Rate-1 Key-Dependent Message Security via Reusable Homomorphic Extractor against Correlated-Source Attacks

Qiqi Lai[1,2], Feng-Hao Liu[3], Zhedong Wang[4] (Corresponding Author)

[1] School of Computer Science, Shaanxi Normal University, Xi'an, China
laiqq@snnu.edu.cn.
[2] Henan Key Laboratory of Network Cryptography Technology, Zhengzhou, China
[3] Florida Atlantic University, Boca Raton, FL, USA
fenghao.liu@fau.edu.
[4] School of Cyber Science and Engineering, Shanghai Jiao Tong University, Shanghai, China
wzdstill@sjtu.edu.cn.

Abstract. In this work, we first present general methods to construct information rate-1 PKE that is $\mathsf{KDM}^{(n)}$-secure with respect to *block-affine* functions for any unbounded polynomial n. To achieve this, we propose a new notion of extractor that satisfies *reusability*, *homomorphism*, and *security against correlated-source attacks*, and show how to use this extractor to improve the information rate of the KDM-secure PKE of Brakerski et al. (Eurocrypt 18). Then, we show how to amplify KDM security from block-affine function class into general bounded size circuits via a variant of the technique of Applebaum (Eurocrypt 11), achieving better efficiency. Furthermore, we show how to generalize these approaches to the IBE setting.

Additionally, our PKE and IBE schemes are also leakage resilient, with leakage rates $1 - o(1)$ against a slightly smaller yet still general class – block leakage functions. We can instantiate the required building blocks from LWE or DDH.

1 Introduction

The classic notion of *semantic security* by Goldwasser and Micali [25] guarantees security when the secret key is generated randomly and independently of the message being encrypted. This notion however, is not sufficient in various scenarios, e.g., [1, 13, 17, 33]. To tackle this issue, [11, 12] formally defined *Key Dependent Message* (KDM) security, which requires $\mathsf{Enc}(\mathsf{pk}, \mathsf{f}(\mathsf{sk}))$ to be indistinguishable from $\mathsf{Enc}(\mathsf{pk}, 0)$ for all f in a certain class. The setting can be generalized to n-users, i.e., $\mathsf{KDM}^{(n)}$-security, where security holds even when the attacker obtains the encryption of $\mathsf{f}(\mathsf{sk}_1, \ldots, \mathsf{sk}_n)$ under some user's (public) key. The community has established various theoretical feasibility results – we know how to construct $\mathsf{KDM}^{(n)}$-secure PKE for unbounded polynomial n from the LWE [8], DDH [12], or LPN [8, 22, 31] assumption, for bounded polynomial n from QR/DCR assumption [13], and for $n = 1$ from CDH [16].

On the other hand however, all the prior constructions have relatively small information rate[5] even for the class of linear functions, resulting in very large overhead in scenarios that require encrypting large data, e.g., storing large encrypted files in the cloud, or streaming encrypted high-resolution movies over the internet. To remove this limitation and enhance usability, it is necessary to determine whether a low information rate is inherent for KDM security.

As folklore, this issue (low information rate) can be solved easily for regular PKE, as one can always achieve rate $1 - o(1)$ by using the technique of hybrid encryption (the KEM-DEM paradigm). It is however, not clear whether KDM security can be preserved under a general hybrid encryption [29]. This direction has remained an important open problem (ref. [12, 13]). Therefore, we ask:

> **Main Question**: Can we construct a $\mathsf{KDM}^{(n)}$-secure PKE with better information rate, e.g. $1 - o(1)$, even for $n = 1$ and linear functions?

1.1 Our Contributions

This work answers the main question and makes the following contributions:

Contribution 1. We show how to construct a $\mathsf{KDM}^{(1)}$-secure PKE with information rate $1 - o(1)$ with respect to *block-affine* functions, a slightly more restricted class than that of bit-affine functions. To achieve this, we first propose a new primitive – *reusable homomorphic extractor against correlated-source attacks*, and instantiate it based on DDH or LWE. Next, we show how to use this primitive to improve the approach of Batch Encryption (BE) [16], which was used to derive $\mathsf{KDM}^{(1)}$-secure PKE (albeit low information rates.)

Particularly, we identify that BE implies a weak hash proof system (wHPS) with important additional properties. Then we show that our new extractor can be integrated with such a wHPS to achieve $\mathsf{KDM}^{(1)}$-security with information rate $1 - o(1)$. Our proof technique connects wHPS and the new reusable homomorphic extractor in a novel way, which deviates from the prior simulation approach [8, 10, 12–14, 16]. The new extractor and proof technique can be of independent interest.

Contribution 2. We show how to upgrade the above approach to achieve $\mathsf{KDM}^{(n)}$-secure PKE for unbounded polynomial n. Particularly, we identify the technical barrier of the current BE-based approach [16], which inherently can only achieve a bounded polynomial n. To tackle this, we construct an enhanced variant of the current BE by adding a new *reusable* property. By using this stronger BE as the underlying building block of wHPS, the scheme in Contribution 1 can be proved $\mathsf{KDM}^{(n)}$-secure for any unbounded polynomial n. For instantiations, we construct the required extractor and BE from DDH or LWE. Thus, either of these assumptions implies $\mathsf{KDM}^{(n)}$-secure PKE with the optimal information rate, i.e., $1 - o(1)$.

[5] Information rate is defined as the message-length-to-ciphertext-length ratio when one encrypts sufficiently long plaintexts.

Our design of KDM-PKE is quite modular, which might open a path for further constructions from other assumptions, as long as we can construct the required building blocks.

Contribution 3. We generalize the above approach in two directions. First, we show that the class of block-affine function is still sufficient for KDM amplification to the class of general bounded-sized circuits via a variant of the technique in [7], even the class of block-affine functions is more restricted, i.e., it does not contain all projection functions, so that the generic KDM amplification of Applebaum [7] does not work. Thus, the block-affine function class is still sufficiently general, and can yield more efficient constructions.

Second, we construct $\mathsf{KDM}^{(n)}$-secure IBE for unbounded n with the $1 - o(1)$ information rate. The corresponding KDM function class here is slightly smaller than the allowable KDM class for our PKE. We discuss this allowable class next. Moreover, the required building blocks can be instantiated based on DDH in the bilinear group or LWE.

In addition to KDM security, our PKE schemes (both DDH and LWE-based) are leakage resilient. The leakage rate is optimal, i.e., $1 - o(1)$, against block leakage, which is slightly smaller than the general leakage class[6]. The IBE schemes are as well leakage resilient. For the same class of leakage functions, the IBE leakage rate can achieve $1 - o(1)$ under LWE or DDH with respect to some bilinear maps.

1.2 Technical Overview

In this section, we present a technical overview of our contributions. We start with the construction of $\mathsf{KDM}^{(1)}$-secure PKE with information rate $1 - o(1)$. To achieve this target, we first identify several new properties from (Identity-based) weak Hash Proof Systems (wHPS) [4, 28], Batch Encryption (BE) [16], and randomness extractors [5], and then describe our new idea to integrate these properties. Before describing our new insights, we first review the following two important tools – wHPS and BE.

(Weak) Hash Proof System. A hash proof system can be described as a key encapsulation mechanism that consists of four algorithms $(\mathsf{Setup}, \mathsf{Encap}, \mathsf{Encap}^*, \mathsf{Decap})$: (1) Setup generates a key pair $(\mathsf{pk}, \mathsf{sk})$, (2) $\mathsf{Encap}(\mathsf{pk})$ outputs a pair $(\mathsf{CT}, \mathsf{k})$ where k is a key encapsulated in a "valid" ciphertext CT, (3) $\mathsf{Encap}^*(\mathsf{pk})$ outputs an "invalid" ciphertext CT^*, and (4) $\mathsf{Decap}(\mathsf{sk}, \mathsf{CT})$ outputs a key k'. A (weak) hash proof system needs to satisfy the following three properties:

- **Correctness.** For a valid ciphertext CT, the Decap algorithm always outputs the encapsulated key k' such that $\mathsf{k}' = \mathsf{k}$, where $(\mathsf{CT}, \mathsf{k}) \xleftarrow{\$} \mathsf{Encap}(\mathsf{pk})$.
- **Ciphertext Indistinguishability.** Valid ciphertexts and invalid ciphertexts are computationally indistinguishable, *even given the secret key* sk. This property is essential for achieving leakage resilience and KDM security.

[6] When the secret key is stored in blocks, a block leakage function can leak individual blocks one after another, as long as the blocks still remain a block source.

– **Universal.** The wHPS is (ℓ, w)-universal if given the public key pk and an invalid ciphertext CT^*, the decapsulated key length is ℓ and the conditional min-entropy of the decapsulation of CT^* is greater or equal to w, i.e., $H_\infty(\mathsf{Decap}(\mathsf{sk}, \mathsf{CT}^*) \mid (\mathsf{pk}, \mathsf{CT}^*)) \geq w$. A wHPS only requires this property to hold for a random invalid ciphertext, i.e. $\mathsf{CT}^* \xleftarrow{\$} \mathsf{Encap}^*(\mathsf{pk})$, while a full-fledged HPS requires it to hold for any invalid ciphertext.

We note that wHPS has been used to achieve leakage resilience (LR) in prior work [4,36]. Homomorphic wHPS has been used to achieve $\mathsf{KDM}^{(1)}$-security [42]. It was not clear whether wHPS can be used to achieve $\mathsf{KDM}^{(1)}$-security with the optimal information rate.

Batch Encryption [16]. A Batch Encryption (BE) consists of four algorithms: $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$. The secret key is a vector $\boldsymbol{x} \in \mathbb{Z}_B^n$ for $B, n \in \mathbb{N}$. The Setup algorithm simply outputs a random common reference string CRS, and $\mathsf{KeyGen}(\mathsf{CRS}, \boldsymbol{x})$ is a projection function that outputs (a short) hash value h of $\boldsymbol{x}$ and CRS. The encryption algorithm takes an $n \times B$ matrix $\mathbf{M}$ and (CRS, h) as input, and outputs a ciphertext $\mathsf{CT} \leftarrow \mathsf{Enc}((\mathsf{CRS}, h), \mathbf{M})$. The decryption algorithm taking as input a ciphertext CT and a secret key $\boldsymbol{x}$, can only recover M_{i,x_i}, i.e., the x_i-th entry in the i-th row, for $1 \leq i \leq n$, while the other entries remain hidden *even given* the secret key $\boldsymbol{x}$. The work [16] showed that BE can be instantiated from LWE, CDH, and LPN with the *succinctness* property, i.e. the size of $|h|$ depends only on the security parameter and can be set as $o(n)$. Using a succinct BE as a central building block, the work [16] constructed a PKE that simultaneously achieves $\mathsf{KDM}^{(1)}$-security for affine functions and leakage resilience with the optimal leakage rate, i.e., $1 - o(1)$.

Even though the above tools have been demonstrated powerful, there are two common limitations for the current techniques – (1) KDM-security can be achieved only for bounded users, and (2) the information rate is quite low, e.g., $\frac{1}{O(\lambda)}$. Next, we present our new insights to break these technical barriers.

1.2.1 Our New Insights

We start with a simple observation that BE can be used to construct wHPS with additional structures, which are critical in achieving KDM-security. Then we introduce our new variant of random extractor, and sketch its instantiations from DDH and LWE. With all these preparations, we show our new ideas to achieve KDM security.

wHPS from BE. We can construct a wHPS from BE in the following simple way. $\mathsf{wHPS.sk}$ is a random $\boldsymbol{x} \in \mathbb{Z}_B^n$, and $\mathsf{wHPS.pk} = (\mathsf{CRS}, h)$ where CRS, h are generated according to the underlying BE. The valid encapsulation algorithm $\mathsf{wHPS.Encap}$ just samples a random vector $\boldsymbol{k} = (k_1, \ldots, k_n)^\top \in \mathbb{Z}_B^n$ as the encapsulated key and generates the ciphertext by $\mathsf{BE.Enc}(\mathbf{M})$, where the i-th row of $\mathbf{M}$ is set as $(k_i, k_i, \ldots, k_i)$ for $i \in [n]$. On the other hand, the invalid encapsulation algorithm $\mathsf{wHPS.Encap}^*$ generates an invalid ciphertext $\mathsf{CT}^* \leftarrow \mathsf{BE.Enc}(\mathbf{M})$

by first sampling a random vector $\boldsymbol{k}' = (k'_1, \ldots, k'_n)^\top$ and then setting the i-th row of $\mathbf{M}$ as $(k'_i+0, k'_i+1, \ldots, k'_i+B-1)$ for $i \in [n]$. Moreover, the decapsulation algorithm $\mathsf{wHPS.Decap}$ simply outputs the decryption result of $\mathsf{BE.Dec}(\boldsymbol{x}, \mathsf{CT})$.

It is not hard to show that this construction is an $(n \log B, n \log B - |h|)$-universal wHPS, which can be used to achieve a $\mathsf{LR\text{-}PKE}$ that tolerates $(n \log B - |h| - k)$-bit leakage by using a (k, ε)-extractor (ref. [4, 36]).[7] Particularly, the corresponding leakage resilient public-key encryption scheme PKE_1 can be constructed as follows: $\mathsf{PKE}_1.\mathsf{pk} = \mathsf{wHPS.pk}$ and $\mathsf{PKE}_1.\mathsf{sk} = \mathsf{wHPS.sk}$. To encrypt a message m, the encryption algorithm first generates $(\mathsf{CT}, \boldsymbol{k}) \leftarrow \mathsf{wHPS.Encap}$ and samples $\boldsymbol{r}$ as the randomness of a strong randomness extractor $\mathsf{Ext}(\cdot, \cdot)$, and then outputs $(\mathsf{CT}, \boldsymbol{r}, \mathsf{Ext}(\boldsymbol{r}, \boldsymbol{k}) + m)$ as the ciphertext.

Generally, a plain extractor is not sufficient to derive KDM security for PKE_1 in the above paradigm. Interestingly, this task is possible if we use our reusable homomorphic extractor against correlated-source attacks, and the wHPS has appropriate additional structures. Next, we describe the required extractor.

Our New Notion of Extractor and Constructions. We identify three properties of an extractor: (1) reusable, (2) homomorphic, and (3) secure against correlated-source attacks.

Let $\mathsf{Ext}(\boldsymbol{r}, \boldsymbol{s})$ be an extractor, where $\boldsymbol{s}$ is the source and $\boldsymbol{r}$ is the seed. A reusable extractor requires that the same source $\boldsymbol{s}$ can be repeatedly extracted by different seeds for any polynomially many times while maintaining pseudorandomness. That is, for any $m = \mathsf{poly}(\lambda)$ and source $\boldsymbol{s}$ with sufficient entropy, we have $(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_m, \mathsf{Ext}(\boldsymbol{r}_1, \boldsymbol{s}), \ldots, \mathsf{Ext}(\boldsymbol{r}_m, \boldsymbol{s})) \approx (\boldsymbol{r}_1, \ldots, \boldsymbol{r}_m, u_1, \ldots, u_m)$, where each u_i is uniformly random.[8] Previously, the work [5, 20, 36] showed that under computational assumptions, e.g., DDH or LWE, the reusability can be achieved.

The extractor $\mathsf{Ext}(\boldsymbol{r}, \boldsymbol{s})$ is (output) homomorphic with respect to a function $\mathfrak{h}$ if there exists a related function $\mathfrak{h}'$ such that $\mathsf{Ext}(\boldsymbol{r}, \boldsymbol{s}) + \mathfrak{h}(\boldsymbol{s}) = \mathsf{Ext}(\mathfrak{h}'(\boldsymbol{r}), \boldsymbol{s})$. Similar to the work of [42], we will use this homomorphic property in a critical way to achieve KDM security.

We say the (reusable) extractor $\mathsf{Ext}(\boldsymbol{r}, \boldsymbol{s})$ is secure against correlated-source attacks if for functions (perhaps chosen adaptively by the attacker) in some class $\mathcal{F}$, such that for $m = \mathsf{poly}(\lambda)$ and $\mathfrak{g}_1, \ldots, \mathfrak{g}_m \in \mathcal{F}$, the extractor remains pseudorandom as follows:

$$(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_m, \mathsf{Ext}(\boldsymbol{r}_1, \mathfrak{g}_1(\boldsymbol{s})), \ldots, \mathsf{Ext}(\boldsymbol{r}_m, \mathfrak{g}_m(\boldsymbol{s}))) \approx (\boldsymbol{r}_1, \ldots, \boldsymbol{r}_m, u_1, \ldots, u_m)\,.^{9}$$

Our notion of correlated-source attacks is similar to that of a recent work by Goyal and Song [26], yet with the following major differences. First, the security requirements are different. The work [26] considers information-theoretic

[7] The extractor can extract uniform string (up to statistical distance ε) for any source with min-entropy k.

[8] Clearly, this notion cannot be achieved unconditionally, as an information-theoretic extractor requires (conditional) min-entropy from the source, which would be exhausted after a bounded number of extractions.

[9] Clearly, this notion is stronger than the reusable extractor, which can be viewed as a special case where $\mathfrak{g}_i$'s are all the identity function. Thus, this notion is only possible under computational assumptions.

indistinguishability of *one* instance of extraction from the *original source*, even given multiple extractions from the modified source, i.e.,

$$(\boldsymbol{r}, \mathsf{Ext}(\boldsymbol{r}, \boldsymbol{s}), \{\boldsymbol{r}_i, \mathsf{Ext}(\boldsymbol{r}_i, \mathfrak{g}_i(\boldsymbol{s}))\}_{i\in m}) \approx (\boldsymbol{r}, u, \{\boldsymbol{r}_i, \mathsf{Ext}(\boldsymbol{r}_i, \mathfrak{g}_i(\boldsymbol{s}))\}_{i\in m}) .$$

In contrast, our notion requires that all instances of $\mathsf{Ext}(\boldsymbol{r}_i, \mathfrak{g}_i(\boldsymbol{s}))$ remain pseudorandom, which is a stronger requirement (in this aspect).

Second, the ranges of feasible function classes are different. Specifically, our notion is too strong to achieve for the class of all functions. For example, if $\mathfrak{g}_i$ is a constant function, then $\mathsf{Ext}(\boldsymbol{r}_i, \mathfrak{g}_i(\boldsymbol{s}))$ becomes a fixed value given $\boldsymbol{r}_i$, and thus cannot be pseudorandom. This indicates a necessary condition for feasibility that the function must be entropy preserving. However, the notion in [26] is possible to achieve even unconditionally for the class of all functions, as their challenge instance is extracted from an unmodified source.

Third, to achieve information-theoretic extraction for all arbitrary input correlated functions, the number m of extraction samples given extra in the distribution in [26] must be bounded inherently, and thus cannot be fully reusable.

Summing up the above analyses, we conclude that our security requirement is stronger, resulting in a relatively smaller feasible function class. Moreover, our notion requires reusability for an unbounded polynomial samples, and thus a computational assumption is necessary.

Next, we discuss how to construct such an extractor that simultaneously achieves all the three properties.

Construction based on DDH. We start with a review of the existing DDH-based reusable extractor. Let $\mathbb{G}$ be the DDH group of order q, $\boldsymbol{r} \in \mathbb{G}^n$ be seed, and $\boldsymbol{s} \in \mathbb{Z}_q^n$ be source. The following function has been proved to be a reusable extractor in [5,36]: $\mathsf{Ext}(\boldsymbol{r}, \boldsymbol{s}) = \prod_{i=1}^n r_i^{s_i}$. Moreover, we notice the following two properties about this extractor: (1) it is output homomorphic with respect to functions of the form $\mathfrak{h}_{\boldsymbol{b}}(\boldsymbol{s}) = \prod_{i=1}^n b_i^{s_i}$, as $\mathsf{Ext}(\boldsymbol{r}, \boldsymbol{s}) \cdot \mathfrak{h}_{\boldsymbol{b}}(\boldsymbol{s}) = \prod_{i=1}^n (r_i \cdot b_i)^{s_i} = \mathsf{Ext}(\boldsymbol{r} \circ \boldsymbol{b}, \boldsymbol{s})$, where $\circ$ is the component-wise group multiplication; (2) the extractor remains pseudorandom against the correlated source attacks with respect to linear shift functions of the form $\mathfrak{g}_{\boldsymbol{v}}(\boldsymbol{s}) = \boldsymbol{s}+\boldsymbol{v}$. Due to the fact $\mathsf{Ext}(\boldsymbol{r}, \boldsymbol{s}+\boldsymbol{v}) = \prod_{i=1}^n r_i^{s_i+v_i} = \mathsf{Ext}(\boldsymbol{r}, \boldsymbol{s}) \cdot \mathsf{Ext}(\boldsymbol{r}, \boldsymbol{v})$, we can simulate $\mathsf{Ext}(\boldsymbol{r}, \mathfrak{g}_{\boldsymbol{v}}(\boldsymbol{s}))$ given $(\boldsymbol{r}, \mathsf{Ext}(\boldsymbol{r}, \boldsymbol{s}))$ and $\mathfrak{g}_{\boldsymbol{v}}$. Via this simple reduction, the security of the reusable extractor directly translates to the security against correlated-source attacks with respect to linear shifts.

At first, it seems that the existing construction already fulfills the three required properties. However, when considering the application to KDM-secure PKE, we notice an obstacle that this extractor is still not compatible with the above mentioned framework of the weak hash proof system based on batch encryption (BE). Below, we sketch the major reason for this incompatibility, and discuss our solution in the following.

Particularly, the BE-based system requires that each component of the secret vector comes from a polynomial-sized domain, i.e., $\boldsymbol{s} \in S^n$ for $|S| = \mathsf{poly}(\lambda)$. However, the above construction has the domain $\mathbb{G}^n$, which is clearly too large, as DDH assumption holds only when the order q is super-polynomial. To tackle this issue, one might set $S = \mathbb{Z}_p$ for some small p. However, for a subtle reason this approach faces an additional technical difficulty. More specifically, due to the BE

feature, the linear shift should work in $\mathbb{Z}_p$, i.e., $\mathfrak{g}_{\boldsymbol{v}}(\boldsymbol{s}) = \boldsymbol{s} + \boldsymbol{v} \mod p$. However, this equation might not hold for the above mentioned reduction on correlated-source security, i.e., $\mathsf{Ext}(\boldsymbol{r}, (\boldsymbol{s} + \boldsymbol{v} \mod q)) = \mathsf{Ext}(\boldsymbol{r}, \boldsymbol{s}) \cdot \mathsf{Ext}(\boldsymbol{r}, \boldsymbol{v}) \neq \mathsf{Ext}(\boldsymbol{r}, (\boldsymbol{s} + \boldsymbol{v} \mod p))$. Thus, it is unclear whether we can achieve correlated-source security against linear shifts (modulo p) by using the linearity of the extractor, which essentially works only in modulo q.

To solve this issue, we set $S = \mathbb{Z}_2$, and use another route of reduction that avoids using the above linearity equation. Particularly, we show a way to transform an instance of the form $(\boldsymbol{r}, z = \mathsf{Ext}(\boldsymbol{r}, \boldsymbol{s}))$ into $(\boldsymbol{r}', \mathsf{Ext}(\boldsymbol{r}', (\boldsymbol{s} + \boldsymbol{b} \mod 2)))$ given $\boldsymbol{b}$, without using the linearity of the extractor. Furthermore, via a reduction from the reusability of the extractor, we can establish security against correlated-source attacks for linear shifts in modulo 2. This would suffice to achieve KDM security as we discuss later. More formally, the transformation works as follow:

- For $i \in [n]$, if $b_i = 0$, set $r'_i = r_i$ and $z'_i = 1$; otherwise for $b_i = 1$, set $r'_i = r_i^{-1}$ and $z'_i = r_i^{-1}$.
- Output $(\boldsymbol{r}', z \cdot \prod_{i=1}^{n} z'_i)$.

We note that if $b_i = 0$, then the term $r_i^{s_i}$ would appear in z, or otherwise $r_i^{1-s_i}$. With a simple check, our transformation is consistent with this fact. It is not hard to formalize the security proof using this idea.

Construction based on LWE. Next we look at the LWE-based reusable extractor [5]. Let $q > p > 1$ be parameters, S be some small set over $\mathbb{Z}_q$, $\boldsymbol{r} \in \mathbb{Z}_q^n$ be seed, and $\boldsymbol{s} \in S^n$ be source. The work [5,9] showed that $\mathsf{Ext}(\boldsymbol{r}, \boldsymbol{s}) = \lceil \langle \boldsymbol{r}, \boldsymbol{s} \rangle \rfloor_p$ is a reusable extractor where $\lceil \cdot \rfloor$ is some rounding function, and the number of reusable samples can be any arbitrary polynomial if $q/p = \lambda^{\omega(1)}$. For general settings of parameters however, this extractor might not be output homomorphic, as linearity might not hold for rounding of inner products. Nevertheless, we identify that if $p|q$, then the extractor is output homomorphic with respect to linear functions (i.e. $\mathfrak{h}_{\boldsymbol{b}}(\boldsymbol{s}) = \langle \boldsymbol{b}, \boldsymbol{s} \rangle \mod p$) by using the following equation:

$$\lceil \langle \boldsymbol{r}, \boldsymbol{s} \rangle \rfloor_p + \langle \boldsymbol{b}, \boldsymbol{s} \rangle = \lceil \langle \boldsymbol{r}, \boldsymbol{s} \rangle + (q/p) \langle \boldsymbol{b}, \boldsymbol{s} \rangle \rfloor_p = \lceil \langle \boldsymbol{r} + (q/p)\boldsymbol{b}, \boldsymbol{s} \rangle \rfloor_p.$$

Thus, we can set $\mathfrak{h}'_{\boldsymbol{b}}(\boldsymbol{r}) = \boldsymbol{r} + (q/p)\boldsymbol{b}$, achieving the desired property.

Next, we would like to show that the construction is secure against correlated-source attacks for linear shifts. Similar to the DDH construction, we need to tackle the issue that $\mathfrak{g}_{\boldsymbol{b}}(\boldsymbol{s})$ and $\mathsf{Ext}(\boldsymbol{r}, \boldsymbol{s})$ are working on different moduli. To solve this issue, we first apply the same idea by setting $S = \mathbb{Z}_2^n$, and then hopefully a similar reduction would work. However, this method does not work in a straight-forward way as rounding breaks linearity. Let us consider a simple case where $\boldsymbol{b} = (1, 0, 0, \ldots, 0)^T$, i.e., only $b_1 = 1$ and others 0. Then the reduction would need to simulate $\lceil r_1(1 - s_1) + \sum_{i=2}^{n} r_i s_i \rfloor_p = \lceil -r'_1 + r'_1 s_1 + \sum_{i=2}^{n} r'_i s_i \rfloor_p$, where $r'_1 = -r_1$ and $r'_i = r_i$ for $i = 2 \sim n$. However, $\lceil -r'_1 + r'_1 s_1 + \sum_{i=2}^{n} r'_i s_i \rfloor_p \neq \lceil -r'_1 \rfloor_p + \lceil r'_1 s_1 + \sum_{i=2}^{n} r'_i s_i \rfloor_p$ in general, and thus the previous transformation would break down.

To solve this, we use the proof technique of [9], who first switches the rounded inner products into rounded LWE samples. Then we show that LWE is resilient to correlated-source attacks for linear shifts, translating to security of the whole

construction. More specifically, we first switch $\lceil \langle \boldsymbol{r}, \boldsymbol{s} \rangle \rfloor_p$ to $\lceil \langle \boldsymbol{r}, \boldsymbol{s} \rangle + e \rfloor_p$. The switch incurs a negligible statistical distance if $q/p = \lambda^{\omega(1)}$, which is required for the reusability for an arbitrary polynomial samples anyway (under current proof techniques). Then by using the above idea, we can easily show that samples of the form $\langle \boldsymbol{r}, (\boldsymbol{s} + \boldsymbol{b} \mod 2) \rangle + e$ are computationally indistinguishable from random samples, and therefore so are their rounded versions. This describes the proof ideas.

1.2.2 $\mathsf{KDM}^{(1)}$-PKE with $1 - o(1)$ Rate via the Extractor

Below, we first sketch how to achieve $\mathsf{KDM}^{(1)}$-security, and then present how to improve the information rate to $1 - o(1)$.

Achieving $\mathsf{KDM}^{(1)}$-security. To illustrate our idea, we consider the case where Ext is homomorphic with respect to linear functions and secure against correlated-attacks with respect to linear shifts. Next, we identify three important *additional structures* of wHPS from the above construction:

1. The secret key of wHPS (and PKE_1) is just a vector $\boldsymbol{x} \in \mathbb{Z}_B^n$ as the BE.
2. The decapsulation of an invalid ciphertext CT^* has the following form: $\mathsf{Decap}(\boldsymbol{x}, \mathsf{CT}^*) = \boldsymbol{x} + \boldsymbol{k}'$, where $\boldsymbol{k}' \in \mathbb{Z}_B^n$ is certain vector related to CT^*.
3. Given $\boldsymbol{k}'$, the above CT^* can be simulated faithfully.

Let $\mathfrak{h}$ be some linear functions, and let us take a look at the equation of upon a KDM query of an encryption of $\mathfrak{h}(\mathsf{sk})$.

$$
\begin{aligned}
& \mathsf{PKE}_1.\mathsf{Enc}(\mathfrak{h}(\mathsf{sk})) \\
=& (\mathsf{CT}, \boldsymbol{r}, \mathsf{Ext}(\boldsymbol{r}, \boldsymbol{k}) + \mathfrak{h}(\boldsymbol{x})) && \text{By Structure 1} \\
=& (\mathsf{CT}, \boldsymbol{r}, \mathsf{Ext}(\boldsymbol{r}, \mathsf{Decap}(\boldsymbol{x}, \mathsf{CT})) + \mathfrak{h}(\boldsymbol{x})) && \text{Correctness of } \mathsf{wHPS} \\
\approx& (\mathsf{CT}^*, \boldsymbol{r}, \mathsf{Ext}(\boldsymbol{r}, \mathsf{Decap}(\boldsymbol{x}, \mathsf{CT}^*)) + \mathfrak{h}(\boldsymbol{x})) && \text{Ciphertext indistinguishability} \\
=& (\mathsf{CT}^*, \boldsymbol{r}, \mathsf{Ext}(\boldsymbol{r}, \boldsymbol{x} + \boldsymbol{k}') + \mathfrak{h}(\boldsymbol{x})) && \text{By Structure 2} \\
=& (\mathsf{CT}^*, \boldsymbol{r}, \mathsf{Ext}(\boldsymbol{r}, \boldsymbol{x}') + \mathfrak{h}(\boldsymbol{x}' - \boldsymbol{k}')) && \text{Change of variable} \\
=& (\mathsf{CT}^*, \boldsymbol{r}, \mathsf{Ext}(\boldsymbol{r}, \boldsymbol{x}') + \mathfrak{h}(\boldsymbol{x}') - \mathfrak{h}(\boldsymbol{k}')) && \text{Linearity of } \mathfrak{h} \\
=& (\mathsf{CT}^*, \boldsymbol{r}, \mathsf{Ext}(\mathfrak{h}'(\boldsymbol{r}), \boldsymbol{x}') - \mathfrak{h}(\boldsymbol{k}')) && \text{Homomorphism of } \mathsf{Ext} \\
=& (\mathsf{CT}^*, \mathfrak{h}'^{-1}(\boldsymbol{r}), \mathsf{Ext}(\boldsymbol{r}, \boldsymbol{x}') - \mathfrak{h}(\boldsymbol{k}')) && \text{Change of variable} \\
=& (\mathsf{CT}^*, \mathfrak{h}'^{-1}(\boldsymbol{r}), \mathsf{Ext}(\boldsymbol{r}, \boldsymbol{x} + \boldsymbol{k}') - \mathfrak{h}(\boldsymbol{k}')) && \text{Change of variable}
\end{aligned}
$$

Via a hybrid argument, we can switch all the adversary's KDM queries to the form in the last equation. However, as $\mathsf{Ext}(\boldsymbol{r}, \boldsymbol{x} + \boldsymbol{k}') - \mathfrak{h}(\boldsymbol{k}')$ still depends on the secret key $\boldsymbol{x}$, we cannot follow the prior proof technique in [8,10,12–14,16], which requires to simulate the KDM queries without using the secret key. To handle this, we observe that now the adversary's view of his Q queries is of the form $\left\{(\mathsf{CT}_i^*, \mathfrak{h}_i'^{-1}(\boldsymbol{r}_i), \mathsf{Ext}(\boldsymbol{r}_i, \boldsymbol{x} + \boldsymbol{k}_i') - \mathfrak{h}(\boldsymbol{k}_i'))\right\}_{i \in [Q]}$. We can then leverage the security of the extractor to switch these outputs of the extractor to uniformly random strings at one shot. Since CT_i^* can be generated given $\boldsymbol{k}_i'$ (the third additional property of wHPS), and Ext is secure against correlated-source attacks (even

given $\boldsymbol{k}_i'$'s), we can prove that $\{\mathsf{PKE}_1.\mathsf{Enc}(\mathfrak{h}_i(\mathsf{sk}))\}_{i\in[Q]}$ is indistinguishable from random via a simple reduction from the required extractor. We refer the details to Section 5.2.

Improving Information Rate. The information rate of the above scheme is $\frac{w}{|\mathsf{CT}|+|\boldsymbol{r}|+w}$, where w denotes the length of the output of extractor. In our instantiations of the extractor, we have $|\mathsf{CT}| > |\boldsymbol{k}|\lambda$ and $|\boldsymbol{k}| \geq w$, and thus $|\mathsf{CT}|$ dominates the denominator, resulting in the ratio at most $O(1/\lambda)$. To improve the rate, we use the same CT (encapsulation) and repeatedly extract from the same source with different seeds when encrypting many different messages. That is, we consider the following scheme PKE_2, where $\mathsf{PKE}_2.\mathsf{pk}$ and $\mathsf{PKE}_2.\mathsf{sk}$ are the same as the above PKE_1. The encryption algorithm is modified as below:

$$\begin{aligned}&\mathsf{PKE}_2.\mathsf{Enc}((m_1, m_2, \ldots, m_t))\\ &= (\mathsf{CT}, \boldsymbol{r}_1, \mathsf{Ext}(\boldsymbol{r}_1, \boldsymbol{k}) + m_1, \boldsymbol{r}_2, \mathsf{Ext}(\boldsymbol{r}_2, \boldsymbol{k}) + m_2, \ldots, \boldsymbol{r}_t, \mathsf{Ext}(\boldsymbol{r}_t, \boldsymbol{k}) + m_t)\,.\end{aligned}$$

By using the same proof idea of PKE_1, we can show that suppose the reusable extractor is homomorphic with respect to linear functions and secure against correlated-source attacks, then the scheme PKE_2 is $\mathsf{KDM}^{(1)}$-secure with respect to affine functions. In this case, the information rate would be $\frac{wt}{|\mathsf{CT}|+|\boldsymbol{r}|t+wt}$, approaching $\frac{w}{|\boldsymbol{r}|+w}$ for sufficiently large t.

However, in our both LWE and DDH instantiations, $w \ll |\boldsymbol{r}|$, and thus the rate is still far from the optimal. To tackle this issue, we use a parallel repetition of the source, i.e., let $\boldsymbol{K} = (\boldsymbol{k}_1, \boldsymbol{k}_2, \ldots, \boldsymbol{k}_d)$, and define

$$\mathsf{Ext}_{||}(\boldsymbol{r}, \boldsymbol{K}) = (\mathsf{Ext}(\boldsymbol{r}, \boldsymbol{k}_1), \mathsf{Ext}(\boldsymbol{r}, \boldsymbol{k}_2), \ldots, \mathsf{Ext}(\boldsymbol{r}, \boldsymbol{k}_d))\,.$$

We can show that suppose $\boldsymbol{K}$ forms a block source, then the output of $\mathsf{Ext}_{||}$ will be computationally indistinguishable from random. Moreover, $\mathsf{Ext}_{||}$ is as well homomorphic and secure against correlated-source attacks for appropriate classes. By using $\mathsf{Ext}_{||}$, we can still derive $\mathsf{KDM}^{(1)}$ security, for a slightly weaker class of *block*-affine functinos. Now, the information rate would be $\frac{wd}{|\boldsymbol{r}|+wd}$, approaching $1 - o(1)$ by setting d such that $|\boldsymbol{r}| = o(wd)$.

1.2.3 Achieving Arbitrary Polynomial $\bar{n}$

Next, we discuss how to upgrade the above framework to achieve $\mathsf{KDM}^{(\bar{n})}$-security for an unbounded polynomial $\bar{n}$. Before presenting our approach, we first abstract some important features from the above schemes PKE_1 and PKE_2 – (1) the schemes are based on BE as the most underlying tool, and (2) they have the following features: (a) the secret key is just a vector $\boldsymbol{x}$, and (b) the public key has the form $(\mathsf{CRS}, \mathsf{H}(\mathsf{CRS}, \boldsymbol{x}))$, where H denotes the projection function $\mathsf{KeyGen}(\cdot,\cdot)$ of BE in [16]. We call this type of schemes as BE-based public key encryption scheme.

Next we generalize the idea of [12], showing that if a BE-based scheme satisfies certain key and ciphertext homomorphic properties, then one can prove $\mathsf{KDM}^{(\bar{n})}$-security from $\mathsf{KDM}^{(1)}$ by the following two steps: Let Π be a BE-based PKE.

1. First we define an intermediate scheme $\Pi^{\bar{n}}$ that runs $\bar{n}$ times the encryption algorithm of Π to encrypt the same message, with $\bar{n}$ distinct public parameters but corresponding to the same secret key. Particularly, $\Pi^{\bar{n}}.\mathsf{sk} = \Pi.\mathsf{sk} = \boldsymbol{x}$, and $\Pi^{\bar{n}}.\mathsf{pk} = (\Pi.\mathsf{pk}_1, \ldots, \Pi.\mathsf{pk}_{\bar{n}})$, where $\Pi.\mathsf{pk}_i = (\mathsf{CRS}_i, h_i = \mathsf{H}(\mathsf{CRS}_i, \boldsymbol{x}))$ for all $i \in [\bar{n}]$. The encryption algorithm works as follows:

$$\Pi^{\bar{n}}.\mathsf{Enc}(m) = (\Pi.\mathsf{Enc}(\mathsf{pk}_1, m), \Pi.\mathsf{Enc}(\mathsf{pk}_2, m), \ldots, \Pi.\mathsf{Enc}(\mathsf{pk}_{\bar{n}}, m)).$$

2. Then we show, if $\Pi^{\bar{n}}$ is $\mathsf{KDM}^{(1)}$-secure with respect to affine functions, then Π is $\mathsf{KDM}^{(\bar{n})}$-secure with respect to affine functions.

Thus, to show that PKE_2 is $\mathsf{KDM}^{(\bar{n})}$ secure, it suffices to show that its intermediate scheme, i.e., $\mathsf{PKE}_2^{\bar{n}}$, is $\mathsf{KDM}^{(1)}$ secure.

However, the current instantiation of the underlying BE [16] can only derive $\mathsf{KDM}^{(1)}$ security of $\mathsf{PKE}_2^{\bar{n}}$ for a bounded polynomial $\bar{n}$. When $\bar{n}$ becomes too large, $\mathsf{PKE}_2^{\bar{n}}$ may completely loses security. As each $h_i = \mathsf{H}(\mathsf{CRS}_i, \boldsymbol{x})$ leaks some small information of the secret $\boldsymbol{x}$, thus the secret might have no entropy given too many hashes in the pk_i's. Even worse, in the LWE-based instantiation of [16], one can obtain $\boldsymbol{x}$ given only n (the dimension of $\boldsymbol{x}$) hashes of h_i's by simply solving linear equations. This approach seems to hit an entropy barrier, inherently.

To tackle this challenge, we propose a new pseudorandom property of BE (and BE-based PKE) by adding *reusability* to the projection function H. Particularly, the reusable property requires that the following two distributions are indistinguishable, even in conjunction with the reusable extractor against correlated-source attacks for any $\bar{n}, m = \mathsf{poly}(\lambda)$:

$$\left(\{\mathsf{CRS}_i, \mathsf{H}(\mathsf{CRS}_i, \boldsymbol{x})\}_{i\in[\bar{n}]}, \{\boldsymbol{r}_j, \mathsf{Ext}(\boldsymbol{r}, \mathfrak{h}_j(\boldsymbol{x}))\}_{j\in[m]}\right) \approx_c \left(\{\mathsf{CRS}_i, u_i\}_{i\in[\bar{n}]}, \{\boldsymbol{r}_j, u'_j\}_{j\in[m]}\right)$$

Conceptually, this would guarantee secrecy of $\boldsymbol{x}$ even if the adversary can obtain many hashes on the same $\boldsymbol{x}$ and samples from the reusable extractor (under correlated-source attacks).

As a result, by using a BE with this reusable property as the underlying building block of wHPS, we are able to show that $\mathsf{PKE}_2^{\bar{n}}$ is $\mathsf{KDM}^{(1)}$-secure for any $\bar{n} = \mathsf{poly}(\lambda)$, implying that PKE_2 is $\mathsf{KDM}^{(\bar{n})}$-secure for any $\bar{n} = \mathsf{poly}(\lambda)$.

New BE Constructions. To instantiate the required BE, we observe that the CDH-based scheme in [16] as is, can achieve the reusability property if DDH is further assumed. However, the LWE-based scheme becomes insecure if n hashes are given to the adversaries as we stated before, where n is the dimension of $\boldsymbol{x}$. To solve this, we design a new projection function H' that makes a simple yet essential modification of the original H of [16]. Particularly, $\mathsf{H}'(\mathsf{CRS}, \boldsymbol{x}) = \mathsf{H}(\mathsf{CRS}, \boldsymbol{x}) + \boldsymbol{e}$, for some appropriate noise $\boldsymbol{e}$. In this way, the distribution $(\mathsf{CRS}, \mathsf{H}'(\mathsf{CRS}, \boldsymbol{x}))$ in this modified BE would be a sample of LWE, which is pseudorandom even when polynomially many samples are given, and can be used in conjunction with the LWE-based reusable extractor. This enables us to achieve $\mathsf{KDM}^{(\bar{n})}$-PKE for any unbounded polynomial $\bar{n}$ with optimal information rate with respect to affine functions.

Amplification. We first notice that the class of block-affine functions does not contain all projection functions, so the generic technique of Applebaum [7] does not apply to amplify the class. Nevertheless, we show that this class can still be used to encode the labels of Garbled Circuits (a common realization of randomized encoding), and thus we can amplify the class to be any bounded-sized boolean circuits.

As our scheme can encrypt messages of an indefinite length, we are able to further achieve KDM function class for $(\mathcal{F}_s||\mathcal{Q}^\tau)$, where $\mathcal{F}_s$ is the class of circuits up to sized s, $\mathcal{Q}^\tau$ is the class of affine functions with τ-element outputs, and $(\mathcal{F}_s||\mathcal{Q}^\tau)$ denotes the concatenation of two classes, i.e., every function $\mathfrak{f}$ in the class can be represented by $\mathfrak{f} = (\mathfrak{h}, \mathfrak{q})$ for some $\mathfrak{h} \in \mathcal{F}_s$ and $\mathfrak{q} \in \mathcal{Q}^\tau$ such that $\mathfrak{f}(\mathsf{sk}) = (\mathfrak{h}(\mathsf{sk})||\mathfrak{q}(\mathsf{sk}))$. For the parameter range $\tau \gg s$, our scheme achieves the optimal information rate, i.e., $1 - o(1)$.

1.2.4 Upgrade to KDM-IBE

The above framework can be further generalized to construct IBE with KDM-security and leakage resilience. Particularly, we design a new compiler that uses an $\mathsf{IB\text{-}wHPS}$ to amply the on-the-fly KDM-security (a new notion) of PKE into KDM-security of IBE, and simultanesouly the resulting IBE achieves leakage resilience. This improves the compiler of [31], which might not be leakage resilient.

Our compiler is straight-forward. Let Π be a BE-based PKE and $\mathsf{IB\text{-}wHPS}$ be an identity-based wHPS that has additional structures: (1) the secret key has the structure $\mathsf{sk}_{\mathsf{id}} = (\boldsymbol{x}, \mathsf{sk}_{\mathsf{id},\boldsymbol{x}})$, (2) $\mathsf{IB\text{-}wHPS.Decap}(\mathsf{sk}_{\mathsf{id}}, \mathsf{CT}^*) = \boldsymbol{x} + \boldsymbol{k}$, and (3) given $\boldsymbol{k}$, the above CT^* can be simulated faithfully. (This is similar to the additional structures of our required wHPS above). Then we can design an $\mathsf{IBE}.\{\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}\}$ as follows. $\mathsf{IBE}.\{\mathsf{Setup}, \mathsf{KeyGen}\}$ and $\mathsf{IBE}.\{\mathsf{mpk}, \mathsf{msk}, \mathsf{sk}_{\mathsf{id}}\}$ are the same as those of $\mathsf{IB\text{-}wHPS}$. To encrypt a message m with an id, $\mathsf{IBE.Enc}$ first generates an encapsulation $(\mathsf{CT}_1, \boldsymbol{k}) \leftarrow \mathsf{IB\text{-}wHPS.Encap}(\mathsf{mpk}, \mathsf{id})$, then generates $\mathsf{pk} = (\mathsf{CRS}, \mathfrak{h}(\mathsf{CRS}, \boldsymbol{k}))$ from the BE, and then computes $\mathsf{CT}_2 \leftarrow \Pi.\mathsf{Enc}(\mathsf{pk}, m)$. The resulting ciphertext would be $(\mathsf{CT}_1, \mathsf{pk}, \mathsf{CT}_2)$.

Next we present a simple case that demonstrates the key idea of our KDM-security proof. Consider the simple case of only one KDM query, i.e., an encryption for some message $\mathfrak{f}(\mathsf{sk}_{\mathsf{id}}) = \mathfrak{f}(\boldsymbol{x}, \mathsf{sk}_{\mathsf{id},\boldsymbol{x}})$ (by Structure 2 of $\mathsf{IB\text{-}wHPS}$) with respect to some id. We can derive the following:

$$
\begin{aligned}
&\mathsf{IBE.Enc}(\mathfrak{f}(\boldsymbol{x}, \mathsf{sk}_{\mathsf{id},\boldsymbol{x}})) \\
=&(\mathsf{CT}_1, \mathsf{CRS}, \mathsf{H}(\mathsf{CRS}, \boldsymbol{k}), \mathsf{CT}_2) \\
\approx_c&(\mathsf{CT}_1^*, \mathsf{CRS}, \mathsf{H}(\mathsf{CRS}, \mathsf{IB\text{-}wHPS.Decap}(\mathsf{CT}_1^*)), \mathsf{CT}_2) && \text{Valid/Invalid Ciphertext Indistinguishability} \\
=&(\mathsf{CT}_1^*, \mathsf{CRS}, \mathsf{H}(\mathsf{CRS}, \boldsymbol{x} + \boldsymbol{k}'), \mathsf{CT}_2) && \text{By Structure 1} \\
=&\Big(\mathsf{CT}_1^*, \mathsf{CRS}, \mathsf{H}(\mathsf{CRS}, \boldsymbol{x}'), \Pi.\mathsf{Enc}\big(\mathsf{CRS}, \mathsf{H}(\mathsf{CRS}, \boldsymbol{x}'), \\
&\qquad \mathfrak{f}((\boldsymbol{x}' - \boldsymbol{k}'), \mathsf{sk}_{\mathsf{id},(\boldsymbol{x}'-\boldsymbol{k}')})\big)\Big) && \text{Change of Variable} \\
=&\Big(\mathsf{CT}_1^*, \mathsf{CRS}, \mathsf{H}(\mathsf{CRS}, \boldsymbol{x}'), \Pi.\mathsf{Enc}\big(\mathsf{CRS}, \mathsf{H}(\mathsf{CRS}, \boldsymbol{x}'), \mathfrak{g}(\boldsymbol{x}')\big)\Big) && \text{(*) Explain Below} \\
\approx_c&\Big(\mathsf{CT}_1^*, \mathsf{CRS}, \mathsf{H}(\mathsf{CRS}, \boldsymbol{x}'), \Pi.\mathsf{Enc}\big(\mathsf{CRS}, \mathsf{H}(\mathsf{CRS}, \boldsymbol{x}'), u\big)\Big) && \mathsf{KDM} \text{ of } \Pi
\end{aligned}
$$

We observe that if $\mathfrak{f}((\boldsymbol{x}' - \boldsymbol{k}'), \mathsf{sk}_{(\boldsymbol{x}'-\boldsymbol{k}')})$ can be expressed as $\mathfrak{g}(\boldsymbol{x}')$ and the underlying Π is KDM-secure with respect to the function $\mathfrak{g}$, then the resulting IBE is KDM-secure with respect to $\mathfrak{f}$. We further identify that the equation (*) holds even if the master secret key msk of IB-wHPS is given. Thus, we can hardcode msk and $\boldsymbol{k}'$ and randomness r into $\mathfrak{g}$ and set the function as follow: $\mathfrak{g}_{\mathsf{msk},\boldsymbol{k}',r}(\boldsymbol{x}')$ first computes $\mathsf{sk}_{\mathsf{id},\boldsymbol{x}'-\boldsymbol{k}'} = \mathsf{IB\text{-}wHPS.KeyGen}(\mathsf{msk}, \mathsf{id}, \boldsymbol{x}' - \boldsymbol{k}'; r)$ and then outputs $\mathfrak{f}((\boldsymbol{x}' - \boldsymbol{k}'), \mathsf{sk}_{\mathsf{id},(\boldsymbol{x}'-\boldsymbol{k}')})$. We can instantiate Π by using the above mentioned schemes PKE_1 or PKE_2 and the bootstrapping technique of Applebaum [7]. In this way, we can obtain a KDM-secure PKE with respect to the class of bounded polynomial circuits, which includes the required $\mathfrak{g}$.

The above idea cannot be trivially extended to the general case where there are many KDM queries. A simple reason is that pk needs to be generated on-the-fly for each ciphertext. This does not match the traditional notion of KDM-security for PKE. To handle this technical issue, we propose a new notion called *on-the-fly* KDM-security, where there is no pk upfront, and the adversary receives an on-the-fly $\mathsf{pk} = (\mathsf{CRS}, \mathsf{H}(\mathsf{CRS}, \boldsymbol{x}'))$ with respect to the same secret key $\boldsymbol{x}'$ upon each KDM query. By using this on-the-fly KDM-PKE with the IB-wHPS, we are able to achieve KDM-IBE. Moreover, we can prove that the above PKE_2 satisfies the on-the-fly notion. We refer details to Section 10.

1.3 Reading Map

We first present the necessary mathematical notations, cryptographical definitions and related lemmas in Section 2. In Section 3, we first define a new variant of (computational) randomness extractors, which serve as the most important tools of this paper. And then, we instantiate the required extractors based on LWE or DDH, respectively. In Section 4, we first identify several new important structures of wHPS, and then show an instantiation of the required wHPS from BE. In section 5, we show that a weak hash proof system with the additional structure as Definition 4.1 can be used to obtain a public-key encryption scheme that is simultaneously leakage resilient and KDM secure. Then, in Section 6, we show how to upgrade our Construction 5.3 to achieve $\mathsf{KDM}^{(\bar{n})}$-security for an unbounded polynomial $\bar{n}$. Furthermore, in Section 7, we show how to instantiate the required BE in Section 6.3. Moreover, in Section 8, after putting all above things together, we conclude that we can get PKE that is leakage resilient against block leakage and $\mathsf{KDM}^{(\bar{n})}$-secure w.r.t. *block*-affine functions for any unbounded polynomial $\bar{n}$, from DDH or LWE. In Sections 9 and 10, we further extend our above results in Section 8 in two directions: the first one is to enlarge the class of KDM functions via Garbled Circuits; the second one is to generalize our results to the setting of IBE. Besides, in Section 11, we instantiate the generic construction of IBE proposed in Section 10.

2 Preliminaries

In this section, we first introduce several standard mathematical notations for our constructions, then present necessary definitions and related lemmas.

2.1 Notations

In this paper, $\mathbb{N}$, $\mathbb{Z}$ and $\mathbb{R}$ denote the sets of natural numbers, integers and real numbers, respectively. We use λ to denote the security parameter, which is the implicit input for all algorithms presented in this paper. A function $\mathfrak{f}(\lambda) > 0$ is negligible and denoted by $\mathsf{negl}(\lambda)$ if for any $c > 0$ and sufficiently large λ, $\mathfrak{f}(\lambda) < 1/\lambda^c$. A probability is called to be overwhelming if it is $1 - \mathsf{negl}(\lambda)$. A column vector is denoted by a bold lower case letter (e.g., $\boldsymbol{x}$). A matrix is denoted by a bold upper case letter (e.g., $\mathbf{A}$). For a vector $\boldsymbol{x}$, its Euclidean norm (also known as the ℓ_2 norm) is defined to be $\|\boldsymbol{x}\| = (\sum_i x_i^2)^{1/2}$. For a matrix $\mathbf{A}$, its ith column vector is denoted by $\boldsymbol{a}_i$ and its transposition is denoted by $\mathbf{A}^\top$, and its component in i-th row and j-th column is denoted by $A_{i,j}$. The Euclidean norm of a matrix is the norm of its longest column: $\|\mathbf{A}\| = \max_i \|\boldsymbol{a}_i\|$.

For a set D, we denote by $u \xleftarrow{\$} D$ the operation of sampling a uniformly random element u from D, and represent $|u|$ as the bit length of u. For an integer $\ell \in \mathbb{N}$, we use U_ℓ to denote the uniform distribution over $\{0,1\}^\ell$. Given a randomized algorithm or function $\mathfrak{f}(\cdot)$, we use $y \xleftarrow{\$} \mathfrak{f}(x)$ to denote y as the output of $\mathfrak{f}$ and x as input. For a distribution X, we denote by $x \xleftarrow{\$} X$ the operation of sampling a random x according to the distribution X. Given two different distributions X and Y over a countable domain D, we can define their statistical distance to be $\Delta(X,Y) = \frac{1}{2}\sum_{d\in D} |X(d) - Y(d)|$, and say that X and Y are $\Delta(X,Y)$ close. Moreover, if $\Delta(X,Y)$ is negligible in λ, we say that the two distributions are statistically close, which is always denoted by $X \overset{s}{\approx} Y$. If for any PPT algorithm $\mathcal{A}$ that $\left|\Pr[\mathcal{A}(1^\lambda, X) = 1] - \Pr[\mathcal{A}(1^\lambda, Y) = 1]\right|$ is negligible in λ, then we say that the two distributions are computationally indistinguishable, denoted by $X \overset{c}{\approx} Y$.

2.2 KDM-Security and Leakage Resilience of PKE Scheme

Definition 2.1 (KDM-secure PKE in [7, 8, 12]) *Let $\Pi_{\mathsf{PKE}} = \mathsf{PKE}.\{\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}\}$ be a PKE scheme,and let $\mathcal{F}$ be a class of functions $\mathcal{F} := \{f : \mathcal{SK}^n \to \mathcal{M}\}$, where $n > 0$ is an integer, $\mathcal{SK}$ and $\mathcal{M}$ denote the secret key space and the message space, respectively. Then the KDM-security with respect to $\mathcal{F}$ can be defined by the following experiment.*

Experiment $\mathbf{Exp}^{\mathcal{F}\text{-}\mathsf{KDM}}_{\mathsf{PKE},\mathcal{A}}(\lambda)$
Key Generation: The challenger selects a random bit $b \xleftarrow{\$} \{0,1\}$. It generates n pairs of $(\mathsf{pk}_1, \mathsf{sk}_1), \ldots, (\mathsf{pk}_n, \mathsf{sk}_n)$ by running $\mathsf{PKE.KeyGen}(1^\lambda)$ n times, and then sends $(\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$ to the advarsary $\mathcal{A}$.
KDM Queries: The adversary $\mathcal{A}$ repeatedly queries the challenger with $(i, f) \in [n] \times \mathcal{F}$, The challenger responds by setting $y = f(\mathsf{sk}_1, \ldots, \mathsf{sk}_n) \in \mathcal{M}$, and computing $c \xleftarrow{\$} \mathsf{PKE.Enc}(\mathsf{pk}_i, y)$ if $b = 0$ or $c \xleftarrow{\$} \mathsf{PKE.Enc}(\mathsf{pk}_i, 0^{\lvert f(\cdot)\rvert})$ if $b = 1$. Then the challenger returns c to the adversary $\mathcal{A}$.
Output: The adversary $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$.

Table 1. Security Experiment of KDM-secure PKE

We define the advantage of $\mathcal{A}$ in the above experiment to be

$$\mathbf{Adv}^{\mathcal{F}\text{-}\mathsf{KDM}}_{\mathsf{PKE},\mathcal{A}}(\lambda) = \left|\Pr[b = b'] - 1/2\right|.$$

A PKE *scheme is said to be* $\mathsf{KDM}^{(n)}$*-secure with respect to $\mathcal{F}$ if for any* PPT *adversary $\mathcal{A}$, we have* $\mathbf{Adv}^{\mathcal{F}\text{-}\mathsf{KDM}}_{\mathsf{PKE},\mathcal{A}}(\lambda) \leq \mathsf{negl}(\lambda)$.

Definition 2.2 (ℓ-Leakage-Resilient PKE) *A leakage-resilient* PKE *in the relative leakage model consists of three algorithms* $\mathsf{PKE}.\{\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}\}$*, which are parameterized by a security parameter λ and a leakage parameter ℓ. In particular, its ℓ-leakage-resilient security can be defined by the following experiment.*

Experiment $\mathbf{Exp}^{\mathsf{LR}}_{\mathsf{PKE},\mathcal{A}}(\lambda, \ell)$
Key Generation: The challenger $\mathcal{C}$ gets a pair of $(\mathsf{pk}, \mathsf{sk})$ by running $\mathsf{PKE.KeyGen}(1^\lambda, 1^\ell)$, and sends pk to $\mathcal{A}$.
Leakage Queries: The adversary $\mathcal{A}$ queries the challenger $\mathcal{C}$ with a leakage function $f : \{0,1\}^* \to \{0,1\}^\ell$, and gets $f(\mathsf{sk})$ from $\mathcal{C}$.
Challenge Stage: $\mathcal{A}$ chooses two message $\mu_0, \mu_1 \in \mathcal{M}$ and sends them to $\mathcal{C}$. Then $\mathcal{C}$ selects $b \xleftarrow{\$} \{0,1\}$ and computes $c_b \xleftarrow{\$} \mathsf{PKE.Enc}(\mathsf{pk}, \mu_b)$. Finally, $\mathcal{C}$ returns c_b to $\mathcal{A}$.
Output: The adversary $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$.

Table 2. Security Experiment of Leakage-Resilient PKE

We define the advantage of $\mathcal{A}$ in the above experiment to be

$$\mathbf{Adv}^{\mathsf{LR}}_{\mathsf{PKE},\mathcal{A}}(\lambda, \ell) = \left|\Pr[b = b'] - 1/2\right|.$$

A PKE *scheme is said to be ℓ-leakage resilient if for any* PPT *adversary $\mathcal{A}$, we have* $\mathbf{Adv}^{\mathsf{LR}}_{\mathsf{PKE},\mathcal{A}}(\lambda, \ell) \leq \mathsf{negl}(\lambda)$*, and the leakage rate of the scheme is* $\frac{\ell}{|\mathsf{sk}|}$.

Remark 2.3 *Furthermore, we define more general case of leakage on block source secret, i.e., block leakage. More formally, for the case that* $\mathsf{sk} := S =$

$(S_1, \ldots, S_m)$ is an $m \times e$ block source as in Definition 5.1, we define leakage functions $f_i : \{0,1\}^* \to \{0,1\}^\ell$ independently for each block S_i with all $i \in [m]$. We say $(f_1, \ldots, f_m)$ are block leakage functions, if the min-entropy of S_i is still large enough even given leakage $(f_1(S_1), \ldots, f_{i-1}(S_{i-1}))$ for any $i \in [m]$. Clearly, when $m = 1$, this is the trivial case in Definition 2.2. Here, we call $\frac{m\ell}{|\mathsf{sk}|}$ the block leakage rate of the corresponding scheme.

Below, we present the syntax of two important tools – *batch encryption* and *weak hash proof systems*, which are important for our constructions.

2.3 Batch Encryption (BE)

Definition 2.4 (Batch Encryption in [16]) *A batch encryption (BE) scheme consists of the following four algorithms $\{\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}\}$:*

- $\mathsf{Setup}(1^\lambda, 1^n)$*: The algorithm takes as input the security parameter λ and key length n, and outputs a common reference string CRS which includes a parameter $B = B(\lambda, n)$.*
- $\mathsf{KeyGen}(\mathsf{CRS}, \boldsymbol{x})$*: Given a common reference string CRS and the secret key $\boldsymbol{x} \in \mathbb{Z}_B^n$ as input, the algorithm projects the secret key $\boldsymbol{x}$ to a public key h.*
- $\mathsf{Enc}(\mathsf{CRS}, h, \mathbf{M})$*: Given a common reference string CRS, a public key h, and a message matrix $\mathbf{M} = (M_{i,j})_{i \in [n], j \in \mathbb{Z}_B} \in \mathbb{Z}_B^{n \times B}$ as input, the algorithm outputs a ciphertext CT.*
- $\mathsf{Dec}(\mathsf{CRS}, \boldsymbol{x}, \mathsf{CT})$*: Given a common reference string CRS, a secret key $\boldsymbol{x}$, and a ciphertext CT as input, the algorithm outputs a message vector $\boldsymbol{m}' = (M_{i,x_i})_{i \in [n]}$.*

Remark 2.5 *Let $\hat{\ell}$ denote the bit-length of the public key h. Then we notice that given the public key pk, the conditional min-entropy of sk is $H_\infty(\mathsf{sk}|\mathsf{pk}) = H_\infty(\boldsymbol{x}|h) \geq n \log B - \hat{\ell}$.*

Correctness of Batch Encryption. The correctness of a batch encryption scheme is defined as follows.

Definition 2.6 (Correctness of BE) *For all $\mathsf{CRS} = \mathsf{Setup}(1^\lambda, 1^n)$, $\boldsymbol{x} = (x_i)_{i \in [n]} \in \mathbb{Z}_B^n$ and $\mathbf{M} = (M_{i,j})_{i \in [n], j \in \mathbb{Z}_B} \in \mathbb{Z}_B^{n \times B}$, $h = \mathsf{KeyGen}(\mathsf{CRS}, \boldsymbol{x})$, $\mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{CRS}, h, \mathbf{M})$, and $\boldsymbol{m}' = (m'_i)_{i \in [n]} = \mathsf{Dec}(\mathsf{CRS}, \boldsymbol{x}, \mathsf{CT})$, it holds that for all $i \in [n]$, $m'_i = M_{i,x_i}$ with probability at least $1 - 2^\lambda$ over the randomness of Enc.*

Semantic Security of Batch Encryption.

Definition 2.7 (Batch Encryption Security) *The security of a batch encryption scheme can be defined through the following game between a challenger and an adversary.*

We define the advantage of $\mathcal{A}$ in the above game to be

$$\mathbf{Adv}_{\mathcal{A}}^{\mathsf{BE}}(\lambda) = \left| \Pr[\mathcal{A} \text{ wins }] - 1/2 \right|.$$

Semantic security means that $\mathbf{Adv}_{\mathcal{A}}^{\mathsf{BE}}(\lambda) \leq \mathsf{negl}(\lambda)$ for any polynomial time $\mathcal{A}$.

Ciphertext Indistinguishability Experiment
Pre-stage: With the security parameter λ and the size value B as input, the adversary chooses and sends $n, \boldsymbol{x} \in \mathbb{Z}_B^n$ to the challenger.
Setup: The challenger generates $\mathsf{CRS} \leftarrow \mathsf{Setup}(\lambda, n)$, and sends CRS to the adversary.
Challenge Stage: The adversary chooses two matrices $\mathbf{M}^{(0)}, \mathbf{M}^{(1)} \in \mathbb{Z}_B^{n \times B}$ such that $M^{(0)}_{i,x_i} = M^{(1)}_{i,x_i}$, and sends these two matrices to the challenger.
Respond Stage: The challenger computes $h = \mathsf{KeyGen}(\mathsf{CRS}, \boldsymbol{x})$ and encrypt $\mathsf{CT} = \mathsf{Enc}(\mathsf{CRS}, h, \mathbf{M}^{(\beta)})$ for a random bit $\beta \in \{0,1\}$. Then, $\mathcal{C}$ sends CT to the adversary.
Output: The adversary outputs a bit $\beta' \in \{0,1\}$ and wins the game if $\beta = \beta'$.

Table 3. Security Experiment of Batch Encryption

Instantiations: BE can be constructed from LWE, LPN, and CDH [16].

2.4 Weak Hash Proof System (wHPS)

Definition 2.8 (Weak Hash Proof System in [28]) *A weak hash proof system (*wHPS*) with the encapsulated-key-space $\mathcal{K}$ consists of four algorithms* wHPS.{Setup,Encap,Encap*,Decap}* as follows. (We will omit* wHPS *when the context is clear).*

- **Key generation.** $\mathsf{Setup}(1^\lambda)$ *takes a security parameter λ as input, and generates a pair of public key and secret key* $(\mathsf{pk}, \mathsf{sk})$.
- **Valid encapsulation.** $\mathsf{Encap}(\mathsf{pk})$ *takes a public key* pk *as input, and outputs a valid ciphertext* CT *and its corresponding encapsulated key* $\mathsf{k} \in \mathcal{K}$.
- **Invalid encapsulation.** $\mathsf{Encap}^*(\mathsf{pk})$ *takes a public key* pk *as input, and outputs an invalid ciphertext* CT^*.
- **Decapsulation.** $\mathsf{Decap}(\mathsf{sk}, \mathsf{CT})$ *takes as input a secret key* sk *and ciphertext* CT*, and deterministically outputs* $\mathsf{k} \in \mathcal{K}$.

A (weak) hash proof system needs to satisfy the following three properties.

Correctness. For all $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{Gen}(\lambda)$, it holds

$$\Pr\left[\mathsf{k} = \mathsf{k}' \middle| (\mathsf{CT}, \mathsf{k}) \xleftarrow{\$} \mathsf{Encap}(\mathsf{pk}), \mathsf{k}' = \mathsf{Decap}(\mathsf{sk}, \mathsf{CT})\right] = 1.$$

Ciphertext Indistinguishability. For $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{Setup}(\lambda)$, $(\mathsf{CT}, \mathsf{k}) \xleftarrow{\$} \mathsf{Encap}(\mathsf{pk})$, and $\mathsf{CT}^* \xleftarrow{\$} \mathsf{Encap}^*(\mathsf{pk})$, it holds $(\mathsf{pk}, \mathsf{sk}, \mathsf{CT}) \overset{c}{\approx} (\mathsf{pk}, \mathsf{sk}, \mathsf{CT}^*)$. Namely, for any PPT adversary even given the secret key sk, a valid ciphertext CT sampled by Encap is still computationally indistinguishable from an invalid ciphertext CT^* sampled by Encap^*.

Universality. Except for the above mentioned correctness and indistinguishability, we need one additional information theoretic property. Particularly, for the adversary with public parameters, the decapsulation of an invalid ciphertext should have information entropy. Below, we define this property in the following way.

Definition 2.9 (Universal wHPS) *We say that a* wHPS *is* (ℓ, w)*-universal, if for* $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{Setup}(\lambda)$, $\mathsf{CT}^* \xleftarrow{\$} \mathsf{Encap}^*(\mathsf{pk})$, *and* $k^* = \mathsf{Decap}(\mathsf{CT}^*, \mathsf{sk})$, *it holds*

$$|k^*| = \ell \quad \text{and} \quad H_\infty(k^*|(\mathsf{pk}, \mathsf{CT}^*)) \geq w.$$

Namely, the decapsulated value $\mathsf{Decap}(\mathsf{sk}, \mathsf{CT}^*)$ *has bounded length and min-entropy over the encapsulated key space* $\mathcal{K}$*, where the randomness comes from the choice of* sk*. This implicitly means that there are many possible choices of* sk *for a fixed* pk*. Clearly,* w *should be a value between 0 and* ℓ*.*

Remark 2.10 *A weak hash proof system only requires the universal property to hold for random invalid ciphertexts, i.e.* $\mathsf{CT}^* \xleftarrow{\$} \mathsf{Encap}^*(\mathsf{pk})$*, instead of all possible ciphertexts (in the worst case manner). This is the main difference between weak hash proof system and standard hash proof system [19], which was originally designed for achieving CCA2 security. The weak hash proof system is not sufficient to achieve the CCA2 security, but nevertheless, can achieve leakage resilience as pointed out by [28].*

Furthermore, the notion of KDM-security, leakage resilience and wHPS can be easily generalized to the identity-based setting, resulting in the notion of KDM-secure IBE, Leakage-Resilient IBE and identity-based weak hash proof system in Section 2.5 and 2.6.

2.5 KDM-security and Leakage Resilience of IBE scheme.

Definition 2.11 (KDM-secure IBE by [3, 31]) *Let* Π *be an* IBE *scheme, and* $\mathcal{F}$ *be a function family. We define the* $\mathcal{F}$*-KDM-CPA game between a challenger and an adversary* $\mathcal{A}$ *as follows. Let* $\mathcal{SK}, \mathcal{ID}$*, and* $\mathcal{M}$ *be the user secret key space, identity space, and message space of* IBE*, respectively. Then the KDM security of* Π *with respect to* $\mathcal{F}$ *can be defined by the following experiment.*

Experiment $\mathbf{Exp}^{\mathcal{F}\text{-KDM}}_{\mathsf{IBE},\mathcal{A}}(\lambda)$

Key Generation: First, the challenger chooses a challenge bit $b \xleftarrow{\$} \{0,1\}$. Next, the challenger generates $(\mathsf{pp}, \mathsf{msk}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$ and sends pp to $\mathcal{A}$. Finally, the challenger prepares lists $L_{\mathsf{ext}}, L_{\mathsf{ch}}$, and L_{sk} all of which are initially empty.

KDM Queries: $\mathcal{A}$ may adaptively make the following three types of queries polynomiallymany times.

1. **Extraction queries**. $\mathcal{A}$ sends $\mathsf{id} \in \mathcal{ID} \backslash (L_{\mathsf{ext}} \cup L_{\mathsf{ch}})$ to the challenger. The challengerreturns $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$ to $\mathcal{A}$ and adds id to L_{ext}.
2. **Registration queries**. $\mathcal{A}$ sends $\mathsf{id} \in \mathcal{ID} \backslash (L_{\mathsf{ext}} \cup L_{\mathsf{ch}})$ to the challenger. The challenger generates $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$ and adds id to L_{ch} and $\mathsf{sk}_{\mathsf{id}}$ to L_{sk}.
3. **KDM queries**. $\mathcal{A}$ sends $(\mathsf{id}, f) \in L_{\mathsf{ch}} \times \mathcal{F}$ to the challenger. We require that f be a function such that $f : \mathcal{SK}^{|L_{\mathsf{ch}}|} \to \mathcal{M}$. If $b = 1$, the challenger returns $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{id}, f(\mathsf{sk}))$to $\mathcal{A}$. Otherwise, the challenger returns $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{id}, 0^{|f(\cdot)|})$ to $\mathcal{A}$.

Output: The adversary $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$.

Table 4. Security Experiment of KDM-secure IBE

In this game, we define the advantage of the adversary $\mathcal{A}$ as

$$\mathsf{Adv}^{\mathcal{F}\text{-KDM}}_{\mathsf{IBE},\mathcal{A}}(\lambda) = \left|\Pr[b = b'] - \frac{1}{2}\right|.$$

We say that IBE *is* $\mathcal{F}$-KDM *secure if for any* PPT *adversary* $\mathcal{A}$*, we have* $\mathsf{Adv}^{\mathcal{F}\text{-KDM}}_{\mathsf{IBE},\mathcal{A}}(\lambda) = \mathsf{negl}(\lambda)$.

Definition 2.12 (Leakage-Resilient IBE) *A leakage-resilient* IBE *with respect to identity secret keysin the relative leakage model consists of four algorithms* $\mathsf{IBE}.\{\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}\}$*, which are all parameterized by a security parameter λ and a leakage parameter ℓ. In particular, ℓ-leakage-resilient security can be defined by the following experiment.*

Experiment $\mathbf{Exp}^{\mathsf{ids\text{-}LR}}_{\mathsf{IBE},\mathcal{A}}(\lambda,\ell)$
Setup: The challenger $\mathcal{C}$ gets a pair of $(\mathsf{mpk},\mathsf{msk})$ by running $\mathsf{IBE.Setup}(1^\lambda)$, and sends mpk to $\mathcal{A}$.
Secret Key Query 1: $\mathcal{A}$ adaptively queries the challenger $\mathcal{C}$ with function $\mathsf{id} \in \mathcal{ID}$. For each query, $\mathcal{C}$ responds with $\mathsf{sk}_{\mathsf{id}}$.
Leakage Queries Stage: $\mathcal{A}$ adaptively queries the challenger $\mathcal{C}$ with pair h where $h : \{0,1\}^* \to \{0,1\}^\ell$ is a leakage function. The adversary gets $h(\mathsf{sk}_f)$ from $\mathcal{C}$.
Challenge Stage: $\mathcal{A}$ chooses the challenge identity $\mathsf{id}^* \in \mathcal{ID}$ and two message μ_0, $\mu_1 \in \mathcal{M}$ and sends them to $\mathcal{C}$, where id^* has not been queried before. Then $\mathcal{C}$ chooses $b \xleftarrow{\$} \{0,1\}$ and computes $c_b \xleftarrow{\$} \mathsf{IBE.Enc}(\mathsf{mpk}, x^*, \mu_b)$. Finally, $\mathcal{C}$ returns c_b to $\mathcal{A}$.
Secret Key Query 2: $\mathcal{A}$ adaptively queries the challenger $\mathcal{C}$ with $\mathsf{id} \in \mathcal{ID}$. Then $\mathcal{C}$ responds with $\mathsf{sk}_{\mathsf{id}}$ if $\mathsf{id} \neq \mathsf{id}^*$ and $\perp$ otherwise.
Output: The adversary $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$.

Table 5. Security Experiment of Leakage-Resilient IBE

We define the advantage of $\mathcal{A}$ in the above experiment to be

$$\mathbf{Adv}^{\mathsf{ids\text{-}LR}}_{\mathsf{IBE},\mathcal{A}}(\lambda,\ell) = |\Pr[b = b'] - 1/2|\,.$$

The scheme is ℓ-leakage resilient if for any PPT *adversary $\mathcal{A}$, we have* $\mathbf{Adv}^{\mathsf{ids\text{-}LR}}_{\mathsf{IBE},\mathcal{A}}(\lambda,\ell) \leq \mathsf{negl}(\lambda)$*, and the leakage rate of this* ABE *is* $\frac{\ell}{|\mathsf{sk}|}$.

Note that, this definition can be extended to the selective security with respect to identity, if the challenge identity is fixed advanced before seeing the master public-key, rather than chosen adaptively by the adversary. Also, in this paper, we only focus on leakage query to the identity secret key, but not the master secret key.

Similar to the case of PKE for Definition 2.2, we can also generalize this leakage for IBE to the case of block leakage. Hence, we call $\frac{\ell}{m|\mathsf{sk}|}$ the block leakage rate of the corresponding scheme.

2.6 Identity-Based Weak Hash Proof System

Definition 2.13 (IB-wHPS in [4]) *An identity-based weak hash proof system (*IB-wHPS*) with the encapsulated key space $\mathcal{K}$ consists of five algorithms* $\mathsf{IB\text{-}wHPS}.\{\mathsf{Setup},\mathsf{KeyGen},\mathsf{Encap},\mathsf{Encap}^*,\mathsf{Decap}\}$*:*

- **Setup.** $\mathsf{IB\text{-}wHPS.Setup}(1^\lambda)$ *takes a security parameter λ as input, and generates a pair of master public key and master secret key* $(\mathsf{mpk},\mathsf{msk})$*. The identity space $\mathcal{ID}$ and the encapsulated key space $\mathcal{K}$ are determined by* mpk*.*
- **Key generation.** $\mathsf{IB\text{-}wHPS.KeyGen}\ (\mathsf{id},\mathsf{msk})$ *takes as input an identity* $\mathsf{id} \in \mathcal{ID}$ *and the master secret key* msk*, and generates a secret key* $\mathsf{sk}_{\mathsf{id}}$*.*
- **Valid encapsulation.** $\mathsf{IB\text{-}wHPS.Encap}(\mathsf{mpk},\mathsf{id})$ *takes as input the master public key* mpk *and an identity* $\mathsf{id} \in \mathcal{ID}$*, and outputs a valid ciphertext* CT *and its corresponding encapsulated key* $\mathsf{k} \in \mathcal{K}$*.*

– **Invalid encapsulation.** $\mathsf{IB\text{-}wHPS.Encap}^*(\mathsf{mpk}, \mathsf{id})$ *takes as input the master public key* mpk *and an identity* $\mathsf{id} \in \mathcal{ID}$*, and outputs an invalid ciphertext* CT^*.
– **Decapsulation.** $\mathsf{IB\text{-}wHPS.Decap}(\mathsf{CT}, \mathsf{sk_{id}})$ *takes as input a secret key* $\mathsf{sk_{id}}$ *and a ciphertext* CT*, and deterministically outputs the encapsulated key* $\mathsf{k} \in \mathcal{K}$.

Similarly, an $\mathsf{IB\text{-}wHPS}$ needs to satisfy three properties: correctness, ciphertext indistinguishability, and smoothness.

Correctness. For $(\mathsf{mpk}, \mathsf{msk}) \xleftarrow{\$} \mathsf{IB\text{-}wHPS.Setup}(\lambda)$, any $\mathsf{id} \in \mathcal{ID}$, we have

$$\Pr\Big[\mathsf{k} = \mathsf{k}' \Big| \mathsf{sk_{id}} \xleftarrow{\$} \mathsf{IB\text{-}wHPS.KeyGen}(\mathsf{msk}, \mathsf{id}), \\ (\mathsf{CT}, \mathsf{k}) \xleftarrow{\$} \mathsf{IB\text{-}wHPS.Encap}(\mathsf{mpk}, \mathsf{id}), \mathsf{k}' = \mathsf{IB\text{-}wHPS.Decap}(\mathsf{CT}, \mathsf{sk_{id}})\Big] = 1.$$

Ciphertext Indistinguishability. The valid ciphertexts output by $\mathsf{IB\text{-}wHPS.Encap}$ and the invalid ciphertexts output by $\mathsf{IB\text{-}wHPS.Encap}^*$ are indistinguishable even given the identity secret key $\mathsf{sk_{id}}$. More formally, this indistinguishability is always described by the experiment between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ in the following Table 6.

Valid/Invalid Ciphertext Indistinguishability Experiment
Setup: The challenger $\mathcal{C}$ gets a pair of $(\mathsf{mpk}, \mathsf{msk})$ by running $\mathsf{IB\text{-}wHPS.Setup}(1^\lambda)$, and sends mpk to $\mathcal{A}$.
Test Stage 1: $\mathcal{A}$ adaptively queries the challenger $\mathcal{C}$ with $\mathsf{id} \in \mathcal{ID}$, and $\mathcal{C}$ responds with $\mathsf{sk_{id}}$.
Challenge Stage: $\mathcal{A}$ selects an arbitrary challenge identity $\mathsf{id}^* \in \mathcal{ID}$, and sends it to $\mathcal{C}$. $\mathcal{C}$ selects $b \xleftarrow{\$} \{0, 1\}$. If $b = 0$, $\mathcal{C}$ computes $(\mathsf{CT}, \mathsf{k}) \xleftarrow{\$} \mathsf{IB\text{-}wHPS.Encap}(\mathsf{mpk}, \mathsf{id}^*)$. If $b = 1$, $\mathcal{C}$ computes $\mathsf{CT} \xleftarrow{\$} \mathsf{AB\text{-}wHPS.Encap}^*(\mathsf{mpk}, \mathsf{id}^*)$. Then $\mathcal{C}$ returns CT to $\mathcal{A}$.
Test Stage 2: $\mathcal{A}$ adaptively queries the challenger $\mathcal{C}$ with $\mathsf{id} \in \mathcal{ID}$. Then $\mathcal{C}$ responds with $\mathsf{sk_{id}}$.
Output: $\mathcal{A}$ outputs a bit $b' \in \{0, 1\}$ as the output of the experiment. We say that $\mathcal{A}$ wins the experiment if $b = b'$.

Table 6. Ciphertext Indistinguishability of $\mathsf{IB\text{-}wHPS}$

We define the advantage of $\mathcal{A}$ in the above experiment to be

$$\mathbf{Adv}^{\mathsf{VI\text{-}IND}}_{\mathsf{IB\text{-}wHPS},\mathcal{A}}(\lambda) = \left|\Pr[\mathcal{A} \text{ wins}] - 1/2\right|.$$

Ciphertext indistinguishability means that $\mathbf{Adv}^{\mathsf{VI\text{-}IND}}_{\mathsf{IB\text{-}wHPS},\mathcal{A}}(\lambda) \leq \mathsf{negl}(\lambda)$.

Remark 2.14 *Similar to* IBE*, the ciphertext indistinguishability of an* IB-wHPS *can also be categorized into two types: selective security and adaptive security. The selective security requires the adversary to commit to the challenge identity before seeing the* mpk*. In contrast, adaptive security means the adversary can choose the challenge identity adaptively. Our definition can be easily twisted to the selective security setting.*

Universality. We say that an IB-wHPS is (ℓ, w)-universal, if for $(\mathsf{mpk}, \mathsf{msk}) \xleftarrow{\$} \mathsf{IB\text{-}wHPS.Setup}(\lambda)$, any $\mathsf{id} \in \mathcal{ID}$ and $\mathsf{k}^* = \mathsf{Decap}(\mathsf{CT}^*, \mathsf{sk}_{\mathsf{id}})$, it holds

$$|\mathsf{k}^*| = \ell \quad \text{and} \quad H_\infty(\mathsf{Decap}(\mathsf{CT}^*, \mathsf{sk}_{\mathsf{id}})|\mathsf{mpk}, \mathsf{CT}^*) \geq w.$$

where $\mathsf{CT}^* \xleftarrow{\$} \mathsf{IB\text{-}wHPS.Encap}^*(\mathsf{mpk}, \mathsf{id})$, and $\mathsf{sk}_{\mathsf{id}} \xleftarrow{\$} \mathsf{IB\text{-}wHPS.KeyGen}(\mathsf{id}, \mathsf{msk})$.

2.7 Useful Function Classes for KDM-Security

Additionally, we define the following function families that are useful for our results on KDM security.

Definition 2.15 (Linear, Affine and Shift Functions) *Let $\mathcal{X}, \mathcal{Y}$ be some additive groups. A function $\mathfrak{g} : \mathcal{X} \to \mathcal{Y}$ is linear if for every $x, x' \in \mathcal{X}$, we have $\mathfrak{g}(x + x') = \mathfrak{g}(x) + \mathfrak{g}(x')$; a function $\mathfrak{h} : \mathcal{X} \to \mathcal{Y}$ is affine if there exist a linear function $\mathfrak{g} : \mathcal{X} \to \mathcal{Y}$ and a constant $a \in \mathcal{Y}$ such that $\mathfrak{h}(x) = \mathfrak{g}(x) + a$ for every $x \in \mathcal{X}$. Moreover, a function $\mathfrak{s} : \mathcal{X} \to \mathcal{X}$ indexed by certain element $x \in \mathcal{X}$ is shift, if for every $x, x' \in \mathcal{X}$, we have $\mathfrak{s}_x(x') = x + x'$.*

Definition 2.16 *Let $\mathcal{X}, \mathcal{Y}$ be some additive groups. Given a class of linear functions $\mathcal{G} = \{\mathfrak{g} : \mathcal{X} \to \mathcal{Y}\}$, we define a related class of affine functions $\mathcal{G}^t = \{\mathfrak{g}' : \mathcal{X} \to \mathcal{Y}^t\}$ where each $\mathfrak{g}' \in \mathcal{G}^t$ can be indexed by a constant vector $\boldsymbol{a} = (a_1, \ldots, a_t)^\top \in \mathcal{Y}^t$ and t functions in $\mathcal{G}$, i.e., $\mathfrak{g}_1, \mathfrak{g}_2, \ldots, \mathfrak{g}_t \in \mathcal{G}$, such that for every $x \in \mathcal{X}$, $\mathfrak{g}'(x) = (\mathfrak{g}_1(x), \mathfrak{g}_2(x), \ldots, \mathfrak{g}_t(x))^\top + \boldsymbol{a} = (\mathfrak{g}_1(x) + a_1, \mathfrak{g}_2(x) + a_2, \ldots, \mathfrak{g}_t(x) + a_t)^\top$.*

Besides, if the underlying linear functions $\mathfrak{g} \in \mathcal{G}$ is a block function, i.e., each output component of $\mathfrak{g}$ depends only on one block of its input, then the resulting functions $\mathfrak{g}' \in \mathcal{G}^t$ are called block-affine function.

3 Randomness Extractor and Its Variants

In this section, we first define a new variant of (computational) randomness extractors, which serve as the most important tools of this paper. Then, we instantiate the required extractors based on LWE or DDH, respectively.

3.1 Our New Variant of Randomness Extractors

We require an extractor that is (1) reusable, (2) secure against correlated-source attacks, and (3) homomorphic. We present their definitions below.

Definition 3.1 (Reusable Extractor in [5]) *Let $\mathcal{X}, \mathcal{S}, \mathcal{Y}$ be efficient ensembles parameterized by the security parameter λ. An efficient function $\mathsf{Ext} : \mathcal{X} \times \mathcal{S} \to \mathcal{Y}$ is an (e,t)-reusable-extractor[10], if for any correlated random variables (s, aux) where s is over $\mathcal{S}$ and $H_\infty(s|\mathsf{aux}) \geq e$, the following two distributions are computationally (statistically) indistinguishable:*

$$(\mathsf{aux}, r_1, \ldots, r_t, \mathsf{Ext}(r_1, s), \ldots, \mathsf{Ext}(r_t, s)) \approx (\mathsf{aux}, r_1, \ldots, r_t, u_1, \ldots, u_t),$$

where the strings $\{r_i \xleftarrow{\$} \mathcal{X}\}$, $\{u_i \xleftarrow{\$} \mathcal{Y}\}$ are sampled independently.

If $e > t \log |\mathcal{Y}| + O(\log(1/\varepsilon))$ for some $\varepsilon = \mathsf{negl}(\lambda)$, we can construct an (e,t)-reusable extractor information theoretically, e.g,. Leftover hash lemma [21]. On the other hand for $e < t \log |\mathcal{Y}| + O(\log(1/\varepsilon))$, it is still possible to construct (e,t)-reusable extractor under appropriate computational assumptions, such as DDH or LWE [5, 36], for t being a bounded or even any arbitrary polynomial depending on the parameter settings.

Definition 3.2 (Reusable Extractor against Correlated-Source Attacks) *Let $\mathsf{Ext} : \mathcal{X} \times \mathcal{S} \to \mathcal{Y}$ be some function, and $\mathcal{F} = \{\mathfrak{f} : \mathcal{S} \to \mathcal{S}\}$ be some function class. We say Ext is an (e,t)-reusable extractor against correlated-source attacks with respect to $\mathcal{F}$, if for every random variables s, aux where s is over $\mathcal{S}$ and $\mathsf{H}_\infty(s|\mathsf{aux}) \geq e$, the following oracles, $O_s(\cdot)$ and $U(\cdot)$, are computationally indistinguishable given up to t queries:*

- *$O_s(\cdot)$: Take a function $\mathfrak{f} \in \mathcal{F}$ as input, sample a fresh random $r \leftarrow \mathcal{X}$, and return $(r, \mathsf{Ext}(r, \mathfrak{f}(s)))$ upon each query.*
- *$U(\cdot)$: Take a function $\mathfrak{f} \in \mathcal{F}$ as input, return a uniform sample $(r,u) \leftarrow (\mathcal{X}, \mathcal{Y})$ upon each query.*

Remark 3.3 *The above Definition 3.2 can also be described in the indistinguishability form – for any correlated random variables (s, aux) such that s is over $\mathcal{S}$ and $H_\infty(s|\mathsf{aux}) \geq e$, the following two distributions are computationally (statistically) indistinguishable: $\left(\mathsf{aux}, \{r_i, \mathsf{Ext}(r_i, \mathfrak{f}_i(s))\}_{i\in[t]}\right) \approx \left(\mathsf{aux}, \{r_i, u_i\}_{i\in[t]}\right)$, where the strings $\{r_i \xleftarrow{\$} \mathcal{X}\}_{i\in[t]}$, $\{u_i \xleftarrow{\$} \mathcal{Y}\}_{i\in[t]}$ are sampled independently, and $\{\mathfrak{f}_i \in \mathcal{F}\}_{i\in[t]}$ are chosen (adaptively) by any* PPT *adversary $\mathcal{A}$.*

Clearly, an (e,t)-reusable extractor is also one against correlated-source attacks with respect to the identity function.

Definition 3.4 (Homomorphic Extractor) *Let $\mathcal{Y}$ be a group associated with operation '$\circ$', $\mathsf{Ext} : \mathcal{X} \times \mathcal{S} \to \mathcal{Y}$ be an extractor following the syntax as in Definition 3.1, and $\mathcal{H} = \{\mathfrak{h} : \mathcal{S} \to \mathcal{Y}\}$ be some function class. We say that Ext is homomorphic with respect to $\mathcal{H}$, if for any function $\mathfrak{h} \in \mathcal{H}$, there exists an invertible function $\mathfrak{h}' : \mathcal{X} \to \mathcal{X}$ (efficiently computable given $\mathfrak{h}$) such that for any $x \in \mathcal{X}$ and $s \in \mathcal{S}$, we have $\mathsf{Ext}(x,s) \circ \mathfrak{h}(s) = \mathsf{Ext}(\mathfrak{h}'(x), s)$.*

[10] Here, t denotes the number of times the weak source being reused.

3.2 Instantiations from LWE and DDH

Definition 3.5 *For integers n and δ, we define the linear function class.*

- $\mathcal{SF}_{\delta,n} = \{\mathfrak{s}_{\boldsymbol{a}} : \mathbb{Z}_\delta^n \to \mathbb{Z}_\delta^n\}$ *where each function $\mathfrak{s}_{\boldsymbol{a}}$ is indexed by a vector $\boldsymbol{a} \in \mathbb{Z}_\delta^n$ such that $\mathfrak{s}_{\boldsymbol{a}}(\boldsymbol{x}) = \boldsymbol{x} + \boldsymbol{a} \bmod \delta$, for every $\boldsymbol{x} \in \mathbb{Z}_\delta^n$.*

Construction 3.6 (LWE-Based Extractor) *Let $\mathcal{X} = \mathbb{Z}_q^n$, $\mathcal{S} = \mathbb{Z}_2^n$ and $\mathcal{Y} = \mathbb{Z}_p$, where p is a prime and $p|q$. We define $\mathsf{Ext} : \mathcal{X} \times \mathcal{S} \to \mathcal{Y}$ as:*

$$\mathsf{Ext}(\boldsymbol{a}, \boldsymbol{s}) = \lfloor \langle \boldsymbol{a}, \boldsymbol{s} \rangle \mod q \rceil_{q,p},$$

where $\boldsymbol{a} \in \mathcal{X}, \boldsymbol{s} \in \mathcal{S}$ and $\lfloor \cdot \rceil_{q,p}$ is defined as the definition of LWR *in [9]. The construction has ratio $\frac{|\mathcal{Y}|}{|\mathcal{X}|} = \frac{\log p}{n \log q}$.*

Theorem 3.7 *Let λ be the security parameter, q, p, d, β, σ be parameters such that $q \geq p\beta\lambda^{\omega(1)}$, $\beta = \sigma\lambda^{\omega(1)}$, and $p|q$. Let χ be some σ-bounded distribution over $\mathbb{Z}_q^n$, $e \geq (d + \Omega(\lambda)) \log q$. Assuming the hardness of $\mathsf{LWE}_{d,q,\chi}$, then* Ext *in Construction 3.6 is an $(e, \ell = \mathsf{poly}(\lambda))$-reusable extractor against correlated-source attacks with respect to the function class $\mathcal{SF}_{2,n}$. Furthermore, this* Ext *is homomorphic with respect to the function class $\mathcal{G}_{p,n} = \{\mathfrak{g}_{\boldsymbol{b}} : \mathbb{Z}_2^n \to \mathbb{Z}_p\}$, where each function $\mathfrak{g}_{\boldsymbol{b}}$ is indexed by a vector $\boldsymbol{b} \in \mathbb{Z}_q^n$ such that $\mathfrak{g}_{\boldsymbol{b}}(\boldsymbol{x}) = \langle \boldsymbol{b}, \boldsymbol{x} \rangle \bmod p$, for every $\boldsymbol{x} \in \mathbb{Z}_2^n$.*

Proof. We start the proof by recalling a prior result of Alwen et al. [5], which states that LWE holds even if the secret comes from a weak source.

Claim 3.8 ([5]) *Let ψ_β be the discrete Gaussian distribution with width β, and the other parameters are as specified in this theorem. Then the following variant of $\mathsf{bin\text{-}LWE}_{n,q,\psi_\beta}$ holds for any polynomially number of samples, as long as the secret vector is binary, i.e., $\mathbb{Z}_2^n$, and has min-entropy e, i.e., the following two distributions are computationally indistinguishable for any $\ell = \mathsf{poly}(\lambda)$:*

$$(\mathsf{aux}, \mathbf{A}, \boldsymbol{s}^t \cdot \mathbf{A} + \boldsymbol{e}) \approx_c (\mathsf{aux}, \mathbf{A}, \boldsymbol{u}),$$

where $\boldsymbol{e} \leftarrow \psi_\beta^\ell$, $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times \ell}$, $\boldsymbol{u} \leftarrow \mathbb{Z}_q^\ell$, and $H_\infty(\boldsymbol{s}|\mathsf{aux}) \geq e$.

Next we would show that Ext is an $(e, \ell = \mathsf{poly}(\lambda))$-reusable extractor against correlated-source attacks with respect to $\mathcal{SF}_{2,n}$. In particular, we need to show that the two oracles $O_{\boldsymbol{s}}(\cdot)$ and $U(\cdot)$ are computationally indistinguishable given up to ℓ queries. To achieve this, we define an intermediate oracles below:
$\tilde{O}_{\boldsymbol{s}}(\cdot)$: on input a function $\mathfrak{s}_{\boldsymbol{b}} \in \mathcal{SF}_{2,n}$, the oracle samples a random $\boldsymbol{a} \in \mathbb{Z}_q^n$ and $\eta \leftarrow \psi_\beta$, and returns $(\boldsymbol{a}, \lfloor \langle \boldsymbol{a}, \boldsymbol{s} + \boldsymbol{b} \rangle \rceil + \eta \rceil_{q,p})$. Note, $(\boldsymbol{s} + \boldsymbol{b})$ is operated in $\mathbb{Z}_2^n$.

Below we will show that $O_{\boldsymbol{s}}(\cdot)$ is statistically indistinguishable from $\tilde{O}_{\boldsymbol{s}}(\cdot)$, which is computationally indistinguishable from $U(\cdot)$, up to ℓ queries.

Claim 3.9 *Given $\ell = \mathsf{poly}(\lambda)$ oracle queries, the distribution of $O_{\boldsymbol{s}}(\cdot)$ is statistically close to that of $\tilde{O}_{\boldsymbol{s}}(\cdot)$.*

Proof. We know η is sampled from ψ_β, so $|\eta| < \beta\sqrt{\ell}$ with $1-\mathsf{negl}(\lambda)$ probability. We also know that as long as $\langle \boldsymbol{a}, \boldsymbol{s}+\boldsymbol{b}\rangle$ does not fall within $\pm\beta\sqrt{\ell}$ of a multiple of $q/2$, we will have $\lfloor\langle \boldsymbol{a}, \boldsymbol{s}+\boldsymbol{b}\rangle) + \eta\rceil_{q,2} = \lfloor\langle \boldsymbol{a}, \boldsymbol{s}+\boldsymbol{b}\rangle)\rceil_{q,2}$.

Since $\boldsymbol{s}+\boldsymbol{b}$ operates in $\mathbb{Z}_2^n$, we know for any $\boldsymbol{s}+\boldsymbol{b} \neq \boldsymbol{0}$, the probability over random $\boldsymbol{a} \leftarrow \mathbb{Z}_q^n$ that $\langle \boldsymbol{a}, \boldsymbol{s}+\boldsymbol{b}\rangle$ falls within $\pm\beta\sqrt{\ell}$ of a multiple of $q/2$ is $2(2\beta\sqrt{\ell}+1)/q$. By the union bound, we have the probability that there exists a query that $\lfloor\langle \boldsymbol{a}, \boldsymbol{s}+\boldsymbol{b}\rangle+\eta\rceil_{q,p} \neq \lfloor\langle \boldsymbol{a}, \boldsymbol{s}+\boldsymbol{b}\rangle\rceil_{q,p}$ is upper bounded by $2\ell(2\beta\sqrt{\ell}+1)/q$, which is negligible in our setting of parameter.

Finally, the probability that there exists a query that $\boldsymbol{s}+\boldsymbol{b} = \boldsymbol{0}$ is upper bounded by $\ell 2^{-e}$, according to the min-entropy of $\boldsymbol{s}$ and the union bound. Clearly, this probability is negligible. This concludes the proof of the claim. □

Claim 3.10 *Assume the hardness of* $\mathsf{bin\text{-}LWE}_{n,q,\psi_\beta}$. *Then the oracle* $\tilde{O}_{\boldsymbol{s}}(\cdot)$ *and* $U(\cdot)$ *are computationally indistinguishable, given* $\ell = \mathsf{poly}(\lambda)$ *oracle queries.*

Proof. We prove this claim by reduction. Assume that there exists an adversary $\mathcal{A}$ that distinguishes $\tilde{O}_{\boldsymbol{s}}(\cdot)$ from $U(\cdot)$ using ℓ oracle queries with probability ε, then we would construct a reduction $\mathcal{B}$ that breaks $\mathsf{LWE}^*_{n,q,\psi_\beta}$ with probability ε, using ℓ samples.

$\mathcal{B}$ runs $\mathcal{A}$ as a subroutine, and is defined as follows.

- When $\mathcal{A}$ makes an oracle query with $\mathfrak{s}_{\boldsymbol{b}}$ for certain $\boldsymbol{b} \in \mathbb{Z}_2^n$, $\mathcal{B}$ queries the LWE challenge oracle and receives a sample $(\boldsymbol{a}, z)$.
- Then for $i \in [n]$,
  1. if $b_i = 0$, $\mathcal{B}$ sets $z'_i = 0$ and $a'_i = a_i$.
  2. if $b_i = 1$, $\mathcal{B}$ sets $z'_i = -a_i$ and $a'_i = -a_i$.
- $\mathcal{B}$ computes $z' = \lfloor \Sigma_{i=1}^n z'_i + z\rceil_{q,p}$ and sets $\boldsymbol{a}' = (a'_1, \ldots, a'_n)^\top$, and then forwards $(\boldsymbol{a}', z')$ to $\mathcal{A}$.
- $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs.

Clearly, if $(\boldsymbol{a}, z)$ comes from the uniform distribution, then $\mathcal{B}$ simulates faithfully the oracle $U(\cdot)$ to $\mathcal{A}$. On the other hand, if $(\boldsymbol{a}, z)$ comes from the LWE oracle, then $z' = \lfloor \Sigma_{i=1}^n z'_i + z\rceil_{q,2} = \lfloor \Sigma_{i=1}^n z'_i + \langle \boldsymbol{a}, \boldsymbol{s}\rangle + \eta\rceil_{q,2} = \lfloor \Sigma_{i=1}^n (z'_i + a_i s_i) + \eta\rceil_{q,2} = \lfloor \Sigma_{i=1}^n a'_i(s_i + b_i \mod 2) + \eta\rceil_{q,2} = \lfloor\langle \boldsymbol{a}', \mathfrak{g}_{\boldsymbol{b}}(\boldsymbol{s})\rangle + \eta\rceil_{q,2}$. This faithfully simulates the oracle $\tilde{O}_{\boldsymbol{s}}(\cdot)$. Thus, $\mathcal{B}$ has the same distinguishing probability as $\mathcal{A}$ does. □

Combining the above three claims, we prove that Ext is an $(e, \ell = \mathsf{poly}(\lambda))$-reusable extractor against the correlated-source attacks with respect to the class $\mathcal{SF}_{2,n}$.

Finally, we show the homomorphic property. $\mathcal{M}$ is associate with the addition operation over $\mathbb{Z}_p^m$. Given any function $\mathfrak{g}_{\boldsymbol{b}} \in \mathcal{G}$ indexed by $\boldsymbol{b} \in \mathbb{Z}_q^n$, we define the

function $\mathfrak{g}'_{\boldsymbol{b}}$ as follows: $\mathfrak{g}'_{\boldsymbol{b}}(\boldsymbol{x}) = \boldsymbol{x} + \frac{q}{p}\boldsymbol{b} \bmod q$ for any $\boldsymbol{x} \in \mathbb{Z}_p^n$, then

$$\begin{aligned}
\mathsf{Ext}(\boldsymbol{a}, \boldsymbol{s}) + \mathfrak{g}(\boldsymbol{s}) &= \lfloor \frac{p}{q}\langle \boldsymbol{a}, \boldsymbol{s}\rangle \rceil + \langle \boldsymbol{b}, \boldsymbol{s}\rangle \bmod p \\
&= \lfloor \frac{p}{q}\langle \boldsymbol{a}, \boldsymbol{s}\rangle \rceil + \lfloor \frac{p}{q} \cdot \frac{q}{p}\langle \boldsymbol{b}, \boldsymbol{s}\rangle \rceil \bmod p \\
&= \lfloor \frac{p}{q}\langle (\boldsymbol{a} + \frac{q}{p}\boldsymbol{b}), \boldsymbol{s}\rangle \rceil \\
&= \mathsf{Ext}(\boldsymbol{a} + \frac{q}{p}\boldsymbol{b}, \boldsymbol{s}) = \mathsf{Ext}(\mathfrak{g}'_{\boldsymbol{b}}(\boldsymbol{a}), \boldsymbol{s}).
\end{aligned}$$

This satisfies Definition 3.4. □

Construction 3.11 (DDH-Based Extractor) *Let $\mathbb{G}$ be a group of prime order q, $\mathcal{X} = \mathbb{G}^n$, $\mathcal{S} = \mathbb{Z}_2^n$, and $\mathcal{Y} = \mathbb{G}$. We define $\mathsf{Ext} : \mathcal{X} \times \mathcal{S} \to \mathcal{Y}$ as:*

$$\mathsf{Ext}(\boldsymbol{a}, \boldsymbol{s}) = \prod_{i=1}^{n} a_i^{s_i},$$

where $\boldsymbol{a} \in \mathcal{X}$, $\boldsymbol{s} \in \mathbb{Z}_2^n$. The construction has ratio $\frac{|\mathcal{Y}|}{|\mathcal{X}|} = \frac{1}{n}$.

Theorem 3.12 *Let λ be the security parameter, $\mathbb{G}$ be a group of prime order q. Assuming that DDH is hard with respect to the group $\mathbb{G}$ and $e \geq \log q + 2\log(1/\varepsilon)$ where $\varepsilon \in (0, 1)$ is negligible, then Ext defined as 3.11 is an $(e, t = \mathsf{poly}(\lambda))$-reusable extractor against correlated-source attacks with respect to the function class $\mathcal{SF}_{2,n}$. Furthermore, Ext is homomorphic with respect to the function class $\mathcal{G}'_{q,n}$, where each $\mathfrak{g} \in \mathcal{G}'_{q,n}$ is indexed by certain vector $\boldsymbol{b} \in \mathbb{G}^n$, i.e., $\mathfrak{g}_{\boldsymbol{b}}(\boldsymbol{s}) = \prod_{i=1}^{n} b_i^{s_i}$ for input $\boldsymbol{s} \in \mathbb{Z}_2^n$.*

Proof. We prove that the DDH-based Ext is an $(e, t = \mathsf{poly}(\lambda))$-reusable extractor against correlated-source attacks with respect to the function class $\mathcal{SF}_{q,n}$, in the following steps. (1) We recall that Ext is an $(e, t = \mathsf{poly}(\lambda))$-reusable extractor, which can be easily derived from the work of Naor and Segev [36], as stated in the work [5]. (2) We prove our desired statement via a reduction from (1), i.e., we give an efficient transformation that maps a sample $(\boldsymbol{a}, \mathsf{Ext}(\boldsymbol{a}, \boldsymbol{s}))$ to $(\boldsymbol{a}', \mathsf{Ext}(\boldsymbol{a}', \mathfrak{g}_{\boldsymbol{b}}(\boldsymbol{s})))$, and $(\boldsymbol{a}, u)$ to $(\boldsymbol{a}', u')$, for any $\mathfrak{g}_{\boldsymbol{b}} \in \mathcal{SF}_{2,n}$. Thus, if a PPT adversary can break the correlated-source security, then there exists an efficient reduction that breaks (1), reaching a contradiction.

The transformation takes input $(\boldsymbol{a}, z)$ and $\mathfrak{g}_{\boldsymbol{b}} \in \mathcal{SF}_{2,n}$ as input ($\boldsymbol{b} \in \mathbb{Z}_2^n$), and does the following:

- For $i \in [n]$, if $b_i = 0$, set $a'_i = a_i$ and $z'_i = 1$; Else if $b_i = 1$, set $a'_i = a_i^{-1}$ and $z'_i = a_i^{-1}$.
- Output $(\boldsymbol{a}', z \cdot \prod_{i=1}^{n} z'_i)$.

Clearly, the transformation maps the uniform distribution to the uniform distribution. On the other hand if $(\boldsymbol{a}, z = \prod_{i=1}^{n} a_i^{s_i})$, then the output would be of the

form $(\boldsymbol{a}', \prod_{i=1}^{n} a_i^{s_i} \cdot \prod_{i=1}^{n} z_i') = \left(\boldsymbol{a}', \prod_{i=1}^{n} a_i'^{(s_i+b_i \mod 2)}\right) = (\boldsymbol{a}', \mathsf{Ext}(\boldsymbol{a}', \mathfrak{g}_{\boldsymbol{b}}(\boldsymbol{s})))$.

Finally, we show the homomorphic property. $\mathcal{Y}$ is associated with multiplication operation over $\mathbb{G}$. Given any function $\mathfrak{g}_{\boldsymbol{b}} \in \mathcal{G}'_{q,n}$ indexed by $\boldsymbol{b} \in \mathbb{G}^n$, we set another function $\mathfrak{g}'_{\boldsymbol{b}}$ as follows: for any $\boldsymbol{x} \in \mathbb{G}^n$, $\mathfrak{g}'_{\boldsymbol{b}}(\boldsymbol{x}) = (x_1 b_1, \ldots, x_n b_n) \in \mathbb{G}^n$. It is clear that the function $\mathfrak{g}'_{\boldsymbol{b}}$ is invertible. Then for any $\boldsymbol{a} \in \mathbb{G}^n, \boldsymbol{s} \in \mathbb{Z}_2^n$ and $\mathfrak{g}_{\boldsymbol{b}} \in \mathcal{G}$, we have

$$\begin{aligned}
\mathsf{Ext}(\boldsymbol{a}, \boldsymbol{s}) \circ \mathfrak{g}_{\boldsymbol{b}}(\boldsymbol{s}) &= \prod_{i=1}^{n} a_i^{s_i} \cdot \prod_{i=1}^{n} b_i^{s_i} \\
&= \prod_{i=1}^{n} (a_i b_i)^{s_i} \\
&= \mathsf{Ext}(\mathfrak{g}'_{\boldsymbol{b}}(\boldsymbol{a}), \boldsymbol{s}).
\end{aligned}$$

This satisfies Definition 3.4. □

4 wHPS and Its Instantiation from Batch Encryption

In this section, we first identify several new important structures of wHPS, and then show an instantiation of the required wHPS from BE.

4.1 Additional Structure of wHPS

Definition 4.1 (wHPS with Additional Structures) *We say that Π is a* wHPS *with additional structures, if the following conditions hold:*

1. *Π satisfies all conditions for a* wHPS *defined in Definition 2.8;*
2. *The secret key,* sk*, of Π can be written as* $\mathsf{sk} := (\boldsymbol{a}, \mathsf{sk}_{\boldsymbol{a}}) \in \mathbb{Z}_m^n \times \{0,1\}^*$*, for certain positive integers $m, n \in \mathbb{Z}$. In particular,* $\mathsf{sk}_{\boldsymbol{a}} \in \{0,1\}^*$ *can be viewed as an arbitrary bit string, but is related to the prefix vector* $\boldsymbol{a} \in \mathbb{Z}_m^n$*.*
3. *The decapsulation of an invalid ciphertext,* $\mathsf{Decap}(\mathsf{sk}, \mathsf{CT}^*)$*, can be written as* $\mathfrak{s}_{\boldsymbol{k}'}(\boldsymbol{a}) = \boldsymbol{a} + \boldsymbol{k}' \bmod m$*, where the $\boldsymbol{a}$ is the first part of the secrete key* sk*, and $\boldsymbol{k}' \in \mathbb{Z}_m^n$ is the index vector related to the invalid ciphertext* CT^**.*
4. *Given some $\boldsymbol{k}' \in \mathbb{Z}_m^n$, one can generate* CT^* *such that* $\mathsf{Decap}(\mathsf{sk}, \mathsf{CT}^*) = \mathfrak{s}_{\boldsymbol{k}'}(\boldsymbol{a})$ *and the distribution of* CT^* *is identical to that of* $\mathsf{Encap}^*(\mathsf{pk})$*.*

Remark 4.2 *This additional structure can also be generalized to the notion of* IB-wHPS *in Definition 2.13. In particular, for the case of* $\mathsf{sk}_{\mathsf{id}} := (\boldsymbol{a}, \mathsf{sk}_{\boldsymbol{a},\mathsf{id}})$ *in the* IB-wHPS*,* $\mathsf{sk}_{\boldsymbol{a},\mathsf{id}}$ *is the output of an integrated algorithm* $\mathsf{IB\text{-}wHPS.KeyGen}(\mathsf{msk}, \mathsf{id}, \boldsymbol{a})$*, where* msk *denotes the master secret key.*

4.2 wHPS from BE

Construction 4.3 (Construction of wHPS from BE) *Let* $\Pi = \Pi.\{\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}\}$ *be a batch encryption scheme with the message space* $\mathbb{Z}_B^{n\times B}$*, the secret-key space* $\mathbb{Z}_B^n$ *and the projected public key size* $\hat{\ell}$*. Then, we construct a weak hash proof system* HPS *scheme* $\Pi_{\mathsf{wHPS}} = \Pi_{\mathsf{wHPS}}.\{\mathsf{Setup}, \mathsf{Encap}, \mathsf{Encap}^*, \mathsf{Decap}\}$ *with the same ciphertext space as* Π *and the encapsulated key space* $\mathcal{K} = \mathbb{Z}_B^n$ *as follows:*

- $\Pi_{\mathsf{wHPS}}.\mathsf{Setup}(1^\lambda)$*: The algorithm runs* $\mathsf{CRS} \xleftarrow{\$} \Pi.\mathsf{Setup}(1^\lambda, 1^n)$ *for an integer* $n \in \mathbb{N}$*, and then runs* $\Pi.\mathsf{KeyGen}(\mathsf{CRS}, \boldsymbol{x})$ *to generate* h *for a randomly chosen vector* $\boldsymbol{x} \in \mathbb{Z}_B^n$ *. Finally, the algorithm outputs* $\mathsf{pk} := (\mathsf{CRS}, h)$ *and* $\mathsf{sk} := \boldsymbol{x}$*.*
- $\Pi_{\mathsf{wHPS}}.\mathsf{Encap}(\mathsf{pk})$*: Given a public-key* pk *as input, the algorithm first chooses a random vector* $\boldsymbol{k} = (k_1, \ldots, k_n)^\top \in \mathbb{Z}_B^n$*, and set matrix* $\mathbf{M} = (M_{i,j})_{i\in[n], j\in\mathbb{Z}_B}$ *such that* $M_{i,j} = k_i$ *for every* $i \in [n], j \in \mathbb{Z}_B$*, i.e., all components in each row of* $\mathbf{M}$ *are the same. Then the algorithm runs* $\mathsf{CT} \xleftarrow{\$} \Pi.\mathsf{Enc}(\mathsf{CRS}, h, \mathbf{M})$*, and outputs* CT *and* $\boldsymbol{k}$ *as a valid ciphertext and its encapsulated key, respectively.*
- $\Pi_{\mathsf{wHPS}}.\mathsf{Encap}^*(\mathsf{pk})$*: Given a public-key* pk *as input, the algorithm chooses a random vector* $\boldsymbol{k} = (k_1, \ldots, k_n)^\top \in \mathbb{Z}_B^n$*, and set matrix* $\mathbf{M} = (M_{i,j})_{i\in[n], j\in\mathbb{Z}_B}$ *such that* $M_{i,j} = k_i + j \mod B$ *for every* $i \in [n], j \in \mathbb{Z}_B$*. (In this way, every element in a row is different from the others in the same row.) Then the algorithm runs* $\mathsf{CT}^* \xleftarrow{\$} \Pi.\mathsf{Enc}(\mathsf{CRS}, h, \mathbf{M})$*, and outputs* CT^* *as an invalid ciphertext.*
- $\Pi_{\mathsf{wHPS}}.\mathsf{Decap}(\mathsf{sk}, \mathsf{CT})$*: Given a ciphertext* CT *and a secret key* $\mathsf{sk} := \boldsymbol{x}$ *as input, the algorithm runs* $\boldsymbol{m}' = \Pi.\mathsf{Dec}(\mathsf{CRS}, \boldsymbol{x}, \mathsf{CT})$*, and outputs* $\boldsymbol{m}'$ *as the encapsulated key.*

It is clear that this construction of wHPS satisfies the additional structures in Definition 4.1. Moreover, the secret key of wHPS does not have the second part $\mathsf{sk}_{\boldsymbol{a}}$, which is one of our key points to prove the KDM security. Below we present the formal theorem and its proof.

Theorem 4.4 (wHPS from BE) *Suppose* Π *is a semantically secure batch encryption scheme with the message space* $\mathbb{Z}_B^{n\times B}$*, the secret-key space* $\mathbb{Z}_B^n$ *and the projected public key size* $\hat{\ell}$*. Then Construction 4.3 is an* $(n \log B, w)$*-universal weak hash proof system with the encapsulated key space* $\mathcal{K} = \mathbb{Z}_B^n$ *and* $w = n \log B - \hat{\ell}$*, and has the additional structure as Definition 4.1.*

Proof. According to the definition of a wHPS, we need to prove the following three properties: correctness, universality and ciphertext indistinguishability.

Correctness. Correctness of this wHPS follows directly from the correctness of the underlying BE.

Universality and the Additional Structure as Def 4.1. Given the public key pk and a random invalid ciphertext $\mathsf{CT}^* \xleftarrow{\$} \Pi.\mathsf{Enc}(\mathsf{CRS}, h, \mathbf{M})$, we have

$$\Pi_{\mathsf{wHPS}}.\mathsf{Decap}(\mathsf{sk}, \mathsf{CT}^*) = \Pi_{\mathsf{wHPS}}.\mathsf{Decap}(\boldsymbol{x}, \mathsf{CT}^*) = \boldsymbol{x} + \boldsymbol{k}',$$

where $\boldsymbol{k}'$ is the vector used to generate the invalid ciphertext. Clearly, this function is an efficiently computable and invertible permutation, i.e., the decryption function can be written as the permutation $\mathfrak{s}_{\boldsymbol{k}'}(\boldsymbol{x}) = \boldsymbol{x} + \boldsymbol{k}'$.

As this is an injective function of $\boldsymbol{x}$ (for any fixed $\boldsymbol{k}'$), the min-entropy of $\boldsymbol{x}$ remains the same after applying this function, i.e., $H_\infty(\mathsf{Decap}(\mathsf{sk}, \mathsf{CT}^*)|(h, \mathsf{CT}^*)) = H_\infty(\boldsymbol{x} + \boldsymbol{k}'|(h, \mathsf{CT}^*)) = H_\infty(\boldsymbol{x}|(h, \mathsf{CT}^*))$. Moreover, we note that given h, CT^* is independent of $\boldsymbol{x}$, so $H_\infty(\boldsymbol{x}|(h, \mathsf{CT}^*)) = H_\infty(\boldsymbol{x}|h)$. Therefore, we have

$$H_\infty(\boldsymbol{x}+\boldsymbol{k}|(h, \mathsf{CT}^*)) = H_\infty(\boldsymbol{x}|(h, \mathsf{CT}^*)) = H_\infty(\boldsymbol{x}|h) \geq H_\infty(\boldsymbol{x}) - |h| = n \log B - \hat{\ell}.$$

It is also clear from the argument that the scheme Π_{wHPS} satisfies the additional structure as Definition 4.1, i.e. the secret key sk has the structure $\boldsymbol{x} \in \mathbb{Z}_B^n$, and $\Pi_{\mathsf{wHPS}}.\mathsf{Decap}(\mathsf{sk}, \mathsf{CT}^*) = \boldsymbol{x} + \boldsymbol{k}'$, where $\boldsymbol{k}'$ is a vector related to the invalid ciphertext CT^*.

Ciphertext Indistinguishability. Directly from the security of BE, we can prove that the ciphertexts output by $\Pi_{\mathsf{wHPS}}.\mathsf{Encap}(\mathsf{pk})$ and $\Pi_{\mathsf{wHPS}}.\mathsf{Encap}^*(\mathsf{pk})$ are computationally indistinguishable, even given the secret key $\boldsymbol{x}$. Particularly, given the adversary $\mathcal{A}$ distinguishing valid and invalid ciphertexts successfully, we can directly use it to break the ciphertexs indistinguishability of BE. This is because, we directly use the BE ciphertexts encrypting different message as the valid/invalid ciphertexts of Π_{wHPS}. □

5 Generic construction PKE from wHPS

In this section, we show that a weak hash proof system with the additional structure as Definition 4.1 can be used to obtain a public-key encryption scheme that is simultaneously leakage resilient and KDM secure.

Before presenting our generic construction, we introduce a useful definition of block source, and a parallel repetition description of randomness extractor.

Definition 5.1 (Block Source [41]) *A random variable $S = (S_1, \ldots, S_m)$ is a $(e_1, \ldots, e_m)$ block source if for every $s_1, \ldots, s_{i-1}$, $S_i|_{S_1=s_1, S_2=s_2, \ldots, S_{i-1}=s_{i-1}}$ is a e_i-source, which means the conditional min-entropy of $S_i|_{S_1=s_1, S_2=s_2, \ldots, S_{i-1}=s_{i-1}}$ is e_i. If $e_1 = e_2 = \cdots = e_m = e$, then we call S an $m \times e$ block source.*

Definition 5.2 (Parallel Repetition of Extractor) *For any input $\boldsymbol{s} = (s_1, \ldots, s_m) \in \mathcal{S}^m$ and an underlying extractor $\mathsf{Ext} : \mathcal{R} \times \mathcal{S} \to \mathcal{Y}$, we use $\mathsf{Ext}_{||}(r, \boldsymbol{s}) = (\mathsf{Ext}(r, s_1), \ldots, \mathsf{Ext}(r, s_m))$ to denote a parallel repetition of extractor.*

Clearly, for a sufficient $m \times e$ block source, the output indistinguishability of $\mathsf{Ext}_{||}$ follows naturally from that of the underlying Ext.

Next, our generic construction of PKE can be derived from wHPS and Ext in the following way.

Construction 5.3 (PKE from wHPS and Ext) *Suppose that* $\Pi_{\mathsf{wHPS}} = \Pi_{\mathsf{wHPS}}.\{\mathsf{Setup}, \mathsf{Encap}, \mathsf{Encap}^*, \mathsf{Decap}\}$ *is a* wHPS *with the secret key space and the encapsulated key space being* $\mathcal{K} = \mathbb{Z}_B^n$ *with* $n = n' \cdot m$*. For convenience, we denote* $\mathcal{S} = \mathbb{Z}_B^{n'}$ *with* $\mathcal{K} = \mathcal{S}^m$*, and let* $\mathsf{Ext} : \mathcal{R} \times \mathcal{S} \to \mathcal{M}$ *is an* (e, poly)*-reusable extractor. Then, for any polynomial integer* t*, we define a public-key encryption scheme* $\Pi_{\mathsf{PKE}} = \Pi_{\mathsf{PKE}}.\{\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}\}$ *with message space* $\mathcal{M}^{t \times m}$ *as follows:*

- $\Pi_{\mathsf{PKE}}.\mathsf{KeyGen}(1^\lambda)$*: The algorithm runs* $(\mathsf{pk}^{\Pi_{\mathsf{wHPS}}}, \mathsf{sk}^{\Pi_{\mathsf{wHPS}}}) \xleftarrow{\$} \Pi_{\mathsf{wHPS}}.\mathsf{Setup}(1^\lambda)$*, and then outputs* $\mathsf{pk} := \mathsf{pk}^{\Pi_{\mathsf{wHPS}}}$ *and* $\mathsf{sk} := \mathsf{sk}^{\Pi_{\mathsf{wHPS}}}$*.*
- $\Pi_{\mathsf{PKE}}.\mathsf{Enc}(\mathsf{pk}, \boldsymbol{\mu})$*: Given a public-key* pk *and a message* $\boldsymbol{\mu} = (\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_t) \in \mathcal{M}^{t \times m}$ *as input with each* $\boldsymbol{\mu}_j \in \mathcal{M}^m$*, the algorithm runs* $\mathsf{wHPS}.\mathsf{Encap}$ *to generate* $(\mathsf{CT}_0, \boldsymbol{k}) \xleftarrow{\$} \Pi_{\mathsf{wHPS}}.\mathsf{Encap}(\mathsf{pk})$ *for* $\boldsymbol{k} \in \mathbb{Z}_B^n$*. The algorithm interprets* $\boldsymbol{k} \in (\mathbb{Z}_B^{n'})^m$*, and then samples* $r_j \xleftarrow{\$} \mathcal{R}$ *for* $j \in [t]$*. Furthermore, the algorithm computes and outputs* $\mathsf{CT} = (\mathsf{CT}_0, \mathsf{CT}_1, \ldots, \mathsf{CT}_t)$*, where*

$$\mathsf{CT}_j = (\mathsf{CT}_j^{(1)}, \mathsf{CT}_j^{(2)}) = (r_j, \mathsf{Ext}_{||}(r_j, \boldsymbol{k}) + \boldsymbol{\mu}_j), \text{ for } j \in [t].$$

- $\Pi_{\mathsf{PKE}}.\mathsf{Dec}(\mathsf{sk}, \mathsf{CT})$*: Given a ciphertext* $\mathsf{CT} = (\mathsf{CT}_0, \mathsf{CT}_1, \ldots, \mathsf{CT}_t)$ *and a secret key* sk *as input, the algorithm first computes* $\boldsymbol{k}' = \Pi_{\mathsf{wHPS}}.\mathsf{Decap}(\mathsf{sk}, \mathsf{CT}_0)$*, and then outputs* $\boldsymbol{\mu} = (\boldsymbol{\mu}'_1, \ldots, \boldsymbol{\mu}'_t)$*, where*

$$\boldsymbol{\mu}'_j = \mathsf{CT}_j^{(2)} - \mathsf{Ext}_{||}(\mathsf{CT}_j^{(1)}, \boldsymbol{k}').$$

Our construction achieves KDM security and leakage-resilience simultaneously. We summarize the results in the following theorem.

Theorem 5.4 *Assume that (1)* Π_{wHPS} *is a* $(n \log B, w)$*-universal* wHPS *with the secret key space and the encapsulated key space being* $\mathcal{K} = \mathbb{Z}_B^n$*,* $n = mn'$*,* $\mathcal{S} = \mathbb{Z}_B^{n'}$ *with* $\mathcal{K} = \mathcal{S}^m$*,* $w = n \log B - \hat{\ell}$*, where* $\hat{\ell}$ *denotes the bit length of* pk*, and* $n' \log B \geq \hat{\ell} + \lambda + e$*, (2)* Π_{wHPS} *has the additional structures as Def 4.1 and the secret key does not have the additional string* $\mathsf{sk}_{\boldsymbol{x}}$*, (3) the extractor* $\mathsf{Ext} : \mathcal{R} \times \mathcal{S} \to \mathcal{M}$ *is an* (e, poly)*-reusable extractor*[11]*, which is also homomorphic with respect to the class of linear functions* $\mathcal{G} : \{\mathfrak{g} : \mathbb{Z}_B^{n'} \to \mathcal{M}\}$ *and robust against correlated-source attacks with respect to the class of the shift functions* $\mathcal{SF}_{B,n'} : \{\mathfrak{s} : \mathbb{Z}_B^{n'} \to \mathbb{Z}_B^{n'}\}$*.Then the above scheme* Π_{PKE} *is*

[11] This means that for the individual source $\mathcal{S} = \mathbb{Z}_B^{n'}$ with sufficient entropy, the output of the underlying extractor Ext is indistinguishable from uniform. This further implies that for the block source with sufficient entropy, the output of the parallel repetition extractor $\mathsf{Ext}_{||}$ is still indistinguishable from uniform.

1. *leakage-resilient against block leakage*[12]*, with block leakage rate* $(1 - \frac{e+\hat{\ell}+\lambda}{n' \log B})$ *per block.*
2. $\mathsf{KDM}^{(1)}$*-secure with respect to the block-affine functionclass* $\mathcal{G}^t = \{\mathfrak{g}' : \mathbb{Z}_B^n \to \mathcal{M}^{m \times t}\}$ *as defined in Definition 2.16.*
3. *The information rate is* $\frac{|\mathcal{M}|mt}{|\mathsf{CT}_0|+|\mathcal{R}|t+|\mathcal{M}|mt}$*, where* $|\cdot|$ *denotes the bit description length of its elements. As a result, for large enough* t *and* m*, we obtain rate-1* KDM*-secure* PKE *scheme.*

Remark 5.5 *We note that any* wHPS *(without the additional structures) and reusable extractor (without the homomorphic and robust property) already suffice to prove leakage resilience, which detailed proof is deferred to Section 5.1 for the clarity of our presentation. The extra properties will be used for deriving* KDM *security, which will be formally presented in Sections 5.2 and 6. In Section 3.2, we have presented homomorphic extractors from* DDH *and* LWE*.*

5.1 Proof of Leakage-Resilience of Construction 5.3

In this section, we present the proof of the first part of Theorem 5.4.

As this Theorem follows from prior work [28], we just present a sketch of the hybrids in the proof for self-completeness.

Hybrid $\mathbf{H_0}$: This hybrid is defined to be the security experiment with ℓ block-leakage in Definition 2.2, with $\ell = (n' \log B - e - \hat{\ell} - \lambda)$. In this hybrid, the view of $\mathcal{A}$ consists of the public-key pk, leakage information $\mathfrak{l}(\mathsf{sk})$, and challenge ciphertext $\mathsf{CT} = (\mathsf{CT}_0, \mathsf{CT}_1, \ldots, \mathsf{CT}_t)$, where $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{wHPS.Setup}(1^\lambda)$, $r_i \xleftarrow{\$} \mathcal{R}$ for $i \in [t]$, $\boldsymbol{\mu}^{(b)} = (\boldsymbol{\mu}_1^{(b)}, \ldots, \boldsymbol{\mu}_t^{(b)}) \in \mathcal{M}^{m \times t}$, and

$$(\mathsf{CT}_0, \boldsymbol{k}) \xleftarrow{\$} \mathsf{wHPS.Encap}(\mathsf{pk}), \quad \mathsf{CT}_i = (\mathsf{CT}_i^{(1)}, \mathsf{CT}_i^{(2)}) = (r_i, \boldsymbol{\mu}_i^{(b)} + \mathsf{Ext}(r_i, \boldsymbol{k})),$$

for $i \in [t]$. Notice that the leakage function $\mathfrak{l} : \{0,1\}^* \to \{0,1\}^\ell$ is chosen adaptively by the adversary.

Hybrid $\mathbf{H_1}$: This hybrid is almost identical to the Hybrid 0, except the challenge ciphertext is computed in the following way:

$$(\mathsf{CT}_0, \boldsymbol{k}) \xleftarrow{\$} \mathsf{wHPS.Encap}(\mathsf{pk}), \quad \boldsymbol{k}_1 = \mathsf{wHPS.Decap}(\mathsf{sk}, \mathsf{CT}_0),$$

$$\mathsf{CT}_i = (\mathsf{CT}_i^{(1)}, \mathsf{CT}_i^{(2)}) = (r_i, \boldsymbol{\mu}_i^{(b)} + \mathsf{Ext}(r_i, \boldsymbol{k}_1)),$$

for $i \in [t]$. The only difference between Hybrid 0 and Hybrid 1 is the usage of $\boldsymbol{k}$ and $\boldsymbol{k}_1$ in the computation of CT_i. In fact, $\boldsymbol{k} = \boldsymbol{k}_1$ according to the correctness of the underlying wHPS. Hence, Hybrid 0 and Hybrid 1 are identical.

[12] Just as described in Remark 2.3, block leakage means that each block of source is leaked by an independent function and remain enough entropy conditioned on leakage against other blocks.

Hybrid $\mathbf{H_2}$: This hybrid is almost same to Hybrid 1, except the challenge ciphertext is computed in the following way:

$$\mathsf{CT}_0^* \xleftarrow{\$} \mathsf{wHPS.Encap}^*(\mathsf{pk}), \quad \boldsymbol{k}_1 = \mathsf{wHPS.Decap}(\mathsf{sk}, \mathsf{CT}_0^*),$$

$$\mathsf{CT}_i = (\mathsf{CT}_i^{(1)}, \mathsf{CT}_i^{(2)}) = (r_i, \boldsymbol{\mu}_i^{(b)} + \mathsf{Ext}(r_i, \boldsymbol{k}_1)),$$

for $i \in [t]$. The only difference between Hybrid 1 and Hybrid 2 is the computation and usage of CT_0 and CT_0^*. In fact, according to the ciphertext indistinguishability of the underlying wHPS, CT_0 and CT_0^* are computationally indistinguishable even for the adversary having secret key sk. Hence, Hybrid 1 and Hybrid 2 are indistinguishable for the adversary even holding the leakage information $\mathfrak{l}(\mathsf{sk})$.

Hybrid $\mathbf{H_3}$: This hybrid is almost same to Hybrid 2, except that the challenge ciphertext is computed in the following way:

$$\mathsf{CT}_0^* \xleftarrow{\$} \mathsf{wHPS.Encap}^*(\mathsf{pk}), \quad \boldsymbol{u}_i \xleftarrow{\$} \mathcal{M}^m, \quad \mathsf{CT}_i = (r_i, \boldsymbol{u}_i),$$

for $i \in [t]$. Essentially, pk, CT_0^*, $\boldsymbol{k}_1 = \mathsf{wHPS.Decap}(\mathsf{sk}, \mathsf{CT}_0^*)$ and $\mathfrak{l}(\mathsf{sk})$ are correlated variables. According to the universality of underlying wHPS, we know that the conditional min-entropy of $\boldsymbol{k}_1$ when given pk and CT_0^* is

$$H_\infty(\boldsymbol{k}_1 | (\mathsf{pk}, \mathsf{CT}_0^*)) \geq n \log B - \hat{\ell}.$$

By [41], we know that $\boldsymbol{k}_1$ is $2^{-\lambda}$ close to $(n' \log B - \hat{\ell} - \lambda) \times m$ block source. Furthermore, since the block-leakage $\mathfrak{l}(\mathsf{sk})$ has length $\ell = n' \log B - e - \hat{\ell} - \lambda$ per block, we have that $\boldsymbol{k}_1$ is an $m \times e$ block source, when given $\mathsf{aux} := (\mathsf{pk}, \mathsf{CT}_0^*, \mathfrak{l}(\mathsf{sk}))$. Since $\mathsf{Ext}_{||}$ is an $(m \times e, t)$-block source reusable-computational-extractor, we have

$$\begin{aligned} &(\mathsf{aux}, r_1, \ldots, r_t, \boldsymbol{\mu}_1^{(b)} + \mathsf{Ext}_{||}(r_1, \boldsymbol{k}_1), \ldots, \boldsymbol{\mu}_t^{(b)} + \mathsf{Ext}_{||}(r_t, \boldsymbol{k}_1)) \\ \approx_c &(\mathsf{aux}, r_1, \ldots, r_t, \boldsymbol{u}_1, \ldots, \boldsymbol{u}_t), \end{aligned}$$

As a result, Hybrid 2 and Hybrid 3 are computationally indistinguishable.

Notice that the view of $\mathcal{A}$ in Hybrid 3 is completely independent of $\boldsymbol{\mu}^{(b)}$ and b. Therefore, the advantage of $\mathcal{A}$ in Hybrid 3 is negligible. Finally, combining all the above hybrids together, we conclude that the advantage of $\mathcal{A}$ in Hybrid 0 is also negligible in λ. Thus the PKE scheme Π_{PKE} is ℓ-block-leakage-resilient, and the corresponding leakage rate is $(1 - \frac{e + \hat{\ell} + \lambda}{n' \log B})$.

5.2 Proof of $\mathsf{KDM}^{(1)}$-security of Construction 5.3

In this section, we present the proof of the second part of Theorem 5.4. Our proof takes the following high-level steps:

– We first define a modified encryption algorithm Enc', and then switch the responses of the KDM queries by using Enc' instead of the real Enc. By a hybrid argument, we argue that the adversary cannot distinguish whether he is answered by Enc or Enc'.
– We next modify the KDM responses by using Enc'', which essentially generates random strings as the ciphertexts. We argue that this is indistinguishable from the above case by the security of the reusable extractor robust against correlated-source attacks with respect to the class of shift functions (ref. Definition 2.15);
– Finally, we show that even given multiple KDM encryption queries, Enc'' is indistinguishable from $\mathsf{Enc}(0)$, implying KDM-security.

Below, we first define the modified encryption algorithm Enc'. On input a public-key pk, a secret-key $\mathsf{sk} := \boldsymbol{x} \in \mathbb{Z}_B^n = (\mathbb{Z}_B^{n'})^m$ and a function $\mathfrak{g}' \in \mathcal{G}^t$, where $\mathfrak{g}'$ can be indexed by a vector $\boldsymbol{a} = (\boldsymbol{a}_1^\top, \ldots, \boldsymbol{a}_t^\top)^\top \in \mathcal{M}^{m \times t}$ and t functions $\mathfrak{g}_1, \ldots, \mathfrak{g}_t \in \mathcal{G}$ (ref. Definition 2.16). Besides, for each $j \in [t]$, $\boldsymbol{a}_j = (a_{j,1}, \ldots, a_{j,m})^\top \in \mathcal{M}^m$, $\mathfrak{g}_j = (\mathfrak{g}_{j,1}, \ldots, \mathfrak{g}_{j,m})$ with $\mathfrak{g}_{j,l} : \mathbb{Z}_B^{n'} \to \mathcal{M}$ and $l \in [m]$, the algorithm does the following:

1. Generate an invalid ciphertext CT_0^*. By Property 4 in Definition 4.1, set $\boldsymbol{x}' := \mathsf{Decap}(\mathsf{sk}, \mathsf{CT}_0^*) = \boldsymbol{x} + \boldsymbol{k}'$ for some $\boldsymbol{k}'$.
2. Compute $t \cdot m$ invertible functions $\{\mathfrak{h}_{1,l}\}_{l\in[m]}, \ldots, \{\mathfrak{h}_{t,l}\}_{l\in[m]}$ such that $\mathsf{Ext}_{||}(r, \boldsymbol{s}) + \mathfrak{g}_j(\boldsymbol{s}) = (\mathsf{Ext}(r, \boldsymbol{s}_1), \ldots, \mathsf{Ext}(r, \boldsymbol{s}_m)) + (\mathfrak{g}_{j,1}(\boldsymbol{s}_1), \ldots, \mathfrak{g}_{j,m}(\boldsymbol{s}_m)) = (\mathsf{Ext}(\mathfrak{h}_{j,1}(r), \boldsymbol{s}_1), \ldots, \mathsf{Ext}(\mathfrak{h}_{j,m}(r), \boldsymbol{s}_m))$ for any $j \in [t]$, by the property of homomorphic extractor (ref. Definition 3.4). Here, $\boldsymbol{s}$ is a block source, i.e., $\boldsymbol{s} = (\boldsymbol{s}_1, \ldots, \boldsymbol{s}_m)$.
3. Then sample t random seeds $r_1, \ldots, r_t \in \mathcal{R}$ for the extractor, and compute $z_j = \{\mathsf{Ext}(\mathfrak{h}_{j,l}(r_j), \boldsymbol{x}'_l) - \mathfrak{g}_{j,l}(\boldsymbol{k}'_l) + a_{j,l}\}_{l\in[m]}$ for $j \in [t]$, where $\boldsymbol{x}' = (\boldsymbol{x}'_l)_{l\in[m]}$ and $\boldsymbol{k}' = (\boldsymbol{k}'_l)_{l\in[m]}$.
4. Output the ciphertext CT': $\big(\mathsf{CT}_0^*, r_1, z_1, \ldots, r_t, z_t\big)$.

Then, we define the other modified encryption algorithm Enc'':

1. Generate an invalid ciphertext CT_0^*.
2. Then for each $j \in [t]$, sample $r_j \xleftarrow{\$} \mathcal{R}$ and $z_j \xleftarrow{\$} \mathcal{M}^m$;
3. Output the ciphertext CT'': $\big(\mathsf{CT}_0^*, r_1, z_1, \ldots, r_t, z_t\big)$.

Furthermore, we define a series of hybrids as follows:

– **Hybrid** H_0: This hybrid is identical to the original KDM queries case, i.e. the responses of all the Q KDM queries are generated as the real encryptions of the $\mathfrak{g}'^{(i)}(\mathsf{sk})$ for $i \in [Q]$.
– **Hybrid** $\mathsf{H}_{0.i}$ for each $i \in [Q]$: Upon receiving the first i KDM queries, this hybrid uses Enc' to reply and then generates the remaining KDM responses according to the original encryption algorithm as H_0.
– **Hybrid** H_1: This hybrid replies all KDM queries with Enc''.
– **Hybrid** H_2: This hybrid replies all KDM queries with $\mathsf{Enc}(0)$.

Let events $\mathcal{E}_0$, $\mathcal{E}_1$, $\mathcal{E}_2$ denote that the KDM adversary $\mathcal{A}$ outputs 1 in H_0, H_1, and H_2, respectively. Similarly, we define events $\mathcal{E}_{0.i}$. To show that $\mathsf{Pr}[\mathcal{E}_0] \approx \mathsf{Pr}[\mathcal{E}_2]$, we will take the following path:

$$\mathsf{Pr}[\mathcal{E}_0] \approx \mathsf{Pr}[\mathcal{E}_{0.1}] \approx \cdots \approx \mathsf{Pr}[\mathcal{E}_{0.Q}] \approx \mathsf{Pr}[\mathcal{E}_1] \approx \mathsf{Pr}[\mathcal{E}_2].$$

We note that proving indistinguishability of H_1 and H_2 follows essentially the same idea from proving its semantic security. This can be captured in the proof of leakage resilience (ref. Subsection 5.1), so we just omit the proof to avoid repetition. For notational convenience, we define $\mathsf{H}_{0.0} := \mathsf{H}_0$.

Finally, we use the following three lemmas to accomplish the above mentioned proof idea.

Lemma 5.6 *For $i \in [Q]$, $\left|\mathsf{Pr}[\mathcal{E}_{0.i-1}] - \mathsf{Pr}[\mathcal{E}_{0.i}]\right| \leq \mathsf{negl}(\lambda)$, assuming the ciphertext indistinguishability of the underlying* wHPS.

Proof. We prove this lemma by contradiction. Assume that for some i and adversary $\mathcal{A}$, we have $\left|\mathsf{Pr}[\mathcal{E}_{0,i-1}] - \mathsf{Pr}[\mathcal{E}_{0,i}]\right| = \varepsilon$ for some non-negligible ε. Then we are going to construct a reduction $\mathcal{B}$ that breaks the ciphertext indistinguishability of the underlying wHPS with the same non-negligible advantage ε. This reaches a contradiction.

Let $\mathsf{CT}_0^{*(i)}$ be the invalid wHPS ciphertext generated in the i-th KDM response by Enc', such that $\boldsymbol{x}'_i := \mathsf{Decap}(\mathsf{sk}, \mathsf{CT}_0^{*(i)}) = \boldsymbol{x} + \boldsymbol{k}'_i$. Then the overall ciphertext with respect to KDM function $\mathfrak{g}'_i \in \mathcal{G}^t$ in this round can be re-written as

$$\begin{aligned}
&\Big(\mathsf{CT}_0^{*(i)}, r_{i,1}, \{\mathsf{Ext}(\mathfrak{h}_{i,1,l}(r_{i,1}), \boldsymbol{x}'_{i,l}) - \mathfrak{g}_{i,1,l}(\boldsymbol{k}'_{i,l}) + a_{i,1,l}\}_{l\in[m]}, \\
&\qquad \ldots, r_{i,t}, \{\mathsf{Ext}(\mathfrak{h}_{i,t,l}(r_{i,t}), \boldsymbol{x}'_{i,l}) - \mathfrak{g}_{i,t,l}(\boldsymbol{k}'_{i,l}) + a_{i,t,l}\}_{l\in[m]}\Big) \\
=&\Big(\mathsf{CT}_0^{*(i)}, r_{i,1}, \{\mathsf{Ext}(r_{i,1}, \boldsymbol{x}'_{i,l}) + \mathfrak{g}_{i,1,l}(\boldsymbol{x}'_{i,l}) - \mathfrak{g}_{i,1,l}(\boldsymbol{k}'_{i,l}) + a_{i,1,l}\}_{l\in[m]}, \\
&\qquad \ldots, r_{i,t}, \{\mathsf{Ext}(r_{i,t}, \boldsymbol{x}'_{i,l}) + \mathfrak{g}_{i,t,l}(\boldsymbol{x}'_{i,l}) - \mathfrak{g}_{i,t,l}(\boldsymbol{k}'_{i,l}) + a_{i,t,l}\}_{l\in[m]}\Big) \qquad (1) \\
=&\Big(\mathsf{CT}_0^{*(i)}, r_{i,1}, \{\mathsf{Ext}(r_{1,1}, \boldsymbol{x}'_{i,l}) + \mathfrak{g}_{i,1,l}(\boldsymbol{x}'_{i,l} - \boldsymbol{k}'_{i,l}) + a_{i,1,l}\}_{l\in[m]}, \\
&\qquad \ldots, r_{i,t}, \{\mathsf{Ext}(r_{i,t}, \boldsymbol{x}'_{i,l}) + \mathfrak{g}_{i,t,l}(\boldsymbol{x}'_{i,l} - \boldsymbol{k}'_{i,l}) + a_{i,t,l}\}_{l\in[m]}\Big) \qquad (2) \\
=&\Big(\mathsf{CT}_0^{*(i)}, r_{i,1}, \{\mathsf{Ext}(r_{i,1}, \mathsf{Decap}(\boldsymbol{x}, \mathsf{CT}_0^{*(i)})_l) + \mathfrak{g}_{i,1,l}(\boldsymbol{x}_l) + a_{i,1,l}\}_{l\in[m]}, \\
&\qquad \ldots, r_{i,t}, \{\mathsf{Ext}(r_{i,t}, \mathsf{Decap}(\boldsymbol{x}, \mathsf{CT}_0^{*(i)})_l) + \mathfrak{g}_{i,t,l}(\boldsymbol{x}_l) + a_{i,t,l}\}_{l\in[m]}\Big). \qquad (3) \\
=&\Big(\mathsf{CT}_0^{*(i)}, r_{i,1}, \mathsf{Ext}_{||}(r_{i,1}, \mathsf{Decap}(\boldsymbol{x}, \mathsf{CT}_0^{*(i)})) + \mathfrak{g}_{i,1}(\boldsymbol{x}) + \boldsymbol{a}_{i,1}, \\
&\qquad \ldots, r_{i,t}, \mathsf{Ext}_{||}(r_{i,t}, \mathsf{Decap}(\boldsymbol{x}, \mathsf{CT}_0^{*(i)})) + \mathfrak{g}_{i,t}(\boldsymbol{x}) + \boldsymbol{a}_{i,t}\Big). \qquad (4)
\end{aligned}$$

Here, we use $\mathsf{Decap}(\boldsymbol{x}, \mathsf{CT}_0^{*(i)})_l \in \mathbb{Z}_B^{n'}$ to denote the l-th block of $\mathsf{Decap}(\boldsymbol{x}, \mathsf{CT}_0^{*(i)}) \in (\mathbb{Z}_B^{n'})^m$. Similarly, $\boldsymbol{x}_l \in \mathbb{Z}_B^{n'}$ denote the l-th block of $\boldsymbol{x} \in (\mathbb{Z}_B^{n'})^m$.

Then we rewrite the real KDM ciphertext as

$$\begin{aligned}
&\Big(\mathsf{CT}_0^{(i)}, r_{i,1}, \{\mathsf{Ext}(r_{i,1}, \boldsymbol{k}_{i,l}) + \mathfrak{g}_{i,1,l}(\boldsymbol{x}_l) + a_{i,1,l}\}_{l\in[m]},\\
&\qquad \ldots, r_{i,t}, \{\mathsf{Ext}(r_{i,t}, \boldsymbol{k}_{i,l}) + \mathfrak{g}_{i,t,l}(\boldsymbol{x}_l) + a_{i,t,l}\}_{l\in[m]}\Big)\\
=&\Big(\mathsf{CT}_0^{(i)}, r_{i,1}, \{\mathsf{Ext}(r_{i,1}, \mathsf{Decap}(\boldsymbol{x}, \mathsf{CT}_0^{(i)})_l) + \mathfrak{g}_{i,1,l}(\boldsymbol{x}_l) + a_{i,1,l}\}_{l\in[m]},\\
&\qquad \ldots, r_{i,t}, \{\mathsf{Ext}(r_{i,t}, \mathsf{Decap}(\boldsymbol{x}, \mathsf{CT}_0^{(i)})_l) + \mathfrak{g}_{i,t,l}(\boldsymbol{x}_l) + a_{i,t,l}\}_{l\in[m]}\Big) \qquad (5)\\
=&\Big(\mathsf{CT}_0^{(i)}, r_{i,1}, \mathsf{Ext}_{||}(r_{i,1}, \mathsf{Decap}(\boldsymbol{x}, \mathsf{CT}_0^{(i)})) + \mathfrak{g}_{i,1}(\boldsymbol{x}) + \boldsymbol{a}_{i,1},\\
&\qquad \ldots, r_{i,t}, \mathsf{Ext}_{||}(r_{i,t}, \mathsf{Decap}(\boldsymbol{x}, \mathsf{CT}_0^{(i)})) + \mathfrak{g}_{i,t}(\boldsymbol{x}) + \boldsymbol{a}_{i,t}\Big) \qquad (6)
\end{aligned}$$

Similarly, we use $\mathsf{Decap}(\boldsymbol{x}, \mathsf{CT}_0^{(i)})_l \in \mathbb{Z}_B^{n'}$ to denote the l-th block of $\mathsf{Decap}(\boldsymbol{x}, \mathsf{CT}_0^{(i)}) \in (\mathbb{Z}_B^{n'})^m$.

The equation (1) follows from the homomorphic property of the underlying extractor Ext, where each function $\mathfrak{h}_{i,j,l}$ is generated according to the function $\mathfrak{g}_{i,j,l}$ for $i \in [Q]$, $j \in [t]$, $l \in [m]$. The equation of (2) follows from the linear property of $\mathfrak{g}_{i,j,l}$. The equations of (3) and (5) follow from the property 4 in definition 4.1 together with the block source property of $\boldsymbol{x}'_i = (\boldsymbol{x}'_{i,l})_{l\in[m]}$ and $\boldsymbol{k}_i = (\boldsymbol{k}_{i,l})_{l\in[m]}$. The equations of (4) and (6) follow from the parallel repetition description of the underlying extractor Ext.

Now we define the reduction $\mathcal{B}$. On input $(\mathsf{pk}, \mathsf{sk} := \boldsymbol{x})$ and the challenge ciphertext $\widetilde{\mathsf{CT}}$ (either $\mathsf{CT} \leftarrow \mathsf{Encap}(\mathsf{pk})$ or $\mathsf{CT}^* \leftarrow \mathsf{Encap}^*(\mathsf{pk})$) acts as follows:

- $\mathcal{B}$ first forwards pk to $\mathcal{A}$;
- Upon receiving the $\bar{i}$-th KDM query with function $\mathfrak{g}'^{(\bar{i})} \in \mathcal{G}^t$,[13] if $\bar{i} < i$, then $\mathcal{B}$ replies with $\mathsf{Enc}'(\mathsf{pk}, \mathsf{sk}, \mathfrak{g}'^{(\bar{i})})$; if $\bar{i} = i$, $\mathcal{B}$ replies

$$\begin{aligned}
&\Big(\widetilde{\mathsf{CT}}, r_{i,1}, \mathsf{Ext}_{||}(r_{i,1}, \mathsf{Decap}(\boldsymbol{x}, \widetilde{\mathsf{CT}})) + \mathfrak{g}_{i,1}(\boldsymbol{x}) + \boldsymbol{a}_{i,1},\\
&\qquad \ldots, r_{i,t}, \mathsf{Ext}_{||}(r_{i,t}, \mathsf{Decap}(\boldsymbol{x}, \widetilde{\mathsf{CT}})) + \mathfrak{g}_{i,t}(\boldsymbol{x}) + \boldsymbol{a}_{i,t}\Big);
\end{aligned}$$

  otherwise if $\bar{i} > i$, $\mathcal{B}$ replies with $\mathsf{Enc}(\mathsf{pk}, \mathfrak{g}'^{(\bar{i})}(\mathsf{sk}))$.
- Finally, $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs.

Then, we analyze the above reduction process. If $\widetilde{\mathsf{CT}}$ is a valid ciphertext, then $\boldsymbol{k} = \mathsf{Decap}(\boldsymbol{x}, \widetilde{\mathsf{CT}})$ is independent of $\boldsymbol{x}$. This implies the distribution of the replied ciphertext for i-th query is identical to that of $\mathsf{Enc}(\mathsf{pk}, \mathfrak{g}'^{(i)}(\mathsf{sk}))$, which means that $\mathcal{B}$ simulates $\mathsf{H}_{0,i-1}$. If $\widetilde{\mathsf{CT}}$ is an invalid ciphertext, then $\boldsymbol{x}'_i = \mathsf{Decap}(\boldsymbol{x}, \widetilde{\mathsf{CT}})$ and $\boldsymbol{x}'_i = \boldsymbol{k}'_i + \boldsymbol{x}$. Then the distribution of the replied ciphertext for i-th query is identical to that of $\mathsf{Enc}'(\mathsf{pk}, \mathsf{sk}, \mathfrak{g}'^{(i)})$, meaning that $\mathcal{B}$ simulates $\mathsf{H}_{0,i}$.

[13] Here, for each $\bar{i} \in [Q]$, $\mathfrak{g}'^{(\bar{i})}$ is indexed is indexed by a vector $\boldsymbol{a}_{\bar{i}} = (\boldsymbol{a}_{\bar{i},1}^\top, \ldots, \boldsymbol{a}_{\bar{i},t}^\top)^\top \in \mathcal{M}^{m\times t}$ and t functions $\mathfrak{g}_{\bar{i},1}, \ldots, \mathfrak{g}_{\bar{i},t} \in \mathcal{G}$ (ref. Definition 2.16), where for each $j \in [t]$, $\boldsymbol{a}_{\bar{i},j} = (a_{\bar{i},j,1}, \ldots, a_{\bar{i},j,m})^\top \in \mathcal{M}^m$, $\mathfrak{g}_{\bar{i},j} = (\mathfrak{g}_{\bar{i},j,1}, \ldots, \mathfrak{g}_{\bar{i},j,m})$ with $\mathfrak{g}_{\bar{i},j,l} : \mathbb{Z}_B^{n'} \to \mathcal{M}$ and $l \in [m]$.

Thus, $\mathcal{B}$ can break the ciphertext indistinguishability of wHPS with the same advantage as $\mathcal{A}$ who distinguished the two adjacent hybrids. This reaches a contradiction. □

Lemma 5.7 $\left|\mathsf{Pr}[\mathcal{E}_{0.Q}] - \mathsf{Pr}[\mathcal{E}_1]\right| \leq \mathsf{negl}(\lambda)$*, assuming that* (e, poly)*-reusable extractor is homomorphic with respect to the class of linear functions* $\mathcal{G} : \{\mathfrak{g} : \mathbb{Z}_B^{n'} \to \mathcal{M}\}$ *and robust against correlated-source attacks with respect to the class of the shift functions* $\mathcal{SF}_{B,n'} : \{\mathfrak{s} : \mathbb{Z}_B^{n'} \to \mathbb{Z}_B^{n'}\}$.

Proof. Clearly, it suffices to prove that for $m > 1$, the view of the adversary in $\mathsf{H}_{0.Q}$ is computationally indistinguishable from that of H_1. Recall that the view of the adversary in $\mathsf{H}_{0.Q}$ consists of

$$\left(\mathsf{pk}, \mathfrak{g}'^{(1)}, \mathsf{CT}'^{(1)}, \ldots, \mathfrak{g}'^{(Q)}, \mathsf{CT}'^{(Q)}\right), \tag{7}$$

where

$$\begin{aligned}
\mathsf{CT}'^{(i)} &= \Big(\mathsf{CT}_0^{*(i)}, r_{i,1}, \mathsf{Ext}_{||}(r_{i,1}, \mathsf{Decap}(\boldsymbol{x}, \mathsf{CT}_0^{*(i)})) + \mathfrak{g}_{i,1}(\boldsymbol{x}_i) + \boldsymbol{a}_{i,1}, \\
&\qquad \ldots, r_{i,t}, \mathsf{Ext}_{||}(r_{i,t}, \mathsf{Decap}(\boldsymbol{x}, \mathsf{CT}_0^{*(i)})) + \mathfrak{g}_{i,t}(\boldsymbol{x}_i) + \boldsymbol{a}_{i,t}\Big) \\
&= \Big(\mathsf{CT}_0^{*(i)}, r_{i,1}, \{\mathsf{Ext}(\mathfrak{h}_{i,1,l}(r_{i,1}), \boldsymbol{x}'_{i,l}) - \mathfrak{g}_{i,1,l}(\boldsymbol{k}'_{i,l}) + a_{i,1,l}\}_{l\in[m]}, \\
&\qquad \ldots, r_{i,t}, \{\mathsf{Ext}(\mathfrak{h}_{i,t,l}(r_{i,t}), \boldsymbol{x}'_{i,l}) - \mathfrak{g}_{i,t,l}(\boldsymbol{k}'_{i,l}) + a_{i,t,l}\}_{l\in[m]}\Big), \\
&= \Big(\mathsf{CT}_0^{*(i)}, r_{i,1}, \{\mathsf{Ext}(\mathfrak{h}_{i,1,l}(r_{i,1}), \boldsymbol{x}_l + \boldsymbol{k}'_{i,l}) - \mathfrak{g}_{i,1,l}(\boldsymbol{k}'_{i,l}) + a_{i,1,l}\}_{l\in[m]}, \\
&\qquad \ldots, r_{i,t}, \{\mathsf{Ext}(\mathfrak{h}_{i,t,l}(r_{i,t}), \boldsymbol{x}_l + \boldsymbol{k}'_{i,l}) - \mathfrak{g}_{i,t,l}(\boldsymbol{k}'_{i,l}) + a_{i,t,l}\}_{l\in[m]}\Big),
\end{aligned}$$

Next, we analyze the the view of the adversary in H_1, which consists of

$$\left(\mathsf{pk}, \mathfrak{g}'^{(1)}, \mathsf{CT}''^{(1)}, \ldots, \mathfrak{g}'^{(Q)}, \mathsf{CT}''^{(Q)}\right), \tag{8}$$

where

$$\mathsf{CT}''^{(i)} = \Big(\mathsf{CT}_0^{*(i)}, r_{i,1}, \{u_{i,1,l}\}_{l\in[m]}, \ldots, r_{i,t}, \{u_{i,t,l}\}_{l\in[m]}\Big),$$

with $r_{i,j} \xleftarrow{\$} \mathcal{R}$ and $u_{i,j,l} \xleftarrow{\$} \mathcal{M}$. We argue that the two views are indistinguishable by the security of the reusable extractor against correlated-source attacks with respect to shift functions.

More specifically, we define a series of hybrids between $\mathsf{H}_{0,Q}$ and H_1 as follows:

- **Hybrid** $\bar{\mathsf{H}}_0$: This hybrid is the view of adversary in $\mathsf{H}_{0,Q}$, i.e.,

$$\left(\mathsf{pk}, \mathfrak{g}'^{(1)}, \mathsf{CT}'^{(1)}, \ldots, \mathfrak{g}'^{(Q)}, \mathsf{CT}'^{(Q)}\right).$$

- **Hybrid** $\bar{\mathsf{H}}_{0.\bar{l}}$ for each $\bar{l} \in [m]$: For all $\mathsf{CT}'^{(i)}$ with $i \in [Q]$, their final l exactors with respect to certain $r_{i,j}$, i.e., $\{\mathsf{Ext}(\mathfrak{h}_{i,j,l}(r_{i,j}), \boldsymbol{x}'_{i,l})\}_{j\in[t], l\in[(m-\bar{l}+1),m]}$, are replaced with random values $\{u_{i,j,l}\}_{j\in[t], l\in[(m-\bar{l}+1),m]}$ from $\mathcal{M}$.

– **Hybrid** $\bar{\mathsf{H}}_1$: This hybrid is the view of adversary in H_1, i.e.,

$$\left(\mathsf{pk}, \mathfrak{g}'^{(1)}, \mathsf{CT}''^{(1)}, \ldots, \mathfrak{g}'^{(Q)}, \mathsf{CT}''^{(Q)}\right).$$

Then, we can prove the indistinguishability of $\bar{\mathsf{H}}_{0.\bar{l}}$ and $\bar{\mathsf{H}}_{0.(\bar{l}+1)}$, through relying on the correlated source security of reusable extractor in Definition 3.2 and the block-source property of $\boldsymbol{x} = (\boldsymbol{x}_l)_{l\in[m]}$. Notice that the differences between $\bar{\mathsf{H}}_{0.\bar{l}}$ and $\bar{\mathsf{H}}_{0.(\bar{l}+1)}$ are that for $i \in [Q], j \in [t]$, $z_{i,j,m-\bar{l}}$ are whether $\mathsf{Ext}(\mathfrak{h}_{i,j,m-\bar{l}})(r_{i,j}), \boldsymbol{x}_{m-\bar{l}} + \boldsymbol{k}'_{i,m-\bar{l}}) - \mathfrak{g}_{i,j,m-\bar{l}}(\boldsymbol{k}'_{i,m-\bar{l}}) + a_{i,j,m-\bar{l}}$ or $u_{i,j,m-\bar{l}}$.

More specifically, we prove this indistinguishability by reduction. Assume there exists an efficient adversary $\mathcal{A}$ that distinguishes $\bar{\mathsf{H}}_{0.\bar{l}}$ and $\bar{\mathsf{H}}_{0.(\bar{l}+1)}$ with certain non-negligible advantage ε, then we can construct a reduction algorithm $\mathcal{B}$ that breaks the correlated-source security of $(e, t\cdot Q)$-reusable extractor with respect to the class of shift functions, with the same non-negligible advantage ε.

According to Definition 3.2, the target of $\mathcal{B}$ is, given an unknown oracle $\widetilde{O}(\cdot)$, to distinguish whether it is $O_{\boldsymbol{x}^*}(\cdot)$ or a random oracle, with some auxiliary input aux, such that $\mathsf{H}_\infty(\boldsymbol{x}^*|\mathsf{aux}) \geq e$. Here, in order to help the following reduction algorithm $\mathcal{B}$ to simulate the environment of the adversary $\mathcal{A}$, we assume the challenger $\mathcal{C}$ for the correlated-source security of reusable extractor to generate aux in the following way: (1) choose $\mathsf{sk} := (\boldsymbol{x}_l)_{l\in[m-\bar{l}]} \in (\mathbb{Z}_B^{n'})^{m-\bar{l}}$ with $\boldsymbol{x}_l \xleftarrow{\$} \mathbb{Z}_B^{n'}$; (2) invoke the key generation algorithm to generate the public key pk; (3) set $\mathsf{aux} := \mathsf{pk}$. Clearly, from the same argument as that in Section 5.1, we know that $\mathsf{H}_\infty(\boldsymbol{x}^* := \boldsymbol{x}_{m-\bar{l}}|\mathsf{aux}, \{\boldsymbol{x}_l\}_{l\in[m-\bar{l}-1]}) \geq e$, under our choice of parameters.

Now we define the reduction $\mathcal{B}$ in details.

– After receiving $\mathsf{aux} := \mathsf{pk}$ and $\{\boldsymbol{x}_l\}_{l\in[m-\bar{l}-1]}$ from $\mathcal{C}$, $\mathcal{B}$ forwards pk to $\mathcal{A}$;
– Whenever $\mathcal{A}$ makes the i-th KDM query with some function $\mathfrak{g}'^{(i)} \in \mathcal{G}^t$ for each $i \in [Q]$,[14] $\mathcal{B}$ conducts the following steps:
  - Choose a random $\boldsymbol{k}'_i = (\boldsymbol{k}'_{i,l})_{l\in[m]} \in (\mathbb{Z}_B^{n'})^m$ and then generate $\mathsf{CT}_0^{*(i)}$.
  - Make t queries, i.e., $\{\widetilde{O}(\boldsymbol{k}'_{i,m-\bar{l}})\}$, and get replies $\{(r_{i,j}, z_{i,j,m-\bar{l}})\}_{j\in[t]}$.
  - Compute the corresponding invertible functions $\{\mathfrak{h}_{i,j,l} : \mathcal{R} \to \mathcal{R}\}_{j\in[t],l\in[m-\bar{l}]}$ (ref. Definition 3.4).
  - Set
    1. $z'_{i,j,m-\bar{l}} = z_{i,j,m-\bar{l}} - \mathfrak{g}_{i,j,m-\bar{l}}(\boldsymbol{k}'_{i,m-\bar{l}}) + a_{i,j,m-\bar{l}}$ for each $j \in [t]$;
    2. Let $r'_{i,j} = \mathfrak{h}_{i,j}^{-1}(r_{i,j})$ for $j \in [t]$, and set $z'_{i,j,l} = \mathsf{Ext}(\mathfrak{h}_{i,j,l}(r'_{i,j}), \boldsymbol{x}_l + \boldsymbol{k}'_{i,l}) - \mathfrak{g}_{i,j,l}(\boldsymbol{k}'_{i,l}) + a_{i,j,l}$ for all $j \in [t]$ and $l \in [m-\bar{l}-1]$;
  - Reply $\mathcal{A}$ with the ciphertext

$$\widetilde{\mathsf{CT}}^{(i)} = \Big(\mathsf{CT}_0^{*(i)}, r'_{i,1}, \{z'_{i,1,l}\}_{l\in[m-\bar{l}]}, \{u_{i,1,l}\}_{l\in[m-\bar{l}+1,m]}, \ldots, r'_{i,t}, \{z'_{i,t,l}\}_{l\in[m-\bar{l}]}, \{u_{i,t,l}\}_{l\in[m-\bar{l}+1,m]}\Big).$$

[14] From above, we know that the function $\mathfrak{g}'^{(i)} \in \mathcal{G}^t$ is indexed by a vector $\boldsymbol{a}_i = (\boldsymbol{a}_{i,1}^\top, \ldots, \boldsymbol{a}_{i,t}^\top)^\top \in \mathcal{M}^{m\times t}$ and t functions $\mathfrak{g}_{i,1}, \ldots, \mathfrak{g}_{i,t} \in \mathcal{G}$ (ref. Definition 2.16), where for each $i \in [Q], j \in [t]$, $\boldsymbol{a}_{i,j} = (a_{i,j,1}, \ldots, a_{i,j,m})^\top \in \mathcal{M}^m$, $\mathfrak{g}_{i,j} = (\mathfrak{g}_{i,j,1}, \ldots, \mathfrak{g}_{i,j,m})$ with $\mathfrak{g}_{i,j,l} : \mathbb{Z}_B^{n'} \to \mathcal{M}$ and $l \in [m]$.

– Finally, $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs.

Note that in order to answer all KDM queries from $\mathcal{A}$ for Q times, $\mathcal{B}$ need to query the correlate source oracle $\widetilde{O}(\cdot)$ up to $t \cdot Q$ times.

Then, we analyze the above reduction. If $\widetilde{O}(\cdot)$ is the real extractor oracle $O_{\boldsymbol{x}^*}(\cdot)$, then the reply of $z_{i,j,m-\bar{l}}$ would be $\mathsf{Ext}(r_{i,j}, \boldsymbol{x}_{m-\bar{l}} + \boldsymbol{k}'_{i,m-\bar{l}})$ for all $i \in [Q], j \in [t]$. In this case, the view of $\mathcal{A}$ is identical to that of $\bar{\mathsf{H}}_{0.\bar{l}}$. On the other hand, if $\widetilde{O}(\cdot)$ is a random oracle, then the view of the adversary $\mathcal{A}$ follows $\bar{\mathsf{H}}_{0.\bar{l}+1}$. Thus, the reduction $\mathcal{B}$ breaks the correlated-source security of $(e, t \cdot Q)$-reusable extractor with respect to the class of shift functions, with the same advantage as that of $\mathcal{A}$. Clearly, this implies the indistinguishability between $\bar{\mathsf{H}}_{0.\bar{l}}$ and $\bar{\mathsf{H}}_{0.\bar{l}+1}$. Furthermore, a simple hybrid argument implies that for any polynomial parameter m and any PPT adversary, $\bar{\mathsf{H}}_0$ and $\bar{\mathsf{H}}_1$ are computational indistinguishability. Up until now, we complete the proof of this lemma. □

Lemma 5.8 $\left|\mathsf{Pr}[\mathcal{E}_1] - \mathsf{Pr}[\mathcal{E}_2]\right| \leq \mathsf{negl}(\lambda)$*, assuming the ciphertext indistinguishability of the underlying* wHPS*.*

Proof. Just as pointed above, this proof has been captured by the proof of leakage resilience (ref. Section 5.1), so we just omit the proof to avoid repetition. □

Combining Lemma 5.6, 5.7 and 5.8, we can conclude that the advantage $\mathbf{Adv}^{\mathcal{F}\text{-}\mathsf{KDM}}_{\mathsf{PKE},\mathcal{A}}(\lambda)$ of $\mathcal{A}$ in the KDM security game satisfies that:

$$\mathbf{Adv}^{\mathcal{F}\text{-}\mathsf{KDM}}_{\mathsf{PKE},\mathcal{A}}(\lambda) \leq (Q+2) \cdot \mathsf{negl}(\lambda) \leq \mathsf{negl}(\lambda).$$

This completes the proof that Π_{PKE} in Construction 5.3 is $\mathsf{KDM}^{(1)}$-secure with respect to $\mathcal{G}^t$.

6 Achieving $\mathsf{KDM}^{(\bar{n})}$-security from $\mathsf{KDM}^{(1)}$-security

In this section, we show how to upgrade our Construction 5.3 to achieve $\mathsf{KDM}^{(\bar{n})}$-security for an unbounded polynomial $\bar{n}$. In order to do this, we first define a more general design paradigm called BE-*based scheme*, capturing several important features of Construction 5.3. Then we identify two homomorphic properties of BE-based scheme, which only implies the $\mathsf{KDM}^{(\bar{n})}$-security for bounded polynomial $\bar{n}$. Finally, we define an additional pseudorandom property for BE-based scheme, and prove $\mathsf{KDM}^{(\bar{n})}$-security for unbounded polynomial $\bar{n}$ with all these properties.

6.1 BE-Based PKE and its Two Key-homomorphic Properties

Definition 6.1 (BE-based PKE) *Let* BE *be a batch encryption as Definition 2.4. A* BE*-based* PKE *Π is a public-key encryption scheme with the following properties: (1) the secret key of Π is a vector $\boldsymbol{x} \in \mathbb{Z}_B^n$ for some $B, n \in \mathbb{Z}$, as in the scheme* BE*, (2) the public key is* $(\mathsf{CRS}, \mathsf{H}(\mathsf{CRS}, \boldsymbol{x}))$*, where* CRS *is generated by* $\mathsf{BE.Setup}$*, and* $\mathsf{H}(\cdot,\cdot) = \mathsf{BE.KeyGen}(\cdot,\cdot)$ *is the projection function of* BE*. In this way,* CRS *is independent of the secret key.*

Clearly, Construction 5.3, whose wHPS scheme is instantiated by Construction 4.2, is BE-based PKE. Next, we identify two crucial key-homomorphic properties on BE-based PKE schemes, which can be used to achieve the $\mathsf{KDM}^{(\bar{n})}$-security.

Property 1: There is a deterministic algorithm $\mathcal{T}_1$ that takes as input a pair $(\mathsf{CRS}, \mathsf{H}(\mathsf{CRS}, \boldsymbol{x}))$ and a vector $\boldsymbol{k} \in \mathbb{Z}_B^n$, and outputs $(\mathsf{CRS}', \mathsf{H}(\mathsf{CRS}', \boldsymbol{x}+\boldsymbol{k}))$, i.e., $\mathcal{T}_1(\mathsf{CRS}, \mathsf{H}(\mathsf{CRS}, \boldsymbol{x}), \boldsymbol{k}) = (\mathsf{CRS}', \mathsf{H}(\mathsf{CRS}', \boldsymbol{x}+\boldsymbol{k}))$.

Moreover, for any vectors $\boldsymbol{x}, \boldsymbol{k} \in \mathbb{Z}_B^n$ and $\mathsf{CRS} \xleftarrow{\$} \Pi.\mathsf{Setup}(1^\lambda, 1^n)$, the following two distributions are identical (or statistically close):

$$(\mathsf{CRS}, \mathsf{H}(\mathsf{CRS}, \boldsymbol{x}+\boldsymbol{k}), \boldsymbol{x}, \boldsymbol{k}) \equiv (\mathcal{T}_1(\mathsf{CRS}, \mathsf{H}(\mathsf{CRS}, \boldsymbol{x}), \boldsymbol{k}), \boldsymbol{x}, \boldsymbol{k})\,.$$

Property 2: There exists a deterministic algorithm $\mathcal{T}_2$ that takes a pair $(\mathsf{CT}, \boldsymbol{k})$ as input and outputs a ciphertext CT', i.e., $\mathcal{T}_2(\mathsf{CT}, \boldsymbol{k}) = \mathsf{CT}'$. Moreover,

For any message $\mu \in \mathcal{M}$, vectors $\boldsymbol{x}, \boldsymbol{k} \in \mathbb{Z}_B^n$, and CRS, the following distributions are identical (or statistically close):

$$(\mathsf{CT}_1, \mathcal{T}_1(\mathsf{CRS}, \mathsf{H}(\mathsf{CRS}, \boldsymbol{x}), \boldsymbol{k}), \boldsymbol{x}, \boldsymbol{k}) \equiv (\mathcal{T}_2(\mathsf{CT}, \boldsymbol{k}), \mathcal{T}_1(\mathsf{CRS}, \mathsf{H}(\mathsf{CRS}, \boldsymbol{x}), \boldsymbol{k}), \boldsymbol{x}, \boldsymbol{k})\,,$$

where $\mathsf{CT} \leftarrow \Pi.\mathsf{Enc}(\mathsf{CRS}, \mathsf{H}(\mathsf{CRS}, \boldsymbol{x}), \mu)$, and $\mathsf{CT}_1 \leftarrow \Pi.\mathsf{Enc}(\mathsf{CRS}, \mathsf{H}(\mathsf{CRS}, \boldsymbol{x}+\boldsymbol{k}), \mu)$.

Remark 6.2 *These two properties can also be defined for* BE *schemes. Furthermore, if the underlying* BE *scheme has these two properties, Construction 5.3 would inherit these two properties, due to its designs of public key and ciphertext.*

6.2 Intermediate Scheme $\Pi^{\bar{n}}$

Following the above mentioned BE-based PKE scheme $\Pi = \Pi.\{\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}\}$, we define the following intermediate scheme $\Pi^{\bar{n}}$.

Construction 6.3 (Intermediate BE-based PKE $\Pi^{\bar{n}}$) *Given a* BE*-based* PKE $\Pi = \Pi.\{\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}\}$ *with the message space* $\mathcal{M}$*, we construct a new scheme* $\Pi^{\bar{n}} = \Pi^{\bar{n}}.\{\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}\}$ *with the same message space* $\mathcal{M}$ *as follows:*

- $\Pi^{\bar{n}}.\mathsf{KeyGen}(1^\lambda, 1^{\bar{n}})$*: The algorithm does the following steps:*
  1. *Take the security parameter* λ *and an integer* $\bar{n} \in \mathbb{N}$ *as input, run* $\Pi.\mathsf{KeyGen}$ *for* $\bar{n}$ *times to obtain* $\mathsf{CRS}_i \xleftarrow{\$} \Pi.\mathsf{KeyGen}(1^\lambda, 1^{\bar{n}})$ *for* $1 \le i \le \bar{n}$*, where all these* CRS_i *contain the same size parameter* $B \in \mathbb{Z}$*.*
  2. *Choose a random vector* $\boldsymbol{x} \xleftarrow{\$} \mathbb{Z}_B^n$ *to generates* $h_i = \mathsf{H}(\mathsf{CRS}_i, \boldsymbol{x})$ *for* $1 \le i \le \bar{n}$*;*
  3. *Output* $\mathsf{pk} := (\mathsf{pk}_i)_{1 \le i \le \bar{n}}$ *and* $\mathsf{sk} := \boldsymbol{x}$*, where* $\mathsf{pk}_i = (\mathsf{CRS}_i, h_i)$*.*
- $\Pi^{\bar{n}}.\mathsf{Enc}(\mathsf{pk}, \mu)$*: Given a public-key* pk *and a message* $\mu \in \mathcal{M}$ *as input, the algorithm runs* $\Pi.\mathsf{Enc}$ *for* $\bar{n}$ *times to generate* $\mathsf{CT}_i \xleftarrow{\$} \Pi.\mathsf{Enc}(\mathsf{pk}_i, \mu)$ *for* $1 \le i \le \bar{n}$*, and then outputs* $\mathsf{CT} = (\mathsf{CT}_1, \ldots, \mathsf{CT}_{\bar{n}})$ *as the ciphertext of* $\mu \in \mathcal{M}$*.*

– $\Pi^{\bar{n}}.\mathsf{Dec}(\mathsf{sk}, \mathsf{CT})$*: Given a ciphertext* $\mathsf{CT} = (\mathsf{CT}_1, \ldots, \mathsf{CT}_{\bar{n}})$ *and a secret key* sk *as input, the algorithm runs* $\Pi.\mathsf{Dec}$ *to generate* $\mu' = \Pi.\mathsf{Dec}(\mathsf{sk}, \mathsf{CT}_i)$ *for some* $i \in [\bar{n}]$*, and then output* μ' *as a plaintext for* CT*.*

We note that the correctness of the scheme $\Pi^{\bar{n}}$ follows clearly from that of Π. Next we present a KDM-security reduction between Π and $\Pi^{\bar{n}}$.

Theorem 6.4 ($\mathsf{KDM}^{(\bar{n})}$-security of Π) *Suppose that (1) a* BE*-based* PKE *scheme* Π *satisfies Properties 1 and 2 in Section 6.1, and (2) the intermediate scheme* $\Pi^{\bar{n}}$ *in Definition 6.3 is* $\mathsf{KDM}^{(1)}$*-secure with respect to the class* $\mathcal{G} = \{\mathfrak{g} : \mathcal{SK} \to \mathcal{M}\}$ *of all affine (resp., block-affine) functions from* $\mathcal{SK}$ *to* $\mathcal{M}$*. Then* Π *is* $\mathsf{KDM}^{(\bar{n})}$*-secure with respect to the class* $\mathcal{F} = \{\mathfrak{f} : \mathcal{SK}^{\bar{n}} \to \mathcal{M}\}$ *of all affine (resp., block-affine) functions from* $\mathcal{SK}^{\bar{n}}$ *to* $\mathcal{M}$*.*

Proof. We prove this theorem by using security reduction. In particular, suppose there is an efficient adversary $\mathcal{A}$ breaking the $\mathsf{KDM}^{(\bar{n})}$-security of Π, then we can construct an efficient reduction algorithm $\mathcal{R}$ that can break the $\mathsf{KDM}^{(1)}$-security of $\Pi^{\bar{n}}$ with the same advantage as $\mathcal{A}$. Here, we denote $\mathcal{C}$ as the challenger in the $\mathsf{KDM}^{(1)}$-security experiment of $\Pi^{\bar{n}}$. We summarize the reduction idea in Figure 1, and describe the detailed process below.

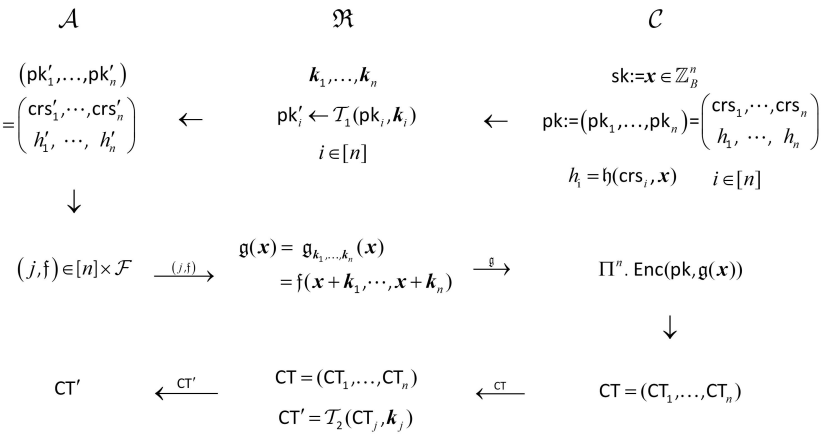

Fig. 1. Our Reduction Idea.

– Setup.
1. $\mathcal{C}$ first runs the algorithm $\Pi^{\bar{n}}.\mathsf{KeyGen}$ to generate $\mathsf{pk} := (\mathsf{pk}_i)_{1 \le i \le \bar{n}}$ and $\mathsf{sk} := \boldsymbol{x}$, where $\mathsf{pk}_i = (\mathsf{CRS}_i, h_i)$. Then, $\mathcal{C}$ chooses a random bit $b \in \{0, 1\}$ and sends pk to $\mathcal{R}$.
2. $\mathcal{R}$ selects vectors $\boldsymbol{k}_1, \ldots, \boldsymbol{k}_{\bar{n}} \xleftarrow{\$} \mathbb{Z}_B^n$, and runs algorithm $\mathcal{T}_1(\mathsf{pk}_i, \boldsymbol{k}_i)$ to generate $\mathsf{pk}'_i = (\mathsf{CRS}'_i, h'_i)$ for $1 \le i \le \bar{n}$. Then, $\mathcal{R}$ sends all these pk'_i to $\mathcal{A}$ as the challenge public keys of Π.

– KDM Queries.
  1. With $\{\mathsf{pk}'_i\}_{1\le i\le \bar{n}}$ from $\mathcal{R}$, $\mathcal{A}$ can adaptively conduct KDM queries in the form of $(i, \mathfrak{f})$ to $\mathcal{R}$, where $1 \le i \le \bar{n}$ and $\mathfrak{f} \in \mathcal{F}$.
  2. With the KDM query $(i, \mathfrak{f})$ from $\mathcal{A}$, $\mathcal{R}$ first transforms the function $\mathfrak{f} : \mathcal{SK}^{\bar{n}} \to \mathcal{M}$ into the function $\mathfrak{g}_{\{\boldsymbol{k}_i\}_{i\in[\bar{n}]}} : \mathcal{SK} \to \mathcal{M}$ with the help of the previously chosen $\boldsymbol{k}_1, \ldots, \boldsymbol{k}_{\bar{n}}$. Then, $\mathcal{R}$ sends this function $\mathfrak{g}_{\{\boldsymbol{k}_i\}_{i\in[\bar{n}]}} \in \mathcal{G}$ to $\mathcal{C}$ as a KDM query.

– KDM Responses.
  1. After receiving $\mathfrak{g}_{\{\boldsymbol{k}_i\}_{i\in[\bar{n}]}} \in \mathcal{G}$ from $\mathcal{R}$, $\mathcal{C}$ computes

$$\mathsf{CT}_0 = \Pi^{\bar{n}}.\mathsf{Enc}(\mathsf{pk}, \mathfrak{g}_{\{\boldsymbol{k}_i\}_{i\in[\bar{n}]}}(\boldsymbol{x})) = (\mathsf{CT}_{0,1}, \ldots, \mathsf{CT}_{0,\bar{n}})$$

  if $b = 0$, otherwise computes

$$\mathsf{CT}_1 = \Pi^{\bar{n}}.\mathsf{Enc}(\mathsf{pk}, 0) = (\mathsf{CT}_{1,1}, \ldots, \mathsf{CT}_{1,\bar{n}}).$$

  Then, $\mathcal{C}$ responds CT_b to $\mathcal{R}$.
  2. After receiving CT_b from $\mathcal{C}$, in order to respond the KDM query $(i, \mathfrak{f})$ from $\mathcal{A}$, $\mathcal{R}$ runs the algorithm $\mathcal{T}_2(\mathsf{CT}_{b,i}, \boldsymbol{k}_i)$, and returns it to $\mathcal{A}$.

– Output Stage.
  1. $\mathcal{A}$ outputs a bit $b' \in \{0, 1\}$ and sends it to $\mathcal{R}$.
  2. $\mathcal{R}$ outputs the same b' as the guess for b.

Notice that, the KDM query function $\mathfrak{f}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{\bar{n}})$ from $\mathcal{A}$ is essential $\mathfrak{f}(\boldsymbol{x} + \boldsymbol{k}_1, \ldots, \boldsymbol{x}+\boldsymbol{k}_{\bar{n}})$, since $\boldsymbol{x}_i = \boldsymbol{x}+\boldsymbol{k}_i$ for $1 \le i \le \bar{n}$ according to the assumed Property 1 in Section 6.1. Clearly, with the previously known knowledge of $(\boldsymbol{k}_1, \ldots, \boldsymbol{k}_{\bar{n}})$, it is easy to rewrite

$$\mathfrak{f}(\boldsymbol{x} + \boldsymbol{k}_1, \ldots, \boldsymbol{x} + \boldsymbol{k}_{\bar{n}}) = \mathfrak{g}_{\{\boldsymbol{k}_i\}_{i\in[\bar{n}]}}(\boldsymbol{x}).$$

At the same time, according to Property 2, the transformations among ciphertexts under different CRS are perfect. From all above analyses, we can conclude that the above simulation process is perfect, no matter $b = 0$ or 1. Therefore, it makes sense for $\mathcal{R}$ to output the same value as $\mathcal{A}$. And the success advantage of $\mathcal{R}$ is the same as $\mathcal{A}$.

Finally, this complete the proof of this theorem. □

Remark 6.5 *Our construction can support more general relationship between $\mathcal{F}$ and $\mathcal{G}$. Particularly, the theorem also holds for the following relation. For every $\boldsymbol{k}_1, \ldots, \boldsymbol{k}_{\bar{n}}$ and $\mathfrak{h} \in \mathcal{F}$, we have $\mathfrak{g}_{\boldsymbol{k}_1,\ldots,\boldsymbol{k}_{\bar{n}}}(\boldsymbol{x}) := \mathfrak{h}(\boldsymbol{x} + \boldsymbol{k}_1, \ldots, \boldsymbol{x} + \boldsymbol{k}_{\bar{n}}) \in \mathcal{G}$.*

6.3 Proving $\mathsf{KDM}^{(1)}$-Security of $\Pi^{\bar{n}}$

In this section, we first define the required new pseudorandom property, and then show how it derives $\mathsf{KDM}^{(1)}$-security of $\Pi^{\bar{n}}_{\mathsf{PKE}}$ for unbounded polynomial $\bar{n}$. In the next section, we show how to construct such an underlying BE.

Definition 6.6 *Let* $\mathsf{Ext} : \mathcal{R} \times \mathbb{Z}_B^n \to \mathcal{M}$ *be some (reusable) extractor. A* BE*-based* PKE *satisfies an additional pseudorandom property if the following holds. For any polynomial* $\bar{n} = \mathsf{poly}(\lambda)$*, the following two distributions are computationally indistinguishable:*

$$\left(\begin{pmatrix} \mathsf{CRS}_1, \cdots, \mathsf{CRS}_{\bar{n}} \\ h_1, \ \cdots, \ h_{\bar{n}} \end{pmatrix}, \{r_i, \mathsf{Ext}(r_i, \boldsymbol{x} + \boldsymbol{k}_i)\}_{i\in[t]}\right)$$
$$\approx_c \left(\begin{pmatrix} \mathsf{CRS}_1, \cdots, \mathsf{CRS}_{\bar{n}} \\ u_1, \ \cdots, \ u_{\bar{n}} \end{pmatrix}, \{r_i, u_i'\}_{i\in[t]}\right)$$

where $\{\mathsf{CRS}_i\}_{i\in[\bar{n}]}$*,* $\{u_i\}_{i\in[\bar{n}]}$ *and* $\{u_i'\}_{i\in[\bar{t}]}$ *are uniformly random,* $\boldsymbol{x} \xleftarrow{\$} \mathbb{Z}_B^n$*, and* $h_i = \mathsf{H}(\mathsf{CRS}_i, \boldsymbol{x})$ *for all* $i \in [\bar{n}]$*.*

Theorem 6.7 *Let* Π_{PKE} *be the* BE*-based scheme as Construction 5.3. Suppose the underlying* BE *satisfies the pseudorandom property as Definition 6.6. Then for any polynomial* $\bar{n}$*, the intermediate scheme* $\Pi_{\mathsf{PKE}}^{\bar{n}}$ *is* $\mathsf{KDM}^{(1)}$*-secure with respect to all block-affine functions.*

The proof of this theorem is similar to that of Theorem 5.4. Particularly, we would switch all the real KDM responses to Enc' as in the **Hybrid** $\mathsf{H}_{0,Q}$, and then use the reusable extractor (against shift functions) to further switch the responses to Enc'' as in the **Hybrid** H_1. The key observation is the following: $\mathsf{H}(\mathsf{CRS}, \boldsymbol{x})$ does not leak $\boldsymbol{x}$ in the computational sense and can be used in connection with the extractor. Thus, the same argument of Theorem 5.4 goes through in this case.

Proof. This proof takes the same high-level steps and the algorithms $\mathsf{Enc}'_{\boldsymbol{k}}$ and $\mathsf{Enc}''_{\boldsymbol{k}}$ as that of Section 5.2. Thus, we omit them here.

Furthermore, in order to deal with $h_i = \mathfrak{h}(\mathsf{CRS}_i, \boldsymbol{x})$ for $i \in [\bar{n}]$ in the public key, we adopt the pseudorandom property of the public key in Definition 6.6, and redefine the series of hybrids as follows:

- **Hybrid** H_0: This hybrid is identical to the original KDM queries case, i.e. the responses of all the Q KDM queries are generated as the real encryptions of the $\mathfrak{g}'^{(i)}(\mathsf{sk})$ for $i = 1, ..., Q$.
- **Hybrid** $\mathsf{H}_{0.i}$ for each $i \in [Q]$: Upon receiving the first i KDM queries, this hybrid uses Enc' to reply and then generates the remaining KDM responses according to the original encryption algorithm as H_0.
- **Hybrid** H_1: This hybrid first chooses and uses the real random $\{u_i\}_{i\in[\bar{n}]}$ in public key, instead of the real ones $\{h_i = \mathsf{H}(\mathsf{CRS}_i, \boldsymbol{x})\}_{i\in[\bar{n}]}$, and then does the following. Upon receiving all KDM queries, this hybrid uses Enc'' to reply through using the random public key.
- **Hybrid** H_2: This hybrid replies all KDM queries with $\mathsf{Enc}(0)$ through using the random public key.
- **Hybrid** H_3: This hybrid replies all KDM queries with $\mathsf{Enc}(0)$ through using the real public key.

Let events $\mathcal{E}_0$, $\mathcal{E}_1$, $\mathcal{E}_2$ and $\mathcal{E}_3$ denote that the KDM adversary $\mathcal{A}$ outputs 1 in H_0, H_1, H_2 and H_3 respectively. Similarly, we define events $\mathcal{E}_{0.i}$. To show that $\mathsf{Pr}[\mathcal{E}_0] \approx \mathsf{Pr}[\mathcal{E}_3]$, we will take the following path:

$$\mathsf{Pr}[\mathcal{E}_0] \approx \mathsf{Pr}[\mathcal{E}_{0.1}] \approx \cdots \approx \mathsf{Pr}[\mathcal{E}_{0.Q}] \approx \mathsf{Pr}[\mathcal{E}_1] \approx \mathsf{Pr}[\mathcal{E}_2] \approx \mathsf{Pr}[\mathcal{E}_3].$$

For notational convenience, we define $\mathsf{H}_{0.0} := \mathsf{H}_0$.

Lemma 6.8 *For $i \in [Q]$, $\left|\mathsf{Pr}[\mathcal{E}_{0.i-1}] - \mathsf{Pr}[\mathcal{E}_{0.i}]\right| \leq \mathsf{negl}(\lambda)$, assuming the ciphertext indistinguishability of the underlying* wHPS.

Proof. This proof is the same as that of Lemma 5.6. Hence, we omit the details here. □

Lemma 6.9 $\left|\mathsf{Pr}[\mathcal{E}_{0.Q}] - \mathsf{Pr}[\mathcal{E}_1]\right| \leq \mathsf{negl}(\lambda)$, *assuming the pseudorandomness in Definition 6.6.*

Proof. Similar to the proof in Lemma 5.7, the view of the adversary in $\mathsf{H}_{0.Q}$ consists of

$$\left(\mathsf{pk}, \mathfrak{g}'^{(1)}, \mathsf{CT}'^{(1)}, \ldots, \mathfrak{g}'^{(Q)}, \mathsf{CT}'^{(Q)}\right), \tag{9}$$

where

$$\mathsf{pk} := \begin{pmatrix} \mathsf{CRS}_1, \cdots, \mathsf{CRS}_{\bar{n}} \\ h_1, \ \cdots, \ h_{\bar{n}} \end{pmatrix}$$

and

$$\begin{aligned}
\mathsf{CT}'^{(i)} &= \Big(\mathsf{CT}_0^{*(i)}, r_{i,1}, \mathsf{Ext}_{||}(r_{i,1}, \mathsf{Decap}(\boldsymbol{x}, \mathsf{CT}_0^{*(i)})) + \mathfrak{g}_{i,1}(\boldsymbol{x}_i) + \boldsymbol{a}_{i,1}, \\
&\qquad \ldots, r_{i,t}, \mathsf{Ext}_{||}(r_{i,t}, \mathsf{Decap}(\boldsymbol{x}, \mathsf{CT}_0^{*(i)})) + \mathfrak{g}_{i,t}(\boldsymbol{x}_i) + \boldsymbol{a}_{i,t}\Big) \\
&= \Big(\mathsf{CT}_0^{*(i)}, r_{i,1}, \{\mathsf{Ext}(\mathfrak{h}_{i,1,l}(r_{i,1}), \boldsymbol{x}'_{i,l}) - \mathfrak{g}_{i,1,l}(\boldsymbol{k}'_{i,l}) + a_{i,1,l}\}_{l\in[m]}, \\
&\qquad \ldots, r_{i,t}, \{\mathsf{Ext}(\mathfrak{h}_{i,t,l}(r_{i,t}), \boldsymbol{x}'_{i,l}) - \mathfrak{g}_{i,t,l}(\boldsymbol{k}'_{i,l}) + a_{i,t,l}\}_{l\in[m]}\Big), \\
&= \Big(\mathsf{CT}_0^{*(i)}, r_{i,1}, \{\mathsf{Ext}(\mathfrak{h}_{i,1,l}(r_{i,1}), \boldsymbol{x}_l + \boldsymbol{k}'_{i,l}) - \mathfrak{g}_{i,1,l}(\boldsymbol{k}'_{i,l}) + a_{i,1,l}\}_{l\in[m]}, \\
&\qquad \ldots, r_{i,t}, \{\mathsf{Ext}(\mathfrak{h}_{i,t,l}(r_{i,t}), \boldsymbol{x}_l + \boldsymbol{k}'_{i,l}) - \mathfrak{g}_{i,t,l}(\boldsymbol{k}'_{i,l}) + a_{i,t,l}\}_{l\in[m]}\Big),
\end{aligned}$$

for any $i \in [Q]$.

Similarly, the view of the adversary in H_1 consists of

$$\left(\mathsf{pk}', \mathfrak{g}'^{(1)}, \mathsf{CT}''^{(1)}, \ldots, \mathfrak{g}'^{(Q)}, \mathsf{CT}''^{(Q)}\right), \tag{10}$$

where

$$\mathsf{pk}' = \begin{pmatrix} \mathsf{CRS}_1, \cdots, \mathsf{CRS}_{\bar{n}} \\ u_1, \ \cdots, \ u_{\bar{n}} \end{pmatrix}$$

and

$$\mathsf{CT}''^{(i)} = \Big(\mathsf{CT}_0^{*(i)}, r_{i,1}, \{u_{i,1,l}\}_{l\in[m]}, \ldots, r_{i,t}, \{u_{i,t,l}\}_{l\in[m]}\Big),$$

with $r_{i,j} \xleftarrow{\$} \mathcal{R}$ and $u_{i,j,l} \xleftarrow{\$} \mathcal{M}$.

Next, it is sufficient for us to prove (9) $\approx_c$ (10) through using the reduction approach from the pseudorandom property in Definition 6.6. This is quite similar to the reduction in Lemma 5.7, which is omitted here. □

Lemma 6.10 $\left|\mathsf{Pr}[\mathcal{E}_1]-\mathsf{Pr}[\mathcal{E}_2]\right| \leq \mathsf{negl}(\lambda)$*, assuming the ciphertext indistinguishability of the underlying* wHPS*.*

Proof. We omit the detailed proof, since it is quite similar to that of Lemma 5.8. □

Lemma 6.11 $\left|\mathsf{Pr}[\mathcal{E}_2] - \mathsf{Pr}[\mathcal{E}_3]\right| \leq \mathsf{negl}(\lambda)$*, assuming the pseudorandomness of the public key in Definition 6.6.*

Proof. We omit the detailed proof, since it is quite similar to that of Lemma 6.9. □

Combining Lemma 6.8, 6.9, 6.10 and 6.11, we can conclude that the advantage $\mathbf{Adv}^{\mathcal{F}\text{-}\mathsf{KDM}}_{\mathsf{PKE},\mathcal{A}}(\lambda)$ of $\mathcal{A}$ in the KDM security game satisfies that:

$$\mathbf{Adv}^{\mathcal{F}\text{-}\mathsf{KDM}}_{\mathsf{PKE},\mathcal{A}}(\lambda) \leq (Q+3) \cdot \mathsf{negl}(\lambda) \leq \mathsf{negl}(\lambda).$$

This completes the proof that $\Pi^{\bar{n}}_{\mathsf{PKE}}$ in Construction 6.3 is $\mathsf{KDM}^{(1)}$-secure with respect to $\mathcal{G}^t$. □

For the clarity of our presentation, we defer the detailed constructions of the required BE to Section 7.

Summing up Theorems 6.4, 6.7 and the instantiations of the required BE in Section 7, we conclude that for any polynomial $\bar{n}$, Construction 5.3 is $\mathsf{KDM}^{(\bar{n})}$-secure with respect to block-affine functions.

7 Instantiations of BE

In this section, we show how to instantiate the required BE in Section 6.3 as below.

- In Section 7.1, we verify that the existing CDH-based BE scheme in [16] satisfy the two key-homomorphic properties, which can be used to derive a $\mathsf{KDM}^{(\bar{n})}$-secure PKE for bounded polynomial $\bar{n}$. Furthermore, we observe that this CDH-based BE scheme also satisfies the pseudorandom condition if we further assume the DDH assumption. In this case, we can further prove the corresponding construction is $\mathsf{KDM}^{(\bar{n})}$-secure for unbounded polynomial $\bar{n}$ from DDH.
- In Section 7.2, we construct a new LWE-based BE scheme that satisfies the two key-homomorphic properties and the pseudorandom condition simultaneously, which can be used to derive a $\mathsf{KDM}^{(\bar{n})}$-secure PKE for unbounded polynomial $\bar{n}$.

7.1 CDH/DDH-based BE Scheme in [16]

In this section, we first recall the existing CDH-based BE Scheme in [16], and then verity that it satisfies two key-homomorphic properties. Moreover, this BE scheme can also be verified to satisfy the additional pseudorandom condition, if the DDH assumption holds.

Definition 7.1 *Define* $\mathsf{gl\text{-}enc}(x, \mu)$ *as a randomized function that on input* $x \in \{0,1\}^\ell, \mu \in \{0,1\}$ *samples* $\alpha \in \{0,1\}^\ell$ *and outputs* $(\alpha, \langle \alpha, x\rangle \oplus \mu)$*, where the inner product is over the binary field. Define* $\mathsf{gl\text{-}dec}(x, (\alpha, \sigma))$ *be the function that takes* $x \in \{0,1\}^\ell$ *and* $(\alpha, \sigma) \in \{0,1\}^{\ell+1}$ *and outputs* $\sigma \oplus \langle \alpha, x\rangle$*.*

Construction 7.2 (CDH-based BE in [16]) *Let* $\mathcal{G}$ *be a group sampler as the definition of* CDH *assumption, and Goldreich-Levin encoding/decoding procedure* gl-enc/gl-dec *defined as Definition 7.1. The* CDH*-based batch encryption scheme is as follows:*

- $\mathsf{CDH\text{-}BE.Setup}(1^\lambda, 1^n)$*. Taking* λ *and* n *as input, the algorithm first samples* $(\mathbb{G}, g, q) \leftarrow \mathcal{G}(1^\lambda)$ *such that* $n \geq \log q + 2\log(1/\varepsilon)$ *for some negligible function* ε *of* λ*, and then samples* $\alpha_{i,b} \leftarrow \mathbb{Z}_q$ *for* $i \in [n], b \in \{0,1\}$*. Furthermore, the algorithm defines* $g_{i,b} = g^{\alpha_{i,b}}$*, and outputs* $\mathsf{CRS} = ((\mathbb{G}, g, q), \{g_{i,b}\}_{i\in[n], b\in\{0,1\}})$*.*
- $\mathsf{CDH\text{-}BE.KeyGen}(\mathsf{CRS}, \boldsymbol{x})$*. The algorithm computes and outputs* $h = \prod_i g_{i,x_i}$*.*
- $\mathsf{CDH\text{-}BE.Enc}(\mathsf{CRS}, h, \mathbf{M})$*. Taking as input a common reference string* CRS*, the public key* h*, and a matrix* $\mathbf{M} \in \{0,1\}^{n\times 2}$*, the algorithm does the following steps. For each* $i \in [n]$*, sample* $r_i \leftarrow \mathbb{Z}_q$*. Then for all* $j \neq i$ *and* $b \in \{0,1\}$ *compute:* $\hat{g}_{j,b} = g_{j,b}^{r_i}$*. Compute* $\hat{g}_{i,b} = h^{r_i} g_{i,b}^{-r_i}$*, and let* $\mu_{i,b} = \mathsf{gl\text{-}enc}(\hat{g}_{i,b}, M_{i,b})$*. Output* $\mathsf{CT} = \{\mathsf{ct}_i\}_{i\in[n]} = \{(\{\hat{g}_{j,b}\}_{j\neq i, b\in\{0,1\}}, \{\mu_{i,b}\}_{b\in\{0,1\}})\}_{i\in[n]}$*.*
- $\mathsf{CDH\text{-}BE.Dec}(\mathsf{CRS}, \boldsymbol{x}, \mathsf{CT})$*. Taking as input* CRS*,* $\boldsymbol{x}$ *and* CT*, the algorithm first parses* $\mathsf{CT} = (\mathsf{ct}_1, ..., \mathsf{ct}_n)$*. Then for* $\mathsf{ct}_i = (\{\hat{g}_{j,b}\}_{j\neq i, b\in\{0,1\}}, \{\mu_{i,b}\}_{b\in\{0,1\}})$ *with* $i \in [n]$*, the algorithm computes* $\hat{g}_{i,x_i} = \prod_{j\neq i} \hat{g}_{j,x_j}$ *and* $m_{i,x_i} = \mathsf{gl\text{-}dec}(\hat{g}_{i,x_i}, \mu_{i,x_i})$*. Finally, the algorithm outputs* $\boldsymbol{m} = (m_{1,x_1}, ..., m_{n,x_n})$*.*

Two key-homomorphic properties of CDH-BE. We show that the CDH-based BE satisfies the two properties in Section 6.1. We first define algorithm $\mathcal{T}_1$ as follows:

$\mathcal{T}_1(\mathsf{CRS}, h, \boldsymbol{k}')$: Take $\mathsf{CRS} = ((\mathbb{G}, g, q), \{g_{i,b}\}_{i,b})$, $h = \prod_i g_{i,x_i}$ for a random key vector $\boldsymbol{x} \in \{0,1\}^n$, and a random vector $\boldsymbol{k}' \in \{0,1\}^n$ as input. Do the followings:

1. For $\boldsymbol{k}' = (k'_1, ..., k'_1) \in \{0,1\}^n$, define $g'_{i,b} = g_{i,b+k'_i}$, where $b + k'_i$ is the bit addition operation.
2. Set $\mathsf{CRS}' = ((\mathbb{G}, g, q), \{g'_{i,b}\}_{i,b})$, and set $h' = h$.
3. Output (CRS', h').

It's easy to see that

$$h = \prod_{i=1}^{n} g_{i,x_i} = \prod_{i=1}^{n} g'_{i,x_i+k'_i}.$$

So h' can also be viewed as a projection of the vector $(\boldsymbol{x}+\boldsymbol{k}')$ under CRS', i.e., $h' = \mathfrak{h}(\mathsf{CRS}', \boldsymbol{x}+\boldsymbol{k}')$. Then we can show that the Property 1 in Section 6.1 holds, i.e., the following equation holds

$$\left(\mathsf{CRS}, \mathfrak{h}(\mathsf{CRS}, \boldsymbol{x}+\boldsymbol{k}'), \boldsymbol{x}, \boldsymbol{k}'\right) \equiv \left(\mathcal{T}_1(\mathsf{CRS}, \mathfrak{h}(\mathsf{CRS}, \boldsymbol{x}), \boldsymbol{k}'), \boldsymbol{x}, \boldsymbol{k}'\right).$$

It's sufficient to show

$$\left(\mathsf{CRS}, \mathfrak{h}(\mathsf{CRS}, \boldsymbol{x}+\boldsymbol{k}'), \boldsymbol{x}, \boldsymbol{k}'\right) \equiv \left(\mathsf{CRS}', \mathfrak{h}(\mathsf{CRS}', \boldsymbol{x}+\boldsymbol{k}'), \boldsymbol{x}, \boldsymbol{k}'\right).$$

According to the relationship between CRS and CRS', it is clear that

$$(\mathsf{CRS}, \boldsymbol{k}') \equiv (\mathsf{CRS}', \boldsymbol{k}').$$

As a result, the Property 1 holds.

Next, we define the algorithm $\mathcal{T}_2$ in the following way:

$\mathcal{T}_2(\mathsf{CT}, \boldsymbol{k}')$: Take a random ciphertext $\mathsf{CT} = \{(\{\hat{g}_{j,b}\}_{j\neq i, b\in\{0,1\}}, \{\mu_{i,b}\}_{b\in\{0,1\}})\}_{i\in[n]}$ and $\boldsymbol{k}' \in \{0,1\}^n$ as input, where $\mathsf{CT} = \mathsf{CDH\text{-}BE.Enc}(\mathsf{CRS}, h, \mathbf{M})$ for $\mathsf{CRS} = (\mathbb{G}, g, q), \{g_{i,b}\}_{i,b}), h = \mathfrak{h}(\mathsf{CRS}, \boldsymbol{x}) = \prod_i g_{i,x_i}$ and plaintext $\mathbf{M} = (m_{i,j})_{i\in[n], j\in\{0,1\}}$. Do the followings:

1. For the vector $\boldsymbol{k}' = (k'_1, \ldots, k'_n) \in \{0,1\}^n$, define $\hat{g}'_{j,b} = \hat{g}_{j,b+k'_i}$ for each $i \in [n]$, where $j \neq i, b \in \{0,1\}$.
2. Set $\mu'_{i,b} = \mu_{i,b+k'_i}$ for $i \in [n], b \in \{0,1\}$.
3. Output $\mathsf{CT}' = \{(\{\hat{g}'_{j,b}\}_{j\neq i, b\in\{0,1\}}, \{\mu'_{i,b}\}_{b\in\{0,1\}})\}_{i\in[n]}$.

We want to show the property 2 in Section 6.1 holds, i.e., the following equation holds

$$(\mathsf{CT}_1, \mathcal{T}_1(\mathsf{CRS}, \mathfrak{h}(\mathsf{CRS}, \boldsymbol{x}), \boldsymbol{k}), \boldsymbol{x}, \boldsymbol{k}) \equiv (\mathcal{T}_2(\mathsf{CT}, \boldsymbol{k}), \mathcal{T}_1(\mathsf{CRS}, \mathfrak{h}(\mathsf{CRS}, \boldsymbol{x}), \boldsymbol{k}), \boldsymbol{x}, \boldsymbol{k}),$$

where $\mathsf{CT} \leftarrow \mathit{\Pi}.\mathsf{Enc}(\mathsf{CRS}, \mathfrak{h}(\mathsf{CRS}, \boldsymbol{x}), \mu)$, and $\mathsf{CT}_1 \leftarrow \mathit{\Pi}.\mathsf{Enc}(\mathsf{CRS}, \mathfrak{h}(\mathsf{CRS}, \boldsymbol{x}+\boldsymbol{k}), \mu)$. By our setting, it's easy to see $\mathcal{T}_2(\mathsf{CT}, \boldsymbol{k}) = \mathsf{CT}' = \mathit{\Pi}.\mathsf{Enc}(\mathsf{CRS}', \mathfrak{h}(\mathsf{CRS}', \boldsymbol{x}+\boldsymbol{k}'), \mu)$. It's sufficient to show

$$\begin{aligned}&(\mathit{\Pi}.\mathsf{Enc}(\mathsf{CRS}, \mathfrak{h}(\mathsf{CRS}, \boldsymbol{x}+\boldsymbol{k}), \mu), \mathcal{T}_1(\mathsf{CRS}, \mathfrak{h}(\mathsf{CRS}, \boldsymbol{x}), \boldsymbol{k}), \boldsymbol{x}, \boldsymbol{k})\\ \equiv &\left(\mathit{\Pi}.\mathsf{Enc}(\mathsf{CRS}', \mathfrak{h}(\mathsf{CRS}', \boldsymbol{x}+\boldsymbol{k}'), \mu), \mathcal{T}_1(\mathsf{CRS}, \mathfrak{h}(\mathsf{CRS}, \boldsymbol{x}), \boldsymbol{k}), \boldsymbol{x}, \boldsymbol{k}\right),\end{aligned}$$

which is clear by our setting. As a result, the Property 2 holds.

Pseudorandom properties of DDH-BE. We show that the BE scheme constructed above satisfies the pseudorandom property as definition 6.6 under DDH assumption. Firstly, we need to show that for any polynomial $t = \mathsf{poly}(\lambda)$, the following two distributions are computationally indistinguishable:

$$\begin{pmatrix}\mathsf{CRS}_1, \cdots, \mathsf{CRS}_t\\ h_1, \cdots, h_t\end{pmatrix} \approx_c \begin{pmatrix}\mathsf{CRS}_1, \cdots, \mathsf{CRS}_t\\ u_1, \cdots, u_t\end{pmatrix},$$

where $\{\mathsf{CRS}_s\}_{s\in[t]}$ and $\{u_s\}_{s\in[t]}$ are uniformly random, $\boldsymbol{x} \xleftarrow{\$} \mathbb{Z}_2^n$, and $h_s = \mathfrak{h}(\mathsf{CRS}_s, \boldsymbol{x})$ for all $s \in [t]$. By our construction, $\mathsf{CRS}_s = ((\mathbb{G}, g, q), \{g_{i,b}^{(s)}\}_{i,b})$, $h_s = \prod_{i=1}^n g_{i,x_i}^{(s)}$, for each $s \in [t]$. We can rewrite $\{g_{i,b}^{(s)}\}_{i,b}$ as $g^{\boldsymbol{\alpha}^{(s)}}$ where $\boldsymbol{\alpha}^{(s)} = (\alpha_{1,0}^{(s)}, \alpha_{2,0}^{(s)}, \ldots, \alpha_{n,0}^{(s)}, \alpha_{1,1}^{(s)}, \alpha_{2,1}^{(s)}, \ldots, \alpha_{n,1}^{(s)}) \in \mathbb{Z}_q^{2n}$, then $h_s = g^{\langle \boldsymbol{\alpha}^{(s)}, \tilde{\boldsymbol{x}}\rangle}$, where $\tilde{\boldsymbol{x}}^\top = (\bar{\boldsymbol{x}}^\top, \boldsymbol{x}^\top) = (1-x_1, \ldots, 1-x_n, x_1, \ldots, x_n) \in \{0,1\}^{2n}$. Therefore, we can rephrase $\begin{pmatrix} \mathsf{CRS}_1, \cdots, \mathsf{CRS}_t \\ h_1, \ \cdots, \ h_t \end{pmatrix}$ as $((\mathbb{G}, g, q), g^{\mathbf{A}^\top}, g^{\mathbf{A}^\top \tilde{\boldsymbol{x}}})$, where $\mathbf{A} = (\boldsymbol{\alpha}^{(1)}, \cdots, \boldsymbol{\alpha}^{(t)})$, $(\mathsf{CRS}_1, \cdots, \mathsf{CRS}_t) = ((\mathbb{G}, g, q), g^{\mathbf{A}^\top})$ and $(h_1, \cdots, h_t)^\top = g^{\mathbf{A}^\top \tilde{\boldsymbol{x}}}$. Then it's sufficient to show that

$$((\mathbb{G}, g, q), g^{\mathbf{A}^\top}, g^{\mathbf{A}^\top \tilde{\boldsymbol{x}}}) \approx_c ((\mathbb{G}, g, q), g^{\mathbf{A}^\top}, \boldsymbol{u}),$$

where $\boldsymbol{u}$ is uniformly random in $\mathbb{G}^t$. Clearly we have $H_\infty(\tilde{\boldsymbol{x}}) = n \geq \log q + 2\log(1/\varepsilon)$ from the parameter setting. Then by Theorem 3.12, the above two distributions are computationally indistinguishable.

Next, we show that for any $\mathbf{M}_1, \mathbf{M}_2 \in \mathbb{Z}_B^{n\times B}$, we have

$$\mathsf{DDH\text{-}BE.Enc}(\mathsf{CRS}, u, \mathbf{M}_1) \approx_c \mathsf{DDH\text{-}BE.Enc}(\mathsf{CRS}, u, \mathbf{M}_2)$$

for uniformly random CRS, u. It's equivalent to show

$$\{(\{\hat{g}_{j,b}\}_{j\neq i, b\in\{0,1\}}, \{\mu_{i,b}^{(1)}\}_{b\in\{0,1\}})\}_{i\in[n]} \approx_c \{(\{\hat{g}_{j,b}\}_{j\neq i, b\in\{0,1\}}, \{\mu_{i,b}^{(2)}\}_{b\in\{0,1\}})\}_{i\in[n]}, \tag{11}$$

where $\mu_{i,b}^{(a)} = \mathsf{gl\text{-}enc}(\hat{g}_{i,b}, M_{i,b}^{(a)})$ and $M_{i,b}^{(a)}$ is the component of $\mathbf{M}_a$, for $a \in \{1,2\}$.

According to the DDH assumption, given $\{\hat{g}_{j,b} = g_{j,b}^{r_i}\}_{j\neq i, b\in\{0,1\}}$, u^{r_i} is computationally indistinguishable from the uniform over $\mathbb{G}$. This means that $\hat{g}_{i,b} = u^{r_i} g_{i,b}^{-r_i}$ is also computationally indistinguishable from the uniform over $\mathbb{G}$. Furthermore, by Goldreich-Levin Theorem [24], $(\{\hat{g}_{j,b}\}_{j\neq i, b\in\{0,1\}}, \{\mu_{i,b}^{(a)}\}_{b\in\{0,1\}})$ is computationally indistinguishable from $(\{\hat{g}_{j,b}\}_{j\neq i, b\in\{0,1\}}, \boldsymbol{u})$ for any $\{\mu_{i,b}^{(a)}\}_{b\in\{0,1\}}$, where $\boldsymbol{u}$ is uniformly random. Summing us above analyses, we conclude that (11) is set up.

7.2 New LWE-based BE Scheme

In this section, we first propose a new BE scheme based on LWE, and then show this BE satisfies two key-homomorphic properties in Section 6.1 and the pseudorandomness property in Definition 6.6.

Construction 7.3 (LWE-based BE) *In order to simplify the description of this construction, we set the size parameter to be $B = 2$. More formally, this* LWE*-based* BE *scheme can be described as follows.*

- $\mathsf{LWE\text{-}BE.Setup}(\lambda, n)$*: Given the security parameter λ and the size parameter $n \in \mathbb{N}$ as input, the algorithm does the following*

1. *Set $\beta > 0$ such that $\beta q > \sqrt{\tilde{n}}$, and $\beta < \frac{1}{24\tilde{n}^{3/2}}$. Select random matrices $\mathbf{A}_0 = [\boldsymbol{a}_{0,1}|\cdots|\boldsymbol{a}_{0,n}] \xleftarrow{\$} \mathbb{Z}_q^{\tilde{n}\times n}$, $\mathbf{A}_1 = [\boldsymbol{a}_{1,1}|\cdots|\boldsymbol{a}_{1,n}] \xleftarrow{\$} \mathbb{Z}_q^{\tilde{n}\times n}$;*
2. *Output $\mathsf{CRS} := (\mathbf{A}_0, \mathbf{A}_1)$;*

– $\mathsf{LWE\text{-}BE.KeyGen}(\mathsf{CRS}, \boldsymbol{x})$*: Given the common parameter* CRS *and the secret key* $\mathsf{sk} := \boldsymbol{x} \in \{0,1\}^n$ *as input, the algorithm first samples a noise vector* $\boldsymbol{e} \xleftarrow{\$} \bar{\psi}_\beta^{\tilde{n}}$*, and then computes* $\boldsymbol{h} = (\mathbf{A}_0 \cdot \bar{\boldsymbol{x}} + \mathbf{A}_1 \cdot \boldsymbol{x} + \boldsymbol{e}) \bmod q$*, where* $\bar{\boldsymbol{x}}$ *is the complement of* $\boldsymbol{x}$*. Finally, the algorithm output* $\boldsymbol{h}$ *as the public key* pk*.*

– $\mathsf{LWE\text{-}BE.Enc}(\mathsf{CRS}, \boldsymbol{h}, \mathbf{M})$*: Given the common parameter* CRS*, the public key* $\mathsf{pk} := \boldsymbol{h}$ *and a message matrix* $\mathbf{M} = (M_{i,b})_{i\in[n],b\in\{0,1\}} \in \mathbb{Z}_2^{n\times 2}$ *as input, the algorithm does the following:*

1. *For each $M_{i,b}$ with $i \in [n]$, $b \in \{0,1\}$, define $\mathbf{A}_{b,-i} = [\boldsymbol{a}_{b,1}|\cdots|\boldsymbol{a}_{b,i-1}|\boldsymbol{a}_{b,i+1}|\cdots|\boldsymbol{a}_{b,n}] \in \mathbb{Z}_q^{\tilde{n}\times(n-1)}$;*
2. *Choose $\boldsymbol{s}_{(i,b)} \xleftarrow{\$} \{0,1\}^{\tilde{n}}$, $\boldsymbol{e}_{(i,b)}, \boldsymbol{e}'_{(i,b)} \xleftarrow{\$} \bar{\psi}_\beta^{(n-1)}$ and $e''_{(i,b)} \xleftarrow{\$} \bar{\psi}_\beta$;*
3. *Compute*

$$\boldsymbol{v}_0^{(i,b)} = \left(\mathbf{A}_{0,-i}^\top \cdot \boldsymbol{s}_{(i,b)} + \boldsymbol{e}_{(i,b)}\right) \bmod q \in \mathbb{Z}_q^{n-1},$$

$$\boldsymbol{v}_1^{(i,b)} = \left(\mathbf{A}_{1,-i}^\top \cdot \boldsymbol{s}_{(i,b)} + \boldsymbol{e}'_{(i,b)}\right) \bmod q \in \mathbb{Z}_q^{n-1}$$

and

$$v_2^{(i,b)} = \left(\langle \boldsymbol{s}_{(i,b)}, \boldsymbol{h} - \boldsymbol{a}_{b,i}\rangle + e''_{(i,b)} + M_{i,b}\lfloor\frac{q}{2}\rfloor\right) \bmod q \in \mathbb{Z}_q;$$

4. *Output $\mathsf{CT} = (\boldsymbol{v}^{(i,b)})_{i\in[n],b\in\{0,1\}}$ as the ciphertext, where $(\boldsymbol{v}^{(i,b)})^\top = ((\boldsymbol{v}_0^{(i,b)})^\top \| (\boldsymbol{v}_1^{(i,b)})^\top \| v_2^{(i,b)})$.*

– $\mathsf{LWE\text{-}BE.Dec}(\mathsf{CRS}, \boldsymbol{x}, \mathsf{CT})$*: Given the common parameter* CRS*, the secret key* $\mathsf{sk} := \boldsymbol{x}$ *and a ciphertext* CT *as input, the algorithm does the following:*

1. *For each $i \in [n]$, define $\boldsymbol{x}_{-i} = (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)^\top \in \{0,1\}^{n-1}$ and $\bar{\boldsymbol{x}}_{-i}$ as the complement of $\boldsymbol{x}_{-i}$;*
2. *Compute $y_i = \left(v_2^{(i,x_i)} - \langle \bar{\boldsymbol{x}}_{-i}, \boldsymbol{v}_0^{(i,x_i)}\rangle - \langle \boldsymbol{x}_{-i}, \boldsymbol{v}_1^{(i,x_i)}\rangle\right)$. Finally, set $k_i = 0$ if y_i is closer to 0 than to $\lfloor\frac{q}{2}\rfloor$. Otherwise, set $k_i = 1$.*
3. *The algorithm finally outputs $\boldsymbol{k} = (k_1, k_2, \ldots, k_n)^\top$.*

Lemma 7.4 (Correctness of LWE-BE) *The scheme* $\mathsf{LWE\text{-}BE}$ *is correct except with a negligible probability.*

Proof. Let $\mathsf{CRS} := (\mathbf{A}_0, \mathbf{A}_1)$, $\mathsf{sk} := \boldsymbol{x}$, $\mathbf{M} = (M_{i,b})_{i\in[n],b\in\mathbb{Z}_2} \in \{0,1\}^{n\times 2}$ be arbitrary, and consider the public key $\boldsymbol{h} = (\mathbf{A}_0 \cdot \bar{\boldsymbol{x}} + \mathbf{A}_1 \cdot \boldsymbol{x} + \boldsymbol{e}) \bmod q$ and $\boldsymbol{v}^{(i,x_i)} = (\boldsymbol{v}_0^{(i,x_i)} \| \boldsymbol{v}_1^{(i,x_i)} \| v_2^{(i,x_i)})$ as the encryption of $M_{(i,x_i)} \in \{0,1\}$. Then by

the decryption algorithm, we have

$$
\begin{aligned}
& v_2^{(i,x_i)} - \langle \bar{\boldsymbol{x}}_{-i}, \boldsymbol{v}_0^{(i,x_i)} \rangle - \langle \boldsymbol{x}_{-i}, \boldsymbol{v}_1^{(i,x_i)} \rangle \\
=& \langle \boldsymbol{s}_{(i,x_i)}, \boldsymbol{h} - \boldsymbol{a}_{x_i,i} \rangle + e''_{(i,x_i)} + M_{i,x_i} \lfloor \frac{q}{2} \rfloor - \langle \bar{\boldsymbol{x}}_{-i}, \mathbf{A}_{0,-i}^{\top} \cdot \boldsymbol{s}_{(i,x_i)} + \boldsymbol{e}_{(i,x_i)} \rangle - \\
& \langle \boldsymbol{x}_{-i}, \mathbf{A}_{1,-i}^{\top} \cdot \boldsymbol{s}_{(i,x_i)} + \boldsymbol{e}'_{(i,x_i)} \rangle \\
=& \langle \boldsymbol{s}_{(i,x_i)}, \boldsymbol{h} \rangle - \langle \boldsymbol{s}_{(i,x_i)}, \boldsymbol{a}_{x_i,i} \rangle + e''_{(i,x_i)} + M_{i,x_i} \lfloor \frac{q}{2} \rfloor - \langle (\mathbf{A}_{0,-i} \cdot \bar{\boldsymbol{x}}_{-i} + \mathbf{A}_{1,-i} \cdot \boldsymbol{x}_{-i}), \boldsymbol{s}_{(i,x_i)} \rangle - \\
& \langle \bar{\boldsymbol{x}}_{-i}, \boldsymbol{e}_{(i,x_i)} \rangle - \langle \boldsymbol{x}_{-i}, \boldsymbol{e}'_{(i,x_i)} \rangle \\
=& \langle \boldsymbol{s}_{(i,x_i)}, \boldsymbol{e} \rangle + e''_{(i,x_i)} + M_{i,x_i} \lfloor \frac{q}{2} \rfloor - \langle \bar{\boldsymbol{x}}_{-i}, \boldsymbol{e}_{(i,x_i)} \rangle - \langle \boldsymbol{x}_{-i}, \boldsymbol{e}'_{(i,x_i)} \rangle.
\end{aligned}
$$

Hence, $\langle \boldsymbol{s}_{(i,x_i)}, \boldsymbol{e} \rangle + e''_{(i,x_i)} - \langle \bar{\boldsymbol{x}}_{-i}, \boldsymbol{e}_{(i,x_i)} \rangle - \langle \boldsymbol{x}_{-i}, \boldsymbol{e}'_{(i,x_i)} \rangle$ is the noise in the decryption. For $\boldsymbol{s}_{(i,x_i)} \in \{0,1\}^{\tilde{n}}$, $\bar{\boldsymbol{x}}_{-i}, \boldsymbol{x}_{-i} \in \{0,1\}^{n-1}$, $\boldsymbol{e}_{(i,j)}, \boldsymbol{e}'_{(i,j)} \xleftarrow{\$} \bar{\psi}_{\beta}^{(n-1)}$, $e''_{(i,j)} \xleftarrow{\$} \bar{\psi}_{\beta}$ and $\boldsymbol{e} \xleftarrow{\$} \bar{\psi}_{\beta}^{\tilde{n}}$, it holds

$$
\begin{aligned}
& \left| \langle \boldsymbol{s}_{(i,x_i)}, \boldsymbol{e} \rangle + e''_{(i,x_i)} - \langle \bar{\boldsymbol{x}}_{-i}, \boldsymbol{e}_{(i,x_i)} \rangle - \langle \boldsymbol{x}_{-i}, \boldsymbol{e}'_{(i,x_i)} \rangle \right| \\
\leq & \left| \langle \boldsymbol{s}_{(i,x_i)}, \boldsymbol{e} \rangle \right| + \left| e''_{(i,x_i)} \right| + \left| \langle \bar{\boldsymbol{x}}_{-i}, \boldsymbol{e}_{(i,x_i)} \rangle \right| + \left| \langle \boldsymbol{x}_{-i}, \boldsymbol{e}'_{(i,x_i)} \rangle \right| \\
\leq & \beta q \tilde{n}^{3/2} + \frac{1}{2} \tilde{n}^{3/2} + \beta q \tilde{n}^{1/2} + \frac{1}{2} + 2\beta q (\tilde{n}-1)^{3/2} + (\tilde{n}-1)^{3/2} \\
< & \beta q (6\tilde{n}^{3/2}),
\end{aligned}
$$

except with a negligible probability. Furthermore, by condition $\beta < \frac{1}{24\tilde{n}^{3/2}}$, we know that the absolute value of the above noise is bounded by $\frac{q}{4}$. As a result, the decryption algorithm of LWE-BE is correct with overwhelming probability. □

Lemma 7.5 (Security of LWE-BE) *The scheme* LWE-BE *is semantically secure under the* LWE *assumption.*

Proof. According to Definition 2.7, our target is to prove that the view of the adversary is computationally indistinguishable from one where all $\boldsymbol{v}^{(i,b)}$ are uniformly random for all $i \in [n]$ and $b \neq x_i$. Here we just focus on the computational indistinguishability with respect to one row of $\mathbf{M}$ (e.g., the i-th row of $\mathbf{M}$ for certain $i \in [n]$), which implies the security of the batch encryption scheme by a simple hybrid argument [16].

Let t be index of the row that we want to prove indistinguishability. Consider that the simulator receives the challenge $\boldsymbol{x}$ from the adversary and the LWE challenge of the form $\boldsymbol{a}'_{0,1}, \boldsymbol{a}'_{1,1}, \ldots, \boldsymbol{a}'_{0,n}, \boldsymbol{a}'_{1,n} \in \mathbb{Z}_q^{\tilde{n}}$, $\{z_{b,i}\}_{i \in [n], b \in \{0,1\}}$, where each $z_{b,i}$ is either uniformly random or from the LWE distribution, i.e., $z_{b,i} = \boldsymbol{s}^{\top} \cdot \boldsymbol{a}_{b,i} + e_{b,i}$. Define $\boldsymbol{z}_{b,-t} = (z_{b,1}, \ldots, z_{b,t-1}, z_{b,t+1}, \ldots, z_{b,n})^{\top}$ for $b \in \{0,1\}$.

The simulator sets $\mathbf{A}' = (\boldsymbol{a}'_{0,1} | \ldots | \boldsymbol{a}'_{0,n} | \boldsymbol{a}'_{1,1} | \ldots | \boldsymbol{a}'_{1,n}) \in \mathbb{Z}_q^{\tilde{n} \times 2n}$ and samples a noise vector $\boldsymbol{e}' \xleftarrow{\$} \psi_{\beta}^{\tilde{n}}$, then computes $\boldsymbol{h}' = \left(\mathbf{A}' \cdot \begin{pmatrix} \bar{\boldsymbol{x}} \\ \boldsymbol{x} \end{pmatrix} + \boldsymbol{e}' \right) \bmod q$. Then, for all

$i \neq t$ it sets $\boldsymbol{a}_{0,i} = \boldsymbol{a}'_{0,i}, \boldsymbol{a}_{1,i} = \boldsymbol{a}'_{1,i}$, and sets $\boldsymbol{a}_{x_t,t} = \boldsymbol{a}'_{x_t,t}, \boldsymbol{a}_{1-x_t,t} = \boldsymbol{a}'_{1-x_t,t} + \boldsymbol{h}'$. The simulator sets $\mathsf{CRS} := \mathbf{A} = (\boldsymbol{a}_{0,1}|\dots|\boldsymbol{a}_{0,n}|\boldsymbol{a}_{1,1}|\dots|\boldsymbol{a}_{1,n})$, and $\boldsymbol{h} = \boldsymbol{h}'$.

It's easy to see the distribution of $(\mathsf{CRS}, \boldsymbol{h})$ are identical to that in the original game. Particularly, it is easy to verify that $\mathfrak{h}(\mathsf{CRS}, \boldsymbol{x}) = \mathfrak{h}(\mathsf{CRS}', \boldsymbol{x}) = \boldsymbol{h}' = \boldsymbol{h}$.

The simulator sends $(\mathsf{CRS}, \boldsymbol{h})$ to the adversary and receives the message vectors. It then samples $\boldsymbol{s}^{(x_t)}, \boldsymbol{e}^{(x_t)}$ itself and generates $\boldsymbol{v}^{(t,x_t)}$ properly. For the other part, set

$$\boldsymbol{v}_0^{(t,1-x_t)} = \boldsymbol{z}_{0,-t}, \quad \boldsymbol{v}_1^{(t,1-x_t)} = \boldsymbol{z}_{1,-t}, \quad \textit{and} \quad v_2^{(t,1-x_t)} = -z_{1-x_t,t} + M_{1-x_t,t}.$$

We notice that if the vectors $\{z_{b,i}\}_{b\in\{0,1\},i\in[n]}$ were generated from the LWE distribution, then $\boldsymbol{v}^{(t,1-x_t)}$ has the distribution of the ciphertext. On the other hand, if the vectors are uniform, then $\boldsymbol{v}^{(t,1-x_t)}$ is uniform. Thus, if the adversary can distinguish the ciphertext from the uniform distribution, the simulator can break the LWE assumption with the same advantage. □

Two key-homomorphic properties of LWE-BE. We show that the LWE-based BE satisfies the two properties in Section 6.1. We first define algorithm $\mathcal{T}_1$ as follows:

$\mathcal{T}_1(\mathsf{CRS}, h, \boldsymbol{k}')$: Take $\mathsf{CRS} = \mathbf{A} = [\mathbf{A}_0|\mathbf{A}_1] = [\boldsymbol{a}_{0,1}|\dots|\boldsymbol{a}_{0,n}|\boldsymbol{a}_{1,1}|\dots|\boldsymbol{a}_{1,n}] \in \mathbb{Z}_q^{\tilde{n}\times 2n}$, $\boldsymbol{h} = \mathbf{A} \cdot \begin{pmatrix} \bar{\boldsymbol{x}} \\ \boldsymbol{x} \end{pmatrix} + \boldsymbol{e} \in \mathbb{Z}_q^{\tilde{n}}$ for a random key vector $\boldsymbol{x} \in \{0,1\}^n$, and a random vector $\boldsymbol{k}' \in \{0,1\}^n$ as input. Do the followings:

1. For $\boldsymbol{k}' = (k'_1, ..., k'_1) \in \{0,1\}^n$, set $\boldsymbol{a}'_{b,i} = \boldsymbol{a}_{b\oplus k'_i,i}$ for each $i \in [n], b \in \{0,1\}$, then set $\mathsf{CRS}' = \mathbf{A}' = [\boldsymbol{a}'_{0,1}|\dots|\boldsymbol{a}'_{0,n}|\boldsymbol{a}'_{1,1}|\dots|\boldsymbol{a}'_{1,n}]$.
2. Set $\boldsymbol{h}' = \boldsymbol{h}$.
3. Output $(\mathsf{CRS}', \boldsymbol{h}')$.

Then we have

$$\begin{aligned}
\boldsymbol{h} &= \mathbf{A} \cdot \begin{pmatrix} \bar{\boldsymbol{x}} \\ \boldsymbol{x} \end{pmatrix} + \boldsymbol{e} \\
&= \sum_{i\in[n]} (1 - x_i)\boldsymbol{a}_{0,i} + \sum_{i\in[n]} x_i \boldsymbol{a}_{1,i} + \boldsymbol{e} \\
&= \sum_{i\in[n]} (1 - x_i)\boldsymbol{a}'_{k'_i,i} + \sum_{i\in[n]} x_i \boldsymbol{a}'_{1-k'_i,i} + \boldsymbol{e} \\
&= \sum_{i\in[n]} (1 - (x_i \oplus k'_i))\boldsymbol{a}'_{0,i} + \sum_{i\in[n]} (x_i \oplus k'_i)\boldsymbol{a}'_{1,i} + \boldsymbol{e} \\
&= \mathbf{A}' \cdot \begin{pmatrix} \overline{(\boldsymbol{x} + \boldsymbol{k}')} \\ \boldsymbol{x} + \boldsymbol{k}' \end{pmatrix} + \boldsymbol{e} \\
&= \boldsymbol{h}'.
\end{aligned}$$

So $\boldsymbol{h}'$ can also be viewed as a projection of the vector $(\boldsymbol{x}+\boldsymbol{k}')$ under CRS', i.e., $\boldsymbol{h}' = \mathfrak{h}(\mathsf{CRS}', \boldsymbol{x}+\boldsymbol{k}')$. Then we can show that the Property 1 in Section 6.1 holds, i.e., the following equation holds

$$\left(\mathsf{CRS}, \mathfrak{h}(\mathsf{CRS}, \boldsymbol{x}+\boldsymbol{k}'), \boldsymbol{x}, \boldsymbol{k}'\right) \equiv \left(\mathcal{T}_1(\mathsf{CRS}, \mathfrak{h}(\mathsf{CRS}, \boldsymbol{x}), \boldsymbol{k}'), \boldsymbol{x}, \boldsymbol{k}'\right).$$

It's sufficient to show

$$\left(\mathsf{CRS}, \mathfrak{h}(\mathsf{CRS}, \boldsymbol{x}+\boldsymbol{k}'), \boldsymbol{x}, \boldsymbol{k}'\right) \equiv \left(\mathsf{CRS}', \mathfrak{h}(\mathsf{CRS}', \boldsymbol{x}+\boldsymbol{k}'), \boldsymbol{x}, \boldsymbol{k}'\right).$$

According to the relationship between CRS and CRS', it is clear that

$$(\mathsf{CRS}, \boldsymbol{k}') \equiv (\mathsf{CRS}', \boldsymbol{k}').$$

As a result, the Property 1 holds.

Next, we define the algorithm $\mathcal{T}_2$ in the following way:

$\mathcal{T}_2(\mathsf{CT}, \boldsymbol{k}')$: Take a random ciphertext $(\boldsymbol{v}^{(i,b)})_{i\in[n],b\in\{0,1\}}$ and $\boldsymbol{k}' \in \{0,1\}^n$ as input, where $\mathsf{CT}=\mathsf{LWE\text{-}BE.Enc}(\mathsf{CRS}, h, \mathbf{M})$ for $\mathsf{CRS} = [\mathbf{A}_0|\mathbf{A}_1], \boldsymbol{h} = \mathfrak{h}(\mathsf{CRS}, \boldsymbol{x}) = \mathbf{A}_0 \cdot \bar{\boldsymbol{x}} + \mathbf{A}_1 \cdot \boldsymbol{x} + \boldsymbol{e}$ and plaintext $\mathbf{M} = (m_{i,b})_{i\in[n],b\in\{0,1\}}$. Do the followings:

1. For the vector $\boldsymbol{k}' = (k'_1, \ldots, k'_n) \in \{0,1\}^n$, define the matrix

$$\mathbf{P}_i = [\boldsymbol{e}_{0,1}|\ldots|\boldsymbol{e}_{0,i-1}|\boldsymbol{e}_{0,i+1}|\ldots|\boldsymbol{e}_{0,n}|\boldsymbol{e}_{1,1}|\ldots|\boldsymbol{e}_{1,i-1}|\boldsymbol{e}_{1,i+1}|\ldots|\boldsymbol{e}_{1,n}] \in \{0,1\}^{2(n-1)\times 2(n-1)},$$

   where $\boldsymbol{e}_{0,j}$ is the $2(n-1)$-dimensional unit vector such that 1 is in the $j + k'_j(n-1)$-th position, and $\boldsymbol{e}_{1,j}$ is the $2(n-1)$-dimensional unit vector such that 1 is in the $j + (1-k'_j)(n-1)$-th position.
2. Set $(\boldsymbol{v}_0'^{(i,b)}|\boldsymbol{v}_1'^{(i,b)}) = (\boldsymbol{v}_0^{(i,b)}|\boldsymbol{v}_1^{(i,b)}) \cdot \mathbf{P}_i$.
3. Output $\mathsf{CT}' = \{\boldsymbol{v}'^{(i,b)}\}_{i\in[n],b\in\{0,1\}} = \{(\boldsymbol{v}_0'^{(i,b)}|\boldsymbol{v}_1'^{(i,b)}|v_2^{(i,b+k_i)})\}_{i\in[n],b\in\{0,1\}}$.

Then we have

$$\begin{aligned}(\boldsymbol{v}_0'^{(i,b)}|\boldsymbol{v}_1'^{(i,b)}) &= \left(\boldsymbol{s}_{(i,b)}[\mathbf{A}_{0,-i}|\mathbf{A}_{1,-i}] + (\boldsymbol{e}_{i,b}|\boldsymbol{e}'_{i,b})\right) \cdot \mathbf{P}_i \\ &= \boldsymbol{s}_{(i,b)} \cdot [\mathbf{A}_{0,-i}|\mathbf{A}_{1,-i}] \cdot \mathbf{P}_i + (\boldsymbol{e}_{i,b}|\boldsymbol{e}'_{i,b}) \cdot \mathbf{P}_i \\ &= \boldsymbol{s}_{(i,b)} \cdot \mathbf{A}'_{-i} + (\boldsymbol{e}_{i,b}|\boldsymbol{e}'_{i,b}) \cdot \mathbf{P}_i,\end{aligned}$$

where

$$\mathbf{A}'_{-i} = [\boldsymbol{a}_{k'_1,1}|\ldots|\boldsymbol{a}_{k'_{i-1},i-1}|\boldsymbol{a}_{k'_{i+1},i+1}|\ldots|\boldsymbol{a}_{k'_n,n}|\boldsymbol{a}_{1-k'_1,1}|\ldots|\boldsymbol{a}_{1-k'_{i-1},i-1}|\boldsymbol{a}_{1-k'_{i+1},i+1}|\ldots|\boldsymbol{a}_{1-k'_n,n}].$$

So $\boldsymbol{v}'^{(i,b)}$ can be represented as

$$\begin{aligned}\boldsymbol{v}'^{(i,b)} &= \boldsymbol{s}_{(i,b)} \cdot [\mathbf{A}'_{-i}|\boldsymbol{h} - \boldsymbol{a}_{b+k'_i,i}] + [(\boldsymbol{e}_{i,b}|\boldsymbol{e}'_{i,b}) \cdot \mathbf{P}_i|e''_{i,b+k'_i}] + M_{i,b}\lfloor q/2\rfloor \\ &= \boldsymbol{s}_{(i,b)} \cdot [\mathbf{A}'_{-i}|\boldsymbol{h}' - \boldsymbol{a}'_{b,i}] + [(\boldsymbol{e}_{i,b}|\boldsymbol{e}'_{i,b}) \cdot \mathbf{P}_i|e''_{i,b+k'_i}] + M_{i,b}\lfloor q/2\rfloor.\end{aligned}$$

By the setting of $\mathbf{P}_i$, the distribution of $(\boldsymbol{e}_{i,b}|\boldsymbol{e}'_{i,b}) \cdot \mathbf{P}_i$ is identical to the original vector. So the resulting ciphertext CT' can be viewed as a valid ciphertext of $\mathbf{M}$ under $\mathsf{CRS}' := \mathbf{A}'$ and $\boldsymbol{h} = \mathfrak{h}(\mathsf{CRS}', \boldsymbol{x}+\boldsymbol{k}')$.

On the other hand, we want to show the property 2 in Section 6.1 holds, i.e., the following equation holds

$$(\mathsf{CT}_1, \mathcal{T}_1(\mathsf{CRS}, \mathfrak{h}(\mathsf{CRS}, \boldsymbol{x}), \boldsymbol{k}), \boldsymbol{x}, \boldsymbol{k}) \equiv (\mathcal{T}_2(\mathsf{CT}, \boldsymbol{k}), \mathcal{T}_1(\mathsf{CRS}, \mathfrak{h}(\mathsf{CRS}, \boldsymbol{x}), \boldsymbol{k}), \boldsymbol{x}, \boldsymbol{k}),$$

where $\mathsf{CT} \leftarrow \mathit{\Pi}.\mathsf{Enc}(\mathsf{CRS}, \mathfrak{h}(\mathsf{CRS}, \boldsymbol{x}), \mu)$, and $\mathsf{CT}_1 \leftarrow \mathit{\Pi}.\mathsf{Enc}(\mathsf{CRS}, \mathfrak{h}(\mathsf{CRS}, \boldsymbol{x} + \boldsymbol{k}), \mu)$. By our setting, $\mathcal{T}_2(\mathsf{CT}, \boldsymbol{k}) = \mathsf{CT}' = \mathit{\Pi}.\mathsf{Enc}(\mathsf{CRS}', \mathfrak{h}(\mathsf{CRS}', \boldsymbol{x} + \boldsymbol{k}'), \mu)$. It's sufficient to show

$$\begin{aligned}&(\mathit{\Pi}.\mathsf{Enc}(\mathsf{CRS}, \mathfrak{h}(\mathsf{CRS}, \boldsymbol{x} + \boldsymbol{k}), \mu), \mathcal{T}_1(\mathsf{CRS}, \mathfrak{h}(\mathsf{CRS}, \boldsymbol{x}), \boldsymbol{k}), \boldsymbol{x}, \boldsymbol{k})\\ \equiv &\left(\mathit{\Pi}.\mathsf{Enc}(\mathsf{CRS}', \mathfrak{h}(\mathsf{CRS}', \boldsymbol{x} + \boldsymbol{k}'), \mu), \mathcal{T}_1(\mathsf{CRS}, \mathfrak{h}(\mathsf{CRS}, \boldsymbol{x}), \boldsymbol{k}), \boldsymbol{x}, \boldsymbol{k}\right),\end{aligned}$$

which is clear by our setting. As a result, the Property 2 holds.

Pseudorandom properties of LWE-BE. We show that the BE scheme constructed above satisfies the pseudorandom property as definition 6.6 under LWE assumption. Firstly, we need to show that for any polynomial $t = \mathsf{poly}(\lambda)$, the following two distributions are computationally indistinguishable:

$$\begin{pmatrix} \mathsf{CRS}_1, \cdots, \mathsf{CRS}_t \\ h_1, \ \cdots, \ h_t \end{pmatrix} \approx_c \begin{pmatrix} \mathsf{CRS}_1, \cdots, \mathsf{CRS}_t \\ u_1, \ \cdots, \ u_t \end{pmatrix},$$

where $\{\mathsf{CRS}_s\}_{s\in[t]}$ and $\{u_s\}_{s\in[t]}$ are uniformly random, $\boldsymbol{x} \xleftarrow{\$} \{0,1\}^n$, and $h_s = \mathfrak{h}(\mathsf{CRS}_s, \boldsymbol{x})$ for all $s \in [t]$. By our construction, $\mathsf{CRS}_s = \mathbf{A}^s = [\boldsymbol{a}_{0,1}^{(s)}| \ldots |\boldsymbol{a}_{0,n}^{(s)}|\boldsymbol{a}_{1,1}^{(s)}| \ldots |\boldsymbol{a}_{1,n}^{(s)}]$, $\boldsymbol{h}_s = \mathbf{A}^s \cdot \begin{pmatrix} \bar{\boldsymbol{x}} \\ \boldsymbol{x} \end{pmatrix} + \boldsymbol{e}_s$, for each $s \in [t]$. Therefore, we can rephrase $\begin{pmatrix} \mathsf{CRS}_1, \cdots, \mathsf{CRS}_t \\ h_1, \ \cdots, \ h_t \end{pmatrix}$ as $\left(\mathbf{A}^*, \mathbf{A}^* \cdot \begin{pmatrix} \bar{\boldsymbol{x}} \\ \boldsymbol{x} \end{pmatrix} + \boldsymbol{e}^*\right)$, where $\mathbf{A}^* = \begin{bmatrix} \mathbf{A}^1 \\ \vdots \\ \mathbf{A}^t \end{bmatrix} \in \mathbb{Z}_q^{t\tilde{n}\times 2n}$, $\boldsymbol{e}^* = (\boldsymbol{e}_1| \ldots |\boldsymbol{e}_t) \in \psi_\beta^{t\tilde{n}}$.

Then we need to show that

$$(\mathbf{A}^*, \mathbf{A}^* \cdot \begin{pmatrix} \bar{\boldsymbol{x}} \\ \boldsymbol{x} \end{pmatrix} + \boldsymbol{e}^*) \overset{c}{\approx} (\mathbf{A}^*, \boldsymbol{u}),$$

where $\boldsymbol{u}$ is uniformly random in $\mathbb{Z}_q^{t\tilde{n}}$. In order to show this, we decompose $\mathbf{A}^*$ into two part $\mathbf{A}_0^*$ and $\mathbf{A}_1^*$, where $\mathbf{A}_b^* = \begin{bmatrix} \mathbf{A}_b^1 \\ \vdots \\ \mathbf{A}_b^t \end{bmatrix}$, then $\left(\mathbf{A}^*, \mathbf{A}^* \cdot \begin{pmatrix} \bar{\boldsymbol{x}} \\ \boldsymbol{x} \end{pmatrix} + \boldsymbol{e}^*\right) = (\mathbf{A}_0^*, \mathbf{A}_1^*, \mathbf{A}_0^* \cdot \bar{\boldsymbol{x}} + \mathbf{A}_1^* \cdot \boldsymbol{x} + \boldsymbol{e}^*)$. If we can show

$$(\mathbf{A}_0^*, \mathbf{A}_1^*, \mathbf{A}_0^* \cdot \bar{\boldsymbol{x}} + \boldsymbol{e}_0^*, \mathbf{A}_1^* \cdot \boldsymbol{x} + \boldsymbol{e}_1^*) \overset{c}{\approx} (\mathbf{A}_0^*, \mathbf{A}_1^*, \boldsymbol{u}_0^*, \boldsymbol{u}_1^*),$$

where $\boldsymbol{e}_b^* \leftarrow \psi_{\frac{\beta}{\sqrt{2}}}^{t\tilde{n}}$ for $b \in \{0,1\}$, then we can easily derive the following approximate equation

$$(\mathbf{A}_0^*, \mathbf{A}_1^*, \mathbf{A}_0^* \cdot \bar{\boldsymbol{x}} + \mathbf{A}_1^* \cdot \boldsymbol{x} + \boldsymbol{e}^*) \overset{c}{\approx} (\mathbf{A}_0^*, \mathbf{A}_1^*, \boldsymbol{u}^*),$$

for $\boldsymbol{e}^* \in \psi_\beta^{t\bar{n}}$. It's easy to see that the distribution of $(\mathbf{A}_0^*, \mathbf{A}_1^*, \mathbf{A}_0^* \cdot \bar{\boldsymbol{x}} + \boldsymbol{e}_0^*, \mathbf{A}_1^* \cdot \boldsymbol{x} + \boldsymbol{e}_1^*)$ is identical to $(\mathbf{A}_0^*, \mathbf{A}_1^*, \mathbf{A}_0^* - (\mathbf{A}_0^* \cdot \boldsymbol{x} + \boldsymbol{e}_0^*), \mathbf{A}_1^* \cdot \boldsymbol{x} + \boldsymbol{e}_1^*)$, it's sufficient to show

$$(\mathbf{A}_0^*, \mathbf{A}_1^*, \mathbf{A}_0^* \cdot \boldsymbol{x} + \boldsymbol{e}_0^*, \mathbf{A}_1^* \cdot \boldsymbol{x} + \boldsymbol{e}_1^*) \overset{c}{\approx} (\mathbf{A}_0^*, \mathbf{A}_1^*, \boldsymbol{u}_0'^*, \boldsymbol{u}_1'^*)\,,$$

which can be proved by the hardness of LWE with Binary Secrets [34].

8 Putting Things Together

By instantiating Construction 5.3 with (1) the specific reusable extractor from LWE in Construction 3.6 and (2) the LWE-based BE in Construction 7.3, we are able to achieve the following corollary via Theorems 6.4, 6.7:

Corollary 8.1 *Assuming that* LWE *is hard, there exists a rate-1 (both information and leakage rates)* PKE *that is leakage resilient against block leakage and* $\mathsf{KDM}^{(\bar{n})}$*-secure w.r.t.* block*-affine functions for any unbounded polynomial* $\bar{n}$*.*

Similarly, by instantiating Construction 5.3 with (1) the specific reusable extractor from DDH in Construction 3.11 and (2) the DDH-based BE in Construction 7.2, we are able to achieve the following corollary via Theorems 6.4, 6.7:

Corollary 8.2 *Assuming that* DDH *is hard, there exists a rate-1 (both information and leakage rates)* PKE *that is leakage resilient against block leakage and* $\mathsf{KDM}^{(\bar{n})}$*-secure w.r.t.* block*-affine functions for any unbounded polynomial* $\bar{n}$*.*

We notice that the overall construction of the DDH-based scheme resembles a modification of the scheme of [12]. We do not present this variant. Instead, we take a more modular approach by identifying a framework that suffices for KDM security and can be instantiated from various assumptions.

Remark 8.3 *The class of block affine functions is more restricted than the regular (bit) affine class. In particular, each output component of a block affine function can depend only on one block of the input, whereas the output of a bit affine function can depend on every bit of the input. Nevertheless, this restricted class already suffices for* KDM *amplification to any bounded-size functions, and moreover allows constructions with better information rate. We discuss how to amplify the function class in the following section.*

9 Extension 1: Enlarge the class of KDM functions

In this section, we further extend our above results in Section 8, i.e., enlarging the class of KDM functions via Garbled Circuits.

9.1 Garbled Circuits

In this section, we recall the key ingredient for the KDM amplification of Applebaum [7]: Garbled Circuits.

Definition 9.1 (Garbled Circuits [16]) *A garbling scheme consists of three algorithms* $(\mathsf{Garble}, \mathsf{Eval}, \mathsf{Sim})$ *as follows:*

- $\mathsf{Garble}(1^\lambda, 1^n, 1^m, C)$ *is a* PPT *algorithm that first takes as input* λ*, a circuit* $C : \{0,1\}^n \to \{0,1\}^m$ *together with its input length* n *and output length* m*, and then outputs a garbled circuit* $\widehat{C}$ *along with labels* $\{\mathsf{lab}_{i,b}\}_{i\in[n],b\in\{0,1\}}$*, where each label* $\mathsf{lab}_{i,b} \in \{0,1\}^\lambda$*.*
- $\mathsf{Eval}(1^\lambda, \widehat{C}, \widehat{L})$ *is a deterministic algorithm that first takes as input a garbled circuit* $\widehat{C}$ *along with a set of* n *labels* $\widehat{L} = \{\mathsf{lab}_i\}_{i\in[n]}$*, and then outputs a string* $y \in \{0,1\}^m$*.*
- $\mathsf{Sim}(1^\lambda, 1^{|C|}, 1^n, y)$ *is a* PPT *algorithm that first takes as input* λ *and a bit description length of circuit* C*, an input length* n *and a string* $y \in \{0,1\}^m$*, then outputs a simulated garbled circuit* $\widetilde{C}$ *and labels* $\widetilde{L} = \{\widetilde{\mathsf{lab}}_i\}_{i\in[n]}$*.*

Moreover, the garbling scheme needs to satisfy the following two properties.

1. ***Correctness****. For any circuit* $C : \{0,1\}^n \to \{0,1\}^m$*, any input* $x = (x_i)_{i\in[n]} \in \{0,1\}^n$*, and any* $(\widehat{C}, \{\mathsf{lab}_{i,b}\}) \leftarrow \mathsf{Garble}(C)$*, it holds* $\mathsf{Eval}(\widehat{C}, \widehat{L}) = C(x)$ *where* $\widehat{L} = \{\mathsf{lab}_{i,x_i}\}_{i\in n}$*.*
2. ***Simulation Security****. For any circuit* $C : \{0,1\}^n \to \{0,1\}^m$*, any input* $x = (x_i)_{i\in[n]} \in \{0,1\}^n$*, the following two distributions are computational indistinguishability:*

$$\begin{aligned}&\{(\widehat{C}, \widehat{L}) : (\widehat{C}, \{\mathsf{lab}_{i,b}\}) \leftarrow \mathsf{Garble}(C), \widehat{L} = \{\mathsf{lab}_{i,x_i}\}_{i\in n}\}\\ \approx&\{(\widetilde{C}, \widetilde{L}) : (\widetilde{C}, \widetilde{L}) \leftarrow \mathsf{Sim}(1^\lambda, 1^{|C|}, 1^n, C(x))\}.\end{aligned}$$

9.2 Bootstrapping to Larger Classes of KDM Functions

We first present a bootstrapped variant of Construction 5.3 by using the technique of garbled circuits.[15] Then, we analyze the KDM-security and information rate of this improved scheme.

Construction 9.2 (Amplification of Our KDM Security) *Let* $\Pi = \Pi.\{\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}\}$ *be the* PKE *of Construction 5.3 instantiated with parameter* $B = 2$ *such that its secret key size* $|\mathsf{sk}| = n = n' \cdot m$*. And let* $\mathsf{GC} = \mathsf{GC}.(\mathsf{Garble}, \mathsf{Eval}, \mathsf{Sim})$ *be a garbled scheme, whose label size* $|\mathsf{lab}_{i,j}|$ *is equivalent to the bit length of element in* $\mathcal{M}$*. Then, we construct a new scheme* $\widehat{\Pi} = \widehat{\Pi}.\{\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}\}$ *with the message space* $\widehat{\mathcal{M}} = \mathcal{M}^{(t-n'+1)\times m}$ *as follows:*

[15] In [7], Applebaum leverages the abstract notion of randomized encoding to achieve KDM amplification. Here, we directly amplify our scheme through using Garbled Circuits, which is a well-known instantiation of randomized encoding.

– $\widehat{\Pi}.\mathsf{KeyGen}(1^\lambda)$*: The algorithm gets* $(\mathsf{pk},\mathsf{sk})$ *just as* $\Pi.\mathsf{KeyGen}(1^\lambda)$.
– $\widehat{\Pi}.\mathsf{Enc}(\mathsf{pk},\mu)$*: Given a public-key* pk *and a message* $\mu=\{\mu_{i,j}\}_{i\in[t-n'+1],j\in[m]}\in\mathcal{M}^{(t-n'+1)\times m}$ *as input, the algorithm first invokes* $(\widetilde{C},\widetilde{L})\leftarrow\mathsf{GC.Sim}(\mu_{1,1},\ldots,\mu_{1,m})$ *with* $\widetilde{L}=\{\mathsf{lab}_{i,j}\}_{i\in[n'],j\in[m]}$*, and then runs* $\Pi.\mathsf{Enc}$ *to output the ciphertext*

$$\begin{aligned}\mathsf{CT}:=&\Big(\widetilde{C},\Pi.\mathsf{Enc}(\mathsf{pk},\widetilde{L},\{\mu_{i,j}\}_{i\in[2,t-n'+1],j\in[1,m]})\Big)\\=&\Big(\widetilde{C},\mathsf{CT}_0,r_1,\{\mathsf{Ext}(r_1,\boldsymbol{k}_1)+\mathsf{lab}_{1,1},\ldots,\mathsf{Ext}(r_1,\boldsymbol{k}_m)+\mathsf{lab}_{1,m}\},\\&\qquad\ldots,r_{n'},\{\mathsf{Ext}(r_{n'},\boldsymbol{k}_1)+\mathsf{lab}_{n',1},\ldots,\mathsf{Ext}(r_{n'},\boldsymbol{k}_m)+\mathsf{lab}_{n',m}\},\\&\qquad\quad r_{n'+1},\{\mathsf{Ext}(r_{n'+1},\boldsymbol{k}_1)+\mu_{2,1},\ldots,\mathsf{Ext}(r_{n'+1},\boldsymbol{k}_m)+\mu_{2,m}\},\\&\qquad\ldots,r_t,\{\mathsf{Ext}(r_t,\boldsymbol{k}_1)+\mu_{(t-n'+1),1},\ldots,\mathsf{Ext}(r_t,\boldsymbol{k}_m)+\mu_{(t-n'+1),m}\}\Big).\end{aligned}$$

Here, we use $\{\mathsf{lab}_{i,j}\}_{i\in[n']}$ *to denote the garbled results of the* j*-th block of* sk *for any* $j\in[m]$.
– $\widehat{\Pi}.\mathsf{Dec}(\mathsf{sk},\mathsf{CT})$*: Given a ciphertext* CT *and a secret key* sk *as input, the algorithm first runs* $\Pi.\mathsf{Dec}$ *to recover all* $\{\mathsf{lab}_{i,j}\}_{i\in[n'],j\in[m]}$ *and* $\{\mu'_{i,j}\}_{i\in[2,t-n'+1],j\in[m]}$*, and then runs* $\mathsf{GC.Dec}(\widetilde{C},\{\mathsf{lab}_{i,j}\})$ *to get* $\{\mu'_{1,j}\}_{j\in[m]}$*. Finally, the algorithm outputs*

$$\mu'=\{\mu'_{i,j}\}_{i\in[t-n'+1],j\in[m]}\in\mathcal{M}^{(t-n'+1)\times m}.$$

Remark 9.3 *For simplicity of presentation, we have implicitly assumed that* $|\mathsf{lab}_{i,j}|=|\mathcal{M}|$*. For the more general case such that* $|\mathsf{lab}_{i,j}|>|\mathcal{M}|$*, we can easily handle through using many more elements in* $\mathcal{M}$ *to cover each* $\mathsf{lab}_{i,j}$.

It is not hard to verify that the correctness of $\widehat{\Pi}$ follows from that of the underlying scheme Π and garble scheme GC. Below, we first argue the KDM-security of the scheme $\widehat{\Pi}$, and then analyze its information rate.

Before presenting the formal theorem about the KDM security of $\widehat{\Pi}$, we define a particular KDM function class $\hat{\mathcal{F}}=(\mathcal{F}_s||\mathcal{Q}^\tau)$ as follows.

Definition 9.4 *Let* $\mathcal{F}_s$ *be the class of functions of the secret key* $\mathsf{sk}:=\boldsymbol{x}\in\mathbb{Z}_B^n$*, where the circuit size of each function in* $\mathcal{F}_s$ *is up to* s*. Let* $\mathcal{Q}^\tau$ *denote the block-affine function class* $\{\mathfrak{g}':\mathbb{Z}_B^n\to\mathcal{M}^{\tau\times m}\}$*, which is defined similarly as in Definition 2.16. Moreover,* $(\mathcal{F}_s||\mathcal{Q}^\tau)$ *denotes the concatenation of two classes, i.e., every function* $\mathfrak{f}$ *in the class can be represented by* $\mathfrak{f}=(\mathfrak{h},\mathfrak{q})$ *for some* $\mathfrak{h}\in\mathcal{F}_s$ *and* $\mathfrak{q}\in\mathcal{Q}^\tau$ *such that* $\mathfrak{f}(\mathsf{sk})=(\mathfrak{h}(\mathsf{sk})||\mathfrak{q}(\mathsf{sk}))$.

Theorem 9.5 *For the parameter setting in Construction 9.2, if* Π *is* $\mathsf{KDM}^{(1)}$*-secure with respect to* $\mathcal{G}^t=\{\mathfrak{g}':\mathbb{Z}_B^n\to\mathcal{M}^{t\times m}\}$ *as defined in Definition 2.16, and* GC *is a secure garbling scheme, then* $\widehat{\Pi}$ *is* $\mathsf{KDM}^{(1)}$*-secure with respect to* $\widehat{\mathcal{F}}=(\mathcal{F}_s||\mathcal{Q}^\tau)$ *as defined in Definition 9.4.*

Proof (Sketch). As pointed out by [7], we just need to focus on KDM reduction from $\mathcal{F}_s$ to the corresponding part of block-affine function class $\mathcal{G}^t$, denoted by $\mathcal{G}^{n'}$, i.e., $\mathcal{F}_s\le_{\mathsf{KDM}}\mathcal{G}^{n'}$. Particularly, it suffices to show that block-affine functions

in $\mathcal{G}^{n'}$ can encode any bounded size circuits of $\boldsymbol{x} \in \mathbb{Z}_2^n$, according to Applebaum's concepts on the KDM reduction in [7].

More specifically, suppose that $\mathcal{A}$ is the adversary against the KDM-security of $\widehat{\Pi}$ with respect to $\mathfrak{h} \in \mathcal{F}_s$, and $\mathcal{C}$ is the challenger for the KDM-security of Π with respect to $\mathcal{G}^{n'}$. Then, through using $\mathcal{A}$ as a building block, we can establish a reduction algorithm $\mathcal{B}$ to break the KDM-security of Π with the same advantage as that of $\mathcal{A}$.

In particular, after receiving a function $\mathfrak{h}(\cdot) \in \mathcal{F}_s$ of sk from $\mathcal{A}$, $\mathcal{B}$ conducts the followings

1. Choose $2n$ labels $\{\mathsf{lab}_{i,j,0}, \mathsf{lab}_{i,j,1}\}_{i\in[n'],j\in[m]}$, with $|\mathsf{sk}| = n = n' \cdot m$.
2. For each $j \in [m]$,
   - Set a matrix $\mathbf{A}^{(j)} = (\boldsymbol{a}_1^{(j)}, \ldots, \boldsymbol{a}_{n'}^{(j)})$ of dimension $(n' \times n')$, where for $l \in [n']$, the l-th component of $\boldsymbol{a}_l^{(j)}$ is $(\mathsf{lab}_{l,j,1} - \mathsf{lab}_{l,j,0})$ and all others are 0.
   - Set a vector $\boldsymbol{b}^{(j)} = (\mathsf{lab}_{1,j,0}, \ldots, \mathsf{lab}_{n',j,0})^\top$ of n' dimension.
   - Take $(\mathbf{A}^{(j)})^\top$ and $\boldsymbol{b}^{(j)}$ as the index of the j-th block-affine function $\mathfrak{g}_j(\mathsf{sk}_j) = (\mathbf{A}^{(j)})^\top \cdot \mathsf{sk}_j + \boldsymbol{b}^{(j)}$, where $\mathsf{sk}_j \in \{0,1\}^{n'}$ is the j-th block of sk.
3. Send the indexes of all m block-affine functions to $\mathcal{C}$ to conduct KDM query.
4. Receive the KDM ciphertexts $\{\mathsf{ct}_{i,j}\}_{i\in[n'],j\in[m]}$ from $\mathcal{C}$.
5. Run the algorithm $\mathsf{GC.Garble}$ to obtain the garbled circuit $\widehat{C}$ with respect to the KDM query function $\mathfrak{h}(\cdot)$ from $\mathcal{A}$.
6. Send $\mathsf{CT} := (\widehat{C}, \{\mathsf{ct}_{i,j}\}_{i\in[n'],j\in[m]})$ to $\mathcal{A}$.
7. Finally, $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs.

It is not hard to verify that $\mathsf{ct}_{i,j}$ will be a encryption of $\mathsf{lab}_{i,j,b}$ for $b := \mathsf{sk}_{i,j}$. Thus, the above reduction process is clearly set up. Finally, this theorem holds. □

Remark 9.6 *Although Theorem 9.5 just focuses on the case of* $\mathsf{KDM}^{(1)}$*, the above construction and analysis can be easily (though somewhat tedious) extended to* $\mathsf{KDM}^{(\bar{n})}$ *for any polynomially unbounded* $\bar{n}$*.*

Finally, we focus on the information rate of the above construction. We remark that for this amplified KDM function class $\hat{\mathcal{F}} = (\mathcal{F}_s || \mathcal{Q}^\tau)$, the parameters t, s and τ should satisfy: $\tau < t$ and s is the size of circuits amplified from block-affine function with outputs $(t - \tau)$ vectors over $\mathcal{M}^m$.

By setting $\tau \gg s$, our scheme achieves the optimal information rate, i.e., $1 - o(1)$. This is because although the additional garble circuit in the ciphertext and the encryption of labels will increase the ciphertext length to certain bounded size, we can use large enough $\tau \gg s$ such that the last τ part of ciphertext dominates the whole information rate.

10 Extension 2: Upgrade to KDM-Secure and Leakage Resilient IBE

In this section, we further extend our above results in Section 8 to generalize our results to the setting of IBE.

Particularly, we present our general compiler to construct an IBE that is both KDM-secure and leakage resilient. The compiler uses as key ingredients an IB-wHPS (ref. Definition 2.13) with additional structure (ref. Remark 4.2) and an on-the-fly KDM-secure PKE (ref. Section 10.1). Conceptually, this general IBE scheme can be view as the hybrid encryption of the IB-wHPS and PKE: to encrypt a message m, the IBE encryption algorithm first generates (1) a pair of encapsulated key and ciphertext $(\mathsf{CT}, \boldsymbol{k})$ according to the IB-wHPS, and then generates (2) a pair of session public-key and ciphertext according to the PKE, i.e., $\mathsf{pk} = (\mathsf{CRS}, \mathsf{H}(\mathsf{CRS}, \boldsymbol{k}))$ and $\mathsf{Enc}(\mathsf{pk}, m)$, respectively, under the same encapsulated key $\boldsymbol{k}$. By connecting the two security properties in a novel way, we are able to derive the desired IBEwith optimal information rate and leakage resilient security.

10.1 On-the-fly KDM-security

Given a BE-based PKE $\Pi = \Pi.\{\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}\}$, we define a new security notion called on-the-fly KDM-security. Intuitively, this on-the-fly notion captures security where the CRS and $\mathsf{H}(\mathsf{CRS}, \boldsymbol{x})$ are generated on-the-fly, i.e., upon a query for a new ciphertext, the adversary gets a newly sampled CRS and $\mathsf{H}(\mathsf{CRS}, \boldsymbol{x})$ where $\boldsymbol{x}$ remains the same secret. This is different from the classical security guaranteed by PKE, where CRS and $\mathsf{H}(\mathsf{CRS}, \boldsymbol{x})$ are sampled at the beginning and will remain the same for the whole security experiment. We identify that this on-the-fly notion is the key to our compiler.

Definition 10.1 (On-the-fly $\mathsf{KDM}^{(n)}$-security) *Let $\Pi.\{\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}\}$ be a BE-based PKE (as Definition 6.1), and let $\mathcal{F}$ be a class of functions $\mathcal{F} := \{\mathsf{f} : \mathcal{SK}^n \to \mathcal{M}\}$, where $n > 0$ is an integer, $\mathcal{SK}$ and $\mathcal{M}$ denote the secret key space and the message space, respectively. Then the on-the-fly $\mathsf{KDM}^{(n,Q)}$-security with respect to $\mathcal{F}$ can be defined by the following experiment. Let $\mathcal{A}$ be an adversary and Q be the upper bound of the number of KDM queries.*

Experiment $\mathbf{Exp}^{\mathcal{F}\text{-OTF KDM}}_{\mathsf{PKE},\mathcal{A}}(\lambda)$

Key Generation: The challenger selects a random bit $b \xleftarrow{\$} \{0,1\}$. It then sample n secret keys $\{\mathsf{sk}_i\}_{i\in[n]}$, which are n vectors $\{\boldsymbol{a}_i\}_{i\in[n]}$ according to the BE-based PKE.

KDM Queries: The adversary $\mathcal{A}$ repeatly queries the challenger with $(i, \mathsf{f}) \in [n] \times \mathcal{F}$. Upon each query, the challenger samples a new CRS and computes $h = \mathsf{H}(\mathsf{CRS}, \boldsymbol{a}_i)$. Then he sets $\mathsf{pk} = (\mathsf{CRS}, h)$, $y = \mathsf{f}(\mathsf{sk}_1, \ldots, \mathsf{sk}_n) \in \mathcal{M}$, and $c \xleftarrow{\$} \mathsf{PKE.Enc}(\mathsf{pk}, y)$ if $b = 0$ or $c \xleftarrow{\$} \mathsf{PKE.Enc}(\mathsf{pk}, u)$ where $u \xleftarrow{\$} \{0,1\}^{|y|}$ if $b = 1$. Finally the challenger returns (pk, c) to the adversary $\mathcal{A}$.

Output: The adversary $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$.

We define the advantage of $\mathcal{A}$ in the above experiment to be

$$\mathbf{Adv}^{\mathcal{F}\text{-OTF KDM}}_{\mathsf{PKE},\mathcal{A}}(\lambda) = |\Pr[b = b'] - 1/2| \,.$$

A PKE *scheme is said to be on-the-fly* $\mathsf{KDM}^{(n,Q)}$*-secure with respect to $\mathcal{F}$ if for any* PPT *adversary $\mathcal{A}$ who makes at most Q* KDM *queries, we have* $\mathbf{Adv}^{\mathcal{F}\text{-OTF KDM}}_{\mathsf{PKE},\mathcal{A}}(\lambda) \leq \mathsf{negl}(\lambda)$.

For the clarity of our presentation, we defer the instantiation of on-the-fly KDM-secure PKE to Section 11.1.

10.2 Generic Construction of KDM-secure and Leakage Resilient IBE

In this section, we present our general compiler to construct an IBE that is both leakage resilient and KDM-secure. The compiler uses as key ingredients an IB-wHPS (ref. Definition 2.13) with additional structure (ref. Remark 4.2) and an on-the-fly KDM-secure PKE (ref. Section 10.1). Conceptually, this general IBE scheme can be view as the hybrid encryption of the IB-wHPS and PKE: to encrypt a message m, the IBE encryption algorithm first generates (1) a pair of encapsulated key and ciphertext $(\mathsf{CT}, \boldsymbol{k})$ according to the IB-wHPS, and then generates (2) a pair of session public-key and ciphertext according to the PKE, i.e., $\mathsf{pk} = (\mathsf{CRS}, \mathsf{H}(\mathsf{CRS}, \boldsymbol{k}))$ and $\mathsf{Enc}(\mathsf{pk}, m)$, respectively, under the same encapsulated key $\boldsymbol{k}$. By connecting the two security properties in a novel way, we are able to derive the desired IBE.

Construction 10.2 (KDM-secure IBE) *Let* $\Pi_{\mathsf{IB\text{-}wHPS}} = \Pi_{\mathsf{IB\text{-}wHPS}}.\{\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encap}, \mathsf{Encap}^*, \mathsf{Decap}\}$ *be an* IB-wHPS *with the encapsulated key space $\mathcal{K}$ and the identity space $\mathcal{ID}$. Let* $\Pi_{\mathsf{PKE}} = \Pi_{\mathsf{PKE}}.\{\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}\}$ *be a* BE*-based* PKE*. Then, we construct an* IBE *scheme* $\Pi_{\mathsf{IBE}} = \Pi_{\mathsf{IBE}}.\{\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}\}$ *for message space $\mathcal{M}$ as follows.*

- $\Pi_{\mathsf{IBE}}.\mathsf{Setup}(1^\lambda)$*: The algorithm runs* $(\mathsf{mpk}^{\Pi_{\mathsf{IB\text{-}wHPS}}}, \mathsf{msk}^{\Pi_{\mathsf{IB\text{-}wHPS}}}) \xleftarrow{\$} \Pi_{\mathsf{IB\text{-}wHPS}}.\mathsf{Setup}(1^\lambda)$*, and then outputs* $\mathsf{mpk} := \mathsf{mpk}^{\Pi_{\mathsf{IB\text{-}wHPS}}}$ *and* $\mathsf{msk} := \mathsf{msk}^{\Pi_{\mathsf{IB\text{-}wHPS}}}$.
- $\Pi_{\mathsf{IBE}}.\mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$*: Given a master secret-key* msk *and an identity* $\mathsf{id} \in \mathcal{ID}$ *as input, the algorithm runs* IB-wHPS.KeyGen *to generate and output* $\mathsf{sk}_{\mathsf{id}} := \mathsf{sk}_{\mathsf{id}}^{\Pi_{\mathsf{IB\text{-}wHPS}}} \xleftarrow{\$} \Pi_{\mathsf{IB\text{-}wHPS}}.\mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$.
- $\Pi_{\mathsf{IBE}}.\mathsf{Enc}(\mathsf{mpk}, \mathsf{id}, \mu)$*: Given a master public-key* mpk*, an identity* $\mathsf{id} \in \mathcal{ID}$ *and a message $m \in \mathcal{M}$ as input, the algorithm does the following steps:*
  1. *Generates* $(\mathsf{CT}_1, \boldsymbol{k}) \leftarrow \Pi_{\mathsf{IB\text{-}wHPS}}.\mathsf{Encap}(\mathsf{mpk}, \mathsf{id})$*;*
  2. *Chooses an on-the-fly common reference string* CRS *for* Π_{PKE}*;*
  3. *Computes* $\mathsf{CT}_2 = \Pi_{\mathsf{PKE}}.\mathsf{Enc}(\mathsf{CRS}, h, \mu)$ *where* $h = \mathsf{H}(\mathsf{CRS}, \boldsymbol{k})$*;*
  4. *Outputs* $\mathsf{CT} = (\mathsf{CT}_1, \mathsf{CRS}, h, \mathsf{CT}_2)$ *as the ciphertext of m under the identity* id.
- $\Pi_{\mathsf{IBE}}.\mathsf{Dec}(\mathsf{sk}_{\mathsf{id}}, \mathsf{CT})$*: Given a ciphertext* $\mathsf{CT} = (\mathsf{CT}_1, \mathsf{CRS}, h, \mathsf{CT}_2)$ *and a secret key* $\mathsf{sk}_{\mathsf{id}}$ *as input, the algorithm does the following steps:*

1. *Run* $\Pi_{\mathsf{IB\text{-}wHPS}}.\mathsf{Decap}$ *to generate* $\boldsymbol{k}' = \Pi_{\mathsf{IB\text{-}wHPS}}.\mathsf{Decap}(\mathsf{sk}_{\mathsf{id}}, \mathsf{CT}_1)$*;*
2. *Output* $m' = \Pi_{\mathsf{PKE}}.\mathsf{Dec}(\mathsf{CRS}, \boldsymbol{k}', \mathsf{CT}_2)$.

Before presenting the theorem statement about the above Construction 10.2, we first define a specific function class that is convenient for us to argue KDM security for IBE. In particular, for IBE with the additional structure on identity secret key in Remark 4.2, i.e., $\mathsf{sk}_{\mathsf{id}} := (\boldsymbol{x}, \mathsf{sk}_{\boldsymbol{x},\mathsf{id}})$, we define KDM-IBE function class $\hat{\mathcal{F}}' = (\mathcal{F}_{s'}||\hat{\mathcal{Q}}^{\tau'})$ in the following way.

Definition 10.3 *For any integer* $\bar{n}$*,* $\mathcal{F}_{s'}$ *is the class of functions of all* $\bar{n}$ *identity secret keys* $\mathsf{sk}_{\mathsf{id}_i} := (\boldsymbol{x}_i, \mathsf{sk}_{\boldsymbol{x}_i,\mathsf{id}_i})$ *with* $i \in [\bar{n}]$*, and the circuit size of each function in* $\mathcal{F}_{s'}$ *is up to* s'*.* $\hat{\mathcal{Q}}^{\tau'}$ *is the class* $\{\mathfrak{g}' : \mathbb{Z}_B^n \to \mathcal{M}^{\tau' \times m}\}$ *of block-affine functions of all vectors* $\boldsymbol{x}_i$ *with* $i \in [\bar{n}]$*, which is defined similarly as in Definition 2.16. Moreover,* $(\mathcal{F}_{s'}||\hat{\mathcal{Q}}^{\tau'})$ *denotes the concatenation of two classes, i.e., every function* $\mathfrak{f}$ *in the class can be represented by* $\mathfrak{f} = (\mathfrak{h}, \mathfrak{q})$ *for some* $\mathfrak{h} \in \mathcal{F}_s$ *and* $\mathfrak{q} \in \hat{\mathcal{Q}}^{\tau'}$ *such that* $\mathfrak{f}(\mathsf{sk}) = (\mathfrak{h}(\mathsf{sk})||\mathfrak{q}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{\bar{n}}))$*, where* $\mathsf{sk} := (\mathsf{sk}_{\mathsf{id}_1}, \ldots, \mathsf{sk}_{\mathsf{id}_{\bar{n}}})$.

Next we prove the following theorem, and present its proof.

Theorem 10.4 *Assume that (1)* $\Pi_{\mathsf{IB\text{-}wHPS}}$ *is an universal* IB-wHPS *against adaptive (or selective) adversaries, (2)* $\Pi_{\mathsf{IB\text{-}wHPS}}$ *has the additional structure as Definition 4.1, (3)* $\Pi_{\mathsf{PKE}} = \Pi_{\mathsf{PKE}}.\{\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}\}$ *is on-the-fly* $\mathsf{KDM}^{(\bar{n},Q)}$*-secure for any polynomial* $\bar{n}, Q$ *with respect to the class* $\hat{\mathcal{F}} = (\mathcal{F}_s||\mathcal{Q}^{\tau})$ *in Definition 9.4, and (4)* Π_{PKE} *is leakage resilient to* ℓ*-bit block leakage with* m *blocks. Then the above scheme* Π_{IBE} *is adaptive (or selective) secure and*

1. *an leakage-resilient* IBE *against* ℓ*-bit (block) leakage; the (block) leakage rate is* $\frac{m \cdot \ell}{|\mathsf{sk}_{\mathsf{id}}|} = \frac{m \cdot \ell}{|\boldsymbol{a}|+|\mathsf{sk}_{\boldsymbol{a}}|}$*, (according to the additional structure.) The rate can be optimal, i.e.* $1 - o(1)$*, if* $\ell = (1 - o(1))\frac{|\boldsymbol{a}|}{m}$ *and* $|\mathsf{sk}_{\boldsymbol{a}}| = o(|\boldsymbol{a}|)$.
2. $\mathsf{KDM}^{(\bar{n})}$*-secure up to* Q *queries with respect to the class* $\hat{\mathcal{F}}' = (\mathcal{F}_{s'}||\hat{\mathcal{Q}}^{\tau'})$ *in Definition 10.3, where* $s' < s$ *and* $t' = t$ *such that* $s - s'$ *is the size of the algorithm* $\Pi_{\mathsf{IBE}}.\mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$ *with known* msk *and* id*, but unknown* $\boldsymbol{x}$*.*
3. *The information rate is* $\frac{|\mathcal{M}|}{|\mathsf{CT}_1|+|\mathsf{CT}_2|+|\mathsf{CRS}|+|h|} = \frac{\mathsf{rate}_{\mathsf{PKE}}}{(|\mathsf{CT}_1|+|\mathsf{CRS}|+|h|)/|\mathsf{CT}_2|+1}$*, where* $|\cdot|$ *denotes the bit-length,* $\mathsf{rate}_{\mathsf{PKE}}$ *denotes the information rate of the underlying* PKE*. As a result, for the underlying* PKE *with optimal information rate and large enough ciphertext size* $|\mathsf{CT}_2|$*, we obtain rate-1* KDM*-secure* IBE *scheme.*

Proof. Overall, the whole proof consists of two parts: leakage resilience and KDM security.

Proof of Leakage-Resilience of Construction 10.2

First we analyze the leakage-resilience by the following lemma.

Lemma 10.5 *Assume* $\Pi_{\mathsf{IB\text{-}wHPS}}$ *is an unversal* IB-wHPS*, and* PKE *is leakage-resilient against* ℓ*-bit block leakage. Then the scheme* Π_{IBE} *in Construction 10.2 is leakage-resilient against* ℓ*-bit block leakage.*

Proof. According to the definition of $\Pi_{\mathsf{IBE}}.\mathsf{Enc}(\mathsf{mpk}, \mathsf{id}, \mu)$, we know

$$\begin{aligned}\mathsf{CT} &= (\mathsf{CT}_1, \mathsf{CRS}, h, \mathsf{CT}_2)\\ &= (\mathsf{CT}_1, \mathsf{CRS}, \mathfrak{h}(\mathsf{CRS}, \boldsymbol{k}), \Pi_{\mathsf{PKE}}.\mathsf{Enc}(\mathsf{CRS}, \mathfrak{h}(\mathsf{CRS}, \boldsymbol{k}), \mu))\\ &\approx_c (\mathsf{CT}_1^*, \mathsf{CRS}, \mathfrak{h}(\mathsf{CRS}, \mathsf{Decap}(\boldsymbol{x}, \mathsf{CT}_1^*)), \Pi_{\mathsf{PKE}}.\mathsf{Enc}(\mathsf{CRS}, \mathfrak{h}(\mathsf{CRS}, \mathsf{Decap}(\boldsymbol{x}, \mathsf{CT}_1^*)), \mu)).\end{aligned}$$

Clearly, the ℓ bits block leakage of $\boldsymbol{x}$ will at most result in the ℓ bits block leakage of $\mathsf{Decap}(\boldsymbol{x}, \mathsf{CT}_1^*) = \boldsymbol{x} + \boldsymbol{k}'$, which happens to be the secret key of Π_{PKE}. Thus, the leakage resilience of Π_{IBE} follows naturally from that of Π_{PKE}. As the detailed proof is very similar to that of Section 5.1, we just omit the hybrids. □

It is clear that the allowed block leakage rate of Π_{IBE} is $\frac{m\ell}{|\mathsf{sk}_{\mathsf{id}}|}$ if $|\mathsf{sk}_{\mathsf{id}}| > m\ell$, where m is the number of blocks of secret key. Furthermore, according to the additional structure of IB-wHPS, we have $\mathsf{sk}_{\mathsf{id}} := (\boldsymbol{a}, \mathsf{sk}_{\boldsymbol{a},\mathsf{id}}) \in (\mathbb{Z}_B^n, \{0,1\}^*)$ and $\boldsymbol{k} \in \mathbb{Z}_B^n$ for certain parameters $n, B \in \mathbb{N}$. In order to achieve the optimal block leakage-resilience rate, we need to (1) enlarge the leakage size ℓ to be optimal, and (2) compress the bit-length of $\mathsf{sk}_{\mathsf{id}}$ such that $|\mathsf{sk}_{\mathsf{id}}| \approx m\ell$, which means that $\ell = (1 - o(1))\frac{|\boldsymbol{a}|}{m}$ and $\mathsf{sk}_{\boldsymbol{a},\mathsf{id}} = o(|\boldsymbol{a}|)$. It seems that all existing IB-wHPS with additional structure, including the generic IB-wHPS constructed from IBE by Hazay et al. [28], can not satisfy this requirement for the bit-length of identity secret key and encapsulated-key.

To overcome this technical dilemma, we introduce a new notion IB-ABE, which can be used to construct an IB-wHPS. Essentially, The intuitive idea of IB-ABE is that of compressing $|\mathsf{sk}_{\boldsymbol{a},\mathsf{id}}|$ by means of attribute-based idea.

Proof of $\mathsf{KDM}^{(\bar{n})}$-security of Construction 10.2

Below, we prove the $\mathsf{KDM}^{(\bar{n})}$-security with respect to $\hat{\mathcal{F}}' = (\mathcal{F}_{s'}||\hat{\mathcal{Q}}^{\tau'})$ in Definition 10.3 by a sequence of hybrid experiments. For each experiment i, we use W_i to denote the event that the adversary $\mathcal{A}$ wins the experiment H_i.

Hybrid H_0: This hybrid is defined to be the KDM-security experiment for IBE in Definition 2.11. In this hybrid, $\mathcal{A}$ can adaptively conduct three types of queries, including extraction queries, registration queries and KDM queries. For convenience, we assume that the challenge identities registered by $\mathcal{A}$ is $\{\mathsf{id}_1, \ldots, \mathsf{id}_{\bar{n}}\}$, and the number of KDM queries is Q for any polynomial $Q = \mathsf{poly}(\lambda)$. In this case, for any KDM-query in the form of $(\mathsf{id}_i, \mathfrak{f}) \in L_{\mathsf{ch}} \times \hat{\mathcal{F}}'$, the ciphertext returned from the challenger $\mathcal{C}$ with a random bit $b \in \{0,1\}$ should be

$$\begin{aligned}\mathsf{CT} &= (\mathsf{CT}_1, \mathsf{CRS}, h, \mathsf{CT}_2)\\ &= (\mathsf{CT}_1, \mathsf{CRS}, \mathsf{H}(\mathsf{CRS}, \boldsymbol{k}), \Pi_{\mathsf{PKE}}.\mathsf{Enc}(\mathsf{CRS}, \mathsf{H}(\mathsf{CRS}, \boldsymbol{k}), \mu_b)),\end{aligned}$$

where $\mu_0 = \mathfrak{f}(\mathsf{sk}_{\mathsf{id}_1}, \ldots, \mathsf{sk}_{\mathsf{id}_1})$, $\mu_1 = 0 \in \mathcal{M}$, $(\mathsf{CT}_1, \boldsymbol{k}) \xleftarrow{\$} \Pi_{\mathsf{IB\text{-}wHPS}}.\mathsf{Encap}(\mathsf{mpk}, \mathsf{id}_i)$ and CRS is chosen randomly and freshly.

In this H_0, W_0 indicates that $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$ such that $b = b'$. As a result, we have

$$\Pr[W_0] = \mathsf{Adv}^{\mathsf{KDM}}_{\mathsf{IBE},\hat{\mathcal{F}}',\mathcal{A}}(\lambda).$$

Hybrid $\mathsf{H_1}$: This experiment is identical to the H_0, except that all KDM ciphertexts are computed as

$$\begin{aligned}\mathsf{CT} &= (\mathsf{CT}_1, \mathsf{CRS}, h, \mathsf{CT}_2) \\ &= (\mathsf{CT}_1, \mathsf{CRS}, \mathsf{H}(\mathsf{CRS}, \boldsymbol{k}_1), \mathit{\Pi}_{\mathsf{PKE}}.\mathsf{Enc}(\mathsf{CRS}, \mathsf{H}(\mathsf{CRS}, \boldsymbol{k}_1), \mu_b)),\end{aligned}$$

where $(\mathsf{CT}_1, \boldsymbol{k}) \xleftarrow{\$} \mathit{\Pi}_{\mathsf{IB\text{-}wHPS}}.\mathsf{Encap}(\mathsf{mpk}, \mathsf{id}_i)$ and $\boldsymbol{k}_1 = \mathit{\Pi}_{\mathsf{IB\text{-}wHPS}}.\mathsf{Decap}(\mathsf{sk}_{\mathsf{id}_i}, \mathsf{CT}_1)$.

Hybrid $\mathsf{H_2}$: This experiment is identical to the H_1, except that all KDM ciphertexts are computed as

$$\begin{aligned}\mathsf{CT} &= (\mathsf{CT}_1^*, \mathsf{CRS}, h, \mathsf{CT}_2) \\ &= (\mathsf{CT}_1^*, \mathsf{CRS}, \mathsf{H}(\mathsf{CRS}, \boldsymbol{k}_1), \mathit{\Pi}_{\mathsf{PKE}}.\mathsf{Enc}(\mathsf{CRS}, \mathsf{H}(\mathsf{CRS}, \boldsymbol{k}_1), \mu_b)),\end{aligned}$$

where $\mathsf{CT}_1^* \xleftarrow{\$} \mathit{\Pi}_{\mathsf{IB\text{-}wHPS}}.\mathsf{Encap}^*(\mathsf{mpk}, \mathsf{id}_i)$ and $\boldsymbol{k}_1 = \mathit{\Pi}_{\mathsf{IB\text{-}wHPS}}.\mathsf{Decap}(\mathsf{sk}_{\mathsf{id}_i}, \mathsf{CT}_1^*)$.

Hybrid $\mathsf{H_3}$: This experiment is identical to the H_2, except that all KDM ciphertexts are computed as

$$\begin{aligned}\mathsf{CT} &= (\mathsf{CT}_1^*, \mathsf{CRS}, h, \mathsf{CT}_2) \\ &= (\mathsf{CT}_1^*, \mathsf{CRS}, \mathsf{H}(\mathsf{CRS}, \boldsymbol{k}_1), \mathit{\Pi}_{\mathsf{PKE}}.\mathsf{Enc}(\mathsf{CRS}, \mathsf{H}(\mathsf{CRS}, \boldsymbol{k}_1), \mu_b)),\end{aligned}$$

where (1) $\mathsf{CT}_1^* \xleftarrow{\$} \mathit{\Pi}_{\mathsf{IB\text{-}wHPS}}.\mathsf{Encap}^*(\mathsf{mpk}, \mathsf{id}_i)$ and $\boldsymbol{k}_1 = \boldsymbol{k}' + \boldsymbol{x}_i$, and $\boldsymbol{k}' \in \mathbb{Z}_B^n$ is a vector used to generate the invalid ciphertext CT_1^*, and (2) $\mathsf{sk}_{\mathsf{id}_i} := (\boldsymbol{x}_i, \mathsf{sk}_{\boldsymbol{x}_i, \mathsf{id}_i}) \in \mathbb{Z}_B^n \times \{0,1\}^*$ according to the additional structure of $\mathit{\Pi}_{\mathsf{IB\text{-}wHPS}}$.

Next, we use the following claims to prove the indistinguishability among the above games.

Claim 10.6 *The distributions of* H_0 *and* H_1 *are identical.*

Proof. This follows clearly from the correctness of $\mathit{\Pi}_{\mathsf{IB\text{-}wHPS}}$. □

Claim 10.7 *The distributions of* H_1 *and* H_2 *are computationally indistinguishability.*

Proof. This follows from the valid/invalid ciphertext indistinguishability of $\mathit{\Pi}_{\mathsf{IB\text{-}wHPS}}$. □

Claim 10.8 *The distributions of* H_2 *and* H_3 *are identical.*

Proof. This follows clearly from the additional structure of $\mathit{\Pi}_{\mathsf{IB\text{-}wHPS}}$. □

Summing up the above three claims, we conclude that $|\Pr[W_0] - \Pr[W_3]| \leq \mathsf{negl}(\lambda)$ for any efficient adversary $\mathcal{A}$. To complete the proof, it suffices to show that the probability $\Pr[W_3]$ is negligible, i.e. the advantage of an efficient adversary in H_3.

Next, we analyze the KDM-security of hybrid H_3 in details.

Claim 10.9 *Suppose the* PKE *scheme* Π_{PKE} *is on-the-fly* $\mathsf{KDM}^{(\bar{n},Q)}$*-secure with respect to* $\hat{\mathcal{F}} = (\mathcal{F}_s || \mathcal{Q}^\tau)$ *in Definition 9.4, then for any efficient adversary* $\mathcal{A}$ *who makes* KDM *queries for* Q *times in the* $\mathsf{KDM}^{(\bar{n})}$ *experiment with respect to* $\hat{\mathcal{F}}' = (\mathcal{F}_{s'} || \hat{\mathcal{Q}}^{\tau'})$ *in Definition 10.3 according to* H_3*, it holds* $\Pr[W_3] \leq \mathsf{negl}(\lambda)$*, where* $s' < s$ *and* $\tau' = \tau$ *such that* $s - s'$ *is the size of the algorithm* $\Pi_{\mathsf{IBE}}.\mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$ *with known* msk *and* id*, but unknown* $\boldsymbol{x}$*.*

Proof. Similar to the proof of Theorem 6.4, we prove this claim by establishing a security reduction between the $\mathsf{KDM}^{(\bar{n})}$-security of Π_{IBE} in H_3 and the on-the-fly KDM-security of Π_{PKE}. More specifically, suppose there is an efficient adversary $\mathcal{A}$ wining H_3 with success probability ε (within Q times KDM queries), then we construct an efficient reduction algorithm $\mathcal{R}$ that breaks the on-the-fly $\mathsf{KDM}^{(\bar{n},Q)}$-security of Π_{PKE} with the same probability ε.

Let $\mathcal{C}$ be the challenger in the on-the-fly $\mathsf{KDM}^{(\bar{n},Q)}$-security experiment of Π_{PKE}. We describe the detailed reduction algorithm below.

- Setup.
  1. $\mathcal{C}$ runs the algorithm $\Pi_{\mathsf{PKE}}.\mathsf{KeyGen}$ $\bar{n}$ times to generate $\mathsf{sk}_i := \boldsymbol{k}_i \in \mathcal{K}$, and selects a random bit $b \in \{0,1\}$.
  2. $\mathcal{R}$ runs the algorithm $\Pi_{\mathsf{IBE}}.\mathsf{Setup}$ to generate $(\mathsf{mpk}, \mathsf{msk})$, and sends mpk to the adversary $\mathcal{A}$. At the same time, $\mathcal{R}$ prepares lists $L_{\mathsf{ext}}, L_{\mathsf{ch}}$, all of which are initially empty.
- Query Stage. $\mathcal{A}$ may adaptively make the following three types of queries polynomially many times.
  1. **Extraction queries**. $\mathcal{A}$ sends $\mathsf{id} \in \mathcal{ID} \backslash (L_{\mathsf{ext}} \cup L_{\mathsf{ch}})$ to $\mathcal{R}$. As a response, $\mathcal{R}$ first chooses a random vector $\boldsymbol{k}_{\mathsf{id}}$, then generates $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id}, \boldsymbol{k}_{\mathsf{id}})$ and sends it to $\mathcal{A}$. Finally, $\mathcal{R}$ adds id to L_{ext}.
  2. **Registration queries**. $\mathcal{A}$ sends $\mathsf{id} \in \mathcal{ID} \backslash (L_{\mathsf{ext}} \cup L_{\mathsf{ch}})$ to $\mathcal{R}$. Then, $\mathcal{R}$ adds id to L_{ch} and relates it to one of the unknown vectors $\{\boldsymbol{k}_i\}_{i \in [\bar{n}]}$ generated secretly by $\mathcal{C}$. E.g., $\mathcal{R}$ can keep a counter t, relate a new id to the next unknown secret vector $\boldsymbol{k}_t$, and then increase the counter t.
  3. **KDM queries**. $\mathcal{A}$ sends $(\mathsf{id}, \mathfrak{f}) \in L_{\mathsf{ch}} \times \hat{\mathcal{F}}'$ to $\mathcal{R}$. With the KDM query $(\mathsf{id}, \mathfrak{f}) \in L_{\mathsf{ch}} \times \hat{\mathcal{F}}'$ from $\mathcal{A}$, $\mathcal{R}$ first transforms the function $\mathfrak{f} : \mathcal{SK}^{\bar{n}} \to \mathcal{M}$ into a new function $\mathfrak{g}_{\mathsf{msk}} \in \hat{\mathcal{F}}$ mapping $\mathcal{K}^{\bar{n}}$ to $\mathcal{M}$ with the following information hard-coded: (1) the master secret key msk, (2) the algorithm $\Pi_{\mathsf{IBE}}.\mathsf{KeyGen}$, and (3) the one-to-one correspondence between $\mathsf{id}_i \in L_{\mathsf{ch}}$ and the secret vectors $\boldsymbol{k}_i$. Particularly, $\mathfrak{g}_{\mathsf{msk}}(\boldsymbol{k}_1, \ldots, \boldsymbol{k}_n)$ does the following:
     - It first samples $\mathsf{sk}_{\boldsymbol{k}_i, \mathsf{id}_i} \leftarrow \mathsf{IB\text{-}wHPS.KeyGen}(\mathsf{msk}, \mathsf{id}_i, \boldsymbol{k}_i)$ (according to the additional structure as Remark 4.2) for $i \in [\bar{n}]$, where id_i corresponds to the secret vector $\boldsymbol{k}_i$.
     - Then it defines $\mathsf{sk}_{\mathsf{id}_i} = (\boldsymbol{k}_i, \mathsf{sk}_{\boldsymbol{k}_i, \mathsf{id}_i})$ for $i \in [\bar{n}]$, and outputs $\mathfrak{f}(\mathsf{sk}_1, \ldots, \mathsf{sk}_{\bar{n}})$.

     Then, $\mathcal{R}$ sends $(i, \mathfrak{g}_{\mathsf{msk}}) \in [\bar{n}] \times \hat{\mathcal{F}}$ to $\mathcal{C}$ as a KDM query for the scheme Π_{PKE}.
- KDM Responses.

1. After receiving $(i, \mathfrak{g}_{\mathsf{msk}}) \in [\bar{n}] \times \hat{\mathcal{F}}$ from $\mathcal{R}$, $\mathcal{C}$ computes

$$\mathsf{CT}_0 = \Pi_{\mathsf{PKE}}.\mathsf{Enc}(\mathsf{pk}, \mathfrak{g}_{\mathsf{msk}}(\boldsymbol{k}_1, \ldots, \boldsymbol{k}_{\bar{n}})) = (\mathsf{CRS}, h = \mathsf{H}(\mathsf{CRS}, \boldsymbol{k}_i), \mathsf{CT}'_0)$$

if $b = 0$, otherwise computes

$$\mathsf{CT}_1 = \Pi_{\mathsf{PKE}}.\mathsf{Enc}(\mathsf{pk}, 0) = (\mathsf{CRS}, h = \mathsf{H}(\mathsf{CRS}, \boldsymbol{k}_i), \mathsf{CT}'_1),$$

for $0 \in \mathcal{M}$. Then, $\mathcal{C}$ responds CT_b to $\mathcal{R}$.

2. After receiving $\mathsf{CT}_b = (\mathsf{CRS}, h, \mathsf{CT}'_b)$ from $\mathcal{C}$, in order to respond the KDM query $(\mathsf{id}, \mathfrak{f})$ from $\mathcal{A}$, $\mathcal{R}$ first chooses a random vector $\boldsymbol{k}' \xleftarrow{\$} \mathbb{Z}_B^n$ to generate an invalid ciphertext $\mathsf{CT}^*_{\mathsf{id}}$, and then uses both transformations $\mathcal{T}_1$ and $\mathcal{T}_2$ on CT_b to generate and send

$$\mathsf{CT} = (\mathsf{CT}^*_{\mathsf{id}}, \mathsf{CRS}', h', \mathsf{CT}''_b)$$

to $\mathcal{A}$, where $(\mathsf{CRS}', h') = \mathcal{T}_1(\mathsf{CRS}, h, \boldsymbol{k}')$ and $\mathsf{CT}''_b = \mathcal{T}_2(\mathsf{CT}'_b, \boldsymbol{k}')$.

– Output Stage.
1. $\mathcal{A}$ outputs a bit $b' \in \{0, 1\}$ and sends it to $\mathcal{R}$.
2. $\mathcal{R}$ outputs the same b' as the guess for b.

We notice that the following facts about the above reduction process:

1. As $\mathcal{R}$ generates the master secret key msk for IBE scheme Π_{IBE} by himself, the extraction queries from $\mathcal{A}$ can be answered perfectly by $\mathcal{R}$.
2. For $i \in [\bar{n}]$, we have $\mathsf{sk}_{\mathsf{id}_i} \leftarrow \Pi_{\mathsf{IBE}}.\mathsf{KeyGen}(\mathsf{msk}, id_i, \boldsymbol{k}_i)$ according to the additional structure of the used $\mathsf{IB\text{-}wHPS}$. In this case, with the help of msk, the KDM query function $\mathfrak{f} \in \hat{\mathcal{F}}'$, i.e., $\mathfrak{f}(\mathsf{sk}_{\mathsf{id}_1}, \ldots, \mathsf{sk}_{\mathsf{id}_{\bar{n}}})$, can be viewed as the function of unknown $\boldsymbol{k}_1, \ldots, \boldsymbol{k}_{\bar{n}}$. More formally, we have

$$\mathfrak{f}(\mathsf{sk}_{\mathsf{id}_1}, \ldots, \mathsf{sk}_{\mathsf{id}_{\bar{n}}}) = \mathfrak{g}_{\mathsf{msk}}(\boldsymbol{k}_1, \ldots, \boldsymbol{k}_{\bar{n}}).$$

Thus, $\mathcal{R}$ can use $\mathfrak{g}_{\mathsf{msk}} \in \hat{\mathcal{F}} = (\mathcal{F}_s || \mathcal{Q}^\tau)$ to simulate $\mathfrak{f} \in \hat{\mathcal{F}}' = (\mathcal{F}_{s'} || \hat{\mathcal{Q}}^{\tau'})$ faithfully, where $s' < s$ and $\tau' = \tau$ such that $s - s'$ is the size of the algorithm $\Pi_{\mathsf{IBE}}.\mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$ with known msk and id, but unknown $\boldsymbol{x}$.

3. For the KDM query on certain challenge identity id, which is in the form of $(\mathsf{id}, \mathfrak{f}) \in L_{\mathsf{ch}} \times \hat{\mathcal{F}}'$, $\mathcal{R}$ returns the ciphertext $\mathsf{CT} = (\mathsf{CT}^*_{\mathsf{id}}, \mathsf{CRS}', h', \mathsf{CT}''_b)$ to $\mathcal{A}$. According to the encryption algorithm $\Pi_{\mathsf{PKE}}.\mathsf{Enc}$ and the properties of $\mathcal{T}_1$ and $\mathcal{T}_2$, this ciphertext CT is statistically identical to the ciphertext in H_3.

From all above analyses, we can conclude that the above simulation process is perfect, no matter $b = 0$ or 1. Therefore, the advantage of $\mathcal{R}$ in breaking the on-the-fly KDM security is ε, the same as that of $\mathcal{A}$ in breaking the IBE KDM security. This completes the proof of this claim. □

Combining Claims 10.6-10.9, we complete the proof of $\mathsf{KDM}^{(\bar{n})}$-security of Construction 10.2. □

Notice that one of interesting aspects of Theorem 10.4 is that the resulting $\mathsf{KDM\text{-}IBE}$ function $\hat{\mathcal{F}}' = (\mathcal{F}_{s'} || \hat{\mathcal{Q}}^{\tau'})$ is large enough to encode all bounded size functions of the secret keys, just as in Theorem 9.5. This means that our KDM results on IBE is essentially complete, i.e., we can further apply the approach of randomized encoding to amplify the class of $\mathsf{KDM\text{-}IBE}$ functions to circuits of any bounded size in a black box way.

11 Instantiations of Construction 10.2

In this section, we show how to instantiate the building blocks for Construction 10.2.

11.1 Instantiations of the Required Ingredients in Section 10

In this section, we show how to instantiate the required ingredients in Section 10 to achieve an IBE that is both leakage resilient and $\mathsf{KDM}^{(\bar{n})}$-secure for any unbounded polynomial $\bar{n}$. Particularly, we need the following two tools: (1) a PKE Π that is both leakage resilient up to ℓ-bit leakage and on-the-fly KDM-secure with respect to some class $\mathcal{G}$ (discussed later), and (2) an IB-wHPS with the additional structure as Definition 4.1 and Remark 4.2. In Section 11.1, we show how to instantiate the former, and in Section 11.1, we show how to instantiate the latter.

Instantiations of PKE with On-the-fly KDM-security

To instantiate such a PKE, we prove that Construction 5.3 can achieve leakage resilience and as well this notion of on-the-fly KDM security. Clearly by Theorem 5.4 (particularly security proof in Section 5.1) with the computational reusable extractor in Section 3.2, Construction 5.3 is leakage resilient with any (block-source) leakage function with bounded output length. We next focus on proving the on-the-fly KDM-security. Particularly, we first show that for any BE-based PKE Π, if

1. Π satisfies the two key-homomorphic properties (ref. Section 6.1),
2. $\Pi^{\bar{n}Q}$ is $\mathsf{KDM}^{(1)}$-secure,

then Π is on-the-fly-$\mathsf{KDM}^{(\bar{n})}$-secure up to Q KDM queries. More formally, we state the following theorem:

Theorem 11.1 (On-the-fly $\mathsf{KDM}^{(\bar{n})}$-security of Π) *Suppose that (1) the* BE*-based* PKE *scheme Π satisfies Properties 1 and 2 in Section 6.1, and (2) the intermediate scheme $\Pi^{\bar{n}Q}$ in Definition 6.3 is $\mathsf{KDM}^{(1)}$-security with respect to the class $\mathcal{G} = \{\mathfrak{g} : \mathcal{SK} \to \mathcal{M}\}$. Then Π is on-the-fly $\mathsf{KDM}^{(\bar{n})}$-secure up to Q* KDM *queries, with respect to the class $\mathcal{F} = \{\mathfrak{f} : \mathcal{SK}^{\bar{n}} \to \mathcal{M}\}$ that satisfies the relation as stated in Remark 6.5.*

The proof is essentially the same as that of Theorem 6.4, so we omit the details.

Next, as stated in Section 6, we have proved that the scheme $\Pi^{\bar{n}}_{\mathsf{PKE}}$ (derived from Π_{PKE} in Construction 5.3) is $\mathsf{KDM}^{(1)}$-secure for unbounded polynomial $\bar{n}$ (when the underlying BE is instantiated properly, e.g., from the DDH in Section 7.1, or LWE in Section 7.2). Then by using Applebaum's randomized encoding technique [7], we can obtain $\Pi'^{\bar{n}}_{\mathsf{PKE}}$ that is $\mathsf{KDM}^{(1)}$-secure with respect to $\mathcal{G} = \{\text{All bounded polynomial sized circuits}\}$. Since Applebaum's randomized

encoding technique does not change the public key and secret key, and it just uses the underlying encryption algorithm in a black-box way, the scheme $\Pi'^{\bar{n}}_{\mathsf{PKE}}$ also satisfies the two key-homomorphic properties in Section 6.1 if the original one (i.e., $\Pi^{\bar{n}}_{\mathsf{PKE}}$) does.

Combining the above arguments and Theorem 11.1, we conclude the following corollary:

Corollary 11.2 *Under the* DDH*/*LWE *assumptions, there exists a* PKE *that is both leakage resilient with leakage rate* $1 - o(1)$*, and on-the-fly* $\mathsf{KDM}^{(\bar{n})}$*-secure for any polynomial* $\bar{n}$ *with respect to any bounded size circuits.*

The parameters of LWE are referred to Constructions 7.3.

Instantiations of the Required IB-wHPS

In this section, we show how to instantiate the required IB-wHPS for Section 10.2. Particularly, we need an IB-wHPS with the additional structure as Definition 4.1 and Remark 4.2, i.e., the secret key has the form $\mathsf{sk}_{\mathsf{id}} = (\boldsymbol{a}, \mathsf{sk}_{\boldsymbol{a}})$, and $\mathsf{Decap}(\mathsf{sk}, \mathsf{CT}^*) = \boldsymbol{k} + \boldsymbol{a}$ for an invalid ciphertext CT^*. As shown in Section 10.2, any IB-wHPS with such a structure can be used to achieve IBE schemes that are both KDM-secure and leakage resilient with the leakage rate $\frac{\ell}{|\boldsymbol{a}|+|\mathsf{sk}_{\boldsymbol{a}}|}$, where ℓ is the number of leakage bits that the Π_{PKE} can tolerate. By Corollary 11.2 and Construction 10.2, we know that $\ell = (1 - o(1))|\boldsymbol{a}|$, implying the leakage rate $\frac{1-o(1)}{1+|\mathsf{sk}_{\boldsymbol{a}}|/|\boldsymbol{a}|}$.

Below, we show several instantiations with different $|\mathsf{sk}_{\boldsymbol{a}}|/|\boldsymbol{a}|$. First, Hazay et al. [28] showed an elegant construction of IB-wHPS with the additional structure from any IBE[16]. Their construction yields $|\mathsf{sk}_{\boldsymbol{a}}|/|\boldsymbol{a}| \approx \frac{s(\lambda)|\boldsymbol{a}|}{|\boldsymbol{a}|} = s(\lambda)$[17], where $s(\lambda)$ is the size of the secret key in the underlying IBE. In particular, the work [28] proved the following theorem:

Theorem 11.3 ([28]) *Suppose there exists an adaptive (or selective)* IBE *scheme with the secret-key size* $s = s(\lambda)$*, then there exists an adaptive (or selective)* IB-wHPS *with key space* $\mathcal{K} = \mathbb{Z}_m^n$ *and additional structure. Furthermore, for* $\mathsf{sk}_{\mathsf{id}} := (\boldsymbol{a}, \mathsf{sk}_{\boldsymbol{a}})$*, it holds* $|\mathsf{sk}_{\boldsymbol{a}}|/|\boldsymbol{a}| = \frac{s(\lambda)n}{|\boldsymbol{a}|}$.

By using this general construction [28], our IBE Construction 10.2 would achieve leakage rate $\frac{1-o(1)}{1+s(\lambda)}$. Next, we describe better instantiations of the required IB-wHPS, improving $|\mathsf{sk}_{\boldsymbol{a}}|/|\boldsymbol{a}|$ significantly. In particular, this ratio can achieve $o(1)$ from LWE, and $o(1)$ from certain assumptions in bilinear groups. Thus, our IBE Construction 10.2 would achieve leakage rate $1-o(1)$ from LWE and $1-o(1)$ from these assumptions in bilinear groups.

[16] Even though the work [28] did not consider the additional structure, their construction clearly satisfies this structure.

[17] This rate can be improved to $s(\lambda)/\log\lambda$.

Better Instantiations of IB-wHPS

In this section, we present better instantiations of IB-wHPS, improving the ratio $|\mathsf{sk}_{\boldsymbol{a}}|/|\boldsymbol{a}|$ significantly. We observe that any adaptive (resp. selective) ABE with succinct secret keys[18] suffices to derive an adaptive (resp. selective) IB-wHPS with the ratio $|\mathsf{sk}_{\boldsymbol{a}}|/|\boldsymbol{a}| = o(1)$, and thus achieves the optimal leakage rate. For lattice-based instantiations however, we are not aware of any adaptively secure ABE. To construct the desired IB-wHPS from lattices, we use a more fine-grained primitive IB-ABE [32].

Briefly, an IB-ABE consists of four algorithms IB-ABE.$\{\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}\}$ similar to IBE and ABE. Each secret key $\mathsf{sk}_{\mathsf{id},f}$ is associated with an id and a policy function f, and a ciphertext is associated with an id' and attribute x. The secret key $\mathsf{sk}_{\mathsf{id},f}$ can decrypt the ciphertext if and only if the IDs match and the attribute satisfies the policy function, i.e. $\mathsf{id} = \mathsf{id}'$ and $f(x) = 1$. We present the formal definition of syntax and security below.

Definition 11.4 (IB-ABE) *An identity-attribute-based encryption* (IB-ABE) *scheme for a class of functions $\mathcal{F}_\lambda = \{f : \mathcal{X}_\lambda \to \mathcal{Y}_\lambda\}$ consists of four algorithms* IB-ABE.$\{\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}\}$ *as follows.*

- **Setup.** IB-ABE.$\mathsf{Setup}(1^\lambda)$ *takes a security parameter λ as input, and generates a pair of master public key and master secret key* $(\mathsf{mpk}, \mathsf{msk})$*, where* mpk *will define the identity space $\mathcal{ID}$, attribute space $\mathcal{X}_\lambda$, policy function class $\mathcal{F}_\lambda$, message space $\mathcal{M}$ and ciphertext space $\mathcal{C}$.*
- **Key generation.** IB-ABE.KeyGen $(\mathsf{id}, f, \mathsf{msk})$ *takes as input an identity* $\mathsf{id} \in \mathcal{ID}$*, a function $f \in \mathcal{F}_\lambda$ and the master secret key* msk*, and generates a secret key* $\mathsf{sk}_{\mathsf{id},f}$*.*
- **Encryption.** IB-ABE.$\mathsf{Enc}(\mathsf{mpk}, \boldsymbol{x}, \mu)$ *takes as input the master public key* mpk*, an attribute $\boldsymbol{x} \in \mathcal{X}_\lambda$ and message $m \in \mathcal{M}$, and outputs a ciphertext $c \in \mathcal{C}$.*
- **Decryption.** IB-ABE.$\mathsf{Dec}(\mathsf{sk}_{\mathsf{id},f}, c)$ *takes as input a secret key* $\mathsf{sk}_{\mathsf{id},f}$ *and ciphertext c, and outputs $m \in \mathcal{M}$ if $f(\boldsymbol{x}) \neq 0$ or $\perp$ if $f(\boldsymbol{x}) = 0$, where $\boldsymbol{x}$ is the corresponding attribute used to generate c.*

Here, for an IB-ABE scheme, we consider its security notion with respect to adaptive identity and selective attribute.

Definition 11.5 (Partial-adaptive Security of IB-ABE [32]) *An* IB-ABE *scheme $\Pi =$* IB-ABE.$\{\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}\}$ *for a class of functions $\mathcal{F}_\lambda = \{f : \mathcal{X}_\lambda \to \{0,1\}\}$ is partial-adaptively secure, i.e., with respect to adaptive identities and selective attributes, if for all probabilistic polynomial-time adversaries $\mathcal{A}$, it holds*

$$\mathbf{Adv}^{\mathsf{IB\text{-}ABE}}_{\Pi,\mathcal{A}}(\lambda) = \left|\Pr\left[\mathbf{Exp}^{\mathsf{IB\text{-}ABE}}_{\Pi,\mathcal{A}}(1^\lambda)\right] - 1/2\right| \leq \mathsf{negl}(\lambda).$$

where for each $b \in \{0,1\}$ and $\lambda \in \mathbb{N}$, the experiment $\mathbf{Exp}^{\mathsf{IB\text{-}ABE}}_{\Pi,\mathcal{A}}(1^\lambda)$ is defined in Table 7.

[18] The secret key size only depends on the depth of the policy function when the function description is given.

Experiment $\mathbf{Exp}^{\mathsf{IB\text{-}ABE}}_{\Pi,\mathcal{A}}(1^\lambda)$
Pre-stage: The adversary $\mathcal{A}$ chooses an challenge attribute $\boldsymbol{x}^* \in \mathcal{X}_\lambda$ and sends it to the challenger $\mathcal{C}$.
Setup: The challenger $\mathcal{C}$ runs $\mathsf{IB\text{-}ABE.Setup}(1^\lambda)$ to obtain $(\mathsf{mpk}, \mathsf{msk})$, and sends mpk to $\mathcal{A}$.
Test Stage 1: $\mathcal{A}$ queries the challenger $\mathcal{C}$ with pairs $\{(\mathsf{id}, f)\}$ where each $(\mathsf{id}, f) \in \mathcal{ID} \times \mathcal{F}_\lambda$. Then $\mathcal{C}$ responds with $\mathsf{sk}_{\mathsf{id},f}$, and stores the tuple.
Challenge Stage: $\mathcal{A}$ chooses an arbitrary challenge identity $\mathsf{id}^* \in \mathcal{ID}$ and two messages μ_0, $\mu_1 \in \mathcal{M}$, and sends them to $\mathcal{C}$ for challenge query. • If there exists a pair (id^*, f) such that $f(\boldsymbol{x}^*) = 1$ that had been queried in Test Stage 1, $\mathcal{C}$ aborts the experiment. • Otherwise, $\mathcal{C}$ first selects $b \xleftarrow{\$} \{0,1\}$, then computes $c_b \xleftarrow{\$} \mathsf{IB\text{-}ABE.Enc}(\mathsf{mpk}, \mathsf{id}^*, \boldsymbol{x}^*, \mu_b)$. Finally, $\mathcal{C}$ returns c_b to $\mathcal{A}$.
Test Stage 2: $\mathcal{A}$ queries the challenger $\mathcal{C}$ with with pairs $\{(\mathsf{id}', f') \in \mathcal{ID} \times \mathcal{F}_\lambda\}$ under the constraint that $\mathsf{id}^* \in \{\mathsf{id}'\}$ and $f'(\boldsymbol{x}^*) = 1$ can not happen at the same time. Then $\mathcal{C}$ responds with $\mathsf{sk}_{\mathsf{id}',f'}$.
Output: The adversary $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$. If $b' = b$, the experiment outputs 1; and 0 otherwise.

Table 7. Security Experiment of IB-ABE

A secure IB-ABE with respect to selective identity and attribute can be defined similarly, except that both the challenge identity id^* and attribute x^* must be provided simultaneously to the challenger $\mathcal{C}$ prior to the setup step.

Next we present the general construction in [32], which derives an adaptive IB-wHPS from IB-ABE that is partial-adaptively secure, i.e., adaptive over ID and selective over attribute. Then we discuss several instantiations.

Construction 11.6 (Generic Construction of IB-wHPS from IB-ABE [32]) *Let $m = m(\lambda)$ and $n = n(\lambda)$ be two parameters and let $\Pi = \mathsf{IB\text{-}ABE.}\{\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}\}$ be an* IB-ABE *scheme with identity space $\mathcal{ID} = \{0,1\}^*$, attribute space $\mathcal{X} = [n] \times [m]$, message-space $\mathcal{M} = \mathbb{Z}_m$ and ciphertext space $\mathcal{C}$ for a class of functions $\mathcal{F}_\lambda : \{f : \mathcal{X}_\lambda \to \{0,1\}\}$, where all function $f_{\boldsymbol{y}} \in \mathcal{F}_\lambda$ is indexed by a vector $\boldsymbol{y} = (y_1, \ldots, y_n)^\top \in [m]^n$. More specifically, for any vector $\boldsymbol{x} = (x_1, x_2)^\top \in [n] \times [m]$, $f_{\boldsymbol{y}}(\boldsymbol{x}) = 1$ if and only if $x_2 = y_{x_1}$. Then, an* IB-wHPS *with output space $\mathcal{K} = \mathbb{Z}_m^n$ can be constructed in the following way:*

- $\mathsf{IB\text{-}wHPS.Setup}(1^\lambda)$*: Given the security parameter λ as input, the algorithm runs the algorithm* $\mathsf{IB\text{-}ABE.Setup}$ *to generate* $(\mathsf{mpk}^{\mathsf{IB\text{-}ABE}}, \mathsf{msk}^{\mathsf{IB\text{-}ABE}}) \xleftarrow{\$} \mathsf{IB\text{-}ABE.Setup}(1^\lambda)$*, and outputs* $\mathsf{mpk} := \mathsf{mpk}^{\mathsf{IB\text{-}ABE}}$ *and* $\mathsf{msk} := \mathsf{msk}^{\mathsf{IB\text{-}ABE}}$.
- $\mathsf{IB\text{-}wHPS.KeyGen}(\mathsf{msk}, \mathsf{id})$*: Given a master secret-key* msk *and an identity* $\mathsf{id} \in \mathcal{ID}$ *as input, the algorithm first chooses a function $f_{\boldsymbol{y}} \in \mathcal{F}$ indexed by a random vector $\boldsymbol{y} \xleftarrow{\$} [m]^n$, and then runs* $\mathsf{IB\text{-}ABE.KeyGen}$ *to generate* $\mathsf{sk}^{\mathsf{IB\text{-}ABE}}_{\mathsf{id},f_{\boldsymbol{y}}} \xleftarrow{\$} \mathsf{IB\text{-}ABE.KeyGen}(\mathsf{msk}, \mathsf{id}, f_{\boldsymbol{y}})$*. Then the algorithm outputs* $\mathsf{sk}_{\mathsf{id}} := (\boldsymbol{y}, \mathsf{sk}^{\mathsf{IB\text{-}ABE}}_{\mathsf{id},f_{\boldsymbol{y}}})$ *as the secret key for identity* id.

– $\mathsf{IB\text{-}wHPS.Encap}(\mathsf{mpk}, \mathsf{id})$*: Given a master public-key* mpk *and an identity* $\mathsf{id} \in \mathcal{ID}$ *as input, the algorithm samples a random vector* $\boldsymbol{k} = (k_1, \ldots, k_n)^\top \in \mathbb{Z}_m^n$*, and run the algorithm* $\mathsf{IB\text{-}ABE.Enc}$ *mn times with attributes* $\boldsymbol{x}_{i,j} = (i,j)^\top \in [n] \times [m]$ *to set*

$$\mathsf{c} := \{c_{i,j} \xleftarrow{\$} \mathsf{IB\text{-}ABE.Enc}(\mathsf{mpk}, \mathsf{id}, \boldsymbol{x}_{i,j}, k_i)\}_{(i,j)\in[n]\times[m]} \in \mathcal{C}^{n\times m}, \text{ i.e.,}$$

$$\mathsf{c} := \begin{bmatrix} \mathsf{IB\text{-}ABE.Enc}(\mathsf{mpk}, \mathsf{id}, \boldsymbol{x}_{1,1}, k_1) & \ldots & \mathsf{IB\text{-}ABE.Enc}(\mathsf{mpk}, \mathsf{id}, \boldsymbol{x}_{1,m}, k_1) \\ \vdots & \ddots & \vdots \\ \mathsf{IB\text{-}ABE.Enc}(\mathsf{mpk}, \mathsf{id}, \boldsymbol{x}_{n,1}, k_n) & \ldots & \mathsf{IB\text{-}ABE.Enc}(\mathsf{mpk}, \mathsf{id}, \boldsymbol{x}_{n,m}, k_n) \end{bmatrix}.$$

Finally, the algorithm outputs $(\mathsf{c}, \boldsymbol{k})$.

– $\mathsf{IB\text{-}wHPS.Encap}^*(\mathsf{mpk}, \mathsf{id})$*: Given a master public-key* mpk *and an identity* $\mathsf{id} \in \mathcal{ID}$ *as input, the algorithm samples a random vector* $\boldsymbol{k} = (k_1, \ldots, k_n)^\top \in \mathbb{Z}_m^n$*, and run the algorithm* $\mathsf{IB\text{-}ABE.Enc}$ *mn times with attributes* $\boldsymbol{x}_{i,j} = (i,j)^\top$ *to set*

$$\mathsf{c}^* := \{c^*_{i,j} \xleftarrow{\$} \mathsf{IB\text{-}ABE.Enc}(\mathsf{mpk}, \mathsf{id}, \boldsymbol{x}_{i,j}, k_i + j)\}_{(i,j)\in[n]\times[m]} \in \mathcal{C}^{n\times m}, \text{ i.e.,}$$

$$\mathsf{c}^* := \begin{bmatrix} \mathsf{IB\text{-}ABE.Enc}(\mathsf{mpk}, \mathsf{id}, \boldsymbol{x}_{1,1}, k_1 + 1) & \ldots & \mathsf{IB\text{-}ABE.Enc}(\mathsf{mpk}, \mathsf{id}, \boldsymbol{x}_{1,m}, k_1 + m) \\ \vdots & \ddots & \vdots \\ \mathsf{IB\text{-}ABE.Enc}(\mathsf{mpk}, \mathsf{id}, \boldsymbol{x}_{n,1}, k_n + 1) & \ldots & \mathsf{IB\text{-}ABE.Enc}(\mathsf{mpk}, \mathsf{id}, \boldsymbol{x}_{n,m}, k_n + m) \end{bmatrix},$$

where the addition $k_i + j$ *is performed over* $\mathbb{Z}_m$*. Finally, the algorithm outputs* c^*.

– $\mathsf{IB\text{-}wHPS.Decap}(\mathsf{sk}_{\mathsf{id}}, \mathsf{c})$*: Given a secret key* $\mathsf{sk}_{\mathsf{id}} := (\boldsymbol{y}, \mathsf{sk}^{\mathsf{IB\text{-}ABE}}_{\mathsf{id},f_{\boldsymbol{y}}})$ *and* $\mathsf{c} := \{c_{i,j}\}_{(i,j)\in[n]\times[m]}$ *as input, the algorithm runs* $\mathsf{IB\text{-}ABE.Dec}$ *to compute* $k_i = \mathsf{IB\text{-}ABE.Dec}(\mathsf{sk}^{\mathsf{IB\text{-}ABE}}_{\mathsf{id},f_{\boldsymbol{y}}}, c_{i,y_i})$ *for all* $i \in [n]$*, and then outputs* $\boldsymbol{k} = (k_1, \ldots, k_n)^\top$*, if* $f_{\boldsymbol{y}}(i, y_i) = 1$ *for all* $i \in [n]$*, and* $\perp$ *otherwise.*

Theorem 11.7 (IB-wHPS from IB-ABE [32]) *Suppose* $\Pi_{\mathsf{IB\text{-}ABE}}$ *is a secure* $\mathsf{IB\text{-}ABE}$ *scheme for the function class* $\mathcal{F} = \{f : [n] \times [m] \to \{0,1\}\}$*, then the construction* $\Pi_{\mathsf{IB\text{-}wHPS}}$ *described above is an* $\mathsf{IB\text{-}wHPS}$ *with the encapsulated-key-space* $\mathcal{K} = \mathbb{Z}_m^n$*. Furthermore,*

– *if the underlying* $\mathsf{IB\text{-}ABE}$ $\Pi_{\mathsf{IB\text{-}ABE}}$ *is secure with respect to adaptive identity and selective attribute, then the* $\mathsf{IB\text{-}wHPS}$ $\Pi_{\mathsf{IB\text{-}wHPS}}$ *is adaptively secure;*
– *if the key-size of the* $\mathsf{IB\text{-}ABE}$ *scheme for policy function* f *is* $s(f)$*, then the key size of the* $\mathsf{IB\text{-}wHPS}$ *is* $s(f) + n \log m$.

11.2 Instantiations of IB-ABE

We observe that any adaptive ABE for polynomial evaluation [30] suffices to construct the required IB-ABE, as we can encode $(\mathsf{id}, 1, y_1), (\mathsf{id}, 2, y_2), \ldots, (\mathsf{id}, n, y_n)$ as n roots of the polynomial $p(x)$, and $\mathsf{sk}_{\mathsf{id},f_{\boldsymbol{y}}}$ corresponds to $\mathsf{sk}_{p(x)}$. Moreover, the work [30] constructed such an ABE from some bilinear groups, even though their

secret key size is not succinct. Subsequent work [18] constructed a more efficient ABE for inner products (which implies ABE for polynomial evaluation [30]) with succinct secret keys from some bilinear groups. Thus, the construction of [18] suffices to achieve the desired IB-wHPS and leakage resilient IBE with the optimal rate $1 - o(1)$, via our framework. Recently, Nishimaki and Yamakawa [37] proposed a new method to succinctly encode several different identity secret keys through using inner product encryption, which can also be viewed as an instantiation for IB-ABE that achieves the ratio $|\mathsf{sk}_{\boldsymbol{a}}|/|\boldsymbol{a}| = o(1)$. However, since there is not existing adaptive inner product encryption scheme from lattices, we still can not obtain the required IB-ABE from lattices by using the approach of inner product encryption as Nishimaki and Yamakawa [37].

To tackle this challenge, we use the new construction of the work [32]. For completeness, we present the necessary lattice back ground information in Section 11.2. And then, in Section 11.2 we present their construction, which achieves the desired ratio $|\mathsf{sk}_{\boldsymbol{a}}|/|\boldsymbol{a}| = o(1)$, and thus leakage-resilient adaptive IBE with the optimal leakage rate.

Background Knowledge on Lattices

A lattice is a discrete additive subgroup of $\mathbb{R}^m$. Let $\mathbf{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m) \subset \mathbb{R}^m$ consists of m linearly independent vectors. The m-dimensional lattice Λ generated by the basis $\mathbf{B}$ is $\Lambda = \mathcal{L}(\mathbf{B}) = \{\mathbf{B} \cdot \boldsymbol{c} = \sum_{i \in [m]} c_i \cdot \boldsymbol{b}_i : \boldsymbol{c} = (c_1, \ldots, c_m) \in \mathbb{Z}^m\}$.

The minimum distance $\lambda_1(\Lambda)$ of a lattice Λ is the length in the Euclidean ℓ_2 norm of the shortest nonzero vector: $\lambda_1(\Lambda) = \min_{0 \neq \mathbf{x} \in \Lambda} \|\boldsymbol{x}\|$. For an approximation factor $\gamma = \gamma(n) > 1$, we define the problem GapSVP$_\gamma$ as follows: given a basis $\mathbf{B}$ of an m-dimensional lattice $\Lambda = \mathcal{L}(\mathbf{B})$ and a positive number d, distinguish between the case where $\lambda_1(\Lambda) \leq d$ and the case where $\lambda_1(\Lambda) \geq \gamma d$. We let $\widetilde{\mathbf{B}}$ denote the Gram-Schmidt orthogonalization of $\mathbf{B}$, and $\|\widetilde{\mathbf{B}}\|$ is the length of the longest vector in it.

In this paper, we will focus on a particular family of integer lattices. Let $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ for three positive integers m, n, q, where m and q are functions of n. We consider the following two kinds of full-rank m-dimensional integer lattices defined by $\Lambda_q^{\perp}(\mathbf{A}) = \{\boldsymbol{e} \in \mathbb{Z}^m : \mathbf{A}^\top \cdot \boldsymbol{e} = 0 \bmod q\}$ and its shift $\Lambda_q^{\boldsymbol{u}}(\mathbf{A}) = \{\boldsymbol{e} \in \mathbb{Z}^m : \mathbf{A}^\top \cdot \boldsymbol{e} = \boldsymbol{u} \bmod q\}$.

Lemma 11.8 ([6]) *For any integers $n \geq 1$, $q \geq 2$, and sufficiently large $m = \lceil 6n \log q \rceil$, there is a probabilistic polynomial-time algorithm* $\mathsf{TrapGen}(q, n)$ *that outputs a pair $(\mathbf{A} \in \mathbb{Z}_q^{m \times n}, \mathbf{T}_{\mathbf{A}} \in \mathbb{Z}^{m \times m})$ such that the distribution of $\mathbf{A}$ is statistically close to the uniform distribution over $\mathbb{Z}_q^{m \times n}$ and $\mathbf{T}_{\mathbf{A}}$ is a short basis for $\Lambda_q^{\perp}(\mathbf{A})$ satisfying $\|\widetilde{\mathbf{T}}_{\mathbf{A}}\| \leq O(\sqrt{n \log q})$ and $\|\mathbf{T}_{\mathbf{A}}\| \leq O(n \log q)$ with overwhelming probability.*

Gaussians on Lattices Let σ be any positive real number. The Gaussian distribution $\mathcal{D}_{\sigma, \boldsymbol{c}}$ with parameter σ and $\boldsymbol{c}$ is defined by probability distribution function $\rho_{\sigma, \boldsymbol{c}}(\boldsymbol{x}) = exp(-\pi \|\boldsymbol{x} - \boldsymbol{c}\|^2 / \sigma^2)$. For any set $S \in \mathbb{R}^m$, define

$\rho_{\sigma,\boldsymbol{c}}(S) = \sum_{\boldsymbol{x}\in S} \rho_{\sigma,\boldsymbol{c}}(\boldsymbol{x})$. The discrete Gaussian distribution $D_{S,\sigma,\boldsymbol{c}}$ over S with parameter σ and $\boldsymbol{c}$ is defined by the probability distribution function $\rho_{\sigma,\boldsymbol{c}}(\boldsymbol{x}) = \rho_{\sigma,\boldsymbol{c}}(\boldsymbol{x})/\rho_{\sigma,\boldsymbol{c}}(S)$ for all $\boldsymbol{x} \in S$.

Lemma 11.9 ([2], Lemma 8) *Let* $\mathbf{A}$ *and* $\mathbf{T_A}$ *be a pair of matrices output by* $\mathsf{TrapGen}(q,n)$*, and* $r \geqslant \|\widetilde{\mathbf{T}}_{\mathbf{A}}\| \cdot \omega(\sqrt{\log m})$*. Then for* $\boldsymbol{c} \in \mathbb{R}^m$ *and* $\boldsymbol{u} \in \mathbb{Z}_q^n$*, we have:*

1. $\mathsf{Pr}[\boldsymbol{x} \leftarrow D_{\Lambda_q^{\boldsymbol{u}}(\mathbf{A}),r} : \|\boldsymbol{x}\| > r\sqrt{m}] \leq \mathsf{negl}(n)$.
2. *There is a probabilistic polynomial-time algorithm* $\mathsf{SampleGaussian}(\mathbf{A}, \mathbf{T_A}, r, \boldsymbol{c})$ *that outputs a sample from a distribution statistically to* $D_{\Lambda,r,\boldsymbol{c}}$.
3. *There is a probabilistic polynomial-time algorithm* $\mathsf{SamplePre}(\mathbf{A}, \mathbf{T_A}, \boldsymbol{u}, r)$ *that outputs a sample from a distribution statistically to* $D_{\Lambda_q^{\boldsymbol{u}}(\mathbf{A}),r}$.

Definition 11.10 *For an* $\beta \in (0,1)$ *and a integer* q*, let* $\bar{\psi}_\beta$ *denote the distribution over* $\mathbb{Z}_q$ *of the random variable* $\lfloor qx \rceil$*, where* x *is a normal random variable with mean 0 and standard deviation* $\beta/\sqrt{2\pi}$.

The next two efficient algorithms $\mathsf{SampleLeft}$ is used to generate identity secret key for our new constructions.

Lemma 11.11 ([2], Theorem 17) *Let* $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ *and* $\mathbf{T_A} \in \mathbb{Z}^{m\times m}$ *be a pair of matrices output by* $\mathsf{TrapGen}(q,n)$*, let* $\mathbf{M}$ *be a matrix in* $\mathbb{Z}_q^{n\times m_1}$*,* $\boldsymbol{u}$ *be a vector in* $\mathbb{Z}_q^n$*, and* r *satisfy* $r > \|\widetilde{\mathbf{T}}_{\mathbf{A}}\| \cdot \omega(\sqrt{\log(m+m_1)})$*. There is a probabilistic polynomial-time algorithm* $\mathsf{SampleLeft}(\mathbf{A}, \mathbf{M}, \mathbf{T_A}, \boldsymbol{u}, r)$ *that outputs a vector* $\boldsymbol{e} \in \mathbb{Z}^{m+m_1}$ *distributed statistically close to* $\mathcal{D}_{\Lambda_q^{\boldsymbol{u}}(\mathbf{F}_1),r}$*, where* $\mathbf{F}_1 = (\mathbf{A}|\mathbf{M})$.

Learning With Errors. The Learning with errors problem, or LWE, is the problem of determining a secret vector over $\mathbb{F}_q$ given a polynomial number of "noisy" inner products. The decision variant is to distinguish such samples from random. More formally, we define the problem as follows:

Definition 11.12 ([39]) *Let* $n \geq 1$ *and* $q \geq 2$ *be integers, and let* χ *be a probability distribution on* $\mathbb{Z}_q$*. For* $\boldsymbol{s} \in \mathbb{Z}_q^n$*, Let* $A_{\boldsymbol{s},\chi}$ *be the probability distribution on* $\mathbb{Z}_q^n \times \mathbb{Z}_q$ *obtained by choosing a vector* $\boldsymbol{a} \in \mathbb{Z}_q^n$ *uniformly at random, choosing* $e \in \mathbb{Z}_q$ *according to* χ *and outputting* $(\boldsymbol{a}, \langle \boldsymbol{a}, \boldsymbol{s}\rangle + e)$.

The decision $\mathsf{LWE}_{q,n,\chi}$ *problem is: for uniformly random* $\boldsymbol{s} \in \mathbb{Z}_q^N$*, given a poly(n) number of samples that are either (all) from* $A_{\boldsymbol{s},\chi}$ *or (all) uniformly random in* $\mathbb{Z}_q^n \times \mathbb{Z}_q$*, output 0 if the former holds and 1 if the latter holds.*

We say the decision-$\mathsf{LWE}_{q,n,\chi}$ problem is infeasible if for all polynomial-time algorithms $\mathcal{A}$, the probability that $\mathcal{A}$ solves the decision-LWE problem (over $\boldsymbol{s}$ and $\mathcal{A}$'s random coins) is negligibly close to $1/2$ as a function of n. the works of [15,38,39] show that the LWE assumption is as hard as (quantum or classical) solving GapSVP and SIVP under various parameter regimes.

Concrete Instantiation of IB-ABE from Lattices [32]

We describe the construction of [32] that achieves a partial-adaptively secure IB-ABE as needed in Section 11.1 from LWE with a polynomial modulus.

- $\mathsf{IB\text{-}ABE.Setup}(1^\lambda)$ The setup algorithm takes as input a security parameter λ, and then dose the following:
  1. Sample a random matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ along with a trapdoor basis $\mathbf{T_A} \in \mathbb{Z}^{m\times m}$ of lattice $\Lambda_q^\perp(\mathbf{A})$ by running $\mathsf{TrapGen}$.
  2. Select $\ell_1 + 1$ uniformly random matrices $\mathbf{A}_1, \ldots, \mathbf{A}_{\ell_1}, \mathbf{B} \in \mathbb{Z}_q^{n\times m}$.
  3. Select ℓ_2 uniformly random matrices $\mathbf{C}_1, \ldots, \mathbf{C}_{\ell_2} \in \mathbb{Z}_q^{n\times m}$.
  4. Select a random matrix $\mathbf{U} \xleftarrow{\$} \mathbb{Z}_q^{n\times z}$.
  5. Output the public parameters

  $$\mathsf{mpk} = (\mathbf{A}, \{\mathbf{A}_i\}_{i\in[\ell_1]}, \{\mathbf{C}_i\}_{i\in[\ell_2]}, \mathbf{B}, \mathbf{U})$$

  and the master secret key $\mathsf{msk} = (\mathbf{T_A})$.
- $\mathsf{IB\text{-}ABE.KeyGen}(\mathsf{mpk}, \mathsf{msk}, \mathsf{id}, f)$ The key generation algorithm takes as input $\mathsf{mpk}, \mathsf{msk}$, an identity $\mathsf{id} = (b_1, b_2, ..., b_{\ell_1}) \in \{1, -1\}^{\ell_1}$ and a policy function f with depth d, and then does the following:
  1. Compute $\mathbf{A}_{\mathsf{id}} = \mathbf{B} + \sum_{i=1}^{\ell_1}(b_i\mathbf{A}_i) \in \mathbb{Z}_q^{n\times m}$.
  2. Define function $\bar{f}(\cdot) = 1 - f(\cdot)$, and compute $\mathbf{H}_f = \mathsf{Eval}_{\mathsf{pk}}(\bar{f}, \mathbf{C}_1, \ldots, \mathbf{C}_{\ell_2}) \in \mathbb{Z}_q^{n\times m}$.
  3. Let $\mathbf{F}_{\mathsf{id},f} = (\mathbf{A}|\mathbf{A}'_{\mathsf{id},f}) = (\mathbf{A}|\mathbf{A}_{\mathsf{id}}|\mathbf{H}_f) \in \mathbb{Z}_q^{n\times 3m}$.
  4. Sample $\mathbf{D} \in \mathbb{Z}^{3m\times z}$ as $\mathbf{D} \leftarrow \mathsf{SampleLeft}(\mathbf{A}, \mathbf{T_A}, \mathbf{A}'_{\mathsf{id},f}, \mathbf{U}, \sigma)$.
  5. Output $\mathsf{sk}_{\mathsf{id},f} := \mathbf{D}$, where $\mathbf{F}_{\mathsf{id},f} \cdot \mathbf{D} = \mathbf{U} \bmod q$.
- $\mathsf{IB\text{-}ABE.Enc}(\mathsf{mpk}, \mathsf{id}, \boldsymbol{x}, \boldsymbol{\mu})$ In order to encrypt a message $\boldsymbol{\mu} \in \{0,1\}^z$ with respect to an identity $\mathsf{id} = (b_1, b_2, ..., b_{\ell_1}) \in \{1, -1\}^{\ell_1}$ and attribute $\boldsymbol{x} = (x_1, \ldots, x_{\ell_2}) \in \mathbb{Z}_q^{\ell_2}$, the encryption algorithm first chooses a random vector $\boldsymbol{s} \leftarrow \mathbb{Z}_q^n$ and two error vectors $\boldsymbol{e}_0 \leftarrow \chi^m$, $\boldsymbol{e}_1 \leftarrow \chi^z$ where χ is a B bounded discrete Gaussian distribution, and then does the following:
  1. Compute $\mathbf{A}_{\mathsf{id}} = \mathbf{B} + \sum_{i=1}^{\ell_1}(b_i\mathbf{A}_i) \in \mathbb{Z}_q^{n\times m}$.
  2. Choose ℓ_1 uniformly random matrices $\mathbf{R}_i \leftarrow \{-1, 1\}^{m\times m}$ for $i \in [\ell_1]$, and compute $\mathbf{R}_{\mathsf{id}} = \sum_{i=1}^{\ell_1}(b_i\mathbf{R}_i) \in \{-\ell_1, \ldots, \ell_1\}^{m\times m}$.
  3. Set $\boldsymbol{e}_2 = \mathbf{R}_{\mathsf{id}}^\top \cdot \boldsymbol{e}_0 \in \mathbb{Z}_q^m$.
  4. Set $\mathbf{H}_{\boldsymbol{x}} = (x_1\mathbf{G} + \mathbf{C}_1|\cdots|x_{\ell_2}\mathbf{G} + \mathbf{C}_{\ell_2}) \in \mathbb{Z}_q^{n\times m\ell_2}$.
  5. Choose ℓ_2 uniformly random matrices $\mathbf{R}'_j \leftarrow \{-1, 1\}^{m\times m}$ for $j \in [\ell_2]$, and set $\boldsymbol{e}_3 = (\mathbf{R}'_1|\cdots|\mathbf{R}'_{\ell_2})^\top \cdot \boldsymbol{e}_0 \in \mathbb{Z}_q^{m\ell_2}$.
  6. Set $\mathbf{F}_{\mathsf{id},\boldsymbol{x}} = (\mathbf{A}|\mathbf{A}'_{\mathsf{id},\boldsymbol{x}}) = (\mathbf{A}|\mathbf{A}_{\mathsf{id}}|\mathbf{H}_{\boldsymbol{x}}) \in \mathbb{Z}_q^{n\times(2+\ell_2)m}$.
  7. Output $\boldsymbol{c} = (\mathbf{F}_{\mathsf{id},f}^\top \cdot \boldsymbol{s} + (\boldsymbol{e}_0^\top, \boldsymbol{e}_2^\top, \boldsymbol{e}_3^\top)^\top, \mathbf{U}^\top \cdot \boldsymbol{s} + \boldsymbol{e}_1 + \lfloor q/2 \rceil \boldsymbol{\mu}) \in \mathbb{Z}_q^{(2+\ell_2)m+z}$.
- $\mathsf{IB\text{-}ABE.Dec}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{id},f}, (\boldsymbol{x}, \boldsymbol{c}))$ The decryption algorithm uses the key $\mathsf{sk}_{\mathsf{id},f} := \mathbf{D}$ to decrypt $\boldsymbol{c}$ with attribute $\boldsymbol{x}$. If $f(\boldsymbol{x}) \neq 1$, output $\perp$. Otherwise, let the ciphertext $\boldsymbol{c} = (\boldsymbol{c}_{in,1}, \boldsymbol{c}_{in,2}, \boldsymbol{c}_1, \ldots, \boldsymbol{c}_{\ell_2}, \boldsymbol{c}_{out}) \in \mathbb{Z}_q^{(2+\ell_2)m+z}$, compute $\boldsymbol{c}_f = \mathsf{Eval}_{ct}(\bar{f}, \{(x_i, \mathbf{C}_i, \boldsymbol{c}_i)\}_{i=1}^{\ell_2}) \in \mathbb{Z}_q^m$, where $\boldsymbol{c}_{in,1}, \boldsymbol{c}_{in,2} \in \mathbb{Z}_q^m$, $\boldsymbol{c}_{out} \in \mathbb{Z}_q^z$ and $\boldsymbol{c}_i \in \mathbb{Z}_q^m$ for $1 \leq i \leq \ell_2$.
  Let $\boldsymbol{c}'_f = (\boldsymbol{c}_{in,1}, \boldsymbol{c}_{in,2}, \boldsymbol{c}_f) \in \mathbb{Z}_q^{3m}$ and output $\mathsf{Round}(\boldsymbol{c}_{out} - \mathbf{D}^\top \cdot \boldsymbol{c}'_f) \in \{0,1\}^m$.

Correctness. The correctness of the scheme follows from our choice of parameters. Specifically, to show correctness first note that when $f(\boldsymbol{x}) = 1$ we know $\boldsymbol{c}_f = \mathbf{H}_f^\top \cdot \boldsymbol{s} + \boldsymbol{e}_f$, then we have during decryption,

$$\begin{aligned}
\boldsymbol{\mu}' &= \mathsf{Round}(\boldsymbol{c}_{out} - \mathbf{D}^\top \cdot \boldsymbol{c}'_f) \\
&= \mathsf{Round}(\boldsymbol{c}_{out} - \mathbf{D}^\top \cdot ((\mathbf{A}|\mathbf{A}_{\mathsf{id}}|\mathbf{H}_f)^\top \cdot \boldsymbol{s} - (\boldsymbol{e}_0, \boldsymbol{e}_2, \boldsymbol{e}_f + \boldsymbol{e}_3))) \\
&= \mathsf{Round}(\mathbf{U}^\top \cdot \boldsymbol{s} + \boldsymbol{e}_1 + \lfloor q/2 \rceil \boldsymbol{\mu} - \mathbf{U}^\top \cdot \boldsymbol{s} - \mathbf{D}^\top \cdot (\boldsymbol{e}_0, \boldsymbol{e}_2, \boldsymbol{e}_f + \boldsymbol{e}_3)) \\
&= \mathsf{Round}(\lfloor q/2 \rceil \boldsymbol{\mu} + \boldsymbol{e}_1 - \mathbf{D}^\top \cdot (\boldsymbol{e}_0, \boldsymbol{e}_2, \boldsymbol{e}_f + \boldsymbol{e}_3)) \\
&= \boldsymbol{\mu}
\end{aligned}$$

This completes the proof of correctness.

Parameter Setting for our Construction. For arbitrarily small constant ϵ, we set the system parameters according to the Table below.

Parameters	Description	Setting
λ	security parameter	
z	message length	$O(\log \lambda)$
n	PK-lattice row dimension	λ
m	PK-lattice column dimension	$n^{1+\epsilon}$
q	modulus	$n^5 m^4$
d	depth of f	$O(\log \lambda)$
σ	SampleLeft and SampleRight parameter	$n^2 m^2$
B	bound of errors	$\sqrt{\lambda}$
ℓ_1	identity length	n
ℓ_2	attribute length	n

Table 8. Parameter Setting

The work [32] showed the following theorem, which is what we need and omitted just for clarity.

Theorem 11.13 *For parameter setting in Table 8, the above* IB-ABE *scheme in the Appendix 11.2 is correct and secure with respect to adaptive identity and selective attribute, assuming the* $\mathsf{LWE}_{n,q,\chi}$ *assumption holds.*

Finally, by instantiating Construction 10.2 with (1) the specific IB-wHPS from LWE in Construction 11.6 and (2) the LWE-based on-the-fly KDM-secure PKE in Corollary 11.2, we are able to achieve the following corollary via Theorem 10.4:

Corollary 11.14 *Assuming that* LWE *is hard, there exists a rate-1 (both information and leakage rates)* IBE *that is leakage resilient against block leakage and* $\mathsf{KDM}^{(\bar{n})}$*-secure w.r.t.* $\hat{\mathcal{F}}' = (\mathcal{F}_{s'} || \hat{\mathcal{Q}}^{\tau'})$ *for any unbounded polynomial* $\bar{n}$*.*

Similarly, by instantiating Construction 10.2 with (1) the specific IB-wHPS from DDH in the bilinear group in Construction 11.6 and (2) the DDH-based on-the-fly KDM-secure PKE in Corollary 11.2, we are able to achieve the following corollary via Theorem 10.4:

Corollary 11.15 *Assuming that* DDH *(in the bilinear group) is hard, there exists a rate-1 (both information and leakage rates)* IBE *that is leakage resilient against block leakage and* $\mathsf{KDM}^{(\bar{n})}$*-secure w.r.t.* $\hat{\mathcal{F}}' = (\mathcal{F}_{s'}||\hat{\mathcal{Q}}^{\tau'})$ *for any unbounded polynomial* $\bar{n}$*.*

Acknowledgements. We would like to thank the reviewers of PKC 2022 for their insightful advices. Qiqi Lai is supported by the National Natural Science Foundation of China (62172266, 61802241, U2001205), and Henan Key Laboratory of Network Cryptography Technology (LNCT2021-A03). Feng-Hao Liu is supported by the NSF Career Award CNS-1942400. Zhedong Wang is supported by the National Science Foundation of China (62202305) and the Shanghai Pujiang Program (22PJ1407700).

References

1. P. Adão, G. Bana, J. Herzog, and A. Scedrov. Soundness of formal encryption in the presence of key-cycles. In S. D. C. di Vimercati, P. F. Syverson, and D. Gollmann, editors, *ESORICS 2005*, volume 3679 of *LNCS*, pages 374–396. Springer, Heidelberg, Sept. 2005.
2. S. Agrawal, D. Boneh, and X. Boyen. Efficient lattice (H)IBE in the standard model. In Gilbert [23], pages 553–572.
3. J. Alperin-Sheriff and C. Peikert. Circular and KDM security for identity-based encryption. In M. Fischlin, J. Buchmann, and M. Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 334–352. Springer, Heidelberg, May 2012.
4. J. Alwen, Y. Dodis, M. Naor, G. Segev, S. Walfish, and D. Wichs. Public-key encryption in the bounded-retrieval model. In Gilbert [23], pages 113–134.
5. J. Alwen, S. Krenn, K. Pietrzak, and D. Wichs. Learning with rounding, revisited - new reduction, properties and applications. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 57–74. Springer, Heidelberg, Aug. 2013.
6. J. Alwen and C. Peikert. Generating shorter bases for hard random lattices. *Theory of Computing Systems*, 48(3):535–553, 2010.
7. B. Applebaum. Key-dependent message security: Generic amplification and completeness. In K. G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 527–546. Springer, Heidelberg, May 2011.
8. B. Applebaum, D. Cash, C. Peikert, and A. Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In Halevi [27], pages 595–618.
9. A. Banerjee, C. Peikert, and A. Rosen. Pseudorandom functions and lattices. In D. Pointcheval and T. Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 719–737. Springer, Heidelberg, Apr. 2012.
10. B. Barak, I. Haitner, D. Hofheinz, and Y. Ishai. Bounded key-dependent message security. In Gilbert [23], pages 423–444.

11. J. Black, P. Rogaway, and T. Shrimpton. Encryption-scheme security in the presence of key-dependent messages. In K. Nyberg and H. M. Heys, editors, *Selected Areas in Cryptography – SAC 2002*, pages 62–75, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.
12. D. Boneh, S. Halevi, M. Hamburg, and R. Ostrovsky. Circular-secure encryption from decision Diffie-Hellman. In D. Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 108–125. Springer, Heidelberg, Aug. 2008.
13. Z. Brakerski and S. Goldwasser. Circular and leakage resilient public-key encryption under subgroup indistinguishability - (or: Quadratic residuosity strikes back). In T. Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 1–20. Springer, Heidelberg, Aug. 2010.
14. Z. Brakerski, S. Goldwasser, and Y. T. Kalai. Black-box circular-secure encryption beyond affine functions. In Y. Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 201–218. Springer, Heidelberg, Mar. 2011.
15. Z. Brakerski, A. Langlois, C. Peikert, O. Regev, and D. Stehlé. Classical hardness of learning with errors. In D. Boneh, T. Roughgarden, and J. Feigenbaum, editors, *45th ACM STOC*, pages 575–584. ACM Press, June 2013.
16. Z. Brakerski, A. Lombardi, G. Segev, and V. Vaikuntanathan. Anonymous IBE, leakage resilience and circular security from new assumptions. In J. B. Nielsen and V. Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 535–564. Springer, Heidelberg, Apr. / May 2018.
17. J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In B. Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 93–118. Springer, Heidelberg, May 2001.
18. J. Chen, R. Gay, and H. Wee. Improved dual system ABE in prime-order groups via predicate encodings. In E. Oswald and M. Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 595–624. Springer, Heidelberg, Apr. 2015.
19. R. Cramer and V. Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In L. R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 45–64. Springer, Heidelberg, Apr. / May 2002.
20. Y. Dodis, Y. T. Kalai, and S. Lovett. On cryptography with auxiliary input. In Mitzenmacher [35], pages 621–630.
21. Y. Dodis, R. Ostrovsky, L. Reyzin, and A. D. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139, 2008.
22. N. Döttling. Low noise LPN: KDM secure public key encryption and sample amplification. In J. Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 604–626. Springer, Heidelberg, Mar. / Apr. 2015.
23. H. Gilbert, editor. *EUROCRYPT 2010*, volume 6110 of *LNCS*. Springer, Heidelberg, May / June 2010.
24. O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions. In *21st ACM STOC*, pages 25–32. ACM Press, May 1989.
25. S. Goldwasser and S. Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.
26. V. Goyal and Y. Song. Correlated-source extractors and cryptography with correlated-random tapes. In Y. Ishai and V. Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 562–592. Springer, Heidelberg, May 2019.
27. S. Halevi, editor. *CRYPTO 2009*, volume 5677 of *LNCS*. Springer, Heidelberg, Aug. 2009.

28. C. Hazay, A. López-Alt, H. Wee, and D. Wichs. Leakage-resilient cryptography from minimal assumptions. In T. Johansson and P. Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 160–176. Springer, Heidelberg, May 2013.
29. D. Hofheinz and D. Unruh. Towards key-dependent message security in the standard model. In Smart [40], pages 108–126.
30. J. Katz, A. Sahai, and B. Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In Smart [40], pages 146–162.
31. F. Kitagawa and K. Tanaka. Key dependent message security and receiver selective opening security for identity-based encryption. In M. Abdalla and R. Dahab, editors, *PKC 2018, Part I*, volume 10769 of *LNCS*, pages 32–61. Springer, Heidelberg, Mar. 2018.
32. Q. Lai, F.-H. Liu, and Z. Wang. Leakage-resilient ibe/abe with optimal leakage rates from lattices. *Public Key Cryptography (2) 2022: 225-255*, 2020.
33. P. Laud and R. Corin. Sound computational interpretation of formal encryption with composed keys. In J. I. Lim and D. H. Lee, editors, *ICISC 03*, volume 2971 of *LNCS*, pages 55–66. Springer, Heidelberg, Nov. 2004.
34. D. Micciancio. On the hardness of learning with errors with binary secrets. *Theory of Computing*, 14(1):1–17, 2018.
35. M. Mitzenmacher, editor. *41st ACM STOC*. ACM Press, May / June 2009.
36. M. Naor and G. Segev. Public-key cryptosystems resilient to key leakage. In Halevi [27], pages 18–35.
37. R. Nishimaki and T. Yamakawa. Leakage-resilient identity-based encryption in bounded retrieval model with nearly optimal leakage-ratio. In D. Lin and K. Sako, editors, *PKC 2019, Part I*, volume 11442 of *LNCS*, pages 466–495. Springer, Heidelberg, Apr. 2019.
38. C. Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In Mitzenmacher [35], pages 333–342.
39. O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In H. N. Gabow and R. Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005.
40. N. P. Smart, editor. *EUROCRYPT 2008*, volume 4965 of *LNCS*. Springer, Heidelberg, Apr. 2008.
41. S. P. Vadhan. Pseudorandomness. *Foundations and Trends® in Theoretical Computer Science*, 7(1-3):1–336.
42. H. Wee. KDM-security via homomorphic smooth projective hashing. In C.-M. Cheng, K.-M. Chung, G. Persiano, and B.-Y. Yang, editors, *PKC 2016, Part II*, volume 9615 of *LNCS*, pages 159–179. Springer, Heidelberg, Mar. 2016.