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

Qiqi Lai[1,2], Feng-Hao Liu[3], Zhedong Wang[3]

[1] School of Computer Science, Shaanxi Normal University, Xi'an, China.
laiqq@snnu.edu.cn.
[2] State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an, China
[3] Florida Atlantic University, Boca Raton, FL, USA.
{fenghao.liu,wangz}@fau.edu.

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*, *homomorphic*, 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 [18] 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, 10, 13, 24]. To tackle this issue, [8, 9] formally defined *Key Dependent Message* (KDM) security, which requires $\mathsf{Enc}(\mathsf{pk}, \mathfrak{f}(\mathsf{sk}))$ to be indistinguishable from $\mathsf{Enc}(\mathsf{pk}, 0)$ for all $\mathfrak{f}$ 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 $\mathfrak{f}(\mathsf{sk}_1, \ldots, \mathsf{sk}_n)$ 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 [5], DDH [9], or LPN [5, 16, 23] assumption, for bounded polynomial n from QR/DCR assumption [10], and for $n = 1$ from CDH [12].

On the other hand however, all the prior constructions have relatively small information rate[4] even for the class of linear functions, resulting in very large

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

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 [22]. This direction has remained an important open problem (ref. [9, 10]). 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) [12], 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 [5, 7, 9–12]. 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 [12], 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., $1 - o(1)$.

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 [4], 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 [4] does not work. Thus, the 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[5]. 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) [2, 21], Batch Encryption (BE) [12], and randomness extractors [3], 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.
- **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.,

[5] 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.

$H_\infty(\mathsf{Decap}(\mathsf{sk}, \mathsf{CT}^*) \mid \mathsf{pk}, \mathsf{CT}^*) \geq w$. 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 [2,25]. Homomorphic wHPS has been used to achieve $\mathsf{KDM}^{(1)}$-security [27]. It was not clear whether wHPS can be used to achieve $\mathsf{KDM}^{(1)}$-security with the optimal information rate.

Batch Encryption [12]. 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 [12] 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 [12] 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.

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. [2, 25]).[6] 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.[7] Previously, the work [3, 14, 25] 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 [27], 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:

$$(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_m, \mathsf{Ext}(\boldsymbol{r}_1, \mathfrak{g}_1(\boldsymbol{s})), \ldots, \mathsf{Ext}(\boldsymbol{r}_m, \mathfrak{g}_m(\boldsymbol{s}))) \approx (\boldsymbol{r}_1, \ldots, \boldsymbol{r}_m, u_1, \ldots, u_m)\,.^{8}$$

Our notion of correlated-source attacks is similar to that of a recent work by Goyal and Song [19], yet with the following major differences. First, the security requirements are different. The work [19] considers information-theoretic indistinguishability of *one* instance of extraction from the *original source*, even given multiple extractions from the modified source, i.e.,

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

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

[7] 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.

[8] 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.

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 [19] 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 [19] 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 [3,25]: $\mathsf{Ext}(\boldsymbol{r}, \boldsymbol{s}) = \prod_{i=1}^n r_i^{s_i}$. 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 [3]. 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 [3,6] 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 [6], 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.

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

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 [5,7,9–12], 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 5.1.

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 $|\boldsymbol{r}| = o(wd)$.

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 [12]. We call this type of schemes as BE-based public key encryption scheme.

Next we generalize the idea of [9], 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 [12] 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 [12], 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 [12] 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 [12]. 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 [4] 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., $1 - o(1)$.

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 IB-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 [23], which might not be leakage resilient.

Our compiler is straight-forward. Let Π be a BE-based PKE and IB-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 IB-wHPS. To encrypt a message m with an id, 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 IB-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 $\mathsf{IB\text{-}wHPS}$ 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 [4]. 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 $\mathsf{KDM\text{-}PKE}$ with the $\mathsf{IB\text{-}wHPS}$, we are able to achieve $\mathsf{KDM\text{-}IBE}$. Moreover, we can prove that the above PKE_2 satisfies the on-the-fly notion. We refer details to full version.

2 Preliminaries

We use several standard mathematical notations, whose detailed descriptions are deferred to the full version. In the full version we present the formal definitions of KDM-security and leakage resilience. Below, we present the syntax of two important tools – *batch encryption* and *weak hash proof systems*. Due to space limit, we defer their detailed security properties to the full version.

Definition 2.1 (Batch Encryption in [12]) *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 2.2 *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* $H_\infty(\mathsf{sk}|\mathsf{pk}) = H_\infty(\boldsymbol{x}|h) \geq n \log B - \hat{\ell}$*.*

Definition 2.3 (Weak Hash Proof System in [21]) *A weak hash proof system (*wHPS*) with the encapsulated-key-space* $\mathcal{K}$ *consists of four algorithms* $\mathsf{wHPS}.\{\mathsf{Setup},\mathsf{Encap},\mathsf{Encap}^*,\mathsf{Decap}\}$ *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* $\mathsf{k} \in \mathcal{K}$.

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

Definition 2.4 (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 2.5 *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 [3]) *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*[9]*, if for any correlated random variables*

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

(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 [15]. 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 [3, 25], 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* $\mathcal{F} = \{\mathfrak{f} : \mathcal{S} \to \mathcal{Y}\}$ *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* $\mathsf{Ext}(x, s) \circ \mathfrak{h}(s) = \mathsf{Ext}(\mathfrak{h}'(x), s)$*.*

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 [6]. 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* $\boldsymbol{x} \in \mathbb{Z}_2^n$*.*

Due to space limit, we defer the detailed proof to the full version.

Construction 3.8 (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 3.9 *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 3.8 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* $\boldsymbol{s} \in \mathbb{Z}_2^n$*.*

Due to space limit, we defer the detailed proof to the full version.

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 2.3;*
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 *described in full version. 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 $\boldsymbol{x}$. □

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 [26]) *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. 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 $\mathsf{Ext}_{||}(r, \boldsymbol{s}) = (r, \mathsf{Ext}(r, s_1), \ldots, \mathsf{Ext}(r, s_m))$ to denote a parallel repetition of extractor.*

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 $\mathcal{S} = \mathcal{K} = \mathbb{Z}_B^n$ with $n = n' \cdot m$, and $\mathsf{Ext} : \mathcal{R} \times \mathbb{Z}_B^{n'} \to \mathcal{M}$ 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 $\mathsf{wHPS.Encap}$ 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 $\boldsymbol{k}' = \mathsf{wHPS.Decap}(\mathsf{sk}, \mathsf{CT}_0)$, 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 $\mathcal{S} = \mathcal{K} = \mathbb{Z}_B^n$, $n = mn'$, $w = n \log B - \hat{\ell}$, 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 $\mathsf{Ext} : \mathcal{R} \times \mathbb{Z}_B^{n'} \to \mathcal{M}$ is an (e, poly)-reusable extractor, 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*

1. *leakage-resilient against block leakage*[10]*, 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 2.5.*
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 full version due to space limit. The extra properties will be used for deriving* KDM *security, which will be formally presented in Sections 5.1 and 6. In Section 3.2, we have presented homomorphic extractors from* DDH *and* LWE*.*

5.1 Proof of $\mathsf{KDM}^{(1)}$-security

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 2.4);
- 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 2.5), where 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 $\boldsymbol{k}'$.

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

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 in the full version, 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. Due to space limit, we defer the detailed proof to the full version.

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

Lemma 5.7 $\big|\mathsf{Pr}[\mathcal{E}_{0.Q}] - \mathsf{Pr}[\mathcal{E}_1]\big| \leq \mathsf{negl}(\lambda)$, *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 $\mathcal{SF}_{B,n'} : \{\mathfrak{s} : \mathbb{Z}_B^{n'} \to \mathbb{Z}_B^{n'}\}$.*

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

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 $\bar{n}$. To achieve 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 2.1. 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 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* $\mathsf{KDM}^{(1)}$*-security 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* $\mathcal{M}$*.*

Due to the limitation of space, the detailed proof is deferred to the full version.

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 $\Pi^{\bar{n}}_{\mathsf{PKE}}$ 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.

Due to the limitation of space, we defer the detailed proof and the constructions of the required BE to the full version.

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

7 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 full version. we are able to achieve the following corollary via Theorems 6.4, 6.7.

Corollary 7.1 *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 3.8 and (2) the DDH-based BE in full version, we are able to achieve the following corollary via Theorems 6.4, 6.7:

Corollary 7.2 *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 [9]. 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 7.3 *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.*

8 Extensions

In this section, we further extend our above results in Section 7 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.

8.1 Garbled Circuits

In this section, we recall the key ingredient for the KDM amplification of Applebaum [4]: Garbled Circuits.

Definition 8.1 (Garbled Circuits [12]) *A garbling scheme consists of three algorithms* (Garble, Eval, 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:*

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

8.2 Bootstrapping to Larger Classes of KDM Functions

We first present a bootstrapped variant of Construction 5.3 by using the technique of garbled circuits.[11] Then, we analyze the KDM-security and information rate of this improved scheme.

Construction 8.2 (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:*

– $\widehat{\Pi}.\mathsf{KeyGen}(1^\lambda)$*: 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* $(\widetilde{C}, \widetilde{L}) \leftarrow \mathsf{GC}.\mathsf{Sim}(\mu_{1,1}, \ldots, \mu_{1,m})$ *with* $\widetilde{L} = \{\mathsf{lab}_{i,j}\}_{i\in[n'], j\in[m]}$*, and then runs* $\Pi.\mathsf{Enc}$ *to output the ciphertext*

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

[11] In [4], 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.

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* $\mathsf{GC}.\mathsf{Dec}(\widetilde{C}, \{\mathsf{lab}_{i,j}\})$ *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 8.3 *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 8.4 *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 2.5. 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 8.5 *For the parameter setting in Construction 8.2, 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 2.5, 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 8.4.*

Proof (Sketch). As pointed out by [4], 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., $\mathcal{F}_s \leq_{\mathsf{KDM}} \mathcal{G}^{n'}$. 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 [4].

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 8.6 *Although Theorem 8.5 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.

8.3 Upgrade to KDM-Secure and Leakage Resilient IBE

In this section, we present our general compiler to construct an IBE that is both KDM-secure and leakage resilient. The compiler uses as key ingredients an $\mathsf{IB\text{-}wHPS}$ (described in full version) with additional structure (ref. Remark 4.2) and an on-the-fly KDM-secure PKE (described in full version). Conceptually, this general IBE scheme can be view as the hybrid encryption of the $\mathsf{IB\text{-}wHPS}$ 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 $\mathsf{IB\text{-}wHPS}$, 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 8.7 (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* $\mathsf{IB\text{-}wHPS}$ *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* $\mathsf{IB\text{-}wHPS.KeyGen}$ *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* $m' = \Pi_{\mathsf{PKE}}.\mathsf{Dec}(\mathsf{CRS}, \boldsymbol{k}', \mathsf{CT}_2)$.

We sketch that the above construction can be proven to be 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. a restricted block-function class for any polynomial unbounded $\bar{n}$. Due to space limit, the corresponding formal theorem statement and its detailed proof are deferred to the full version.

Acknowledgements. We would like to thank the anonymous reviewers of P-KC 2021 for their insightful advices. Qiqi Lai is supported by the National Key R&D Program of China (2017YFB0802000), the National Natural Science Foundation of China (61802241, U2001205, 61772326, 61802242), the Natural Science Basic Research Plan in Shaanxi Province of China (2019JQ-360), the National Cryptography Development Foundation during the 13th Five-year Plan Period (MMJJ20180217), and the Fundamental Research Funds for the Central Universities (GK202103093). Feng-Hao Liu and Zhedong Wang are supported by an NSF Award CNS-1657040 and an NSF Career Award CNS-1942400. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the sponsors.

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. J. Alwen, Y. Dodis, M. Naor, G. Segev, S. Walfish, and D. Wichs. Public-key encryption in the bounded-retrieval model. In Gilbert [17], pages 113–134.
3. 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.

4. 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.
5. B. Applebaum, D. Cash, C. Peikert, and A. Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In Halevi [20], pages 595–618.
6. 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.
7. B. Barak, I. Haitner, D. Hofheinz, and Y. Ishai. Bounded key-dependent message security. In Gilbert [17], pages 423–444.
8. 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.
9. 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.
10. 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.
11. 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.
12. 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.
13. 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.
14. Y. Dodis, Y. T. Kalai, and S. Lovett. On cryptography with auxiliary input. In M. Mitzenmacher, editor, *41st ACM STOC*, pages 621–630. ACM Press, May / June 2009.
15. 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.
16. 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.
17. H. Gilbert, editor. *EUROCRYPT 2010*, volume 6110 of *LNCS*. Springer, Heidelberg, May / June 2010.
18. S. Goldwasser and S. Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.
19. 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.
20. S. Halevi, editor. *CRYPTO 2009*, volume 5677 of *LNCS*. Springer, Heidelberg, Aug. 2009.

21. 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.
22. D. Hofheinz and D. Unruh. Towards key-dependent message security in the standard model. In N. P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 108–126. Springer, Heidelberg, Apr. 2008.
23. 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.
24. 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.
25. M. Naor and G. Segev. Public-key cryptosystems resilient to key leakage. In Halevi [20], pages 18–35.
26. S. P. Vadhan. Pseudorandomness. *Foundations and Trends® in Theoretical Computer Science*, 7(1-3):1–336.
27. 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.