and PRF is secure. Then the construction of miPE is secure as per Definition 2.20.

The proof of the above theorem is exactly the same as Theorem 4.3 except that here $\kappa = \lambda^n$. Since $\kappa = \mathrm{poly}(\lambda)$, we do not need subexponential security for PRF and PE. In addition, since prMIFE for constant arity with $\kappa = \lambda^n$ can be constructed from (plain) evasive LWE, which is weaker assumption than non-uniform κ-version of it that is necessary for the general case.

We encapsulate the results of this section using the following theorems.

Theorem 4.5 (miPE **for poly arity).** Let $\kappa = \lambda^{n^2 \log \lambda}$. Assuming non-uniform κ-evasive LWE (Assumption 2.4), subexponentially secure PRF against non-uniform adversary, and non-uniform sub-exponential LWE (Assumption A.5), there exists a miPE scheme for arity $n = \mathrm{poly}(\lambda)$, supporting functions of bounded polynomial depth $\mathsf{dep} = \mathsf{dep}(\lambda)$, and satisfying security as per Definition 2.20.

Theorem 4.6 (miPE **for constant arity).** Assuming evasive LWE (Assumption 2.1) and LWE (Assumption A.5), there exists a prMIFE scheme for arity $n = O(1)$, supporting functions of bounded polynomial depth $d = d(\lambda)$, and satisfying security as per Definition 2.20.

5 Indistinguishability Obfuscation for Pseudorandom Functionalities

5.1 Definition

In this section we give the definitions for indistinguishability obfuscation for pseudorandom functionalities (prIO) for circuits.

Syntax. An indistinguishability obfuscator for pseudorandom functionalities consists of the following algorithms.

$\mathsf{iO}(1^\lambda, C) \rightarrow \tilde{C}$**.** The obfuscation algorithm takes as input the security parameter λ and a circuit $C : \{0,1\}^n \rightarrow \{0,1\}^m$ with arbitrary n and m. It outputs an obfuscated circuit $\tilde{C}$.

$\mathsf{Eval}(\tilde{C}, x) \rightarrow y$. 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 5.1 (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 5.2 (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 5.3 (Indistinguishability for Pseudorandom Functionality). For the security parameter $\lambda = \lambda(\lambda)$, let Samp be a PPT algorithm that on input 1^λ, outputs

$$(\mathsf{C}_0, \mathsf{C}_1,\ \mathsf{aux} \in \{0,1\}^*)$$

where $\mathsf{C}_0 : \{0,1\}^n \to \{0,1\}^m$ and $\mathsf{C}_1 : \{0,1\}^n \to \{0,1\}^m$ have the same description size. We say that a prIO scheme is secure with respect to the sampler class $\mathcal{SC}$ if for every PPT sampler Samp $\in \mathcal{SC}$ the following holds.

$$\text{If } \left(1^\kappa, \{\mathsf{C}_0(x)\}_{x\in\{0,1\}^n}, \mathsf{aux}\right) \approx_c \left(1^\kappa, \{\Delta_x \leftarrow \{0,1\}^m\}_{x\in\{0,1\}^n}, \mathsf{aux}\right) \approx_c \left(1^\kappa, \{\mathsf{C}_1(x)\}_{x\in\{0,1\}^n}, \mathsf{aux}\right)$$
$$\text{then } \left(\mathsf{iO}(1^\lambda, \mathsf{C}_0), \mathsf{aux}\right) \approx_c \left(\mathsf{iO}(1^\lambda, \mathsf{C}_1), \mathsf{aux}\right)$$

where $\kappa \geq 2^n$.

Remark 5.4. 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 5.5. It is shown in [AMYY25, BDJ+24] that there is no prIO scheme satisfying the above security for all general samplers. Therefore, when we use the security of prIO, we invoke the security *with respect to a specific sampler class* that is induced by the respective applications.

5.2 Construction

Our construction follows the blueprint of the multi-input FE to iO conversion by Ananth and Jain [AJ15]. 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-security defined as per Definition 3.2. We can instantiate the scheme by our construction in Section 3.2 with $\kappa = \lambda^{n^2 \log \lambda}$.

Next, we describe the construction of prIO 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 $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{prMIFE}.\mathsf{Setup}(1^\lambda, 1^{n+1}, \mathsf{prm})$, 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 $\mathsf{ct}_{n+1} \leftarrow \mathsf{prMIFE}.\mathsf{Enc}_{n+1}(\mathsf{msk}, C)$.
- For $i \in [n]$ and $b \in \{0,1\}$, compute $\mathsf{ct}_{i,b} = \mathsf{prMIFE}.\mathsf{Enc}_i(\mathsf{msk}, b)$.

– 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 $\mathsf{prMIFE.Dec}(\mathsf{mpk}, \mathsf{sk}_U, U, \mathsf{ct}_{x_1}, \ldots, \mathsf{ct}_{x_n}, \mathsf{ct}_{n+1})$.

Remark 5.6. 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.

5.3 Security

Theorem 5.7. Let $\mathcal{SC}_{\mathsf{prIO}}$ be a sampler class for prIO. Suppose prMIFE scheme is secure (Definition 3.2) for $\kappa = \kappa(\lambda)$ w.r.t sampler class that contains all $\mathsf{Samp}_{\mathsf{prMIFE},0}$ and $\mathsf{Samp}_{\mathsf{prMIFE},1}$, induced by $\mathsf{Samp}_{\mathsf{prIO}} \in \mathcal{SC}_{\mathsf{prIO}}$, as defined in Equation (39). Then the prIO scheme satisfies security as defined in Definition 5.3 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 as an obfuscation query.
2. **Auxiliary Information.** It outputs the auxiliary information $\mathsf{aux}_{\mathcal{A}}$.

To prove the security as per Definition 5.3, we show that

$$\left(\{\mathsf{ct}_{i,b}\}_{i\in[n],b\in\{0,1\}}, \mathsf{ct}^0_{n+1}, \mathsf{sk}_U, \mathsf{mpk}, \mathsf{aux}_{\mathcal{A}}\right) \approx_c \left(\{\mathsf{ct}_{i,b}\}_{i\in[n],b\in\{0,1\}}, \mathsf{ct}^1_{n+1}, \mathsf{sk}_U, \mathsf{mpk}, \mathsf{aux}_{\mathcal{A}}\right)$$

$$\text{if } \left(1^\kappa, \{C_0(x_i)\}_{\forall\, x_i\in\mathcal{X}_\lambda}, \mathsf{aux}_{\mathcal{A}}\right) \approx_c \left(1^\kappa, \{\Delta_i \leftarrow \mathcal{Y}_\lambda\}_i, \mathsf{aux}_{\mathcal{A}}\right) \approx_c \left(1^\kappa, \{C_1(x_i)\}_{\forall\, x_i\in\mathcal{X}_\lambda}, \mathsf{aux}_{\mathcal{A}}\right) \tag{37}$$

where

$$\begin{aligned}
&(C_0, C_1, \mathsf{aux}_{\mathcal{A}}) \leftarrow \mathsf{Samp}_{\mathsf{prIO}}(1^\lambda),\\
&(\mathsf{msk}, \mathsf{mpk}) \leftarrow \mathsf{prMIFE.Setup}(1^\lambda, 1^{n+1}, \mathsf{prm}),\\
&\mathsf{sk}_U \leftarrow \mathsf{prMIFE.KeyGen}(\mathsf{msk}, U),\\
&\mathsf{ct}^0_{n+1} \leftarrow \mathsf{prMIFE.Enc}_{n+1}(\mathsf{msk}, C_0), \mathsf{ct}^1_{n+1} \leftarrow \mathsf{prMIFE.Enc}_{n+1}(\mathsf{msk}, C_1),\\
&\mathsf{ct}_{i,b} = \mathsf{prMIFE.Enc}_i(\mathsf{msk}, b), \text{ for } i \in [n], b \in \{0,1\}.
\end{aligned}$$

From the right hand indistinguishability of Equation (37), we have that

$$\begin{aligned}
&\left(\{\mathsf{ct}_{i,b}\}_{i\in[n],b\in\{0,1\}}, \mathsf{ct}^0_{n+1}, \mathsf{sk}_U, \mathsf{mpk}, \mathsf{aux}_{\mathcal{A}}\right)\\
\approx_c &\left(\{\gamma_{i,b} \leftarrow \mathsf{prMIFE.Sim}_i(\mathsf{msk})\}_{i\in[n],b\in\{0,1\}}, \gamma_{n+1} \leftarrow \mathsf{prMIFE.Sim}_{n+1}(\mathsf{msk}), \mathsf{sk}_U, \mathsf{mpk}, \mathsf{aux}_{\mathcal{A}}\right)
\end{aligned} \tag{38}$$

using the prMIFE security with simulator $\{\mathsf{prMIFE.Sim}_i\}_{i\in[n+1]}$ and sampler $\mathsf{Samp}_{\mathsf{prMIFE},0}$ that outputs

$$\begin{pmatrix}
\text{Function:} & \text{Universal Circuit } U\\
\text{Inputs:} & \{x_1^{j_1} = j_1, \ldots, x_n^{j_n} = j_n\}_{j_1\in\{0,1\},\ldots,j_n\in\{0,1\}}, x_{n+1} = C_0\\
\text{Auxiliary Information:} & \mathsf{aux}_{\mathcal{A}}
\end{pmatrix} \tag{39}$$

To see this we note that the pseudorandomness of $U(x_1 \in \{0,1\} \ldots, x_n \in \{0,1\}, x_{n+1} = C_{n+1} \in \{0,1\}^L) = C_0(x_1, \ldots, x_n)$ is implied from Equation (37).
Similarly, from the right hand indistinguishability of Equation (37), we have

$$\begin{aligned}&\left(\{\mathsf{ct}_{i,b}\}_{i\in[n],b\in\{0,1\}}, \mathsf{ct}^1_{n+1}, \mathsf{sk}_U, \mathsf{mpk}, \mathsf{aux}_\mathcal{A}\right) \\ \approx_c &\left(\{\gamma_{i,b} \leftarrow \mathsf{prMIFE.Sim}_i(\mathsf{msk})\}_{i\in[n],b\in\{0,1\}}, \gamma_{n+1} \leftarrow \mathsf{prMIFE.Sim}_{n+1}(\mathsf{msk}), \mathsf{sk}_U, \mathsf{mpk}, \mathsf{aux}_\mathcal{A}\right)\end{aligned} \tag{40}$$

using the prMIFE security with simulator $\{\mathsf{prMIFE.Sim}_i\}_{i\in[0,n]}$ and sampler $\mathsf{Samp}_{\mathsf{prMIFE},1}$ whose output is same as that of Samp_0 except that $x_{n+1} = C_1$.
From Equation (38) and Equation (40), we have

$$\begin{aligned}&\left(\{\mathsf{ct}_{i,b}\}_{i\in[n],b\in\{0,1\}}, \mathsf{ct}^0_{n+1}, \mathsf{sk}_U, \mathsf{mpk}, \mathsf{aux}_\mathcal{A}\right) \\ \approx_c &\left(\{\mathsf{ct}_{i,b}\}_{i\in[n],b\in\{0,1\}}, \mathsf{ct}^1_{n+1}, \mathsf{sk}_U, \mathsf{mpk}, \mathsf{aux}_\mathcal{A}\right)\end{aligned}$$

hence the proof. □

The following theorem holds for the specific class of samplers we described in the proof.

Theorem 5.8. Let $\kappa = \lambda^{n^2 \log \lambda}$. Assuming non-uniform κ-evasive LWE (Assumption 2.4), subexponentially secure PRF against non-uniform adversary, and non-uniform sub-exponential LWE (Assumption A.5), there exists a prIO scheme for circuits with input domain $\{0,1\}^n$ satisfying security as per Definition 5.3.

6 Polynomial Domain IO for Pseudorandom Functionalities

In this section, we define and construct indistinguishability obfuscation for pseudorandom functionalities with polynomial size domain (pPRIO). The advantage of considering this restricted variant is that we can base the security of the construction on (plain) evasive LWE, rather than non-uniform κ-variant of it. Here, we introduce online-offline property and reusable security and provide a construction satisfying these properties. The reason why we introduce these properties is that they seem to be useful for some applications. Indeed, in our companion paper [AKY24b], we use pPRIO with these properties to construct ABE with optimized parameter size. We consider two variants of pPRIO: pPRIO with fixed input domain and pPRIO with flexible domain. As the name suggests, the latter is more flexible and desirable. To obtain the latter, we first construct the former from prMIFE in Section 6.2 and then convert it into the latter in Section 6.3.

6.1 Definition

Definition 6.1 (Syntax). A pPRIO scheme consists of the following algorithms.

$\mathsf{Obf}(1^\lambda, C) \to \mathsf{obf}$. The obfuscation algorithm takes as input the security parameter λ and a circuit $C : [N] \to [M]$ with $\mathsf{size}(C) \le L$ for some arbitrary polynomial $L = L(\lambda)$. It outputs an obfuscation of the circuit obf.
We consider a definition where the Obf algorithm can be decomposed into the following two phases.

- $\mathsf{ObfOff}(1^\lambda, 1^L) \to (\mathsf{obf}_{\mathsf{off}}, \mathsf{st})$. The offline obfuscation algorithm takes as input the security parameter λ and the circuit size bound L. It outputs $\mathsf{obf}_{\mathsf{off}}$ and st.
- $\mathsf{ObfOn}(\mathsf{st}, C) \to \mathsf{obf}_{\mathsf{on}}$. The online obfuscation algorithm takes as input the security parameter st and the circuit C and outputs $\mathsf{obf}_{\mathsf{on}}$.

The final output of Obf is $\mathsf{obf} = (\mathsf{obf}_{\mathsf{off}}, \mathsf{obf}_{\mathsf{on}})$.

$\mathsf{Eval}(\mathsf{obf}, x) \to y$. The evaluation algorithm takes as input an obfuscated circuit obf and an input $x \in [N]$. It outputs $y \in [M]$.

We consider the following syntax variants of a pPRIO scheme. We consider fixed input domain pPRIO and flexible input domain pPRIO. In the former, we need that ObfOff algorithm should know the input domain $[N]$ of C that is going to be input to ObfOn while in the latter, we do not.

Definition 6.2 (Fixed Input Domain pPRIO). A *fixed input domain* pPRIO scheme has syntax as in Definition 6.1 except that ObfOff takes N (in binary) as an additional input.

Definition 6.3 (Flexible Input Domain pPRIO). A *flexible input domain* pPRIO scheme has syntax exactly as in Definition 6.1, i.e., a variant without inputting N into ObfOff.

Next, we define the properties of a pPRIO scheme.

Definition 6.4 (Correctness for Flexible Input Domain Case). For all security parameters $\lambda \in \mathbb{N}$, for any $C : [N] \to [M]$, $L = L(\lambda)$ such that $\mathsf{size}(C) \leq L$ and every input $x \in [N]$, we have that:

$$\Pr\left[\mathsf{Eval}(\mathsf{obf}, x) = C(x) \middle| \begin{array}{c} \mathsf{obf} = (\mathsf{obf}_{\mathsf{off}}, \mathsf{obf}_{\mathsf{on}}),\ (\mathsf{obf}_{\mathsf{off}}, \mathsf{st}) \leftarrow \mathsf{ObfOff}(1^\lambda, 1^L), \\ \mathsf{obf}_{\mathsf{on}} \leftarrow \mathsf{ObfOn}(\mathsf{st}, C) \end{array}\right] = 1$$

where the probability is taken over the coin-tosses of the obfuscator Obf. The correctness for the fixed input domain case is defined as above, except that $\mathsf{ObfOff}(1^\lambda, 1^L)$ is replaced with $\mathsf{ObfOff}(1^\lambda, 1^L, N)$.

We introduce a bit unusual efficiency requirement for pPRIO. Namely, we require that pPRIO can obfuscate circuits from exponentially large input domain efficiently. One may think that with this property, pPRIO is equivalent to prIO. However, for pPRIO, we require that it can securely obfuscate a circuit only when it has polynomial size domain and do not require any security guarantee when it has superpolynomial domain size. This relaxation leads to the construction with weaker assumption (i.e., uniform and non-κ variant of evasive LWE).

The following efficiency requirement is the consequence of inputting N into ObfOff in *binary* form and the requirement that ObfOff is a PPT algorithm, which implies that it runs in polynomial time in the input length. We explicitly require this for emphasizing the property.

Definition 6.5 (Efficiency for Fixed Input Domain). We require that the running time of $\mathsf{ObfOff}(1^\lambda, 1^L, N)$ is bounded by $\mathrm{poly}(\lambda, L, \log N)$.

We introduce the security definition for pPRIO.

Definition 6.6 (Security for Flexible Input Domain). Let Samp be a PPT algorithm that on input 1^λ, outputs

$$\left(1^{N_1+N_2+\cdots N_Q}, 1^L, \mathsf{aux}, C^1, \ldots, C^Q\right), \quad \text{where} \quad C^i : [N_i] \to [M_i],\ \ \mathsf{size}(C^i) \leq L$$

where we enforce Samp to output $1^{N_1+N_2+\cdots N_Q}$ to make sure that all N_i are bounded by $\mathrm{poly}(\lambda)$. We say that a flexible input domain pPRIO scheme is secure with respect to the sampler class $\mathcal{SC}$ if for every PPT sampler Samp $\in \mathcal{SC}$ the following holds.

$$\begin{array}{ll} \text{If} & \left(\mathsf{aux}, 1^L, \{C^1(i)\}_{i\in[N_1]}, \ldots, \{C^Q(i)\}_{i\in[N_Q]}\right) \approx_c \left(\mathsf{aux}, 1^L, \{\Delta_i^1\}_{i\in[N_1]}, \ldots, \{\Delta_i^Q\}_{i\in[N_Q]}\right) \\ \text{then} & \left(\mathsf{aux}, \mathsf{obf}_{\mathsf{off}}, \mathsf{obf}^1_{\mathsf{on}}, \ldots, \mathsf{obf}^Q_{\mathsf{on}}\right) \approx_c \left(\mathsf{aux}, \mathsf{obf}_{\mathsf{off}}, \delta^1 \leftarrow \mathcal{CT}^1 \ldots, \delta^Q \leftarrow \mathcal{CT}^Q\right), \end{array}$$

where $\Delta_i^j \leftarrow [M_j]$ for $j \in [Q], i \in [N_j]$, $(\mathsf{obf}_{\mathsf{off}}, \mathsf{st}) \leftarrow \mathsf{ObfOff}(1^\lambda, 1^L)$, $\mathsf{obf}^j_{\mathsf{on}} \leftarrow \mathsf{ObfOn}(\mathsf{st}, C^j)$ for $j \in [Q]$, and $\mathcal{CT}^j$ denotes the set of binary strings of the same length as the output of $\mathsf{obf}_{\mathsf{on}}(\mathsf{st}, C^j)$ algorithm.

Definition 6.7 (Security for Fixed Input Domain). The security for flexible input domain pPRIO is defined exactly as in Definition 6.6, except that we have $N_1 = N_2 = \cdots = N_Q$.

Remark 6.8 (Comparison with Definition 5.3). Here, we compare the above security definitions for pPRIO with that for prIO (Definition 5.3). On the one hand, the above security definition is weaker than Definition 5.3 in that the adversary is not allowed to submit a circuit with exponential size input domain. This restriction is captured by the above definitions where we enforce the adversary to output the size of the input domain for each circuit in *unary*.

On the other hand, the above security definitions are stronger than Definition 5.3 in two folds. First, it requires the online part of the obfuscation $\mathsf{obf}_{\mathsf{on}}$ to be *pseudorandom*, whereas Definition 5.3 only requires the obfuscation of the circuits are *computationally indistinguishable* (when their truth tables are pseudorandom). Secondly, the above security definition considers a security game where the same internal state st is reused for obfuscating *multiple* circuits, whereas only a single circuit is obfuscated in Definition 5.3. The above security definition may look odd, but pPRIO with this security notion has proven useful in our companion paper [AKY24b] to obtain optimal parameters for ABE schemes.

Remark 6.9. Similar to Remark 5.5, there is no pPRIO scheme satisfying the above security for all general samplers. Therefore, when we use the security of pPRIO, we invoke the security *with respect to a specific sampler class* that is induced by the respective applications.

6.2 Construction for Fixed Input Domain

In this section, we provide a construction of fixed input domain pPRIO scheme $\mathsf{pPRIO} = (\mathsf{Obf} = (\mathsf{ObfOff}, \mathsf{ObfOn}), \mathsf{Eval})$ for circuit class $\mathcal{C} = \{C : [\lambda^c] \to \{0,1\}\}$, for some constant $c \in \mathbb{N}$, from prMIFE for $c+1$ arity. The construction is extended to be able to handle flexible input domain in Section 6.3.

Building Blocks. We use the following ingredient for our construction.

1. A mi-prFE scheme $\mathsf{prMIFE} = \mathsf{prMIFE}.(\mathsf{Setup}, \mathsf{KeyGen}, \{\mathsf{Enc}_i\}_{i\in[c+1]}, \mathsf{Dec})$ for the circuit class with fixed input length $L = |C|$, bounded depth $d = d(\lambda)$, and binary output. We require prMIFE to satisfy the pseudorandomness of the last slot ciphertext as per Definition 3.4. We can instantiate the scheme by our construction in Section 3.2.

Next, we describe our construction.

$\mathsf{ObfOff}(1^\lambda, 1^L, N = \lambda^c)$. The offline phase of obfuscation algorithm does the following.

- Run $(\mathsf{prMIFE.mpk}, \mathsf{prMIFE.msk}) \leftarrow \mathsf{prMIFE.Setup}(1^\lambda, 1^{c+1}, \mathsf{prm})$, where prm specifies the maximum size L of the circuit that is going to input to ObfOn and the maximum depth d of the circuit class supported by the prMIFE instance. We set d to be the depth of the universal circuit U that, upon input a c-ary circuit C and vector $\mathbf{x} \in [\lambda]^c$, outputs $U(C, \mathbf{x}) = C(\mathbf{x})$, where $\mathbf{x}$ is interpreted as an integer in $[\lambda^c]$ by some efficient bijective mapping between $[\lambda^c]$ and $[\lambda]^c$.
- For $i \in [c]$ and $j \in [\lambda]$, compute $\mathsf{prMIFE.ct}_{i,j} = \mathsf{prMIFE.Enc}_i(\mathsf{prMIFE.msk}, j)$.[16]
- Compute $\mathsf{prMIFE.sk}_U = \mathsf{prMIFE.KeyGen}(\mathsf{prMIFE.msk}, U)$, where U is supported by the prMIFE instance because of our choice of prm.
- Output $\mathsf{obf}_{\mathsf{off}} := \left(\mathsf{prMIFE.mpk}, \{\mathsf{prMIFE.ct}_{i,j}\}_{i\in[c], j\in[\lambda]}, \mathsf{prMIFE.sk}_U\right)$ and $\mathsf{st} = \mathsf{prMIFE.msk}$.

$\mathsf{ObfOn}(\mathsf{st}, C)$. The online phase of obfuscation algorithm does the following.

- Run $\mathsf{prMIFE.ct}_{c+1} \leftarrow \mathsf{prMIFE.Enc}_{c+1}(\mathsf{prMIFE.msk}, C)$.
- Output $\mathsf{obf}_{\mathsf{on}} := \mathsf{prMIFE.ct}_{c+1}$.

$\mathsf{Eval}(\mathsf{obf}, \mathbf{x})$. The evaluation algorithm does the following.

1. Parse $\mathsf{obf} = (\mathsf{obf}_{\mathsf{off}}, \mathsf{obf}_{\mathsf{on}})$ where $\mathsf{obf}_{\mathsf{off}} = \left(\mathsf{prMIFE.mpk}, \{\mathsf{prMIFE.ct}_{i,j}\}_{i\in[c], j\in[\lambda]}, \mathsf{prMIFE.sk}_U\right)$ and $\mathsf{obf}_{\mathsf{on}} = \mathsf{prMIFE.ct}_{c+1}$.

[16] Using Remark 5.6, it is safe to use this instance of prMIFE to encrypt messages with smaller length than L.

2. Output $\mathsf{prMIFE.Dec}(\mathsf{prMIFE.mpk}, \mathsf{prMIFE.sk}_U, U, \mathsf{prMIFE.ct}_{1,x_1}, \ldots, \mathsf{prMIFE.ct}_{c,x_c}, \mathsf{prMIFE.ct}_{c+1})$, where $\mathbf{x} \in [\lambda^c]$ is mapped to $(x_1, \ldots, x_c) \in [\lambda]^c$ by the bijective mapping between $[\lambda^c]$ and $[\lambda]^c$.

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.

Efficiency. The scheme satisfies

$$|\mathsf{obf}_{\mathsf{off}}| = \mathrm{poly}(c, L, \lambda) = \mathrm{poly}(\lambda, L), |\mathsf{obf}_{\mathsf{on}}| = \mathrm{poly}(c, L, \lambda) = \mathrm{poly}(\lambda, L),$$

where $(\mathsf{obf}_{\mathsf{off}}, \mathsf{obf}_{\mathsf{on}}) \leftarrow \mathsf{Obf}(1^\lambda, C)$ for circuit $C : [N] \to [M]$ whose size is bounded by $L = L(\lambda)$.

Security. We prove the security of the fixed input pPRIO using the following theorem.

Theorem 6.10. Let $\mathcal{SC}_{\mathsf{pPRIO}}$ be a sampler class for pPRIO. Suppose prMIFE scheme is secure (Definition 3.2) with $\kappa = \lambda^{c+1}$ and sampler class that contains all $\mathsf{Samp}_{\mathsf{prMIFE}}$ induced by $\mathsf{Samp}_{\mathsf{prIO}} \in \mathcal{SC}_{\mathsf{prIO}}$, as defined in Equation (42). Then the pPRIO scheme satisfies security as defined in Definition 6.7.

Proof. Consider a sampler Samp that generates

$$\left(1^{Q \cdot N}, 1^L, \mathsf{aux}_{\mathcal{A}}, \{C^k\}_{k \in [Q]}\right).$$

To prove the theorem, we show that

$$\left(\mathsf{aux}_{\mathcal{A}}, \mathsf{obf}_{\mathsf{off}} = \left(\mathsf{prMIFE.mpk}, \{\mathsf{prMIFE.ct}_{i,j}\}_{i \in [c], j \in [\lambda]}, \mathsf{prMIFE.sk}_U\right), \left\{\mathsf{obf}^k_{\mathsf{on}} = \mathsf{prMIFE.ct}^k_{c+1}\right\}_{k \in [Q]}\right)$$
$$\approx_c \left(\mathsf{aux}_{\mathcal{A}}, \mathsf{obf}_{\mathsf{off}} = \left(\mathsf{prMIFE.mpk}, \{\mathsf{prMIFE.ct}_{i,j}\}_{i \in [c], j \in [\lambda]}, \mathsf{prMIFE.sk}_U\right), \left\{\mathsf{obf}^k_{\mathsf{on}} \leftarrow \mathcal{CT}_{\mathsf{prMIFE}, c+1}\right\}_{k \in [Q]}\right),$$

holds assuming

$$\left(\mathsf{aux}_{\mathcal{A}}, 1^L, \{C^1(i)\}_{i \in [N]}, \ldots, \{C^Q(i)\}_{i \in [N]}\right) \approx_c \left(\mathsf{aux}_{\mathcal{A}}, 1^L, \{\Delta^1_i\}_{i \in [N]}, \ldots, \{\Delta^Q_i\}_{i \in [N]}\right) \tag{41}$$

where $\mathcal{CT}_{\mathsf{prMIFE}, c+1}$ denotes the set of binary strings with the same length as $\mathsf{prMIFE.ct}^k_{c+1}$ and

$$\begin{aligned}
&\left(1^{Q \cdot N}, 1^L, \mathsf{aux}_{\mathcal{A}}, \{C^k\}_{k \in [Q]}\right) \leftarrow \mathsf{Samp}(1^\lambda),\\
&(\mathsf{prMIFE.mpk}, \mathsf{prMIFE.msk}) \leftarrow \mathsf{prMIFE.Setup}(1^\lambda, 1^{c+1}, \mathsf{prm}),\\
&\mathsf{prMIFE.sk}_U \leftarrow \mathsf{prMIFE.KeyGen}(\mathsf{prMIFE.msk}, U),\\
&\mathsf{prMIFE.ct}_{i,j} \leftarrow \mathsf{prMIFE.Enc}_i(\mathsf{prMIFE.msk}, j) \text{ for } i \in [c], j \in [\lambda],\\
&\Delta^k_1, \ldots, \Delta^k_N \leftarrow \{0,1\} \text{ for } k \in [Q],\\
&\mathsf{prMIFE.ct}^k_{c+1} \leftarrow \mathsf{prMIFE.Enc}_{c+1}(\mathsf{prMIFE.msk}, C^k) \text{ for } k \in [Q].
\end{aligned}$$

We invoke the security of prMIFE with sampler $\mathsf{Samp}_{\mathsf{prMIFE}}$ that outputs

$$\begin{pmatrix} \text{Function:} & \text{Universal Circuit } U \\ \text{Inputs:} & \{x^{j_1}_1 = j_1, \ldots, x^{j_c}_c = j_c, x^k_{c+1} = C^k\}_{j_1, \ldots, j_c \in [\lambda], k \in [Q]} \\ \text{Auxiliary Information:} & \mathsf{aux}_{\mathcal{A}} \end{pmatrix} \tag{42}$$

Using the prMIFE security with sampler $\mathsf{Samp}_{\mathsf{prMIFE}}$, we have that

$$\left(\mathsf{aux}_{\mathcal{A}}, \mathsf{prMIFE.mpk}, \{\mathsf{prMIFE.ct}_{i,j}\}_{i \in [c], j \in [\lambda]}, \mathsf{prMIFE.sk}_U, \left\{\mathsf{prMIFE.ct}^k_{c+1}\right\}_{k \in [Q]}\right)$$
$$\approx_c \left(\mathsf{aux}_{\mathcal{A}}, \mathsf{prMIFE.mpk}, \{\delta_{i,j}\}_{i \in [c], j \in [\lambda]}, \mathsf{prMIFE.sk}_U, \left\{\delta^k_{c+1}\right\}_{k \in [Q]}\right) \tag{43}$$

given

$$\left(\mathsf{aux}_{\mathcal{A}}, \left\{U(C^k, x_1^{j_1}, \ldots, x_c^{j_c})\right\}_{j_1,\ldots,j_c\in[\lambda],k\in[Q]}\right) \approx_c \left(\mathsf{aux}_{\mathcal{A}}, \{\Delta_i^k \leftarrow \{0,1\}\}_{i\in[N],k\in[Q]}\right) \tag{44}$$

where $\delta_{i,j} \leftarrow \mathsf{Sim}_i(\mathsf{prMIFE.msk})$ for $i \in [c], j \in [\lambda]$ and $\delta_{c+1}^k \leftarrow \mathcal{CT}_{\mathsf{prMIFE},c+1}$[17].
This implies

$$\begin{aligned}
&\left(\mathsf{aux}_{\mathcal{A}}, \mathsf{prMIFE.mpk}, \{\mathsf{prMIFE.ct}_{i,j}\}_{i\in[c],j\in[\lambda]}, \mathsf{prMIFE.sk}_U, \left\{\mathsf{prMIFE.ct}_{c+1}^k\right\}_{k\in[Q]}\right)\\
\approx_c &\left(\mathsf{aux}_{\mathcal{A}}, \mathsf{prMIFE.mpk}, \{\delta_{i,j}\}_{i\in[c],j\in[\lambda]}, \mathsf{prMIFE.sk}_U, \left\{\delta_{c+1}^k\right\}_{k\in[Q]}\right)\\
\approx_c &\left(\mathsf{aux}_{\mathcal{A}}, \mathsf{prMIFE.mpk}, \{\mathsf{prMIFE.ct}_{i,j}\}_{i\in[c],j\in[\lambda]}, \mathsf{prMIFE.sk}_U, \left\{\delta_{c+1}^k\right\}_{k\in[Q]}\right)
\end{aligned}$$

given Equation (44), where the first indistingusihability follows from Equation (43) and the second also from Equation (43) by noting that replacing the last term with independently chosen random variable makes the distinguishing task of the distributions even harder. Equation (44) follows from Equation (41) and the definition of circuit U, hence the proof. □

Instantiation. Instantiating the underlying prMIFE scheme with constant arity $c+1$ as in Theorem 3.17, we get the following theorem.

Theorem 6.11. Assuming evasive LWE (Assumption 2.1) and LWE (Assumption A.5), there exists a secure pPRIO scheme for fixed input domain supporting circuits of bounded size $L = \mathrm{poly}(\lambda)$ with

$$|\mathsf{obf}_{\mathsf{off}}| = \mathrm{poly}(L, \lambda), \quad |\mathsf{obf}_{\mathsf{on}}| = \mathrm{poly}(L, \lambda).$$

6.3 Construction for Flexible Input Domain

Here, we provide a construction of pPRIO for flexible input domain $\mathsf{Flex.pPRIO} = \mathsf{Flex.}(\mathsf{ObfOff}, \mathsf{ObfOn}, \mathsf{Eval})$ for any circuits with binary output space from pPRIO for fixed input domain. The restriction that the output domain of the circuits being binary is removed in Section 6.4.

Building Blocks. Below, we list the ingredients for our construction.

1. A pseudorandom function $\mathsf{PRF} : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^\lambda$ with key space, input space and output space as $\{0,1\}^\lambda$. The input to the PRF will be in the form of $\mathbf{x} \in [\lambda^c]$ for some constant $c \in \mathbb{N}$. Since we have $2^\lambda > \mathrm{poly}(\lambda)$, the input space of the PRF is large enough to accommodate inputs $\mathbf{x} \in [\lambda^c]$ with an appropriate encoding. It is known that PRFs can be constructed from one-way functions.
2. A fixed input domain pPRIO scheme $\mathsf{Fixed.}(\mathsf{ObfOff}, \mathsf{ObfOn}, \mathsf{Eval})$ as constructed in Section 6.2 with the efficiency requirement defined in Definition 6.5 and the security defined as per Definition 6.6. The input domain of pPRIO should be $[\lambda^c]$ for some constant c.

$\mathsf{Flex.Obf}(1^\lambda, C)$. For a circuit C such that $\mathsf{size}(C) \le L$ for $L = L(\lambda)$, the obfuscation algorithm works as following.

$\mathsf{Flex.ObfOff}(1^\lambda, 1^L)$. The offline phase of obfuscation algorithm does the following.

- Run $(\mathsf{Fixed.obf}_{\mathsf{off},i}, \mathsf{Fixed.st}_i) \leftarrow \mathsf{Fixed.ObfOff}(1^\lambda, 1^L, \lambda^i)$ for $i \in [\lambda]$.
- Output $\mathsf{obf}_{\mathsf{off}} := \{\mathsf{Fixed.obf}_{\mathsf{off},i}\}_{i\in[\lambda]}$ and $\mathsf{st} = \{\mathsf{Fixed.st}_i\}_{i\in[\lambda]}$.

$\mathsf{Flex.ObfOn}(\mathsf{st}, C)$. The online phase of obfuscation algorithm takes as input a circuit $C : [N] \to \{0,1\}$ and parses the input as $\mathsf{st} = \{\mathsf{Fixed.st}_i\}_{i\in[\lambda]}$. It then does the following:

[17]This follows from the fact that the simulator for the $c+1$-th slot outputs a random string as is required by Definition 3.4.

- Find $c \in \mathbb{N}$ such that $\lambda^{c-1} < N \le \lambda^c$.
- Sample $\mathsf{sd} \leftarrow \{0,1\}^\lambda$
- Construct a circuit $C[\mathsf{sd}] : [\lambda^c] \to \{0,1\}$ that is defined as, on input $\mathbf{x} \in [\lambda^c]$, $C[\mathsf{sd}](\mathbf{x}) = \begin{cases} C(\mathbf{x}) & \text{if } \mathbf{x} \le N \\ \mathsf{PRF}(\mathsf{sd}, \mathbf{x}) & \text{if } N < \mathbf{x} \le \lambda^c \end{cases}$.
- Compute $\mathsf{Fixed.obf}_{\mathsf{on},c} \leftarrow \mathsf{Fixed.ObfOn}(\mathsf{Fixed.st}_c, C[\mathsf{sd}])$.
- Output $\mathsf{obf}_{\mathsf{on}} := \mathsf{Fixed.obf}_{\mathsf{on},c}$.

$\mathsf{Flex.Eval}(\mathsf{obf}, \mathbf{x})$. The evaluation algorithm does the following.

- Parse $\mathsf{obf} = (\mathsf{obf}_{\mathsf{off}}, \mathsf{obf}_{\mathsf{on}}) = (\{\mathsf{Fixed.obf}_{\mathsf{off},i}\}_{i\in[\lambda]}, \mathsf{Fixed.obf}_{\mathsf{on},c})$.
- Construct $\mathsf{Fixed.obf} = (\mathsf{Fixed.obf}_{\mathsf{off},c}, \mathsf{Fixed.obf}_{\mathsf{on},c})$[18].
- Output $\mathsf{Fixed.Eval}(\mathsf{Fixed.obf}, \mathbf{x})$.

Correctness. For $\mathsf{obf} = (\mathsf{obf}_{\mathsf{off}}, \mathsf{obf}_{\mathsf{on}}) = (\{\mathsf{Fixed.obf}_{\mathsf{off},i}\}_{i\in[\lambda]}, \mathsf{Fixed.obf}_{\mathsf{on},c})$, we have $\mathsf{Fixed.obf}_{\mathsf{off},c} \leftarrow \mathsf{Fixed.ObfOff}(1^\lambda, 1^L, \lambda^c)$ and $\mathsf{Fixed.obf}_{\mathsf{on},c} \leftarrow \mathsf{Fixed.ObfOn}(\mathsf{Fixed.st}_c, C[\mathsf{sd}])$ where $C[\mathsf{sd}] : [\lambda^c] \to \{0,1\}$. From the correctness of the underlying $\mathsf{Fixed.pPRIO}$ scheme we have $\mathsf{Fixed.Eval}(\mathsf{Fixed.obf}, \mathbf{x}) = C[\mathsf{sd}](\mathbf{x}) = C(\mathbf{x})$ for any input $\mathbf{x} \in [\lambda^c]$.

Efficiency. It is straightforward to see that the algorithms run in polynomial time in the input length and λ. The only non-trivial part is to bound the running time of $\mathsf{Fixed.ObfOff}(1^\lambda, 1^L, \lambda^i)$ for $i \in [\lambda]$. The running time of this part can be bounded by $\mathrm{poly}(\lambda, L, \log \lambda^i) = \mathrm{poly}(\lambda, L)$ by the efficiency property of the underlying fixed input domain pPRIO as per Definition 6.5. In particular, we have

$$|\mathsf{obf}_{\mathsf{off}}| = \mathrm{poly}(L, \lambda), \quad |\mathsf{obf}_{\mathsf{on}}| = \mathrm{poly}(L, \lambda)$$

where $(\mathsf{obf}_{\mathsf{off}}, \mathsf{obf}_{\mathsf{on}}) \leftarrow \mathsf{Obf}(1^\lambda, C)$ for circuit $C : [N] \to [M]$ whose size is bounded by $L = L(\lambda)$.

Security. We prove the security of the above construction. Before doing so, we prove the following useful lemma. The lemma essentially says that if a part of the auxiliary information is pseudorandom in the pre-condition distribution, then it is pseudorandom in the corresponding post-condition distribution where we apply pPRIO. Conceptually similar lemma is proven in Lemma 3.4 of [ARYY23] in the context of evasive LWE.

Lemma 6.12. Let $\mathsf{pPRIO} = (\mathsf{Fixed.ObfOff}, \mathsf{Fixed.ObfOn}, \mathsf{Fixed.Eval})$ be a pPRIO scheme for fixed input domain and Samp be a PPT algorithm that takes as input 1^λ and outputs

$$\left(1^{NQ}, 1^L, \mathsf{aux} = (\mathsf{aux}_1, \mathsf{aux}_2) \in \{0,1\}^* \times \mathcal{X}, \{C^j\}_{j\in[Q]}\right)$$

for some set $\mathcal{X}$. Let us assume that

$$\left((\mathsf{aux}_1, \mathsf{aux}_2), 1^L, \left\{C^j(i)\right\}_{i\in[N], j\in[Q]}\right) \approx_c \left((\mathsf{aux}_1, \mathbf{x}), 1^L, \left\{\Delta_i^j\right\}_{i\in[N], j\in[Q]}\right)$$

holds for $\mathbf{x} \leftarrow \mathcal{X}, \Delta_i^j \leftarrow \{0,1\}$ and assume the security of pPRIO with respect to Samp. We then have

$$\left((\mathsf{aux}_1, \mathsf{aux}_2), \mathsf{obf}_{\mathsf{off}}, \left\{\mathsf{obf}_{\mathsf{on}}^j\right\}_{j\in[Q]}\right) \approx_c \left((\mathsf{aux}_1, \mathbf{x}), \mathsf{obf}_{\mathsf{off}}, \left\{\delta^j\right\}_{j\in[Q]}\right),$$

where we have $(\mathsf{obf}_{\mathsf{off}}, \mathsf{st}) \leftarrow \mathsf{Fixed.ObfOff}(1^\lambda, 1^L, N)$, $\mathsf{obf}_{\mathsf{on}}^j \leftarrow \mathsf{Fixed.ObfOn}(\mathsf{st}, C^j)$ and $\delta^j \leftarrow \mathcal{CT}_{\mathsf{ObfOn}}$ for $j \in [Q]$, where $\mathcal{CT}_{\mathsf{ObfOn}}$ is the set of binary strings with the same length as $\mathsf{obf}_{\mathsf{on}}^j$.

[18] We assume that we can obtain c from the length of $\mathbf{x}$.

Proof. From the assumption, we have

$$\left((\mathsf{aux}_1, \mathsf{aux}_2), 1^L, \left\{C^j(i)\right\}_{i\in[N], j\in[Q]}\right) \approx_c \left((\mathsf{aux}_1, \mathbf{x}), 1^L, \left\{\Delta_i^j\right\}_{i\in[N], j\in[Q]}\right) \tag{45}$$

which implies $(\mathsf{aux}_1, \mathsf{aux}_2, 1^L) \approx_c (\mathsf{aux}_1, \mathbf{x}, 1^L)$. This further implies

$$(\mathsf{aux}_1, \mathsf{aux}_2, 1^L, \{\Delta_i^j\}_{i\in[N], j\in[Q]}) \approx_c (\mathsf{aux}_1, \mathbf{x}, 1^L, \{\Delta_i^j\}_{i\in[N], j\in[Q]}) \tag{46}$$

since adding independently sampled random terms does not make the task of indistinguishing the distributions easier. Equation (45) and Equation (46) implies $(\mathsf{aux}_1, \mathsf{aux}_2, 1^L, \{C^j(i)\}_{i\in[N], j\in[Q]}) \approx_c (\mathsf{aux}_1, \mathsf{aux}_2, 1^L, \{\Delta_i^j\}_{i\in[N], j\in[Q]})$. Applying pPRIO security definition with respect to Samp, we get

$$\left(\mathsf{aux}_1, \mathsf{aux}_2, \mathsf{obf}_{\mathsf{off}}, \left\{\mathsf{obf}_{\mathsf{on}}^j\right\}_{j\in[Q]}\right) \approx_c \left(\mathsf{aux}_1, \mathsf{aux}_2, \mathsf{obf}_{\mathsf{off}}, \left\{\delta^j\right\}_{j\in[Q]}\right). \tag{47}$$

Recall that we have $(\mathsf{aux}_1, \mathsf{aux}_2, 1^L) \approx_c (\mathsf{aux}_1, \mathbf{x}, 1^L)$. This further implies

$$\left(\mathsf{aux}_1, \mathsf{aux}_2, \mathsf{obf}_{\mathsf{off}}, \left\{\delta^j\right\}_{j\in[Q]}\right) \approx_c \left(\mathsf{aux}_1, \mathbf{x}, \mathsf{obf}_{\mathsf{off}}, \left\{\delta^j\right\}_{j\in[Q]}\right), \tag{48}$$

since $\mathsf{obf}_{\mathsf{off}}$ can be sampled independently given $1^L, N$ and $\{\delta^j\}_{j\in[Q]}$.

From Equation (47) and Equation (48) we deduce

$$\left(\mathsf{aux}_1, \mathsf{aux}_2, \mathsf{obf}_{\mathsf{off}}, \left\{\mathsf{obf}_{\mathsf{on}}^j\right\}_{j\in[Q]}\right) \approx_c \left(\mathsf{aux}_1, \mathbf{x}, \mathsf{obf}_{\mathsf{off}}, \left\{\delta^j\right\}_{j\in[Q]}\right)$$

and hence the lemma. □

Following theorem asserts the security of our construction.

Theorem 6.13. Our pPRIO scheme for flexible input domain is secure.

Proof. Let us consider a sampler Samp that outputs $\left(1^{\sum_{j\in[Q]} N_j}, 1^L, \mathsf{aux}, \{C^j\}_{j\in[Q]}\right)$. To prove the theorem, we show that

$$\left(\mathsf{aux}, \mathsf{obf}_{\mathsf{off}}, \mathsf{obf}_{\mathsf{on}}^1, \ldots, \mathsf{obf}_{\mathsf{on}}^Q\right) \approx_c \left(\mathsf{aux}, \mathsf{obf}_{\mathsf{off}}, \delta^1 \ldots, \delta^Q\right), \tag{49}$$

holds assuming

$$\left(\mathsf{aux}, \{C^1(i)\}_{i\in[N_1]}, \ldots, \{C^Q(i)\}_{i\in[N_Q]}\right) \approx_c \left(\mathsf{aux}, \{\Delta_i^1\}_{i\in[N_1]}, \ldots, \{\Delta_i^Q\}_{i\in[N_Q]}\right)$$

where

$\mathsf{obf}_{\mathsf{off}} = \{\mathsf{Fixed.obf}_{\mathsf{off},i}\}_{i\in[\lambda]}$, $(\mathsf{Fixed.obf}_{\mathsf{off},i}, \mathsf{Fixed.st}_i) \leftarrow \mathsf{Fixed.ObfOff}(1^\lambda, 1^L, \lambda^i)$ for $i \in [\lambda]$,

$\mathsf{obf}_{\mathsf{on}}^1 = \mathsf{Fixed.obf}_{\mathsf{on},c_j}$, $\mathsf{Fixed.obf}_{\mathsf{on},c_j} \leftarrow \mathsf{Fixed.ObfOn}(\mathsf{Fixed.st}_{c_j}, C^j[\mathsf{sd}^j])$ for $c_j \in \mathbb{N}$ s.t. $\lambda^{c_j-1} < N_j \leq \lambda^{c_j}$,

$\delta^j \leftarrow \mathcal{CT}^j$ where $\mathcal{CT}^j$ is the set of binary strings with the same length as $\mathsf{obf}_{\mathsf{on}}^j$, for $j \in [Q]$

$\Delta_i^j \leftarrow \{0,1\}$ for $j \in [Q], i \in [N_j]$

We define S_k to be $S_k := \{j \in [Q] : \lambda^{k-1} < N_j \leq \lambda^k\}$. Since Samp is a PPT algorithm, there exists a constant c_{max} such that $\lambda^{c_{\mathsf{max}}-1} < \mathsf{max}_{j\in[Q]} N_j \leq \lambda^{c_{\mathsf{max}}}$ and therefore we have $[Q] = \cup_{k\in[c_{\mathsf{max}}]} S_k$. Rearranging the terms, it suffices to show that

$$\left(\mathsf{aux}, \mathsf{obf}_{\mathsf{off}}, \{\mathsf{obf}_{\mathsf{on}}^j\}_{j\in S_1}, \ldots, \{\mathsf{obf}_{\mathsf{on}}^j\}_{j\in S_{c_{\mathsf{max}}}}\right) \approx_c \left(\mathsf{aux}, \mathsf{obf}_{\mathsf{off}}, \{\delta^j\}_{j\in S_1}, \ldots, \{\delta^j\}_{j\in S_{c_{\mathsf{max}}}}\right)$$

holds assuming

$$\left(\mathsf{aux}, \left\{C^j(i)\right\}_{j\in S_1, i\in[N_j]}, \ldots, \left\{C^j(i)\right\}_{j\in S_{c_{\mathsf{max}}}, i\in[N_j]}\right) \approx_c \left(\mathsf{aux}, \left\{\Delta_i^j\right\}_{j\in S_1, i\in[N_j]}, \ldots, \left\{\Delta_i^j\right\}_{j\in S_{c_{\mathsf{max}}}, i\in[N_j]}\right). \quad (50)$$

To prove this, we prove the following lemma.

Lemma 6.14. For $c \in [0, c_{\mathsf{max}}]$, let us consider the following statement:

$$\begin{aligned}&\left(\mathsf{aux}, \mathsf{obf}_{\mathsf{off}}, \{\mathsf{obf}_{\mathsf{on}}^j\}_{j\in S_1}, \ldots, \{\mathsf{obf}_{\mathsf{on}}^j\}_{j\in S_c}, \left\{C^j[\mathsf{sd}^j](i)\right\}_{j\in S_{c+1}, i\in[\lambda^{c+1}]}, \ldots, \left\{C^j[\mathsf{sd}^j](i)\right\}_{j\in S_{c_{\mathsf{max}}}, i\in[\lambda^{c_{\mathsf{max}}}]}\right)\\ \approx_c &\left(\mathsf{aux}, \mathsf{obf}_{\mathsf{off}}, \{\delta^j\}_{j\in S_1}, \ldots, \{\delta^j\}_{j\in S_c}, \left\{\Delta_i^j\right\}_{j\in S_{c+1}, i\in[\lambda^{c+1}]}, \ldots, \left\{\Delta_i^j\right\}_{j\in S_{c_{\mathsf{max}}}, i\in[\lambda^{c_{\mathsf{max}}}]}\right). \quad (51)\end{aligned}$$

Then, assuming the security of pPRIO, the above computational indistinguishability for $c = c^*$ implies that for $c^* + 1$ for all $c^* \in [0, c_{\mathsf{max}} - 1]$.

Proof. By setting

$$\begin{aligned}\mathsf{aux}_1 &:= \left(\mathsf{aux}, \left\{\mathsf{Fixed.obf}_{\mathsf{off},k}\right\}_{k\in[\lambda]\setminus\{c^*+1\}}\right),\\ \mathsf{aux}_2 &:= \left(\{\mathsf{obf}_{\mathsf{on}}^j\}_{j\in S_1}, \ldots, \{\mathsf{obf}_{\mathsf{on}}^j\}_{j\in S_{c^*}}, \left\{C^j[\mathsf{sd}^j](i)\right\}_{j\in S_{c^*+2}, i\in[\lambda^{c^*+2}]}, \ldots, \left\{C^j[\mathsf{sd}^j](i)\right\}_{j\in S_{c_{\mathsf{max}}}, i\in[\lambda^{c_{\mathsf{max}}}]}\right),\\ \text{Circuits} &:= \left\{C^j[\mathsf{sd}^j]\right\}_{j\in S_{c^*+1}}\end{aligned}$$

Equation (51) with $c = c^*$ implies

$$\left((\mathsf{aux}_1, \mathsf{aux}_2), \left\{C^j[\mathsf{sd}^j](i)\right\}_{j\in S_{c^*+1}, i\in[\lambda^{c^*+1}]}\right) \approx_c \left((\mathsf{aux}_1, \mathbf{x}), \left\{\Delta_i^j\right\}_{j\in S_{c^*+1}, i\in[\lambda^{c^*+1}]}\right),$$

where $\mathbf{x}$ is a random string with the same length as aux_2. By applying Lemma 6.12, we have

$$\left((\mathsf{aux}_1, \mathsf{aux}_2), \mathsf{obf}_{\mathsf{off},c^*+1}, \left\{\mathsf{obf}_{\mathsf{on}}^j\right\}_{j\in S_{c^*+1}}\right) \approx_c \left((\mathsf{aux}_1, \mathbf{x}), \mathsf{obf}_{\mathsf{off},c^*+1}, \left\{\delta^j\right\}_{j\in S_{c^*+1}}\right).$$

By rearranging the terms, we can observe that the above equation is equivalent to Equation (51) with $c = c^* + 1$. ☐

We then conclude the proof of Theorem 6.13 using Lemma 6.14. Our goal is to prove Equation (51) with $c = c_{\mathsf{max}}$, since it is equivalent to Equation (49). This is proven by the induction, where the induction step is already provided by Lemma 6.14. Therefore, it suffices to prove Equation (51) with $c = 0$. This immediately follows from Equation (50) and by the security of PRF, since the LHS distribution of Equation (51) with $c = 0$ is obtained by padding the LHS distribution of Equation (50) with PRF values $\{\mathsf{PRF}(\mathsf{sd}^j, i)\}_{c\in[c_{\mathsf{max}}], j\in S_c, i\in[N_j+1, \lambda^c]}$, which are pseudorandom. This completes the proof of Theorem 6.13. ☐

Instantiation. Instantiating the underlying pPRIO scheme as in Theorem 6.11, we get the following theorem.

Theorem 6.15. Assuming evasive LWE (Assumption 2.1) and LWE (Assumption A.5), there exists a secure pPRIO scheme for flexible input domain supporting circuits of bounded size $L = \mathrm{poly}(\lambda)$ with

$$|\mathsf{obf}_{\mathsf{off}}| = \mathrm{poly}(L, \lambda), \quad |\mathsf{obf}_{\mathsf{on}}| = \mathrm{poly}(L, \lambda).$$

6.4 Extending the Output Length.

In our construction of pPRIO with flexible input space in Section 6.3, we only consider the case of the output space of the obfuscated circuit being $\{0,1\}$. We can extend the output space to be $\{0,1\}^{\ell_{\mathsf{out}}}$ for arbitrary polynomial ℓ_{out} as follows. We do not change the off-line phase of the obfuscation algorithm. When we run on-line phase of the obfuscation algorithm, we derive sub-circuits $C_1, \ldots C_{\ell_{\mathsf{out}}}$ of C with binary outputs, where C_j takes on input i and outputs the j-th bit of $C(i)$. We then run $\mathsf{ObfOn}(\mathsf{st}, C_j)$ for all $j \in [\ell_{\mathsf{out}}]$ and output them as the obfuscation of C. The evaluation algorithm then recovers each output separately and combines them. It is straightforward to see that the security of the construction is preserved with this modification. As for the efficiency, we still maintain the asymptotic efficiency of

$$|\mathsf{obf}_{\mathsf{off}}| = \mathrm{poly}(L, \lambda), \quad |\mathsf{obf}_{\mathsf{on}}| = \ell_{\mathsf{out}}\mathrm{poly}(L, \lambda) = \mathrm{poly}(L, \lambda)$$

where $(\mathsf{obf}_{\mathsf{off}}, \mathsf{obf}_{\mathsf{on}}) \leftarrow \mathsf{Obf}(1^\lambda, C)$ for circuit C whose size is bounded by $L = L(\lambda)$.

7 Instantiating the Random Oracles Using prIO

Hohenberger, Sahai, and Waters [HSW14] show that some applications of random oracles (ROM) can be made secure in the standard model by instantiating the hash functions using iO in a specific manner. In this section, we discuss 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. We also refer to Section 8 for the application of prIO for instantiating random oracle in Sakai-Ohgishi-Kasahara ID-based NIKE.

7.1 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.

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_{SK}^{-1}.
   - 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 of the program Full Domain Hash Figure 2. 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 $\sigma = g_{\mathsf{SK}}^{-1}(H(m))$.
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.

Full Domain Hash H

Constants: PRF key K, TDP key PK
Input: Message m.

- Output $g_{\mathsf{PK}}(F(K, m))$.

Figure 2: Full Domain Hash

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 3: Full Domain Hash*

Security The security proof closely resembles that of [HSW14], except that it can be simpler since the security guarantee of prIO is stronger than iO for the specific case of pseudorandom functionalities. In particular, prIO does not require two circuits to have identical truth tables in order to guarantee security, but allows different truth tables so long as they are pseudorandom (given auxiliary information).

Theorem 7.1. If the obfuscation scheme in Section 5.2 is secure as per Definition 5.3 with respect to a parameter κ and samplers that output circuits defined as Figure 2 and 3, F is subexponentially secure punctured PRF, and the trapdoor permutation scheme TDP is one-way, then the above signature scheme is selectively secure.

Proof. The proof follows a similar sequence of hybrids as [HSW14] [19]. In the first hybrid we move to using the obfuscation of the circuit in Figure 3 with z^* being a random point in $\{0, 1\}^n$. Since the truth tables of the two programs are pseudorandom given PK and SK (even against an adversary that runs in polynomial time in κ), indistinguishability holds by the guarantee of prIO. 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 PRF key. □

7.2 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 7.1, 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.

[19]We can skip Hybrid 1 in [HSW14] because we don't need the programs to be exactly equivalent as discussed above.

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 revert to the original ROM-based proof strategy [BLS01, Cor00]. Unlike the iO-based approach, prIO-based proof does not require the truth table of the hash function to remain unchanged across game hops; it only requires the truth table to be pseudorandom. This relaxed condition enables the use of the original ROM security proofs. We omit the details on these applications, since they are similar to and simpler than our random oracle instantiation for the Sakai-Ohgishi-Kasahara ID-based NIKE in Section 8.

8 ID-Based Non-Interactive Key Exchange

In this section we show our construction of ID-NIKE scheme. The construction is the same as the ID-based NIKE system by Sakai, Ohgishi, and Kasahara [SOK00] except that the hash function modeled as random oracle is replaced with an obfuscation of a PRF. This leads to the first instantiation of ID-NIKE without multi-linear maps [FHPS13] or indistinguishability obfuscation in the standard model.

8.1 Construction

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

1. We use prIO scheme $(\mathsf{iO}, \mathsf{Eval})$ with input space $\mathcal{ID} \stackrel{\text{def}}{=} \{0,1\}^n$ that is secure as per Definition 5.3 with parameter κ. By using our construction in Section 5.2 instantiated by prMIFE construction in Section 3.2, we have $\kappa = \lambda^{n^2 \log \lambda}$.

2. Symmetric pairing $\mathcal{G} = (\mathbb{G}, \mathbb{G}_T, p, e, g)$ with prime order p that is equipped with an efficiently computable function $\mathsf{MapToPoint} : \mathcal{D}_{\mathsf{MTP}} \to \mathbb{G}$, where $\mathcal{D}_{\mathsf{MTP}}$ is an efficiently samplable domain. We assume that the DBDH assumption (Assumption 2.6) holds on $\mathcal{G}$. We need that there is an efficiently computable *randomized* function $\mathsf{MapToPoint}^{-1}$ satisfying $\mathsf{MapToPoint}(\mathsf{MapToPoint}^{-1}(h)) = h$ for all $h \in \mathbb{G}$. We also need $\mathsf{MapToPoint}^{-1}(h) \equiv x$ for $x \leftarrow \mathcal{D}_{\mathsf{MTP}}$ and $h \leftarrow \mathbb{G}$. We denote the randomness space of the algorithm $\mathsf{MapToPoint}^{-1}$ by $\mathcal{R}_{\mathsf{MTP}}$.

3. A subexponentially secure pseudorandom function $\mathsf{PRF} : \{0,1\}^\Lambda \times \mathcal{ID} \to \mathcal{D}_{\mathsf{MTP}}$. Here, $\Lambda = \Lambda(\lambda)$ is set so that $2^{\Lambda^\delta} > \kappa^{\omega(1)}$ holds, where δ is defined as a constant such that there is no adversary with size 2^{λ^δ} and distinguishing advantage $2^{-\lambda^\delta}$ against PRF for all sufficiently large λ. An example choice would be to take $\Lambda := (n^2\lambda)^{1/\delta}$.

We describe the construction below.

$\mathsf{Setup}(1^\lambda)$**:** On input the security parameter 1^λ, it chooses the description of symmetric pairing groups $\mathcal{G} = (\mathbb{G}, \mathbb{G}_T, p, e, g)$ with prime order $p > 2^{2\lambda}$, a PRF key $\mathsf{sd} \leftarrow \{0,1\}^\Lambda$, and $\alpha \leftarrow \mathbb{Z}_p$ and computes $\tilde{\mathsf{C}} \leftarrow \mathsf{iO}(\mathsf{PRF}(\mathsf{sd}, \cdot))$. Here, the obfuscated circuit $\mathsf{PRF}(\mathsf{sd}, \cdot)$ is appropriately padded so that its size is the same as the circuit $F[\overline{\mathsf{sd}}, \widetilde{\mathsf{sd}}, g^\beta, g^\gamma]$ described in Figure 4. Finally, it outputs $\mathsf{mpk} = (\mathcal{G}, \tilde{\mathsf{C}})$ and $\mathsf{msk} = \alpha$.

$\mathsf{Ext}(\mathsf{mpk}, \mathsf{msk}, \mathsf{id})$**:** On input $\mathsf{mpk} = (\mathcal{G}, \tilde{\mathsf{C}})$, $\mathsf{msk} = \alpha$, and an identity id, it computes $\mathsf{H}(\mathsf{id})$, where the hash function $\mathsf{H} : \mathcal{ID} \to \mathbb{G}$ is defined as $\mathsf{H}(\mathsf{id}) \stackrel{\text{def}}{=} \mathsf{MapToPoint}(\tilde{\mathsf{C}}(\mathsf{id}))$. It then computes and outputs $\mathsf{usk}_{\mathsf{id}} = \mathsf{H}(\mathsf{id})^\alpha$.

$\mathsf{Share}(\mathsf{mpk}, \mathsf{usk}_{\mathsf{id}_1}, \mathsf{id}_2)$**.** Given the master public key mpk, a user secret key $\mathsf{usk}_{\mathsf{id}_1} = \mathsf{H}(\mathsf{id}_1)^\alpha$, and an identity $\mathsf{id}_2 \in \mathcal{ID}$, it computes and outputs $K \stackrel{\text{def}}{=} e(\mathsf{usk}_{\mathsf{id}_1}, \mathsf{H}(\mathsf{id}_2))$ as the shared key.

Correctness. The correctness of the construction can be seen by $e(\mathsf{usk}_{\mathsf{id}_1}, \mathsf{H}(\mathsf{id}_2)) = e(\mathsf{H}(\mathsf{id}_1), \mathsf{H}(\mathsf{id}_2))^\alpha = e(\mathsf{usk}_{\mathsf{id}_2}, \mathsf{H}(\mathsf{id}_1))$.

8.2 Security Proof

Theorem 8.1. The construction above is secure as per Definition 2.22 assuming that the DBDH assumption over $\mathcal{G}$ holds (See Assumption 2.6), prIO is secure with parameter κ (as per Definition 5.3) and with respect to a sampler specified in the proof of Lemma 8.2, and PRF is subexponentially secure.

Proof. We prove the security of the scheme via the following hybrids. Let us fix a PPT adversary $\mathcal{A}$. We denote the advantage of the adversary by ϵ and without loss of generality, we assume that $\mathcal{A}$ makes exactly Q key extraction queries. For the sake of contradiction, let us assume that ϵ is non-negligible. In the following, we denote the advantage of the adversary in Game-xx by ϵ_{xx}.

Game-0. This is the security game for ID-NIKE. By definition, we have $\epsilon_0 = \epsilon$.

Game-1. In this game, we sample random function $\mathsf{R} : \mathcal{ID} \to [2Q]$ at the end of the game independently from anything else in the game. We then change the adversary $\mathcal{A}$ so that it outputs a random coin as its guess if the following does *not* hold:

$$\mathsf{R}(\mathsf{id}_1^*) = 1 \ \wedge\ \mathsf{R}(\mathsf{id}_2^*) = 2 \ \wedge\ \mathsf{R}(\mathsf{id}_1) \notin \{1,2\} \ \wedge\ \cdots \ \wedge\ \mathsf{R}(\mathsf{id}_Q) \notin \{1,2\}, \tag{52}$$

where id_1^* and id_2^* are the challenge identities and $\mathsf{id}_1, \ldots, \mathsf{id}_Q$ are the identities for which key extraction queries are made by $\mathcal{A}$. If the abort condition above is not satisfied, the output of $\mathcal{A}$ is unchanged. Since Equation (52) holds with probability $\frac{1}{4Q^2} \cdot \left(1 - \frac{2}{2Q}\right)^Q = \Theta(\epsilon_0/Q^2)$, we have $\epsilon_1 = \Theta(\epsilon_0/Q^2)$.

Game-2. In this game, we choose R at the beginning of the game and stop the game immediately and force $\mathcal{A}$ to output random guess if $\mathcal{A}$ makes a key extraction query or challenge query that violates Equation (52). With this change, the distribution of the output by $\mathcal{A}$ is unchanged and therefore we have $\epsilon_2 = \epsilon_1$.

Game-3. In this game, we replace R with a pseudorandom function $\overline{\mathsf{PRF}} : \{0,1\}^\lambda \times \mathcal{ID} \to [2Q]$. In more detail, the game samples $\overline{\mathsf{sd}} \leftarrow \{0,1\}^\lambda$ and uses $\overline{\mathsf{PRF}}(\overline{\mathsf{sd}}, \cdot)$ in place of the function $R(\cdot)$ when we check the abort condition. It is straightforward to see that $|\epsilon_3 - \epsilon_2| \le \mathsf{negl}(\lambda)$ by the security of PRF.

Game-4. In this game, we change how $\tilde{\mathsf{C}}$ is computed. In more detail, we sample $\beta, \gamma \leftarrow \mathbb{Z}_p$ and $\widetilde{\mathsf{sd}} \leftarrow \{0,1\}^\Lambda$ at the beginning of the game. We then set $F[\overline{\mathsf{sd}}, \widetilde{\mathsf{sd}}, g^\beta, g^\gamma]$ as Figure 4, where we use yet another pseudorandom function $\widetilde{\mathsf{PRF}} : \{0,1\}^\Lambda \times \mathcal{ID} \to \mathcal{R}_{\mathsf{MTP}} \times \mathbb{Z}_p$ in the description of the circuit. In this game, $\tilde{\mathsf{C}}$ is computed as $\tilde{\mathsf{C}} \leftarrow \mathsf{iO}(F[\overline{\mathsf{sd}}, \widetilde{\mathsf{sd}}, g^\beta, g^\gamma])$. As we will show in Lemma 8.2, we have $|\epsilon_4 - \epsilon_3| \le \mathsf{negl}(\lambda)$ by the security of PRIO and by the subexponential security of PRF and $\widetilde{\mathsf{PRF}}$.

Circuit $F[\overline{\mathsf{sd}}, \widetilde{\mathsf{sd}}, g^\beta, g^\gamma]$

- On input id, compute $(R_{\mathsf{id}}, x_{\mathsf{id}}) = \widetilde{\mathsf{PRF}}(\widetilde{\mathsf{sd}}, \mathsf{id})$.
- Compute

$$v_{\mathsf{id}} \stackrel{\text{def}}{=} \begin{cases} \mathsf{MapToPoint}^{-1}(g^{\beta + x_{\mathsf{id}}}; R_{\mathsf{id}}) & \text{if } \overline{\mathsf{PRF}}(\overline{\mathsf{sd}}, \mathsf{id}) = 1, \\ \mathsf{MapToPoint}^{-1}(g^{\gamma + x_{\mathsf{id}}}; R_{\mathsf{id}}) & \text{if } \overline{\mathsf{PRF}}(\overline{\mathsf{sd}}, \mathsf{id}) = 2, \\ \mathsf{MapToPoint}^{-1}(g^{x_{\mathsf{id}}}; R_{\mathsf{id}}) & \text{otherwise.} \end{cases} \tag{53}$$

- Output v_{id}.

Figure 4: Description of the circuit $F[\overline{\mathsf{sd}}, \widetilde{\mathsf{sd}}, g^\beta, g^\gamma]$.

Game-5. In this game, we no longer use the exponents β and γ explicitly in answering the key extraction queries. In more detail, when the adversary makes a key extraction query to id, we proceed as follows. First, we check whether $\overline{\mathsf{PRF}}(\overline{\mathsf{sd}}, \cdot) \notin \{1,2\}$ or not. If it does not hold, we abort the game as specified in Game-2. Otherwise, we compute $(R_{\mathsf{id}}, x_{\mathsf{id}}) = \widetilde{\mathsf{PRF}}(\widetilde{\mathsf{sd}}, \mathsf{id})$ and return $\mathsf{usk}_{\mathsf{id}} \stackrel{\text{def}}{=} (g^{\alpha})^{x_{\mathsf{id}}}$ as the secret key.

We claim that this change does not alter the view of the adversary. To see this, we observe that

$$\mathsf{H}(\mathsf{id}) = \mathsf{MapToPoint}(\tilde{C}(\mathsf{id})) = \mathsf{MapToPoint}\big(F[\overline{\mathsf{sd}}, \widetilde{\mathsf{sd}}, g^{\beta}, g^{\gamma}](\mathsf{id})\big) = \mathsf{MapToPoint}\big(\mathsf{MapToPoint}^{-1}(g^{x_{\mathsf{id}}}; R_{\mathsf{id}})\big) = g^{x_{\mathsf{id}}}$$

holds, where the first equation holds by the definition of H, the second by the correctness of iO, the third by $\overline{\mathsf{PRF}}(\overline{\mathsf{sd}}, \mathsf{id}) \notin \{1,2\}$ and Equation (53), and the fourth by the property of $\mathsf{MapToPoint}^{-1}$. This implies $(g^{\alpha})^{x_{\mathsf{id}}} = \mathsf{H}(\mathsf{id})^{\alpha}$ as desired. We therefore have $\epsilon_5 = \epsilon_4$.

Game-6. In this game, we change the way K^* is computed when $\mathsf{coin} = 0$. In particular, we compute $T = e(g,g)^{\alpha\beta\gamma}$ at the beginning of the game and when the adversary asks for the challenge key, we return

$$K^* = T \cdot e(g^{\alpha}, g^{\beta})^{x_{\mathsf{id}_2^*}} \cdot e(g^{\alpha}, g^{\gamma})^{x_{\mathsf{id}_1^*}} \cdot e(g, g^{\alpha}) \tag{54}$$

to the adversary, where $(R_{\mathsf{id}_i^*}, x_{\mathsf{id}_i^*}, y_{\mathsf{id}_i^*}) = \widetilde{\mathsf{PRF}}(\widetilde{\mathsf{sd}}, \mathsf{id}_i^*)$ for $i = 1, 2$.

We claim that this change does not alter the view of the adversary. To see this, recall that the game aborts unless $\overline{\mathsf{PRF}}(\overline{\mathsf{sd}}, \mathsf{id}_1^*) = 1 \wedge \overline{\mathsf{PRF}}(\overline{\mathsf{sd}}, \mathsf{id}_2^*) = 2$. The first condition implies $\mathsf{H}(\mathsf{id}_1^*) = \mathsf{MapToPoint}(v_{\mathsf{id}_1^*}) = g^{\beta + x_{\mathsf{id}_1^*}}$ by Equation (53) and by the property of $\mathsf{MapToPoint}^{-1}$. Similarly, we have $\mathsf{H}(\mathsf{id}_1^*) = g^{\gamma + x_{\mathsf{id}_2^*}}$. Thus, we have

$$e(\mathsf{H}(\mathsf{id}_1^*), \mathsf{H}(\mathsf{id}_2^*))^{\alpha} = e(g^{\beta + x_{\mathsf{id}_1^*}}, g^{\gamma + x_{\mathsf{id}_2^*}})^{\alpha} = T \cdot e(g^{\alpha}, g^{\beta})^{x_{\mathsf{id}_2^*}} \cdot e(g^{\alpha}, g^{\gamma})^{x_{\mathsf{id}_1^*}} \cdot e(g, g^{\alpha})^{x_{\mathsf{id}_1^*} x_{\mathsf{id}_2^*}}$$

as desired. We therefore have $\epsilon_6 = \epsilon_5$.

Note that in this game, we do not need to know α any more to efficiently simulate the game. Rather, it suffices to know $g^{\alpha}, g^{\beta}, g^{\gamma}$, and $T = e(g,g)^{\alpha\beta\gamma}$.

Game-7. In this game, we replace T with random $T \leftarrow \mathbb{G}_T$. We claim that the adversary cannot distinguish this game from the previous game. This can be seen by straightforward reduction to the DBDH assumption, where we simulate Game-6 if $T = e(g,g)^{\alpha\beta\gamma}$ and Game-7 if $T \leftarrow \mathbb{G}_T$. Indeed, we only need to know g^{β} and g^{γ} for simulating mpk, g^{α} for answering the key extraction queries, and g^{α}, g^{β}, g^{γ}, and T for simulating the challenge key. We therefore have $|\epsilon_6 - \epsilon_7| \leq \mathsf{negl}(\lambda)$.

Game-8. In this game, we change the challenge shared key K^* to be random group element in $\mathbb{G}_T$, regardless of whether $\mathsf{coin} = 0$ or $\mathsf{coin} = 1$. We claim that this is a conceptual change. This can be easily seen by observing that K^* computed as Equation (54) is distributed uniformly at random over $\mathbb{G}_T$ if so is T. Thefefore, we have $\epsilon_8 = \epsilon_7$.

Clearly, we have $\epsilon_8 = 0$, since no information of coin is leaked to the adversary in Game-8. However, from the above discussion, we have $0 = \epsilon_8 \geq \Theta(\epsilon)/Q^2 - \mathsf{negl}(\lambda)$ by the triangle inequality. This contradicts our assumption that ϵ is non-negligible. □

To complete the proof of Theorem 8.1, it remains to prove Lemma 8.2.

Lemma 8.2. Assuming that iO is a secure PRIO scheme as per Definition 5.3 with the parameter κ and PRF and $\widetilde{\mathsf{PRF}}$ are subexponentially secure, we have $|\epsilon_3 - \epsilon_4| \leq \mathsf{negl}(\lambda)$.

Proof. To prove the lemma, it suffices to show $(\mathsf{iO}(\mathsf{PRF}(\mathsf{sd}, \cdot)), \overline{\mathsf{sd}}, \mathcal{G}) \approx_c (\mathsf{iO}(F[\overline{\mathsf{sd}}, \widetilde{\mathsf{sd}}, g^{\beta}, g^{\gamma}]), \overline{\mathsf{sd}}, \mathcal{G})$, since we can simulate the game in a way that we simulate Game-3 if the given terms come from the left distribution and Game-4 otherwise. Any adversary distingushing the games can be turned into an adversary that distinguishes the distributions.

By considering a sampler Samp that takes as input 1^λ and outputs the auxiliary information $\mathsf{aux} \stackrel{\text{def}}{=} (\overline{\mathsf{sd}}, \mathcal{G})$ and circuits $C_0 \stackrel{\text{def}}{=} \mathsf{PRF}(\mathsf{sd}, \cdot)$ and $C_1 \stackrel{\text{def}}{=} F[\overline{\mathsf{sd}}, \widetilde{\mathsf{sd}}, g^\beta, g^\gamma]$ and invoking the security of PRIO, we can see that it suffices to prove

$$(1^\kappa, \mathsf{aux}, \{\mathsf{PRF}(\mathsf{sd}, \mathsf{id})\}_{\mathsf{id} \in \mathcal{ID}}) \approx_c (1^\kappa, \mathsf{aux}, \{\delta_{\mathsf{id}} \leftarrow \mathcal{D}_{\mathsf{MTP}}\}_{\mathsf{id} \in \mathcal{ID}}) \approx_c \left(1^\kappa, \mathsf{aux}, \{v_{\mathsf{id}} = F[\overline{\mathsf{sd}}, \widetilde{\mathsf{sd}}, g^\beta, g^\gamma]\}_{\mathsf{id} \in \mathcal{ID}}\right). \tag{55}$$

The former indistinguishability holds by the subexponential security of PRF. In particular, by our choice of the parameter $2^{\Lambda^\delta} > \kappa^{\omega(1)}$, we can conclude that the adversary with running time $\mathrm{poly}(\kappa)$ cannot distinguish the distributions with advantage more than $\mathsf{negl}(\kappa)$. To prove the latter indistinguishability, we further consider the following distributions. In the following, we do not change the distribution of $\mathsf{aux} = (\overline{\mathsf{sd}}, \mathcal{G})$ and only focus on the distribution of $\{v_{\mathsf{id}}\}_{\mathsf{id}}$.

- This is the rightmost distribution of Equation (55), where v_{id} is computed as Equation (53) for $(R_{\mathsf{id}}, x_{\mathsf{id}}) = \widetilde{\mathsf{PRF}}(\mathsf{id})$.
- This is the same as the previous distribution except that $(R_{\mathsf{id}}, x_{\mathsf{id}})$ that is used for computing v_{id} is chosen uniformly at random from their respective domains.
- $\{v_{\mathsf{id}}\}_{\mathsf{id}}$, where $v_{\mathsf{id}} \leftarrow \mathcal{D}_{\mathsf{MTP}}$. Observe that this is the middle distribution in Equation (55).

It suffices to prove that the first and the third distributions are indistinguishable given $(1^\kappa, \mathsf{aux})$. We first observe that the first and the second distributions are computationally indistinguishable by the subexponential security of $\widetilde{\mathsf{PRF}}$ and our choice of parameter Λ, where we have $2^{\Lambda^\delta} > \kappa^{\omega(1)}$. To conclude the proof, we show that the second and the third distributions are actually the same distribution. In particular, we show that v_{id} is uniformly distributed over $\mathcal{D}_{\mathsf{MTP}}$ for each id in the second distribution. Here, we consider the case of $\overline{\mathsf{PRF}}(\overline{\mathsf{sd}}, \mathsf{id}) = 1$. The other cases follow similarly. By the assumption, we know that $g^{\beta + x_{\mathsf{id}}}$ is distributed uniformly at random over $\mathbb{G}$. Then, by the property of $\mathsf{MapToPoint}^{-1}$ and by the fact that R_{id} is random, v_{id} is uniformly distributed over $\mathcal{D}_{\mathsf{MTP}}$ as desired. □

Theorem 8.3. Let $\kappa = \lambda^{n^2 \log \lambda}$. Assuming non-uniform κ-evasive LWE (Assumption 2.4), subexponentially secure PRF against non-uniform adversary, non-uniform sub-exponential LWE, and the DBDH assumption, there exists a secure ID-NIKE scheme.

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 65.)

[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 65.)

[ABG19] Michel Abdalla, Fabrice Benhamouda, and Romain Gay. From single-input to multi-client inner-product functional encryption. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part III*, volume 11923 of *LNCS*, pages 552–582. Springer, Heidelberg, December 2019. (Cited on page 3.)

[ABKW19] Michel Abdalla, Fabrice Benhamouda, Markulf Kohlweiss, and Hendrik Waldner. Decentralizing inner-product functional encryption. In Dongdai Lin and Kazue Sako, editors, *PKC 2019, Part II*, volume 11443 of *LNCS*, pages 128–157. Springer, Heidelberg, April 2019. (Cited on page 3.)

[ACF$^+$18] Michel Abdalla, Dario Catalano, Dario Fiore, Romain Gay, and Bogdan Ursu. Multi-input functional encryption for inner products: Function-hiding realizations and constructions without pairings. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 597–627. Springer, Heidelberg, August 2018. (Cited on page 3.)

[Agr19] Shweta Agrawal. Indistinguishability obfuscation without multilinear maps: New techniques for bootstrapping and instantiation. In *Eurocrypt*, 2019. (Cited on page 3.)

[AGRW17] Michel Abdalla, Romain Gay, Mariana Raykova, and Hoeteck Wee. Multi-input inner-product functional encryption from pairings. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 601–626. Springer, Heidelberg, April / May 2017. (Cited on page 3.)

[AGT21a] Shweta Agrawal, Rishab Goyal, and Junichi Tomida. Multi-input quadratic functional encryption from pairings. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part IV*, volume 12828 of *LNCS*, pages 208–238, Virtual Event, August 2021. Springer, Heidelberg. (Cited on page 3.)

[AGT21b] Shweta Agrawal, Rishab Goyal, and Junichi Tomida. Multi-party functional encryption. In *TCC*, 2021. (Cited on page 3.)

[AGT22] Shweta Agrawal, Rishab Goyal, and Junichi Tomida. Multi-input quadratic functional encryption: Stronger security, broader functionality. In *TCC*, 2022. (Cited on page 3.)

[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 3, 4, 8, 9, 13, 42.)

[AKY24a] Shweta Agrawal, Simran Kumari, and Shota Yamada. Attribute Based Encryption for Turing Machines from Lattices. In *Crypto*, 2024. (Cited on page 4.)

[AKY24b] Shweta Agrawal, Simran Kumari, and Shota Yamada. Compact Pseudorandom Functional Encryption from Evasive LWE , 2024. (Cited on page 5, 6, 7, 12, 14, 16, 20, 21, 44, 46, 67, 71, 76.)

[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.)

[AMYY25] Shweta Agrawal, Anuja Modi, Anshu Yadav, and Shota Yamada. Evasive lwe: Attacks, variants & obfustopia, 2025. (Cited on page 4, 6, 17, 21, 26, 42.)

[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 3.)

[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 4, 5, 14, 17, 37, 49.)

[ATY23] Shweta Agrawal, Junichi Tomida, and Anshu Yadav. Attribute-based multi-input FE (and more) for attribute-weighted sums. In *CRYPTO 2023, Part IV*, LNCS, pages 464–497. Springer, Heidelberg, August 2023. (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 14, 23.)

[BDGM20] Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Factoring and pairings are not necessary for io: Circular-secure lwe suffices. *Cryptology ePrint Archive*, 2020. (Cited on page 3.)

[BDGM23] Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Candidate io from homomorphic encryption schemes. *Journal of Cryptology*, 36(3):27, 2023. (Cited on page 3.)

[BDJ+24] Pedro Branco, Nico Döttling, Abhishek Jain, Giulio Malavolta, Surya Mathialagan, Spencer Peters, and Vinod Vaikuntanathan. Pseudorandom obfuscation and applications. Cryptology ePrint Archive, Paper 2024/1742, 2024. (Cited on page 6, 26, 42.)

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer, Heidelberg, August 2001. (Cited on page 3.)

[BFM14] Christina Brzuska, Pooya Farshim, and Arno Mittelbach. Indistinguishability obfuscation and UCEs: The case of computationally unpredictable sources. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 188–205. Springer, Heidelberg, August 2014. (Cited on page 3.)

[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, 37.)

[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 6.)

[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 19.)

[BGV14] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3):1–36, 2014. (Cited on page 3.)

[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 65, 66.)

[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 5, 15, 16, 53, 54.)

[BPR15] Nir Bitansky, Omer Paneth, and Alon Rosen. On the cryptographic hardness of finding a Nash equilibrium. In Venkatesan Guruswami, editor, *56th FOCS*, pages 1480–1498. IEEE Computer Society Press, October 2015. (Cited on page 3.)

[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 5, 15.)

[Bra12] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In *Annual Cryptology Conference*, pages 868–886. Springer, 2012. (Cited on page 3.)

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 253–273. Springer, Heidelberg, March 2011. (Cited on page 3.)

[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 67.)

[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 6.)

[BV14] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. *SIAM Journal on computing*, 43(2):831–871, 2014. (Cited on page 3.)

[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 3, 4, 13.)

[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 19.)

[CDG+18] Jérémy Chotard, Edouard Dufour Sans, Romain Gay, Duong Hieu Phan, and David Pointcheval. Decentralized multi-client functional encryption for inner product. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 703–732. Springer, Heidelberg, December 2018. (Cited on page 3.)

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, 2004. (Cited on page 6.)

[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 65.)

[CHN+16] Aloni Cohen, Justin Holmgren, Ryo Nishimaki, Vinod Vaikuntanathan, and Daniel Wichs. Watermarking cryptographic capabilities. In Daniel Wichs and Yishay Mansour, editors, *48th ACM STOC*, pages 1115–1127. ACM Press, June 2016. (Cited on page 3.)

[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 5, 15, 16, 53, 54.)

[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 6.)

[DJM+25] Nico Döttling, Abhishek Jain, Giulio Malavolta, Surya Mathialagan, and Vinod Vaikuntanathan. Simple and general counterexamples for private-coin evasive lwe. *Cryptology ePrint Archive*, 2025. (Cited on page 6.)

[DOT18] Pratish Datta, Tatsuaki Okamoto, and Junichi Tomida. Full-hiding (unbounded) multi-input inner product functional encryption from the k-Linear assumption. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part II*, volume 10770 of *LNCS*, pages 245–277. Springer, Heidelberg, March 2018. (Cited on page 3.)

[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 3.)

[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 14.)

[FHPS13] Eduarda S. V. Freire, Dennis Hofheinz, Kenneth G. Paterson, and Christoph Striecks. Programmable hash functions in the multilinear setting. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 513–530. Springer, Heidelberg, August 2013. (Cited on page 15, 24, 54.)

[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 3.)

[GGH+13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*, pages 40–49. IEEE Computer Society Press, October 2013. (Cited on page 3.)

[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 3.)

[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 19.)

[GJLS21] Romain Gay, Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from simple-to-state hard problems: New assumptions, new techniques, and simplification. In *EUROCRYPT*, 2021. (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 23.)

[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 3.)

[GPS16] Sanjam Garg, Omkant Pandey, and Akshayaram Srinivasan. Revisiting the cryptographic hardness of finding a nash equilibrium. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 579–604. Springer, Heidelberg, August 2016. (Cited on page 3.)

[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 65.)

[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 3, 66.)

[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 23.)

[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 14, 15, 37.)

[HJK+16] Dennis Hofheinz, Tibor Jager, Dakshita Khurana, Amit Sahai, Brent Waters, and Mark Zhandry. How to generate and use universal samplers. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 715–744. Springer, Heidelberg, December 2016. (Cited on page 3.)

[HJL21] Sam Hopkins, Aayush Jain, and Huijia Lin. Counterexamples to new circular security assumptions underlying io. In *Advances in Cryptology–CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part II 41*, pages 673–700. Springer, 2021. (Cited on page 3.)

[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 6.)

[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 4, 66.)

[HSW13] Susan Hohenberger, Amit Sahai, and Brent Waters. Full domain hash from (leveled) multilinear maps and identity-based aggregate signatures. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 494–512. Springer, Heidelberg, August 2013. (Cited on page 3.)

[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 5, 15, 52, 53.)

[HY17] Pavel Hubácek and Eylon Yogev. Hardness of continuous local search: Query complexity and cryptographic lower bounds. In Philip N. Klein, editor, *28th SODA*, pages 1352–1371. ACM-SIAM, January 2017. (Cited on page 3.)

[JLLS23] Aayush Jain, Huijia Lin, Paul Lou, and Amit Sahai. Polynomial-time cryptanalysis of the subspace flooding assumption for post-quantum i o. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 205–235. 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 3.)

[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 3.)

[JLS22] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from lpn over f p, dlin, and prgs in nc 0. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 670–699. Springer, 2022. (Cited on page 3.)

[KLW15] Venkata Koppula, Allison Bishop Lewko, and Brent Waters. Indistinguishability obfuscation for Turing machines with unbounded memory. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 419–428. ACM Press, June 2015. (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 19.)

[KS17] Ilan Komargodski and Gil Segev. From minicrypt to obfustopia via private-key functional encryption. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 122–151. Springer, Heidelberg, April / May 2017. (Cited on page 3.)

[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.)

[LT19] Benoît Libert and Radu Titiu. Multi-client functional encryption for linear functions in the standard model from LWE. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part III*, volume 11923 of *LNCS*, pages 520–551. Springer, Heidelberg, December 2019. (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 65.)

[MPV24a] Surya Mathialagan, Spencer Peters, and Vinod Vaikuntanathan. Adaptively sound zero-knowledge snarks for up. In *Crypto*. Springer, 2024. (Cited on page 5, 13.)

[MPV24b] Surya Mathialagan, Spencer Peters, and Vinod Vaikuntanathan. Using prtt obfuscation for unlevelled fhe. Personal Communication, 2024. (Cited on page 5.)

[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 66.)

[RVV24] Seyoon Ragavan, Neekon Vafa, and Vinod Vaikuntanathan. Indistinguishability obfuscation from bilinear maps and lpn variants. *Cryptology ePrint Archive*, 2024. (Cited on page 3.)

[SOK00] Ryuichi Sakai, Kiyoshi Ohgishi, and Masao Kasahara. Cryptosystems based on pairing. *SCIS 2000*, 2000. (Cited on page 15, 54.)

[SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, 2005. (Cited on page 3.)

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *46th ACM STOC*, pages 475–484. ACM Press, May / June 2014. (Cited on page 3.)

[Tom19] Junichi Tomida. Tightly secure inner product functional encryption: Multi-input and function-hiding constructions. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part III*, volume 11923 of *LNCS*, pages 459–488. Springer, Heidelberg, December 2019. (Cited on page 3.)

[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, 6.)

[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, 10, 11, 12, 34, 35.)

[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 6.)

[Wee21] Hoeteck Wee. Broadcast encryption with size $N^{1/3}$ and more from k-lin. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part IV*, volume 12828 of *LNCS*, pages 155–178, Virtual Event, August 2021. Springer, Heidelberg. (Cited on page 5.)

[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 4, 6, 17.)

[WW21] Hoeteck Wee and Daniel Wichs. Candidate obfuscation via oblivious lwe sampling. In *Advances in Cryptology–EUROCRYPT 2021: 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17–21, 2021, Proceedings, Part III*, pages 127–156. Springer, 2021. (Cited on page 3.)

[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 65.)

[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 23, 37.)

A Additional Preliminaries

A.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 $n = \text{poly}(\lambda)$ 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

$$\mathbf{g} = (2^0, 2^1, \ldots, 2^{\frac{m}{(n+1)}-1})^\intercal \ , \ \ \mathbf{G} = \mathbf{I}_{(n+1)} \otimes \mathbf{g}^\intercal$$

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 $(\mathsf{bits}(\mathbf{p}[1]), \ldots, \mathsf{bits}(\mathbf{p}[n+1]))^\intercal$, where $\mathsf{bits}(\mathbf{p}[i])$ are $m/(n+1)$ bits for each $i \in [n+1]$. 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 $\text{poly}(n, m, m', \log q)$ 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 A.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 A.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 A.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 A.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 $q^n \sqrt{2^{1-m}}$.

Additional Hardness Assumptions.

Assumption A.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

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

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 $\mathrm{LWE}(n, m, q, \chi)$ problem is as hard as solving worst case lattice problems such as gapSVP and SIVP with approximation factor $\mathrm{poly}(n) \cdot (q/\gamma)$ for some $\mathrm{poly}(n)$. 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 $\mathrm{LWE}(n, m, q, \chi)$ with noise-to-modulus ratio 2^{-n^ϵ} is likely to be (subexponentially) hard for some constant ϵ.

A.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 A.6. 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 A.7. (homomorphic evaluation for vector-valued functions [HLL23]) There is an efficient algorithm

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

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

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

A.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
$$\mathbf{s}^\intercal(\mathbf{A}_{\mathsf{att}} - (1, \mathbf{x}^\intercal) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{att}}^\intercal,$$
where $\mathbf{s}^\intercal = (\bar{\mathbf{s}}^\intercal, -1)$ with $\bar{\mathbf{s}} \in \mathbb{Z}^n$ and $\mathbf{e}_{\mathsf{att}}^\intercal \in \mathbb{Z}^{(L+1)m}$.
- 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 $\|\mathbf{H}_C^\intercal\|, \left\|\mathbf{H}_{C,\mathbf{x}}^\intercal\right\| \leq (m+2)^d \lceil \log q \rceil$.
  - The matrix encoding homomorphism is $(\mathbf{A}_{\mathsf{att}} - (1, \mathbf{x}^\intercal) \otimes \mathbf{G})\mathbf{H}_{C,\mathbf{x}} = \mathbf{A}_{\mathsf{att}}\mathbf{H}_C - C(\mathbf{x})$.

Dual-Use Technique and Extension. In [BTVW17], the attribute encoded with secret $\mathbf{s}^\intercal$ is FHE ciphertexts under key $\mathbf{s}^\intercal$ (the same, "dual-use") and the circuit being $\mathsf{MEvalCX}$'ed is some HEval_C. This leads to automatic decryption. Let C be a vector-valued circuit, with co-domain $\mathbb{Z}_q^{1\times m'}$, then VEval_C is $\mathbb{Z}_q^{(n+1)\times m'}$-valued and

$$\begin{aligned}
&(\mathbf{s}^\intercal(\mathbf{A}_{\mathsf{att}} - (1, \mathsf{bits}(\mathbf{X})) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{att}}^\intercal) \cdot \mathbf{H}_{\mathsf{VEval}_C, \mathbf{X}} \\
&= \mathbf{s}^\intercal \mathbf{A}_{\mathsf{att}} \mathsf{VEval}_C - \mathbf{s}^\intercal \mathsf{VEval}_C(\mathbf{X}) + (\mathbf{e}')^\intercal \quad (\mathsf{MEvalCX}) \\
&= \mathbf{s}^\intercal \mathbf{A}_{\mathsf{att}} \mathsf{VEval}_C - C(\mathbf{x}) + (\mathbf{e}'')^\intercal. \qquad (\mathsf{VEval\ decryption}\ )
\end{aligned}$$

B Pseudorandom FE with Stronger Security

In this section, we provide the description of the pseudorandom FE construction proposed in [AKY24b]. For the reference for the readers, we provide the full description of the construction and the proof for prCT security (Definition 2.12) below. Furthermore, we explain how to extend their construction to achieve strengthened security notion for pseudorandom FE that we call non-uniform κ-prCT security (Definition 2.15), which is required for many of our applications.

B.1 Construction

The construction below is exactly the same as [AKY24b]. The construction supports the function family $\mathcal{F}_{L(\lambda),\ell(\lambda),\mathsf{dep}(\lambda)} = \{f : \{0,1\}^L \to \{0,1\}^\ell\}$, 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 $\mathsf{prm} = (1^{L(\lambda)}, 1^{\ell(\lambda)}, 1^{\mathsf{dep}(\lambda)})$.

Ingredients. Below, we list the ingredients for our construction.

1. 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 exponentially small.

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

- Set $\mathrm{L}_{\mathrm{X}} = m\mathrm{L}(n+1)\lceil \log q \rceil$, 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 $w \in O(\log q)$.
- Sample a constant $M \in \mathbb{Z}$ such that M divides 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 $\mathbf{r} \leftarrow \{0,1\}^\lambda$.
- Define function $\mathrm{F}[f, \mathbf{r}]$ with $f, \mathbf{r}$ hardwired as follows:
  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 $f_{\mathsf{low}}(\mathbf{x}, \mathsf{sd}) \in [0, M-1]^\ell$.[20] 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 A.7, 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 $\mathbf{s} = (\bar{\mathbf{s}}^\top, -1)^\top$.
- Sample $\mathbf{e}_{\mathbf{B}} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma_{\mathbf{B}}}^{mw}$ and compute $\mathbf{c}_{\mathbf{B}}^\top := \mathbf{s}^\top\mathbf{B} + \mathbf{e}_{\mathbf{B}}^\top$.
- 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.
$$\mathbf{A}_{\mathsf{fhe}} := \begin{pmatrix} \bar{\mathbf{A}}_{\mathsf{fhe}} \\ \bar{\mathbf{s}}^\top\bar{\mathbf{A}}_{\mathsf{fhe}} + \mathbf{e}_{\mathsf{fhe}}^\top \end{pmatrix}, \quad \mathbf{X} = \mathbf{A}_{\mathsf{fhe}}\mathbf{R} - (\mathbf{x}, \mathsf{sd}) \otimes \mathbf{G} \in \mathbb{Z}_q^{(n+1)\times m(\lambda+\mathrm{L})}.$$
  Let $\mathrm{L}_{\mathrm{X}} = m(\lambda+\mathrm{L})(n+1)\lceil \log q \rceil$ be the bit length of $\mathbf{X}$.
- Compute a BGG^+ encoding as follows.
$$\mathbf{e}_{\mathsf{att}} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}^{(\mathrm{L}_{\mathrm{X}}+1)m}, \qquad \mathbf{c}_{\mathsf{att}}^\top := \mathbf{s}^\top(\mathbf{A}_{\mathsf{att}} - \mathsf{bits}(1, \mathbf{X}) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{att}}^\top.$$
- 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 the key generation algorithm.
- Compute $\mathbf{z} := \mathbf{c}_{\mathbf{B}}^\top \cdot \mathbf{K} - \left(M \cdot \left\lfloor \frac{\mathbf{c}_{\mathsf{att}}^\top \cdot \mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}}}{M} \right\rfloor + \left\lfloor \frac{\mathbf{c}_{\mathsf{att}}^\top \cdot \mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{low}}}}{M} \right\rfloor \right)$.
- 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 $\mathbf{y} = (y_1, \ldots, y_\ell)$.

[20] Please note that f_{high} and f_{low} hardwire the information about f and r even though it is not written explicitly.

Parameters. We set our parameters as follows.

$$\beta = 2^{O(\mathsf{dep}\cdot\log\lambda)},\ q = 2^{12\lambda}\beta,\ M = 2^{4\lambda}\beta,\ n = \mathrm{poly}(\lambda, \mathsf{dep}),\ m = O(n\log q),\ \tau = O\left(\sqrt{(n+1)\log q}\right)$$
$$B = 2^{10\lambda}\beta,\ \sigma_{\mathbf{s}} = \sigma = 2^{2\lambda},\ \sigma_{\mathbf{B}} = 2^{9\lambda}\beta,\ \sigma_1 = 2^{8\lambda+O(1)}\beta/\mathrm{poly}(\lambda). \quad (56)$$

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 analyse the correctness of our scheme below.

– First, we note that

$$\begin{aligned}
\mathbf{c}_{\mathsf{att}}^{\mathsf{T}}\cdot\mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}} &= (\mathbf{s}^{\mathsf{T}}(\mathbf{A}_{\mathsf{att}} - \mathsf{bits}(1,\mathbf{X})\otimes\mathbf{G}) + \mathbf{e}_{\mathsf{att}}^{\mathsf{T}})\mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}}\\
&= \mathbf{s}^{\mathsf{T}}\mathbf{A}_{\mathsf{att}}\mathbf{H}_{\mathbf{A}_{\mathsf{att}}}^{\mathrm{F}_{\mathsf{high}}} - \mathbf{s}^{\mathsf{T}}\mathsf{VEval}_{\mathsf{high}}(\mathsf{bits}(\mathbf{X})) + \mathbf{e}_{\mathsf{att}}^{\mathsf{T}}\mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}}\\
&= \mathbf{s}^{\mathsf{T}}\mathbf{A}_{\mathsf{high}} - \mathrm{F}_{\mathsf{high}}(\mathbf{x},\mathsf{sd}) + \mathbf{e}_{\mathsf{fhe}}^{\mathsf{T}}\mathbf{R}_{\mathsf{high}} + \mathbf{e}_{\mathsf{att}}^{\mathsf{T}}\mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}}\\
\implies \left\lfloor\frac{\mathbf{c}_{\mathsf{att}}^{\mathsf{T}}\cdot\mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}}}{M}\right\rfloor &= \left\lfloor\frac{\mathbf{s}^{\mathsf{T}}\mathbf{A}_{\mathsf{high}} - M\cdot f_{\mathsf{high}}(\mathbf{x},\mathsf{sd}) + \mathbf{e}_{\mathsf{fhe}}^{\mathsf{T}}\mathbf{R}_{\mathsf{high}} + \mathbf{e}_{\mathsf{att}}^{\mathsf{T}}\mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}}}{M}\right\rfloor\\
&= \left\lfloor\frac{\mathbf{s}^{\mathsf{T}}\mathbf{A}_{\mathsf{high}} + \mathbf{e}_{\mathsf{fhe}}^{\mathsf{T}}\mathbf{R}_{\mathsf{high}} + \mathbf{e}_{\mathsf{att}}^{\mathsf{T}}\mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}}}{M}\right\rfloor - f_{\mathsf{high}}(\mathbf{x},\mathsf{sd}) \qquad (57)
\end{aligned}$$

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 A.7, we have

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

and using the depth bound from Appendix A.3,

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

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 (57),

$$\begin{aligned}\left\lfloor\frac{\mathbf{c}_{\mathsf{att}}^{\mathsf{T}}\cdot\mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}}}{M}\right\rfloor &= \left\lfloor\frac{\mathbf{s}^{\mathsf{T}}\mathbf{A}_{\mathsf{high}}}{M}\right\rfloor - f_{\mathsf{high}}(\mathbf{x},\mathsf{sd}) + \mathsf{err}_{\mathsf{high}}\\ &= \mathbf{s}^{\mathsf{T}}\left\lfloor\frac{\mathbf{A}_{\mathsf{high}}}{M}\right\rfloor + \mathbf{e}_{\mathbf{s},\mathsf{high}}^{\mathsf{T}} - f_{\mathsf{high}}(\mathbf{x},\mathsf{sd}) + \mathsf{err}_{\mathsf{high}}^{\mathsf{T}} \qquad (58)\end{aligned}$$

where $\mathsf{err}_{\mathsf{high}}^{\mathsf{T}} \in \{0,1\}^{\ell}$, is the rounding error which is 1 if $\mathbf{s}^{\mathsf{T}}\mathbf{A}_{\mathsf{high}}$ sits around the boundary of multiple of M and the $\left\|\mathbf{e}_{\mathsf{fhe}}^{\mathsf{T}}\mathbf{R}_{\mathsf{high}} + \mathbf{e}_{\mathsf{att}}^{\mathsf{T}}\mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}}\right\|$ exceeds the distance from that boundary 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 (58), we get

$$\left\lfloor \frac{\mathbf{c}_{\mathsf{att}}^{\mathsf{T}} \cdot \mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{low}}}}{M} \right\rceil = \mathbf{s}^{\mathsf{T}} \left\lfloor \frac{\mathbf{A}_{\mathsf{low}}}{M} \right\rceil + \mathbf{e}_{\mathbf{s},\mathsf{low}}^{\mathsf{T}} - f_{\mathsf{low}}(\mathbf{x}, \mathsf{sd}) + \mathsf{err}_{\mathsf{low}}^{\mathsf{T}} \tag{59}$$

where $\mathsf{err}_{\mathsf{low}}^{\mathsf{T}} \in \{0,1\}^{\ell}$ and $\|\mathbf{e}_{\mathbf{s},\mathsf{low}}\| \leq (n+1) \cdot \|\mathbf{s}\|$. Using Equation (58) and Equation (59), we get

$$\begin{aligned}
& M \cdot \left\lfloor \frac{\mathbf{c}_{\mathsf{att}}^{\mathsf{T}} \cdot \mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}}}{M} \right\rceil + \left\lfloor \frac{\mathbf{c}_{\mathsf{att}}^{\mathsf{T}} \cdot \mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{low}}}}{M} \right\rceil \\
&= \mathbf{s}^{\mathsf{T}} \left(M \cdot \left\lfloor \frac{\mathbf{A}_{\mathsf{high}}}{M} \right\rceil + \left\lfloor \frac{\mathbf{A}_{\mathsf{low}}}{M} \right\rceil \right) - (M \cdot f_{\mathsf{high}}(\mathbf{x}, \mathsf{sd}) + f_{\mathsf{low}}(\mathbf{x}, \mathsf{sd})) + \mathsf{err} \\
&= \mathbf{s}^{\mathsf{T}} \left(M \cdot \left\lfloor \frac{\mathbf{A}_{\mathsf{high}}}{M} \right\rceil + \left\lfloor \frac{\mathbf{A}_{\mathsf{low}}}{M} \right\rceil \right) - \mathrm{F}[f, \mathbf{r}](\mathbf{x}, \mathsf{sd}) + \mathsf{err}
\end{aligned} \tag{60}$$

where

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

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 and

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

– Next, we note that

$$\mathbf{c}_{\mathbf{B}}^{\mathsf{T}} \cdot \mathbf{K} = \mathbf{s}^{\mathsf{T}} \left(M \cdot \left\lfloor \frac{\mathbf{A}_{\mathsf{high}}}{M} \right\rceil + \left\lfloor \frac{\mathbf{A}_{\mathsf{low}}}{M} \right\rceil \right) + \mathbf{c}_{\mathbf{B}}^{\mathsf{T}} \cdot \mathbf{K}. \tag{61}$$

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

– Using Equations (60) and (61), 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.

B.2 Proof for prCT Security

The proof below is adapted from [AKY24b].

Theorem B.1. Let $\mathcal{SC}_{\mathsf{prFE}}$ be a sampler class for prFE. Assuming LWE (Assumption A.5) and private coin Evasive LWE (Assumption 2.1) 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 5, our prFE scheme satisfies prCT security with respect to $\mathcal{SC}_{\mathsf{prFE}}$ as defined in Definition 2.12.

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 2.12, we show

$$\begin{pmatrix} \mathsf{mpk} = (\mathbf{A}_{\mathsf{att}}, \mathbf{B}, M),\ \mathsf{aux}_{\mathcal{A}},\ \ \mathbf{C}_{\mathbf{B}} = \mathbf{S}\mathbf{B} + \mathbf{E}_{\mathbf{B}}, \\ \{\mathbf{X}_j = \mathbf{A}_{\mathsf{fhe},j}\mathbf{R}_j - (\mathbf{x}_j, \mathsf{sd}_j) \otimes \mathbf{G}\}_{j \in [Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}_{\mathsf{att},j}^{\mathsf{T}} = \mathbf{s}_j^{\mathsf{T}}(\mathbf{A}_{\mathsf{att}} - \mathsf{bits}(1, \mathbf{X}_j) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{att},j}^{\mathsf{T}}\}_{j \in [Q_{\mathsf{msg}}]}, \\ \{\mathbf{K}_k, \mathbf{r}_k\}_{k \in [Q_{\mathsf{key}}]} \end{pmatrix} \approx_c \begin{pmatrix} \mathsf{mpk} = (\mathbf{A}_{\mathsf{att}}, \mathbf{B}, M),\ \mathsf{aux}_{\mathcal{A}},\ \ \mathbf{C}_{\mathbf{B}} \leftarrow \mathbb{Z}_q^{Q_{\mathsf{msg}} \times mw}, \\ \{\mathbf{X}_j \leftarrow \mathbb{Z}_q^{(n+1) \times m(\lambda + \mathrm{L})}\}_{j \in [Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}_{\mathsf{att},j} \leftarrow \mathbb{Z}_q^{(\mathrm{L_X}+1)m}\}_{j \in [Q_{\mathsf{msg}}]}, \\ \{\mathbf{K}_k, \mathbf{r}_k\}_{k \in [Q_{\mathsf{key}}]} \end{pmatrix} \tag{62}$$

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

$$(1^\lambda,\ \mathsf{aux}_{\mathcal{A}},\ \{f_k,\ f_k(x_j)\}_{j \in [Q_{\mathsf{msg}}], k \in [Q_{\mathsf{key}}]}) \approx_c (1^\lambda,\ \mathsf{aux}_{\mathcal{A}},\ \{f_k,\ \Delta_{j,k} \leftarrow \{0,1\}^\ell\}_{j \in [Q_{\mathsf{msg}}], k \in [Q_{\mathsf{key}}]}). \tag{63}$$

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 $\mathsf{coins}_{\mathsf{priv}}^{\mathsf{Samp}_{\mathsf{evs}}} = \{\mathsf{sd}_j, \mathbf{R}_j, \mathbf{e}_{\mathsf{att},j}, \mathbf{A}_{\mathsf{fhe},j}\}_{j \in [Q_{\mathsf{msg}}]}$, defined as follows.

By Lemma 2.3, to prove Equation (62) assuming evasive LWE, it suffices to show

$$\begin{pmatrix} \mathsf{aux}_2,\ \ \mathbf{B},\ \ \mathbf{C}_{\mathbf{B}} = \mathbf{S}\mathbf{B} + \mathbf{E}_{\mathbf{B}}, \\ \{\mathbf{X}_j = \mathbf{A}_{\mathsf{fhe},j}\mathbf{R}_j - (\mathbf{x}_j, \mathsf{sd}_j) \otimes \mathbf{G}\}_{j \in [Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}_{\mathsf{att},j}^{\mathsf{T}} = \mathbf{s}_j^{\mathsf{T}}(\mathbf{A}_{\mathsf{att}} - \mathsf{bits}(1, \mathbf{X}_j) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{att},j}^{\mathsf{T}}\}_{j \in [Q_{\mathsf{msg}}]}, \\ \mathbf{C}_{\mathbf{P}} = \mathbf{S}\mathbf{P} + \mathbf{E}_{\mathbf{P}} \end{pmatrix} \approx_c \begin{pmatrix} \mathsf{aux}_2,\ \ \mathbf{B},\ \ \mathbf{C}_{\mathbf{B}} \leftarrow \mathbb{Z}_q^{Q_{\mathsf{msg}} \times mw}, \\ \{\mathbf{X}_j \leftarrow \mathbb{Z}_q^{(n+1) \times m(\lambda+\mathrm{L})}\}_{j \in [Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}_{\mathsf{att},j} \leftarrow \mathbb{Z}_q^{(\mathrm{L_X}+1)m}\}_{j \in [Q_{\mathsf{msg}}]}, \\ \mathbf{C}_{\mathbf{P}} \leftarrow \mathbb{Z}_q^{Q_{\mathsf{msg}} \times \ell \cdot Q_{\mathsf{key}}} \end{pmatrix} \tag{64}$$

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

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 (56)[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 $\bar{\mathbf{s}}_j \leftarrow \mathcal{D}^n_{\mathbb{Z},\sigma_{\mathsf{s}}}$, $\mathbf{e}_{\mathsf{att},j} \leftarrow \mathcal{D}^{(\mathrm{L_X}+1)m}_{\mathbb{Z},\sigma}$, 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

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

$$\mathsf{aux}_1 = \left(\{\mathbf{X}_j = \mathbf{A}_{\mathsf{fhe},j}\mathbf{R}_j - (\mathbf{x}_j,\mathsf{sd}_j)\otimes\mathbf{G}\}_{j\in[Q_{\mathsf{msg}}]}, \{\mathbf{c}^\mathsf{T}_{\mathsf{att},j} = \mathbf{s}_j^\mathsf{T}(\mathbf{A}_{\mathsf{att}} - \mathsf{bits}(1,\mathbf{X}_j)\otimes\mathbf{G}) + \mathbf{e}^\mathsf{T}_{\mathsf{att},j}\}_{j\in[Q_{\mathsf{msg}}]}\right),$$
$$\mathsf{aux}_2 = (f_1,\ldots,f_{Q_{\mathsf{key}}}, \mathsf{aux}_{\mathcal{A}}, \mathbf{r}_1,\ldots,\mathbf{r}_{Q_{\mathsf{key}}}, \mathbf{A}_{\mathsf{att}}, M).$$

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

Figure 5: Description of the Sampler for Evasive LWE

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 (64) as follows.

$$\begin{pmatrix} \mathsf{aux}_2,\ \mathbf{B}, \{\mathbf{c}^\mathsf{T}_{\mathbf{B},j} = \mathbf{s}_j^\mathsf{T}\mathbf{B} + \mathbf{e}^\mathsf{T}_{\mathbf{B},j}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{X}_j = \mathbf{A}_{\mathsf{fhe},j}\mathbf{R}_j - (\mathbf{x}_j,\mathsf{sd}_j)\otimes\mathbf{G}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}^\mathsf{T}_{\mathsf{att},j} = \mathbf{s}_j^\mathsf{T}(\mathbf{A}_{\mathsf{att}} - \mathsf{bits}(1,\mathbf{X}_j)\otimes\mathbf{G}) + \mathbf{e}^\mathsf{T}_{\mathsf{att},j}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}^\mathsf{T}_{\mathbf{P},j,k} = \mathbf{s}_j^\mathsf{T}\mathbf{A}_{\mathrm{F}_k} + \mathbf{e}^\mathsf{T}_{\mathbf{P},j,k}\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]}\end{pmatrix} \approx_c \begin{pmatrix} \mathsf{aux}_2,\ \mathbf{B}, \{\mathbf{c}_{\mathbf{B},j} \leftarrow \mathbb{Z}_q^{mw}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{X}_j \leftarrow \mathbb{Z}_q^{(n+1)\times m(\lambda+\mathrm{L})}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}_{\mathsf{att},j} \leftarrow \mathbb{Z}_q^{(\mathrm{L_X}+1)m}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}^\mathsf{T}_{\mathbf{P},j,k} \leftarrow \mathbb{Z}_q^\ell\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]}\end{pmatrix} \tag{65}$$

where $\mathbf{e}_{\mathbf{P},j,k} \leftarrow \mathcal{D}^\ell_{\mathbb{Z},\sigma_1}$. Now, to prove Equation (62) it suffices to show Equation (65).
We prove Equation (65) via the following sequence of hybrids.

Hyb_0. This is L.H.S distribution of Equation (65).

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

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

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 (60) we note that

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

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

- Next, we note that $\left\| \mathsf{err}_{j,k} \right\| \leq 2^{8\lambda + O(1)}\beta/\mathrm{poly}(\lambda) = \chi_1 = \left\| \mathbf{e}_{\mathbf{P},j,k} \right\|$. Thus by noise flooding (Lemma A.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

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

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 $1 \leq j \leq i$. 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 B.2. $\mathsf{Hyb}_{1.i-1} \approx_c \mathsf{Hyb}_{1.i}$, 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 $\mathbf{b}^\mathsf{T} = (\mathbf{b}^\mathsf{T}_{\mathbf{B}}, \mathbf{b}^\mathsf{T}_{\mathsf{fhe}}, \mathbf{b}^\mathsf{T}_{\mathsf{att}})$. 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 $\mathbf{c}^\mathsf{T}_{\mathbf{B},i} := \mathbf{b}^\mathsf{T}_{\mathbf{B}} - \underline{\mathbf{b}}^\mathsf{T}$.
     - Sets $\mathbf{A}_{\mathsf{fhe},i} := \begin{pmatrix} \hat{\mathbf{A}}_{\mathsf{fhe}} \\ \mathbf{b}^\mathsf{T}_{\mathsf{fhe}} \end{pmatrix}$ 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 $\bar{\mathbf{A}}_{\mathsf{att}} = \mathbf{A}'_{\mathsf{att}} + \mathsf{bits}(1, \mathbf{X}_i) \otimes \bar{\mathbf{G}}$, $\mathbf{A}_{\mathsf{att}} = \begin{pmatrix} \bar{\mathbf{A}}_{\mathsf{att}} \\ \underline{\mathbf{a}}^\mathsf{T}_{\mathsf{att}} \end{pmatrix}$, where $\underline{\mathbf{a}}_{\mathsf{att}} \leftarrow \mathbb{Z}_q^{(\mathrm{L_X}+1)m}$, and $\mathbf{c}^\mathsf{T}_{\mathsf{att},i} = \mathbf{b}^\mathsf{T}_{\mathsf{att}} - (\underline{\mathbf{a}}^\mathsf{T}_{\mathsf{att}} - \mathsf{bits}(1, \mathbf{X}_i) \otimes \underline{\mathbf{G}})$, 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 $\mathbf{c}^\mathsf{T}_{\mathbf{B},j}, \mathbf{X}_j$ and $\mathbf{c}^\mathsf{T}_{\mathsf{att},j}$ as in the construction, where $\mathbf{c}^\mathsf{T}_{\mathsf{att},j}$ is computed using $\mathbf{A}_{\mathsf{att}} = \begin{pmatrix} \bar{\mathbf{A}}_{\mathsf{att}} \\ \underline{\mathbf{a}}^\mathsf{T}_{\mathsf{att}} \end{pmatrix}$.
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 $(\mathsf{aux}_2, \{\mathbf{c}^\mathsf{T}_{\mathbf{B},j}, \mathbf{X}_j, \mathbf{c}^\mathsf{T}_{\mathsf{att},j}, \tilde{\mathrm{F}}_{j,k}\})$ 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 $\mathbf{b} = \bar{\mathbf{s}}\mathbf{A}_{\mathsf{LWE}} + \mathbf{e}^\mathsf{T}_{\mathsf{LWE}} = \bar{\mathbf{s}}(\mathbf{B}', \hat{\mathbf{A}}_{\mathsf{fhe}}, \mathbf{A}'_{\mathsf{att}}) + (\mathbf{e}^\mathsf{T}_{\mathbf{B}}, \mathbf{e}^\mathsf{T}_{\mathsf{fhe}}, \mathbf{e}^\mathsf{T}_{\mathsf{att}})$, then $\mathbf{b}_{\mathbf{B}} = \bar{\mathbf{s}}\mathbf{B}' + \mathbf{e}^\mathsf{T}_{\mathbf{B}}$, $\mathbf{b}_{\mathsf{fhe}} := \bar{\mathbf{s}}\hat{\mathbf{A}}_{\mathsf{fhe}} + \mathbf{e}^\mathsf{T}_{\mathsf{fhe}}$, and $\mathbf{b}_{\mathsf{att}} := \bar{\mathbf{s}}\mathbf{A}'_{\mathsf{att}} + \mathbf{e}^\mathsf{T}_{\mathsf{att}}$. Thus we have

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

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 $\mathbf{c}^\mathsf{T}_{\mathbf{B},i} := \mathbf{b}^\mathsf{T}_{\mathbf{B}} - \underline{\mathbf{b}}^\mathsf{T}$.
- 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 $\mathbf{c}^\mathsf{T}_{\mathsf{att},i} = \mathbf{b}^\mathsf{T}_{\mathsf{att}} - (\underline{\mathbf{a}}^\mathsf{T}_{\mathsf{att}} - \mathsf{bits}(1, \mathbf{X}_i) \otimes \underline{\mathbf{G}})$.

□

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

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

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 A.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

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

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 $Q_{\mathsf{key}}^2/2^\lambda$ by taking the union bound with respect to all the combinations of k, k'. Thus the probability of outputting the failure symbol is $Q_{\mathsf{key}}^2/2^\lambda$ which is $\mathsf{negl}(\lambda)$.

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

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

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

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

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,

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

Hyb_8. This hybrid is same as the previous one except we sample $\tilde{\mathrm{F}}_{j,k} \leftarrow \mathbb{Z}_q^\ell$. 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

$$(1^{\lambda},\ \mathsf{aux}_{\mathcal{A}},\ \{f_k,\ \tilde{\mathrm{F}}_{j,k} = f_k(x_j)\ \lfloor q/2 \rceil + R_{j,k} + \mathbf{e}^{\mathsf{T}}_{\mathbf{P},j,k}\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]}) \approx_c (1^{\lambda},\ \mathsf{aux}_{\mathcal{A}},\ \{f_k,\ \tilde{\mathrm{F}}_{j,k} \leftarrow \mathbb{Z}_q^{\ell}\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]}) \tag{66}$$

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 (66) and noting that adding random strings does not make the task of distinguishing the two distributions any easier, we achieve the following distribution

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

which is the R.H.S distribution of Equation (65), hence the proof.

□

B.3 Proof for Non-Uniform κ-prCT Security

Our construction also achieves stronger non-uniform κ-prCT security assuming the corresponding strengthening of evasive LWE. This stronger version of the security is not proven in [AKY24b], but will be required for the construction of prMIFE in Section 3. The formal statement follows.

Theorem B.3. Let $\kappa = 2^{\lambda^c}$ for some constant c. Assuming non-uniform κ-evasive LWE (Assumption 2.4), subexponentially secure PRF against non-uniform adversary, and non-uniform sub-exponential LWE (Assumption A.5), there exists a prFE scheme satisfying κ-prCT security as per Definition 2.15.

Proof. We prove that the construction in Appendix B.1 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 $2^{\lambda^{\delta}}$ 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 B.1.

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 $\lambda' < \kappa$.[21] We denote the size of $\mathcal{A}_1$ by t and the distinguishing advantage for the distributions in Equation (62) by ϵ. Assuming non-uniform κ-evasive LWE with respect to Samp defined from $\mathsf{Samp}_{\mathsf{prFE}}$ as in the proof of Theorem B.1, we obtain an adversary $\mathcal{A}_0$ whose size is $Q(\lambda')t$ and the distinguishing advantage against the distributions in Equation (65) is $\epsilon/Q(\lambda') - \mathsf{negl}(\kappa)$ for some polynomial Q by applying Lemma 2.5. We then consider the same sequence of hybrids as that for the proof of Theorem B.1. 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.

[21] 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'.

- 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 (63) with the same advantage and almost the same size. This concludes the proof of the theorem. □

Theorem B.4. Let $\kappa = 2^{\lambda^c}$ for some constant c. Assuming non-uniform κ-evasive LWE (Assumption 2.4), subexponentially secure PRF against non-uniform adversary, and non-uniform sub-exponential LWE (Assumption A.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 2.15 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.