Compact Pseudorandom Functional Encryption from Evasive LWE

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

Abstract

We provide the first construction of *compact* Functional Encryption (FE) for *pseudorandom* functionalities from the (private coin) evasive LWE and LWE assumptions. Intuitively, a pseudorandom functionality means that the output of the circuit is indistinguishable from uniform for *every* input seen by the adversary. This yields the first compact FE for a nontrivial class of functions which does not rely on pairings. We demonstrate the power of our new tool by using it to achieve *optimal* parameters for both key-policy and ciphertext-policy Attribute Based Encryption (ABE) schemes for circuits of *unbounded* depth, from just the LWE and evasive LWE assumptions. This improves prior work along the twin axes of assumptions and performance. In more detail, this allows to: (i) replace the assumption of *circular* evasive LWE used in the work of Hseih, Lin and Luo (FOCS 2023) by plain evasive LWE, (ii) remove the need for the circular tensor LWE assumption in the work of Agrawal, Kumari and Yamada (CRYPTO, 2024), (iii) improve parameters obtained by both aforementioned works to achieve asymptotic optimality.

Previously, optimal parameters for ABE schemes were only achieved using compact FE for P (Jain, Lin and Luo, Eurocrypt 2023) – we show that compact FE for a much weaker class (albeit with incomparable security) suffices. Thus we obtain the first optimal ABE schemes for unbounded depth circuits which can be conjectured post-quantum secure. Along the way, we define and construct a new primitive which we term *laconic* pseudorandom obfuscation from the same assumptions – this may be of independent interest.

[*]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) [SW05, BSW11] generalizes public key encryption to support computation of non-trivial functions on encrypted data, beyond "all or nothing" access. More formally, in FE, a ciphertext is associated with a vector $\mathbf{x}$, a secret key is associated with a circuit f and decryption enables recovery of $f(\mathbf{x})$ and nothing else. Aside from its direct relevance to real world applications for computing on encrypted data, FE has proved to be a powerful tool in the theory of cryptography, and can be used to build a number of advanced primitives. The most prominent amongst these is Indistinguishability Obfuscation (iO), which is considered essentially "crypto-complete" by virtue of its ability to instantiate almost every known cryptographic primitive [BGI$^+$01a, GGH$^+$13, JLS21].

For FE to succeed in being bootstrapped all the way to iO, it must satisfy a strong efficiency property known as *compactness* – at a high level, this posits that the size of the ciphertext should be sublinear in the size of the circuits being supported by the scheme. There has been substantial research effort in the community for instantiating FE (or directly iO) from well-understood assumptions, leading to a sequence of exciting results [JLMS19, GJLS21, Agr19, APM20, WW21, GP21, DQV$^+$21, JLS21, JLS22, RVV24]. The breakthrough work of Jain, Lin and Sahai [JLS21] finally obtained the first construction of compact FE for P from standard assumptions. This has been subsequently improved by [JLS22, RVV24]. However, all these works rely quite crucially on pairings which is dissatisfying. Not only are pairings quantum insecure, it is also desirable to have alternate pathways for constructing such important primitives as FE and iO, even for restricted classes of functions. In the realm of conjectured quantum safety, there exist several candidates from lattices but their security is either based on heuristics, or their underlying assumptions have been broken [Agr19, APM20, WW21, GP21, DQV$^+$21, HJL21, AJS23]. Thus, an outstanding open question in the area is:

Can we construct compact Functional Encryption for any nontrivial functionality from simple lattice assumptions?

Attribute Based Encryption. A special case of Functional Encryption is the notion of Attribute Based Encryption (ABE) [SW05, GPSW06]. ABE is similar to FE but with a crucial difference – in ABE the computation is performed on *public* attribiutes encoded in the ciphertext, while the burden of privacy is only on an unchanging message. In more detail, the ciphertext encodes a public *attribute* $\mathbf{x}$ together with a secret message m, the secret key is generated for a public function f, and decryption outputs m if and only if $f(\mathbf{x}) = 1$. Security is similar to FE, except that $\mathbf{x}$ need not be hidden. This is formalized in an indistinguishability style game which asks that an adversary should be unable to distinguish between an encryption of $(m_0, \mathbf{x})$ and $(m_1, \mathbf{x})$, even given secret keys for functions f_i so long as $f_i(\mathbf{x}) = 0$ for all i. ABE comes in two avatars – "key-policy" where the function f is encoded in the secret key, or "ciphertext-policy" where it is encoded in the ciphertext. These are denoted by kpABE and cpABE respectively. An interesting generalization of ABE is the so-called "Predicate Encryption" (PE) [KSW13] where the attribute $\mathbf{x}$ is also hidden but only against an adversary that does not receive any decrypting key, namely $f_i(\mathbf{x}) = 0$ for all f_i queried by the adversary.

Prior Work in ABE. There has been significant progress in constructing ABE for circuits over the last several years [GPSW06, GVW13, BGG$^+$14, AY20, Wee22, HLL23] from well-understood assumptions. Here, there has been asymmetry between the key and ciphertext policy variants – while kpABE can be constructed from the standard Learning With Errors (LWE) assumption, it's cpABE counterpart additionally requires a relatively new, strong assumption called *Evasive* LWE [Wee22, Tsa22].

Recently, an important focus area in ABE research has been to achieve asymptotic optimality for both kpABE and cpABE schemes [Wee24, JLL23]. The very recent work of Wee [Wee24] constructs kpABE for *bounded* depth circuits using a new assumption called L-succinct LWE, which is weaker than evasive LWE. Assuming compact FE, [JLL23] achieve optimality even for *unbounded* depth circuits, but this assumption necessitates the reliance on pairings as discussed above. We summarize the state of the art in Table 1. Thus, an outstanding open question in ABE research is:

Can we construct kp/cp ABE *for* P *with optimal parameters from lattices?*

Reference	KP/CP	$\lvert\mathsf{mpk}\rvert$	$\lvert\mathsf{sk}\rvert$	$\lvert\mathsf{ct}\rvert$	Depth	Assumptions
[JLL23]	KP	$O(1)$	$O(1)$	$O(1)$	Unbdd	Compact FE for P
[JLL23]	CP	$O(1)$	$O(1)$	$O(1)$	Unbdd	Compact FE for P
[HLL23]	KP	$O(L)$	$O(1)$	$O(L)$	Unbdd	evasive circular LWE+ circular LWE
[AKY24a]	CP	$O(L)$	$O(L)$	$O(1)$	Unbdd	LWE + evasive LWE + circular tensor LWE
[Wee24]	KP	$L^2 \cdot O(d)$	$O(d)$	$O(d)$	Bdd	L-succinct LWE
[HLL24]	CP	$O(d)$	$L \cdot O(d)$	$L \cdot O(d)$	Bdd	LWE + evasive learning with structured errors
This Work	KP	$O(1)$	$O(1)$	$O(1)$	Unbdd	LWE + evasive LWE
This Work	CP	$O(1)$	$O(1)$	$O(1)$	Unbdd	LWE + evasive LWE

Table 1: Here L denotes the input length and d denotes the depth of the circuit supported by the ABE schemes. $O(\cdot)$ hides $\mathrm{poly}(\lambda)$ factors. We highlight that L-succinct LWE is a falsifiable assumption.

1.1 Our Results

In this work, we make progress on both the above questions and achieve the following: (i) we provide the first construction of compact FE from private-coin evasive LWE (and LWE) for a nontrivial class of *pseudorandom* functionalities, (ii) we use our new FE to construct kpABE and cpABE achieving *optimal* parameters for *unbounded depth* circuits, also from evasive LWE and LWE. We summarize our results as informal theorems below.

The first theorem shows that we can obtain partially hiding pseudorandom FE (PHprFE) with optimal parameter sizes. In PHprFE, an input $\mathbf{x}$ to the function f is divided into a public part $\mathbf{x}_{\mathsf{pub}}$ and private part $\mathbf{x}_{\mathsf{priv}}$. While we consider the same correctness requirement as usual FE, we consider a relaxed security notion where the public part is allowed to leak to the adversary. The advantage of allowing part of the input to be public is that it leads to shorter ciphertexts whose size does not depend on the length of $\mathbf{x}_{\mathsf{pub}}$, when we follow the convention of ignoring the length of the public part when measuring ciphertext size.

Theorem 1.1 (Partially Hiding prFE for Unbounded Depth). Assuming LWE and evasive LWE assumptions, there exists a partially hiding pseudorandom FE scheme, for function class $\mathcal{F} = \{f : \{0,1\}^{L_{\mathsf{pub}}} \times \{0,1\}^{L_{\mathsf{priv}}} \to \{0,1\}\}$, that satisfies very selective security (more accurately, security as per Definition 5.4) whose master public key and the secret key sizes are fixed polynomial $\mathrm{poly}(\lambda)$. Furthermore, the size of the ciphertext is $L_{\mathsf{priv}} + \mathrm{poly}(\lambda)$.

Here, very selective security refers to the security notion where the adversary has to choose its challenge query and key queries before seeing the public parameter.

Note that in particular, this implies a plain FE scheme for pseudorandom functionalities with optimal parameters (by setting the public part of the input to $\perp$). We then leverage the above FE to construct Attribute Based and Predicate Encryption (ABE and PE respectively) for unbounded depth circuits with optimal parameters, in both the KP and CP setting. In more detail, we prove the following.

Theorem 1.2 (Optimal KP-ABE). [Theorem 6.1] Under the LWE and Evasive LWE assumptions, there exists very selectively secure KP-ABE scheme supporting circuits $\{C : \{0,1\}^\ell \to \{0,1\}\}$ with unbounded depth and message space $\{0,1\}^\lambda$ with

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

Theorem 1.3 (Optimal KP-PE). [Theorem 6.5] Under the LWE and Evasive LWE assumptions, there exists very selectively secure KP-PE scheme supporting circuits $\{C : \{0,1\}^\ell \to \{0,1\}\}$ with unbounded depth and message space $\{0,1\}^\lambda$ with

$$|\mathsf{mpk}| = \mathrm{poly}(\lambda),\ |\mathsf{sk}_C| = \mathrm{poly}(\lambda),\ |\mathsf{ct}| = \mathrm{poly}(\lambda) + |\mathbf{x}|$$

where $\mathbf{x} \in \{0,1\}^\ell$.

Theorem 1.4 (Optimal CP-ABE). [Theorem 7.2] Under the LWE and Evasive LWE assumptions, there exists very selectively secure CP-ABE scheme supporting circuits with unbounded depth $\{C : \{0,1\}^\ell \to \{0,1\}\}$ and message space $\{0,1\}^\lambda$ with

$$|\mathsf{mpk}| = \mathrm{poly}(\lambda),\ |\mathsf{sk}_{\mathbf{x}}| = \mathrm{poly}(\lambda),\ |\mathsf{ct}| = \mathrm{poly}(\lambda)$$

where $\mathbf{x} \in \{0,1\}^\ell$.

We observe that since our schemes support message space $\{0,1\}^\lambda$, they can be used to support arbitrary message space by using the hybrid encryption framework. We note that our unbounded CP-ABE scheme Theorem 7.2 and unbounded KP-ABE scheme Theorem 6.1 can be used to instantiate ABE for Turing Machines ([AKY24a]). We obtain the following corollary, which improves both the assumptions and the parameters of [AKY24a].

Corollary 1.5 (KP-ABE for TM). [Corollary 7.4] Under the LWE and evasive LWE assumptions, there exists a very selectively secure ABE for TM with

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

where the Turing machine M runs on input $\mathbf{x}$ for time step t.

In contrast, [AKY24a] uses LWE, evasive LWE and circular tensor LWE and achieves $|\mathsf{mpk}| = \mathrm{poly}(\lambda)$, $|\mathsf{sk}| = \mathrm{poly}(|M|, \lambda)$, $|\mathsf{ct}| = \mathrm{poly}(\lambda, |\mathbf{x}|, t)$.

1.2 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. Moreover, FE or MIFE for any class of functionalities was not considered.

Our Companion Paper. In a companion work [AKY24b], we bootstrap our compact FE for pseudorandom functionalities to the multi-input setting. This yields the first multi-input FE for pseudorandom functionalities from LWE and (a stronger variant of) private-coin evasive LWE. Using the techniques of [AJ15], this multi-input FE can be naturally deployed to construct the first iO for pseudorandom functionalities. We then leverage our pseudorandom multi-input FE and iO for applications – we refer the reader to [AKY24b] for details.

In the present work, our construction of ABE with optimal parameters (Section 5) uses a restricted iO (called pPRIO) as a building block which is developed in the companion work [AKY24b]. This creates 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 companion work. By *not* using it in the present work, we would achieve ABE schemes with sub-optimal parameters which nevertheless outperform the state of the art. We would then improve these schemes in the companion work, which was undesirable. Another alternative was to provide the multi-input compiler for constant arity in the present work and generalize this to polynomial arity (from a stronger assumption) in the companion 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 developed here is the best option.

1.3 Recent Attacks on Evasive LWE and Repercussions.

Subsequent to the first online appearance of the present work, some counter-examples for evasive LWE were developed. We discuss these and their impact on the present work below. To begin, we recall the definition of evasive LWE.

The evasive LWE assumption [Wee22, Tsa22] roughly says that if

$$\big(\mathbf{B},\ \mathbf{P},\ \mathbf{s}^{\mathsf{T}}\mathbf{B}+\mathbf{e}_{\mathbf{B}},\ \mathbf{s}^{\mathsf{T}}\mathbf{P}+\mathbf{e}_{\mathbf{P}},\ \mathsf{aux}\ \big) \approx_c \big(\ \mathbf{B},\ \mathbf{P},\ \$,\ \$,\ \mathsf{aux}\big)$$

where \$ represents random, then

$$\big(\ \mathbf{B},\ \mathbf{P},\ \mathbf{s}^{\mathsf{T}}\mathbf{B}+\mathbf{e}_{\mathbf{B}},\ \mathbf{B}^{-1}(\mathbf{P}),\ \mathsf{aux}\big) \approx_c \big(\ \mathbf{B},\ \mathbf{P},\ \ \$,\ \mathbf{B}^{-1}(\mathbf{P}),\ \mathsf{aux}\big)$$

Above $\mathbf{B}^{-1}(\mathbf{P})$ refers to a low norm matrix, say $\mathbf{K}$, such that $\mathbf{B}\mathbf{K} = \mathbf{P} \mod q$. It is clear that given $\mathbf{B}^{-1}(\mathbf{P})$ and $\mathbf{s}^{\mathsf{T}}\mathbf{B}+\mathbf{e}_{\mathbf{B}}$, the adversary can compute $\mathbf{s}^{\mathsf{T}}\mathbf{P}+\mathbf{e}_{\mathbf{B}}\mathbf{K}$. Evasive LWE intuitively says that the adversary cannot exploit $\mathbf{B}^{-1}(\mathbf{P})$ in any other way. The rationale behind the assumption, discussed at length in [Wee22], is that the final value obtained by the adversary after decryption, represented above by $\mathbf{s}^{\mathsf{T}}\mathbf{P}+\mathbf{e}_{\mathbf{B}}\mathbf{K}$, is large and completely avoids the so-called "zeroizing" regime, namely the situation where the attacker obtains low norm polynomial equations over the integers and can solve these to obtain harmful leakage. Thus, evasive LWE should allow us to construct schemes where the decryption yields a pseudorandom output, as required by the pre-condition. Indeed, this is the motivation for the name "evasive" LWE – it should only support construction of evasive functionalities where decryption of challenge ciphertexts is not allowed, such as ABE, but not FE/iO.

Evasive LWE has been studied in two main regimes, namely "public-coin" and "private-coin", where the former means that the randomness used to sample $\mathbf{P}$ and auxiliary information aux, is made available to the adversary, and the latter means that this information needs to be hidden. The private-coin version was known to have some contrived counter-examples since the work of [VWW22], but this was not considered too problematic as it relied on highly unnatural auxiliary information which contained obfuscations that would output secrets given $\mathbf{B}^{-1}(\mathbf{P})$ but not otherwise. No attacks were known in the public-coin setting used by Wee's original formulation or its extensions, such as the circular evasive LWE by Hseih, Lin and Luo [HLL23]. Thus, evasive LWE has been seen as a meaningful "middle point" in the land between LWE on one hand, and lattice assumptions used for iO on the other. The community has tried to make progress on long standing "evasive" problems in the world of lattice based cryptography using this assumption, while simultaneously trying to improve the assumption (see for instance [Wee24]).

New Attacks. In a very exciting, very recent development subsequent to the first online posting of the present work, some surprising new counter-examples against evasive LWE became known [AMYY25, HJL25, DJM$^+$25]. In a nutshell, these attacks show that the intuition that evasive LWE evades the zeroizing regime is not always true, even in the public-coin setting. However, by taking suitable precautions, security can be recovered as discussed below.

Malicious Sampler Attacks. The works of [AMYY25, HJL25, DJM$^+$25] demonstrated that the general formulation of evasive LWE, which allows for arbitrary malicious samplers, is *false as stated*. In particular, as related to the present work, the works of [AMYY25, HJL25] showed (via essentially the same attack) that by carefully crafting a contrived circuit to implement a PRF which is used (non black-box) in our scheme, the pre-condition can still be argued true while the post-condition can be shown false. Additionally, besides attacking private coin versions of Evasive LWE that are prevalent in the literature, the attacks by [AMYY25] also affect some public-coin variants, including the "circular, small-secret evasive LWE" by [HLL23].

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 – thus, by refining/restraining the formulation of evasive LWE, even the original construction (presented in the first online posting of this work) is secure. In addition, we modify our original construction of prFE to implement a modulus reduction step, which negates the effect of choosing contrived circuits in the construction – for this modified construction, we do not even know of any attack using malicious samplers. We also remark that in the real world, the circuit representation of functions is chosen by the key generator who is an honest party (it holds the master secret key), and we view the counter-examples emerging from such circuit choices as indicating the limits of the assumption rather than the security of the existing constructions.

Contrived Functionality Attacks. The work of [BDJ$^+$24] and [AMYY25] show that there exists a contrived "self-referential" functionality for which pseudorandom functional encryption or pseudorandom obfuscation cannot exist. As discussed in [AMYY25], this result may be seen as analogous to the impossibilities known for the random oracle model [CGH04] or virtual black box (VBB) obfuscation [BGI$^+$01b]. In more detail, despite impossibilities known for ROM and VBB obfuscation [CGH04, BGI$^+$01b], 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 blind garbled circuits or FE ciphertexts, are quite natural and do not fall prey to such attacks.

Attacks by withholding information about $\mathbf{B}$ *or* $\mathbf{P}$. The elegant work of [BÜW24] 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 – indeed $\mathbf{P}$ is publicly computable using auxiliary information.

Our Perspective. The emergence of attacks on any assumption, such as the recent ones on evasive LWE, can be interpreted in various ways. While counter-examples can (and should) create concern about indiscriminately using the assumption, one school of thought might be to completely discard the assumption by labeling it false and meaningless. Another school of thought might be that counter-examples are also progress, and that they shed light on if/how the assumption should be refined to regain security, and constructions from these refined assumptions are still meaningful (especially in the absence of alternatives).

Our perspective is that disciplined new conjectures are important to make meaningful progress on problems that have resisted solutions from standard assumptions. In this context, we believe that the evasive LWE assumption has played a crucial role in enabling constructions that could not be built from plain LWE despite significant effort by the community over several years. By examining carefully the nature of counter-examples and seeing whether there are meaningful lessons to learn, we can hope to arrive at stable versions that allow to expand the boundaries of cryptography. These constructions and their proofs could yield insights that would eventually enable candidates from standard assumptions (as happened in the world of pairings, for instance) [Wat12, GWW19, AMY19, GZ21, KLVW23].

Finally, we remark that proposing principled new assumptions, by its very nature, highly non-trivial and it is unrealistic to expect the perfect formulation in the very first attempt. In our judgment, a balance of caution and risk is beneficial in this context.

1.4 Technical Overview

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

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

Providing Advice for Bootstrapping. The essential barrier in supporting circuits of unbounded depth for homomorphic computation is that the norm of the matrix $\mathbf{H}_{\mathbf{A},f,\mathbf{x}}$ grows exponentially with the depth of the circuit being computed, causing the the noise in the ciphertext encoding to blow out of control after some number of evaluations. The same barrier was encountered while constructing fully homomorphic encryption (FHE), to resolve which, Gentry proposed the beautiful "bootstrapping" idea. Roughly speaking, bootstrapping suggests performing homomorphic evaluation of the decryption circuit on the large-noise ciphertext – this has the effect of internally throwing away the large noise accumulated in the ciphertext and "refreshing" it with smaller, more manageable noise. Since this procedure can be performed every time the ciphertext has accumulated large noise, the evaluator can keep on going! However, to execute this approach, one needs to assume that the scheme satisfies circular security, namely it should be safe to provide a ciphertext encrypting the scheme's own secret key, as this is required for homomorphic decryption described above.

While it has been long known how to use circular security in the context of unbounded depth FHE [Gen09], its utility in the context of ABE was uncovered only very recently, in an elegant work by Hseih, Lin and Luo [HLL23] (HLL). In more detail, HLL supports unbounded homomorphism in ABE via two steps: (i) noise removal, and (ii) bootstrapping. The noise removal step is via modulus reduction à la BV/BGV [BV11, BGV14] which destroys the algebraic structure of the BGG^+ encoding required for further evaluation, while the bootstrapping step makes use of a circular encoding, or "advice", which enables to restore the structure of the BGG^+ encoding, making it suitable for further evaluation. In more detail, the HLL advice has the following structure:

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

Above, $\mathsf{hct}_{\mathbf{s}}(\cdot)$ is an FHE ciphertext decryptable by secret key $\mathbf{s}$ denoted in the subscript – thus $\mathbf{S}$ is a circular FHE ciphertext, and $\mathbf{E}$ is a BGG^+ encoding with attribute $\mathbf{S}$ and *re-using* the FHE secret $\mathbf{s}$ as the LWE secret! The trick of reusing the FHE secret as the LWE secret in the BGG^+ encoding of attribute $\mathsf{hct}_{\mathbf{s}}(\cdot)$, was introduced by Brakerski et al. [BTVW17] and can lead to "automatic decryption" of the FHE ciphertext, as described next. Recall that in the GSW FHE scheme [GSW13], the secret key is $\mathbf{s}^{\mathsf{T}}$, a ciphertext for message $\mathbf{y}^{\mathsf{T}}$ is a matrix $\mathbf{C}$ and decryption computes $\mathbf{s}^{\mathsf{T}}\mathbf{C}$ to recover (a noisy version of) $\mathbf{y}^{\mathsf{T}}$. Brakerski et al. [BTVW17] suggested "vectorizing" the BGG^+ ciphertext evaluation procedure so that homomorphic evaluation on the encoding produces a term of the form $\underline{\mathbf{s}^{\mathsf{T}}(\mathbf{A}_f - \mathsf{hct}_{\mathbf{s}}(\mathbf{y}^{\mathsf{T}}))}$ (i.e. without $\mathbf{G}$). Now, the inner product of $\mathbf{s}$ and $\mathsf{hct}_{\mathbf{s}}(\mathbf{y}^{\mathsf{T}})$ causes FHE decryption to occur automatically and we obtain the encoding $\underline{\mathbf{s}^{\mathsf{T}}\mathbf{A}_f + \mathbf{y}^{\mathsf{T}}}$, where the noise in the encoding is low. It turns out that this term is exactly what is needed to restore the structure of the ill-formed encoding obtained by step (i) of HLL – this allows to create a low noise BGG^+ encoding which can be used to evaluate further. Put together, HLL prove the following:

Theorem 1.6. Assuming circular evasive LWE and LWE, there exists a very selectively secure kpABE scheme for circuits of unbounded depth and attribute length ℓ with $|\mathsf{mpk}| = \mathrm{poly}(\lambda, \ell)$, $|\mathsf{sk}_C| = \mathrm{poly}(\lambda)$, $|\mathsf{ct}| = \mathrm{poly}(\lambda, \ell)$.

Above *circular evasive LWE* is a new assumption introduced by HLL which "mixes" circularity into the recently introduced evasive LWE assumption [Wee22, Tsa22].

Randomizing Advice for CP-ABE. Recently, Agrawal, Kumari and Yamada [AKY24a] (AKY) built upon the construction by HLL to obtain the first ABE for Turing machines from lattice assumptions. A key technical contribution of the AKY construction is a way to randomize the advice provided in the HLL ciphertext, making it suitable for integration with Wee's bounded depth CP-ABE. These techniques led to the first CP-ABE for unbounded depth circuits, which they further leveraged to construct a (KP-)ABE for Turing machines. In more detail, the AKY transformation requires computation of the following randomized HLL terms:

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

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

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

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

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

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

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

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

Theorem 1.7. [AKY24a][Thm 5.6] Under LWE, circular tensor LWE and evasive LWE, there exists a very selectively secure ABE for TM with $|\mathsf{mpk}| = \mathrm{poly}(\lambda)$, $|\mathsf{sk}| = \mathrm{poly}(\lambda, |M|)$, $|\mathsf{ct}| = \mathrm{poly}(\lambda, |\mathbf{x}|, t)$.

Above M is the Turing machine, $\mathbf{x}$ is the input and t is the worst case running time of M on any input. Note that while the AKY construction does not need the circular evasive LWE assumption used by HLL, they require a circular *tensor* assumption which is a new assumption that they introduce.

Compact Functional Encryption for Pseudorandom Functionalities. Our starting point is the observation that the techniques developed in AKY are quite a bit more general and can be leveraged to compute functionalities beyond the randomized HLL advice they were developed for. Taking a step back, let us analyze what their technique enables: the encryptor provides an FHE ciphertext $\mathbf{T}$ of a message (say $\mathbf{x}$), and a BGG^+ encoding of attribute $\mathbf{T}$ with the FHE secret doubling up as the encoding randomness. Homomorphic evaluation of any function f coupled with automatic decryption allows to recover a masked version of $f(\mathbf{x})$ and evasive LWE allows to cancel the mask. Thus, this technique seems to enable computation of any function f on the input $\mathbf{x}$, while keeping it hidden! Intuitively, security follows from evasive LWE as long as the output of the functionality is pseudorandom, such as their $(\mathbf{S}_{\mathbf{r}}, \mathbf{E}_{\mathbf{r}})$, but more generally the output of any pseudorandom function, such as a PRF.

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

- The setup algorithm samples matrices $\mathbf{A}_{\mathsf{att}}$ and $(\mathbf{B}, \mathbf{B}^{-1})$ of appropriate dimensions and outputs $\mathsf{mpk} := (\mathbf{A}_{\mathsf{att}}, \mathbf{B})$ and $\mathsf{msk} := \mathbf{B}^{-1}$. Here, $\mathbf{B}^{-1}$ is the trapdoor for $\mathbf{B}$ which allows to compute short preimages $\mathbf{B}^{-1}(\mathbf{U})$ for any target matrix $\mathbf{U}$.
- The encryptor on input $\mathbf{x}$ first samples a GSW secret key $\mathbf{s}$ and a PRF seed $\mathsf{sd} \leftarrow \{0,1\}^{\lambda}$. It then computes a GSW ciphertext, $\mathbf{X} = \mathsf{hct}_{\mathbf{s}}(\mathbf{x}, \mathsf{sd})$, using public key $\mathbf{A}_{\mathsf{fhe}} = (\bar{\mathbf{A}}_{\mathsf{fhe}} \;\; \bar{\mathbf{s}}^{\mathsf{T}}\bar{\mathbf{A}}_{\mathsf{fhe}} + \mathbf{e}_{\mathsf{fhe}}^{\mathsf{T}})$ and randomness $\mathbf{R}$, – followed by a BGG^+ encoding of $\mathbf{X}$ using randomness $\mathbf{s}$ as $\mathbf{c}_{\mathsf{att}}^{\mathsf{T}} := \mathbf{s}^{\mathsf{T}}(\mathbf{A}_{\mathsf{att}} - \mathbf{X} \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{att}}^{\mathsf{T}}$. It additionally computes $\mathbf{c}_{\mathbf{B}}^{\mathsf{T}} := \mathbf{s}^{\mathsf{T}}\mathbf{B} + \mathbf{e}_{\mathbf{B}}^{\mathsf{T}}$ and outputs the ciphertext $\mathsf{ct} = (\mathbf{c}_{\mathbf{B}}, \mathbf{c}_{\mathsf{att}}, \mathbf{X})$.
- The key generator on input $\mathsf{msk} = \mathbf{B}^{-1}$ and function f does the following.
  (a) Samples a nonce $\mathbf{r} \leftarrow \{0,1\}^{\lambda}$ and defines function $\mathrm{F}[f, \mathbf{r}]$, with f and $\mathbf{r}$ hardwired, as

  $$\mathrm{F}[f, \mathbf{r}](\mathbf{x}, \mathsf{sd}) = f(\mathbf{x}) \lfloor q/2 \rceil + \mathsf{PRF}(\mathsf{sd}, \mathbf{r}).$$

  It then computes the FHE evaluation circuit $\mathsf{VEval}_{\mathrm{F}}$ w.r.t. the function $\mathrm{F}[f, \mathbf{r}]$ (this can be computed using the knowledge of $\mathrm{F}[f, \mathbf{r}]$). Note that the circuit $\mathsf{VEval}_{\mathrm{F}}$ can be used to compute on a GSW ciphertext encoding an input, say $\mathbf{y}$, to recover a GSW ciphertext encoding $\mathrm{F}[f, \mathbf{r}](\mathbf{y})$.

(b) Next, it computes the matrix $\mathbf{H}^{\mathrm{F}}_{\mathbf{A}_{\mathsf{att}}}$ for the circuit $\mathsf{VEval}_{\mathrm{F}}$ using the public matrix $\mathbf{A}_{\mathsf{att}}$. Recall that the matrix $\mathbf{H}^{\mathrm{F}}_{\mathbf{A}_{\mathsf{att}}}$ and $\mathbf{H}^{\mathrm{F}}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}$ (which can be computed given $\mathsf{VEval}_{\mathrm{F}}$, $\mathbf{A}_{\mathsf{att}}$ and $\mathbf{X}$) will satisfy the relation

$$(\mathbf{A}_{\mathsf{att}} - \mathbf{X} \otimes \mathbf{G})\mathbf{H}^{\mathrm{F}}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}} = \mathbf{A}_{\mathsf{att}}\mathbf{H}^{\mathrm{F}}_{\mathbf{A}_{\mathsf{att}}} - \mathsf{VEval}_{\mathrm{F}}(\mathbf{X}).$$

(c) It sets $\mathbf{A}_{\mathrm{F}} = \mathbf{A}_{\mathsf{att}} \cdot \mathbf{H}^{\mathrm{F}}_{\mathbf{A}_{\mathsf{att}}}$, samples $\mathbf{K} \leftarrow \mathbf{B}^{-1}(\mathbf{A}_{\mathrm{F}})$ and outputs $\mathsf{sk}_f = (\mathbf{K}, \mathbf{r})$.

– The decryption on input $\mathsf{sk}_f = (\mathbf{K}, \mathbf{r})$ and $\mathsf{ct} = (\mathbf{c}_{\mathbf{B}}, \mathbf{c}_{\mathsf{att}}, \mathbf{X})$ work as follows.

(a) It first computes the matrix $\mathbf{H}^{\mathrm{F}}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}$ for the circuit $\mathsf{VEval}_{\mathrm{F}}$ using $\mathbf{A}_{\mathsf{att}}$ and $\mathbf{X}$.

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

To see the correctness of our scheme, we note that

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

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

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

where $\mathsf{VEval}_{\mathrm{F}}(\mathsf{bits}(\mathbf{X})) = \mathbf{A}_{\mathsf{fhe}}\mathbf{R}_{\mathrm{F}} - (\mathbf{0} \;\; \mathrm{F}[f, \mathbf{r}](\mathbf{x}, \mathsf{sd}))^{\mathsf{T}}$.

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

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

As seen above, our construction achieves compactness.

Security. Intuitively, our notion of security says that so long as the *output* of the functionality is pseudorandom, the *ciphertext* is pseudorandom, given all the additional information available to the adversary. We denote this security notion as prCT security. In more detail, let Samp be a PPT algorithm that on input 1^λ, outputs

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

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

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

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

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

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

We observe that the above definition is incomparable to the standard indistinguishability (IND) security definition that is used in FE schemes [BSW11]. Indeed, IND style security does not appear meaningful for our functionality. Recall that in the IND game, the adversary must submit two challenge messages m_0 and m_1 and request keys for functions f_i such that $f_i(m_0) = f_i(m_1)$. However since the output $f_i(m_b)$ is pseudorandom for $b \in \{0,1\}$, the event $f_i(m_0) = f_i(m_1)$ occurs only with negligible probability[1]. However, our definition has a simulation security flavour and is very handy for applications, as we will see.

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

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

This allows to simplify these two terms to $\underline{\mathbf{s}^{\mathsf{T}}\mathbf{A}_{\mathrm{F}}}$ and $\underline{f(\mathbf{x})\lfloor q/2 \rceil + \mathsf{PRF}(\mathsf{sd},\mathbf{r})}$, where the latter term is pseudorandom, hence simulatable and can be ignored hereafter. The terms that remain can now be handled by relying on LWE using standard techniques [Wee22, HLL23, AKY24a]. Please see Section 3 for the detailed proof.

Handling Malicious Samplers. As discussed above, subsequent to the initial posting of this work on eprint, [AMYY25] demonstrated that the general formulation of evasive LWE is false as stated. At a very high level, the attack, building upon clever ideas by [HJL21], shows a way to create a correlation between the error term resulting from FHE evaluation (and automatic decryption) with the PRF output by using a contrived circuit to implement the PRF.

As suggested in [AMYY25], there are multiple ways to restrain malicious samplers to prevent such attacks. The simplest one is to leverage the fact that the secret key is computed by the key generator who is an honest party (it holds the master secret key) in the real world, and can ensure that the circuit representation of any function f as well as the PRF can be made canonical by using the universal circuit or a garbled circuit representation. Nevertheless, here, we present an alternate fix to the scheme which uses modulus reduction to "throw away" the accumulated error after FHE evaluation, replacing it with rounding error which is no longer correlated with the PRF seed, even for a contrived circuit chosen by the adversary. To begin, we provide a high level outline of the attack.

The Attack. The adversary, given $\mathbf{c}_{\mathbf{B}}$, $\mathbf{c}_{\mathsf{att}}$, $\mathbf{X}$, and $\mathbf{K}$, computes $\mathbf{c}_{\mathbf{B}}^{\mathsf{T}} \cdot \mathbf{K} - \mathbf{c}_{\mathsf{att}}^{\mathsf{T}} \cdot \mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}}$. Simplifying, she obtains

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

By correctness, the adversary recovers $f(\mathbf{x})$ and can therefore strip it away to obtain $\mathsf{PRF}(\mathsf{sd},\mathbf{r}) + \mathbf{e}_{\mathbf{B}}^{\top}\mathbf{K} - \mathbf{e}_{\mathsf{fhe}}^{\top}\mathbf{R}_{\mathrm{F}} - \mathbf{e}_{\mathsf{att}}^{\top}\mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}}$, as described in Equation (2). Now, in the proof, the error term $\mathbf{e}_{\mathbf{B}}^{\top}\mathbf{K}$ is replaced by i.i.d error $\mathbf{e}_{\mathbf{P}}$ and is used to break any correlation between $\mathsf{PRF}(\mathsf{sd},\mathbf{r})$ and $\mathbf{e}_{\mathsf{fhe}}^{\top}\mathbf{R}_{\mathrm{F}} - \mathbf{e}_{\mathsf{att}}^{\top}\mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}}$. This allows us to prove the pre-condition based on just plain LWE.

However, in the real world, $\mathbf{e}_{\mathbf{B}}^{\top}\mathbf{K}$ cannot be used to break the dependence between the above two terms. Then, by choosing the circuit implementing F in some contrived way, the authors set it up so that $\mathsf{PRF}(\mathsf{sd},\mathbf{r})$ and $\mathbf{e}_{\mathsf{fhe}}^{\top}\mathbf{R}_{\mathrm{F}}$ are correlated, and in particular cancel each other modulo 2. Note that these terms are small, and do not wraparound modulo q, so computing mod 2 is well defined. Now we are left with $\mathbf{e}_{\mathbf{B}}^{\mathsf{T}}\mathbf{K} + \mathbf{e}_{\mathsf{att}}^{\mathsf{T}}\mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}}$ – but these are linear equations with known coefficients, in the error terms of the original encodings. Using sufficiently many equations, the adversary can easily recover the error terms. On the other hand, had the term $\mathbf{e}_{\mathbf{B}}^{\mathsf{T}}\mathbf{K}$ been truly random, such a system of equations would not admit any solution. This leads to a distinguishing strategy.

Fixing the Scheme. We describe an approach that helps us break the problematic correlation even if the circuit implementation is chosen in a contrived manner – our idea is to use modulus reduction get rid of the problematic error terms involving $\mathbf{e}_{\mathsf{fhe}}^{\mathsf{T}}\mathbf{R}_{\mathrm{F}}$ so that the correlation is destroyed. Informally, we fix a rounding constant $M \in \mathbb{Z}$ such that

[1]A generalization of IND security which requires $f_i(m_b)$ to be pseudorandom and hence computationally indistinguishable for any b might have been more suitable.

$\left\|\mathbf{e}_{\mathsf{fhe}}^{\mathsf{T}}\mathbf{R}_{\mathrm{F}}\right\| < M$ which implies $\lfloor(\mathbf{e}_{\mathsf{fhe}}^{\mathsf{T}}\mathbf{R}_{\mathrm{F}})/M\rfloor = 0$. This gets rid of the problematic error, replacing it with rounding error which is uncorrelated with the PRF seed. For concreteness, we elaborate the changes that must be made to our prFE construction to incorporate the above fix.

1. In setup algorithm, we output M as a part of mpk. The encryption algorithm remains the same.
2. The key generation algorithm has the following changes.
   - We parse $\mathrm{F}[f,\mathbf{r}](\mathbf{x},\mathsf{sd}) = f(\mathbf{x})\lfloor q/2\rceil + \mathsf{PRF}(\mathsf{sd},\mathbf{r}) = M\cdot f_{\mathsf{high}}(\mathbf{x},\mathsf{sd}) + f_{\mathsf{low}}(\mathbf{x},\mathsf{sd})$, where $f_{\mathsf{high}}(\mathbf{x},\mathsf{sd}) \in [0,q/M]^{\ell}$ and $f_{\mathsf{low}}(\mathbf{x},\mathsf{sd}) \in [0,M-1]^{\ell}$. Next we define functions $\mathrm{F}_{\mathsf{high}} := M\cdot f_{\mathsf{high}}$ and $\mathrm{F}_{\mathsf{low}} := M\cdot f_{\mathsf{low}}$, which on input $(\mathbf{x},\mathsf{sd})$ outputs $M\cdot f_{\mathsf{high}}(\mathbf{x},\mathsf{sd})$ and $M\cdot f_{\mathsf{low}}(\mathbf{x},\mathsf{sd})$, respectively.
   - Next, it computes circuits $\mathsf{VEval}_{\mathsf{high}}$ and $\mathsf{VEval}_{\mathsf{low}}$ for the functions $\mathrm{F}_{\mathsf{high}}$ and $\mathrm{F}_{\mathsf{low}}$, respectively and then uses these circuits to compute the matrices $\mathbf{H}_{\mathbf{A}_{\mathsf{att}}}^{\mathrm{F}_{\mathsf{high}}}$ and $\mathbf{H}_{\mathbf{A}_{\mathsf{att}}}^{\mathrm{F}_{\mathsf{high}}}$ (as described in the previous sketch).
   - It sets
   $$\mathbf{A}_{\mathrm{F}} = M\cdot\left\lfloor\frac{\mathbf{A}_{\mathsf{att}}\cdot\mathbf{H}_{\mathbf{A}_{\mathsf{att}}}^{\mathrm{F}_{\mathsf{high}}}}{M}\right\rfloor + \left\lfloor\frac{\mathbf{A}_{\mathsf{att}}\cdot\mathbf{H}_{\mathbf{A}_{\mathsf{att}}}^{\mathrm{F}_{\mathsf{low}}}}{M}\right\rfloor$$
   and outputs $\mathsf{sk}_f = (\mathbf{K},\mathbf{r})$ where $\mathbf{K} = \mathbf{B}^{-1}(\mathbf{A}_{\mathrm{F}})$.
3. The decryption algorithm is the same except that we compute $\mathbf{z}$ differently as
$$\mathbf{z} := \mathbf{c}_{\mathbf{B}}^{\mathsf{T}}\cdot\mathbf{K} - \left(M\cdot\left\lfloor\frac{\mathbf{c}_{\mathsf{att}}^{\mathsf{T}}\cdot\mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}}}{M}\right\rfloor + \left\lfloor\frac{\mathbf{c}_{\mathsf{att}}^{\mathsf{T}}\cdot\mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{low}}}}{M}\right\rfloor\right)$$

We expand the correctness of the scheme to see how the above changes helps us get rid of the problematic noise terms while decryption. Observe that

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

where we set $\left\|\mathbf{e}_{\mathsf{fhe}}^{\mathsf{T}}\mathbf{R}_{\mathrm{F}} + \mathbf{e}_{\mathsf{att}}^{\mathsf{T}}\mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}}\right\| \ll M$ such that the last equation in the above will hold with high probability. Similarly, we get $\left\lfloor\frac{\mathbf{c}_{\mathsf{att}}^{\mathsf{T}}\cdot\mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{low}}}}{M}\right\rfloor = \mathbf{s}^{\mathsf{T}}\left\lfloor\frac{\mathbf{A}_{\mathsf{low}}}{M}\right\rfloor - f_{\mathsf{low}}(\mathbf{x},\mathsf{sd})$ w.h.p. The rest of correctness follows from the same argument as in the previous sketch. Note that the final error obtained now is $\mathsf{PRF}(\mathsf{sd}) + \mathsf{err}$ where (please see Equation (8))

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

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. We conjecture flooding for appropriately defined sizes.

Fixing the Assumption. The authors of the attacks [AMYY25] also suggest multiple ways to refine the assumption to sidestep malicious samplers that may lead to attacks such as the above. We follow their approach and restrict the sampler for which evasive LWE is conjectured true. To the best of our understanding, the original intuition of Evasive LWE still holds if malicious samplers such as the above are avoided, and counter-measures described by [AMYY25] are applied.

Relation to Circular Small-Secret Evasive LWE of HLL23. In Section 3.4, we show that our prFE construction can be based on a variant of the circular small-secret evasive LWE of [HLL23] – the only difference between the assumption stated in [HLL23] and the one used in Section 3.4 is that the FHE encoding in [HLL23] only encodes the secret $\mathbf{s}$, namely $\mathbf{S} = \mathsf{hct}_{\mathbf{s}}(\mathbf{s})$, whereas in our case, this must additionally include $\mathbf{x}$, i.e. $\mathbf{S} = \mathsf{hct}_{\mathbf{s}}(\mathbf{s}, \mathbf{x})$. This difference is created by the distinction between ABE and FE, since the attribute $\mathbf{x}$ can be public in the former and must be hidden in the latter. Hence, in [HLL23], $\mathbf{x}$ can be output by the sampler directly, whereas in our case, it cannot.

Whether this fundamentally changes the assumption is a matter of opinion – in any event the [HLL23] version of circular small-secret evasive LWE is also broken in [AMYY25] by a very similar attack as on the private coin evasive assumption – but we find the connection interesting since the assumption of [HLL23] is considered public coin. We show in Section 3.4 that similar refinement of this assumption can also avoid known attacks.

Applications. While it is exciting to have a compact FE scheme for any nontrivial functionality from purely (conjectured) post-quantum assumptions, we show that our prFE is also surprisingly powerful, and yields important applications.

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

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

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

2. Blindness: The 1ABE ciphertext and secret key should be pseudorandom when decryption is not allowed.

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

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

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

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

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

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

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

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

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

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

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

Building 1ABE. It remains to instantiate 1ABE with the desired properties. Fortunately, these properties are relatively weak and easily satisfied. For instance, we can instantiate 1ABE simply by using blind garbled circuits [BLSV18] (Section 2.6) – here *blindness* is precisely the property that the garbled circuit and its labels should be pseudorandom, when the evaluation result of the garbled circuit using the given labels is random. Given a blind garbled circuit, the labels of the garbled circuit form the 1ABE ciphertext, the set of garbled gates form the 1ABE key, and decomposability follows from the structure of a garbled circuit, where we can garble each gate independently. Care needs to be taken to modify the circuit C in the secret key so that when $C(\mathbf{x}) = 0$, it outputs a random string rather than $\perp$ so as to be compatible with our prFE. This yields the following theorem.

Theorem 1.8. Assuming LWE and Evasive LWE, there exists a very selectively secure kpABE scheme for circuits of unbounded depth and attribute length ℓ with $|\mathsf{mpk}| = \ell \cdot \mathrm{poly}(\lambda)$, $|\mathsf{sk}_C| = |C| \cdot \ell \cdot \mathrm{poly}(\lambda)$, $|\mathsf{ct}| = \ell \cdot \mathrm{poly}(\lambda)$.

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

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

At a high level, our kpABE uses the compression of the abLFE to shorten the secret key. The key generation computes a digest $\hat{C}$ for the circuit C, then computes a prFE key for a circuit which outputs the online part of abLFE ciphertext. The encrypt algorithm computes the offline part of the abLFE ciphertext and the state st using input $(\mathbf{x}, \mu)$. It then encrypts st using prFE encryption. Now, prFE decryption allows to recover the online part of the ciphertext, and abLFE decryption allows to recover μ if $C(\mathbf{x}) = 1$. We refer the reader to Appendix B.4 for details. We prove the following theorem:

Theorem 1.9. Under the circular LWE assumption and the Evasive LWE assumption, there exists a very selectively secure kpABE scheme for circuits of unbounded depth and attribute length ℓ with

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

Above, the parameters achieved match those of HLL and the assumptions are strictly weaker.

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

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

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

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

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

Correctness follows from those of kpABE and prFE: The decryption of prFE ciphertext using the prFE secret key yields kpABE ciphertext encrypted under $\mathbf{x}$. This kpABE ciphertext can be decrypted using $\mathsf{kpABE.sk}_C$ when $C(\mathbf{x}) = 1$. To prove the security, we show $\mathsf{prFE.ct}$ is pseudorandom. By the security of prFE, it suffices to show that the decryption results of $\mathsf{prFE.ct}$ are pseudorandom. This follows from the security of kpABE, since the decryption results are kpABE ciphertexts which cannot be decrypted by $\mathsf{kpABE.sk}_C$. We finally note that while the above overall proof strategy works when the underlying kpABE has pseudorandom ciphertext, we have to consider more general case where underlying kpABE does not necessarily have pseudorandom ciphertext. Therefore, the final construction as well as the security proof are slightly different. We refer the reader to Section 7 for details. Thus, we obtain the following theorem.

Theorem 1.10. Under the circular LWE assumption and the Evasive LWE assumption, there exists a very selectively secure cpABE scheme for circuits of unbounded depth and attribute length ℓ with

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

We note that the cpABE scheme instantiated as above replaces the reliance on circular tensor LWE and LWE assumptions used by AKY by simply circular LWE. It also achieves a shorter mpk as compared to that of AKY (which is $\mathrm{poly}(\lambda, \ell)$), other parameters being the same.

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

Corollary 1.11. Under the circular LWE assumption and evasive LWE assumption, there exists a very selectively secure ABE for TM with the following parameters:

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

In terms of parameters, the above theorem matches those of AKY. In terms of assumptions, it provides a strict improvement, as discussed above. Later, by instantiating the underlying kpABE to be one with optimal parameters (constructed below), we will obtain a cpABE scheme for unbounded depth circuits with optimal parameters – please see Section 7 for details.

1.4.1 The Quest for Optimal Parameters

Achieving optimal parameters is a central open question in the construction of ABE schemes, and one which has received much attention in the literature [JLL23, Wee24]. Ideally, we wish to obtain optimal parameters for kpABE and cpABE for *unbounded* depth circuits and moreover, from the weakest assumptions possible. So far, we do not know how to step around the usage of evasive LWE either for unbounded depth kpABE or even for bounded depth cpABE, so an outstanding open question is:

Can we construct kp/cp ABE *for unbounded depth circuits with optimal parameters from* LWE *and evasive* LWE*?*

Below, we answer the above question in the affirmative. To achieve optimality, we will need a result from our companion paper [AKY24b], which we use black-box below. While reliance on the companion work is not needed to improve the assumptions and match the parameters from HLL and AKY as discussed above, we are aiming for optimality, which necessitates this dependence as discussed previously.

Wishful Thinking: Obfuscation to the Rescue? Since we do not wish to rely on any kind of circularity assumption, we return to our construction of kpABE from 1ABE (which we built using blind garbled circuits) and prFE as a starting point, and ask if we can compress the parameters. Observe that the kpABE secret key in this case contains prFE secret keys for outputting the randomized versions of the 1ABE ciphertext and secret key, i.e. $\mathsf{sk}_C = \{\mathsf{prFE.sk}_{\mathsf{key},i}, \mathsf{prFE.sk}_{\mathsf{ct},j}\}_{\{i,j\}}$. As discussed above, this approach crucially relies on the decomposability of the secret key and ciphertext of the underling 1ABE scheme, which allows it to randomize and output these "piece-wise" – gate by gate for the circuit and bit by bit for the input.

Suppose, as wishful thinking, one could have an obfuscation of a program which, given an index within the 1ABE ciphertext and secret key, could output the appropriate randomized garbled gate/label – this would allow to make the secret key and ciphertext size independent of the circuit and attribute size, and allow us to achieve optimal compression. Can such an approach be realized?

Let us examine the possibility. It is well known [AJ15, BV18] that compact FE for general circuits can be compiled into compact multi-input FE, which in turn can be compiled into iO for general circuits. Perhaps we can hope that such a compiler also be applied to our compact prFE to obtain multi-input prFE and iO for pseudorandom functionalities. Indeed this is true – in our companion work [AKY24b], we show that a similar compiler as [AJ15, BV18] can be made to work, albeit via a very different proof technique (since our security notion is itself quite different). However, even assuming the existence of iO for pseudorandom functionalities, the problem is not solved because:

1. The FE to iO compiler suffers exponential loss in the reduction. Therefore, if we use iO, we do not get security from polynomial assumptions. Since we are finally constructing only ABE schemes, the reliance on exponential security feels too strong.

2. Even if we use iO, the problem of compression does not get solved because we require the program to "authenticate" the input and only provide the output for the legitimate input. As an example, it should not be possible to obtain a ciphertext component for $x_i = 0$ if $x_i = 1$. This would require hardwiring the input $\mathbf{x}$ (similarly C) into the obfuscation which would bring us back to square one!

We resolve these issues by observing that the second issue can, surprisingly, be used to overcome the first. This leaves us with the second issue, to resolve which, we define a new notion which we call *laconic* poly-domain obfuscation for pseudorandom functionalities, which may be of independent interest.

Let us begin with trying to resolve the first issue: observe that the second issue tells us that we need obfuscation for a very special input domain – one that corresponds to the gates of our particular C or the bits of our particular $\mathbf{x}$, and that inputs outside this domain should not be accepted. This immediately has the effect of shrinking down the domain size from exponential to polynomial! While it is as-yet unclear how to force the domain to be restricted to the special polynomial sized set we require, it at least shows that supporting a polynomial sized domain suffices. We term an obfuscation for circuit with polynomial sized domain as "Poly-Domain Obfuscation" [2]. We define iO for pseudorandom circuits which have polynomial sized domain (denoted by pPRIO) as follows.

[2]The informed reader may wonder about the connection with XIO [LPST16] – note that in XIO, the size of the obfuscated circuit is allowed to be only slightly sublinear in the size of the truth table. In contrast, the efficiency requirement of our notion is the same as standard iO.

1. $\mathsf{Obf}(1^\lambda, C) \to \mathsf{obf}$. The obfuscation algorithm takes as input the security parameter λ and a circuit $C : [N] \to [M]$ with $\mathsf{size}(C) \leq L$ for some arbitrary polynomial $L = L(\lambda)$. It outputs an obfuscation of the circuit obf.

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

Security states that $\mathsf{Obf}(1^\lambda, C)$ is pseudorandom if the truth table of C is pseudorandom. Please see Section 2.8 for a formal definition. Restricting the input space to be of polynomial size helps us to avoid the exponential loss of the transformation as discussed above. We now invoke a theorem from our companion work:

Theorem 1.12 ([AKY24b]). Assuming LWE and evasive LWE assumptions, there exists a secure pPRIO scheme.

We now turn to the second issue, which we resolve by defining a new primitive as described next.

Laconic Poly-Domain Obfuscation for Pseudorandom Functionalities. In pPRIO, we obfuscate circuits with a polynomial-sized input domain of the form $1, 2, \ldots, N$. However, we must now find a way to have more control on what this polynomial set can be, so that we can implement some "authentication" of the inputs without making the size of the obfuscated circuit grow with the domain size, as discussed above. Towards this, we extend pPRIO to introduce the new notion of *laconic* pPRIO, where the input domain can be defined as $X := X_1, \ldots, X_N$ for arbitrary strings $X_1, \ldots, X_N \in \{0,1\}^\ell$, with arbitrary length ℓ. The obfuscated circuit allows for the evaluation of inputs that are in the set $X := \{X_1, \ldots, X_N\}$, but it does not allow evaluation for any inputs outside this set. This is exactly what we want! Looking ahead, the set $X_1, \ldots, X_N$ will be instantiated with the gates of C or the bits of $\mathbf{x}$ in the final construction.

To define laconic pPRIO, we modify the pPRIO syntax so that, in order to obfuscate a circuit with a restricted input domain X, the obfuscator only needs a short digest of X, whose size does not depend on N, rather than the entire description of it. This results in a compact obfuscation whose size is independent of N. Now, the evaluation of an input belonging to X can be performed given the description of X in the clear. We present our definition next.

1. $\mathsf{LDigest}(1^\lambda, X = \{X_i\}_{i \in [N]}) \to \mathsf{dig}$. The digest algorithm takes as input the security parameter λ and an input space X of the form $X = \{X_i \in \{0,1\}^\ell\}_{i \in [N]}$ for some $\ell = \ell(\lambda)$ and $N \in \mathbb{N}$. We assume that X encodes the information of ℓ and N and one can retrieve them efficiently. It outputs a string dig.

2. $\mathsf{LObfuscate}(1^\lambda, \mathsf{dig}, E) \to \mathsf{Lobf}$. The encode algorithm takes as input the security parameter λ, string dig and a circuit $E : \{0,1\}^\ell \to \{0,1\}^L$ whose size is S. It outputs a ciphertext Lobf.

3. $\mathsf{LEval}(X, \mathsf{Lobf}) \to Y$. The decode algorithm takes as input $X = \{X_i \in \{0,1\}^\ell\}_{i \in [N]}$ and Lobf. It outputs $Y = \{Y_i \in \{0,1\}^L\}_{i \in [N]}$.

For efficiency, we need that the size of $\mathsf{dig} = \mathsf{LDigest}(1^\lambda, X)$ should be bounded by $\mathrm{poly}(\lambda, S)$. In particular, the size of dig should be independent of N. For security, we require (roughly) that:

$$\begin{aligned}
&\text{If } (X, E(X_1), \ldots, E(X_N)) \approx_c \left(X, \Delta_1 \leftarrow \{0,1\}^L, \ldots, \Delta_N \leftarrow \{0,1\}^L\right) \\
&\text{then } \quad \left(X, \mathsf{LObfuscate}(1^\lambda, \mathsf{dig}, E)\right) \approx_c (X, \delta \leftarrow \mathcal{O})
\end{aligned}$$

where $\mathsf{dig} \leftarrow \mathsf{LDigest}(1^\lambda, X = \{X_i\}_{i \in [N]})$ and $\mathcal{O}$ denotes the output space of LObfuscate algorithm.

Constructing Laconic Poly-Domain Obfuscation. It remains to build laconic pPRIO. To do so, we use pPRIO in addition to blind garbled circuits (bGC) and blind batch encryption (BBE). Since pPRIO and bGC have been introduced earlier, we recall the notion of BBE [BLSV18] here. Intuitively, BBE allows to hash an input $\mathbf{x}$ so that encryption is performed against the hash of $\mathbf{x}$ instead of $\mathbf{x}$ together with some index $i \in [|\mathbf{x}|]$, and decryption allows to recover one of two messages depending on the i^{th} bit of $\mathbf{x}$[3].

[3]The informed reader may notice similarities with laconic oblivious transfer [CDG+17].

In more detail, a BBE scheme $\mathsf{BBE} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{SingleEnc}, \mathsf{SingleDec})$ is defined as follows: The setup algorithm on input security parameter λ and key length N outputs a common reference string crs. The Gen algorithm on input crs and a string $\mathbf{x} \in \{0,1\}^N$ (to be used as secret key) outputs a hash value h to be used as a public key. The $\mathsf{SingleEnc}$ algorithm on input $(\mathsf{crs}, h, i \in [N], (m_0, m_1) \in \mathcal{M}^2)$ outputs a (single) ciphertext ct. The $\mathsf{SingleDec}$ algorithm on input $(\mathsf{crs}, \mathbf{x}, i \in [N], \mathsf{ct})$ outputs a message $m \in \mathcal{M}$. For correctness, it is required that m_b is recovered if $x_i = b$ and h is the hash of $\mathbf{x}$. Security requires the other message to remain hidden. The blindness of the scheme states that if the encrypted message is random then the ciphertext is indistinguishable from random. We give a construction of a BBE scheme from LWE satisfying these properties in Appendix A.

The basic intuition behind the construction of laconic pPRIO (denoted by LprIO) is as follows. To restrict the circuit evaluation to a special input domain X, we use the BBE algorithm Gen to generate the hash h of X and output this as part of the digest. The obfuscate algorithm, given as input a circuit E now provides a pPRIO obfuscation of another circuit which garbles E and generates BBE encryptions of its labels using the hash. The BBE ciphertext can be decrypted to recover the appropriate label, using which, the garbled circuit can be executed. In more detail:

1. The setup algorithm compresses the input domain $X = \{X_i \in \{0,1\}^\ell\}_{i\in[N]}$ by hashing it using $\mathsf{BBE.Gen}$ into a short hash value h. It outputs $\mathsf{dig} = h$.

2. The obfuscate algorithm $\mathsf{LObfuscate}(1^\lambda, \mathsf{dig}, E)$ outputs a pPRIO obfuscation, $\mathsf{pPRIO.obf}$, of $C[\mathsf{dig}, E]$. The circuit $C[\mathsf{dig}, E]$ on input $i \in [N]$ first computes bGC garbled circuit $\tilde{E}_i$ and input labels $\mathsf{lab}_{i,j,b}$ for $j \in [\ell], b \in \{0,1\}$ – next, it encrypts the bGC labels $(\mathsf{lab}_{i,j,0}, \mathsf{lab}_{i,j,1})$ using BBE encryption and finally outputs $\tilde{E}_i$ and BBE ciphertexts $\mathsf{BBE.ct}_{i,j}$. The BBE ciphertexts are encrypted so that one can retrieve the labels for X_i for $\tilde{E}_i$ if a decryptor knows X.

3. The $\mathsf{LEval}(X, \mathsf{Lobf})$ algorithm first runs pPRIO evaluation on $\mathsf{pPRIO.obf}$ and $i \in [N]$ to obtain bGC garbled circuit $\tilde{E}_i$ and BBE ciphertexts $\mathsf{BBE.ct}_{i,j}$ for $j \in [\ell]$. Next it runs BBE decryption using X, index $(i-1)\ell + j$ (this corresponds to j'th bit of X_i) and $\mathsf{BBE.ct}_{i,j}$ to get labels $\{\mathsf{lab}_{i,j}\}_{j\in[\ell]}$ corresponding to X_i. Finally, it runs bGC evaluation using $\tilde{E}_i$ and $\{\mathsf{lab}_{i,j}\}_{j\in[\ell]}$ to obtain $E(X_i)$ for all $i \in [N]$.

Correctness is immediate given the decryption outline above. We then discuss security. Our goal is to show that $\mathsf{pPRIO.obf}$ is pseudorandom assuming $E(X_1), \ldots, E(X_N)$ are pseudorandom. By the security guarantee of pPRIO, it suffices to prove that the truth table $\{C[\mathsf{dig}, E](i) = (\tilde{E}_i, \{\mathsf{BBE.ct}_{i,j}\}_j)\}_{i\in[N]}$ of the circuit $C[\mathsf{dig}, E]$ is pseudorandom. To show this, we first replace the labels that are *not* corresponding to X_i encrypted inside $\mathsf{BBE.ct}_{i,j}$ with random ones using the security of BBE. Then, by the security of bGC, we replace the garbled circuit $\tilde{E}_i$ and labels corresponding to X_i with the simulated one, computed from $E(X_i)$. We then replace $\{E(X_i)\}_i$ with random strings, which can be done by invoking the assumption of the security definition. Then, by the blindness of bGC, $\tilde{E}_i$ and labels can be replaced with random strings. Finally, by the blindness of BBE, we can replace $\{\mathsf{BBE.ct}_{i,j}\}_j$ with random strings, since they encrypt random strings.

Unfortunately, the above proof strategy does *not* work as described. This is because we assumed in our description above that the output of the pPRIO is pseudorandom, which is required for the security of LprIO to hold. However, unfortunately, this is not the case. Rather, what we have is that the obfuscation of pPRIO can be divided into online and offline parts and that the online part of it is pseudorandom, where the offline part is not dependent on the circuit being obfuscated. Furthermore, the offline part can be reused across invocations. We can show similar property for LprIO, which will be sufficient for the upcoming application of LprIO to PHprFE. We refer the reader to Section 4 for details. We will get back to the issue when we discuss the security proof for our PHprFE.

1.4.2 Optimal Parameters at Last.

Using the techniques developed above, we construct a partially hiding compact FE for pseudorandom functionalities, denoted by PHprFE. The syntax of a $\mathsf{PHprFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ scheme is similar to that of a prFE scheme where the input to the encryption scheme now consists of a public part and a secret part. In more detail, the encryption algorithm takes as input $(\mathbf{x}_{\mathsf{pub}}, \mathbf{x}_{\mathsf{priv}})$ and outputs a ciphertext ct. The ciphertext ct can be decrypted using sk_f and $\mathbf{x}_{\mathsf{pub}}$ to recover $f(\mathbf{x}_{\mathsf{pub}}, \mathbf{x}_{\mathsf{priv}})$. The security of the scheme, at a high level, states that if $f(\mathbf{x}_{\mathsf{pub}}, \mathbf{x}_{\mathsf{priv}})$ is pseudorandom then the ciphertext encoding $(\mathbf{x}_{\mathsf{pub}}, \mathbf{x}_{\mathsf{priv}})$ is indistinguishable from random. We note that a PHprFE scheme implies a prFE scheme if we set $\mathbf{x}_{\mathsf{pub}} = \perp$. We provide a PHprFE construction that achieves optimal parameters and supports circuits of unbounded depth. This in turn will be used to construct optimal kpABE and cpABE for unbounded depth circuits.

Partially Hiding Pseudorandom FE. As discussed above, to get parameters independent of sizes of C and $\mathbf{x}_{\mathsf{pub}}$, we compress them using LprIO scheme. Now, instead of letting the prFE keys directly output labels corresponding to the attribute $\mathbf{x}$ or the garbled circuit corresponding to C, we have it output an obfuscation of circuits that compute these components, as described above. In more detail, we use $\mathsf{LprIO}, \mathsf{bGC}$ and prFE scheme to construct a PHprFE scheme as follows.

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

2. The PHprFE keygen algorithm, given as input a circuit C, does the following.

   – First it uses LprIO scheme to compress C, i.e. compute $\mathsf{dig}_C \leftarrow \mathsf{LDigest}(\{i, C_i\})$ where C_i represents the i^{th} gate of C.

   – Next it samples $\mathbf{r} \leftarrow \{0,1\}^\lambda$ and gives $\mathsf{prFE.sk}$ for the circuit $\mathrm{F}[\mathbf{r}, \mathsf{dig}_C]$. The circuit $\mathrm{F}[\mathbf{r}, \mathsf{dig}_C]$ on input $(\mathsf{dig}_{\mathbf{x}_{\mathsf{pub}}}, \mathbf{x}_{\mathsf{priv}})$ outputs the following :

     - bGC garbled labels corresponding to $\mathbf{x}_{\mathsf{priv}}$.
     - Obfuscation, $\mathsf{LprIO.obf}_1$, of a circuit that outputs bGC labels corresponding to $\mathbf{x}_{\mathsf{pub}}$ index by index. We only require short digest of $\mathbf{x}_{\mathsf{pub}}$, $\mathsf{dig}_{\mathbf{x}_{\mathsf{pub}}}$, to compute $\mathsf{LprIO.obf}_1$.
     - Obfuscation, $\mathsf{LprIO.obf}_2$, of a circuit that outputs garbled gates of C, gate by gate. We only require short digest of C, dig_C, to compute $\mathsf{LprIO.obf}_2$.

   – Outputs $\mathsf{sk}_C = (\mathbf{r}, \mathsf{prFE.sk})$.

3. The PHprFE encryptor on input mpk and $\mathbf{x} = (\mathbf{x}_{\mathsf{pub}}, \mathbf{x}_{\mathsf{priv}})$ does the following:

   – Run $\mathsf{dig}_{\mathbf{x}_{\mathsf{pub}}} \leftarrow \mathsf{LDigest}(1^\lambda, \{(i, \mathbf{x}_{\mathsf{pub},i})\}_{i\in[L_{\mathsf{pub}}]})$, where $\mathbf{x}_{\mathsf{pub},i} \in \{0,1\}$ is the i-th bit of $\mathbf{x}_{\mathsf{pub}}$.

   – Run $\mathsf{prFE.ct} \leftarrow \mathsf{prFE.Enc}(\mathsf{prFE.mpk}, (\mathsf{dig}_{\mathbf{x}_{\mathsf{pub}}}, \mathbf{x}_{\mathsf{priv}}))$.

   We note that here the input size of prFE is independent of $|\mathbf{x}_{\mathsf{pub}}|$.

4. The PHprFE decryptor does the following:

   – It first runs the prFE decryption to obtain garbled labels corresponding to $\mathbf{x}_{\mathsf{priv}}$ and obfuscations $\mathsf{LprIO.obf}_1$ and $\mathsf{LprIO.obf}_2$.

   – Next it performs LprIO evaluations using $(\{i, \mathbf{x}_{\mathsf{pub},i}\}_i, \mathsf{LprIO.obf}_1)$ and $(\{i, C_i\}_i, \mathsf{LprIO.obf}_2)$ to get the garbled labels corresponding to $\mathbf{x}_{\mathsf{pub}}$ and garbled circuit of C, respectively. Here $\mathbf{x}_{\mathsf{pub},i}$ denotes the i-th bit of $\mathbf{x}_{\mathsf{pub}}$ and C_i denotes the i-th gate of C.

   – Finally, it uses evaluation of the garbling scheme to evaluate the garbled circuit of C on the labels of $\mathbf{x} = (\mathbf{x}_{\mathsf{pub}}, \mathbf{x}_{\mathsf{priv}})$.

Correctness of the above scheme follows from the correctness of the underlying ingredients.

For security we want to show that the $\mathsf{ct} = \mathsf{prFE.ct}(\mathsf{dig}_{\mathbf{x}_{\mathsf{pub}}}, \mathbf{x}_{\mathsf{priv}})$ is pseudorandom. The adversary additionally sees $\mathsf{mpk}, \mathsf{sk}_C = (\mathbf{r}, \mathsf{prFE.sk})$ and we have the guarantee that $C(\mathbf{x}_{\mathsf{pub}}, \mathbf{x}_{\mathsf{priv}})$ is pseudorandom. We prove security via the following three steps.

– First, we invoke prFE security for function $\mathrm{F}[\mathbf{r}, \mathsf{dig}_C]$. By security guarantee of a prFE scheme, it suffices to show pseudorandomness of $\mathrm{F}[\mathbf{r}, \mathsf{dig}_C](\mathsf{dig}_{\mathbf{x}_{\mathsf{pub}}}, \mathbf{x}_{\mathsf{priv}}) = (\mathsf{lab}_{\mathbf{x}_{\mathsf{priv}}}, \mathsf{LprIO.obf}_1, \mathsf{LprIO.obf}_2)$.

- Next, we invoke the security of LprIO scheme using which we wish to show the pseudorandomness of $\mathsf{LprIO.obf}_1$ and $\mathsf{LprIO.obf}_2$. Recall that the security of LprIO scheme states that it suffices to show pseudorandomness of circuit's output on the compressed input domain. Thus, it suffices to show the pseudorandomness of $\mathsf{lab}_{\mathbf{x}_{\mathsf{pub}}}$ and $\tilde{C}$.
  Now, we are left with showing the pseudorandomness of $\mathsf{lab}_{\mathbf{x}_{\mathsf{priv}}}, \mathsf{lab}_{\mathbf{x}_{\mathsf{pub}}}$ and $\tilde{C}$.

- Next, we use bGC simulator to output the garbled values. That is $(\tilde{C}, \mathsf{lab}_{\mathbf{x}_{\mathsf{priv}}}, \mathsf{lab}_{\mathbf{x}_{\mathsf{pub}}}) \leftarrow \mathsf{bGC.Sim}(C(\mathbf{x}_{\mathsf{pub}}, \mathbf{x}_{\mathsf{priv}}))$. Now using the fact that $C(\mathbf{x}_{\mathsf{pub}}, \mathbf{x}_{\mathsf{priv}})$ is pseudorandom, we invoke the *blindness* of bGC scheme to substitute the simulated value with a random string. Hence the security.

While the above overview for the security proof conveys the main idea, it does not work as it is, since we do not have the ideal LprIO whose obfuscated circuit is pseudorandom. Rather, we have an imperfect one, where only the online part of the obfuscated circuit is pseudorandom. This leads to a mismatch with the security guarantee of prFE, which requires the decryption result to be pseudorandom. To resolve the above issue, we change the construction so that the offline part of the LprIO obfuscation, which is not necessarily pseudorandom, is generated during the encryption and included into the ciphertext. This change is possible since the offline part of the obfuscation is not dependent on the circuit being obfuscated, which is unknown to the encryptor. Furthermore, the (pseudorandom) online part of the obfuscated circuit is generated during the decryption, which is aligned with the security guarantee of prFE. A subtlety that arises here is that this change requires strong security guarantee for LprIO, where pseudorandomness of the online part should hold even if its offline part is reused across many invocations. Luckily, we can prove such security for LprIO and thus the entire proof works. We refer the reader to Section 5 for details.

We reason about efficiency next. First we note that it suffices to use a bounded depth prFE scheme supporting circuits of depth $\mathrm{poly}(\lambda, |\mathbf{x}_{\mathsf{priv}}|)$. Next, we note that the size of prFE input and output is also bounded by $\mathrm{poly}(\lambda, |\mathbf{x}_{\mathsf{priv}}|)$. Thus we get a PHprFE scheme with $|\mathsf{mpk}| = |\mathsf{ct}| = |\mathsf{sk}| = \mathrm{poly}(\lambda, |\mathbf{x}_{\mathsf{priv}}|)$. The sizes of the master public key, ciphertexts, and the secret keys of our construction above are all independent from the length of $\mathbf{x}_{\mathsf{pub}}$ and C. However, they still depend on the length of $\mathbf{x}_{\mathsf{priv}}$. In Section 5.4 we show how to remove this dependency from the master public key and the secret keys using a SKE scheme. We get the following theorem.

Theorem 1.13. Assuming LWE and evasive LWE assumptions, there exists a partially hiding pseudorandom FE scheme, for circuit class $\mathcal{C} = \{C : \{0,1\}^{L_{\mathsf{pub}}} \times \{0,1\}^{L_{\mathsf{priv}}} \to \{0,1\}\}$, that satisfies reusable security (as per Definition 5.4) whose sizes of the master public key and the secret key are fixed polynomial $\mathrm{poly}(\lambda)$. Furthermore, the size of the ciphertext is $L_{\mathsf{priv}} + \mathrm{poly}(\lambda)$.

Optimal kpABE via PHprFE. Finally we are ready to show how to obtain an optimal kpABE scheme using an optimal PHprFE scheme. The construction is straightforward: we set the attribute as the public input of PHprFE scheme and the message as private input. Additionally we hardwire a nonce in the circuit C during keygen so that it outputs a pseudorandom value when $C(\mathbf{x}) = 0$, using some private input sd. In more detail,

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

2. The key generator on input msk and a circuit C does the following
   - Defines circuit $C[\mathbf{r}]$, for $\mathbf{r} \leftarrow \{0,1\}^{\lambda}$, which on input $(\mathbf{x}, \mu, \mathsf{sd})$ outputs μ if $C(\mathbf{x}) = 1$ and $\mathsf{PRF}(\mathsf{sd}, \mathbf{r})$ otherwise.
   - Computes a $\mathsf{PHprFE.sk}$ for circuit $C[\mathbf{r}]$.
   - Outputs $\mathsf{sk}_C := (\mathbf{r}, \mathsf{PHprFE.sk})$.

3. The encryptor on input $\mathsf{mpk}, \mathbf{x}$, and μ, first samples a PRF seed $\mathsf{sd} \leftarrow \{0,1\}^{\lambda}$, sets $\mathbf{x}_{\mathsf{pub}} = \mathbf{x}, \mathbf{x}_{\mathsf{priv}} = (\mu, \mathsf{sd})$ and outputs a PHprFE ciphertext $\mathsf{PHprFE.ct} \leftarrow \mathsf{PHprFE.Enc}(\mathsf{PHprFE.mpk}, \mathbf{x}_{\mathsf{pub}}, \mathbf{x}_{\mathsf{priv}})$.

4. The decryptor on input the secret key and the ciphertext simply runs the PHprFE decryption.

The correctness of the scheme follows in a straightforward manner from the correctness of the PHprFE scheme. Note that the PHprFE decryption will output $C[\mathbf{r}](\mathbf{x}, \mu, \mathsf{sd}) = \mu$ for $C(\mathbf{x}) = 1$. Security of the scheme is implied by the security of the underlying PHprFE scheme and the PRF. To prove security, we need to show the pseudorandomness of $\mathsf{ct} = \mathsf{PHprFE.ct}$ when $C(\mathbf{x}) = 0$ – this follows by our usage of a PRF to generate the output, as discussed above. Finally, from the efficiency of a PHprFE scheme and noting that $|\mathbf{x}_{\mathsf{priv}}| = 1 + \lambda$, we get the following theorem.

Theorem 1.14 (Optimal KP-ABE). Under the LWE and Evasive LWE assumption, there exists very selectively secure KP-ABE scheme supporting circuits $\{C : \{0,1\}^\ell \to \{0,1\}\}$ with unbounded depth and single bit message space with

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

We also obtain a *predicate* encryption scheme for unbounded depth circuits with optimal parameters, where predicate encryption allows to hide the entire input of the ciphertext against restricted adversaries. The optimal kpABE implies an optimal cpABE for unbounded depth circuits as well as optimal kpABE for Turing machines as discussed above. For more details, please see Section 6.

1.5 Organization of the Paper

We define the preliminaries used in this work in Section 2. In Section 3 we define and construct the notion of (bounded-depth) compact functional encryption scheme for pseudorandom functionalities (prFE). In Section 4 we define the notion of laconic pseudorandom poly-domain obfuscation scheme (LprIO) and construct it using a blind garbling scheme, a blind batch encryption scheme (constructed in Appendix A) and a pPRIO scheme (constructed in our companion paper [AKY24b]). In Section 5 we define the notion of partially-hiding prFE (denoted as PHprFE) and construct unbounded-depth PHprFE using a bounded depth prFE and a LprIO. In Section 6, we use unbounded-depth PHprFE to construct unbounded-depth kpABE and unbounded-depth PE with optimal parameters. In Section 7 we construct unbounded-depth cpABE with optimal parameters using our optimal kpABE and unbounded-depth prFE (as a special case of unbounded-depth PHprFE).

In Appendix B we show an alternate pathway to construct an unbounded-depth kpABE scheme with sub-optimal parameters *without* using any building block from our companion paper [AKY24b] but still improving the state of the art.

2 Preliminaries

In this section, we define the notation and provide 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 $\|$. Throughout the paper, we usually denote the security parameter by λ. We say a function $f(\lambda)$ is *negligible* if it is $O(\lambda^{-c})$ for all $c > 0$, and we use $\mathsf{negl}(\lambda)$ to denote a negligible function of λ. We say $f(\lambda)$ is *polynomial* if it is $O(\lambda^c)$ for some constant $c > 0$, and we use $\mathrm{poly}(\lambda)$ to denote a polynomial function of λ. We use the abbreviation PPT for probabilistic polynomial-time. We say an event occurs with *overwhelming probability* if its probability is $1 - \mathsf{negl}(\lambda)$. For two distributions X_λ and Y_λ, $X_\lambda \approx_c Y_\lambda$ denotes that they are computationally indistinguishable for any PPT algorithm. 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.

Definition 2.1 (Symmetric Key Encryption with Pseudorandom Ciphertext). A symmetric key encryption scheme for message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda \in [\mathbb{N}]}$ and 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 $m' \in \mathcal{M}_\lambda$.

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

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

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

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

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 $\mathsf{ct}_1 \leftarrow \mathcal{CT}_{\mathsf{SKE}}$.

2.1 Lattice Preliminaries

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

Let

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

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 $(\mathsf{bits}(\mathbf{p}[1]), \ldots, \mathsf{bits}(\mathbf{p}[n]))^\mathsf{T}$, where $\mathsf{bits}(\mathbf{p}[i])$ are m/n bits for each $i \in [n]$. The notation extends column-wise to matrices and it holds that $\mathbf{G}\mathbf{G}^{-1}(\mathbf{P}) = \mathbf{P}$.

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

Lemma 2.2 (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 2.3 (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 2.4 (Smudging Lemma [WWW22]). Let λ be a security parameter. Take any $a \in \mathbb{Z}$ where $|a| \le B$. Suppose $\gamma \ge B\lambda^{\omega(1)}$. 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 2.5 (Leftover Hash Lemma). Fix some $n, m, q \in \mathbb{N}$. The leftover hash lemma states that if $m \ge 2n \log q$, 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 $q^n \sqrt{2^{1-m}}$.

2.1.1 Hardness Assumptions

Assumption 2.6 (The LWE Assumption). Let $n = n(\lambda)$, $m = m(\lambda)$, and $q = q(\lambda) > 2$ be integers and $\chi = \chi(\lambda)$ be a distribution over $\mathbb{Z}_q$. We say that the LWE(n, m, q, χ) hardness assumption holds if for any PPT adversary $\mathcal{A}$ we have

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

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

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

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

Assumption 2.7 (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:

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

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

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

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

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

Remark 2.8. 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 2.9 (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:

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

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

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

2.2 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 2.10. 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

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

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

$$\mathbf{s}^\intercal \mathbf{C} = -\mathbf{e}_{\mathsf{fhe}}^\intercal \mathbf{R}_C + C(\mathbf{x}) \in \mathbb{Z}_q^{1\times m'}.$$

2.3 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 $\|\mathbf{H}_C^\intercal\|, \|\mathbf{H}^\intercal\|_{C,\mathbf{x}} \le (m+2)^d \lceil \log q \rceil$.
  - 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 codomain $\mathbb{Z}_q^{1\times m'}$, then VEval_C is $\mathbb{Z}_q^{(n+1)\times m'}$-valued and

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

2.4 Attribute Based Encryption

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

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

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

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

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

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

Definition 2.12 (Correctness). An ABE scheme for relation family R is correct if for all $\lambda \in \mathbb{N}$, $X \in A_\lambda, Y \in B_\lambda$ such that $R(X, Y) = 1$, and for all messages $\mu \in \mathcal{M}_\lambda$,

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

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

Definition 2.13 (Sel-IND security for ABE). For an ABE scheme $\mathsf{ABE} = \{\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec}\}$ for a relation family $R = \{R_\lambda : A_\lambda \times B_\lambda \to \{0,1\}\}_{\lambda \in [\mathbb{N}]}$ and a message space $\{\mathcal{M}_\lambda\}_{\lambda \in \mathbb{N}}$ and an adversary $\mathcal{A}$, let us define Sel-IND security game as follows.

1. $\mathcal{A}$ outputs the challenge ciphertext attribute $X^\star \in A_\lambda$.
2. **Setup phase:** On input 1^λ, the challenger samples $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and gives mpk to $\mathcal{A}$.
3. **Query phase:** During the game, $\mathcal{A}$ adaptively makes the following queries, in an arbitrary order. $\mathcal{A}$ can make unbounded many key queries, but can make only single challenge query.
   (a) **Key Queries:** $\mathcal{A}$ chooses an input $Y \in B_\lambda$. For each such query, the challenger replies with $\mathsf{sk}_Y \leftarrow \mathsf{KeyGen}(\mathsf{msk}, Y)$.
   (b) **Challenge Query:** At some point, $\mathcal{A}$ submits a pair of equal length messages $(\mu_0, \mu_1) \in \mathcal{M}^2$ to the challenger. The challenger samples a random bit $b \leftarrow \{0,1\}$ and replies to $\mathcal{A}$ with $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, X^\star, \mu_b)$.

   We require that $R(X^\star, Y) = 0$ holds for any Y such that $\mathcal{A}$ makes a key query for Y in order to avoid trivial attacks.
4. **Output phase:** $\mathcal{A}$ outputs a guess bit b' as the output of the experiment.

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

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

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

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

Definition 2.14 ($\mathsf{Sel\text{-}INDr}$ **security for** ABE)**.** We define $\mathsf{Sel\text{-}INDr}$ security game similarly to $\mathsf{Sel\text{-}IND}$ security game except that the adversary $\mathcal{A}$ chooses single message μ instead of (μ_0, μ_1) at the challenge phase and the challenger returns $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, X^\star, \mu)$ if $b = 0$ and a random ciphertext $\mathsf{ct} \leftarrow \mathcal{CT}$ from a ciphertext space $\mathcal{CT}$ if $b = 1$. Here, we assume that uniform sampling from the ciphertext space $\mathcal{CT}$ is possible without any parameter other than the security parameter λ. We define the advantage $\mathsf{Adv}^{\mathsf{Sel\text{-}INDr}}_{\mathsf{ABE},\mathcal{A}}(1^\lambda)$ of the adversary $\mathcal{A}$ accordingly and say that the scheme satisfies $\mathsf{Sel\text{-}INDr}$ security if the quantity is negligible.

We also consider the very selective notion of security.

Definition 2.15 ($\mathsf{VerSel\text{-}IND}$ **security for** ABE)**.** We define $\mathsf{VerSel\text{-}IND}$ security game similarly to $\mathsf{Sel\text{-}IND}$ security game except that the adversary $\mathcal{A}$ outputs the key queries $Y_1, \ldots, Y_Q$, where Q is the number of key queries made by $\mathcal{A}$, along with the challenge ciphertext attribute $X^\star$ in the beginning of the security game. We define the advantage $\mathcal{A}^{\mathsf{VerSel\text{-}IND}}_{\mathsf{ABE},\mathcal{A}}(1^\lambda)$ of the adversary $\mathcal{A}$ accordingly and say that the scheme satisfies $\mathsf{VerSel\text{-}IND}$ security if the quantity is negligible.

Definition 2.16 ($\mathsf{VerSel\text{-}INDr}$ **security for** ABE)**.** We define $\mathsf{VerSel\text{-}IND}$ security game similarly to $\mathsf{Sel\text{-}INDr}$ security game except that the adversary $\mathcal{A}$ outputs the key queries $Y_1, \ldots, Y_Q$, where Q is the number of key queries made by $\mathcal{A}$, along with the challenge ciphertext attribute $X^\star$ in the beginning of the security game. We define the advantage $\mathcal{A}^{\mathsf{VerSel\text{-}INDr}}_{\mathsf{ABE},\mathcal{A}}(1^\lambda)$ of the adversary $\mathcal{A}$ accordingly and say that the scheme satisfies $\mathsf{VerSel\text{-}INDr}$ security if the quantity is negligible.

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

$$R^{\mathsf{cpABE}}_\lambda(C, \mathbf{x}) = C(\mathbf{x}).$$

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

$$R^{\mathsf{kpABE}}_\lambda(\mathbf{x}, C) = C(\mathbf{x}).$$

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

2.5 Predicate Encryption

In this section we define predicate encryption (PE) scheme. The syntax and correctness is same as that of the ABE scheme in Section 2.4 except that we do not input the ciphertext attribute into the decryption algorithm, i.e $\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_Y, Y, \mathsf{ct}) \to \mu$ or $\perp$. We set $A_\lambda = \{0,1\}^\ell$ and $B_\lambda = \mathcal{C}_{\ell(\lambda),d(\lambda)}$. We also define $R_\lambda : A^{\mathsf{kpABE}}_\lambda \times B^{\mathsf{kpABE}}_\lambda \to \{0,1\}$ as

$$R_\lambda(\mathbf{x}, C) = C(\mathbf{x}).$$

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

Definition 2.17 (Sel-IND **security for** PE)**.** For a PE scheme $\mathsf{PE} = \{\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec}\}$ for a relation family $R = \{R_\lambda : A_\lambda \times B_\lambda \to \{0,1\}\}_{\lambda \in [\mathbb{N}]}$ and a message space $\{\mathcal{M}_\lambda\}_{\lambda \in \mathbb{N}}$ and an adversary $\mathcal{A}$, the Sel-IND security game is defined exactly as in Definition 2.13 except that $\mathcal{A}$ outputs two challenge attributes (X_0^*, X_1^*) in Step 1 and the challenger returns $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, X_b^*, \mu_b)$ in Step 3(a) of the security game. We require that $R(X_0^\star, Y) = R(X_1^\star, Y) = 0$ holds for any Y such that $\mathcal{A}$ makes a key query for Y in order to avoid trivial attacks. We define the advantage $\mathsf{Adv}_{\mathsf{PE},\mathcal{A}}^{\mathsf{Sel\text{-}IND}}(1^\lambda)$ of $\mathcal{A}$ in the above game as

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

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

Definition 2.18 (VerSel-IND **security for** PE)**.** We define VerSel-IND security game similarly to Sel-IND security game except that the adversary $\mathcal{A}$ outputs the key queries $Y_1, \ldots, Y_Q$, where Q is the number of key queries made by $\mathcal{A}$, along with the challenge ciphertext attribute (X_0^*, X_1^*) in the beginning of the security game. We define the advantage $\mathcal{A}_{\mathsf{PE},\mathcal{A}}^{\mathsf{VerSel\text{-}IND}}(1^\lambda)$ of the adversary $\mathcal{A}$ accordingly and say that the scheme satisfies VerSel-IND security if the quantity is negligible.

2.6 Blind Garbled Circuit

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

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

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

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

A garbling scheme satisfies the following properties.

Definition 2.19 (Correctness). A garbling scheme is said to be correct if for any circuit $C \in \mathcal{C}$ and any input $x \in \{0,1\}^{\ell_{\mathsf{in}}}$, the following holds

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

Definition 2.20 (Simulation Security). A garbling scheme is said to satisfy simulation security if for any circuit $C \in \mathcal{C}$ and any input $\mathbf{x} \in \{0,1\}^{\ell_{\mathsf{in}}}$, the following holds

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

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

Definition 2.21 (Blindness). [BLSV18] A garbling scheme $(\mathsf{Garble}, \mathsf{Eval}, \mathsf{Sim})$ is called blind if the distribution $\mathsf{Sim}(1^\lambda, 1^{|C|}, 1^{\ell_{\mathsf{in}}}, \mathbf{y})$ for $\mathbf{y} \leftarrow \{0,1\}^{\ell_{\mathsf{out}}}$, representing the output of the simulator on a completely uniform output, is indistinguishable from a completely uniform bit string. (Note that the distinguisher must not know the random output value that was used for the simulation.)

Definition 2.22 (Decomposability). We note that the $\mathsf{Garble}(1^\lambda, 1^{\ell_{\mathsf{in}}}, 1^{\ell_{\mathsf{out}}}, C)$ algorithm can be decomposed, using shared randomness st, as follows : (i) $\mathsf{Garble}_i(1^\lambda, C_i; \mathsf{st})$ for $i \in [|C|]$, where $\mathsf{Garble}_i(1^\lambda, C_i)$ outputs the garbling of i-th gate of the circuit C (denoted by C_i) and (ii) $\mathsf{Garble}_{\mathsf{inp}}(1^\lambda, 1^{\ell_{\mathsf{in}}}, 1^{\ell_{\mathsf{out}}}; \mathsf{st}) = \mathsf{lab}$ which outputs $2 \cdot \ell_{\mathsf{in}}$ labels.

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

Theorem 2.23. [BLSV18] Assume that one-way function exists. Then, there exists a blind garbled circuits scheme.

2.7 Blind Batch Encryption

Here we provide the definition of a batch encryption scheme largely adapted from [BLSV18]. A batch encryption scheme with the message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_\lambda$ consists of the following algorithms.

$\mathsf{Setup}(1^\lambda, 1^N) \to \mathsf{crs}$. The setup algorithm takes as input the security parameter λ and key length N, and outputs a common reference string crs.

$\mathsf{Gen}(\mathsf{crs}, \mathbf{x}) \to h$. The generation algorithm takes as input the common reference string crs and a secret key $\mathbf{x} \in \{0,1\}^N$. It outputs a public key h.

$\mathsf{SingleEnc}(\mathsf{crs}, h, i, (m_0, m_1)) \to \mathsf{ct}$. The encryption algorithm takes as input a common reference string crs, the public key h, an index $i \in [N]$, and a message $(m_0, m_1) \in \mathcal{M}^2$ and outputs a (single) ciphertext ct.

$\mathsf{SingleDec}(\mathsf{crs}, \mathbf{x}, i, \mathsf{ct}) \to m$. The encryption algorithm takes as input a common reference string crs, the secret key $\mathbf{x}$, an index $i \in [N]$, and a (single) ciphertext ct and outputs a message $m \in \mathcal{M}$.

In [BLSV18], they define additional algorithms Enc and Dec, which can be defined using $\mathsf{SingleEnc}$ and $\mathsf{SingleDec}$ above. We omit the definition of these algorithms since they are not used in our paper.

Definition 2.24 (Correctness.). A batch encryption scheme is said to be correct if for any $\lambda, N \in \mathbb{N}$, secret key $\mathbf{x} \in \{0,1\}^N$, $i \in [N]$, $(m_0, m_1) \in \mathcal{M}^2$, $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^N)$, and $h \leftarrow \mathsf{Gen}(\mathsf{crs}, \mathbf{x})$ it holds that

$$\Pr[m = m_{x_i} \mid m = \mathsf{SingleDec}\left(\mathsf{crs}, \mathbf{x}, i, \mathsf{SingleEnc}(\mathsf{crs}, h, i, (m_0, m_1))\right)] = 1,$$

where x_i is the i-th bit of $\mathbf{x}$.

Definition 2.25 (Succinctness). A batch encryption scheme is α-succinct if letting $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^N)$, $h = \mathsf{Gen}(\mathsf{crs}, \mathbf{x})$ for some $\mathbf{x} \in \{0,1\}^N$, it holds that $|h| \le \alpha N$. It is said fully succinct if $|h| \le p(\lambda)$ for some fixed polynomial $p(\lambda)$.

Definition 2.26 (Security). A batch encryption scheme is said to be secure if for any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that the following holds

$$\Pr\left[\begin{array}{ll} \beta' = \beta : & \begin{array}{l} (1^N, \mathbf{x} \in \{0,1\}^N, i \in [N]) \leftarrow \mathcal{A}(1^\lambda); \\ \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^N); \\ \left(\mathbf{m}^{(0)} = (m_0^{(0)}, m_1^{(0)}), \mathbf{m}^{(1)} = (m_0^{(1)}, m_1^{(1)})\right) \leftarrow \mathcal{A}(\mathsf{crs}); \\ h \leftarrow \mathsf{Gen}(\mathsf{crs}, \mathbf{x}), \beta \leftarrow \{0,1\}, \mathsf{ct}_\beta \leftarrow \mathsf{SingleEnc}(\mathsf{crs}, h, i, \mathbf{m}^{(\beta)}); \\ \beta' \leftarrow \mathcal{A}(\mathsf{crs}, \mathsf{ct}_\beta) \end{array} \end{array}\right] \le \frac{1}{2} + \mathsf{negl}(\lambda)$$

where we require that $\mathbf{m}^{(0)}, \mathbf{m}^{(1)} \in \mathcal{M}^2$ and $m_{x_i}^{(0)} = m_{x_i}^{(1)}$.

We require somewhat stronger blindness condition than that defined in [BLSV18].

Definition 2.27 (Strong Blindness). A batch encryption scheme is said to satisfy blindness if for any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that the following holds

$$\Pr\left[\beta' = \beta : \begin{array}{l} (1^N, \mathbf{x} \in \{0,1\}^N, i \in [N]) \leftarrow \mathcal{A}(1^\lambda); \\ \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^N), h \leftarrow \mathsf{Gen}(\mathsf{crs}, \mathbf{x}); \\ \mathbf{m} \leftarrow \mathcal{M}^2, \beta \leftarrow \{0,1\}; \\ \mathsf{ct}_0 \leftarrow \mathsf{SingleEnc}(\mathsf{crs}, h, i, \mathbf{m}), \mathsf{ct}_1 \leftarrow \mathcal{CT}; \\ \beta' \leftarrow \mathcal{A}(\mathsf{crs}, \mathsf{ct}_\beta) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $\mathcal{CT}$ is the ciphertext space of the scheme.

Remark 2.28. Here, we compare strong blindness defined above with the blindness defined in [BLSV18]. In [BLSV18], they divide a ciphertext ct into an "offline part" subct_1 and "online part" subct_2, where subct_1 only depends on the encryption randomness and the CRS, whereas subct_2 may depend on h, i, and $\mathbf{m}$ additionally. They then define blindness as the security notion that essentially requires that subct_2 is pseudorandom, whereas subct_1 may not be. The above security notion is stronger than theirs in that we require the entire ciphertext to be pseudorandom rather than part of it. In other words, we can see our definition as more stringent version of their blindness notion where we require subct_1 to be an empty string.

2.8 Poly-Input Obfuscation for Pseudorandom Functionalities

In this section we give the definition of poly-input indistinguishability obfuscation for pseudorandom functionalities (pPRIO), adapted from [AKY24b].

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 $\mathsf{size}(C) \leq L$ 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 $y \in [M]$.

Next, we define the properties of a pPRIO scheme.

Definition 2.29 (Correctness). 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:

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

where the probability is taken over the coin-tosses of the obfuscator Obf.

Definition 2.30 (Security). 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 pPRIO scheme is secure with respect to the sampler class $\mathcal{SC}$ if for every PPT sampler $\mathsf{Samp} \in \mathcal{SC}$ the following holds.

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

where $\Delta_i^j \leftarrow [M_j]$ for $j \in [Q], i \in [N_j]$, $(\mathsf{obf}_{\mathsf{off}}, \mathsf{st}) \leftarrow \mathsf{ObfOff}(1^\lambda, 1^L)$, $\mathsf{obf}_{\mathsf{on}}^j \leftarrow \mathsf{ObfOn}(\mathsf{st}, C^j)$ 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.

Remark 2.31. It is shown in [AMYY25, BDJ+24] that 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. For simplicity, we sometimes will treat as if there was pPRIO that is secure for all the samplers.

Theorem 2.32 ([AKY24b]). Assuming LWE and evasive LWE assumptions, there exists a secure pPRIO scheme satisfying

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

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 $L = L(\lambda)$.

3 Functional Encryption for Pseudorandom Functionalities

3.1 Definition

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

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

$\mathsf{Setup}(1^\lambda, \mathsf{prm}) \to (\mathsf{mpk}, \mathsf{msk})$. The setup algorithm takes as input the security parameter λ and a parameter prm and outputs a master public key mpk and a master secret key msk[4].

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

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

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

Definition 3.1 (Correctness). A prFE scheme is said to satisfy *perfect* correctness if for all prm, any input $x \in \mathcal{X}_{\mathsf{prm}}$ and function $f \in \mathcal{F}_{\mathsf{prm}}$, we have

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

We define our security 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.

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

Definition 3.2 (prCT **Security).** For a prFE scheme for function family $\{\mathcal{F}_{\mathsf{prm}} = \{f : \mathcal{X}_{\mathsf{prm}} \to \mathcal{Y}_{\mathsf{prm}}\}\}_{\mathsf{prm}}$, parameter $\mathsf{prm} = \mathsf{prm}(\lambda)$, let Samp be a PPT algorithm that on input 1^λ, outputs

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

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:

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

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

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

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

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

Remark 3.3. 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.

Remark 3.4. We note that the above security definition is in the multi-challenge flavor. One may wonder whether single-challenge version of the definition where $Q_{\mathsf{msg}} = 1$ implies the multi-challenge version or not. However, unlike the standard security notions for public key primitives (e.g., indistinguishability security for functional encryption [GGH$^+$16]), this does not seem to be the case. This is because the standard hybrid argument to prove the multi-challenge security from the single challenge security where we replace the ciphertext to be random one-by-one fails. To see why, recall that in these hybrids, we simulate some of the ciphertexts honestly, while we try to change a particular honest ciphertext to be a random one. To generate honest ciphertexts, we may have to know the corresponding plaintexts. However, this may ruin the precondition for invoking the single challenge security for the target ciphertext, since knowing some of the inputs (say, x_1) may make the output (say, $f_1(x_2)$) not pseudorandom any more when the inputs are correlated with each other.

Definition 3.5 (Compactness). A prFE scheme is said to be compact if for any input message $x \in \mathcal{X}$, the running time of the encryption algorithm is polynomial in the security parameter and the size of x. In particular, it does not depend on the circuit description size or the output length of any function f supported by the scheme.

3.2 Construction

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

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

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

- Sample appropriate parameters $q, n, m, \tau, \sigma, \sigma_{\mathbf{B}}, B$ and M such that M divides q, as in Equation (3)[5].
- Samples appropriate parameters as in Equation (3).
- Set $\mathrm{L_X} = m(\lambda + \mathrm{L})(n+1)\lceil \log q \rceil$, sample $\mathbf{A}_{\mathsf{att}} \leftarrow \mathbb{Z}_q^{(n+1)\times(\mathrm{L_X}+1)m}$ and $(\mathbf{B}, \mathbf{B}_\tau^{-1}) \leftarrow \mathsf{TrapGen}(1^{n+1}, 1^{mw}, q)$, where $w \in O(\log q)$.
- Output $\mathsf{mpk} := (\mathbf{A}_{\mathsf{att}}, \mathbf{B}, M)$ and $\mathsf{msk} := \mathbf{B}_\tau^{-1}$.

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

- Sample $\mathbf{r} \leftarrow \{0,1\}^\lambda$ and define function $\mathrm{F} = \mathrm{F}[f, \mathbf{r}]$ with $f, \mathbf{r}$ hardwired as follows[6]:
  On input $(\mathbf{x}, \mathsf{sd})$, compute and output $f(\mathbf{x})\lfloor q/2 \rceil + \mathsf{PRF}(\mathsf{sd}, \mathbf{r}) \in \mathbb{Z}_q^{1\times\ell}$.
- Parse $\mathrm{F}[f,\mathbf{r}](\mathbf{x},\mathsf{sd}) = M \cdot f_{\mathsf{high}}(\mathbf{x},\mathsf{sd}) + f_{\mathsf{low}}(\mathbf{x},\mathsf{sd})$, where $f_{\mathsf{high}}(\mathbf{x},\mathsf{sd}) \in [0, q/M]^\ell$ and $f_{\mathsf{low}}(\mathbf{x},\mathsf{sd}) \in [0, M-1]^\ell$.Using the fact that the PRF and $f(\mathbf{x})$ can be computed by a circuit of depth at most $\mathsf{dep}(\lambda) = \mathrm{poly}(\lambda)$, the function $\mathrm{F}[f,\mathbf{r}]$ can be computed by a circuit of depth at most $d = \mathrm{poly}(\mathsf{dep})$.
- Define functions $\mathrm{F}_{\mathsf{high}} := M \cdot f_{\mathsf{high}}$ and $\mathrm{F}_{\mathsf{low}} := M \cdot f_{\mathsf{low}}$, which on input $(\mathbf{x}, \mathsf{sd})$ outputs $M \cdot f_{\mathsf{high}}(\mathbf{x},\mathsf{sd})$ and $M \cdot f_{\mathsf{low}}(\mathbf{x},\mathsf{sd})$, respectively. We note that these functions can be computed by a circuit of depth at most $d = \mathrm{poly}(\mathsf{dep})$.
- Define $\mathsf{VEval}_{\mathsf{high}} = \mathsf{MakeVEvalCkt}(n, m, q, \mathrm{F}_{\mathsf{high}})$ and $\mathsf{VEval}_{\mathsf{low}} = \mathsf{MakeVEvalCkt}(n, m, q, \mathrm{F}_{\mathsf{low}})$. From Lemma 2.11, 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 $\mathbf{H}_{\mathbf{A}_{\mathsf{att}}}^{\mathrm{F}_{\mathsf{high}}} = \mathsf{MEvalC}(\mathbf{A}_{\mathsf{att}}, \mathsf{VEval}_{\mathsf{high}})$, $\mathbf{H}_{\mathbf{A}_{\mathsf{att}}}^{\mathrm{F}_{\mathsf{low}}} = \mathsf{MEvalC}(\mathbf{A}_{\mathsf{att}}, \mathsf{VEval}_{\mathsf{low}}) \in \mathbb{Z}_q^{(\mathrm{L_X}+1)m\times\ell}$.
- Compute $\mathbf{A}_{\mathsf{high}} = \mathbf{A}_{\mathsf{att}} \cdot \mathbf{H}_{\mathbf{A}_{\mathsf{att}}}^{\mathrm{F}_{\mathsf{high}}}$ and $\mathbf{A}_{\mathsf{low}} = \mathbf{A}_{\mathsf{att}} \cdot \mathbf{H}_{\mathbf{A}_{\mathsf{att}}}^{\mathrm{F}_{\mathsf{low}}}$.
- Compute
$$\mathbf{A}_{\mathrm{F}} = M \cdot \left\lfloor \frac{\mathbf{A}_{\mathsf{high}}}{M} \right\rfloor + \left\lfloor \frac{\mathbf{A}_{\mathsf{low}}}{M} \right\rfloor$$
  and sample $\mathbf{K} \leftarrow \mathbf{B}_\tau^{-1}(\mathbf{A}_{\mathrm{F}})$.
- Output $\mathsf{sk}_f = (\mathbf{K}, \mathbf{r})$.

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

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

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

– Compute a BGG^+ encoding as follows.

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

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

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

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

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

– Compute

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

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

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

Parameters. We set our parameters as follows.

$$\beta = 2^{O(\mathsf{dep}\cdot\log\lambda)},\ q = 2^{12\lambda}\beta,\ M = 2^{4\lambda}\beta,\ n = \mathrm{poly}(\lambda, \mathsf{dep}),\ m = O(n \log q),\ B = 2^{10\lambda}\beta,$$
$$\tau = O\left(\sqrt{(n+1)\log q}\right)\ \ \sigma_{\mathbf{s}} = \sigma = 2^{2\lambda},\ \sigma_{\mathbf{B}} = 2^{9\lambda}\beta,\ \sigma_1 = 2^{8\lambda+O(1)}\beta/\mathrm{poly}(\lambda). \tag{3}$$

Efficiency. Using the above set parameters, we have

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

Correctness We analyze the correctness of our scheme below.

– First, we note that

$$\begin{aligned}
\mathbf{c}_{\mathsf{att}}^{\mathsf{T}} \cdot \mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}} &= (\mathbf{s}^{\mathsf{T}}(\mathbf{A}_{\mathsf{att}} - \mathsf{bits}(1,\mathbf{X}) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{att}}^{\mathsf{T}})\mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}} \\
&= \mathbf{s}^{\mathsf{T}}\mathbf{A}_{\mathsf{att}}\mathbf{H}_{\mathbf{A}_{\mathsf{att}}}^{\mathrm{F}_{\mathsf{high}}} - \mathbf{s}^{\mathsf{T}}\mathsf{VEval}_{\mathsf{high}}(\mathsf{bits}(\mathbf{X})) + \mathbf{e}_{\mathsf{att}}^{\mathsf{T}}\mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}} \\
&= \mathbf{s}^{\mathsf{T}}\mathbf{A}_{\mathsf{high}} - \mathrm{F}_{\mathsf{high}}(\mathbf{x}, \mathsf{sd}) + \mathbf{e}_{\mathsf{fhe}}^{\mathsf{T}}\mathbf{R}_{\mathsf{high}} + \mathbf{e}_{\mathsf{att}}^{\mathsf{T}}\mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}} \\
\implies \left\lfloor \frac{\mathbf{c}_{\mathsf{att}}^{\mathsf{T}} \cdot \mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}}}{M} \right\rceil &= \left\lfloor \frac{\mathbf{s}^{\mathsf{T}}\mathbf{A}_{\mathsf{high}} - M \cdot f_{\mathsf{high}}(\mathbf{x}, \mathsf{sd}) + \mathbf{e}_{\mathsf{fhe}}^{\mathsf{T}}\mathbf{R}_{\mathsf{high}} + \mathbf{e}_{\mathsf{att}}^{\mathsf{T}}\mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}}}{M} \right\rceil \\
&= \left\lfloor \frac{\mathbf{s}^{\mathsf{T}}\mathbf{A}_{\mathsf{high}} + \mathbf{e}_{\mathsf{fhe}}^{\mathsf{T}}\mathbf{R}_{\mathsf{high}} + \mathbf{e}_{\mathsf{att}}^{\mathsf{T}}\mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}}}{M} \right\rceil - f_{\mathsf{high}}(\mathbf{x}, \mathsf{sd})
\end{aligned} \tag{4}$$

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 2.11, 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 Section 2.3,

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

where $d_{\mathsf{VEval}_{\mathsf{high}}}$ denotes the depth of the circuit $\mathsf{VEval}_{\mathsf{high}}$. So we have $\left\|\mathbf{e}^{\mathsf{T}}_{\mathsf{fhe}}\mathbf{R}_{\mathsf{high}} + \mathbf{e}^{\mathsf{T}}_{\mathsf{att}}\mathbf{H}^{\mathrm{F}_{\mathrm{high}}}_{\mathbf{A}_{\mathrm{att}},\mathbf{X}}\right\| \le 2^{2\lambda+1}\sqrt{\lambda}\beta \le 2^{3\lambda}\beta < M$. Using this in Equation (4),

$$\begin{aligned}\left\lfloor \frac{\mathbf{c}^{\mathsf{T}}_{\mathsf{att}}\cdot\mathbf{H}^{\mathrm{F}_{\mathrm{high}}}_{\mathbf{A}_{\mathrm{att}},\mathbf{X}}}{M}\right\rfloor &= \left\lfloor \frac{\mathbf{s}^{\mathsf{T}}\mathbf{A}_{\mathsf{high}}}{M}\right\rfloor - f_{\mathsf{high}}(\mathbf{x},\mathsf{sd}) + \mathsf{err}_{\mathsf{high}} \\ &= \mathbf{s}^{\mathsf{T}}\left\lfloor \frac{\mathbf{A}_{\mathsf{high}}}{M}\right\rfloor + \mathbf{e}^{\mathsf{T}}_{\mathbf{s},\mathsf{high}} - f_{\mathsf{high}}(\mathbf{x},\mathsf{sd}) + \mathsf{err}^{\mathsf{T}}_{\mathsf{high}} \end{aligned}\tag{5}$$

where $\mathsf{err}^{\mathsf{T}}_{\mathsf{high}} \in \{0,1\}^{\ell}$, is the rounding error which is 1 if $\left\|(\mathbf{s}^{\mathsf{T}}\mathbf{A}_{\mathsf{high}})^{\mathsf{T}} + \mathbf{e}^{\mathsf{T}}_{\mathsf{fhe}}\mathbf{R}_{\mathsf{high}} + \mathbf{e}^{\mathsf{T}}_{\mathsf{att}}\mathbf{H}^{\mathrm{F}_{\mathrm{high}}}_{\mathbf{A}_{\mathrm{att}},\mathbf{X}}\right\| \ge M$ and 0 otherwise, and $\|\mathbf{e}_{\mathbf{s},\mathsf{high}}\| \le (n+1)\cdot\|\mathbf{s}\|$. To see the latter, we use the fact that $\lfloor \mathbf{s}^{\mathsf{T}}X\rfloor - \mathbf{s}^{\mathsf{T}}\lfloor X\rfloor = \lfloor \mathbf{s}^{\mathsf{T}}X - \mathbf{s}^{\mathsf{T}}\lfloor X\rfloor\rfloor = \lfloor \mathbf{s}^{\mathsf{T}}(X-\lfloor X\rfloor)\rfloor$, where $X - \lfloor X\rfloor < 1$. So $\mathbf{e}^{\mathsf{T}}_{\mathbf{s},\mathsf{high}} = \left\lfloor \mathbf{s}^{\mathsf{T}}\left(\frac{\mathbf{A}_{\mathsf{high}}}{M} - \left\lfloor\frac{\mathbf{A}_{\mathsf{high}}}{M}\right\rfloor\right)\right\rfloor$ 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 (5), we get

$$\left\lfloor \frac{\mathbf{c}^{\mathsf{T}}_{\mathsf{att}}\cdot\mathbf{H}^{\mathrm{F}_{\mathrm{low}}}_{\mathbf{A}_{\mathrm{att}},\mathbf{X}}}{M}\right\rfloor = \mathbf{s}^{\mathsf{T}}\left\lfloor\frac{\mathbf{A}_{\mathsf{low}}}{M}\right\rfloor + \mathbf{e}^{\mathsf{T}}_{\mathbf{s},\mathsf{low}} - f_{\mathsf{low}}(\mathbf{x},\mathsf{sd}) + \mathsf{err}^{\mathsf{T}}_{\mathsf{low}}\tag{6}$$

where $\mathsf{err}^{\mathsf{T}}_{\mathsf{low}} \in \{0,1\}^{\ell}$ and $\|\mathbf{e}_{\mathbf{s},\mathsf{low}}\| \le (n+1)\cdot\|\mathbf{s}\|$. Using Equation (5) and Equation (6), we get

$$\begin{aligned}&M\cdot\left\lfloor \frac{\mathbf{c}^{\mathsf{T}}_{\mathsf{att}}\cdot\mathbf{H}^{\mathrm{F}_{\mathrm{high}}}_{\mathbf{A}_{\mathrm{att}},\mathbf{X}}}{M}\right\rfloor + \left\lfloor \frac{\mathbf{c}^{\mathsf{T}}_{\mathsf{att}}\cdot\mathbf{H}^{\mathrm{F}_{\mathrm{low}}}_{\mathbf{A}_{\mathrm{att}},\mathbf{X}}}{M}\right\rfloor \\ &= \mathbf{s}^{\mathsf{T}}\left(M\cdot\left\lfloor\frac{\mathbf{A}_{\mathsf{high}}}{M}\right\rfloor + \left\lfloor\frac{\mathbf{A}_{\mathsf{low}}}{M}\right\rfloor\right) - (M\cdot f_{\mathsf{high}}(\mathbf{x},\mathsf{sd}) + f_{\mathsf{low}}(\mathbf{x},\mathsf{sd})) + \mathsf{err} \\ &= \mathbf{s}^{\mathsf{T}}\left(M\cdot\left\lfloor\frac{\mathbf{A}_{\mathsf{high}}}{M}\right\rfloor + \left\lfloor\frac{\mathbf{A}_{\mathsf{low}}}{M}\right\rfloor\right) - \mathrm{F}[f,\mathbf{r}](\mathbf{x},\mathsf{sd}) + \mathsf{err}\end{aligned}\tag{7}$$

where

$$\begin{aligned}\mathsf{err} &= M\cdot\mathbf{e}^{\mathsf{T}}_{\mathbf{s},\mathsf{high}} + \mathbf{e}^{\mathsf{T}}_{\mathbf{s},\mathsf{low}} + M\cdot\mathsf{err}_{\mathsf{high}} + \mathsf{err}_{\mathsf{low}} \\ &= M\cdot\left(\mathbf{s}^{\mathsf{T}}\left\lfloor\frac{\mathbf{A}_{\mathsf{high}}}{M}\right\rfloor - \left\lfloor\mathbf{s}^{\mathsf{T}}\frac{\mathbf{A}_{\mathsf{high}}}{M}\right\rfloor\right) + \left(\mathbf{s}^{\mathsf{T}}\left\lfloor\frac{\mathbf{A}_{\mathsf{low}}}{M}\right\rfloor - \left\lfloor\mathbf{s}^{\mathsf{T}}\frac{\mathbf{A}_{\mathsf{low}}}{M}\right\rfloor\right) + M\cdot\mathsf{err}_{\mathsf{high}} + \mathsf{err}_{\mathsf{low}}\end{aligned}\tag{8}$$

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

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

– Next, we note that

$$\mathbf{c}^{\mathsf{T}}_{\mathbf{B}}\cdot\mathbf{K} = \mathbf{s}^{\mathsf{T}}\left(M\cdot\left\lfloor\frac{\mathbf{A}_{\mathsf{high}}}{M}\right\rfloor + \left\lfloor\frac{\mathbf{A}_{\mathsf{low}}}{M}\right\rfloor\right) + \mathbf{c}^{\mathsf{T}}_{\mathbf{B}}\cdot\mathbf{K}.\tag{9}$$

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

– Using Equations (7) and (9), 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.

3.3 Security Proof for Pseudorandom Functionalities

Theorem 3.6. Let $\mathcal{SC}_{\mathsf{prFE}}$ be a sampler class for prFE. Assuming LWE (Assumption 2.6) and private coin Evasive LWE (Assumption 2.7) with respect to the sampler class that contains all $\mathsf{Samp}_{\mathsf{evs}}(1^\lambda)$ induced by $\mathsf{Samp}_{\mathsf{prFE}} \in \mathcal{SC}_{\mathsf{prFE}}$ as defined in Figure 1, our prFE scheme satisfies prCT security with respect to $\mathcal{SC}_{\mathsf{prFE}}$ as defined in Definition 3.2.

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

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

To prove the prCT security as per Definition 3.2, we show

$$\begin{pmatrix} \mathsf{mpk} = (\mathbf{A}_{\mathsf{att}}, \mathbf{B}, M),\ \mathsf{aux}_{\mathcal{A}},\ \mathbf{C}_{\mathbf{B}} = \mathbf{S}\mathbf{B} + \mathbf{E}_{\mathbf{B}}, \\ \{\mathbf{X}_j = \mathbf{A}_{\mathsf{fhe},j}\mathbf{R}_j - (\mathbf{x}_j, \mathsf{sd}_j) \otimes \mathbf{G}\}_{j \in [Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}_{\mathsf{att},j}^{\mathsf{T}} = \mathbf{s}_j^{\mathsf{T}}(\mathbf{A}_{\mathsf{att}} - \mathsf{bits}(1, \mathbf{X}_j) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{att},j}^{\mathsf{T}}\}_{j \in [Q_{\mathsf{msg}}]}, \\ \{\mathbf{K}_k, \mathbf{r}_k\}_{k \in [Q_{\mathsf{key}}]} \end{pmatrix} \approx_c \begin{pmatrix} \mathsf{mpk} = (\mathbf{A}_{\mathsf{att}}, \mathbf{B}, M),\ \mathsf{aux}_{\mathcal{A}},\ \mathbf{C}_{\mathbf{B}} \leftarrow \mathbb{Z}_q^{Q_{\mathsf{msg}} \times mw}, \\ \{\mathbf{X}_j \leftarrow \mathbb{Z}_q^{(n+1) \times m(\lambda + \mathrm{L})}\}_{j \in [Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}_{\mathsf{att},j} \leftarrow \mathbb{Z}_q^{(\mathrm{L}_{\mathrm{X}}+1)m}\}_{j \in [Q_{\mathsf{msg}}]}, \\ \{\mathbf{K}_k, \mathbf{r}_k\}_{k \in [Q_{\mathsf{key}}]} \end{pmatrix} \tag{10}$$

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

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

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

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 (3)[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 1: Description of the Sampler for Evasive LWE

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

By Lemma 2.9, to prove Equation (10) assuming evasive LWE, it suffices to show

$$\begin{pmatrix} \mathsf{aux}_2,\ \mathbf{B},\ \mathbf{C}_{\mathbf{B}} = \mathbf{S}\mathbf{B} + \mathbf{E}_{\mathbf{B}}, \\ \{\mathbf{X}_j = \mathbf{A}_{\mathsf{fhe},j}\mathbf{R}_j - (\mathbf{x}_j,\mathsf{sd}_j)\otimes\mathbf{G}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}^{\mathsf{T}}_{\mathsf{att},j} = \mathbf{s}_j^{\mathsf{T}}(\mathbf{A}_{\mathsf{att}} - \mathsf{bits}(1,\mathbf{X}_j)\otimes\mathbf{G}) + \mathbf{e}^{\mathsf{T}}_{\mathsf{att},j}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \mathbf{C}_{\mathbf{P}} = \mathbf{S}\mathbf{P} + \mathbf{E}_{\mathbf{P}} \end{pmatrix} \approx_c \begin{pmatrix} \mathsf{aux}_2,\ \mathbf{B},\ \mathbf{C}_{\mathbf{B}} \leftarrow \mathbb{Z}_q^{Q_{\mathsf{msg}}\times mw}, \\ \{\mathbf{X}_j \leftarrow \mathbb{Z}_q^{(n+1)\times m(\lambda+\mathrm{L})}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}_{\mathsf{att},j} \leftarrow \mathbb{Z}_q^{(\mathrm{L_X}+1)m}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \mathbf{C}_{\mathbf{P}} \leftarrow \mathbb{Z}_q^{Q_{\mathsf{msg}}\times\ell\cdot Q_{\mathsf{key}}} \end{pmatrix} \tag{11}$$

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

$$\mathbf{C}_{\mathbf{B}} = \begin{pmatrix} \mathbf{c}^{\mathsf{T}}_{\mathbf{B},1} = \mathbf{s}_1^{\mathsf{T}}\mathbf{B} + \mathbf{e}^{\mathsf{T}}_{\mathbf{B},1} \\ \vdots \\ \mathbf{c}^{\mathsf{T}}_{\mathbf{B},Q_{\mathsf{msg}}} = \mathbf{s}^{\mathsf{T}}_{Q_{\mathsf{msg}}}\mathbf{B} + \mathbf{e}^{\mathsf{T}}_{\mathbf{B},Q_{\mathsf{msg}}}\end{pmatrix} = \{\mathbf{c}^{\mathsf{T}}_{\mathbf{B},j}\}_{j\in[Q_{\mathsf{msg}}]}, \tag{12}$$

$$\mathbf{C}_{\mathbf{P}} = \begin{pmatrix} \mathbf{c}^{\mathsf{T}}_{\mathbf{P},1} = \mathbf{s}_1^{\mathsf{T}}\mathbf{A}_{\mathrm{F}_1} + \mathbf{e}^{\mathsf{T}}_{\mathbf{P},1,1}||\ldots||\mathbf{s}_1^{\mathsf{T}}\mathbf{A}_{\mathrm{F}_{Q_{\mathsf{key}}}} + \mathbf{e}^{\mathsf{T}}_{\mathbf{P},1,Q_{\mathsf{key}}} \\ \vdots \\ \mathbf{c}^{\mathsf{T}}_{\mathbf{P},Q_{\mathsf{msg}}} = \mathbf{s}^{\mathsf{T}}_{Q_{\mathsf{msg}}}\mathbf{A}_{\mathrm{F}_1} + \mathbf{e}^{\mathsf{T}}_{\mathbf{P},Q_{\mathsf{msg}},1}||\ldots||\mathbf{s}^{\mathsf{T}}_{Q_{\mathsf{msg}}}\mathbf{A}_{\mathrm{F}_{Q_{\mathsf{key}}}} + \mathbf{e}^{\mathsf{T}}_{\mathbf{P},Q_{\mathsf{msg}},Q_{\mathsf{key}}}\end{pmatrix} = \{\mathbf{c}^{\mathsf{T}}_{\mathbf{P},j,k}\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]}, \tag{13}$$

we rewrite Equation (11) as follows.

$$\begin{pmatrix} \mathsf{aux}_2,\ \mathbf{B}, \{\mathbf{c}_{\mathbf{B},j}^{\mathsf{T}} = \mathbf{s}_j^{\mathsf{T}}\mathbf{B} + \mathbf{e}_{\mathbf{B},j}^{\mathsf{T}}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{X}_j = \mathbf{A}_{\mathsf{fhe},j}\mathbf{R}_j - (\mathbf{x}_j, \mathsf{sd}_j)\otimes\mathbf{G}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}_{\mathsf{att},j}^{\mathsf{T}} = \mathbf{s}_j^{\mathsf{T}}(\mathbf{A}_{\mathsf{att}} - \mathsf{bits}(1,\mathbf{X}_j)\otimes\mathbf{G}) + \mathbf{e}_{\mathsf{att},j}^{\mathsf{T}}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}_{\mathbf{P},j,k}^{\mathsf{T}} = \mathbf{s}_j^{\mathsf{T}}\mathbf{A}_{\mathrm{F}_k} + \mathbf{e}_{\mathbf{P},j,k}^{\mathsf{T}}\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]} \end{pmatrix} \approx_c \begin{pmatrix} \mathsf{aux}_2,\ \mathbf{B}, \{\mathbf{c}_{\mathbf{B},j} \leftarrow \mathbb{Z}_q^{mw}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{X}_j \leftarrow \mathbb{Z}_q^{(n+1)\times m(\lambda+\mathrm{L})}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}_{\mathsf{att},j} \leftarrow \mathbb{Z}_q^{(\mathrm{L_X}+1)m}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}_{\mathbf{P},j,k} \leftarrow \mathbb{Z}_q^{\ell}\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]} \end{pmatrix} \tag{14}$$

where $\mathbf{e}_{\mathbf{P},j,k} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma_1}^{\ell}$. Now, to prove Equation (10) it suffices to show Equation (14).
We prove Equation (14) via the following sequence of hybrids.

Hyb_0. This is L.H.S distribution of Equation (14).

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 (7) 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 $\left\|\mathsf{err}_{j,k}\right\| \le 2^{7\lambda}\beta$.

– Next, we note that $\left\|\mathsf{err}_{j,k}\right\| \le 2^{8\lambda+O(1)}\beta/\mathrm{poly}(\lambda) = \chi_1 = \left\|\mathbf{e}_{\mathbf{P},j,k}\right\|$. Thus by noise flooding (Lemma 2.4) we have $\mathbf{e}_{\mathbf{P},j,k}^{\mathsf{T}} \approx_s \mathsf{err}_{j,k} + \mathbf{e}_{\mathbf{P},j,k}^{\mathsf{T}}$ with a statistical distance of $\mathrm{poly}(\lambda)2^{-\lambda}$.

From the above, we have

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

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

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

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

Claim 3.7. $\mathsf{Hyb}_{1.i-1} \approx_c \mathsf{Hyb}_{1.i}$, for $i \in [Q_{\mathsf{msg}}]$, assuming the security of LWE.

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

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

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

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

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

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

– Randomness of $\mathbf{b}_{\mathbf{B}}$ implies the randomness of $\mathbf{c}_{\mathbf{B},i}^{\mathsf{T}} := \mathbf{b}_{\mathbf{B}}^{\mathsf{T}} - \underline{\mathbf{b}}^{\mathsf{T}}$.

– Randomness of $\mathbf{b}_{\mathsf{fhe}}$ implies $\mathbf{A}_{\mathsf{fhe},i} \leftarrow \mathbb{Z}_q^{(n+1)\times m}$.

– Randomness of $\mathbf{b}_{\mathsf{att}}$ implies randomness of $\mathbf{c}_{\mathsf{att},i}^{\mathsf{T}} = \mathbf{b}_{\mathsf{att}}^{\mathsf{T}} - (\underline{\mathbf{a}}_{\mathsf{att}}^{\mathsf{T}} - \mathsf{bits}(1, \mathbf{X}_i) \otimes \underline{\mathbf{G}})$.

□

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

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

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

Hyb_3. This hybrid is same as Hyb_2 except that for $j \in [Q_{\mathsf{msg}}]$ we sample $\mathbf{X}_j \leftarrow \mathbb{Z}_q^{(n+1)\times m(\lambda+\mathrm{L})}$. We have $\mathsf{Hyb}_2 \approx_s \mathsf{Hyb}_3$ using leftover hash lemma. By leftover hash lemma (Lemma 2.5) 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

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

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

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

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

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

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

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

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

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

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

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

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

$$\begin{aligned}(1^\lambda,\ \mathsf{aux}_{\mathcal{A}},\ \{f_k,\ \tilde{\mathrm{F}}_{j,k} = f_k(x_j)\ \lfloor q/2 \rceil + R_{j,k} + \mathbf{e}^{\mathsf{T}}_{\mathbf{P},j,k}\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]}) \\ \approx_c (1^\lambda,\ \mathsf{aux}_{\mathcal{A}},\ \{f_k,\ \tilde{\mathrm{F}}_{j,k} \leftarrow \mathbb{Z}_q^\ell\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]})\end{aligned} \tag{15}$$

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 (15) 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 (14), hence the proof.

□

3.4 Basing Security on Variant of Circular Evasive LWE ([HLL23]

In this section, we provide our construction of a functional encryption scheme for pseudorandom functionalities for function family $\mathcal{F}_{\mathrm{L}(\lambda),\ell(\lambda),\mathsf{dep}(\lambda)} = \{f : \{0,1\}^{\mathrm{L}} \to \{0,1\}^\ell\}$ basing the security on a variant of the circular evasive assumption introduced by [HLL23], which is considered public-coin.

3.4.1 Assumptions

Here, we state the assumptions used in this section.

Assumption 3.8 (Circular Small Secret LWE). [HLL23] Let n, m, m', q, χ, χ' be functions of λ and

$$\bar{\mathbf{A}}_{\mathsf{fhe}} \leftarrow \mathbb{Z}_q^{n\times m},\ \bar{\mathbf{A}}' \leftarrow \mathbb{Z}_q^{n\times m'},\ \mathbf{r} \leftarrow \mathcal{D}^n_{\mathbb{Z},\chi},\ \mathbf{s} \leftarrow (\mathbf{r}^{\mathsf{T}}, -1)^{\mathsf{T}},\ \mathbf{e}_{\mathsf{fhe}} \leftarrow \mathcal{D}^m_{\mathbb{Z},\chi},\ \mathbf{e}' \leftarrow \mathcal{D}^{m'}_{\mathbb{Z},\chi'},$$

$$\mathbf{R} \leftarrow \{0,1\}^{m\times(n+1)\lceil\log_2 q\rceil m},\ \delta_{\mathsf{fhe}} \leftarrow \mathbb{Z}_q^m,\ \delta' \leftarrow \mathbb{Z}_q^{m'},\ \Delta \leftarrow \mathbb{Z}_q^{(n+1)\times(n+1)\lceil\log_2 q\rceil m}$$

The circular small-secret LWE assumption $\mathsf{csLWE}_{n,m,m',q,\chi,\chi'}$ states that

$$\left\{\left(1^\lambda, \begin{pmatrix}\bar{\mathbf{A}}_{\mathsf{fhe}} \\ \mathbf{r}^{\mathsf{T}}\bar{\mathbf{A}}_{\mathsf{fhe}} + \mathbf{e}^{\mathsf{T}}_{\mathsf{fhe}}\end{pmatrix}, \begin{pmatrix}\bar{\mathbf{A}}_{\mathsf{fhe}} \\ \mathbf{r}^{\mathsf{T}}\bar{\mathbf{A}}_{\mathsf{fhe}} + \mathbf{e}^{\mathsf{T}}_{\mathsf{fhe}}\end{pmatrix}\mathbf{R} - \mathsf{bits}(\mathbf{s}) \otimes \mathbf{G}, \bar{\mathbf{A}}', \mathbf{r}^{\mathsf{T}}\bar{\mathbf{A}}' + (\mathbf{e}')^{\mathsf{T}}\right)\right\}_{\lambda\in\mathbb{N}}$$

$$\approx \left\{\left(1^\lambda, \begin{pmatrix}\bar{\mathbf{A}}_{\mathsf{fhe}} \\ \delta^{\mathsf{T}}_{\mathsf{fhe}}\end{pmatrix},\quad \Delta,\quad \bar{\mathbf{A}}',\ (\delta_i)^{\mathsf{T}}\right)\right\}_{\lambda\in\mathbb{N}}$$

Next, we define a variant of evcsLWE assumption introduced by [HLL23] further refined to avoid the attacks as discusses in [AMYY25].

Assumption 3.9 (evcsLWE). Let $\mathcal{S}(1^\lambda; \mathsf{aux})$ be an algorithm that, given randomness aux, outputs

$$\mathbf{A}_{\mathsf{circ}} \leftarrow \mathbb{Z}_q^{(n+1)\times(m(n+1)^2\lceil\log_2 q\rceil^2+1)m},\ \bar{\mathbf{A}}' \in \mathbb{Z}_q^{n\times m'},\ \mathbf{P} \in \mathbb{Z}_q^{n\times J},\ \sigma,\ \sigma',\ \sigma_{-1},\ \sigma_{\mathsf{post}},\ \sigma_{\mathsf{pre}}$$

where $m \geq m_0(n,q)$ and $\sigma_{-1} \geq \sigma_0(n,m)$ and $\sigma_{\mathsf{post}} \geq \sigma_{\mathsf{pre}}$. Suppose

$$\bar{\mathbf{A}}_{\mathsf{fhe}} \leftarrow \mathbb{Z}_q^{n\times m}, \quad (\mathbf{B},\tau) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q), \quad \mathbf{K} \leftarrow \mathbf{B}^{-1}(\mathbf{P}),$$

$$\mathbf{e}_{\mathsf{fhe}} \leftarrow \mathcal{D}^m_{\mathbb{Z},\sigma},\ \mathbf{e}_{\mathsf{circ}} \leftarrow \mathcal{D}^{(m(n+1)^2\lceil\log_2 q\rceil^2+1)m}_{\mathbb{Z},\sigma'},\ \mathbf{e}' \leftarrow \mathcal{D}^{m'}_{\mathbb{Z},\sigma'},\ \mathbf{e}_{\mathbf{B}} \in \mathbb{Z}^m,\ \mathbf{e}_{\mathbf{P}} \in \mathbb{Z}^J,$$

$$\delta_{\mathsf{fhe}} \leftarrow \mathbb{Z}_q^m,\ \delta_{\mathsf{circ}} \leftarrow \mathbb{Z}_q^{(m(n+1)^2\lceil\log_2 q\rceil^2+1)m},\ \delta' \leftarrow \mathbb{Z}_q^{m'},\ \delta_{\mathbf{B}} \leftarrow \mathbb{Z}_q^m,\ \delta_{\mathbf{P}} \leftarrow \mathbb{Z}_q^J,$$

$$\mathbf{r} \leftarrow \mathcal{D}^n_{\mathbb{Z},\sigma}, \quad \mathbf{s} \leftarrow (\mathbf{r}^\intercal, -1)^\intercal, \quad \mathbf{R} \leftarrow \{0,1\}^{m\times(n+1)\lceil\log_2 q\rceil m}, \quad \Delta \leftarrow \mathbb{Z}_q^{(n+1)\times(n+1)\lceil\log_2 q\rceil m},$$

$$\mathbf{S} = \begin{pmatrix} \bar{\mathbf{A}}_{\mathsf{fhe}} \\ \mathbf{r}^\intercal\bar{\mathbf{A}}_{\mathsf{fhe}} + \mathbf{e}_{\mathsf{fhe}}^\intercal \end{pmatrix}\mathbf{R} - (\mathbf{x}, \mathsf{bits}(\mathbf{s})) \otimes \mathbf{G} \ \text{ for } \ \mathbf{x} \in \{0,1\}^{\mathrm{L}}$$

In the precondition, the entries of $\mathbf{e}_{\mathbf{B}}, \mathbf{e}_{\mathbf{P}}$ are independent and follow $\mathcal{D}_{\mathbb{Z},\sigma_{\mathsf{pre}}}$, and $\mathsf{evcsLWE}^{\mathcal{S}}_{\mathsf{pre}}$ states that

$$\left\{\begin{pmatrix} 1^\lambda, \mathsf{aux}, \bar{\mathbf{A}}_{\mathsf{fhe}}, \mathbf{B}, \mathbf{r}^\intercal\bar{\mathbf{A}}_{\mathsf{fhe}} + \mathbf{e}_{\mathsf{fhe}}^\intercal, \mathbf{S}, \\ \mathbf{s}^\intercal(\mathbf{A}_{\mathsf{circ}} - (1, \mathsf{bits}(\mathbf{S})) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{circ}}^\intercal \\ \mathbf{r}^\intercal\bar{\mathbf{A}}' + (\mathbf{e}')^\intercal, \mathbf{r}^\intercal\mathbf{B} + \mathbf{e}_{\mathbf{B}}^\intercal, \mathbf{r}^\intercal\mathbf{P} + \mathbf{e}_{\mathbf{P}}^\intercal \end{pmatrix}\right\}_{\lambda\in\mathbb{N}} \approx \left\{\begin{pmatrix} 1^\lambda, \mathsf{aux}, \bar{\mathbf{A}}_{\mathsf{fhe}}, \mathbf{B}, \delta_{\mathsf{fhe}}^\intercal, \Delta, \\ \delta_{\mathsf{circ}}^\intercal, \\ (\delta')^\intercal, \delta_{\mathbf{B}}^\intercal, \delta_{\mathbf{P}}^\intercal \end{pmatrix}\right\}_{\lambda\in\mathbb{N}}.$$

In the postcondition, the entries of $\mathbf{e}_{\mathbf{B}}$ are independent and follow $\mathcal{D}_{\mathbb{Z},\sigma_{\mathsf{post}}}$, and $\mathsf{evcsLWE}^{\mathcal{S}}_{\mathsf{post}}$ states that

$$\left\{\begin{pmatrix} 1^\lambda, \mathsf{aux}, \bar{\mathbf{A}}_{\mathsf{fhe}}, \mathbf{B}, \mathbf{r}^\intercal\bar{\mathbf{A}}_{\mathsf{fhe}} + \mathbf{e}_{\mathsf{fhe}}^\intercal, \mathbf{S}, \\ \mathbf{s}^\intercal(\mathbf{A}_{\mathsf{circ}} - (1, \mathsf{bits}(\mathbf{S})) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{circ}}^\intercal \\ \mathbf{r}^\intercal\bar{\mathbf{A}}' + (\mathbf{e}')^\intercal, \mathbf{r}^\intercal\mathbf{B} + \mathbf{e}_{\mathbf{B}}^\intercal, \mathbf{K} \end{pmatrix}\right\}_{\lambda\in\mathbb{N}} \approx \left\{\begin{pmatrix} 1^\lambda, \mathsf{aux}, \bar{\mathbf{A}}_{\mathsf{fhe}}, \mathbf{B}, \delta_{\mathsf{fhe}}^\intercal, \Delta, \\ \delta_{\mathsf{circ}}^\intercal, \\ (\delta')^\intercal, \delta_{\mathbf{B}}^\intercal, \mathbf{K} \end{pmatrix}\right\}_{\lambda\in\mathbb{N}}.$$

The *evasive circular small-secret LWE assumption* with respect to the sampler class $\mathcal{SC}$ states that $\mathsf{evcsLWE}^{\mathcal{S}}_{\mathsf{pre}}$ implies $\mathsf{evcsLWE}^{\mathcal{S}}_{\mathsf{post}}$ for all efficient samplers $\mathcal{S}$.
We conjecture that for reasonable class of samplers, the evasive LWE assumption holds. In particular, we conjecture that our sampler $\mathsf{Samp}_{\mathsf{prFE}}(1^\lambda)$ used for the security proof of our prFE for natural class of functions should be in the secure class of samplers $\mathcal{SC}$ for which the evasive LWE holds.

Remark 3.10. In our assumption, the FHE encoding encode the attribute $\mathbf{x}$, whereas the one in [HLL23] it only encode the FHE secret key. This is created by the difference between ABE and FE, since the attribute $\mathbf{x}$ can be public in the former and must be hidden in the latter.

3.4.2 Construction

The construction is same as in Section 3.2 except the following changes.

1. We use a concrete pseudorandom function PRF, $\mathsf{PRF} : \mathbb{Z}^{(n+1)\times m'} \times \mathbb{Z}^{n+1} \to [-q/4+B, q/4-B]^{1\times\ell}$ defined as $\mathsf{PRF}(\mathbf{A},\mathbf{s}) = [\mathbf{G}_q\mathbf{G}_p^{-1}(\lfloor(\mathbf{s}^\intercal\mathbf{A})^\intercal\rfloor_p)]_B$. Here $m' = \lceil\ell(\lceil\log q\rceil \ / \ \lceil\log p\rceil)\rceil$, $\mathbf{G}_q = \mathbf{I}_\ell \otimes (1, 2, 2^2, \ldots, 2^{\lceil\log_2 q\rceil-1})$, $\mathbf{G}_p^{-1}(\mathbf{x})$ for a vector $\mathbf{x} \in \mathbb{Z}^{m'}$ is $(\mathsf{bits}(\mathbf{x}[1]), \ldots, \mathsf{bits}(\mathbf{x}[m']))^\intercal$, where $\mathsf{bits}(\mathbf{x}[i]) \in \{0,1\}^{\log p}$ and $[\mathbf{x}]_B$ for any $\mathbf{x} \in \mathbb{Z}_q^\ell$ represents truncating the range to $[-q/2+B, q/2-B]^\ell$ by mapping the out-of-range values to 0. Here B is chosen to be exponentially smaller than $q/4$. We note that for our choice of B the statistical distance between the uniform distribution over $[-q/4, q/4]$ and $[-q/4+B, q/4-B]$ is negligible. We show how to set p later.

2. In the setup algorithm, we set $\mathrm{L}_{\mathrm{X}} = m(\mathrm{L} + (n+1)\lceil\log q\rceil)(n+1)\lceil\log q\rceil$ and replace the notation $\mathbf{A}_{\mathsf{att}}$ with $\mathbf{A}_{\mathsf{circ}}$ to match the notation of the underlying assumption.

3. The $\mathsf{KeyGen}(\mathsf{msk}, f)$ algorithm has the following changes.

   – It samples $\mathbf{A}' \leftarrow \mathbb{Z}_q^{(n+1)\times m'}$ where $m' = \lceil \ell(\lceil \log q \rceil \,/\, \lceil \log p \rceil) \rceil$ instead of $\mathbf{r} \leftarrow \{0,1\}^\lambda$.
   – It defines function $\mathrm{F}[f, \mathbf{A}']$ (instead of $\mathrm{F} = \mathrm{F}[f, \mathbf{r}]$), with $f, \mathbf{A}'$ hardwired, as follows:
     On input $(\mathbf{x} \in \{0,1\}^{\mathrm{L}}, \mathsf{bits}(\mathbf{s}) \in \{0,1\}^{(n+1)\lceil \log q \rceil})$, first recover $\mathbf{s} \in \mathbb{Z}^{n+1}$ from $\mathsf{bits}(\mathbf{s})$ and then compute and output $f(\mathbf{x}) \lfloor q/2 \rceil + \mathsf{PRF}(\mathbf{A}', \mathbf{s}) \in \mathbb{Z}_q^{1\times \ell}$.

   The rest of the algorithm remains as it is. It outputs $\mathsf{sk}_f = (\mathbf{K}, \mathbf{A}')$.

4. The encryption algorithm has the following changes

   – It samples $\mathbf{R}$ differently as $\mathbf{R} \leftarrow \{0,1\}^{m \times m(\mathrm{L}+(n+1)\lceil \log q \rceil)}$ and computes $\mathbf{X} = \mathbf{A}_{\mathsf{fhe}}\mathbf{R} - (\mathbf{x}, \mathsf{bits}(\mathbf{s})) \otimes \mathbf{G} \in \mathbb{Z}_q^{(n+1)\times m(\mathrm{L}+(n+1)\lceil \log q \rceil)}$.
   – We also change the notation of $\mathbf{c}_{\mathsf{att}}$ and denote this as $\mathbf{c}_{\mathsf{circ}}^{\mathsf{T}} = \mathbf{s}^{\mathsf{T}}(\mathbf{A}_{\mathsf{circ}} - \mathsf{bits}(1, \mathbf{X}) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{circ}}^{\mathsf{T}}$ for $\mathbf{e}_{\mathsf{circ}} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}^{(\mathrm{L}_{\mathrm{X}}+1)m}$.

   The rest of the algorithm remains as it is.

The above modified prFE scheme satisfies correctness. This can be argued using same steps as for the construction in Section 3.2.

3.4.3 Security

Theorem 3.11. Let $\mathcal{SC}_{\mathsf{prFE}}$ be a sampler class for prFE. Assuming circular small-secret LWE (Assumption 3.8) and evasive circular small-secret LWE (Assumption 3.9) with respect to the sampler class that contains all $\mathsf{Samp}_{\mathsf{evcs}}(1^\lambda)$ induced by $\mathsf{Samp}_{\mathsf{prFE}} \in \mathcal{SC}_{\mathsf{prFE}}$ as defined in Figure 2, our prFE scheme satisfies prCT security with respect to $\mathcal{SC}_{\mathsf{prFE}}$ as defined in Definition 3.2.

Proof. For a prFE sampler $\mathsf{Samp}_{\mathsf{prFE}}$ as defined in the proof for Theorem 3.6, we show

$$\begin{pmatrix} \mathsf{mpk} = (\mathbf{A}_{\mathsf{circ}}, \mathbf{B}, M),\ \mathsf{aux}_{\mathcal{A}},\ \ \mathbf{C}_{\mathbf{B}} = \mathbf{S}\mathbf{B} + \mathbf{E}_{\mathbf{B}}, \\ \{\mathbf{X}_j = \mathbf{A}_{\mathsf{fhe},j}\mathbf{R}_j - (\mathbf{x}_j, \mathsf{bits}(\mathbf{s}_j)) \otimes \mathbf{G}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}_{\mathsf{circ},j}^{\mathsf{T}} = \mathbf{s}_j^{\mathsf{T}}(\mathbf{A}_{\mathsf{circ}} - \mathsf{bits}(1, \mathbf{X}_j) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{circ},j}^{\mathsf{T}}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{K}_k, \mathbf{A}'_k\}_{k\in[Q_{\mathsf{key}}]} \end{pmatrix} \tag{16}$$
$$\approx_c \begin{pmatrix} \mathsf{mpk} = (\mathbf{A}_{\mathsf{circ}}, \mathbf{B}, M),\ \mathsf{aux}_{\mathcal{A}},\ \ \mathbf{C}_{\mathbf{B}} \leftarrow \mathbb{Z}_q^{Q_{\mathsf{msg}}\times mw}, \\ \{\mathbf{X}_j \leftarrow \mathbb{Z}_q^{(n+1)\times m(\mathrm{L}+(n+1)\lceil \log q \rceil)}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}_{\mathsf{circ},j} \leftarrow \mathbb{Z}_q^{(\mathrm{L}_{\mathrm{X}}+1)m}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{K}_k, \mathbf{A}'_k\}_{k\in[Q_{\mathsf{key}}]} \end{pmatrix}$$

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 $j \in [Q_{\mathsf{msg}}], \mathbf{s}_j, \mathbf{e}_{\mathbf{B},j}, \mathbf{A}_{\mathsf{fhe},j}, \mathbf{R}_j, \mathsf{sd}_j,\ \mathbf{e}_{\mathsf{circ},j}$ are sampled as in the construction, for $k \in [Q_{\mathsf{key}}]$, we have $\mathbf{A}' \leftarrow \mathbb{Z}_q^{(n+1)\times m'}$, $\mathrm{F}_k = \mathrm{F}[f_k, \mathbf{A}'_k]$ 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

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

Consider the evcsLWE sampler $\mathsf{Samp}_{\mathsf{evcs}}(1^\lambda, \mathsf{aux})$ as defined in Figure 2. To prove Equation (16), assuming evcsLWE

$\mathsf{Samp}_{\mathsf{evcs}}(1^\lambda, \mathsf{aux})$

The sampler does the following.

- Parse $\mathsf{aux} = (\mathsf{aux}_{\mathcal{A}}, \{f_k\}_{k\in[Q_{\mathsf{key}}]}, \{\mathbf{A}'_k \leftarrow \mathbb{Z}_q^{(n+1)\times m'}\}_{k\in[Q_{\mathsf{key}}]}, \mathsf{aux}_{\mathsf{Samp}})$.
- Sample parameters as in Equation (3). Additionally set $p = q/(2^\lambda \sigma\sqrt{\lambda})$[a].
- Samples $\mathbf{A}_{\mathsf{circ}} \leftarrow \mathbb{Z}_q^{(n+1)\times(\mathrm{L}_{\mathrm{X}}+1)m}$ using the randomness $\mathsf{aux}_{\mathsf{Samp}}$.
- Defines $\mathrm{F}_k[f_k, \mathbf{A}'_k]$ and computes $\mathbf{A}_{\mathrm{F}_k}$ as in the construction. $\mathbf{A}_{\mathrm{F}_k}$ is publicly computable given f_k, $\mathbf{A}'_k$, M and $\mathbf{A}_{\mathsf{circ}}$.
- It outputs

$$\mathbf{A}_{\mathsf{circ}}, \quad \mathbf{P} = [\mathbf{A}_{\mathrm{F}_1}||\ldots||\mathbf{A}_{\mathrm{F}_{Q_{\mathsf{key}}}}],\ \mathsf{aux} = (\mathsf{aux}_{\mathsf{Samp}}, \mathsf{aux}_{\mathsf{prFE}}, \{\mathbf{A}'_k\}_{k\in[Q_{\mathsf{key}}]})$$

[a]We assume the parameters to be part of the output of $\mathsf{Samp}_{\mathsf{evcs}}$, even though we do not explicitly write so.

Figure 2: Description of the Sampler for circular small-secret evasive LWE

assumption (Assumption 3.9) with sampler $\mathsf{Samp}_{\mathsf{evcs}}$ it suffices to show

$$\begin{pmatrix} \mathsf{aux}_1,\ \mathbf{B},\ \mathbf{C}_{\mathbf{B}} = \mathbf{S}\mathbf{B} + \mathbf{E}_{\mathbf{B}}, \\ \{\mathbf{X}_j = \mathbf{A}_{\mathsf{fhe},j}\mathbf{R}_j - (\mathbf{x}_j, \mathsf{bits}(\mathbf{s}_j)) \otimes \mathbf{G}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}^{\mathsf{T}}_{\mathsf{circ},j} = \mathbf{s}_j^{\mathsf{T}}(\mathbf{A}_{\mathsf{circ}} - \mathsf{bits}(1, \mathbf{X}_j) \otimes \mathbf{G}) + \mathbf{e}^{\mathsf{T}}_{\mathsf{circ},j}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \mathbf{C}_{\mathbf{P}} = \mathbf{S}\mathbf{P} + \mathbf{E}_{\mathbf{P}} \end{pmatrix} \approx_c \begin{pmatrix} \mathsf{aux}_1,\ \mathbf{B},\ \mathbf{C}_{\mathbf{B}} \leftarrow \mathbb{Z}_q^{Q_{\mathsf{msg}}\times mw}, \\ \{\mathbf{X}_j \leftarrow \mathbb{Z}_q^{(n+1)\times m(\mathrm{L}+(n+1)\lceil \log q \rceil)}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}_{\mathsf{circ},j} \leftarrow \mathbb{Z}_q^{(\mathrm{L}_{\mathrm{X}}+1)m}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \mathbf{C}_{\mathbf{P}} \leftarrow \mathbb{Z}_q^{Q_{\mathsf{msg}}\times \ell\cdot Q_{\mathsf{key}}} \end{pmatrix} \tag{17}$$

where $\mathsf{aux}_1 = (\mathsf{aux}, M)$ and $\mathbf{E}_{\mathbf{P}} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma_1}^{Q_{\mathsf{msg}}\times\ell\cdot Q_{\mathsf{key}}}$. Using the representation in Equations (12) and (13), we rewrite the Equation (17) as

$$\begin{pmatrix} \mathsf{aux}_1,\ \mathbf{B}, \{\mathbf{c}^{\mathsf{T}}_{\mathbf{B},j} = \mathbf{s}_j^{\mathsf{T}}\mathbf{B} + \mathbf{e}^{\mathsf{T}}_{\mathbf{B},j}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{X}_j = \mathbf{A}_{\mathsf{fhe},j}\mathbf{R}_j - (\mathbf{x}_j, \mathsf{bits}(\mathbf{s}_j)) \otimes \mathbf{G}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}^{\mathsf{T}}_{\mathsf{circ},j} = \mathbf{s}_j^{\mathsf{T}}(\mathbf{A}_{\mathsf{circ}} - \mathsf{bits}(1, \mathbf{X}_j) \otimes \mathbf{G}) + \mathbf{e}^{\mathsf{T}}_{\mathsf{circ},j}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}^{\mathsf{T}}_{\mathbf{P},j,k} = \mathbf{s}_j^{\mathsf{T}}\mathbf{A}_{\mathrm{F}_k} + \mathbf{e}^{\mathsf{T}}_{\mathbf{P},j,k}\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]} \end{pmatrix} \approx_c \begin{pmatrix} \mathsf{aux}_1,\ \mathbf{B}, \{\mathbf{c}_{\mathbf{B},j} \leftarrow \mathbb{Z}_q^{mw}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{X}_j \leftarrow \mathbb{Z}_q^{(n+1)\times m(\mathrm{L}+(n+1)\lceil \log q \rceil)}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}_{\mathsf{circ},j} \leftarrow \mathbb{Z}_q^{(\mathrm{L}_{\mathrm{X}}+1)m}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}_{\mathbf{P},j,k} \leftarrow \mathbb{Z}_q^{\ell}\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]} \end{pmatrix} \tag{18}$$

where $\mathbf{e}_{\mathbf{P},j,k} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma_1}^{\ell}$.

We prove Equation (18) using the following sequence of hybrids.

Hyb_0. This is L.H.S distribution of Equation (18).

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

$$\mathbf{c}^{\mathsf{T}}_{\mathbf{P},j,k} = M \cdot \left\lfloor \frac{\mathbf{c}^{\mathsf{T}}_{\mathsf{circ},j} \cdot \mathbf{H}^{\mathrm{F}_{\mathsf{high},k}}_{\mathbf{A}_{\mathsf{circ}},\mathbf{X}_j}}{M} \right\rfloor + \left\lfloor \frac{\mathbf{c}^{\mathsf{T}}_{\mathsf{circ},j} \cdot \mathbf{H}^{\mathrm{F}_{\mathsf{low},k}}_{\mathbf{A}_{\mathsf{circ}},\mathbf{X}_j}}{M} \right\rfloor + f_k(\mathbf{x}_j)\lfloor q/2 \rceil + \mathsf{PRF}(\mathbf{A}'_k, \mathbf{s}_j) + \mathbf{e}^{\mathsf{T}}_{\mathbf{P},j,k}$$

where $\mathbf{e}_{\mathbf{P},j,k} \leftarrow \mathcal{D}^{\ell}_{\mathbb{Z},\sigma_1}$. The proof of $\mathsf{Hyb}_0 \approx_s \mathsf{Hyb}_1$ is exactly the same as the proof of $\mathsf{Hyb}_0 \approx_s \mathsf{Hyb}_1$ in Theorem 3.6, hence we omit it. So it suffices to show pseudorandomness of the following distribution given aux_1

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

where we write $\mathsf{PRF}(\mathbf{A}'_k, \mathbf{s}_j) = \left[\mathbf{G}_q\mathbf{G}_p^{-1}\left(\left\lfloor(\mathbf{s}^{\top}_j\mathbf{A}'_k)^{\top}\right\rceil_p\right)\right]_B$.

Hyb_2**:** This hybrid is same as Hyb_1 except that for all $j \in [Q_{\mathsf{msg}}], k \in [Q_{\mathsf{key}}]$, we compute the PRF differently. We compute $\mathsf{PRF}(\mathbf{A}'_k, \mathbf{s}_j) = \left[\mathbf{G}_q\mathbf{G}_p^{-1}\left(\left\lfloor(\bar{\mathbf{s}}^{\top}_j\bar{\mathbf{A}}'_k - (\underline{\mathbf{a}}'_k)^{\top} + \mathbf{e}^{\top}_k)^{\top}\right\rceil_p\right)\right]_B$ for $\mathbf{e}_k \leftarrow \mathcal{D}^{\ell}_{\mathbb{Z},\sigma}$, where $\bar{\mathbf{A}}'_k$ denotes the first n rows and $\underline{\mathbf{a}}'_k$ denotes the last row of the matrix $\mathbf{A}'_k$. We claim that $\mathsf{Hyb}_1 \approx_s \mathsf{Hyb}_2$ with all but negligible probability. To see this, we note the following.

- The statistical distance between $(\bar{\mathbf{s}}^{\top}_j\bar{\mathbf{A}}'_k - (\underline{\mathbf{a}}'_k)^{\top} + \mathbf{e}^{\top}_k)^{\top}$ and $(\bar{\mathbf{s}}^{\top}_j\bar{\mathbf{A}}'_k - (\underline{\mathbf{a}}'_k)^{\top})^{\top}$ is bounded by $\|\mathbf{e}_k\| \le \sigma\sqrt{\lambda}$.
- So, $\Pr\left[\left\lfloor(\bar{\mathbf{s}}^{\top}_j\bar{\mathbf{A}}'_k - \underline{\mathbf{a}}'_k)^{\top})^{\top}\right\rceil_p \neq \left\lfloor(\bar{\mathbf{s}}^{\top}_j\bar{\mathbf{A}}'_k - \underline{\mathbf{a}}'_k)^{\top} + \mathbf{e}^{\top}_k)^{\top}\right\rceil_p\right] \le \sigma\sqrt{\lambda}p/q = 1/2^{\lambda}$ due to our parameter setting.
- Taking the probability over all $k \in [Q_{\mathsf{key}}]$, we get $\mathsf{Hyb}_1 \approx_s \mathsf{Hyb}_2$ with all but $Q_{\mathsf{key}}/2^{\lambda}$ probability.

So it suffices to show pseudorandomness of the following distribution given aux_1

$$\left(\begin{array}{c} \mathbf{B},\ \{\mathbf{c}^{\top}_{\mathbf{B},j} = \mathbf{s}^{\top}_j\mathbf{B} + \mathbf{e}^{\top}_{\mathbf{B},j}\}_{j\in[Q_{\mathsf{msg}}]} \\ \{\mathbf{X}_j = \mathbf{A}_{\mathsf{fhe},j}\mathbf{R}_j - (\mathbf{x}_j, \mathsf{bits}(\mathbf{s}_j)) \otimes \mathbf{G}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}^{\top}_{\mathsf{circ},j} = \mathbf{s}^{\top}_j(\mathbf{A}_{\mathsf{circ}} - \mathsf{bits}(1,\mathbf{X}_j)\otimes\mathbf{G}) + \mathbf{e}^{\top}_{\mathsf{circ},j}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\tilde{\mathrm{F}}_{j,k} = f_k(\mathbf{x}_j)\,\lfloor q/2\rceil + [\mathbf{G}_q\mathbf{G}_p^{-1}(\lfloor(\bar{\mathbf{s}}^{\top}_j\bar{\mathbf{A}}'_k - (\underline{\mathbf{a}}'_k)^{\top} + \mathbf{e}^{\top}_k)^{\top}\rceil_p)]_B + \mathbf{e}^{\top}_{\mathbf{P},j,k}\}_{j\in[Q_{\mathsf{msg}}]}, k \in [Q_{\mathsf{key}}]. \end{array}\right)$$

Hyb_3**:** This hybrid is same as Hyb_2 except that for all $j \in [Q_{\mathsf{msg}}]$ we sample $\mathbf{c}_{\mathbf{B},j} \leftarrow \mathbb{Z}_q^{mw}$, $\mathbf{c}_{\mathsf{circ},j} \leftarrow \mathbb{Z}_q^{(\mathrm{L_X}+1)m}$, $\mathbf{X}_j \leftarrow \mathbb{Z}_q^{(n+1)\times m(\mathrm{L}+(n+1)\lceil\log q\rceil)}$ and $R_{j,k} \leftarrow \mathbb{Z}_q^{\ell}$, where $R_{j,k} = \mathbf{G}_q\mathbf{G}_p^{-1}(\lfloor(\bar{\mathbf{s}}^{\top}_j\bar{\mathbf{A}}'_k - (\underline{\mathbf{a}}'_k)^{\top} + \mathbf{e}^{\top}_k)^{\top}\rceil_p)$ in Hyb_2 . We have $\mathsf{Hyb}_1 \approx_c \mathsf{Hyb}_2$ assuming the hardness of circular small secret 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 $\mathbf{c}_{\mathbf{B},j} \leftarrow \mathbb{Z}_q^{mw}$, $\mathbf{c}_{\mathsf{att},j} \leftarrow \mathbb{Z}_q^{(\mathrm{L_X}+1)m}$, $\mathbf{X}_j \leftarrow \mathbb{Z}_q^{(n+1)\times m(\mathrm{L}+(n+1)\lceil\log q\rceil)}$ and $R_{j,k} \leftarrow \mathbb{Z}_q^{\ell}$ for $1 \le j \le i$. We set $\mathsf{Hyb}_1 = \mathsf{Hyb}_{1.0}$ and $\mathsf{Hyb}_2 = \mathsf{Hyb}_{1.Q_{\mathsf{msg}}}$. We prove that for all $i \in [Q_{\mathsf{msg}}]$, $\mathsf{Hyb}_{1.i-1} \approx_c \mathsf{Hyb}_{1.i}$ in Claim 3.12. Thus, it suffices to show pseudorandomness of the following distribution given aux_2

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

Hyb_4**.** 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 $\{f_k(x_j)\}_{j,k}$. Thus we achieve the following distribution

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

which is the R.H.S distribution of Equation (14), hence the proof.

□

Claim 3.12. $\mathsf{Hyb}_{1.i-1} \approx_c \mathsf{Hyb}_{1.i}$.

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 csLWE 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 csLWE security game with the csLWE challenger. The csLWE challenger samples a bit $\beta \leftarrow \{0,1\}$ and does the following

   – It samples $\bar{\mathbf{A}}_{\mathsf{fhe},j} \leftarrow \mathbb{Z}_q^{n\times m}$, $\bar{\mathbf{A}}'_j \leftarrow \mathbb{Z}_q^{n\times(mw+(\mathrm{L_X}+1)m+m'\cdot Q_{\mathsf{key}})}$, $\bar{\mathbf{s}}_j \leftarrow \mathcal{D}^n_{\mathbb{Z},\sigma_{\mathsf{s}}}$, $\mathbf{e}_{\mathsf{fhe},j} \leftarrow \mathcal{D}^m_{\mathbb{Z},\sigma}$, $\mathbf{e}'_j \leftarrow \mathcal{D}^{mw+(\mathrm{L_X}+1)m+m'\cdot Q_{\mathsf{key}}}_{\mathbb{Z},\sigma}$, $\mathbf{R}_j \leftarrow \{0,1\}^{m\times(n+1)\lceil \log_2 q\rceil m}$, $\delta_{\mathsf{fhe},j} \leftarrow \mathbb{Z}_q^m$, $\delta'_j \leftarrow \mathbb{Z}_q^{mw+(\mathrm{L_X}+1)m+m'\cdot Q_{\mathsf{key}}}$, $\Delta_j \leftarrow \mathbb{Z}_q^{(n+1)\times(n+1)\lceil\log_2 q\rceil m}$. It sets $\mathbf{s}_j = (\bar{\mathbf{s}}_j^{\mathsf{T}}, -1)^{\mathsf{T}}$.

   – If $\beta = 0$, it sets $\mathbf{b}^{\mathsf{T}}_{\mathsf{fhe},j} := \bar{\mathbf{s}}_j^{\mathsf{T}}\bar{\mathbf{A}}_{\mathsf{fhe},j} + \mathbf{e}^{\mathsf{T}}_{\mathsf{fhe},j}$, $\mathbf{S}_j := \begin{pmatrix} \bar{\mathbf{A}}_{\mathsf{fhe},j} \\ \bar{\mathbf{s}}_j^{\mathsf{T}}\bar{\mathbf{A}}_{\mathsf{fhe},j} + \mathbf{e}^{\mathsf{T}}_{\mathsf{fhe},j}\end{pmatrix}\mathbf{R}_j - \mathsf{bits}(\mathbf{s}_j)\otimes\mathbf{G}$, and $(\mathbf{b}'_j)^{\mathsf{T}} = \bar{\mathbf{s}}_j^{\mathsf{T}}\bar{\mathbf{A}}'_j + (\mathbf{e}'_j)^{\mathsf{T}}$.

   – If $\beta = 1$, it sets $\mathbf{b}_{\mathsf{fhe},j} := \delta_{\mathsf{fhe},j}$, $\mathbf{S}_j := \Delta_j$, and $\mathbf{b}'_j := \delta'_j$.

   – It returns $\left(\mathbf{A}_{\mathsf{fhe},j} = (\bar{\mathbf{A}}_{\mathsf{fhe},j}\ \mathbf{b}^{\mathsf{T}}_{\mathsf{fhe},j})^{\mathsf{T}},\ \mathbf{S}_j,\ \bar{\mathbf{A}}'_j,\ (\mathbf{b}'_j)^{\mathsf{T}}\right)$ to the reduction.

3. $\mathcal{B}$ does the following.

   – For $1 \le j < i$: $\mathcal{B}$ samples $\mathbf{c}_{\mathbf{B},j} \leftarrow \mathbb{Z}_q^{mw}$, $\mathbf{c}_{\mathsf{circ},j} \leftarrow \mathbb{Z}_q^{(\mathrm{L_X}+1)m}$, $\mathbf{X}_j \leftarrow \mathbb{Z}_q^{(n+1)\times m(\mathrm{L}+(n+1)\lceil\log q\rceil)}$ and $R_{j,k} \leftarrow \mathbb{Z}_q^{\ell}$.

   – For $j = i$, it does as follows.

     (a) It samples $\mathbf{R}'_j \leftarrow \{0,1\}^{m\times m\mathrm{L}}$, computes $\mathbf{C}_j := \mathbf{A}_{\mathsf{fhe},j}\mathbf{R}'_j - \mathbf{x}_j \otimes \mathbf{G}$, sets $\mathbf{X}_j = \mathbf{S}_j + \mathbf{C}_j$.

     (b) It parses $\bar{\mathbf{A}}'_j = [\bar{\mathbf{B}}, \bar{\mathbf{A}}'_{\mathsf{circ},j}, \bar{\mathbf{A}}'_{j,1}, \ldots \bar{\mathbf{A}}'_{j,Q_{\mathsf{key}}}]$, where $\bar{\mathbf{B}} \in \mathbb{Z}_q^{n\times mw}$, $\bar{\mathbf{A}}'_{\mathsf{circ},j} \in \mathbb{Z}_q^{n\times(\mathrm{L_X}+1)m}$, $\bar{\mathbf{A}}'_{j,k} \in \mathbb{Z}_q^{n\times m'}$, and similarly $(\mathbf{b}'_j)^{\mathsf{T}} = \left((\mathbf{b}_j)^{\mathsf{T}}, (\mathbf{b}'_{\mathsf{circ},j})^{\mathsf{T}}, (\mathbf{b}'_{j,1})^{\mathsf{T}}, \ldots, (\mathbf{b}'_{j,Q_{\mathsf{key}}})^{\mathsf{T}}\right)$, where $\mathbf{b}_j \in \mathbb{Z}_q^{mw}$, $\mathbf{b}'_{\mathsf{circ},j} \in \mathbb{Z}_q^{(\mathrm{L_X}+1)m}$ and $\mathbf{b}'_{j,k} \in \mathbb{Z}_q^{m'}$ for $k \in [Q_{\mathsf{key}}]$.

     (c) It samples $\underline{\mathbf{b}} \leftarrow \mathbb{Z}_q^{mw}$, $\underline{\mathbf{a}}_{\mathsf{circ},j} \leftarrow \mathbb{Z}_q^{(\mathrm{L_X}+1)m}$ and sets $\mathbf{c}_{\mathbf{B},j} := (\mathbf{b}_j)^{\mathsf{T}} - \underline{\mathbf{b}}^{\mathsf{T}}$ and $\mathbf{c}^{\mathsf{T}}_{\mathsf{circ},j} := (\mathbf{b}'_{\mathsf{circ},j})^{\mathsf{T}} - (\underline{\mathbf{a}}^{\mathsf{T}}_{\mathsf{circ},j} - \mathsf{bits}(1, \mathbf{X}_j)\otimes\mathbf{G})$.

     (d) For $k \in [Q_{\mathsf{key}}]$, it samples $\underline{\mathbf{a}}'_k \leftarrow \mathbb{Z}_q^{m'}$ and sets $R_{j,k} = (\mathbf{b}'_{j,k})^{\mathsf{T}} - (\underline{\mathbf{a}}'_k)^{\mathsf{T}}$.

   – For $j > i$, it computes $\mathbf{c}_{\mathbf{B},j}$, $\mathbf{c}_{\mathsf{circ},j}$, $\mathbf{X}_j$ and $R_{j,k}$ as in Hyb_2.

4. It sets $\mathsf{aux}_1 = (\mathsf{aux}, M)$, $\mathbf{B} = (\bar{\mathbf{B}}\ \underline{\mathbf{b}}^{\mathsf{T}})^{\mathsf{T}}$ and $\tilde{\mathrm{F}}_{j,k} = f_k(\mathbf{x}_j)\lfloor q/2\rceil + [\mathbf{G}_q\mathbf{G}_p^{-1}(\lfloor (R_{j,k})^{\mathsf{T}}\rfloor_p)]_B + \mathbf{e}^{\mathsf{T}}_{\mathbf{P},j,k}$ and sends $(\mathsf{aux}_1, \mathbf{B}, \mathbf{c}_{\mathbf{B},j}, \mathbf{X}_j, \mathbf{c}_{\mathsf{circ},j}, \tilde{\mathrm{F}}_{j,k})$ for all $j \in [Q_{\mathsf{msg}}]$ and $k \in [Q_{\mathsf{key}}]$ to $\mathcal{A}$.

5. $\mathcal{A}$ outputs a bit β'. $\mathcal{B}$ forwards the bit β' to the csLWE challenger.

We note that if the csLWE challenger choose $\beta = 0$ then $\mathcal{B}$ simulated $\mathsf{Hyb}_{1,i-1}$ with $\mathcal{A}$ else it simulated $\mathsf{Hyb}_{1,i}$ with $\mathcal{A}$. □

4 Laconic Pseudorandom Poly-Domain Obfuscation

In this section, we introduce the notion of laconic pPRIO and construct it from several ingredients, which are all implied by the evasive LWE and LWE. The construction of laconic pPRIO will be used in Section 5. Since the intuition for this notion was discussed in Section 1, we proceed directly to the formal definition.

4.1 Definition

Syntax. A laconic pPRIO scheme supporting any circuit consists of the following algorithms.

- $\mathsf{LDigest}(1^\lambda, X = \{X_i\}_{i\in[N]}) \to \mathsf{dig}$. The digest algorithm takes as input the security parameter λ and an input space X of the form $X = \{X_i \in \{0,1\}^\ell\}_{i\in[N]}$ for some $\ell = \ell(\lambda)$ and $N \in \mathbb{N}$. We assume that X encodes the information of ℓ and N and one can retrieve them efficiently. It outputs a string dig.
- $\mathsf{LObfuscate}(1^\lambda, \mathsf{dig}, E) \to \mathsf{Lobf}$. The obfuscate algorithm takes as input the security parameter λ, string dig and a circuit $E : \{0,1\}^\ell \to \{0,1\}^L$ whose size is S. It outputs a ciphertext Lobf.
  We decompose this algorithm into two phases.
  - $\mathsf{LObfOff}(1^\lambda, 1^S) \to (\mathsf{Lobf}_{\mathsf{off}}, \mathsf{st})$. The offline obfuscate algorithm takes as input the security parameter λ and the circuit size S. It outputs $\mathsf{Lobf}_{\mathsf{off}}$ and a state st.
  - $\mathsf{LObfOn}(\mathsf{st}, \mathsf{dig}, E) \to \mathsf{Lobf}_{\mathsf{on}}$. The online obfuscate algorithm takes as input the state st, string dig and circuit E. It outputs $\mathsf{Lobf}_{\mathsf{on}}$.

  With the above decomposition, the obfuscate algorithm outputs $\mathsf{Lobf} = (\mathsf{Lobf}_{\mathsf{off}}, \mathsf{Lobf}_{\mathsf{on}})$.
- $\mathsf{LEval}(X, \mathsf{Lobf}) \to Y$. The evaluation algorithm takes as input $X = \{X_i \in \{0,1\}^\ell\}_{i\in[N]}$ and Lobf. It outputs $Y = \{Y_i \in \{0,1\}^L\}_{i\in[N]}$.

Definition 4.1 (Compactness). For all $\ell, N \in \mathbb{N}$ and $X = \{X_i \in \{0,1\}^\ell\}_{i\in[N]}$, we need that the size of $\mathsf{dig} = \mathsf{LDigest}(1^\lambda, X)$ should be bounded by $\mathrm{poly}(\lambda)$. In particular, the size of dig should be independent of N.

Definition 4.2 (Correctness). For all $\ell, N \in \mathbb{N}$, $X = \{X_i \in \{0,1\}^\ell\}_{i\in[N]}$, and circuit $E : \{0,1\}^\ell \to \{0,1\}^L$ such that $|E| \le S$ for an arbitrary polynomial $S = S(\lambda)$, we have

$$\Pr\left[\mathsf{LEval}\left(X, (\mathsf{Lobf}_{\mathsf{off}}, \mathsf{Lobf}_{\mathsf{on}})\right) = \{E(X_i)\}_{i\in[N]} : \begin{array}{c} \mathsf{LDigest}(1^\lambda, X = \{X_i\}_{i\in[N]}) \to \mathsf{dig} \\ \mathsf{LObfOff}(1^\lambda, 1^S) \to (\mathsf{Lobf}_{\mathsf{off}}, \mathsf{st}) \\ \mathsf{LObfOn}(\mathsf{st}, \mathsf{dig}, E) \to \mathsf{Lobf}_{\mathsf{on}} \end{array}\right] = 1.$$

Remark 4.3. We note that the above correctness requirement implies that for the correctness to hold, $\mathsf{LObfOff}$ only has to know the upper bound S on the size of the circuit E that is going to be input to LObfOn and does not have to know anything else.

Definition 4.4 (Security). Let Samp be a PPT algorithm that on input 1^λ, outputs

$$\left(\mathsf{aux}, 1^S, X^1 = \{X_i^1\}_{i\in[N^1]}, \dots, X^Q = \{X_i^Q\}_{i\in[N^Q]}, E^1, \dots, E^Q\right)$$

where for all for $k \in [Q], i \in [N^k]$, $X_i^k \in \{0,1\}^{\ell^k}$, $E^k : \{0,1\}^{\ell^k} \to \{0,1\}^{L^k}$ and $|E^k| \le S$. We say that a laconic pPRIO scheme is secure with respect to the sampler class $\mathcal{SC}$ if for every PPT sampler $\mathsf{Samp} \in \mathcal{SC}$ the following holds.

If $\left(\mathsf{aux}, 1^S, X^1, \dots, X^Q, \{E^1(X_i^1)\}_{i\in[N^1]}, \dots, \{E^Q(X_i^Q)\}_{i\in[N^Q]},\right) \approx_c \left(\mathsf{aux}, 1^S, X^1, \dots, X^Q, \{\Delta_i^1\}_{i\in[N^1]}, \dots, \{\Delta_i^Q\}_{i\in[N^Q]}\right)$

then $\left(\mathsf{aux}, X^1, \dots, X^Q, \mathsf{Lobf}_{\mathsf{off}}, \mathsf{Lobf}_{\mathsf{on}}^1, \dots, \mathsf{Lobf}_{\mathsf{on}}^Q\right) \approx_c \left(\mathsf{aux}, X^1, \dots, X^Q, \mathsf{Lobf}_{\mathsf{off}}, \delta^1 \dots, \delta^Q\right),$

where

$$\Delta_i^k \leftarrow \{0,1\}^{L^k} \text{ for } k \in [Q], i \in [N^k],$$
$$(\mathsf{Lobf}_{\mathsf{off}}, \mathsf{st}) \leftarrow \mathsf{LObfOff}(1^\lambda, 1^S),\ \mathsf{dig}^k \leftarrow \mathsf{LDigest}(1^\lambda, X^k),\ \mathsf{Lobf}_{\mathsf{on}}^k \leftarrow \mathsf{LObfOn}(\mathsf{dig}^k, E^k),$$
$$\delta^k \leftarrow \mathcal{O}_{\mathsf{on}} \text{ for } k \in [Q], \text{ where } \mathcal{O}_{\mathsf{on}} \text{ is the co-domain of } \mathsf{LObfOn} \text{ algorithm.}$$

Remark 4.5. It is shown in [AMYY25, BDJ+24] that there is no laconic pPRIO scheme satisfying the above security for all general samplers. Therefore, when we use the security of laconic pPRIO, 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 laconic pPRIO that is secure for all the samplers.

4.2 Construction

In this section we construct a laconic pPRIO scheme for any circuit.

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

1. A pseudorandom function 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 $(i\|j\|b)$ where $i \in [N]$, $j \in [\ell]$, and $b \in \{0,1\}$. Here, N and ℓ are some polynomial functions in λ. Since we have $2^\lambda > \mathrm{poly}(\lambda)$, the input space of the PRF is large enough to accommodate such inputs with an appropriate encoding. It is known that PRF can be constructed from one-way functions.

2. A blind garbling scheme $\mathsf{bGC} = (\mathsf{Garble}, \mathsf{bGC.Eval}, \mathsf{bGC.Sim})$ (defined in Section 2.6) for any circuit. Without loss of generality, we assume the labels are in $\{0,1\}^\lambda$. We also assume the randomness space of Garble algorithm to be $\{0,1\}^\lambda$. If longer randomness is required by the algorithm, we can expand it by using a PRF. We require the scheme to be secure as per Definition 2.20 and Definition 2.21. We can construct bGC with the required properties assuming one-way functions (See Theorem 2.23).

3. A pPRIO scheme $\mathsf{pPRIO}.(\mathsf{ObfOff}, \mathsf{ObfOn}, \mathsf{Eval})$. We require the scheme to be secure as per Definition 2.30. We can construct pPRIO with the required properties assuming evasive LWE and LWE (See Theorem 2.32).

4. A blind batch encryption scheme $\mathsf{BBE} = \mathsf{BBE}.(\mathsf{Setup}, \mathsf{Gen}, \mathsf{SingleEnc}, \mathsf{SingleDec})$ with message space $\{0,1\}^\lambda$. We assume the randomness space of SingleEnc algorithm to be $\{0,1\}^\lambda$ and denote the ciphertext space by $\mathcal{CT}_{\mathsf{BBE}} = \{0,1\}^{\ell^{\mathsf{ct}}_{\mathsf{BBE}}}$. We require the scheme to secure as per Definition 2.26 and Definition 2.27. We also require that the CRS is a uniformly random string. As for the efficiency requirement, we need the size of the CRS to be $|\mathsf{crs}| = \mathrm{poly}(\lambda, \log N, \log \ell)$ and the size of the single ciphertext to be $|\mathsf{BBE.ct}| = \mathrm{poly}(\lambda, \log N, \log \ell)$. We can construct BBE with the required properties assuming the LWE assumption (See Theorem A.5).

Now, we describe our construction.

$\mathsf{LDigest}(1^\lambda, X = \{X_i\}_{i\in[N]})$. The digest algorithm does the following.

- Retrieve N and ℓ from the input and run $\mathsf{crs} \leftarrow \mathsf{BBE.Setup}(1^\lambda, 1^{\ell N})$.
- Compute $h := \mathsf{BBE.Gen}(\mathsf{crs}, X_1\|\cdots\|X_N)$, where $X_1\|\cdots\|X_N \in \{0,1\}^{\ell N}$ is the concatenation of the bit strings $X_1, \ldots, X_N \in \{0,1\}^\ell$.
- Output $\mathsf{dig} := (\mathsf{crs}, h, N, \ell)$.

$\mathsf{LObfOff}(1^\lambda, 1^S)$. The offline obfuscate algorithm does the following.

- Generate $(\mathsf{pPRIO.obf}_{\mathsf{off}}, \mathsf{pPRIO.st}) \leftarrow \mathsf{pPRIO.ObfOff}(1^\lambda, 1^{\mathsf{size}})$ where $\mathsf{size} = \mathrm{poly}(S, \lambda)$ is the maximum size of the circuit $C[\mathsf{dig}, E, \mathsf{sd}]$ defined in Figure 3 when the size of E is bounded by S.
- Output $\mathsf{Lobf}_{\mathsf{off}} := \mathsf{pPRIO.obf}_{\mathsf{off}}$ and $\mathsf{st} := \mathsf{pPRIO.st}$.

$\mathsf{LObfOn}(\mathsf{st}, \mathsf{dig}, E)$. The online obfuscate algorithm does the following.

- Parse $\mathsf{dig} = (\mathsf{crs}, h, N, \ell)$ and $\mathsf{st} = \mathsf{pPRIO.st}$.
- Sample a PRF key $\mathsf{sd} \leftarrow \{0,1\}^\lambda$.
- Construct a circuit $C[\mathsf{dig}, E, \mathsf{sd}]$ as in Figure 3 and compute $\mathsf{pPRIO.obf}_{\mathsf{on}} \leftarrow \mathsf{pPRIO.ObfOn}(\mathsf{pPRIO.st}, C[\mathsf{dig}, E, \mathsf{sd}])$.
- Output $\mathsf{Lobf}_{\mathsf{on}} := (\mathsf{crs}, \mathsf{pPRIO.obf}_{\mathsf{on}})$.

$\mathsf{LEval}(X, \mathsf{Lobf})$. The evaluation algorithm does the following.

- Parse $X = \{X_i\}_{i \in [N]}$ and $\mathsf{Lobf} = (\mathsf{Lobf}_{\mathsf{off}} = \mathsf{pPRIO.obf}_{\mathsf{off}}, \mathsf{Lobf}_{\mathsf{on}} = (\mathsf{crs}, \mathsf{pPRIO.obf}_{\mathsf{on}}))$.
- Set $\mathsf{pPRIO.obf} := (\mathsf{pPRIO.obf}_{\mathsf{off}}, \mathsf{pPRIO.obf}_{\mathsf{on}})$ and run $\mathsf{pPRIO.Eval}(\mathsf{pPRIO.obf}, i) \rightarrow y_i$ for $i \in [N]$.
- Parse $y_i = (\{\mathsf{BBE.ct}_{i,j}\}_{j \in [\ell]}, \tilde{E}_i)$ for each $i \in [N]$.
- Compute $\mathsf{lab}_{i,j} \leftarrow \mathsf{BBE.SingleDec}(\mathsf{crs}, X, \ell(i-1)+j, \mathsf{BBE.ct}_{i,j})$ for each $i \in [N]$ and $j \in [\ell]$ and set $\mathsf{lab}_i := \{\mathsf{lab}_{i,j}\}$.
- Compute $z_i = \mathsf{bGC.Eval}(\mathsf{lab}_i, \tilde{E}_i)$ for $i \in [N]$.
- Output $\{z_i\}_{i \in [N]}$.

Circuit $C[\mathsf{dig}, E, \mathsf{sd}](i)$**:**

Hardwired constant: $\mathsf{dig} = (\mathsf{crs}, h, N, \ell)$, circuit E, and PRF seed sd.
Given input $i \in [N]$, do the following:

1. Compute $R_i := \mathsf{PRF}(\mathsf{sd}, i\|1\|0)$ and $S_{i,j} := \mathsf{PRF}(\mathsf{sd}, i\|j\|1)$ for $j \in [\ell]$.
2. $(\{\mathsf{lab}_{i,j,b}\}_{j \in [\ell], b \in \{0,1\}}, \tilde{E}_i) \leftarrow \mathsf{Garble}(1^\lambda, E; R_i)$.
3. Compute $\mathsf{BBE.ct}_{i,j} \leftarrow \mathsf{BBE.SingleEnc}(\mathsf{crs}, h, (i-1)\ell + j, (\mathsf{lab}_{i,j,0}, \mathsf{lab}_{i,j,1}); S_{i,j})$ for $j \in [\ell]$.
4. Output $(\{\mathsf{BBE.ct}_{i,j}\}_{j \in [\ell]}, \tilde{E}_i)$.

Figure 3: The circuit C garbles input circuit E and provides BBE encryptions of its labels.

Correctness. For $X = \{X_i\}_{i \in [N]}$ and $\mathsf{Lobf} = (\mathsf{Lobf}_{\mathsf{off}} = \mathsf{pPRIO.obf}_{\mathsf{off}}, \mathsf{Lobf}_{\mathsf{on}} = (\mathsf{crs}, \mathsf{pPRIO.obf}_{\mathsf{on}}))$, we have $\mathsf{pPRIO.obf}_{\mathsf{on}} \leftarrow \mathsf{pPRIO.ObfOn}(\mathsf{pPRIO.st}, C[\mathsf{dig}, E, \mathsf{sd}])$. From the correctness of pPRIO and the definition of $C[\mathsf{dig}, E, \mathsf{sd}]$, for each $i \in [N]$ we get

$$\mathsf{pPRIO.Eval}(\mathsf{pPRIO.obf}, i) = y_i = (\{\mathsf{BBE.ct}_{i,j}\}_{j \in [\ell]}, \tilde{E}_i)$$

where $\mathsf{BBE.ct}_{i,j} \leftarrow \mathsf{BBE.SingleEnc}(\mathsf{crs}, h, (i-1)\ell + j, (\mathsf{lab}_{i,j,0}, \mathsf{lab}_{i,j,1}); S_{i,j})$ for $(\{\mathsf{lab}_{i,j,b}\}_{j \in [\ell], b \in \{0,1\}}, \tilde{E}_i) \leftarrow \mathsf{Garble}(1^\lambda, E_i; R_i)$. Next, from the perfect correctness of the BBE scheme we get

$$\mathsf{BBE.SingleDec}(\mathsf{crs}, X, \ell(i-1)+j, \mathsf{BBE.ct}_{i,j}) = \mathsf{lab}_{i,j} \text{ for } i \in [N],\ j \in [\ell]$$

Next, we set $\mathsf{lab}_i := \{\mathsf{lab}_{i,j}\}$, which are the labels corresponding to input X_i for $i \in [N]$. Now, from the correctness of the bGC scheme and definition of E_i, for each $i \in [N]$ we get

$$\mathsf{bGC.Eval}(\mathsf{lab}_i, \tilde{E}_i) = E(X_i)$$

and hence the correctness.

Efficiency. First, we note the following.

1. Instantiating the BBE scheme as in Theorem A.5, we have $|\mathsf{crs}| = \mathrm{poly}(\lambda, \log N, \log \ell)$ and $|\mathsf{BBE.ct}| = \mathrm{poly}(\lambda, \log N, \log \ell)$, which are bounded by a fixed polynomial $\mathrm{poly}(\lambda)$ for any polynomially bounded N and ℓ.
2. Instantiating the pPRIO scheme as in Theorem 2.32, we have $|\mathsf{pPRIO.obf}_{\mathsf{off}}| = \mathrm{poly}(S, \lambda)$, $|\mathsf{obf}_{\mathsf{on}}| = \mathrm{poly}(S, \lambda)$.

From the above observations, our laconic pPRIO scheme satisfies

$$|\mathsf{dig}| = O(\lambda),\ |\mathsf{Lobf}_{\mathsf{off}}| = \mathrm{poly}(S, \lambda),\ |\mathsf{Lobf}_{\mathsf{on}}| = \mathrm{poly}(S, \lambda).$$

We formalise the instantiation using the following theorem. The security of our laconic pPRIO will be proven in Theorem 4.7.

Theorem 4.6. Assuming LWE and evasive LWE assumptions, there exists a secure (Definition 4.4) laconic pPRIO scheme satisfying

$$|\mathsf{dig}| = O(\lambda),\ |\mathsf{Lobf}_{\mathsf{off}}| = \mathrm{poly}(S, \lambda),\ |\mathsf{Lobf}_{\mathsf{on}}| = \mathrm{poly}(S, \lambda).$$

where $\mathsf{dig} \leftarrow \mathsf{LDigest}(1^\lambda, X = \{X_i\}_{i \in [N]})$, $(\mathsf{Lobf}_{\mathsf{off}}, \mathsf{Lobf}_{\mathsf{on}}) \leftarrow \mathsf{LObfuscate}(1^\lambda, \mathsf{dig}, E)$ for circuit $E : \{0,1\}^\ell \to \{0,1\}^L$ whose size is bounded by $S = S(\lambda)$.

4.3 Security Proof

Theorem 4.7. Let $\mathcal{SC}_{l\text{-}\mathsf{pPRIO}}$ be a sampler class for laconic pPRIO. Assume pPRIO is secure (Definition 2.30) with respect to the sampler class that contains all $\mathsf{Samp}_{\mathsf{pPRIO}}(1^\lambda)$, induced by $\mathsf{Samp}_{l\text{-}\mathsf{pPRIO}} \in \mathcal{SC}_{l\text{-}\mathsf{pPRIO}}$, as defined in Equation (20), BBE satisfies security (Definition 2.26) and strong blindness (Definition 2.27), bGC satisfies security (Definition 2.20) and blindness (Definition 2.21) and PRF is secure. Then the above construction of laconic pPRIO satisfies security as defined in Definition 4.4.

Proof. Let us consider a sampler $\mathsf{Samp}_{l\text{-}\mathsf{pPRIO}}$ that outputs

$$\left(\mathsf{aux}, \{X^k = \{X_i^k \in \{0,1\}^{\ell^k}\}_{i \in [N^k]}\}_{k \in [Q]}, \{E^k\}_{k \in [Q]}\right).$$

To prove the theorem, we show that

$$\left(\mathsf{aux}, \mathsf{Lobf}_{\mathsf{off}} = \mathsf{pPRIO.obf}_{\mathsf{off}}, \{X^k, \mathsf{Lobf}^k_{\mathsf{on}} = (\mathsf{crs}^k, \mathsf{pPRIO.obf}^k_{\mathsf{on}})\}_{k \in [Q]}\right)$$
$$\approx_c \left(\mathsf{aux}, \mathsf{Lobf}_{\mathsf{off}} = \mathsf{pPRIO.obf}_{\mathsf{off}}, \{X^k, \delta^k \leftarrow \mathcal{O}_{\mathsf{on}}\}_{k \in [Q]}\right)$$

holds assuming

$$\left(\mathsf{aux}, \{X^k, \{E^k(X_i^k)\}_{i \in [N^k]}\}_{k \in [Q]}\right) \approx_c \left(\mathsf{aux}, \{X^k, \{\Delta_i^k \leftarrow \{0,1\}^{L^k}\}_{i \in [N^k]}\}_{k \in [Q]}\right) \tag{19}$$

where $\mathcal{O}_{\mathsf{on}}$ is the co-domain of LObfOn algorithm. Recalling that each crs^k is a random strting, it suffices to prove that $\{\mathsf{pPRIO.obf}^k_{\mathsf{on}}\}_k$ is pseudorandom.
We invoke the security of pPRIO scheme with respect to a sampler $\mathsf{Samp}_{\mathsf{pPRIO}}(1^\lambda)$ that outputs

$$\left(1^{N^1 + \cdots + N^Q},\ \mathsf{aux}_{\mathsf{pPRIO}} = (\mathsf{aux}, \{\mathsf{crs}^k\}_{k \in [Q]}, \{X^k\}_{k \in [Q]}),\ \left\{C^k[\mathsf{dig}^k, E^k, \mathsf{sd}^k]\right\}_{k \in [Q]}\right) \tag{20}$$

where $\mathsf{crs}^k \leftarrow \mathsf{BBE.Setup}(1^\lambda, 1^{\ell^k N^k})$, $h^k = \mathsf{BBE.Gen}(\mathsf{crs}^k, X_1^k \| \cdots \| X_{N^k}^k)$, $\mathsf{dig}^k = (\mathsf{crs}^k, h^k, N^k, \ell^k)$, and $\mathsf{sd}^k \leftarrow \{0,1\}^\lambda$. From the security of pPRIO, we can see that it suffices to prove

$$\left(\mathsf{aux}, \{\mathsf{crs}^k\}_{k \in [Q]}, \{X^k\}_{k \in [Q]}, \left\{C^k[\mathsf{dig}^k, E^k, \mathsf{sd}^k](i)\right\}_{k \in [Q], i \in [N^k]}\right) \approx_c \left(\mathsf{aux}, \{\mathsf{crs}^k\}_{k \in [Q]}, \{X^k\}_{k \in [Q]}, \left\{\gamma_i^k\right\}_{k \in [Q], i \in [N^k]}\right) \tag{21}$$

where $C^k[\mathsf{dig}^k, E^k, \mathsf{sd}^k](i) = (\{\mathsf{BBE.ct}^k_{i,j}\}_{j\in[\ell^k]}, \tilde{E}^k_i)$ for $i \in [N^k], k \in [Q]$ and $\gamma^k_i \leftarrow \mathcal{CT}^{\ell^k}_{\mathsf{BBE}} \times \{0,1\}^{\ell^k_{\mathsf{bGC}}}$. Here, ℓ^k_{bGC} is the length of the binary string $\tilde{E}^k_i$. To prove Equation (21), we consider the following sequence of games.

Hyb_0. This is the LHS distribution of Equation (21). Rearranging the terms and recalling the definition of the circuit, we can rewrite the distribution as

$$\left(\mathsf{aux}, \left\{\mathsf{crs}^k, X^k, \left\{\{\mathsf{BBE.ct}^k_{i,j}\}_{j\in[\ell^k]}, \tilde{E}^k_i\right\}_{i\in[N^k]}\right\}_{k\in[Q]}\right)$$

where $(\{\mathsf{lab}^k_{i,j,b}\}_{j\in[\ell^k],b\in\{0,1\}}, \tilde{E}^k_i) \leftarrow \mathsf{Garble}(1^\lambda, E^k; R^k_i)$ and $\mathsf{BBE.ct}^k_{i,j} \leftarrow \mathsf{SingleEnc}(\mathsf{crs}, h^k, (i-1)\ell^k + j, (\mathsf{lab}^k_{i,j,0}, \mathsf{lab}^k_{i,j,1}); S^k_{i,j})$ for $R^k_i := \mathsf{PRF}(\mathsf{sd}^k, i\|1\|0)$ and $S^k_{i,j} := \mathsf{PRF}(\mathsf{sd}^k, i\|j\|1)$.

Hyb_1. This hybrid is same as the previous one except that we compute each R^k_i and $S^k_{i,j}$ differently. Concretely, we sample $R^k_i, S^k_{i,j} \leftarrow \{0,1\}^\lambda$ for all $j \in [\ell^k], i \in [N^k]$ and $k \in [Q]$. This hybrid is computationally indistinguishable from the previous one due to the security of PRF.

Hyb_2. In this hybrid, we change how we compute $\mathsf{BBE.ct}^k_{i,j}$. Namely, we set

$$\mathsf{BBE.ct}^k_{i,j} \leftarrow \mathsf{SingleEnc}(\mathsf{crs}, h^k, (i-1)\ell^k + j, (\overline{\mathsf{lab}}^k_{i,j,0}, \overline{\mathsf{lab}}^k_{i,j,1})) \quad \text{where} \quad \overline{\mathsf{lab}}^k_{i,j,b} \begin{cases} = \mathsf{lab}^k_{i,j,b} & \text{if} \quad b = X^k_{i,j} \\ \leftarrow \{0,1\}^\lambda & \text{otherwise} \end{cases}.$$

In the above, $X^k_{i,j}$ is the j-th bit of X^k_i. By the security of BBE, this hybrid is computationally indistinguishable from the previous one. To see this, observe that the decryption outputs the labels corresponding to the j-th bit of X^k_i and we only substitute $\overline{\mathsf{lab}}^k_{i,j,b} \leftarrow \{0,1\}^\lambda$ when $b \neq X^k_{i,j}$.

Hyb_3. In this hybrid, we change how we compute $\tilde{E}^k_i$ and $\overline{\mathsf{lab}}^k_{i,j,b}$. Namely, we set

$$(\{\mathsf{lab}^k_{i,j}\}_{j\in[\ell^k]}, \tilde{E}^k_i) \leftarrow \mathsf{bGC.Sim}(1^\lambda, 1^{|E^k|}, 1^{\ell^k}, E^k(X^k_i)) \quad \text{and} \quad \overline{\mathsf{lab}}^k_{i,j,b} \begin{cases} = \mathsf{lab}^k_{i,j} & \text{if} \quad b = X^k_{i,j} \\ \leftarrow \{0,1\}^\lambda & \text{otherwise} \end{cases}.$$

By the simulation security of bGC (Definition 2.20), this hybrid is computationally indistinguishable from the previous one. To see this, it suffices to observe that only the information of $\{\mathsf{lab}^k_{i,j,X_{i,j}}\}_{i,j,k}$ is necessary for simulating Hyb_2 and the labels $\{\mathsf{lab}^k_{i,j,1-X_{i,j}}\}_{i,j,k}$ are not necessary.

Hyb_4. In this hybrid, we change how we compute $\tilde{E}^k_i$ and $\mathsf{lab}^k_{i,j}$. Namely, we set

$$(\{\mathsf{lab}^k_{i,j}\}_{j\in[\ell]}, \tilde{E}^k_i) \leftarrow \mathsf{bGC.Sim}(1^\lambda, 1^{|E^k|}, 1^\ell, \Delta^k_i),$$

where $\Delta^k_i \leftarrow \{0,1\}^{L^k}$ is chosen uniformly at random. This hybrid is computationally indistinguishable from the previous one by Equation (19). To see this note that Equation (19) implies $\{E^k(X^k_i)\}_{i\in[N^k]} \approx_c \{\Delta^k_i \leftarrow \{0,1\}^{L^k}\}_{i\in[N^k]}$, for $k \in [Q]$, given $\mathsf{aux}, \{X^k\}_{k\in[Q]}$.

Hyb_5. In this hybrid, we sample $\mathsf{lab}^k_{i,j}$ and $\tilde{E}^k_i$ as random strings. In particular, we sample $\mathsf{lab}^k_{i,j} \leftarrow \{0,1\}^\lambda$ and $\tilde{E}^k_i \leftarrow \{0,1\}^{\ell^k_{\mathsf{bGC}}}$. By the blindness of bGC scheme (Definition 2.21), this hybrid is computationally indistinguishable from the previous one. To see this, note that in the previous hybrid the simulator $\mathsf{bGC.Sim}$ takes as input Δ^k_i which is a uniformly random string and thus by the blindness property of bGC we can replace the

output of $\mathsf{bGC.Sim}$ by a completely random string.
The view of the adversary in this hybrid is as follows.

$$\left(\mathsf{aux}, \left\{\mathsf{crs}^k, X^k, \left\{\{\mathsf{BBE.ct}^k_{i,j}\}_{j\in[\ell]}, \tilde{E}^k_i\right\}_{i\in[N^k]}\right\}_{k\in[Q]}\right)$$

where $\mathsf{BBE.ct}^k_{i,j} \leftarrow \mathsf{SingleEnc}(\mathsf{crs}, h^k, (i-1)\ell^k + j, (\overline{\mathsf{lab}}^k_{i,j,0}, \overline{\mathsf{lab}}^k_{i,j,1}))$ for $\overline{\mathsf{lab}}^k_{i,j,0} \leftarrow \{0,1\}^\lambda, \overline{\mathsf{lab}}^k_{i,j,1} \leftarrow \{0,1\}^\lambda$ and $\tilde{E}^k_i \leftarrow \mathsf{SIM}_{\mathsf{circ}}$.

Hyb_6. In this hybrid, we replace $\mathsf{BBE.ct}^k_{i,j}$ with a random string. Namely, we sample $\mathsf{BBE.ct}^k_{i,j} \leftarrow \mathcal{CT}_{\mathsf{BBE}}$ for all i, j, k. This hybrid is indistinguishable from previous one by strong blindness (Definition 2.27) property of the BBE scheme. To see this, note that $\mathsf{BBE.ct}^k_{i,j}$ encrypts random strings $\overline{\mathsf{lab}}^k_{i,j,0}, \overline{\mathsf{lab}}^k_{i,j,1} \leftarrow \{0,1\}^\lambda$ in the previous hybrid and thus the blindness property allows us to replace each $\mathsf{BBE.ct}^k_{i,j}$ with a random string.
The view of the adversary in this hybrid is as follows.

$$\left(\mathsf{aux}, \left\{\mathsf{crs}^k, X^k, \left\{\{\mathsf{BBE.ct}^k_{i,j} \leftarrow \mathcal{CT}_{\mathsf{BBE}}\}_{j\in[\ell]}, \tilde{E}^k_i \leftarrow \{0,1\}^{\ell^k_{\mathsf{bGC}}}\right\}_{i\in[N^k]}\right\}_{k\in[Q]}\right).$$

Rearranging the terms, we can observe that the distribution in Hyb_6 corresponds to the RHS distribution of Equation (21). This concludes the proof of Theorem 4.7. □

5 Partial-Hiding prFE for Unbounded Depth with Optimal Parameters

In this section, we extend the notion of prFE to introduce partially-hiding prFE, where the part of the input to the circuit can be public. We then construct partially hiding prFE with short parameter size in this section from several ingredients, which are all implied by evasive LWE and LWE. Our construction of partially hiding prFE will be used in Section 6 and 7.

5.1 Definition

In this section we give the definitions for partial-hiding functional encryption for pseudorandom functionalities. Consider a circuit class $\{\mathcal{C}_{\mathsf{prm}} = \{C : \mathcal{X}_{\mathsf{pub}} \times \mathcal{X}_{\mathsf{priv}} \to \mathcal{Y}\}\}_{\mathsf{prm}}$ where $C \in \mathcal{C}_{\mathsf{prm}}$ takes as input a string $x = (x_{\mathsf{pub}}, x_{\mathsf{priv}}) \in \mathcal{X}_{\mathsf{pub}} \times \mathcal{X}_{\mathsf{priv}}$ and outputs $C(x) \in \mathcal{Y}$.

Syntax. A partial-hiding functional encryption for parameterized circuits $\mathcal{C}_{\mathsf{prm}}$ by 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. We assume w.l.o.g that msk includes mpk. We also assume that prm is implicitly input to all the algorithms below.

$\mathsf{KeyGen}(\mathsf{msk}, C) \to \mathsf{sk}_C$. The key generation algorithm takes as input the master secret key msk and a circuit $C \in \mathcal{C}_{\mathsf{prm}}$ and it outputs a functional secret key sk_C.

$\mathsf{Enc}(\mathsf{mpk}, x = (x_{\mathsf{pub}}, x_{\mathsf{priv}})) \to \mathsf{ct}$. 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.
For the purpose of some applications, we consider a variant where the Enc algorithm can be decomposed into the following two phases.

$\mathsf{EncOff}(\mathsf{mpk}) \to (\mathsf{ct}_{\mathsf{off}}, \mathsf{st})$. The offline encryption algorithm takes as input the security parameter λ and outputs offline part of the ciphertext $\mathsf{ct}_{\mathsf{off}}$ and the state st.

$\mathsf{EncOn}(\mathsf{st}, x) \to \mathsf{ct}_{\mathsf{on}}$. The online encryption algorithm takes as input the state st and the input x and outputs the online part of the ciphertext $\mathsf{ct}_{\mathsf{on}}$.

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

$\mathsf{Dec}(\mathsf{mpk}, \mathbf{x}_{\mathsf{pub}}, \mathsf{sk}_C, C, \mathsf{ct}) \to y$. The decryption algorithm takes as input the master public key mpk, the public input $\mathbf{x}_{\mathsf{pub}}$, secret key sk_C, circuit C and a ciphertext ct and outputs $y \in \mathcal{Y}_{\mathsf{prm}}$.

Definition 5.1 (Correctness). A PHprFE scheme is said to satisfy *perfect* correctness if for all prm, any input $x = (x_{\mathsf{pub}}, x_{\mathsf{priv}}) \in \mathcal{X}_{\mathsf{pub}} \times \mathcal{X}_{\mathsf{priv}}$ and circuit $C \in \mathcal{C}_{\mathsf{prm}}$, we have

$$\Pr\begin{bmatrix} (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{prm}) \ , \ \mathsf{sk}_C \leftarrow \mathsf{KeyGen}(\mathsf{msk}, C), \\ \mathsf{Dec}\big(\mathsf{mpk}, \mathbf{x}_{\mathsf{pub}}, \mathsf{sk}_C, C, \mathsf{Enc}(\mathsf{mpk}, x)\big) = C(x) \end{bmatrix} = 1.$$

Definition 5.2 (Security). For a PHprFE scheme for circuit class $\{\mathcal{C}_{\mathsf{prm}} = \{C : \mathcal{X}_{\mathsf{pub}} \times \mathcal{X}_{\mathsf{priv}} \to \mathcal{Y}\}\}_{\mathsf{prm}}$ parameterized by $\mathsf{prm} = \mathsf{prm}(\lambda)$, let Samp be a PPT algorithm that on input 1^λ, outputs

$$(C^1, \ldots, C^Q, x = (x_{\mathsf{pub}}, x_{\mathsf{priv}}), \mathsf{aux} \in \{0,1\}^*)$$

where Q is the number of key queries, $C^k \in \mathcal{C}_{\mathsf{prm}}$ for $k \in [Q]$, $x = (x_{\mathsf{pub}}, x_{\mathsf{priv}}) \in \mathcal{X}_{\mathsf{pub}} \times \mathcal{X}_{\mathsf{priv}}$.
We define the following advantage functions:

$$\begin{aligned}\mathsf{Adv}^{\mathrm{PRE}}_{\mathcal{A}_0}(\lambda) \overset{\mathrm{def}}{=} &\Pr\Big[\mathcal{A}_0\Big(\ \mathsf{aux},\ \{C^k,\ x_{\mathsf{pub}},\ C^k(x)\}_{k\in[Q]}\Big) = 1\Big] \\ &- \Pr\Big[\mathcal{A}_0\Big(\ \mathsf{aux},\ \{C^k,\ x_{\mathsf{pub}},\ \Delta^k \leftarrow \mathcal{Y}\}_{k\in[Q]}\Big) = 1\Big]\end{aligned}$$

$$\begin{aligned}\mathsf{Adv}^{\mathrm{POST}}_{\mathcal{A}_1}(\lambda) \overset{\mathrm{def}}{=} &\Pr\Big[\mathcal{A}_1(\mathsf{mpk},\ \mathsf{aux},\ \{C^k,\ x_{\mathsf{pub}},\ \mathsf{Enc}(\mathsf{mpk}, x),\ \mathsf{sk}_{C^k}\}_{k\in[Q]}) = 1\Big] \\ &- \Pr\Big[\mathcal{A}_1(\mathsf{mpk},\ \mathsf{aux},\ \{C^k,\ x_{\mathsf{pub}},\ \delta \leftarrow \mathsf{Sim}(1^\lambda),\ \mathsf{sk}_{C^k}\}_{k\in[Q]}) = 1\Big]\end{aligned}$$

where $(C^1, \ldots, C^Q, x = (x_{\mathsf{pub}}, x_{\mathsf{priv}}), \mathsf{aux} \in \{0,1\}^*) \leftarrow \mathsf{Samp}(1^\lambda), (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{prm})$ and $\mathcal{CT}$ is the ciphertext space. We say that a PHprFE scheme for circuit class $\mathcal{C}_{\mathsf{prm}}$ is secure with respect to the sampler class $\mathcal{SC}$ if for every PPT sampler $\mathsf{Samp} \in \mathcal{SC}$, $\mathcal{A}_1$ and Sim, there exists another PPT $\mathcal{A}_0$ such that

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

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

Remark 5.3 (prFE as a special PHprFE). We remark that prFE is a special case of PHprFE with $\mathbf{x}_{\mathsf{pub}} = \perp$.

Next, we define the security notion that we require for the PHprFE variant where we decompose the Enc algorithm as $\mathsf{Enc} = (\mathsf{EncOff}, \mathsf{EncOn})$ and reuse the same state output by EncOff multiple times for generating the online part of the ciphertexts. We require the online part of the ciphertexts to be pseudorandom, whereas the offline part may not be.

Definition 5.4 (Reusable Security). For a PHprFE scheme for circuit class $\{\mathcal{C}_{\mathsf{prm}} = \{C : \mathcal{X}_{\mathsf{pub}} \times \mathcal{X}_{\mathsf{priv}} \to \mathcal{Y}\}\}_{\mathsf{prm}}$ parameterized by $\mathsf{prm} = \mathsf{prm}(\lambda)$, let Samp be a PPT algorithm that on input 1^λ, outputs

$$\Big(C^1, \ldots, C^{Q_{\mathsf{key}}}, x^1 = (x^1_{\mathsf{pub}}, x^1_{\mathsf{priv}}), \ldots, x^{Q_{\mathsf{msg}}} = (x^{Q_{\mathsf{msg}}}_{\mathsf{pub}}, x^{Q_{\mathsf{msg}}}_{\mathsf{priv}}), \mathsf{aux} \in \{0,1\}^*\Big)$$

where Q_{key} and Q_{msg} are the number of key queries and messages respectively, $x^j = (x^j_{\mathsf{pub}}, x^j_{\mathsf{priv}}) \in \mathcal{X}_{\mathsf{pub}} \times \mathcal{X}_{\mathsf{priv}}$ for $j \in [Q_{\mathsf{msg}}]$, $C^k \in \mathcal{C}_{\mathsf{prm}}$ for $k \in [Q]$.
We define the following advantage functions:

$$\begin{aligned}\mathsf{Adv}^{\mathrm{PRE}}_{\mathcal{A}_0}(\lambda) \overset{\mathrm{def}}{=} &\Pr\Big[\mathcal{A}_0\Big(\ \mathsf{aux},\ \{C^k,\ x^j_{\mathsf{pub}},\ C^k(x^j)\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]}\Big) = 1\Big] \\ &- \Pr\Big[\mathcal{A}_0\Big(\ \mathsf{aux},\ \{C^k,\ x^j_{\mathsf{pub}},\ \Delta^{j,k} \leftarrow \mathcal{Y}\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]}\Big) = 1\Big]\end{aligned}$$

$$\mathsf{Adv}^{\mathrm{POST}}_{\mathcal{A}_1}(\lambda) \overset{\mathrm{def}}{=} \Pr\Big[\mathcal{A}_1(\mathsf{mpk},\ \mathsf{aux},\ \mathsf{ct}_{\mathsf{off}},\ \{x^j_{\mathsf{pub}},\ \mathsf{ct}^j_{\mathsf{on}}\}_{j\in[Q_{\mathsf{msg}}]},\ \{C^k,\ \mathsf{sk}_{C^k}\}_{k\in[Q_{\mathsf{key}}]}) = 1\Big]$$
$$- \Pr\Big[\mathcal{A}_1(\mathsf{mpk},\ \mathsf{aux},\ \mathsf{ct}_{\mathsf{off}},\ \{x^j_{\mathsf{pub}},\ \delta^j\}_{j\in[Q_{\mathsf{msg}}]},\ \{C^k,\ \mathsf{sk}_{C^k}\}_{k\in[Q_{\mathsf{key}}]}) = 1\Big]$$

where $(C^1,\dots,C^{Q_{\mathsf{key}}}, x^1 = (x^1_{\mathsf{pub}}, x^1_{\mathsf{priv}}),\dots,x^{Q_{\mathsf{msg}}} = (x^{Q_{\mathsf{msg}}}_{\mathsf{pub}}, x^{Q_{\mathsf{msg}}}_{\mathsf{priv}}), \mathsf{aux} \in \{0,1\}^*) \leftarrow \mathsf{Samp}(1^\lambda)$, $(\mathsf{mpk},\mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda,\mathsf{prm})$, $(\mathsf{ct}_{\mathsf{off}},\mathsf{st}) \leftarrow \mathsf{EncOff}(1^\lambda)$, $\mathsf{ct}^j_{\mathsf{on}} \leftarrow \mathsf{EncOn}(\mathsf{st},x^j)$, $\delta^j \leftarrow \mathcal{CT}_{\mathsf{on}}$ for $j \in [Q_{\mathsf{msg}}]$ and $\mathcal{CT}_{\mathsf{on}}$ is the online part of the ciphertext space. We say that a PHprFE scheme for circuit class $\mathcal{C}_{\mathsf{prm}}$ is secure with respect to the sampler class $\mathcal{SC}$ if for every PPT sampler $\mathsf{Samp} \in \mathcal{SC}$ and $\mathcal{A}_1$, there exists another PPT $\mathcal{A}_0$ such that

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

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

Remark 5.5. Similar to Remark 3.3, when we use the security of PHprFE, 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 PHprFE that is secure for all the samplers.

Remark 5.6. We remark that Definition 5.4 is stronger security notion than Definition 5.2, since the former collapses to the latter if we restrict the adversary so that $Q_{\mathsf{msg}} = 1$ and set the simulator so that it runs $\mathsf{EncOff}(\mathsf{mpk}) \to (\mathsf{ct}_{\mathsf{off}},\mathsf{st})$, samples $\delta \leftarrow \mathcal{CT}_{\mathsf{on}}$, and outputs $(\mathsf{ct}_{\mathsf{off}},\delta)$.

Remark 5.7. We remark that Definition 5.4 is in the single-challenge flavor in that only single offline part of the challenge ciphertext is given to the adversary (though multiple online part of the ciphertext is given to it). While it does not seem to imply multi-challenge flavor of the security definition (See Remark 3.4), we only define this simpler version of the definition since it suffices for our applications and leave the extension to the multi-challenge flavor for the future work.

5.2 Construction

In this section we provide our construction of a PHprFE scheme for circuit family $\mathcal{C}_{L_{\mathsf{pub}},L_{\mathsf{priv}}} = \{C : \{0,1\}^{L_{\mathsf{pub}}(\lambda)} \times \{0,1\}^{L_{\mathsf{priv}}(\lambda)} \to \{0,1\}\}$. Namely, the construction supports a class of circuits whose public and private input lengths are fixed and output is binary, but its size and depth are unbounded.

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

1. A blind garbling scheme $\mathsf{bGC} = (\mathsf{Garble},\mathsf{Eval},\mathsf{bGC.Sim})$ (defined in Section 2.6) with decomposability (defined in Definition 2.22). Without loss of generality, we assume the labels are in $\{0,1\}^\lambda$ and the random coins used by the algorithm $\{\mathsf{Garble}_i\}_i$ is in $\{0,1\}^\lambda$. The latter is for the sake of notational convenience and can be achieved by using a PRF to derive longer (pseudo-)random coins if needed. We also use $\mathcal{CT}^i_{\mathsf{bGC}}$ to denote the co-domain of $\{\mathsf{Garble}_i\}_i$ algorithm. We can construct bGC with the required properties assuming one-way functions (See Theorem 2.23).

2. 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 $\{0,1\}^\lambda$. It is known that PRF can be constructed from one-way functions.

3. A laconic pPRIO scheme $\mathsf{LprIO} = (\mathsf{LDigest},\mathsf{LObfOff},\mathsf{LObfOn},\mathsf{LEval})$. Without loss of generality, the random coins used by LObfOn is in $\{0,1\}^\lambda$ and the state output by LObfOff is in $\{0,1\}^\lambda$. The former can be satisfied by using a PRF. The latter can be achieved in two steps. We first let the state Lst to be the randomness used by LObfOff and then replace it with a string in $\{0,1\}^\lambda$, which can be done by using a PRF. The length of the digest is of fixed polynomial in the security parameter and we denote it by $L_{\mathsf{LDigest}}(\lambda)$. As we show in Theorem 4.6, such a laconic pPRIO can be constructed by assuming evasive LWE and LWE.

4. A prFE scheme $\mathsf{prFE} = \mathsf{prFE}.(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ for circuit class $\mathcal{C}_{\mathsf{inp}(\lambda),\mathsf{dep}(\lambda),\mathsf{out}(\lambda)}$ consisting of circuits with input length $\mathsf{inp}(\lambda) = L_{\mathsf{priv}} + L_{\mathsf{LDigest}}(\lambda) + 4\lambda$, maximum depth $\mathsf{dep}(\lambda)$ and output length $\mathsf{out}(\lambda)$. We set the maximum depth and output length so that the circuit class supports the circuits $F_0[\mathbf{r}]$, $F_1[\mathbf{r}]$, and $F_2[\mathbf{r}, \mathsf{dig}_C]$ defined as Figure 4, 5, and 6, respectively. We denote the information $(1^{\mathsf{inp}(\lambda)}, 1^{\mathsf{dep}(\lambda)}, 1^{\mathsf{out}(\lambda)})$ specifying the circuit class by prm and the ciphertext space of the prFE scheme by $\mathcal{CT}_{\mathsf{prFE}} = \{0,1\}^{\ell^{\mathsf{ct}}_{\mathsf{prFE}}}$. Assuming LWE and evasive LWE, we can construct prFE with the required parameters (See Theorem 3.6).

For our construction, we set the following parameters

$\mathsf{Setup}(1^\lambda, 1^{L_{\mathsf{pub}}}, 1^{L_{\mathsf{priv}}})$. The setup algorithm does the following.

- Run $(\mathsf{prFE.mpk}, \mathsf{prFE.msk}) \leftarrow \mathsf{prFE.Setup}(1^\lambda, \mathsf{prm})$.
- Output $\mathsf{mpk} := \mathsf{prFE.mpk}$ and $\mathsf{msk} := \mathsf{prFE.msk}$.

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

- Parse $\mathsf{msk} = \mathsf{prFE.msk}$.
- Sample a string $\mathbf{r} \leftarrow \{0,1\}^\lambda$.
- Run $\mathsf{dig}_C \leftarrow \mathsf{LDigest}(\{(i, C_i)\}_{i \in [|C|]})$, where C_i is the description of the i-th gate of C and $|C|$ is the number of gates in C. The description of C_i can be encoded into a string of length at most 4λ since it suffices to encode its index, two indices of the incoming wires, and the type of the gate.
- Construct circuits $F_0[\mathbf{r}]$, $F_1[\mathbf{r}]$, and $F_2[\mathbf{r}, \mathsf{dig}_C]$ as in Figure 4, 5, and 6 respectively.
- Run $\mathsf{prFE.sk}_0 \leftarrow \mathsf{prFE.KeyGen}(\mathsf{prFE.msk}, F_0[\mathbf{r}])$, $\mathsf{prFE.sk}_1 \leftarrow \mathsf{prFE.KeyGen}(\mathsf{prFE.msk}, F_1[\mathbf{r}])$, and $\mathsf{prFE.sk}_2 \leftarrow \mathsf{prFE.KeyGen}(\mathsf{prFE.msk}, F_2[\mathbf{r}, \mathsf{dig}_C])$.
- Output $\mathsf{sk}_C := (\mathbf{r}, \mathsf{prFE.sk}_0, \mathsf{prFE.sk}_1, \mathsf{prFE.sk}_2)$.

$\mathsf{Enc}(\mathsf{mpk}, \mathbf{x} = (\mathbf{x}_{\mathsf{pub}}, \mathbf{x}_{\mathsf{priv}}))$. The encryption algorithm does the following. We divide the algorithm into the following two steps.

$\mathsf{EncOff}(\mathsf{mpk})$. It takes as input $\mathsf{mpk} = \mathsf{prFE.mpk}$ and does the following.

- Run $(\mathsf{Lobf}_{\mathsf{off}}, \mathsf{Lst}) \leftarrow \mathsf{LObfOff}(1^\lambda, 1^S)$, where S is the maximum size of the circuits $E_{\mathsf{pub}}[R_0]$ and $E_{\mathsf{cir}}[R_0]$ defined in Figure 5 and Figure 6, respectively.
- Output $\mathsf{st} := (\mathsf{Lst}, \mathsf{prFE.mpk})$ and $\mathsf{ct}_{\mathsf{off}} := \mathsf{Lobf}_{\mathsf{off}}$.

$\mathsf{EncOn}(\mathsf{st}, \mathbf{x} = (\mathbf{x}_{\mathsf{pub}}, \mathbf{x}_{\mathsf{priv}}))$. It does the following.

- Parse the input as $\mathsf{st} \rightarrow (\mathsf{Lst}, \mathsf{prFE.mpk})$.
- Sample $\mathsf{sd}_0, \mathsf{sd}_1, \mathsf{sd}_2 \leftarrow \{0,1\}^\lambda$.
- Run $\mathsf{dig}_{\mathbf{x}_{\mathsf{pub}}} \leftarrow \mathsf{LDigest}(1^\lambda, \{(i, \mathbf{x}_{\mathsf{pub},i})\}_{i \in [L_{\mathsf{pub}}]})$, where $\mathbf{x}_{\mathsf{pub},i} \in \{0,1\}$ is the i-th bit of $\mathbf{x}_{\mathsf{pub}}$.
- Run $\mathsf{prFE.ct} \leftarrow \mathsf{prFE.Enc}(\mathsf{prFE.mpk}, (\mathsf{dig}_{\mathbf{x}_{\mathsf{pub}}}, \mathbf{x}_{\mathsf{priv}}, \mathsf{Lst}, \mathsf{sd}_0, \mathsf{sd}_1, \mathsf{sd}_2))$.
- Output $\mathsf{ct}_{\mathsf{on}} := \mathsf{prFE.ct}$.

The final output of $\mathsf{Enc}(\mathsf{mpk}, \mathbf{x})$ is $\mathsf{ct} := (\mathsf{Lobf}_{\mathsf{off}}, \mathsf{prFE.ct})$.

$\mathsf{Dec}(\mathsf{mpk}, \mathbf{x}_{\mathsf{pub}}, \mathsf{sk}_C, C, \mathsf{ct})$. It parses the input as $\mathsf{mpk} = \mathsf{prFE.mpk}$, $\mathsf{ct} := (\mathsf{Lobf}_{\mathsf{off}}, \mathsf{prFE.ct})$, $\mathsf{sk}_C = (\mathbf{r}, \mathsf{prFE.sk}_0, \mathsf{prFE.sk}_1, \mathsf{prFE.sk}_2)$ and does the following.

- Compute $\mathsf{prFE.Dec}(\mathsf{prFE.mpk}, \mathsf{prFE.sk}_0, F_0[\mathbf{r}], \mathsf{prFE.ct})$ and parse the output as $\{\mathsf{lab}'_i\}_{i \in [L_{\mathsf{pub}}+1, L_{\mathsf{pub}}+L_{\mathsf{priv}}]}$.
- Compute $\mathsf{Lobf}_{\mathsf{on},1} = \mathsf{prFE.Dec}(\mathsf{prFE.mpk}, \mathsf{prFE.sk}_1, F_1[\mathbf{r}], \mathsf{prFE.ct})$.
- Compute $\mathsf{Lobf}_{\mathsf{on},2} = \mathsf{prFE.Dec}(\mathsf{prFE.mpk}, \mathsf{prFE.sk}_1, F_2[\mathbf{r}, \mathsf{dig}_C], \mathsf{prFE.ct})$.
- Set $\mathsf{Lobf}_1 := (\mathsf{Lobf}_{\mathsf{off}}, \mathsf{Lobf}_{\mathsf{on},1})$ and compute $\{\widetilde{\mathsf{lab}}_i\}_{i \in [L_{\mathsf{pub}}]} = \mathsf{LEval}(\{(i, \mathbf{x}_{\mathsf{pub},i})\}_{i \in [L_{\mathsf{pub}}]}, \mathsf{Lobf}_1)$.
- Set $\mathsf{Lobf}_2 := (\mathsf{Lobf}_{\mathsf{off}}, \mathsf{Lobf}_{\mathsf{on},2})$ and compute $\tilde{C} = \{\tilde{C}_i\}_{i \in [|C|]} = \mathsf{LEval}(\{(i, C_i)\}_{i \in |C|}, \mathsf{Lobf}_2)$.

– Set $\mathsf{lab}_i := \begin{cases} \widetilde{\mathsf{lab}}_i & \text{if } i \in [L_{\mathsf{pub}}] \\ \mathsf{lab}'_i & \text{if } i \in [L_{\mathsf{pub}}+1, L_{\mathsf{pub}}+L_{\mathsf{priv}}] \end{cases}$ for $i \in [L_{\mathsf{pub}}+L_{\mathsf{priv}}]$.

– Compute $z = \mathsf{Eval}(\tilde{C}, \{\mathsf{lab}_i\}_{i\in[L_{\mathsf{pub}}+L_{\mathsf{priv}}]})$.

– Output z.

Circuit $F_0[\mathbf{r}]$:

- Parse the input as $(\mathsf{dig}, \mathbf{x}_{\mathsf{priv}}, \mathsf{Lst}, \mathsf{sd}_0, \mathsf{sd}_1, \mathsf{sd}_2)$.
- Compute $R_0 := \mathsf{PRF}(\mathsf{sd}_0, \mathbf{r})$.
- Compute $(\mathsf{lab}_{i,0}, \mathsf{lab}_{i,1}) \leftarrow \mathsf{Garble}_{\mathsf{inp},i}(1^\lambda; R_0)$ for $i \in [L_{\mathsf{pub}}+1, L_{\mathsf{pub}}+L_{\mathsf{priv}}]$.
- Output $\{\mathsf{lab}_{i,\mathbf{x}_{\mathsf{priv},i-L_{\mathsf{pub}}}}\}_{i\in[L_{\mathsf{pub}}+1,L_{\mathsf{pub}}+L_{\mathsf{priv}}]}$.

Figure 4: Circuit to compute garbled labels corresponding to $\mathbf{x}_{\mathsf{priv}}$.

Circuit $F_1[\mathbf{r}]$:

- Parse the input as $(\mathsf{dig}, \mathbf{x}_{\mathsf{priv}}, \mathsf{Lst}, \mathsf{sd}_0, \mathsf{sd}_1, \mathsf{sd}_2)$.
- Compute $R_0 := \mathsf{PRF}(\mathsf{sd}_0, \mathbf{r})$ and $R_1 := \mathsf{PRF}(\mathsf{sd}_1, \mathbf{r})$.
- Compute $\mathsf{Lobf}_{\mathsf{on},1} \leftarrow \mathsf{LObfOn}(\mathsf{Lst}, \mathsf{dig}, E_{\mathsf{pub}}[R_0]; R_1)$, where $E_{\mathsf{pub}}[R_0]$ is a circuit that works as follows.
  - Take $(i,b) \in [L_{\mathsf{pub}}] \times \{0,1\}$ as an input.
  - Compute $(\mathsf{lab}_{i,0}, \mathsf{lab}_{i,1}) \leftarrow \mathsf{Garble}_{\mathsf{inp},i}(1^\lambda; R_0)$.
  - Output $\mathsf{lab}_{i,b}$.
- Output $\mathsf{Lobf}_{\mathsf{on},1}$.

Figure 5: $F_1[\mathbf{r}]$ computes an obfuscation of a circuit computing bGC labels for $\mathbf{x}_{\mathsf{pub}}$.

Correctness. We make the following observations.

– From the perfect correctness of prFE scheme and definition of $F_0[\mathbf{r}], F_1[\mathbf{r}], F_2[\mathbf{r}, \mathsf{dig}_C]$, we get

$$\begin{aligned}\{\mathsf{lab}_{i,\mathbf{x}_{\mathsf{priv},i-L_{\mathsf{pub}}}}\}_{i\in[L_{\mathsf{pub}}+1,L_{\mathsf{pub}}+L_{\mathsf{priv}}]} &= \mathsf{prFE.Dec}(\mathsf{prFE.mpk}, \mathsf{prFE.sk}_0, F_0[\mathbf{r}], \mathsf{prFE.ct}) \\ &= F_0[\mathbf{r}](\mathsf{dig}_{\mathbf{x}_{\mathsf{pub}}}, \mathbf{x}_{\mathsf{priv}}, \mathsf{Lst}, \mathsf{sd}_0, \mathsf{sd}_1, \mathsf{sd}_2) \\ &= \{\mathsf{Garble}_{\mathsf{inp},i}(1^\lambda; R_0)\}_{i\in[L_{\mathsf{pub}}+1,L_{\mathsf{pub}}+L_{\mathsf{priv}}]}\end{aligned}$$

$$\begin{aligned}\mathsf{Lobf}_{\mathsf{on},1} &= \mathsf{prFE.Dec}(\mathsf{prFE.mpk}, \mathsf{prFE.sk}_1, F_1[\mathbf{r}], \mathsf{prFE.ct}) \\ &= F_1[\mathbf{r}](\mathsf{dig}_{\mathbf{x}_{\mathsf{pub}}}, \mathbf{x}_{\mathsf{priv}}, \mathsf{Lst}, \mathsf{sd}_0, \mathsf{sd}_1, \mathsf{sd}_2) \\ &= \mathsf{LObfOn}(\mathsf{Lst}, \mathsf{dig}_{\mathbf{x}_{\mathsf{pub}}}, E_{\mathsf{pub}}[R_0]; R_1),\end{aligned}$$

$$\begin{aligned}\mathsf{Lobf}_{\mathsf{on},2} &= \mathsf{prFE.Dec}(\mathsf{prFE.mpk}, \mathsf{prFE.sk}_2, F_2[\mathbf{r}, \mathsf{dig}_C], \mathsf{prFE.ct}) \\ &= F_2[\mathbf{r}, \mathsf{dig}_C](\mathsf{dig}_{\mathbf{x}_{\mathsf{pub}}}, \mathbf{x}_{\mathsf{priv}}, \mathsf{Lst}, \mathsf{sd}_0, \mathsf{sd}_1, \mathsf{sd}_2) \\ &= \mathsf{LObfOn}(\mathsf{Lst}, \mathsf{dig}_C, E_{\mathsf{cir}}[R_0]; R_2),\end{aligned}$$

Circuit $F_2[\mathbf{r}, \mathsf{dig}_C]$:

- Parse the input as $(\mathsf{dig}, \mathbf{x}_{\mathsf{priv}}, \mathsf{Lst}, \mathsf{sd}_0, \mathsf{sd}_1, \mathsf{sd}_2)$.
- Compute $R_0 := \mathsf{PRF}(\mathsf{sd}_0, \mathbf{r})$ and $R_2 := \mathsf{PRF}(\mathsf{sd}_2, \mathbf{r})$.
- Compute $\mathsf{Lobf}_{\mathsf{on},2} \leftarrow \mathsf{LObfOn}(\mathsf{Lst}, \mathsf{dig}_C, E_{\mathsf{cir}}[R_0]; R_2)$, where $E_{\mathsf{cir}}[R_0]$ is a circuit that works as follows.
  - Take $(i, G) \in [|C|] \times \{0,1\}^{4\lambda}$ as an input, where G encodes the information of a gate.
  - Compute $\tilde{G} \leftarrow \mathsf{Garble}_i(1^\lambda, G; R_0)$.
  - Output $\tilde{G}$.
- Output $\mathsf{Lobf}_{\mathsf{on},2}$.

Figure 6: Circuit $F_2[\mathbf{r}, \mathsf{dig}_C]$ computes an obfuscation of a circuit computing bGC labels for C.

where $R_b := \mathsf{PRF}(\mathsf{sd}_b, \mathbf{r})$ for $b \in \{0,1,2\}$ and $E_{\mathsf{pub}}[R_0]$ and $E_{\mathsf{cir}}[R_0]$ are the circuits as defined in Figure 5 and Figure 6, respectively.

– Next, parsing $\mathsf{Lobf}_1 := (\mathsf{Lobf}_{\mathsf{off}}, \mathsf{Lobf}_{\mathsf{on},1})$, we get

$$\begin{aligned}\{\widetilde{\mathsf{lab}}_i\}_{i\in[L_{\mathsf{pub}}]} &= \mathsf{LEval}(\{(i, \mathbf{x}_{\mathsf{pub},i})\}_{i\in[L_{\mathsf{pub}}]}, \mathsf{Lobf}_1)\\ &= \{E_{\mathsf{pub}}[R_0](i, \mathbf{x}_{\mathsf{pub},i})\}_{i\in[L_{\mathsf{pub}}]}\\ &= \mathsf{lab}_{i,\mathbf{x}_{\mathsf{pub},i}}\end{aligned}$$

where $(\mathsf{lab}_{i,0}, \mathsf{lab}_{i,1}) \leftarrow \mathsf{Garble}_{\mathsf{inp},i}(1^\lambda; R_0)$ and

$$\begin{aligned}\tilde{C} &= \{\tilde{C}_i\}_{i\in[|C|]}\\ &= \mathsf{LEval}(\{(i, C_i)\}_{i\in|C|}, \mathsf{Lobf}_2)\\ &= \{E_{\mathsf{cir}}[R_0](i, C_i)\}_{i\in|C|}\\ &= \{\mathsf{Garble}_i(1^\lambda, C_i; R_0)\}_{i\in|C|}\end{aligned}$$

from the perfect correctness of the laconic pPRIO scheme and the definition of $E_{\mathsf{pub}}[R_0]$ and $E_{\mathsf{cir}}[R_0]$.

– Next, from the perfect correctness of the garbling scheme bGC, we have, for $\mathsf{lab}_i = (\mathsf{lab}_{\mathbf{x}_{\mathsf{pub}}}, \mathsf{lab}_{\mathbf{x}_{\mathsf{priv}}})$,

$$\mathsf{Eval}(\tilde{C}, \{\mathsf{lab}_i\}_{i\in[L_{\mathsf{pub}}+L_{\mathsf{priv}}]}) = C(\mathbf{x}_{\mathsf{pub}}, \mathbf{x}_{\mathsf{priv}})$$

and hence the perfect correctness of the PHprFE scheme.

Efficiency Here, we show

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

To do so, we first show that the sizes of the circuits $F_0[\mathbf{r}]$, $F_1[\mathbf{r}]$, and $F_2[\mathbf{r}, \mathsf{dig}_C]$ are all bounded by $\mathrm{poly}(\lambda, L_{\mathsf{priv}})$.

- In $F_0[\mathbf{r}]$, the computation of the PRF and $\mathsf{Garble}_{\mathsf{inp},i}$ are performed, where both of them can be implemented by circuits of size $\mathrm{poly}(\lambda)$. Since the latter is repeated for L_{priv} times, the overall size of $F_0[\mathbf{r}]$ is $\mathrm{poly}(\lambda, L_{\mathsf{priv}})$.

- To bound the size of $F_1[\mathbf{r}]$, we first bound the size of $E_{\mathsf{pub}}[R_0]$. $E_{\mathsf{pub}}[R_0]$ performs the computation of $\mathsf{Garble}_{\mathsf{inp},i}$ on input $i \in [L_{\mathsf{pub}}]$ (in binary), which can be implemented by a circuit of size $\mathrm{poly}(\log L_{\mathsf{pub}}, \lambda)$. This can be further bounded by $\mathrm{poly}(\lambda)$ since $L_{\mathsf{pub}} \leq 2^\lambda$. We then observe that $F_1[\mathbf{r}]$ consists of the evaluation of two PRF values and LObfOn on input Lst, dig, and $E_{\mathsf{pub}}[R_0]$. We can bound the input length to LObfOn by a fixed polynomial, since we have $|\mathsf{Lst}| = \lambda$ and $|\mathsf{dig}| = \mathrm{poly}(\lambda)$ and LObfOn runs in polynomial time in the input length, its size is bounded by $\mathrm{poly}(\lambda)$. Therefore, the overall size of $F_1[\mathbf{r}]$ is $\mathrm{poly}(\lambda, L_{\mathsf{priv}})$, where we take into account the input $\mathbf{x}_{\mathsf{priv}}$, which is ignored in the computation.
- To bound the size of $F_2[\mathbf{r}, \mathsf{dig}_C]$, we first bound the size of $E_{\mathsf{cir}}[R_0]$. $E_{\mathsf{cir}}[R_0]$ performs the computation of Garble_i on input $i \in [|C|]$ (in binary) and $G \in \{0,1\}^{4\lambda}$, which can be implemented by a circuit of size $\mathrm{poly}(\log |C|, \lambda)$. This can be further bounded by $\mathrm{poly}(\lambda)$, since $|C| \leq 2^\lambda$. Similarly to the case of $F_1[\mathbf{r}]$, the size of $F_2[\mathbf{r}]$ can be bounded by $\mathrm{poly}(\lambda, L_{\mathsf{priv}})$.

We then move to discuss the size of the parameters.

- We can bound $|\mathsf{mpk}| = |\mathsf{prFE.mpk}|$ by $\mathrm{poly}(\lambda, \mathsf{inp}, \mathsf{dep}, \mathsf{out})$, since it is output by $\mathsf{prFE.Setup}(1^\lambda, \mathsf{prm} = (1^{\mathsf{inp}}, 1^{\mathsf{dep}}, 1^{\mathsf{out}}))$. We have $\mathsf{inp}(\lambda) \leq L_{\mathsf{priv}} + L_{\mathsf{LDigest}}(\lambda) + 4\lambda \leq \mathrm{poly}(\lambda, L_{\mathsf{priv}})$ and $\mathsf{dep}(\lambda)$ and $\mathsf{out}(\lambda)$ are bounded by the maximum size of the circuits E_{cir} and E_{pub}, which in turn is bounded by $\mathrm{poly}(\lambda, L_{\mathsf{priv}})$. We therefore have $|\mathsf{mpk}| = \mathrm{poly}(\lambda, L_{\mathsf{priv}})$.
- We have $|\mathsf{sk}_C| = \lambda + |\mathsf{prFE.sk}_0| + |\mathsf{prFE.sk}_1| + |\mathsf{prFE.sk}_2|$. We can bound the size of $|\mathsf{prFE.sk}_0|$ by $\mathrm{poly}(\lambda, L_{\mathsf{priv}})$, since it is generated by $\mathsf{prFE.KeyGen}(\mathsf{prFE.msk}, F_0[\mathbf{r}])$, where the input length to prFE.KeyGen is bounded by $\mathrm{poly}(\lambda, L_{\mathsf{priv}})$. Similarly, we can bound $|\mathsf{prFE.sk}_1|$ and $|\mathsf{prFE.sk}_2|$ by $\mathrm{poly}(\lambda, L_{\mathsf{priv}})$. Therefore, we have $|\mathsf{sk}_C| = \mathrm{poly}(\lambda, L_{\mathsf{priv}})$.
- We have $|\mathsf{ct}| = |\mathsf{Lobf}_{\mathsf{off}}| + |\mathsf{prFE.ct}|$. We first bound $|\mathsf{Lobf}_{\mathsf{off}}|$. From the above discussion, we have $S = \max\{|E_{\mathsf{pub}}[R_0]|, |E_{\mathsf{cir}}[R_0]|\} = \mathrm{poly}(\lambda, L_{\mathsf{priv}})$. This implies $|\mathsf{Lobf}_{\mathsf{off}}| = \mathrm{poly}(\lambda, L_{\mathsf{priv}})$, since $\mathsf{Lobf}_{\mathsf{off}}$ is output by $\mathsf{LObfOff}(1^\lambda, 1^S)$. We then bound $|\mathsf{prFE.ct}|$, which is output by $\mathsf{prFE.ct} \leftarrow \mathsf{prFE.Enc}(\mathsf{prFE.mpk}, (\mathsf{dig}_{\mathbf{x}_{\mathsf{pub}}}, \mathbf{x}_{\mathsf{priv}}, \mathsf{Lst}, \mathsf{sd}_0, \mathsf{sd}_1, \mathsf{sd}_2))$. Since the input length of prFE.Enc is $\mathsf{inp}(\lambda) = L_{\mathsf{priv}} + L_{\mathsf{LDigest}}(\lambda) + 4\lambda = \mathrm{poly}(\lambda, L_{\mathsf{priv}})$. We therefore have $|\mathsf{ct}| = |\mathsf{Lobf}_{\mathsf{off}}| + |\mathsf{prFE.ct}| = \mathrm{poly}(\lambda, L_{\mathsf{priv}})$.

5.3 Security

Before proving the security of our scheme, 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 laconic pPRIO security. A conceptually similar lemma is proven in Lemma 3.4 of [ARYY23] in the context of evasive LWE.

Lemma 5.8. Let $\mathsf{LprIO} = (\mathsf{LDigest}, \mathsf{LObfuscate}, \mathsf{LEval})$ be a laconic pPRIO scheme and Samp be a PPT algorithm that takes as input 1^λ and outputs

$$\left(\mathsf{aux} = (\mathsf{aux}_1, \mathsf{aux}_2) \in \{0,1\}^* \times \mathcal{X}, 1^S, X^1 = \{X_i^1\}_{i \in [N^1]}, \ldots, X^Q = \{X_i^Q\}_{i \in [N^Q]}, E^1, \ldots, E^Q\right)$$

for some set $\mathcal{X}$. Here $X_i^k \in \{0,1\}^{\ell^k}$, $E^k : \{0,1\}^{\ell^k} \to \{0,1\}^{L^k}$ and $|E^k| \leq S$ for $k \in [Q], i \in [N^k]$.
Let us assume that

$$\begin{aligned}&\left((\mathsf{aux}_1, \mathsf{aux}_2), 1^S, X^1, \ldots, X^Q, \{E^1(X_i^1)\}_{i \in [N^1]}, \ldots, \{E^Q(X_i^Q)\}_{i \in [N^Q]}, \right)\\ \approx_c &\left((\mathsf{aux}_1, \mathbf{x}), 1^S, X^1, \ldots, X^Q, \{\Delta_i^1\}_{i \in [N^1]}, \ldots, \{\Delta_i^Q\}_{i \in [N^Q]}\right)\end{aligned}$$

holds for $\mathbf{x} \leftarrow \mathcal{X}, \Delta_i^k \leftarrow \{0,1\}^{L^k}$ for $k \in [Q], i \in [N^k]$ and also assume the security of LprIO with respect to Samp. We then have

$$\left((\mathsf{aux}_1, \mathsf{aux}_2), X^1, \ldots, X^Q, \mathsf{Lobf}_{\mathsf{off}}, \mathsf{Lobf}_{\mathsf{on}}^1, \ldots, \mathsf{Lobf}_{\mathsf{on}}^Q\right) \approx_c \left((\mathsf{aux}_1, \mathbf{x}), X^1, \ldots, X^Q, \mathsf{Lobf}_{\mathsf{off}}, \delta^1 \ldots, \delta^Q\right),$$

where $(\mathsf{Lobf}_{\mathsf{off}}, \mathsf{st}) \leftarrow \mathsf{LObfOff}(1^\lambda, 1^S)$, $\mathsf{dig}^k \leftarrow \mathsf{LDigest}(1^\lambda, X^k)$, $\mathsf{Lobf}^k_{\mathsf{on}} \leftarrow \mathsf{LObfOn}(\mathsf{dig}^k, E^k), \delta^k \leftarrow \mathcal{O}_{\mathsf{on}}$ for $k \in [Q]$.

Proof. From the assumption, we have

$$\left((\mathsf{aux}_1, \mathsf{aux}_2), 1^S, X^1, \ldots, X^Q, \{E^j(X^j_i)\}_{i\in[N^j], j\in[Q]}\right) \approx_c \left((\mathsf{aux}_1, \mathbf{x}), 1^S, X^1, \ldots, X^Q, \{\Delta^j_i\}_{i\in[N^j], j\in[Q]}\right) \quad (22)$$

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

$$(\mathsf{aux}_1, \mathsf{aux}_2, 1^S, X^1, \ldots, X^Q, \{\Delta^j_i\}_{i\in[N^j], j\in[Q]}) \approx_c (\mathsf{aux}_1, \mathbf{x}, 1^S, X^1, \ldots, X^Q, \{\Delta^j_i\}_{i\in[N^j], j\in[Q]}) \quad (23)$$

since adding independently sampled random terms Δ^j_i does not make the task of distinguishing the distributions easier. Equation (22) and Equation (23) implies

$$(\mathsf{aux}_1, \mathsf{aux}_2, 1^S, X^1, \ldots, X^Q, \{E^j(X^j_i)\}_{i\in[N^j], j\in[Q]}) \approx_c (\mathsf{aux}_1, \mathsf{aux}_2, 1^S, X^1, \ldots, X^Q, \{\Delta^j_i\}_{i\in[N^j], j\in[Q]}).$$

Applying LprIO security definition with respect to Samp, we get

$$\left(\mathsf{aux}_1, \mathsf{aux}_2, X^1, \ldots, X^Q, \mathsf{Lobf}_{\mathsf{off}}, \mathsf{Lobf}^1_{\mathsf{on}}, \ldots, \mathsf{Lobf}^Q_{\mathsf{on}}\right) \approx_c \left(\mathsf{aux}_1, \mathsf{aux}_2, X^1, \ldots, X^Q, \mathsf{Lobf}_{\mathsf{off}}, \delta^1 \ldots, \delta^Q\right). \quad (24)$$

Next, from $(\mathsf{aux}_1, \mathsf{aux}_2, 1^S, X^1, \ldots, X^Q) \approx_c (\mathsf{aux}_1, \mathbf{x}, 1^S, X^1, \ldots, X^Q)$ we have

$$\left(\mathsf{aux}_1, \mathsf{aux}_2, X^1, \ldots, X^Q, \mathsf{Lobf}_{\mathsf{off}}, \delta^1 \ldots, \delta^Q\right) \approx_c \left(\mathsf{aux}_1, \mathbf{x}, X^1, \ldots, X^Q, \mathsf{Lobf}_{\mathsf{off}}, \delta^1 \ldots, \delta^Q\right), \quad (25)$$

since $\mathsf{Lobf}_{\mathsf{off}}$ can be sampled using 1^S and $\{\delta^j\}_{j\in[Q]}$ can be sampled independently.

From Equation (24) and Equation (25) we deduce

$$\left(\mathsf{aux}_1, \mathsf{aux}_2, X^1, \ldots, X^Q, \mathsf{Lobf}_{\mathsf{off}}, \mathsf{Lobf}^1_{\mathsf{on}}, \ldots, \mathsf{Lobf}^Q_{\mathsf{on}}\right) \approx_c \left(\mathsf{aux}_1, \mathbf{x}, X^1, \ldots, X^Q, \mathsf{Lobf}_{\mathsf{off}}, \delta^1 \ldots, \delta^Q\right).$$

hence the lemma. □

The following theorem asserts the security of our construction of PHprFE scheme. This in particular implies that the construction satisfies the security notion as per Definition 5.2.

Theorem 5.9. The above construction satisfies reusable security as per Definition 5.4.

Proof. Consider a sampler Samp that generates the following:

1. **Key Queries.** It issues Q_{key} key queries $C^1, \ldots, C^{Q_{\mathsf{key}}}$.
2. **Ciphertext Queries.** It issues messages $\mathbf{x}^1 = (\mathbf{x}^1_{\mathsf{pub}}, \mathbf{x}^1_{\mathsf{priv}}), \ldots, \mathbf{x}^{Q_{\mathsf{msg}}} = (\mathbf{x}^{Q_{\mathsf{msg}}}_{\mathsf{pub}}, \mathbf{x}^{Q_{\mathsf{msg}}}_{\mathsf{priv}})$.
3. **Auxiliary Information.** It outputs the auxiliary information aux.

To prove the security as per Definition 5.4, we prove

$$\begin{pmatrix} \mathsf{mpk} = \mathsf{prFE.mpk},\ \mathsf{aux},\ \left\{C^k\right\}_{k\in[Q_{\mathsf{key}}]}, \\ \left\{\mathsf{sk}^k := (\mathbf{r}^k, \mathsf{prFE.sk}^k_0, \mathsf{prFE.sk}^k_1, \mathsf{prFE.sk}^k_2)\right\}_{k\in[Q_{\mathsf{key}}]}, \\ \mathsf{Lobf}_{\mathsf{off}},\ \left\{\ \mathbf{x}^j_{\mathsf{pub}},\ \mathsf{prFE.ct}^j\ \right\}_{j\in[Q_{\mathsf{msg}}]} \end{pmatrix} \approx_c \begin{pmatrix} \mathsf{mpk} = \mathsf{prFE.mpk},\ \mathsf{aux},\ \left\{C^k\right\}_{k\in[Q_{\mathsf{key}}]}, \\ \left\{\mathsf{sk}^k := (\mathbf{r}^k, \mathsf{prFE.sk}^k_0, \mathsf{prFE.sk}^k_1, \mathsf{prFE.sk}^k_2)\right\}_{k\in[Q_{\mathsf{key}}]}, \\ \mathsf{Lobf}_{\mathsf{off}},\ \left\{\ \mathbf{x}^j_{\mathsf{pub}},\ \delta^j \leftarrow \mathcal{CT}_{\mathsf{prFE}}\ \right\}_{j\in[Q_{\mathsf{msg}}]} \end{pmatrix} \quad (26)$$

assuming we have

$$(1^\lambda,\ \mathsf{aux},\ \{\mathbf{x}^j_{\mathsf{pub}},\ C^k,\ C^k(\mathbf{x}^j)\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]}) \approx_c (1^\lambda,\ \mathsf{aux},\ \{\mathbf{x}^j_{\mathsf{pub}},\ C^k,\ \Delta^{j,k} \leftarrow \{0,1\}\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]}) \quad (27)$$

where

$$\begin{aligned}
&\left(\{C^k\}_{k\in[Q_{\mathsf{key}}]}, \{\mathbf{x}^j = (\mathbf{x}^j_{\mathsf{pub}}, \mathbf{x}^j_{\mathsf{priv}})\}_{j\in[Q_{\mathsf{msg}}]}, \mathsf{aux} \in \{0,1\}^*\right) \leftarrow \mathsf{Samp}(1^\lambda),\\
&(\mathsf{prFE.mpk}, \mathsf{prFE.msk}) \leftarrow \mathsf{prFE.Setup}(1^\lambda, \mathsf{prm}),\\
&\mathbf{r}^k \leftarrow \{0,1\}^\lambda,\ \mathsf{prFE.sk}^k_0 \leftarrow \mathsf{prFE.KeyGen}(\mathsf{prFE.msk}, F_0[\mathbf{r}^k]),\ \text{for } F_0[\mathbf{r}^k] \text{ as defined in Figure 4},\\
&\mathsf{prFE.sk}^k_1 \leftarrow \mathsf{prFE.KeyGen}(\mathsf{prFE.msk}, F_1[\mathbf{r}^k]),\ \text{for} F_1[\mathbf{r}^k, \mathsf{dig}_{C^k}] \text{ as defined in Figure 5}\\
&\mathsf{dig}_{C^k} \leftarrow \mathsf{LDigest}(\{(i, C^k_i)\}_{i\in|C^k|})\\
&\mathsf{prFE.sk}^k_2 \leftarrow \mathsf{prFE.KeyGen}(\mathsf{prFE.msk}, F_2[\mathbf{r}^k, \mathsf{dig}_{C^k}]),\ \text{for } F_2[\mathbf{r}^k, \mathsf{dig}_{C^k}] \text{ as defined in Figure 6},\\
&(\mathsf{Lobf}_{\mathsf{off}}, \mathsf{Lst}) \leftarrow \mathsf{LObfOff}(1^\lambda, 1^S),\\
&\mathsf{dig}_{\mathbf{x}^j_{\mathsf{pub}}} \leftarrow \mathsf{LDigest}(1^\lambda, \{(i, \mathbf{x}^j_{\mathsf{pub},i})\}_{i\in[L_{\mathsf{pub}}]}), \mathsf{sd}^j_0, \mathsf{sd}^j_1, \mathsf{sd}^j_2 \leftarrow \{0,1\}^\lambda,\\
&\mathsf{prFE.ct}^j \leftarrow \mathsf{prFE.Enc}(\mathsf{prFE.mpk}, (\mathsf{dig}_{\mathbf{x}^j_{\mathsf{pub}}}, \mathbf{x}^j_{\mathsf{priv}}, \mathsf{Lst}, \mathsf{sd}^j_0, \mathsf{sd}^j_1, \mathsf{sd}^j_2)) \qquad \text{for } j \in [Q_{\mathsf{msg}}].
\end{aligned}$$

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

$$\begin{pmatrix}
\text{Functions:} & \left\{F_0[\mathbf{r}^k],\ F_1[\mathbf{r}^k],\ F_2[\mathbf{r}^k, \mathsf{dig}_{C^k}]\right\}_{k\in[Q_{\mathsf{key}}]},\\
\text{Inputs:} & \left\{\mathbf{x}^j_{\mathsf{prFE}} := \left(\mathsf{dig}_{\mathbf{x}^j_{\mathsf{pub}}}, \mathbf{x}^j_{\mathsf{priv}}, \mathsf{Lst}, \mathsf{sd}^j_0, \mathsf{sd}^j_1, \mathsf{sd}^j_2\right)\right\}_{j\in[Q_{\mathsf{msg}}]},\\
\text{Auxiliary Information:} & \mathsf{aux}_{\mathsf{prFE}} := \left(\mathsf{aux},\ \left\{\mathbf{x}^j_{\mathsf{pub}}\right\}_{j\in[Q_{\mathsf{msg}}]},\ \mathsf{Lobf}_{\mathsf{off}},\ \{C^k,\ \mathbf{r}^k,\}_{k\in[Q_{\mathsf{key}}]}\right)
\end{pmatrix}$$

By the security guarantee of prFE with sampler $\mathsf{Samp}_{\mathsf{prFE}}$, Equation (26) holds if

$$\begin{pmatrix}
\left\{\begin{matrix}
F_0[\mathbf{r}^k](\mathbf{x}^j_{\mathsf{prFE}}) = \left\{\mathsf{lab}^{j,k}_{i,\mathbf{x}^j_{\mathsf{priv},i-L_{\mathsf{pub}}}}\right\}_{i\in[L_{\mathsf{pub}}+1, L_{\mathsf{pub}}+L_{\mathsf{priv}}]}\\
F_1[\mathbf{r}^k](\mathbf{x}^j_{\mathsf{prFE}}) = \mathsf{Lobf}^{j,k}_{\mathsf{on},1}\\
F_2[\mathbf{r}^k, \mathsf{dig}_{C^k}](\mathbf{x}^j_{\mathsf{prFE}}) = \mathsf{Lobf}^{j,k}_{\mathsf{on},2}
\end{matrix}\right\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]},\\
\mathsf{aux},\ \{\mathbf{x}^j_{\mathsf{pub}}\}_{j\in[Q_{\mathsf{msg}}]},\ \mathsf{Lobf}_{\mathsf{off}},\ \{C^k,\ \mathbf{r}^k\}_{k\in[Q]}
\end{pmatrix}
\approx_c
\begin{pmatrix}
\left\{\begin{matrix}
\left\{\delta^{j,k}_{0,i}\right\}_{i\in[L_{\mathsf{priv}}]}\\
\delta^{j,k}_1\\
\delta^{j,k}_2
\end{matrix}\right\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]},\\
\mathsf{aux},\ \{\mathbf{x}^j_{\mathsf{pub}}\}_{j\in[Q_{\mathsf{msg}}]},\ \mathsf{Lobf}_{\mathsf{off}},\ \{C^k,\ \mathbf{r}^k\}_{k\in[Q_{\mathsf{key}}]}
\end{pmatrix}, \quad (28)$$

where

$$\begin{aligned}
R^{j,k}_b &:= \mathsf{PRF}(\mathsf{sd}^j_b, \mathbf{r}^k) \quad \text{for} \quad b = 0,1,2,\\
(\mathsf{lab}^{j,k}_{i,0}, \mathsf{lab}^{j,k}_{i,1}) &= \mathsf{Garble}_{\mathsf{inp},i}(1^\lambda; R^{j,k}_0) \quad \text{for} \quad i \in [L_{\mathsf{pub}}+1, L_{\mathsf{pub}}+L_{\mathsf{priv}}],\\
(\mathsf{Lobf}_{\mathsf{off}}, \mathsf{Lst}) &\leftarrow \mathsf{LObfOff}(1^\lambda, 1^S)\\
\mathsf{Lobf}^{j,k}_{\mathsf{on},1} &= \mathsf{LObfOn}(\mathsf{Lst}, \mathsf{dig}_{\mathbf{x}^j_{\mathsf{pub}}}, E_{\mathsf{pub}}[R^{j,k}_0]; R^{j,k}_1)\\
\mathsf{Lobf}^{j,k}_{\mathsf{on},2} &= \mathsf{LObfOn}(\mathsf{Lst}, \mathsf{dig}_{C^k}, E_{\mathsf{cir}}[R^{j,k}_0]; R^{j,k}_2) \quad \text{for} \quad j \in [Q_{\mathsf{msg}}],\ k \in [Q_{\mathsf{key}}].
\end{aligned}$$

Therefore, it suffices to prove Equation (28). We observe that it suffices to prove a variant of Equation (28) where $R_0^{j,k}, R_1^{j,k}, R_2^{j,k}$ are replaced with independently chosen truly random strings, since (the original version of) Equation (28) then follows. This can be seen by the following indistinguishability:

$$\begin{aligned}
&\left\{R_b^{j,k} = \mathsf{PRF}(\mathsf{sd}_b^j, \mathbf{r}^k) : \mathsf{sd}_b^j \leftarrow \{0,1\}^\lambda, \mathbf{r}^k \leftarrow \{0,1\}^\lambda\right\}_{b\in\{0,1,2\}, j\in[Q_{\mathsf{msg}}], k\in[Q_{\mathsf{key}}]}\\
\approx_s &\left\{R_b^{j,k} = \mathsf{PRF}(\mathsf{sd}_b^j, \mathbf{r}^k) : \mathsf{sd}_b^j \leftarrow \{0,1\}^\lambda, \mathbf{r}^k \leftarrow \{0,1\}^\lambda \backslash \{\mathbf{r}^1, \dots, \mathbf{r}^{k-1}\}\right\}_{b\in\{0,1,2\}, j\in[Q_{\mathsf{msg}}], k\in[Q_{\mathsf{key}}]}\\
\approx_c &\left\{R_b^{j,k} \leftarrow \{0,1\}^\lambda\right\}_{b\in\{0,1,2\}, j\in[Q_{\mathsf{msg}}], k\in[Q_{\mathsf{key}}]},
\end{aligned}$$

where in the second distribution, $\{\mathbf{r}^k\}_{k\in[Q_{\mathsf{key}}]}$ is distributed uniformly at random over $(\{0,1\}^\lambda)^{Q_{\mathsf{key}}}$ with the constraint that there is no collision among them. We observe that the first indistinguishability holds since there is a colliding pair in $\{\mathbf{r}^k\}_{k\in[Q_{\mathsf{key}}]}$ with probability at most $Q_{\mathsf{key}}^2/2^\lambda$ in the first distribution and the second indistinguishability holds by the security of PRF, since each $R_b^{j,k}$ is generated by fresh pair of seed and input. We then invoke the security of LprIO with sampler $\mathsf{Samp}_{\mathsf{LprIO}}$ that outputs

$$\left(\begin{array}{cc}
\text{Inputs:} & \left\{ X^{j,k} := \{X_i^{j,k} := (i, C_i^k)\}_{i\in[|C^k|]},\ \bar{X}^{j,k} := \{\bar{X}_i^{j,k} := (i, \mathbf{x}_{\mathsf{pub},i}^j)\}_{i\in[L_{\mathsf{pub}}]} \right\}_{j\in[Q_{\mathsf{msg}}], k\in[Q_{\mathsf{key}}]}\\
\text{Functions:} & \left\{ E^{j,k} := E_{\mathsf{cir}}[R_0^{j,k}],\ \bar{E}^{j,k} := E_{\mathsf{pub}}[R_0^{j,k}] \right\}_{j\in[Q_{\mathsf{msg}}], k\in[Q_{\mathsf{key}}]},\\
\text{Auxiliary Information:} & \mathsf{aux}_{\mathsf{LprIO},1} := \left\{ \mathsf{lab}^{j,k}_{i, \mathbf{x}^j_{\mathsf{priv}, i-L_{\mathsf{pub}}}} \right\}_{i\in[L_{\mathsf{pub}}+1, L_{\mathsf{pub}}+L_{\mathsf{priv}}], j\in[Q_{\mathsf{msg}}], k\in[Q_{\mathsf{key}}]},\\
& \mathsf{aux}_{\mathsf{LprIO},2} := \left(\mathsf{aux},\ \{\mathbf{x}^j_{\mathsf{pub}}\}_{j\in[Q_{\mathsf{msg}}]},\ \{C^k,\ \mathbf{r}^k\}_{k\in[Q_{\mathsf{key}}]}\right)
\end{array}\right),$$

where we consider $2Q_{\mathsf{msg}}Q_{\mathsf{key}}$ inputs and functions. For simplifying the notations, we use two indices $j \in [Q_{\mathsf{msg}}]$ and $k \in [Q_{\mathsf{key}}]$ and consider barred and unbarred symbols to represent different variables instead of using the single index for inputs and functions. In the rest of the proof, we prove

$$\begin{aligned}
&\left(\left\{E^{j,k}(X_i^{j,k}) = \tilde{C}_i^{j,k}\right\}_{j\in[Q_{\mathsf{msg}}], k\in[Q_{\mathsf{key}}], i\in[|C^k|]}, \left\{\bar{E}^{j,k}(\bar{X}_i^{j,k}) = \mathsf{lab}^{j,k}_{i, \mathbf{x}^j_{\mathsf{pub},i}}\right\}_{j\in[Q_{\mathsf{msg}}], k\in[Q_{\mathsf{key}}], i\in[L_{\mathsf{pub}}]}, \mathsf{aux}_{\mathsf{LprIO},1}, \mathsf{aux}_{\mathsf{LprIO},2}\right)\\
&\approx_c \left(\left\{\gamma_i^{j,k}\right\}_{j\in[Q_{\mathsf{msg}}], k\in[Q_{\mathsf{key}}], i\in[|C^k|]},\ \left\{\bar{\gamma}_i^{j,k}\right\}_{j\in[Q_{\mathsf{msg}}], k\in[Q_{\mathsf{key}}], i\in[L_{\mathsf{pub}}]}, \alpha, \mathsf{aux}_{\mathsf{LprIO},2}\right) \qquad (29)
\end{aligned}$$

where $\gamma_i^{j,k} \leftarrow \mathcal{CT}^i_{\mathsf{bGC}}$ for $j \in [Q_{\mathsf{msg}}], k \in [Q_{\mathsf{key}}], i \in [|C^k|]$, $\bar{\gamma}_i^{j,k} \leftarrow \{0,1\}^\lambda$ for $j \in [Q_{\mathsf{msg}}], k \in [Q_{\mathsf{key}}], i \in [L_{\mathsf{pub}}]$, and $\alpha \leftarrow \{0,1\}^{\lambda L_{\mathsf{priv}} Q_{\mathsf{msg}} Q_{\mathsf{key}}}$. Note that the length of α is the same as that of $\mathsf{aux}_{\mathsf{LprIO}}$. Here $\mathcal{CT}^i_{\mathsf{bGC}}$ is the co-domain of Garble_i algorithm. This suffices to conclude the proof, since Equation (29) implies Equation (28) by the security of LprIO and Lemma 5.8.

To prove Equation (29), we introduce the following sequence of hybrids.

Hyb_1. This is the LHS distribution of Equation (29). By unrolling the definition of the functions and rearranging the terms, we can see that this is equivalent to the following distribution:

$$\left(\left\{\left\{\mathsf{lab}^{j,k}_{i,\mathbf{x}_i^j}\right\}_{i\in[\mathrm{L}]}, \tilde{C}^{j,k}\right\}_{j\in[Q_{\mathsf{msg}}], k\in[Q_{\mathsf{key}}]}, \mathsf{aux},\ \{\mathbf{x}^j_{\mathsf{pub}}\}_{j\in[Q_{\mathsf{msg}}]},\ \{C^k,\ \mathbf{r}^k\}_{k\in[Q_{\mathsf{key}}]}\right)$$

where $(\{\mathsf{lab}^{j,k}_{i,b}\}_{i\in[\mathrm{L}], b\in\{0,1\}}, \tilde{C}^{j,k}) \leftarrow \mathsf{Garble}(1^\lambda, C^k; R_0^{j,k})$, $\mathbf{r}^k \leftarrow \{0,1\}^\lambda$ for $j \in [Q_{\mathsf{msg}}]$ and $k \in [Q_{\mathsf{key}}]$, $(\mathsf{aux}, \{\mathbf{x}^j_{\mathsf{pub}}\}_j, \{C^k\}_k) \leftarrow \mathsf{Samp}(1^\lambda)$, and $\mathbf{x}_i^j$ is the i-th bit of $\mathbf{x}^j = (\mathbf{x}^j_{\mathsf{pub}}, \mathbf{x}^j_{\mathsf{priv}})$. Note that here, we merge the process of generating $\{\tilde{C}_i^{j,k}\}_i$, $\{\mathsf{lab}_{i,\mathbf{x}_{\mathsf{pub},i}}^{j,k}\}_i$, and $\{\mathsf{lab}_{i,\mathbf{x}_{\mathsf{priv},i}}^{j,k}\}_i$ into a single process of running $\mathsf{Garble}(1^\lambda, C^k; R_0^{j,k})$. This does not change the distribution due to the decomposability of bGC, since the former processes are run on input the common randomness $R_0^{j,k}$ for each j and k.

Hyb_2. This hybrid is same as the previous one except that we compute the labels and the garbled circuits using the simulation algorithm. Namely, the view of the adversary in this hybrid is

$$\left(\left\{ \{\mathsf{lab}_i^{j,k}\}_{i\in[\mathrm{L}]},\ \tilde{C}^{j,k} \right\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]},\ \mathsf{aux},\ \{\mathbf{x}_{\mathsf{pub}}^j\}_{j\in[Q_{\mathsf{msg}}]},\ \{C^k,\ \mathbf{r}^k\}_{k\in[Q_{\mathsf{key}}]} \right)$$

where we compute $(\tilde{C}^{j,k}, \{\mathsf{lab}_i^{j,k}\}_{i\in[\mathrm{L}]}) \leftarrow \mathsf{bGC.Sim}(1^\lambda, 1^{\mathrm{L}}, C^k(\mathbf{x}^j))$ for all $j \in [Q_{\mathsf{msg}}], k \in [Q_{\mathsf{key}}]$. Due to the simulation security of bGC, this hybrid is computationally indistinguishable from the previous one.

Hyb_3. This hybrid is same as the previous one except that we input random strings into the simulator of the blind garbled circuit. Namely, we compute $(\tilde{C}^{j,k}, \{\mathsf{lab}_i^{j,k}\}_{i\in[\mathrm{L}]})$ as $(\tilde{C}^{j,k}, \{\mathsf{lab}_i^{j,k}\}_{i\in[\mathrm{L}]}) \leftarrow \mathsf{bGC.Sim}(1^\lambda, 1^{\mathrm{L}}, \Delta^{j,k})$, where $\Delta^{j,k} \leftarrow \{0,1\}$ for all j and k. We can see that this hybrid is indistinguishable from the previous one by Equation (27).

Hyb_4. This hybrid is same as the previous one except that we replace the output of the simulator for bGC with random strings. By the blindness of bGC, this game is indistinguishable from the previous hybrid.

By rearranging the terms, we can see that the distribution in Hyb_4 is equivalent to that of the RHS of Equation (29). We therefore have that the LHS and RHS of Equation (29). This completes the proof of Theorem 4.7. $\square$

5.4 Reducing the Dependency on Private Input Length

The sizes of the master public key, ciphertexts, and the secret keys of our construction in Section 5.2 are all independent from the length of $\mathbf{x}_{\mathsf{pub}}$ and C. However, they still depend on the length of $\mathbf{x}_{\mathsf{priv}}$. Here, we show a simple conversion that removes this dependency from the sizes of the master public key and secret key. The size of the ciphertext inherently depends on the length of the private input since it should be hidden, but we can make this dependency minimal if we start from the scheme with the ciphertext size being independent of the length of the public input. In particular, the size of the ciphertext only additively depends on the length of $\mathbf{x}_{\mathsf{priv}}$. By applying the conversion in this section to our construction in Section 5.2, we obtain a construction of partial-hiding FE with the optimal parameter size.

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

1. A secret key encryption scheme $\mathsf{SKE} = (\mathsf{SKE.Setup}, \mathsf{SKE.Enc}, \mathsf{SKE.Dec})$ with the message space $\{0,1\}^{L_{\mathsf{priv}}}$ with pseudorandom ciphertext space as per Definition 2.1. We denote the ciphertext space of the scheme by $\mathcal{CT}_{\mathsf{SKE}}$ and the key space of the scheme by $\mathcal{K}_{\mathsf{SKE}}$. Without loss of generality, we assume $\mathcal{K}_{\mathsf{SKE}} = \{0,1\}^\lambda$. Furthermore, we assume that the ciphertext space of SKE is $\mathcal{CT}_{\mathsf{SKE}} = \{0,1\}^{L_{\mathsf{priv}}+\lambda}$. Such a construction can be obtained by using PRF for example.
2. A PHprFE scheme $\mathsf{PHprFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc} = (\mathsf{EncOff}, \mathsf{EncOn}), \mathsf{Dec})$ whose sizes of the master public key, ciphertext, and secret key are all $\mathrm{poly}(\lambda, L_{\mathsf{priv}})$. We can construct such a scheme assuming LWE and evasive LWE as is shown in Section 5. We denote the online part of the ciphertext space by $\mathcal{CT}_{\mathsf{PHprFE}} = \{0,1\}^{\ell^{\mathsf{cton}}_{\mathsf{PHprFE}}}$.

We describe the new scheme $\mathsf{PHprFE}' = (\mathsf{Setup}', \mathsf{KeyGen}', \mathsf{Enc}' = (\mathsf{EncOff}', \mathsf{EncOn}'), \mathsf{Dec}')$ for circuit class $C : \{0,1\}^{L_{\mathsf{pub}}} \times \{0,1\}^{L_{\mathsf{priv}}} \to \{0,1\}$ in the following.

$\mathsf{Setup}'(1^\lambda, 1^{L_{\mathsf{pub}}}, 1^{L_{\mathsf{priv}}})$. It does the following.

- Run $\mathsf{Setup}(1^\lambda, 1^{L_{\mathsf{pub}}+L_{\mathsf{priv}}+\lambda}, 1^\lambda) \to (\mathsf{mpk}, \mathsf{msk})$.
- Output the master public key mpk and the master secret key msk.

$\mathsf{KeyGen}'(\mathsf{msk}, C)$. It does the following.

- Define the circuit C' as follows.
  On input $(\mathsf{SKE.sk}, \mathbf{x}_{\mathsf{pub}}, \mathsf{SKE.ct})$, output
$$C'(\mathbf{x}_{\mathsf{pub}}, \mathsf{SKE.ct}, \mathsf{SKE.sk}) = C(\mathbf{x}_{\mathsf{pub}}, \mathsf{SKE.Dec}(\mathsf{SKE.sk}, \mathsf{SKE.ct})).$$
- It runs $\mathsf{KeyGen}(\mathsf{msk}, C') \to \mathsf{sk}_{C'}$ and outputs $\mathsf{sk}_{C'}$.

$\mathsf{Enc}'(\mathsf{mpk}, \mathbf{x} = (\mathbf{x}_{\mathsf{pub}}, \mathbf{x}_{\mathsf{priv}}))$. The encryption algorithm does the following. We divide the algorithm into the following two steps.

$\mathsf{EncOff}'(\mathsf{mpk})$. It runs $\mathsf{EncOff}(\mathsf{mpk}) \to (\mathsf{ct}_{\mathsf{off}}, \mathsf{st})$ and outputs the offline part of the ciphertext $\mathsf{ct}_{\mathsf{off}}$ and state st.

$\mathsf{EncOn}'(\mathsf{st}, \mathbf{x} = (\mathbf{x}_{\mathsf{pub}}, \mathbf{x}_{\mathsf{priv}}))$. It does the following.

- Run $\mathsf{SKE.Setup}(1^\lambda) \to \mathsf{SKE.sk}$.
- Run $\mathsf{SKE.Enc}(\mathsf{SKE.sk}, \mathbf{x}_{\mathsf{priv}}) \to \mathsf{SKE.ct}$.
- Set $\mathbf{y} := (\mathbf{x}_{\mathsf{pub}}, \mathsf{SKE.ct})$ and $\mathbf{z} := \mathsf{SKE.sk}$ and run $\mathsf{EncOn}(\mathsf{mpk}, (\mathbf{y}, \mathbf{z})) \to \mathsf{ct}_{\mathsf{on}}$.
- Output ${\mathsf{ct}_{\mathsf{on}}}' := (\mathsf{SKE.ct}, \mathsf{ct}_{\mathsf{on}})$.

The final output of $\mathsf{Enc}'(\mathsf{mpk}, \mathbf{x} = (\mathbf{x}_{\mathsf{pub}}, \mathbf{x}_{\mathsf{priv}}))$ is $\mathsf{ct}' := (\mathsf{ct}_{\mathsf{off}}, \mathsf{SKE.ct}, \mathsf{ct}_{\mathsf{on}})$.

$\mathsf{Dec}'(\mathsf{mpk}, \mathbf{x}_{\mathsf{pub}}, \mathsf{sk}_{C'}, C, \mathsf{ct}')$. It does the following.

- Parse $\mathsf{ct}' \to (\mathsf{ct}_{\mathsf{off}}, \mathsf{SKE.ct}, \mathsf{ct}_{\mathsf{on}})$ and set $\mathbf{y} := (\mathbf{x}_{\mathsf{pub}}, \mathsf{SKE.ct})$ and $\mathsf{ct} := (\mathsf{ct}_{\mathsf{off}}, \mathsf{ct}_{\mathsf{on}})$.
- Define C' as in the key generation algorithm.
- Run $\mathsf{Dec}(\mathsf{mpk}, \mathbf{y}, \mathsf{sk}_C, C', \mathsf{ct}) \to w$ and output w.

Correctness. We make the following observations.

- From the perfect correctness of underlying PHprFE scheme, with public input $\mathbf{y} = (\mathbf{x}_{\mathsf{pub}}, \mathsf{SKE.ct})$ and private inpute $\mathsf{SKE.sk}$, we have
$$\begin{aligned}\mathsf{Dec}(\mathsf{mpk}, \mathbf{y}, \mathsf{sk}_{C'}, C', \mathsf{ct}) &= C'(\mathbf{y}, \mathsf{SKE.sk}) \\ &= C'(\mathbf{x}_{\mathsf{pub}}, \mathsf{SKE.ct}, \mathsf{SKE.sk}) \\ &= C(\mathbf{x}_{\mathsf{pub}}, \mathsf{SKE.Dec}(\mathsf{SKE.sk}, \mathsf{SKE.ct})) \quad (\text{ by definition of } C')\end{aligned}$$
- Next, from the perfect correctness of the SKE scheme, we have
$$\mathsf{SKE.Dec}(\mathsf{SKE.sk}, \mathsf{SKE.ct}) = \mathsf{SKE.Dec}(\mathsf{SKE.sk}, \mathsf{SKE.Enc}(\mathsf{SKE.sk}, \mathbf{x}_{\mathsf{priv}})) = \mathbf{x}_{\mathsf{priv}}$$

Thus $\mathsf{Dec}(\mathsf{mpk}, \mathbf{y}, \mathsf{sk}_{C'}, C', \mathsf{ct}) = C(\mathbf{x}_{\mathsf{pub}}, \mathsf{SKE.Dec}(\mathsf{SKE.sk}, \mathsf{SKE.ct})) = C(\mathbf{x}_{\mathsf{pub}}, \mathbf{x}_{\mathsf{priv}})$ and hence the correctness.

Efficiency. Here, we show
$$|\mathsf{mpk}| = \mathrm{poly}(\lambda), \ |\mathsf{sk}_{C'}| = \mathrm{poly}(\lambda), \ |\mathsf{ct}'| = \mathrm{poly}(\lambda) + L_{\mathsf{priv}}.$$
Given that the length of the private input for the underlying PHprFE is λ, it is straightforward to see the first two equations above hold by the efficiency of PHprFE. To bound the length of the ciphertext ct', we first observe that $\mathsf{ct}' = (\mathsf{ct}_{\mathsf{off}}, \mathsf{SKE.ct}, \mathsf{ct}_{\mathsf{on}})$, where $\mathsf{ct} := (\mathsf{ct}_{\mathsf{off}}, \mathsf{ct}_{\mathsf{on}})$ constitutes a ciphertext of the underlying PHprFE encrypting $(\mathbf{y}, \mathbf{z})$. We have $|\mathsf{ct}| = \mathrm{poly}(\lambda, |\mathbf{z}|) = \mathrm{poly}(\lambda)$ and $|\mathsf{SKE.ct}| = \mathrm{poly}(\lambda) + L_{\mathsf{priv}}$. Therefore, the last equation above follows as well.

We therefore have the following theorem. The security of the scheme is proven in Theorem 5.12.

Theorem 5.10. Assuming LWE and evasive LWE assumptions, there exists a partially hiding pseudorandom FE scheme, for circuit class $\mathcal{C} = \{C : \{0,1\}^{L_{\mathsf{pub}}} \times \{0,1\}^{L_{\mathsf{priv}}} \to \{0,1\}\}$, that satisfies reusable security (as per Definition 5.4) whose sizes of the master public key and the secret key are fixed polynomial $\mathrm{poly}(\lambda)$. Furthermore, the size of the ciphertext is $L_{\mathsf{priv}} + \mathrm{poly}(\lambda)$.

Optimal prFE. Using Remark 5.3, we get a prFE scheme as a special case of PHprFE, for $\mathbf{x}_{\mathsf{pub}} = \perp$ and $\mathbf{x}_{\mathsf{priv}} = \mathbf{x}$. Next, observing the fact that (1) with $\mathbf{x}_{\mathsf{pub}} = \perp$ the security of PHprFE (Definition 5.2) is equivalent to single-challenge security of prFE (Definition 3.2, with $Q_{\mathsf{msg}} = 1$) and (2) Definition 5.2 is implied by Definition 5.4, we get a prFE scheme with optimal parameters. We formalise this in the following theorem.

Theorem 5.11. Assuming LWE and evasive LWE assumptions, there exists a prFE scheme, for circuit class $\mathcal{C} = \{C : \{0,1\}^{L_{\mathsf{inp}}} \to \{0,1\}\}$, that satisfies security (as per Definition 3.2, with $Q_{\mathsf{msg}} = 1$) and efficiency

$$|\mathsf{mpk}| = \mathrm{poly}(\lambda), \;\; |\mathsf{sk}_C| = \mathrm{poly}(\lambda), \;\; |\mathsf{ct}| = L_{\mathsf{inp}} + \mathrm{poly}(\lambda).$$

Security The following theorem asserts the security of the construction.

Theorem 5.12. The above construction PHprFE' satisfies reusable security as per Definition 5.4 if so does PHprFE and SKE is secure as per Definition 2.1.

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

1. **Key Queries.** It issues Q_{key} key queries $C_1, \ldots, C_{Q_{\mathsf{key}}}$.
2. **Ciphertext Queries.** It issues messages $\mathbf{x}^1 = (\mathbf{x}^1_{\mathsf{pub}}, \mathbf{x}^1_{\mathsf{priv}}), \ldots, \mathbf{x}^{Q_{\mathsf{msg}}} = (\mathbf{x}^{Q_{\mathsf{msg}}}_{\mathsf{pub}}, \mathbf{x}^{Q_{\mathsf{msg}}}_{\mathsf{priv}})$.
3. **Auxiliary Information.** It outputs the auxiliary information $\mathsf{aux}_{\mathcal{A}}$.

To prove the security as per Definition 5.4, we prove

$$\begin{pmatrix} \mathsf{mpk},\ \mathsf{aux},\ \{C_k\}_{k\in[Q_{\mathsf{msg}}]}, \\ \left\{\mathsf{sk}_k := \mathsf{sk}_{C'_k}\right\}_{k\in[Q_{\mathsf{msg}}]}, \\ \mathsf{Lobf}_{\mathsf{off}},\ \left\{\ \mathbf{x}^j_{\mathsf{pub}},\ \mathsf{SKE.ct}^j,\ \mathsf{ct}^j_{\mathsf{on}}\ \right\}_{j\in[Q_{\mathsf{msg}}]} \end{pmatrix} \approx_c \begin{pmatrix} \mathsf{mpk},\ \mathsf{aux},\ \{C_k\}_{k\in[Q_{\mathsf{msg}}]}, \\ \left\{\mathsf{sk}_k := \mathsf{sk}_{C'_k}\right\}_{k\in[Q_{\mathsf{msg}}]}, \\ \mathsf{Lobf}_{\mathsf{off}},\ \left\{\ \mathbf{x}^j_{\mathsf{pub}},\ \gamma^j,\ \delta^j\ \right\}_{j\in[Q_{\mathsf{msg}}]} \end{pmatrix} \tag{30}$$

where $\gamma^j \leftarrow \mathcal{CT}_{\mathsf{SKE}}$ and $\delta^j \leftarrow \mathcal{CT}_{\mathsf{PHprFE}}$ for $j \in [Q_{\mathsf{msg}}]$, assuming we have

$$(1^\lambda,\ \mathsf{aux},\ \{\mathbf{x}^j_{\mathsf{pub}},\ C_k,\ C_k(\mathbf{x}^j)\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]}) \approx_c (1^\lambda,\ \mathsf{aux},\ \{\mathbf{x}^j_{\mathsf{pub}},\ C_k,\ \Delta^j_k \leftarrow \{0,1\}\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]}) \tag{31}$$

where

$$\begin{aligned}
&\left(\{C_k\}_{k\in[Q_{\mathsf{key}}]}, \{\mathbf{x}^j = (\mathbf{x}^j_{\mathsf{pub}}, \mathbf{x}^j_{\mathsf{priv}})\}_{j\in[Q_{\mathsf{msg}}]}, \mathsf{aux} \in \{0,1\}^*\right) \leftarrow \mathsf{Samp}_{\mathsf{PHprFE}'}(1^\lambda), \\
&\mathsf{SKE.sk} \leftarrow \mathsf{SKE.Setup}(1^\lambda), \quad \mathsf{SKE.ct}^j \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.sk}, \mathbf{x}^j_{\mathsf{priv}}), \\
&(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^{L_{\mathsf{pub}}+L_{\mathsf{priv}}+\lambda}, 1^\lambda), \\
&\mathsf{sk}_k \leftarrow \mathsf{KeyGen}(\mathsf{msk}, C'_k) \text{ for } k \in [Q_{\mathsf{key}}], \text{ where } C'_k \text{ is defined from } C_k \text{ as in the construction,} \\
&(\mathsf{ct}_{\mathsf{off}}, \mathsf{st}) \leftarrow \mathsf{EncOff}(\mathsf{mpk}), \\
&\mathsf{ct_{on}}^j \leftarrow \mathsf{EncOn}(\mathsf{st}, \mathbf{x}^j_{\mathsf{priv}}) \;\; \text{for } j \in [Q_{\mathsf{msg}}].
\end{aligned}$$

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

$$\begin{pmatrix} \text{Functions:} & \{C'_k\}_{k\in[Q_{\mathsf{key}}]}, \\ \text{Inputs:} & \left\{X_j := \left(X^j_{\mathsf{pub}} := (\mathbf{x}^j_{\mathsf{pub}}, \mathsf{SKE.ct}^j), X^j_{\mathsf{priv}} := \mathsf{SKE.sk}^j\right)\right\}_{j\in[Q_{\mathsf{msg}}]}, \\ \text{Auxiliary Information:} & \mathsf{aux}_{\mathsf{prFE}} := \left(\mathsf{aux},\ \{C_k\}_{k\in[Q_{\mathsf{key}}]}\right) \end{pmatrix}$$

By the security guarantee of PHprFE with sampler $\mathsf{Samp}_{\mathsf{PHprFE}}$,

$$\begin{pmatrix} \mathsf{mpk},\ \mathsf{aux},\ \{C_k\}_{k\in[Q_{\mathsf{key}}]}, \\ \left\{\mathsf{sk}_k := \mathsf{sk}_{C'_k}\right\}_{k\in[Q_{\mathsf{key}}]}, \\ \mathsf{Lobf}_{\mathsf{off}}, \left\{X^j_{\mathsf{pub}} = (\mathbf{x}^j_{\mathsf{pub}}, \mathsf{SKE.ct}^j), \mathsf{ct}^j_{\mathsf{on}}\right\}_{j\in[Q_{\mathsf{msg}}]} \end{pmatrix} \approx_c \begin{pmatrix} \mathsf{mpk},\ \mathsf{aux},\ \{C_k\}_{k\in[Q_{\mathsf{key}}]}, \\ \left\{\mathsf{sk}_k := \mathsf{sk}_{C'_k}\right\}_{k\in[Q_{\mathsf{key}}]}, \\ \mathsf{Lobf}_{\mathsf{off}}, \left\{X^j_{\mathsf{pub}} = (\mathbf{x}^j_{\mathsf{pub}}, \mathsf{SKE.ct}^j), \delta^j\right\}_{j\in[Q_{\mathsf{msg}}]} \end{pmatrix} \tag{32}$$

holds if

$$\left(\mathsf{aux}, \left\{C_k,\ X^j_{\mathsf{pub}},\ C'_k(X^j)\right\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]}\right) \approx_c \left(\mathsf{aux}, \left\{C_k,\ X^j_{\mathsf{pub}},\ \Delta^j_k\right\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]}\right). \tag{33}$$

We first observe that Equation (32) implies Equation (30), since we can invoke the security of SKE to conclude that

$$\begin{pmatrix} \mathsf{mpk},\ \mathsf{aux},\ \{C_k\}_{k\in[Q_{\mathsf{key}}]}, \\ \left\{\mathsf{sk}_k := \mathsf{sk}_{C'_k}\right\}_{k\in[Q_{\mathsf{key}}]}, \\ \mathsf{Lobf}_{\mathsf{off}},\ \left\{\ \mathbf{x}^j_{\mathsf{pub}},\ \mathsf{SKE.ct}^j,\ \delta^j\ \right\}_{j\in[Q_{\mathsf{msg}}]} \end{pmatrix} \approx_c \begin{pmatrix} \mathsf{mpk},\ \mathsf{aux},\ \{C_k\}_{k\in[Q_{\mathsf{key}}]}, \\ \left\{\mathsf{sk}_k := \mathsf{sk}_{C'_k}\right\}_{k\in[Q_{\mathsf{key}}]}, \\ \mathsf{Lobf}_{\mathsf{off}},\ \left\{\ \mathbf{x}^j_{\mathsf{pub}},\ \gamma^j,\ \delta^j\ \right\}_{j\in[Q_{\mathsf{msg}}]} \end{pmatrix}$$

holds by noting that $\mathsf{SKE.sk}^j$ is used only for computing $\mathsf{SKE.ct}^j$ and not used anywhere else.

Therefore, it suffices to prove Equation (33) to conclude the proof. We have

$$\begin{aligned} \left(\mathsf{aux}, \left\{C_k,\ X^j_{\mathsf{pub}} = (\mathbf{x}^j_{\mathsf{pub}},\ \mathsf{SKE.ct}^j),\ C'_k(X^j)\right\}_{j,k}\right) &= \left(\mathsf{aux}, \left\{C_k,\ \mathbf{x}^j_{\mathsf{pub}},\ \mathsf{SKE.ct}^j,\ C_k(\mathbf{x}^j)\right\}_{j,k}\right) \\ &\approx_c \left(\mathsf{aux}, \left\{C_k,\ \mathbf{x}^j_{\mathsf{pub}},\ \gamma^j \leftarrow \mathcal{CT}_{\mathsf{SKE}},\ C_k(\mathbf{x}^j)\right\}_{j,k}\right) \\ &\approx_c \left(\mathsf{aux}, \left\{C_k,\ \mathbf{x}^j_{\mathsf{pub}},\ \gamma^j \leftarrow \mathcal{CT}_{\mathsf{SKE}},\ \Delta^j_k \leftarrow \{0,1\}\right\}_{j,k}\right) \\ &\approx_c \left(\mathsf{aux}, \left\{C_k,\ \mathbf{x}^j_{\mathsf{pub}},\ \mathsf{SKE.ct}^j,\ \Delta^j_k \leftarrow \{0,1\}\right\}_{j,k}\right) \end{aligned}$$

where the first line follows from the definition of C'_k and X^j, the second from the security of SKE noting that $\mathsf{SKE.sk}^j$ is used only for computing $\mathsf{SKE.ct}^j$ and not used anywhere else, the third from Equation (31), noting that adding random string $\{\gamma^j\}_j$ to the distributions in Equation (31) does not make the task of distinguishing the distributions any easier, and the fourth from the security of SKE again. This proves Equation (33) and therefore completes the proof. □

5.5 Handling Longer Output

So far we have only considered the case where C is a circuit that outputs a single-bit string. Here, we discuss more general case where the output of the circuit C is longer. To handle such circuits in the construction, we only change the key generation algorithm. To generate a secret key for $C : \{0,1\}^{\mathsf{L}} \to \{0,1\}^{\mathsf{out}}$, we first consider circuits $\{C_j\}_{j\in[\mathsf{out}]}$, where C_j is the circuit that outputs the j-th bit of C's output and then generate secret keys for $\{C_j\}_j$. It is then easy to see that the all bits of $C(\mathbf{x})$ can be recovered by using the secret keys for $\{C_j\}_j$ by decrypting a ciphertext encrypting $\mathbf{x}$. This does not change the size of the master public key and ciphertext, but makes the secret key linearly dependent on the output length of C. Thus, we get the following theorems.

Theorem 5.13. Assuming LWE and evasive LWE assumptions, there exists a PHprFE scheme, for circuit class $\mathcal{C} = \{C : \{0,1\}^{L_{\mathsf{pub}}} \times \{0,1\}^{L_{\mathsf{priv}}} \to \{0,1\}^{\mathsf{out}}\}$, that satisfies reusable security (as per Definition 5.4) and efficiency

$$|\mathsf{mpk}| = \mathrm{poly}(\lambda),\quad |\mathsf{sk}_C| = \mathsf{out}\cdot\mathrm{poly}(\lambda),\quad |\mathsf{ct}| = L_{\mathsf{priv}} + \mathrm{poly}(\lambda).$$

Theorem 5.14. Assuming LWE and evasive LWE assumptions, there exists a prFE scheme, for circuit class $\mathcal{C} = \{C : \{0,1\}^{L_{\mathsf{inp}}} \to \{0,1\}\}$, that satisfies security (as per Definition 3.2, with $Q_{\mathsf{msg}} = 1$) and efficiency

$$|\mathsf{mpk}| = \mathrm{poly}(\lambda),\quad |\mathsf{sk}_C| = \mathsf{out}\cdot\mathrm{poly}(\lambda),\quad |\mathsf{ct}| = L_{\mathsf{inp}} + \mathrm{poly}(\lambda).$$

6 KP-ABE/PE for Unbounded Depth Circuits with Optimal Parameters

Here we construct KP-ABE and KP-PE schemes supporting unbounded depth circuits and achieving optimal parameters.

6.1 Construction of KP-ABE with Optimal Parameters

In this section we construct an ABE scheme $\mathsf{kpABE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ for message space $\{0,1\}$ and circuit family $\mathcal{C}_\ell$ consisting of circuits with input space $\{0,1\}^\ell$ and output space $\{0,1\}$. For the purpose of application in Section 7, we consider a decomposed encryption algorithm as $\mathsf{Enc} = (\mathsf{EncOff}, \mathsf{EncOn})$ in our construction where $\mathsf{EncOff}(\mathsf{mpk}) \to (\mathsf{ct}_{\mathsf{off}}, \mathsf{st})$, $\mathsf{EncOn}(\mathsf{st}, \mathbf{x}, \mu) \to \mathsf{ct}_{\mathsf{on}}$ and Enc outputs $(\mathsf{ct}_{\mathsf{off}}, \mathsf{ct}_{\mathsf{on}})$.

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

1. A PHprFE scheme $\mathsf{PHprFE} = \mathsf{PHprFE}.(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc} = (\mathsf{EncOff}, \mathsf{EncOn}), \mathsf{Dec})$ from Section 5.4 for circuit family $\{\mathcal{C}_{\mathsf{prm}} = \{C : \mathcal{X}_{\mathsf{pub}} \times \mathcal{X}_{\mathsf{priv}} \to \mathcal{Y}\}\}_{\mathsf{prm}}$ where $\mathcal{X}_{\mathsf{pub}} = \{0,1\}^\ell, \mathcal{X}_{\mathsf{priv}} = \{0,1\}^{\lambda+1}$ and $\mathcal{Y} = \{0,1\}$. We denote the online part of the ciphertext space by $\mathsf{PHprFE}.\mathcal{CT}_{\mathsf{on}}$. We require the PHprFE scheme to satisfy reusable security (Definition 5.4). As we show in Theorem 5.10, such a PHprFE scheme can be constructed assuming LWE and evasive LWE assumptions.
2. A PRF scheme $\mathsf{PRF} : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}$. It is known that PRF can be constructed from one-way functions.

Now, we describe our construction.

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

- Generate $(\mathsf{PHprFE.msk}, \mathsf{PHprFE.mpk}) \leftarrow \mathsf{PHprFE.Setup}(1^\lambda, (1^\ell, 1^{\lambda+1}))$.
- Output $\mathsf{msk} = \mathsf{PHprFE.msk}$ and $\mathsf{mpk} = \mathsf{PHprFE.mpk}$.

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

- Parse $\mathsf{msk} = \mathsf{PHprFE.msk}$.
- Sample $\mathbf{r} \leftarrow \{0,1\}^\lambda$ and define the circuit $C[\mathbf{r}]$, with $\mathbf{r}$ hardwired, as follows.
  On input $(\mathbf{x}, \mu, \mathsf{sd})$,
$$C[\mathbf{r}](\mathbf{x}, \mu, \mathsf{sd}) = \begin{cases} \mu & \text{if } C(\mathbf{x}) = 0 \\ \mathsf{PRF}(\mathsf{sd}, \mathbf{r}) & \text{otherwise.} \end{cases}$$
- Compute $\mathsf{PHprFE.sk} \leftarrow \mathsf{PHprFE.KeyGen}(\mathsf{PHprFE.msk}, C[\mathbf{r}])$.
- Output $\mathsf{sk}_C = (\mathsf{PHprFE.sk}, \mathbf{r})$.

$\mathsf{Enc}(\mathsf{mpk}, \mathbf{x}, \mu)$. The encryption algorithm works as follows.

$\mathsf{EncOff}(\mathsf{mpk})$. The offline phase of encryption does the following.

- Parse $\mathsf{mpk} = \mathsf{PHprFE.mpk}$.
- Compute $(\mathsf{PHprFE.ct}_{\mathsf{off}}, \mathsf{PHprFE.st}) \leftarrow \mathsf{PHprFE.EncOff}(\mathsf{PHprFE.mpk})$.
- Output $\mathsf{ct}_{\mathsf{off}} = \mathsf{PHprFE.ct}_{\mathsf{off}}$ and $\mathsf{st} = \mathsf{PHprFE.st}$.

$\mathsf{EncOn}(\mathsf{st}, \mathbf{x}, \mu)$. The online phase of encryption does the following.

- Parse $\mathsf{st} = \mathsf{PHprFE.st}$.
- Sample $\mathsf{sd} \leftarrow \{0,1\}^\lambda$, set $X_{\mathsf{pub}} = \mathbf{x}$ and $X_{\mathsf{priv}} = (\mu, \mathsf{sd})$.
- Compute $\mathsf{PHprFE.ct}_{\mathsf{on}} \leftarrow \mathsf{PHprFE.Enc}(\mathsf{PHprFE.st}, X_{\mathsf{pub}}, X_{\mathsf{priv}})$.
- Output $\mathsf{ct}_{\mathsf{on}} := \mathsf{PHprFE.ct}_{\mathsf{on}}$.

Output $\mathsf{ct} := (\mathsf{ct}_{\mathsf{off}}, \mathsf{ct}_{\mathsf{on}})$.

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

- Parse $\mathsf{mpk} = \mathsf{PHprFE.mpk}, \mathsf{sk}_C = (\mathsf{PHprFE.sk}, \mathbf{r})$ and $\mathsf{ct} = (\mathsf{ct}_{\mathsf{off}}, \mathsf{ct}_{\mathsf{on}}) = (\mathsf{PHprFE.ct}_{\mathsf{off}}, \mathsf{PHprFE.ct}_{\mathsf{on}})$.
- Compute $y = \mathsf{PHprFE.Dec}(\mathsf{PHprFE.mpk}, \mathbf{x}, \mathsf{PHprFE.sk}, C[\mathbf{r}], \mathsf{PHprFE.ct})$, where $C[\mathbf{r}]$ is as defined in the key generation algorithm.
- Output y.

Correctness. For $\mathsf{sk}_C = (\mathsf{PHprFE.sk}, \mathbf{r})$ and $\mathsf{ct} = (\mathsf{ct}_{\mathsf{off}}, \mathsf{ct}_{\mathsf{on}}) = (\mathsf{PHprFE.ct}_{\mathsf{off}}, \mathsf{PHprFE.ct}_{\mathsf{on}}) = \mathsf{PHprFE.ct}$, we have

$$\begin{aligned}\mathsf{PHprFE.Dec}(\mathsf{PHprFE.mpk}, \mathbf{x}, \mathsf{PHprFE.sk}, C[\mathbf{r}], \mathsf{PHprFE.ct}) &= C[\mathbf{r}](\mathbf{x}, \mu, \mathsf{sd}) \\ &= \begin{cases} \mu & \text{if } C(\mathbf{x}) = 1 \\ \mathsf{PRF}(\mathsf{sd}, \mathbf{r}) & \text{otherwise.} \end{cases}\end{aligned}$$

from the correctness of PHprFE scheme. Now, if $C(\mathbf{x}) = 1$, then from the definition of $C[\mathbf{r}]$, we get $C[\mathbf{r}](\mathbf{x}, \mu, \mathsf{sd}) = \mu$ and hence the decryption outputs $y = \mu$ correctly.

Efficiency. Instantiating the PHprFE scheme from Section 5.4 with $|\mathsf{PHprFE.mpk}| = \mathrm{poly}(\lambda)$, $|\mathsf{PHprFE.sk}_C| = \mathrm{poly}(\lambda)$, $|\mathsf{PHprFE.ct}| = \mathrm{poly}(\lambda) + |\mathbf{x}_{\mathsf{priv}}|$, our kpABE scheme satisfies

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

We formalise this instantiation using the following theorem. The security is proved in Section 6.2.

Theorem 6.1. Under the LWE and Evasive LWE assumption, there exists very selectively secure KP-ABE scheme supporting circuits $\{C : \{0,1\}^\ell \to \{0,1\}\}$ with unbounded depth and single bit message space with

$$|\mathsf{mpk}| = \mathrm{poly}(\lambda),\ |\mathsf{sk}_C| = \mathrm{poly}(\lambda),\ |\mathsf{ct}| = \mathrm{poly}(\lambda). \tag{34}$$

6.2 Security of KP-ABE

For our application in Section 7, we introduce the following security notion.

Definition 6.2 ($\mathsf{VerSel\text{-}INDr}$ **Reusable Security).** A kpABE scheme for circuit family $\mathcal{C}_\ell = \{C : \{0,1\}^\ell \to \{0,1\}\}$ is said to satisfy $\mathsf{VerSel\text{-}IND}$ reusable security if for all stateful PPT adversary $\mathcal{A}$, the following holds

$$\Pr\left[\beta' = \beta : \begin{array}{l} (\mathsf{aux}_{\mathcal{A}}, C^1, \ldots, C^{Q_{\mathsf{key}}}, \mathbf{x}^1, \ldots, \mathbf{x}^{Q_{\mathsf{msg}}}, \mu) \leftarrow \mathcal{A}(1^\lambda); \\ (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell); \\ (\mathsf{ct}_{\mathsf{off}}, \mathsf{st}) \leftarrow \mathsf{EncOff}(\mathsf{mpk}); \\ \{\mathsf{ct}^j_{\mathsf{on},0} \leftarrow \mathsf{EncOn}(\mathsf{st}, \mathbf{x}^j, \mu), \mathsf{ct}^j_{\mathsf{on},1} \leftarrow \mathcal{CT}_{\mathsf{on}}\}_{j \in [Q_{\mathsf{msg}}]}, \beta \leftarrow \{0,1\}; \\ \beta' \leftarrow \mathcal{A}(\mathsf{aux}_{\mathcal{A}}, \mathsf{mpk}, \{C^k, \mathsf{sk}_{C^k}\}_{k \in Q_{\mathsf{key}}}, \mathsf{ct}_{\mathsf{off}}, \{\mathbf{x}^j, \mathsf{ct}^j_{\mathsf{on},\beta}\}_{j \in [Q_{\mathsf{msg}}]}) \end{array}\right] \le \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $\mathcal{CT}_{\mathsf{on}}$ is the ciphertext space of EncOn. We require that for all key queries $C^1, \ldots, C^{Q_{\mathsf{key}}}$ and challenge attribute queries $\mathbf{x}^1, \ldots, \mathbf{x}^{Q_{\mathsf{msg}}}$ we have $C^k(\mathbf{x}^j) = 0$.

Remark 6.3. We note that the above security definition implies more standard $\mathsf{VerSel\text{-}IND}$ security for ABE. This can be seen by considering the case of $Q_{\mathsf{msg}} = 1$ and recalling that $\mathsf{ct} = (\mathsf{ct}_{\mathsf{off}}, \mathsf{ct}_{\mathsf{on}})$. The above security definition in this special case implies that the message carrying part of the ciphertext is pseudorandom in $\mathsf{VerSel\text{-}IND}$ security game. This immediately implies $\mathsf{VerSel\text{-}IND}$ security.

We prove the above security of our scheme using the following theorem.

Theorem 6.4. Assume the PHprFE scheme satisfies reusable security (Definition 5.4) w.r.t. the sampler class containing the sampler Samp as defined in Eq. 36 and PRF is secure. Then the above construction of kpABE scheme is secure (Definition 6.2).

Proof. Consider a PPT adversary $\mathcal{A}$ that outputs $\mathsf{coins}_{\mathcal{A}}, C^1, \ldots, C^{Q_{\mathsf{key}}}, \mathbf{x}^1, \ldots, \mathbf{x}^{Q_{\mathsf{msg}}}, \mu$. To prove reusable security as per Definition 6.2, we show

$$\begin{pmatrix} \mathsf{coins}_{\mathcal{A}},\ \mathsf{mpk} = \mathsf{PHprFE.mpk}, \\ \{\mathsf{sk}^k = (\mathsf{PHprFE.sk}^k, \mathbf{r}^k), C^k\}_{k\in[Q_{\mathsf{key}}]}, \\ \mathsf{PHprFE.ct}_{\mathsf{off}}, \{\mathbf{x}^j, \mathsf{PHprFE.ct}_{\mathsf{on}}^j\}_{j\in[Q_{\mathsf{msg}}]} \end{pmatrix} \approx_c \begin{pmatrix} \mathsf{coins}_{\mathcal{A}},\ \mathsf{mpk} = \mathsf{PHprFE.mpk}, \\ \{\mathsf{sk}^k = (\mathsf{PHprFE.sk}^k, \mathbf{r}^k), C^k\}_{k\in[Q_{\mathsf{key}}]}, \\ \mathsf{PHprFE.ct}_{\mathsf{off}},\ \{\mathbf{x}^j, \delta^j \leftarrow \mathsf{PHprFE}.\mathcal{CT}_{\mathsf{on}}\}_{j\in[Q_{\mathsf{msg}}]} \end{pmatrix} \tag{35}$$

Also for all the key queries $C^1, \ldots, C^{Q_{\mathsf{key}}}$ and challenge attribute queries $\mathbf{x}^1, \ldots, \mathbf{x}^{Q_{\mathsf{msg}}}$ issued by the adversary, we have $C^k(\mathbf{x}^j) = 0$.
We invoke the security of PHprFE scheme with sampler Samp that outputs

$$\begin{pmatrix} \text{circuits:} & \{C^k[\mathbf{r}^k]\}_{k\in[Q_{\mathsf{key}}]}, \\ \text{Inputs:} & \{X^j_{\mathsf{pub}} = \mathbf{x}^j, X^j_{\mathsf{priv}} = (\mu, \mathsf{sd}^j)\}_{j\in[Q_{\mathsf{msg}}]}, \\ \text{Auxiliary Information:} & \mathsf{aux}_{\mathcal{A}} = (\mathsf{coins}_{\mathcal{A}}, C^1, \ldots, C^{Q_{\mathsf{key}}}, \mathbf{r}^1, \ldots, \mathbf{r}^{Q_{\mathsf{key}}}, \mathsf{coins}_{\mathcal{A}}) \end{pmatrix} \tag{36}$$

Using the guarantee of PHprFE scheme with sampler Samp we have that

$$\begin{pmatrix} \mathsf{aux}_{\mathcal{A}},\ \mathsf{PHprFE.mpk},\ \{C^k[\mathbf{r}^k]\}_{k\in[Q_{\mathsf{key}}]}, \\ \{\mathsf{PHprFE.sk}^k\}_{k\in[Q_{\mathsf{key}}]}, \\ \mathsf{PHprFE.ct}_{\mathsf{off}},\ \{X^j_{\mathsf{pub}}, \mathsf{PHprFE.ct_{on}}^j\}_{j\in[Q_{\mathsf{msg}}]} \end{pmatrix} \approx_c \begin{pmatrix} \mathsf{aux}_{\mathcal{A}},\ \mathsf{PHprFE.mpk},\ \{C^k[\mathbf{r}^k]\}_{k\in[Q_{\mathsf{key}}]}, \\ \{\mathsf{PHprFE.sk}^k\}_{k\in[Q_{\mathsf{key}}]}, \\ \mathsf{PHprFE.ct}_{\mathsf{off}},\ \{X^j_{\mathsf{pub}}, \delta^j\}_{j\in[Q_{\mathsf{msg}}]} \end{pmatrix} \tag{37}$$

$$\begin{aligned} \text{if}\quad & \Big(\mathsf{aux}_{\mathcal{A}}, \{C^k[\mathbf{r}^k]\}_{k\in[Q_{\mathsf{key}}]},\ \{X^j_{\mathsf{pub}}\}_{j\in[Q_{\mathsf{msg}}]},\ \{C^k[\mathbf{r}^k](X^j_{\mathsf{pub}}, X^j_{\mathsf{priv}})\}_{k\in[Q_{\mathsf{key}}], j\in[Q_{\mathsf{msg}}]}\Big) \\ & \approx_c \Big(\mathsf{aux}_{\mathcal{A}}, \{C^k[\mathbf{r}^k]\}_{k\in[Q_{\mathsf{key}}]},\ \{X^j_{\mathsf{pub}}\}_{j\in[Q_{\mathsf{msg}}]},\ \{\Delta_{k,j} \leftarrow \{0,1\}^{\lambda}\}_{k\in[Q_{\mathsf{key}}], j\in[Q_{\mathsf{msg}}]}\Big) \end{aligned} \tag{38}$$

where $X^j_{\mathsf{pub}} = \mathbf{x}^j, X^j_{\mathsf{priv}} = (\mu, \mathsf{sd}^j)$ for $\mathsf{sd}^j \leftarrow \{0,1\}^{\lambda}$, $j \in [Q_{\mathsf{msg}}]$,

$$\begin{aligned} & (\mathsf{PHprFE.mpk}, \mathsf{PHprFE.msk}) \leftarrow \mathsf{PHprFE.Setup}(1^{\lambda}, (1^{\ell}, 1^{\lambda+1})), \\ & \mathsf{PHprFE.sk}^k \leftarrow \mathsf{PHprFE.KeyGen}(\mathsf{PHprFE.msk}, C^k[\mathbf{r}^k]),\ \text{for } \mathbf{r}^k \leftarrow \{0,1\}^{\lambda}, \\ & (\mathsf{PHprFE.ct}_{\mathsf{off}}, \mathsf{PHprFE.st}) \leftarrow \mathsf{PHprFE.EncOff}(\mathsf{PHprFE.mpk}), \\ & \mathsf{PHprFE.ct_{on}}^j \leftarrow \mathsf{PHprFE.Enc}(\mathsf{PHprFE.st}, X^j_{\mathsf{pub}}, X^j_{\mathsf{priv}}), \\ & \delta^j \leftarrow \mathsf{PHprFE}.\mathcal{CT}_{\mathsf{on}} \text{ for } j \in [Q_{\mathsf{msg}}] \text{ where } \mathsf{PHprFE}.\mathcal{CT}_{\mathsf{on}} \text{ is the ciphertext space of } \mathsf{PHprFE.EncOn}. \end{aligned}$$

First we note that rearranging the terms of Equation (37), it is same as distribution in Equation (35). Thus, to prove Equation (35), it suffices to prove Equation (38).
Equation (38) holds from the security of the underlying PRF scheme. To see this note that

$$C^k[\mathbf{r}^k](X^j_{\mathsf{pub}}, X^j_{\mathsf{priv}}) = C^k[\mathbf{r}^k](\mathbf{x}^j, \mu, \mathsf{sd}^j) = \begin{cases} \mu & \text{if } C^k(\mathbf{x}^j) = 1 \\ \mathsf{PRF}(\mathsf{sd}^j, \mathbf{r}^k) & \text{otherwise.} \end{cases}$$

By the admissibility of the adversary into the kpABE security game, we have $C^k(\mathbf{x}^j) = 0$ for all $k \in [Q_{\mathsf{key}}], j \in [Q_{\mathsf{msg}}]$. So, we get

$$\begin{aligned} C^k[\mathbf{r}^k](\mathbf{x}^j, \mu, \mathsf{sd}^j) &= \mathsf{PRF}(\mathsf{sd}^j, \mathbf{r}^k) \\ &\approx_c \Delta_{j,k} \leftarrow \{0,1\} \end{aligned}$$

where the last equation follows from the security of the PRF scheme. □

6.3 Predicate Encryption with Optimal Parameters

In this section we sketch out the construction of a PE scheme $\mathsf{PE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ supporting unbounded depth circuits and achieving optimal parameters. The construction is same as that in Section 6.1 with the following changes.

1. In the building blocks we use a PHprFE scheme with $\mathcal{X}_{\mathsf{pub}} = \{\perp\}, \mathcal{X}_{\mathsf{priv}} = \{0,1\}^{\ell+\lambda+1}$ and $\mathcal{Y} = \{0,1\}$.
2. In $\mathsf{KeyGen}(\mathsf{msk}, C)$ algorithm the circuit $C[\mathbf{r}]$ is defined as follows. On input $(\perp, \mathbf{x}, \mu, \mathsf{sd})$,
$$\mathsf{C}[\mathbf{r}](\perp, \mathbf{x}, \mu, \mathsf{sd}) = \begin{cases} \mu & \text{if } C(\mathbf{x}) = 1 \\ \mathsf{PRF}(\mathsf{sd}, \mathbf{r}) & \text{otherwise.} \end{cases}$$
3. In EncOn algorithm we set $\mathbf{x}_{\mathsf{pub}} = \perp$ and $\mathbf{x}_{\mathsf{priv}} = (\mathbf{x}, \mu, \mathsf{sd})$.
4. In $\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_C, C, \mathsf{ct})$[7], we compute $y = \mathsf{PHprFE.Dec}(\mathsf{PHprFE.mpk}, \perp, \mathsf{PHprFE.sk}, C[\mathbf{r}], \mathsf{PHprFE.ct})$.

It is easy to see that the above PE scheme satisfies correctness. The PE scheme also satisfies VerSel-INDr reusable security as defined in Definition 6.2 with a similar security proof as in Section 6.2 with the above specified changes. Note that since the online part of the encryption $\mathsf{EncOn}(\mathsf{st}, \mathbf{x}, \mu)$ encodes both the attribute and the message μ, replacing it with a random string hides the information of $\mathbf{x}$ and μ and thus it satisfies the security requirement for a PE scheme. As for the efficiency, since we encode $\mathbf{x}$ into the private part of the input, the ciphertext size is $|\mathbf{x}| + \mathrm{poly}(\lambda)$. Summarizing the above discussion, we have the following theorem.

Theorem 6.5. Under the LWE and Evasive LWE assumption, there exists very selectively secure KP-PE scheme supporting circuits $\{C : \{0,1\}^\ell \to \{0,1\}\}$ with unbounded depth with and single bit message space with

$$|\mathsf{mpk}| = \mathrm{poly}(\lambda), \; |\mathsf{sk}_C| = \mathrm{poly}(\lambda), \; |\mathsf{ct}| = \mathrm{poly}(\lambda) + |\mathbf{x}|. \tag{39}$$

where $\mathbf{x} \in \{0,1\}^\ell$.

6.4 Extending the Message Space

Our construction of ABE and PE only allows us to encrypt a single-bit message. Here, we explain how to extend the message space while maintaining the optimal parameter size. In both cases, it suffices to encrypt message of length λ, since this allows us to employ the hybrid encryption approach, where we encrypt a secret key $\mathsf{SKE.sk} \in \{0,1\}^\lambda$ of an SKE scheme and then use this to encrypt the message.

In the case of ABE, we simply encrypt each bit of $\mathsf{SKE.sk}$, which blows up the ciphertext size by a factor of λ, but this still results in the optimal parameter size of Equation (34). In the case of PE, this approach leads to a ciphertext of size $\mathrm{poly}(\lambda) + \lambda|\mathbf{x}|$, which ruins the optimal ciphertext size of only additively depending on the length of the attribute. Instead, we change the construction of PE from PHprFE by setting $\mathbf{x}_{\mathsf{pub}} = \perp$ and $\mathbf{x}_{\mathsf{priv}} = (\mathbf{x}, \mathsf{SKE.sk}, \mathsf{sd})$ and considering a circuit $C[i, \mathbf{r}]$ for $i \in [\lambda]$ that is defined as

$$\mathsf{C}[i, \mathbf{r}](\perp, \mathbf{x}, \mathsf{SKE.sk}, \mathsf{sd}) = \begin{cases} \mathsf{SKE.sk}_i & \text{if } C(\mathbf{x}) = 1 \\ \mathsf{PRF}(\mathsf{sd}, \mathbf{r}_i) & \text{otherwise} \end{cases},$$

where $\mathsf{SKE.sk}_i$ is the i-th bit of $\mathsf{SKE.sk}$. For generating a secret key of PE for circuit C, we generate PHprFE secret keys for $C[i, \mathbf{r}_i]$ for all $i \in [\lambda]$ with freshly chosen $\mathbf{r}_i$. This allows the decryptor to recover $\mathsf{SKE.sk}$ in a bit-by-bit manner. It is not difficult to see that the construction achieves optimal parameter size of Equation (39) and still maintains the security.

[7]Here we do not give $\mathbf{x}$ as an input.

7 Compiling KP-ABE to CP-ABE using prFE

In this section we give a compiler that converts kpABE scheme for circuits of unbounded depth to a cpABE scheme for circuits of unbounded depth using a prFE scheme for pseudorandom functionality. In particular, if we start from a kpABE scheme with optimal parameter size, the resulting cpABE scheme achieves the optimal parameter size as well. By combining the kpABE and cpABE, we obtain an ABE scheme for Turing machines from LWE and evasive LWE. This improves [AKY24a] in terms of the assumption, which requires the non-standard tensor circular LWE assumption additionally.

7.1 Construction

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

1. A key-policy ABE scheme $\mathsf{kpABE} = \mathsf{kpABE}.(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{EncOff}, \mathsf{EncOn}, \mathsf{Dec})$ for circuit class $\mathcal{C}_{\ell(\lambda)} = \{C : \{0,1\}^{\ell} \to \{0,1\}\}$ consisting of circuits with input length $\ell(\lambda)$ that satisfies $\mathsf{VerSel}\text{-}\mathsf{INDr}$ reusable security (Definition 6.2). We require that the scheme satisfies optimal parameter size, namely, the size of the master public key, secret keys, and ciphertexts are all fixed polynomial. We denote the online ciphertext space of kpABE scheme by $\mathcal{CT}_{\mathsf{on}} := \{0,1\}^{\ell_{\mathsf{on}}}$, online ciphertext size by ℓ_{on}. We also assume that the randomness used by $\mathsf{kpABE}.\mathsf{EncOn}$ is of length λ without loss of generality. If it requires longer randomness, we can derive it by a PRF. We can instantiate such kpABE scheme by our construction in Section 6.1.

2. A FE scheme for pseudorandom functionality $\mathsf{prFE} = (\mathsf{prFE}.\mathsf{Setup}, \mathsf{prFE}.\mathsf{KeyGen}, \mathsf{prFE}.\mathsf{Enc}, \mathsf{prFE}.\mathsf{Dec})$ for circuit class $\mathcal{C} = \{C : \{0,1\}^{\mathrm{L}} \to \{0,1\}^{\ell_{\mathsf{on}}}\}$. Here, the depth of the circuit is unbounded, but the input and the output lengths are fixed. For our compiler, we set $\mathrm{L}(\lambda) = \mathrm{poly}(\lambda)$ for some fixed polynomial $\mathrm{poly}(\cdot)$. We require the size of the master public key and the ciphertext to be fixed polynomial in λ and the size of the secret key to be $\ell_{\mathsf{on}} \cdot \mathrm{poly}(\lambda)$. Such a construction can be obtained by applying the conversion described in Section 5.5 to our construction of PHprFE in Section 5.4. We use $\mathcal{CT}_{\mathsf{prFE}}$ to denote the ciphertext space of the scheme and $\mathsf{prm} = (1^{\mathrm{L}}, 1^{\ell_{\mathsf{on}}})$ to denote the parameters describing the circuit class.

3. A pseudorandom function $\mathsf{PRF} : \{0,1\}^{\lambda} \times \{0,1\}^{\lambda} \to \{0,1\}^{\lambda}$. It is known that PRF can be constructed from one-way functions.

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

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

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

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

- Parse $\mathsf{cpABE}.\mathsf{msk} = \mathsf{prFE}.\mathsf{msk}$ and sample $\mathbf{r} \leftarrow \{0,1\}^{\lambda}$.
- Define circuit $\mathrm{F}[\mathbf{x}, \mathbf{r}]$, with $\mathbf{x}, \mathbf{r}$ hardwired, as follows.
  On input $(\mathsf{kpABE}.\mathsf{st}, \mathsf{sd}, \mu)$:
  - Compute and output $\mathsf{kpABE}.\mathsf{ct}_{\mathsf{on}}$
    where $\mathsf{kpABE}.\mathsf{ct}_{\mathsf{on}} = \mathsf{kpABE}.\mathsf{EncOn}(\mathsf{kpABE}.\mathsf{st}, \mathbf{x}, \mu; \mathsf{PRF}(\mathsf{sd}, \mathbf{r}))$.
- Compute $\mathsf{prFE}.\mathsf{sk} \leftarrow \mathsf{prFE}.\mathsf{KeyGen}(\mathsf{prFE}.\mathsf{msk}, \mathrm{F}[\mathbf{x}, \mathbf{r}])$.
- Output $\mathsf{cpABE}.\mathsf{sk}_{\mathbf{x}} := (\mathbf{r}, \mathsf{prFE}.\mathsf{sk})$.

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

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

– Generate $(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk}) \leftarrow \mathsf{kpABE.Setup}(1^\lambda, 1^\ell)$.
– Generate $(\mathsf{kpABE.ct_{off}}, \mathsf{kpABE.st}) \leftarrow \mathsf{kpABE.EncOff}(\mathsf{kpABE.mpk})$.
– Compute $\mathsf{prFE.ct} \leftarrow \mathsf{prFE.Enc}(\mathsf{prFE.mpk}, (\mathsf{kpABE.st}, \mathsf{sd}, \mu))$.
– Compute $\mathsf{kpABE.sk}_C \leftarrow \mathsf{kpABE.KeyGen}(\mathsf{kpABE.msk}, C)$.
– Output $\mathsf{cpABE.ct} := (\mathsf{prFE.ct},\ \mathsf{kpABE.mpk},\ \mathsf{kpABE.ct_{off}},\ \mathsf{kpABE.sk}_C)$.

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

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

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

Theorem 7.1. Assume kpABE is perfectly correct. Then the above construction of cpABE scheme is correct.

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

$$\mathbf{y} = \mathrm{F}[\mathbf{x}, \mathbf{r}](\mathsf{kpABE.st}, \mathsf{sd}, \mu) = \mathsf{kpABE.ct_{on}}$$

where $\mathsf{kpABE.ct_{on}} = \mathsf{kpABE.EncOn}(\mathsf{kpABE.st}, \mathbf{x}, \mu; \mathsf{PRF}(\mathsf{sd}, \mathbf{r}))$. Next, from the correctness of kpABE, if $C(\mathbf{x}) = 1$, it follows that $\mathsf{kpABE.Dec}(\mathsf{kpABE.mpk}, \mathsf{kpABE.sk}_C, C, \mathsf{kpABE.ct}, \mathbf{x}) = \mu$, where $\mathsf{kpABE.ct} = (\mathsf{kpABE.ct_{off}}, \mathsf{kpABE.ct_{on}})$. □

Efficiency. Our cpABE scheme satisfies

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

To see the above we make the following observations:

1. Instantiating kpABE scheme as in Section 6.1, we have $|\mathsf{kpABE.mpk}| = \mathrm{poly}(\lambda)$, $|\mathsf{kpABE.sk}_C| = \mathrm{poly}(\lambda)$, $|\mathsf{kpABE.ct_{off}}| = |\mathsf{kpABE.ct_{on}}| = \mathrm{poly}(\lambda)$
2. Instantiating prFE scheme as in Theorem 5.14, we have $|\mathsf{prFE.mpk}| = \mathrm{poly}(\lambda)$, $|\mathsf{prFE.sk}| = \ell_{\mathsf{on}}\mathrm{poly}(\lambda) = \mathrm{poly}(\lambda)$, $|\mathsf{prFE.ct}| = L_{\mathsf{inp}} + \mathrm{poly}(\lambda)$, where L_{inp} is the input length of the prFE scheme. In our construction $L_{\mathsf{inp}} = |\mathsf{kpABE.st}| + |\mathsf{sd}| + |\mu| = \mathrm{poly}(\lambda)$.

Using the above we note that

– $|\mathsf{cpABE.mpk}| = |\mathsf{prFE.mpk}| = \mathrm{poly}(\lambda)$.
– $|\mathsf{cpABE.sk}_\mathbf{x}| = |\mathbf{r}| + |\mathsf{prFE.sk}| = \lambda + \mathrm{poly}(\lambda) = \mathrm{poly}(\lambda)$.
– $|\mathsf{cpABE.ct}| = |\mathsf{prFE.ct}| + |\mathsf{kpABE.mpk}| + |\mathsf{kpABE.ct_{off}}| + |\mathsf{kpABE.sk}_C| = \mathrm{poly}(\lambda)$.

We formalise the instantiation using the following theorem.

Theorem 7.2. Under the LWE and Evasive LWE assumption, there exists very selectively secure CP-ABE scheme supporting circuits with unbounded depth $\{C : \{0,1\}^\ell \to \{0,1\}\}$ and single bit message space with

$$|\mathsf{mpk}| = \mathrm{poly}(\lambda),\ |\mathsf{sk}_\mathbf{x}| = \mathrm{poly}(\lambda),\ |\mathsf{ct}| = \mathrm{poly}(\lambda)$$

where $\mathbf{x} \in \{0,1\}^\ell$.

7.2 Security

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

Theorem 7.3. If the prFE scheme is secure (Definition 3.2, with $Q_{\mathsf{msg}} = 1$), with respect to the sampler class containing the sampler Samp as defined in Eq. 41 , kpABE scheme satisfies $\mathsf{VerSel\text{-}INDr}$ reusable security (Definition 6.2) then the construction of cpABE satisfies $\mathsf{VerSel\text{-}IND}$ security.

Proof. Suppose the adversary $\mathcal{A}$ with randomness $\mathsf{coins}_{\mathcal{A}}$ queries for $C, \mu, \mathbf{x}_1, \ldots \mathbf{x}_Q$. We want to prove

$$\begin{pmatrix} \mathsf{coins}_{\mathcal{A}},\ \mathsf{cpABE.mpk} = \mathsf{prFE.mpk}, \\ \{\mathrm{F}[\mathbf{x}_k, \mathbf{r}_k]\}_{k\in[Q]},\ \mathsf{cpABE.sk}_{x_k} = \{\mathsf{prFE.sk}_k\}_{k\in[Q]}, \\ \mathsf{cpABE.ct} = (\mathsf{prFE.ct}, \mathsf{kpABE.mpk}, \mathsf{kpABE.ct}_{\mathsf{off}}, \mathsf{kpABE.sk}_C) \end{pmatrix} \approx_c \begin{pmatrix} \mathsf{coins}_{\mathcal{A}},\ \mathsf{cpABE.mpk} = \mathsf{prFE.mpk}, \\ \{\mathrm{F}[\mathbf{x}_k, \mathbf{r}_k]\}_{k\in[Q]},\ \mathsf{cpABE.sk}_{x_k} = \{\mathsf{prFE.sk}_k\}_{k\in[Q]}, \\ \mathsf{cpABE.ct} = (\Delta, \mathsf{kpABE.mpk}, \mathsf{kpABE.ct}_{\mathsf{off}}, \mathsf{kpABE.sk}_C) \end{pmatrix} \tag{40}$$

where $\Delta \leftarrow \mathcal{CT}_{\mathsf{prFE}}$, assuming we have $C(\mathbf{x}_k) = 1$ for all the key queries $\mathbf{x}_1, \ldots, \mathbf{x}_Q$ and the challenge circuit C issued by the adversary. Observe that the equation on the left is the view of the adversary in the real world and that on the right is the view of the adversary in the ideal world[8]. Here $\mathrm{F}[\mathbf{x}_k, \mathbf{r}_k]$ denotes the functions corresponding to k-th key query $\mathbf{x}_k$ as defined in the KeyGen algorithm, $\mathsf{prFE.sk}_k \leftarrow \mathsf{prFE.KeyGen}(\mathsf{prFE.msk}, \mathrm{F}[\mathbf{x}_k, \mathbf{r}_k])$ for $k \in [Q]$, $(\mathsf{kpABE.ct}_{\mathsf{off}}, \mathsf{kpABE.st}) \leftarrow \mathsf{kpABE.EncOff}(\mathsf{kpABE.mpk})$, $\mathsf{prFE.ct} \leftarrow \mathsf{prFE.Enc}(\mathsf{prFE.mpk}, (\mathsf{kpABE.st}, \mathsf{sd}, \mu))$ for $\mathsf{sd} \leftarrow \{0,1\}^{\lambda}$ and $\mathsf{kpABE.sk}_C \leftarrow \mathsf{kpABE.KeyGen}(\mathsf{kpABE.msk}, C)$.
We invoke the security of prFE with sampler Samp that outputs

$$\begin{pmatrix} \text{Functions:} & \{\mathrm{F}[\mathbf{x}_k, \mathbf{r}_k]\}_{k\in[Q]}, \\ \text{Input:} & (\mathsf{kpABE.st}, \mathsf{sd}, \mu), \\ \text{Auxiliary Information:} & \mathsf{aux} = (\{\mathbf{x}_k, \mathbf{r}_k\}_{k\in[Q]}, \mathsf{kpABE.mpk}, \mathsf{kpABE.sk}_C, \mathsf{kpABE.ct}_{\mathsf{off}}, \mathsf{coins}_{\mathcal{A}}) \end{pmatrix} \tag{41}$$

By the security guarantee of prFE with sampler Samp we have that

$$\text{if} \quad \left(\mathsf{aux}, \{\mathrm{F}[\mathbf{x}_k, \mathbf{r}_k], \mathrm{F}[\mathbf{x}_k, \mathbf{r}_k](\mathsf{kpABE.st}, \mathsf{sd}, \mu)\}_{k\in[Q]}\right) \approx_c \left(\mathsf{aux}, \{\mathrm{F}[\mathbf{x}_k, \mathbf{r}_k],\ \delta_k \leftarrow \{0,1\}^{\ell_{\mathsf{on}}}\}_{k\in[Q]}\right)$$

$$\text{then} \quad \begin{pmatrix} \mathsf{prFE.mpk},\ \mathsf{aux}, \{\mathrm{F}[\mathbf{x}_k, \mathbf{r}_k],\ \mathsf{prFE.sk}_k\}_{k\in[Q]} \\ \mathsf{prFE.ct} \leftarrow \mathsf{prFE.Enc}(\mathsf{prFE.mpk}, (\mathsf{kpABE.st}, \mathsf{sd}, \mu)) \end{pmatrix} \approx_c \begin{pmatrix} \mathsf{prFE.mpk},\ \mathsf{aux}, \{\mathrm{F}[\mathbf{x}_k, \mathbf{r}_k],\ \mathsf{prFE.sk}_k\}_{k\in[Q]} \\ \Delta \leftarrow \mathcal{CT}_{\mathsf{prFE}} \end{pmatrix},$$

where one can see that the latter equation is equivalent to Equation (40). Thus to prove Equation (40), it suffices to prove

$$\begin{aligned} &(\mathsf{aux}, \{\mathbf{x}_k, \mathbf{r}_k\}_{k\in[Q]},\ \{\mathsf{kpABE.ct}_{\mathsf{on},k} = \mathrm{F}[\mathbf{x}_k, \mathbf{r}_k](\mathsf{kpABE.st}, \mathsf{sd}, \mu)\}_{k\in[Q]},\ \mathsf{kpABE.sk}_C) \\ &\qquad \approx_c \left(\mathsf{aux}, \{\mathbf{x}_k, \mathbf{r}_k\}_{k\in[Q]},\ \{\delta_k \leftarrow \{0,1\}^{\ell_{\mathsf{on}}}\}_{k\in[Q]},\ \mathsf{kpABE.sk}_C\right). \end{aligned} \tag{42}$$

For clarity in the further steps of security proof we write the equation on L.H.S. of the above equation as

$$\begin{pmatrix} \mathsf{coins}_{\mathcal{A}},\ \{\mathbf{x}_k, \mathbf{r}_k\}_{k\in[Q]},\ \mathsf{kpABE.mpk},\ \mathsf{kpABE.sk}_C, \\ \mathsf{kpABE.ct}_{\mathsf{off}},\ \{\mathsf{kpABE.ct}_{\mathsf{on},k} := \mathsf{kpABE.EncOn}(\mathsf{kpABE.st}, \mathbf{x}_k, \mu; \mathsf{PRF}(\mathsf{sd}, \mathbf{r}_k))\}_{k\in[Q]} \end{pmatrix}, \tag{43}$$

where we unroll aux and $\mathrm{F}[\mathbf{x}_k, \mathbf{r}_k](\mathsf{kpABE.st}, \mathsf{sd}, \mu)$ based on their definitions. We prove the pseudorandomness of $\{\mathsf{kpABE.ct}_{\mathsf{on},k}\}_k$ in Equation (43) via the following sequence of hybrids.

Hyb_0. This is the distribution in Equation (43).

[8]Note that the information about μ is encoded in $\mathsf{prFE.ct}$.

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

Hyb_2. This hybrid is same as the previous hybrid except that we change all the PRF values $\{\mathsf{PRF}(\mathsf{sd}, \mathbf{r}_k)\}_k$ to truly random values $\{R_k \leftarrow \{0,1\}^\lambda\}_k$. By the change introduced in the previous hybrid, $\mathsf{PRF}(\mathsf{sd}, \cdot)$ is invoked on fresh input for each k. Therefore, we can replace $\{\mathsf{PRF}(\mathsf{sd}, \mathbf{r}_k)\}_k$ with truly random $\{R_k\}_k$ by the security of PRF without being noticed by the adversary. We now consider the following distribution:

$$\left(\begin{array}{c}\mathsf{coins}_{\mathcal{A}},\ \{\mathbf{x}_k, \mathbf{r}_k\}_{k\in[Q]},\ \mathsf{kpABE.mpk},\ \mathsf{kpABE.sk}_C, \\ \mathsf{kpABE.ct}_{\mathsf{off}},\ \{\mathsf{kpABE.ct}_{\mathsf{on},k} \leftarrow \mathsf{kpABE.EncOn}(\mathsf{kpABE.st}, \mathbf{x}_k, \mu)\}_{k\in[Q]}\end{array}\right),$$

Hyb_3. In this hybrid we invoke the reusable security of kpABE scheme to switch $\{\mathsf{kpABE.ct}_{\mathsf{on},k}\}_k$ to be random strings in $\{0,1\}^{\ell_{\mathsf{on}}}$.

We claim that an adversary $\mathcal{A}$ who can distinguish Hyb_2 and Hyb_3 can be used to break reusable security of kpABE. The reduction $\mathcal{B}$ works as follows.

- $\mathcal{A}$ sends $\mathsf{coins}_{\mathcal{A}}, C, \mathbf{x}_1, \ldots, \mathbf{x}_Q, \mu$ to the reduction.
- $\mathcal{B}$ sends $(C, \{\mathbf{x}_k\}_k, \mu)$ to the kpABE challenger. The challenger does the following.
  - Generates $(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk}) \leftarrow \mathsf{kpABE.Setup}(1^\lambda, 1^\ell)$.
  - Computes $\mathsf{kpABE.sk}_C \leftarrow \mathsf{kpABE.KeyGen}(1^\lambda, C)$ and $(\mathsf{kpABE.ct}_{\mathsf{off}}, \mathsf{kpABE.st}) \leftarrow \mathsf{kpABE.EncOff}(\mathsf{kpABE.mpk})$.
  - Computes $\mathsf{kpABE.ct}^0_{\mathsf{on},k} \leftarrow \mathsf{kpABE.Enc}(\mathsf{kpABE.st}, \mathbf{x}_k, \mu)$ and $\mathsf{kpABE.ct}^1_{\mathsf{on},k} \leftarrow \{0,1\}^{\ell_{\mathsf{on}}}$ for all $k \in [Q]$.
  - Samples a bit $\beta \leftarrow \{0,1\}$ and returns $(\mathsf{kpABE.mpk}, \mathsf{kpABE.sk}_C, \mathsf{kpABE.ct}_{\mathsf{off}}, \{\mathsf{kpABE.ct}^\beta_{\mathsf{on},k}\}_{k\in[Q]})$ to $\mathcal{B}$
- $\mathcal{B}$ returns $\left(\mathsf{coins}_{\mathcal{A}}, \{\mathbf{x}_k, \mathbf{r}_k\}_{k\in[Q]}, \mathsf{kpABE.mpk}, \mathsf{kpABE.sk}_C, \mathsf{kpABE.ct}_{\mathsf{off}}, \{\mathsf{kpABE.ct}^\beta_{\mathsf{on},k}\}_{k\in[Q]}\right)$ to $\mathcal{A}$.
- $\mathcal{A}$ outputs a guess bit β'. $\mathcal{B}$ outputs the same bit as its guess.

We note that if the challenger samples $\beta = 0$, then $\mathcal{B}$ simulates Hyb_2 with adversary else it simulates Hyb_3.

Admissibility of $\mathcal{B}$. Observe that by the admissibility of cpABE, $\mathcal{A}$ sends challenge queries $C, \mathbf{x}_1, \ldots, \mathbf{x}_Q, \mu$ such that $C(\mathbf{x}_k) = 0$ for all $k \in [Q]$. Thus the query $(C, \{\mathbf{x}_k\}_k, \mu)$ sent by $\mathcal{B}$ to the kpABE challenger satisfies $C(\mathbf{x}_k) = 0$ for all $k \in [Q]$. This establishes the admissibility of $\mathcal{B}$.

We observe that the view of the adversary in Hyb_3 is the same as the R.H.S of Equation (42) and hence the proof. □

Implications to ABE for Turing Machines. We note that our unbounded CP-ABE scheme Theorem 7.2 and unbounded KP-ABE scheme Theorem 6.1 can be used to instantiate ABE for Turing Machines ([AKY24a]).

Corollary 7.4. Under the LWE and evasive LWE assumptions, there exists a very selectively secure ABE for TM with

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

where the Turing machine M runs on input x for time step t.

[AKY24a] uses the LWE , evasive LWE and circular tensor LWE assumptions for their construction with $|\mathsf{mpk}| = \mathrm{poly}(\lambda)$, $|\mathsf{sk}| = \mathrm{poly}(|M|, \lambda)$, $|\mathsf{ct}| = \mathrm{poly}(\lambda, |\mathbf{x}|, t)$.

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, Berlin, Heidelberg, May / June 2010. (Cited on page 22.)

[ABB10b] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical IBE. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 98–115. Springer, Berlin, Heidelberg, August 2010. (Cited on page 22.)

[Agr19] Shweta Agrawal. Indistinguishability obfuscation without multilinear maps: New techniques for bootstrapping and instantiation. In *Eurocrypt*, 2019. (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, Berlin, Heidelberg, August 2015. (Cited on page 5, 16.)

[AJS23] Paul Lou Aayush Jain, Huijia Lin and Amit Sahai. Polynomial-time cryptanalysis of the subspace flooding assumption for post-quantum io. In *Eurocrypt*, 2023. (Cited on page 3.)

[AKY24a] Shweta Agrawal, Simran Kumari, and Shota Yamada. Attribute Based Encryption for Turing Machines from Lattices. In *Crypto*, 2024. (Cited on page 4, 5, 8, 9, 11, 70, 73.)

[AKY24b] Shweta Agrawal, Simran Kumari, and Shota Yamada. Pseudorandom Multi-Input Functional Encryption and Applications, 2024. (Cited on page 5, 16, 17, 21, 30, 31, 83.)

[AMY19] Shweta Agrawal, Monosij Maitra, and Shota Yamada. Attribute based encryption for deterministic finite automata from DLIN. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part II*, volume 11892 of *LNCS*, pages 91–117. Springer, Cham, December 2019. (Cited on page 7.)

[AMYY25] Shweta Agrawal, Anuja Modi, Anshu Yadav, and Shota Yamada. Evasive lwe: Attacks, variants & obfustopia, 2025. (Cited on page 6, 7, 11, 13, 23, 31, 32, 41, 48.)

[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 Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part IV*, volume 14084 of *LNCS*, pages 532–564. Springer, Cham, August 2023. (Cited on page 23, 24, 58.)

[AY20] Shweta Agrawal and Shota Yamada. CP-ABE for circuits (and more) in the symmetric key setting. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part I*, volume 12550 of *LNCS*, pages 117–148. Springer, Cham, November 2020. (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 7, 31, 48.)

[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, Berlin, Heidelberg, May 2014. (Cited on page 3, 7.)

[BGI$^+$01a] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. In *CRYPTO*, 2001. (Cited on page 3.)

[BGI$^+$01b] 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, Berlin, Heidelberg, August 2001. (Cited on page 7.)

[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 8.)

[BLP$^+$13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 575–584. ACM Press, June 2013. (Cited on page 22, 23.)

[BLSV18] Zvika Brakerski, Alex Lombardi, Gil Segev, and Vinod Vaikuntanathan. Anonymous IBE, leakage resilience and circular security from new assumptions. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 535–564. Springer, Cham, April / May 2018. (Cited on page 14, 17, 28, 29, 30, 82, 83.)

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *Theory of Cryptography: 8th Theory of Cryptography Conference, TCC 2011, Providence, RI, USA, March 28-30, 2011. Proceedings 8*, pages 253–273. Springer, 2011. (Cited on page 3, 11.)

[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, Cham, November 2017. (Cited on page 8, 25.)

[BÜW24] Chris Brzuska, Akin Ünal, and Ivy KY Woo. Evasive lwe assumptions: Definitions, classes, and counterexamples. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 418–449. Springer, 2024. (Cited on page 7.)

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 505–524. Springer, Berlin, Heidelberg, August 2011. (Cited on page 8.)

[BV18] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. *Journal of the ACM*, 65(6):39:1–39:37, 2018. (Cited on page 16.)

[CDG$^+$17] Chongwon Cho, Nico Döttling, Sanjam Garg, Divya Gupta, Peihan Miao, and Antigoni Polychroniadou. Laconic oblivious transfer and its applications. In *Annual International Cryptology Conference*, pages 33–65. Springer, 2017. (Cited on page 17.)

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, 2004. (Cited on page 7.)

[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, Berlin, Heidelberg, May / June 2010. (Cited on page 22.)

[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, Berlin, Heidelberg, February 2010. (Cited on page 7.)

[DGHM18] Nico Döttling, Sanjam Garg, Mohammad Hajiabadi, and Daniel Masny. New constructions of identity-based and key-dependent message secure encryption schemes. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part I*, volume 10769 of *LNCS*, pages 3–31. Springer, Cham, March 2018. (Cited on page 79.)

[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.)

[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.)

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 169–178, 2009. (Cited on page 8.)

[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 *FOCS*, 2013. `http://eprint.iacr.org/`. (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 32.)

[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.)

[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.)

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 89–98. ACM Press, October / November 2006. Available as Cryptology ePrint Archive Report 2006/309. (Cited on page 3.)

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008. (Cited on page 22.)

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 75–92. Springer, Berlin, Heidelberg, August 2013. (Cited on page 8, 24.)

[GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In *STOC*, 2013. (Cited on page 3.)

[GWW19] Junqing Gong, Brent Waters, and Hoeteck Wee. Abe for dfa from k-lin. In *Advances in Cryptology–CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2019, Proceedings, Part II 39*, pages 732–764. Springer, 2019. (Cited on page 7.)

[GZ21] Alonso González and Alexandros Zacharakis. Fully-succinct publicly verifiable delegation from constant-size assumptions. In *Theory of Cryptography Conference*, pages 529–557. Springer, 2021. (Cited on page 7.)

[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, 11.)

[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 2, 3, 4, 6, 8, 11, 13, 14, 24, 25, 41, 42, 83, 89, 90.)

[HLL24] Yao-Ching Hsieh, Huijia Lin, and Ji Luo. A general framework for lattice-based ABE using evasive inner-product functional encryption. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part II*, volume 14652 of *LNCS*, pages 433–464. Springer, Cham, May 2024. (Cited on page 4.)

[JLL23] Aayush Jain, Huijia Lin, and Ji Luo. On the optimal succinctness and efficiency of functional encryption and attribute-based encryption. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part III*, volume 14006 of *LNCS*, pages 479–510. Springer, Cham, April 2023. (Cited on page 3, 4, 16.)

[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 $\mathbb{F}_p$, DLIN, and PRGs in NC^0. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part I*, volume 13275 of *LNCS*, pages 670–699. Springer, Cham, May / June 2022. (Cited on page 3.)

[KLVW23] Yael Kalai, Alex Lombardi, Vinod Vaikuntanathan, and Daniel Wichs. Boosting batch arguments and ram delegation. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pages 1545–1552, 2023. (Cited on page 7.)

[KSW13] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. *Journal of cryptology*, 26:191–224, 2013. (Cited on page 3.)

[LPST16] Huijia Lin, Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation with non-trivial efficiency. In *Public-Key Cryptography–PKC 2016*, pages 447–462. Springer, 2016. (Cited on page 16.)

[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, Berlin, Heidelberg, April 2012. (Cited on page 22.)

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

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

[QWW18] Willy Quach, Hoeteck Wee, and Daniel Wichs. Laconic function evaluation and applications. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 859–870. IEEE, 2018. (Cited on page 14.)

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM*, 56(6):34:1–34:40, 2009. (Cited on page 23.)

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

[SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, 2005. (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, Cham, August 2022. (Cited on page 3, 6, 8.)

[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, Cham, December 2022. (Cited on page 6.)

[Wat12] Brent Waters. Functional encryption for regular languages. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 218–235. Springer, Berlin, Heidelberg, August 2012. (Cited on page 7.)

[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 7.)

[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, Cham, May / June 2022. (Cited on page 3, 6, 8, 11, 23.)

[Wee24] Hoeteck Wee. Circuit abe with poly(depth, λ)-sized ciphertexts and keys from lattices. In *Crypto*, 2024. (Cited on page 3, 4, 6, 16.)

[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, Cham, November 2022. (Cited on page 23.)

A Blind Batch Encryption from LWE

In this section, we construct a blind batch encryption scheme from LWE that is required for the construction of the laconic pseudorandom poly-domain obfuscation scheme in Section 4.

A.1 Basic Scheme from LWE

Here, we provide the construction of basic blind batch encryption under LWE assumption. The construction is a slight variant of the hash encryption scheme proposed in [DGHM18], where we modify the construction so that it has perfect correctness and satisfies our notion of strong blindness (Definition 2.27). The construction here does not satisfy the efficiency requirement required in Section 4. However, we can bootstrap the construction to satisfy the requirement using the conversion in Appendix A.2.

In the following, we use the rounding function $\lfloor\cdot\rceil_2 : \mathbb{Z}_q \to \mathbb{Z}_2$ defined as $\lfloor x\rceil_2 \stackrel{\mathrm{def}}{=} \left\lfloor \frac{2}{q}\cdot x \right\rceil \mod 2$.

$\mathsf{Setup}(1^\lambda, 1^N)$. The setup algorithm does the following.

- Sample $\mathbf{u}_{j,b} \leftarrow \mathbb{Z}_q^n$ for $j \in [N]$ and $b \in \{0,1\}$.
- Output $\mathsf{crs} := \{\mathbf{u}_{j,b}\}_{j\in[N],b\in\{0,1\}}$.

$\mathsf{Gen}(\mathsf{crs}, X \in \{0,1\}^N)$. The generation algorithm does the following.

- Parse $\mathsf{crs} = \{\mathbf{u}_{j,b}\}_{j\in[N],b\in\{0,1\}}$.
- Compute $\mathbf{h} := \sum_{j\in[N]} \mathbf{u}_{j,X_j} \in \mathbb{Z}_q^n$ where $X_j \in \{0,1\}$ is the j-th bit of X.
- Output $\mathbf{h}$.

$\mathsf{SingleEnc}(\mathsf{crs}, \mathbf{h}, i, (\mu_0, \mu_1))$. The single encryption algorithm, for $i \in [N]$ and messages $(\mu_0, \mu_1) \in \{0,1\}^2$, does the following.

- Parse $\mathsf{crs} = \{\mathbf{u}_{j,b}\}_{j\in[N],b\in\{0,1\}}$.
- Sample $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, $e_{j,b} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}$ for $j \in [N]\backslash\{i\}$ and $b \in \{0,1\}$, and $e'_{i,0}, e'_{i,1} \leftarrow \mathcal{D}_{\mathbb{Z},\gamma}$.
- Compute $c_{j,b} := \mathbf{s}^\top \mathbf{u}_{j,b} + e_{j,b}$ for $j \in [N]\backslash\{i\}$ and $b \in \{0,1\}$.
- Compute
$$c_{i,b} := \left\lfloor \mathbf{s}^\top(\mathbf{h} - \mathbf{u}_{i,b}) + e'_{i,b} \right\rceil_2 \oplus \mu_b$$
for $b \in \{0,1\}$.
- Check whether $\mathbf{s}^\top(\mathbf{h} - \mathbf{u}_{i,b}) \in [q/4 - B, q/4 + B] \cup [3q/4 - B, 3q/4 + B]$ holds for $b = 0$ or $b = 1$. If so, replace each of $\{c_{j,b}\}_{j\in[N],b\in\{0,1\}}$ with $\perp$ and set $c_{i,b} = \mu_b$ for $b \in \{0,1\}$ and some $B > 0$[9]. Otherwise, do nothing.
- Output $\mathsf{ct} := \{c_{j,b}\}_{j\in[N],b\in\{0,1\}}$.

$\mathsf{SingleDec}(\mathsf{crs}, X, i, \mathsf{ct})$. The single decryption algorithm, for $X \in \{0,1\}^N$ and $i \in [N]$, does the following.

- Parse $\mathsf{crs} = \{\mathbf{u}_{j,b}\}_{j\in[N],b\in\{0,1\}}$, $\mathsf{ct} \to (\{c_{j,b}\}_{j\in[N],b\in\{0,1\}})$ where $c_{j,b} \in \mathbb{Z}_q \cup \{\perp\}$ for $j \in [N]\backslash\{i\}$ and $c_{i,b} \in \{0,1\}$ for $b \in \{0,1\}$.
- If $c_{j,b} = \perp$ for some j, b, set $\mu' := c_{i,X_i}$.
- Otherwise, compute
$$\mu' := c_{i,X_i} \oplus \left\lfloor \sum_{j\in[N]\backslash\{i\}} c_{j,X_j} \right\rceil_2 .$$

[9]We introduce this step to remove the correctness error.

– Output μ'.

Remark A.1. We note that the crs in the above scheme is a uniformly random bit string.

Parameter. We set q to be a power of 2, so that a random string over $\mathbb{Z}_q$ can be interpreted as a binary random string. We set γ super-polynomially larger than σ so that the smudging is possible.

$$q = 2^{5\lambda}, \;\; B = 2^{3\lambda}, \;\; ,\sigma = 2^{2\lambda}/N, \;\; ,\gamma = 2^{2\lambda}\lambda^{\omega(1)}$$

Succinctness. We can see that the hash is of fixed length and thus the construction is fully succinct.

Correctness. With the above set parameters, we show that our scheme achieves perfect correctness.

– If $c_{j,b} = \perp$ for some $j \in [N]\backslash\{i\}$ and $b \in \{0,1\}$, then $\mu' := c_{i,X_i} = \mu_{X_i}$ with probability 1 and hence the correctness.

– If $c_{j,b} \neq \perp$ for any $j \in [N]\backslash\{i\}$ and $b \in \{0,1\}$, we compute

$$\begin{aligned}
\mu' &= c_{i,X_i} \oplus \left\lfloor \sum_{j\in[N]\backslash\{i\}} c_{j,X_j} \right\rceil_2 \\
&= \left\lfloor \mathbf{s}^\top (\mathbf{h} - \mathbf{u}_{i,X_i}) + e'_{i,X_i} \right\rceil_2 \oplus \mu_{X_i} \oplus \left\lfloor \sum_{j\in[N]\backslash\{i\}} \mathbf{s}^\top \mathbf{u}_{j,X_j} + e_{j,X_j} \right\rceil_2 \\
&= \left\lfloor \mathbf{s}^\top \left(\sum_{j\in[N]} \mathbf{u}_{j,X_j} - \mathbf{u}_{i,X_i} \right) + e'_{i,X_i} \right\rceil_2 \oplus \mu_{X_i} \oplus \left\lfloor \sum_{j\in[N]\backslash\{i\}} \mathbf{s}^\top \mathbf{u}_{j,X_j} + e_{j,X_j} \right\rceil_2 \\
&= \left\lfloor \sum_{j\in[N]\backslash\{i\}} \mathbf{s}^\top \mathbf{u}_{j,X_j} + e'_{i,X_i} \right\rceil_2 \oplus \mu_{X_i} \oplus \left\lfloor \sum_{j\in[N]\backslash\{i\}} \mathbf{s}^\top \mathbf{u}_{j,X_j} + e_{j,X_j} \right\rceil_2 \\
&= \mu_{X_i}
\end{aligned}$$

where the last equality follows from our parameter setting, $|e'_{i,X_i}| < B$ and $\sum_{j\in[N]\backslash\{i\}} |e_{j,X_j}| \leq B$ and the guarantee that $\sum_{j\in[N]\backslash\{i\}} \mathbf{s}^\top \mathbf{u}_{j,X_j}$ is sufficiently far, from the second to the last step of the encryption algorithm, from the "unsafe zone" where small noise can change the rounded value (i.e., $[q/4 - B, q/4 + B] \cup [3q/4 - B, 3q/4 + B]$).

Security. Next, we prove the security of our scheme.

Theorem A.2. The above construction satisfies SingleEnc security (Definition 2.26) under the LWE assumption.

Proof. We consider the following sequence of hybrids.

Hyb_0. Real game.

Hyb_1. Change the encryption algorithm so that it never sets $c_{j,b} = \perp$ (i.e., we erase the branch of the computation introduced to eliminate the possibility of the decryption error). Since $\mathbf{s}^\top(\mathbf{h} - \mathbf{u}_{i,0})$ and $\mathbf{s}^\top(\mathbf{h} - \mathbf{u}_{i,1})$ are distributed uniformly at random over $\mathbb{Z}_q$, the probability of $\perp$ being output is bounded by $4B/q$. Therefore, this hybrid is statistically indistinguishable from the previous one.

Hyb_2. In this hybrid we compute c_{i,X_i} as

$$c_{i,X_i} = \left\lfloor \sum_{j\in[N]\setminus\{i\}} c_{j,X_j} + e'_{i,X_i} \right\rceil_2 \oplus \mu_{X_i}.$$

By the smudging lemma, this hybrid is statistically close to the previous hybrid. To see this, we note the following.

– In Hyb_1 we have

$$c_{i,X_i} := \left\lfloor \mathbf{s}^\top \left(\mathbf{h} - \mathbf{u}_{i,X_i}\right) + e'_{i,X_i} \right\rceil_2 \oplus \mu_{X_i}$$

– Substituting $\mathbf{h} = \sum_{j\in[N]} \mathbf{u}_{j,X_j}$, we get

$$c_{i,X_i} = \left\lfloor \mathbf{s}^\top \sum_{j\in[N]\setminus\{i\}} \mathbf{u}_{j,X_j} + e'_{i,X_i} \right\rceil_2 \oplus \mu_{X_i}. \tag{44}$$

– From our parameter setting we have $|\sum_{j\in[N]\setminus\{i\}} e_{j,X_j}| \le \lambda^{\omega(1)} \sum_{j\in[N]\setminus\{i\}} |e_{j,X_j}| \le |e'_{i,X_i}|$ where $e_{j,b} \in \mathcal{D}_{\mathbb{Z},\sigma}$ and $e'_{i,X_i} \in \mathcal{D}_{\mathbb{Z},\gamma}$. Thus using noise flooding (Lemma 2.4) and Equation (44) we have

$$\begin{aligned} c_{i,X_i} &= \left\lfloor \mathbf{s}^\top \sum_{j\in[N]\setminus\{i\}} \mathbf{u}_{j,X_j} + \sum_{j\in[N]\setminus\{i\}} e_{j,X_j} + e'_{i,X_i} \right\rceil_2 \oplus \mu_{X_i} \\ &= \left\lfloor \sum_{j\in[N]\setminus\{i\}} \left(\mathbf{s}^\top \mathbf{u}_{j,X_j} + e_{j,X_j}\right) + e'_{i,X_i} \right\rceil_2 \oplus \mu_{X_i} \\ &= \left\lfloor \sum_{j\in[N]\setminus\{i\}} c_{j,X_j} + e'_{i,X_i} \right\rceil_2 \oplus \mu_{X_i} \end{aligned}$$

with overwhelming probability.

Hyb_3. To compute $c_{i,1-X_i}$, we set it as

$$c_{i,1-X_i} = \left\lfloor \mathbf{s}^\top \left(\mathbf{h} - \mathbf{u}_{i,1-X_i}\right) + e_{i,1-X_i} + e'_{i,1-X_i} \right\rceil_2 \oplus \mu_{1-X_i},$$

where $e_{i,1-X_i} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}$. By the smudging lemma, this hybrid is statistically close to the previous hybrid. To see this, we note that initially we have

$$c_{i,1-X_i} = \left\lfloor \mathbf{s}^\top \left(\mathbf{h} - \mathbf{u}_{i,1-X_i}\right) + e'_{i,1-X_i} \right\rceil_2 \oplus \mu_{1-X_i} \tag{45}$$

where $e'_{i,1-X_i} \in \mathcal{D}_{\mathbb{Z},\gamma}$. Next, from our parameter setting, we have $\lambda^{\omega(1)}|e_{i,1-X_i}| \le |e'_{i,1-X_i}|$ where $e_{i,1-X_i} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}$. Thus using noise flooding (Lemma 2.4), we can write Equation (45) as

$$c_{i,1-X_i} = \left\lfloor \mathbf{s}^\top \left(\mathbf{h} - \mathbf{u}_{i,1-X_i}\right) + e_{i,1-X_i} + e'_{i,1-X_i} \right\rceil_2 \oplus \mu_{1-X_i}.$$

Hyb_4. Replace $\{c_{j,b}\}_{j\ne i, b\in\{0,1\}}$ and $\mathbf{s}^\top \left(\mathbf{h} - \mathbf{u}_{i,1-X_i}\right) + e_{i,1-X_i}$ computed for $c_{i,1-X_i}$ with random elements in $\mathbb{Z}_q$. This game is computationally indistinguishable from the previous one by LWE.

Hyb_5. Now, $c_{i,1-X_i}$ is sampled as $c_{i,1-X_i} \leftarrow \{0,1\}$. This game is statistically close to the previous game, since the value inside the round function is already uniformly random over $\mathbb{Z}_q$.

□

Theorem A.3. The above construction satisfies strong blindness as per Definition 2.27 under the LWE assumption.

Proof. The proof is almost the same as that of Theorem A.2, where we consider the same sequence of the hybrids. We can skip Hyb_2, since c_{i,X_i} is already random by the definition of the blindness security game. In Hyb_5, the ciphertext is a random string. Therefore, we can conclude the proof of the theorem. □

A.2 Bootstrapping the Basic Scheme

Our construction of BBE in Appendix A.1 has large CRS size and single ciphertext size, both of which linearly depend on N. However, we require a BBE scheme whose sizes of these parameters are independent of N in Section 4. Here, we obtain a scheme with the required properties from LWE by applying the conversion from [BLSV18] to our construction in Appendix A.1.

Theorem A.4 (Adapted from Appendix A.2 of [BLSV18]). Assuming that there exists a $1/2$-succinct BBE scheme (defined in Definition 2.25) whose CRS size is $\mathrm{poly}(\lambda, N)$ and single-ciphertext size is $\mathrm{poly}(\lambda, N)$. Then, the construction can be converted into a fully-succinct BBE scheme with CRS size $\mathrm{poly}(\lambda, \log N)$ and single-ciphertext size $\mathrm{poly}(\lambda, \log N)$. Furthermore, the conversion preserves strong blindness property.

Sketch of proof. Since the above theorem is not explicitly shown in [BLSV18], we provide an explanation on how to extract the above theorem from their result. There, they show a construction of fully-succinct BBE scheme starting from a $1/2$-succinct BBE scheme. They only provide the description of the encryption algorithm, but it is easy to extract a description of single encryption algorithm from it. For the reference for the readers, we sketch their construction and explain how to obtain the single encryption algorithm out of it.

To setup the system, they generate $d = \log(N/\lambda)$ number of CRSes, all of which are generated by $\mathsf{Setup}(1^\lambda, 1^{2\lambda})$. Each CRS string is assigned to each layer of the tree. To compute a hash value on input $X \in \{0,1\}^N$, they consider a Merkle tree of depth d. The leaves of the tree consist of N/λ nodes and X is evenly split and assigned to the corresponding node. Then, we assign hash values to the internal nodes of the tree starting from the layer of the tree right above the leaves to the root. To define a hash value h_v associated to a node v, we hash the values associated with its children, namely, $h_{v\|0}$ and $h_{v\|1}$, where we use the CRS corresponding to that layer. The final output of the hash (i.e., $\mathsf{Gen}(\mathsf{crs}, X)$) is the hash value h_ϵ assigned to the root node ϵ.

We then explain how the encryption algorithm works. For each node v, they generate subct_1 part of the underlying BBE ciphertext and a garbled circuit.[10] We denote the former by $\mathsf{subct}_{v,1}$ and the latter $\tilde{C}_v$. For the garbled circuit corresponding to the root node, we additionally provide the input labels $\mathsf{lab}_{h_\epsilon}$ that corresponds to h_ϵ. The garbled circuits associated with internal nodes are obtained by garbling a circuit that takes as input h_v and outputs subct_2 part of the BBE ciphertext that encrypts the labels of the garbled circuits corresponding to its children under the public key h_v. For the leaf nodes, the garbled circuits encrypt the messages instead of the labels.

To decrypt a ciphertext, for each leaf v, we traverse the hash tree from the root to v and obtain the message corresponding to X_v as follows. We first obtain $\mathsf{subct}_{\epsilon,2}$ by evaluating $\tilde{C}_\epsilon$ on labels $\mathsf{lab}_{h_\epsilon}$. Then, combined with $\mathsf{subct}_{\epsilon,1}$, this recovers the entire BBE ciphertext corresponding to the root node ϵ that encrypts the labels of the garbled circuits of the next layer. This BBE ciphertext can be decrypted by using the string that concatenates h_0 and h_1. This in particular gives the labels $\mathsf{lab}_{h_{v_1}}$ corresponding to h_{v_1}, where v_1 is the first bit of v. This then allows us to recover $\mathsf{subct}_{v_1,2}$ by evaluating the garbled circuit. We traverse down the tree in this way until we reach at the leaf node v, where we recover the corresponding message.

We then explain how we define the single encryption algorithm. To encrypt a message for a position i, we consider the leaf node v corresponding to the index i. We then run the encryption algorithm and remove the ciphertext components that are not necessary for traversing down the tree to the leaf node v. Namely, we only include $\mathsf{lab}_{h_\epsilon}$ and $\mathsf{subct}_{w,1}, \tilde{C}_w$ for all w that is an ancestor of v in the ciphertext. We can see that these components are sufficient for the decryption to work correctly.

Now, we can see that the construction achieves CRS size of $\mathrm{poly}(\lambda, \log N)$, since there are $d = \log(N/\lambda)$ number of CRSes of the base BBE. Similarly, the single-ciphertext size is of $\mathrm{poly}(\lambda, \log N)$, since there are d number of garbled

[10] We refer to Remark 2.28 for the explanation on subct_1 and subct_2.

circuits and $\mathsf{subct}_{v,1}$, each of which is of fixed polynomial size. We also observe that the conversion preserves the strong blindness. In [BLSV18], they show that the ciphertext is pseudorandom except for $\{\mathsf{subct}_{w,1}\}_{w:w \text{ is an ancestor of } v}$. If the underlying BBE satisfies strong blindness, $\mathsf{subct}_{w,1}$ is an empty string for all w. This indicates that the entire ciphertext of the final scheme is pseudorandom, as desired. This holds even if we consider single-ciphertext, since the single ciphertext is obtained by removing some part of the ciphertext. □

By applying the conversion to our construction in Appendix A.1, we obtain the following theorem.

Theorem A.5. There exists a fully-succinct BBE scheme with CRS size of $\mathrm{poly}(\lambda, \log N)$ and single-ciphertext size of $\mathrm{poly}(\lambda, \log N)$ from LWE where N denotes the length of the keys supported by the scheme

B Bootstrapping AB-LFE to KP-ABE with unbounded depth using prFE

In this section, we show a formal description of the construction of kpABE for unbounded circuits using 1ABE sketched in Section 1.4. This provides an alternative pathway to obtain kpABE for unbounded depth circuits different from that given in Section 6. Instead of 1ABE, our construction is formally described using abLFE, but this turns out to be almost equivalent, as discussed later in this section. We can instantiate the abLFE by the recent construction by [HLL23] or blind garbled circuit [BLSV18]. The former instantiation leads to more efficient construction than the latter, but it introduces an additional assumption of circular LWE in addition to LWE and evasive LWE. Compared with Section 6, the constructions obtained here are simpler, though their parameter sizes are sub-optimal. Importantly, we do not rely on the result regarding pPRIO from our companion paper [AKY24b] in this section.

B.1 Attribute Based Laconic Functional Encryption

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

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

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

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

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

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

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

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

Definition B.2 (Pseudorandom Ciphertext Security). For a abLFE scheme and an adversary $\mathcal{A}$, we define the experiment for security $\mathsf{Expt}^{\mathsf{abLFE}}_{\beta,\mathcal{A}}(1^\lambda)$ as follows.

1. Run $\mathcal{A}$ to receive circuit parameters prm. Run $\mathsf{crs} \leftarrow \mathsf{crsGen}(1^\lambda, \mathsf{prm})$ and send crs to $\mathcal{A}$.

2. $\mathcal{A}$ chooses $C \in \mathcal{C}_{\mathsf{prm}}$, $\mathbf{x} \in \mathcal{X}_{\mathsf{prm}}$ and $\mu \in \mathcal{M}$. Run $\mathsf{digest} = \mathsf{Compress}(\mathsf{crs}, C)$, sample $\beta \leftarrow \{0,1\}$. If $\beta = 0$, it computes $\mathsf{ct}_0 \leftarrow \mathsf{Enc}(\mathsf{crs}, \mathsf{digest}, (\mathbf{x}, \mu))$ else if $\beta = 1$, it computes $\mathsf{ct}_1 \leftarrow \mathcal{CT}_{\mathsf{abLFE}}$, where $\mathcal{CT}_{\mathsf{abLFE}}$ is the ciphertext space of abLFE. It sends $\mathsf{digest}, \mathsf{ct}_\beta$ to $\mathcal{A}$.

3. $\mathcal{A}$ outputs a guess bit β' as the output of the experiment.

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

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

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

Definition B.3 (Decomposability). We say that a abLFE scheme for a circuit class $\{\mathcal{C}_{\mathsf{prm}} = \{C : \mathcal{X}_{\mathsf{prm}} \to \{0,1\}\}\}_{\mathsf{prm}}$ satisfies decomposability if for any $\mathsf{crs} \leftarrow \mathsf{crsGen}(1^\lambda, \mathsf{prm})$ and $\mathsf{digest} \leftarrow \mathsf{Compress}(\mathsf{crs}, C)$, we have $\mathsf{digest} = \{\mathsf{digest}_i\}_{i \in [q_C]}$ for some polynomial q_C, which may depend on C, and $\mathsf{size}(\mathsf{digest}_i) \leq \mathrm{poly}(\lambda)$. Furthermore, we have that $\mathsf{Enc}(\mathsf{crs}, \mathsf{digest}, (\mathbf{x}, \mu)) = \{\mathsf{Enc}_i(\mathsf{crs}, \mathsf{digest}_i, (\mathbf{x}, \mu))\}_{i \in [q_C]}$ where size of the encryption circuit $\mathsf{size}(\mathsf{Enc}_i(\cdot, \cdot, (\cdot, \cdot))) \leq \mathrm{poly}(\lambda, |\mathbf{x}|)$.

Remark B.4. Here, we do not require the digest to be much smaller than the circuit description C, unlike the usual convention in the context of abLFE. This relaxation allows us to instantiate abLFE using blind garbled circuits, which do not have compact digests.

B.2 Construction of kpABE with Unbounded Depth

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

1. An attribute-based laconic function evaluation scheme $\mathsf{abLFE} = \mathsf{abLFE}.(\mathsf{crsGen}, \mathsf{Compress}, \mathsf{Enc}, \mathsf{Dec})$ for circuit class $\mathcal{C}_{\ell(\lambda)}$, consisting of circuits with input length $\ell(\lambda)$ and with unbounded depth and size. We let $\ell_{\mathsf{ct}}^{\mathsf{abLFE}}$ and $\mathcal{CT}_{\mathsf{abLFE}} = \{0,1\}^{\ell_{\mathsf{ct}}^{\mathsf{abLFE}}}$ denote the ciphertext length and the ciphertext space of the scheme, respectively. We assume that the abLFE scheme is decomposable (Definition B.3), i.e., we have $\mathsf{abLFE}.\mathsf{Enc} = \{\mathsf{abLFE}.\mathsf{Enc}_i\}_{i \in [q_C]}$ for some q_C and use d_{abLFE} to denote the maximum depth of a circuit required to compute $\{\mathsf{abLFE}.\mathsf{Enc}_i\}_{i \in [q_C]}$.

2. A FE scheme for pseudorandom functionality $\mathsf{prFE} = (\mathsf{prFE}.\mathsf{Setup}, \mathsf{prFE}.\mathsf{KeyGen}, \mathsf{prFE}.\mathsf{Enc}, \mathsf{prFE}.\mathsf{Dec})$ for circuit class $\mathcal{C}_{\mathrm{L}(\lambda), d_{\mathsf{prFE}}(\lambda), \ell_{\mathsf{ct}}^{\mathsf{abLFE}}}$ consisting of circuits with input length $\mathrm{L}(\ell, \lambda)$, maximum depth $d_{\mathsf{prFE}}(\lambda)$ and output length $\ell_{\mathsf{ct}}^{\mathsf{abLFE}}$. We denote by prm the parameters $(1^{\mathrm{L}(\lambda)}, 1^{d_{\mathsf{prFE}}(\lambda)}, 1^{\ell_{\mathsf{ct}}^{\mathsf{abLFE}}})$ that specifies the function class being supported. We also denote the ciphertext space of the scheme by $\mathcal{CT}_{\mathsf{prFE}}$.

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

We assume that uniform sampling from the ciphertext space is possible without any parameter other than the security parameter λ.

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

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

- Run $(\mathsf{prFE}.\mathsf{msk}, \mathsf{prFE}.\mathsf{mpk}) \leftarrow \mathsf{prFE}.\mathsf{Setup}(1^\lambda, \mathsf{prm})$ and $\mathsf{crs} \leftarrow \mathsf{abLFE}.\mathsf{crsGen}(1^\lambda)$.

– Set $\mathsf{msk} = \mathsf{prFE.msk}$[11] and $\mathsf{mpk} = (\mathsf{prFE.mpk}, \mathsf{crs})$. Output $(\mathsf{msk}, \mathsf{mpk})$.

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

- Parse $\mathsf{msk} = \mathsf{prFE.msk}$ and sample $\mathbf{r} \leftarrow \{0,1\}^\lambda$.
- Compute $\mathsf{digest} = \mathsf{abLFE.Compress}(\mathsf{crs}, C)$. Parse $\mathsf{digest} = \{\mathsf{digest}_i\}_{i\in[q_C]}$.
- For $i \in [q_C]$, define circuit $\mathrm{F}[\mathsf{crs}, \mathsf{digest}_i, \mathbf{r}]$, with $\mathsf{crs}, \mathsf{digest}_i, \mathbf{r}$ hardwired, as follows
  On input $(\mathsf{sd}, \mathbf{x}, \mu)$:
  - Compute $\mathsf{abLFE.ct}_i := \mathsf{abLFE.Enc}_i(\mathsf{crs}, \mathsf{digest}_i, (\mathbf{x}, \mu); \mathsf{PRF}(\mathsf{sd}, \mathbf{r}))$.
  - Output $\mathsf{abLFE.ct}_i$.
- For $i \in [q_C]$ compute $\mathsf{prFE.sk}_i \leftarrow \mathsf{prFE.KeyGen}(\mathsf{prFE.msk}, \mathrm{F}[\mathsf{crs}, \mathsf{digest}_i, \mathbf{r}])$.
- Output $\mathsf{sk}_C := \Big(\{\mathsf{digest}_i,\ \mathsf{prFE.sk}_i\}_{i\in[q_C]}, \mathbf{r}\Big)$.

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

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

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

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

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

Theorem B.5. Assume abLFE and prFE schemes are correct, and PRF is secure. Then the above construction of kpABE scheme is correct.

Proof. From the correctness of prFE scheme we have

$$\begin{aligned} y_i =& \mathrm{F}[\mathsf{crs}, \mathsf{digest}_i, \mathbf{r}](\mathsf{sd}, \mathbf{x}, \mu) \\ =& \mathsf{abLFE.ct}_i = \mathsf{abLFE.Enc}_i(\mathsf{crs}, \mathsf{digest}_i, (\mathbf{x}, \mu); \mathsf{PRF}(\mathsf{sd}, \mathbf{r})) \end{aligned}$$

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

$$\mathsf{abLFE.Dec}(\mathsf{crs}, C, \mathbf{y}) = \mathsf{abLFE.Dec}(\mathsf{crs}, C, \mathsf{abLFE.ct}) = \mu$$

with all but negligible probability. □

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

Theorem B.6. Assume that the prFE scheme is secure (Definition 3.2) with respect to the sampler class containing the sampler Samp as defined in Eq. 47, abLFE scheme satisfies very selective pseudorandom ciphertext security (Definition B.2). Then our construction of kpABE scheme satisfies VerSel-INDr security (Definition 2.16).

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

Proof. Suppose the adversary $\mathcal{A}$ with randomness $\mathsf{coins}_{\mathcal{A}}$ queries for $\mathbf{x}, \mu, C_1, \ldots C_Q$. To prove the security of kpABE scheme as per Definition 2.16, we show

$$\begin{pmatrix} \mathsf{coins}_{\mathcal{A}},\ \mathsf{mpk} = (\mathsf{prFE.mpk}, \mathsf{crs}), \\ \{\mathrm{F}[\mathsf{crs}, \mathsf{digest}_{k,i}, \mathbf{r}_k]\}_{k\in[Q], i\in[q_{C_k}]}, \\ \mathsf{sk}_{C_k} = \{\mathsf{prFE.sk}_{k,i}\}_{k\in[Q], i\in[q_{C_k}]}, \\ \mathsf{ct} = \mathsf{prFE.ct} \leftarrow \mathsf{prFE.Enc}(\mathsf{prFE.mpk}, (\mathsf{sd}, \mathbf{x}, \mu)) \end{pmatrix} \approx_c \begin{pmatrix} \mathsf{coins}_{\mathcal{A}},\ \mathsf{mpk} = (\mathsf{prFE.mpk}, \mathsf{crs}), \\ \{\mathrm{F}[\mathsf{crs}, \mathsf{digest}_{k,i}, \mathbf{r}_k]\}_{k\in[Q], i\in[q_{C_k}]}, \\ \mathsf{sk}_{C_k} = \{\mathsf{prFE.sk}_{k,i}\}_{k\in[Q], i\in[q_{C_k}]}, \\ \mathsf{ct} = \delta \leftarrow \mathcal{CT}_{\mathsf{prFE}} \end{pmatrix} \tag{46}$$

where $\mathrm{F}[\mathsf{crs}, \mathsf{digest}_{k,i}, \mathbf{r}_k]$ denotes the functions corresponding to k-th key query C_k as defined in the KeyGen algorithm and $\mathsf{prFE.sk}_{k,i} \leftarrow \mathsf{prFE.KeyGen}(\mathsf{prFE.msk}, \mathrm{F}[\mathsf{crs}, \mathsf{digest}_{k,i}, \mathbf{r}_k])$ for $k \in [Q], i \in [q_{C_k}]$. Also for all the key queries $C_1, \ldots, C_Q$ and the challenge attribute $\mathbf{x}$ issued by the adversary, we have $C_k(\mathbf{x}) = 0$.
We invoke the security of prFE with sampler Samp that outputs

$$\begin{pmatrix} \text{Functions:} & \{\mathrm{F}[\mathsf{crs}, \mathsf{digest}_{k,i}, \mathbf{r}_k]\}_{k\in[Q], i\in[q_{C_k}]}, \\ \text{Input:} & (\mathsf{sd}, \mathbf{x}, \mu), \\ \text{Auxiliary Information:} & \mathsf{aux} = (\mathsf{coins}_{\mathcal{A}}, \{\mathsf{crs}, C_k, \mathsf{digest}_{k,i}, \mathbf{r}_k\}_{k\in[Q], i\in[q_{C_k}]}) \end{pmatrix} \tag{47}$$

By the security guarantee of prFE with sampler Samp we have

$$\begin{pmatrix} \mathsf{prFE.mpk},\ \mathsf{aux}, \{\mathrm{F}[\mathsf{crs}, \mathsf{digest}_{k,i}, \mathbf{r}_k],\ \mathsf{prFE.sk}_{k,i}\}_{k\in[Q], i\in[q_{C_k}]} \\ \mathsf{prFE.ct} \leftarrow \mathsf{prFE.Enc}(\mathsf{prFE.mpk}, (\mathsf{sd}, \mathbf{x}, \mu)) \end{pmatrix}$$
$$\approx_c \begin{pmatrix} \mathsf{prFE.mpk},\ \mathsf{aux}, \{\mathrm{F}[\mathsf{crs}, \mathsf{digest}_{k,i}, \mathbf{r}_k],\ \mathsf{prFE.sk}_{k,i}\}_{k\in[Q], i\in[q_{C_k}]} \\ \Delta \leftarrow \mathcal{CT}_{\mathsf{prFE}} \end{pmatrix}$$

if

$$\begin{aligned} &\left(\mathsf{aux}, \{\mathrm{F}[\mathsf{crs}, \mathsf{digest}_{k,i}, \mathbf{r}_k],\ \mathsf{abLFE.ct}_{k,i}\}_{k\in[Q], i\in[q_{C_k}]}\right) \\ \approx_c &\left(\mathsf{aux}, \{\mathrm{F}[\mathsf{crs}, \mathsf{digest}_{k,i}, \mathbf{r}_k],\ (\delta_{k,1}, \ldots, \delta_{k,q_{C_k}}) \leftarrow \{0,1\}^{\ell^{\mathsf{abLFE}}_{\mathsf{ct}}}\}_{k\in[Q]}\right) \end{aligned} \tag{48}$$

where $\mathsf{abLFE.ct}_{k,i} = \mathrm{F}[\mathsf{crs}, \mathsf{digest}_{k,i}, \mathbf{r}_k](\mathsf{sd}, \mathbf{x}, \mu) = \mathsf{abLFE.Enc}_i(\mathsf{crs}, \mathsf{digest}_{k,i}, (\mathbf{x}, \mu); \mathsf{PRF}(\mathsf{sd}, \mathbf{r}_k))$ for $k \in [Q], i \in [q_C]$.
Thus to prove Equation (46), it suffices to prove Equation (48). We prove Equation (48) via the following sequence of hybrids.

Hyb_0. This is the LHS distribution of Equation (48).

$$\left(\mathsf{aux}, \{\mathrm{F}[\mathsf{crs}, \mathsf{digest}_{k,i}, \mathbf{r}_k],\ \mathsf{abLFE.ct}_{k,i} = \mathsf{abLFE.Enc}_i(\mathsf{crs}, \mathsf{digest}_{k,i}, (\mathbf{x}, \mu); \mathsf{PRF}(\mathsf{sd}, \mathbf{r}_k))\}_{k\in[Q], i\in[q_{C_k}]}\right).$$

We can rewrite the above distribution as

$$\left(\mathsf{aux}, \{\mathrm{F}[\mathsf{crs}, \mathsf{digest}_{k,i}, \mathbf{r}_k],\ \mathsf{abLFE.ct}_k = \mathsf{abLFE.Enc}(\mathsf{crs}, \mathsf{digest}_k, (\mathbf{x}, \mu); \mathsf{PRF}(\mathsf{sd}, \mathbf{r}_k))\}_{k\in[Q], i\in[q_{C_k}]}\right).$$

where $\mathsf{digest}_k = \{\mathsf{digest}_{k,i}\}_{i\in[q_{C_k}]}$ and $\mathsf{abLFE.ct}_k = \{\mathsf{abLFE.ct}_{k,i}\}_{i\in[q_{C_k}]}$ for all $k \in [Q]$.

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

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

$$\left(\mathsf{aux}, \{\mathsf{F}[\mathsf{crs}, \mathsf{digest}_{k,i}, \mathbf{r}_k],\ \mathsf{abLFE.ct}_k = \mathsf{abLFE.Enc}(\mathsf{crs}, \mathsf{digest}_k, (\mathbf{x}, \mu))\}_{k\in[Q], i\in[q_{C_k}]}\right)$$

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

$$\left(\{\mathsf{F}[\mathsf{crs}, \mathsf{digest}_{k,i}, \mathbf{r}_k],\ \mathsf{abLFE.ct}_k \leftarrow \mathcal{CT}_{\mathsf{abLFE}}\}_{k\in[Q], i\in[q_{C_k}]}\right)$$

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

This completes the proof. □

B.3 AB-LFE (or 1ABE) from Blind Garbled circuits.

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

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

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

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

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

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

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

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

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

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

Theorem B.7. Assume that the blind bGC is correct (Definition 2.19). Then the abLFE scheme is correct (Definition B.1).

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

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

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

Efficiency. The abLFE scheme has the following efficiency properties.

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

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

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

Theorem B.8. Assume that the bGC scheme satisfies simulation security (Definition 2.20) and blindness (Definition 2.21). Then the abLFE scheme satisfies adaptive pseudorandom ciphertext security (Definition B.2).

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

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

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

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

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

Equivalence of $\mathsf{1ABE}$ and abLFE from Blind Garbled Circuits. We show that the abLFE scheme instantiated from blind garbled circuits is in fact equivalent to the instantiation of $\mathsf{1ABE}$ using BGC– which we bootstrap to a full fledged KP-ABE using prFE in the technical overview (Section 1.4).

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

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

Instantiating the abLFE scheme as above, and using a prFE scheme supporting $d_{\mathsf{prFE}} = \mathrm{poly}(\lambda)$ depth circuits with input length $\mathrm{L} = \ell + \lambda + 1$, we obtain the following theorem.

Theorem B.9. Under the LWE assumption and the Evasive LWE assumption, there exists a very selectively secure kpABE scheme for circuits of unbounded depth and attribute length ℓ with

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

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

B.4 Using the abLFE from HLL

We can directly instantiate the abLFE in Appendix B.2 with the construction shown by [HLL23] (HLL henceforth). For this instantiation, we do not assume decomposability of abLFE since the HLL construction has succinct digest, i.e. $q_C = 1$ for any C and $|\mathsf{digest}| = O(1)$. This instantiation leads to the parameter size of $|\mathsf{mpk}| = \mathrm{poly}(\ell, \lambda)$, $|\mathsf{sk}_C| = \mathrm{poly}(\ell, \lambda)$, and $|\mathsf{ct}| = \mathrm{poly}(\ell, \lambda)$, which is already better than the previous instantiation.

By adding a twist to the construction in Appendix B.2 exploiting the structural property of HLL, we can improve the secret key size so that its dependency on ℓ can be removed. To do so, we exploit the online-offline structure of the $\mathsf{abLFE.Enc}$ algorithm which can be split as $\mathsf{abLFE.Enc} = (\mathsf{abLFE.EncOff}, \mathsf{abLFE.EncOn})$. Here, $\mathsf{abLFE.EncOff}$ takes as input crs and $\mathbf{x}$ and outputs the offline part of the ciphertext $\mathsf{abLFE.ct_{off}}$ and short state st and $\mathsf{abLFE.EncOn}$ takes as input digest and st and outputs online part of the ciphertext $\mathsf{abLFE.ct_{on}}$. Formally, HLL proved the following theorem:

Theorem B.10 ([HLL23]). Under the circular LWE assumption, there exists a very selectively secure abLFE scheme for circuit class $\mathcal{C} = \{C : \{0,1\}^{\ell} \to \{0,1\}^{\ell'}\}$ satisfying

$$|\mathsf{crs}| = O(\ell, \lambda),\ |\mathsf{digest}| = O(\lambda),\ |\mathsf{st}| = O(\lambda),\ |\mathsf{ct_{off}}| = O(\ell, \lambda),\ |\mathsf{ct_{on}}| = O(\ell', \lambda)$$

We make slight modifications to our construction of kpABE scheme to optimize the secret key size. The high level idea is very simple. Instead of letting the prFE decryption recover the entire abLFE ciphertext, we recover only the online part of it. We then put the offline part of abLFE ciphertext into the ciphertext of kpABE so that the decryptor can recover the entire abLFE ciphertext during the decryption. This eliminates the necessity of hardwiring crs to the prFE secret key, since the online part can be computed only from the short state and digest. This leads to the improvement on the efficiency, since the state and digest are of fixed polynomial size, while crs is of size $O(\ell)$. Concretely, we modify the $\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}$ algorithm of Appendix B.2 as follows.

$\mathsf{KeyGen}(\mathsf{msk}, C)$. This is same as the KeyGen algorithm in Appendix B.2 except the following.

- Define circuit $\mathrm{F}[\mathsf{digest}, \mathbf{r}]$ (instead of $\mathrm{F}[\mathsf{crs}, \mathsf{digest}_i, \mathbf{r}]$) , with $\mathsf{digest}, \mathbf{r}$ hardwired, as follows
  On input $(\mathsf{sd}, \mathsf{st})$:
  - Compute $\mathsf{abLFE.ct_{on}} := \mathsf{abLFE.EncOn}(\mathsf{st}, \mathsf{digest}; \mathsf{PRF}(\mathsf{sd}, \mathbf{r}))$.
  - Output $\mathsf{abLFE.ct_{on}}$.
- Compute $\mathsf{prFE.sk} \leftarrow \mathsf{prFE.KeyGen}(\mathsf{prFE.msk}, \mathrm{F}[\mathsf{digest}, \mathbf{r}])$.
- Output $\mathsf{sk}_C = (\mathsf{digest}, \mathsf{prFE.sk}, \mathbf{r})$.

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

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

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

- Parse $\mathsf{mpk} = (\mathsf{prFE.mpk}, \mathsf{crs})$, $\mathsf{sk}_C = (\mathsf{digest}, \mathsf{prFE.sk}, \mathbf{r})$ and $\mathsf{ct} = (\mathsf{abLFE.ct_{off}}, \mathsf{prFE.ct})$.
- Compute $\mathsf{abLFE.ct_{on}} = \mathsf{prFE.Dec}(\mathsf{prFE.mpk}, \mathsf{prFE.sk}, \mathrm{F}[\mathsf{digest}, \mathbf{r}], \mathsf{prFE.ct})$.

– Set $\mathbf{y} = (\mathsf{abLFE.ct}_{\mathsf{off}}, \mathsf{abLFE.ct}_{\mathsf{on}})$ and output $\mathsf{abLFE.Dec}(\mathsf{crs}, C, \mathbf{y})$.

We note that even with the above changes the correctness and security arguments are same as that of Appendix B.2.

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

Theorem B.11. Under the circular LWE assumption and the Evasive LWE assumption, there exists a very selectively secure kpABE scheme for circuits of unbounded depth and attribute length ℓ with

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

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