Vector Encoding over Lattices and Its Applications

Daniel Apon * Xiong Fan † Feng-Hao Liu ‡

Abstract

In this work, we design a new lattice encoding structure for vectors. Our encoding can be used to achieve a packed FHE scheme that allows some SIMD operations and can be used to improve all the prior IBE schemes and signatures in the series. In particular, with respect to FHE setting, our method improves over the prior packed GSW structure of Hiromasa et al. (PKC '15), as we do not rely on a circular assumption as required in their work. Moreover, we can use the packing and unpacking method to extract each single element, so that the homomorphic operation supports element-wise and cross-element-wise computation as well. In the IBE scenario, we improves over previous constructions supporting $O(\lambda)$-bit length identity from lattices substantially, such as Yamada (Eurocrypt '16), Katsumata, Yamada (Asiacrypt '16) and Yamada (Eprint '17), by shrinking the master public key to three matrices from standard Learning With Errors assumption. Additionally, our techniques from IBE can be adapted to construct a compact digital signature scheme, which achieves existential unforgeability under the standard Short Integer Solution (SIS) assumption with small polynomial parameters.

*University of Maryland, Email: dapon@cs.umd.edu.This work was supported in part by financial assistance award 70NANB15H328 from the U.S. Department of Commerce, National Institute of Standards and Technology.

†Cornell University, Email: xfan@cs.cornell.edu.This material is based upon work supported by IBM under Agreement 4915013672. 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.

‡Florida Atlantic University, Email: fenghao.liu@fau.edu.

1 Introduction

Lattice-based cryptography has matured significantly since the early works of Ajtai [Ajt96] and Regev [Reg05]. Most cryptographic primitives, ranging from basic public-key encryption (PKE) [Reg05] to more advanced schemes e.g., identity-based encryption (IBE) [CHKP10, ABB10], attribute-based encryption (ABE) [GVW13, BGG$^+$14], fully-homomorphic encryption (FHE) [Gen09, BV11, GSW13], etc., can be built from now-canonical lattice hardness assumptions, such as Regev's Learning with Errors (LWE). We refer readers to a beautiful survey of the power of lattices and recent developments by Peikert [Pei15]. In addition to this impressive progress in theory, a large efforts were made to design more efficient lattice-based cryptosystems with smaller parameters, especially for the advanced ones. A key question that persists is, how efficient (especially in space complexity) that lattice-based cryptosystems can be, while retaining a meaningful level of concrete security. We hope to further determine whether the advanced lattice-based cryptosystems inherently require more space and/or can achieve relaxed notions of security. This is not only of theoretic interests, but also can lead to more efficient designs in practice.

In this work, we make progress on the line by developing a novel type of lattice-based encoding, which we call a *vector encoding*. This encoding, for example, can serve as the core component of lattice ciphertexts. We show the usefulness of our new vector encoding abstraction by applying it across a variety of lattice-based encryption schemes. Specifically, we show how to add *single-instruction-multiple-data* (SIMD) style processing to the Gentry, Sahai, Waters [GSW13] fully homomorphic encryption (FHE) scheme, *without* the circular security requirement of previous work [HAO15]. The SIMD technique allows us to further construct lattice-based identity-based encryption (IBE) and digital signature schemes with a *constant number* of (standard sized) matrices in its master public key (respectively, verification key). Our results lead to a partial answer to the key question:

Lattice-based IBE is as compact as lattice-based PKE (Regev's scheme) up to a constant factor.

We hope that the ideas and techniques developed in this work help lead to further progress in determining the full-fledged of the question and optimizing lattice-based cryptosystems.

Fully-Homomorphic Encryption. Fully Homomorphic Encryption (FHE) is a type of encryption scheme where additions and multiplications can be performed directly on the ciphertexts in order to compute on the underlying, encrypted data. The first constructions were given by Gentry [Gen09] on ideal lattices and by van Dijk, et al. [vDGHV10] over the integers. A second generation of FHE schemes, based on the LWE problem, then emerged with the works of [BV11, BGV12]. Current FHE proposals are primarily based on the "GSW-FHE" paradigm [GSW13], where the homomorphic operations are matrix addition and matrix multiplication.

Of particular note for our work, Hiromasa, Abe, and Okamoto [HAO15] showed how to pack messages SIMD-style in GSW-FHE ciphertexts, at the cost of additionally assuming circular security (unrelated to bootstrapping). An interesting question is:

Question #1: *Is there a* (leveled) *GSW-FHE scheme that benefits from SIMD-style message packing, but which does* not *require a circular security assumption?*

Identity-Based Encryption. Identity-Based Encryption (IBE), first introduced by Shamir [Sha84], enables any pair of users to communicate securely and to verify each other's signatures without exchanging private or public keys. In particular, the public key of an IBE user may be an arbitrary string, such as their email address. The first IBE proposals came from [BF01] based on bilinear maps. For lattice-based constructions, Gentry, Peikert, and Vaikuntanathan [GPV08] gave the first proposal in the random oracle

model, and the concurrent works of Cash et al. [CHKP10] and Agrawal et al. [ABB10] constructed the first, lattice-based IBEs in the standard model.

A major drawback of the known lattice-based IBEs is that they either satisfy only the artificial notion of *selective security*, where an adversary declares at the start of the security experiment which identity it will attack, or they incur a large blow-up in the size of the master public key proportional to the security parameter λ or in the tightness of the security reduction. Indeed, designing an adaptively-secure IBE with comparable efficiency and concrete security guarantees to [GPV08] is a key focus of Peikert's [Pei15] Question 9, which we restate part of here:

Are there standard-model, adaptively secure lattice-based IBE/ABE/PE schemes that have comparable efficiency and concrete security to the existing selectively secure ones?

A full solution to the above question remains elusive, but some progress has been made. In one direction, Boyen and Li [BL16], building on the signature scheme of Katz and Wang [KW03], proposed lattice-based IBE and signature schemes in the standard model that are adaptively secure and have a *tight* reduction to the hardness of LWE with a slightly superpolynomial modulus, at the cost of a *large master public key growing with* λ. (The additional space provided by this large public parameter is used to encode a pseudorandom function key in their security proof.)

There has also been another sequence of recent works [Yam16, AFL16, ZCZ16, KY16] (for full details, see Table 1) whose goal is to reduce the size of the master public key while retaining adaptive security, at the cost of an inverse polynomial *security reduction loss*, growing with the number of adversarial key-queries Q. Each of these works provide various trade-offs and achieve a master public key (MPK) with sublinear $o(\lambda)$ blow-up; we briefly review the history of this line below.

Beginning with the works of Cash et al. [CHKP10] and Agrawal et al. [ABB10], adaptively-secure lattice-based IBEs in the standard model used $O(\lambda)$-many basic matrices in the MPK, where λ is the bit-length of users' identities (respectively, the security parameter). The first, significant improvement was due to Yamada [Yam16], who showed how to use the natural homomorphisms of lattice trapdoors to decrease the size of the MPK to $O(\lambda^{1/\mu})$, for arbitrary constant μ (say, 2 or 3), at the cost of using a superpolynomial LWE modulus. Another trade-off was presented in a prior draft of this work [AFL16], achieving a somewhat larger MPK of size $\lambda/\log^2 \lambda$ matrices, but ensuring that the LWE modulus could be a fixed polynomial. Following this, Zhang et al. [ZCZ16] showed a distinct technique that allowed for an MPK of only $O(\log(\lambda))$ matrices, under the assumption that the maximum number Q of secret keys any adversary obtains is known ahead of time.[1] Subsequent to that, Katsumata and Yamada [KY16] gave a *ring*-LWE-based IBE with an MPK of size $O(\lambda^{1/\mu})$ and a fixed, polynomial LWE modulus.

Finally, we mention that very recently, in an independent and concurrent work, Yamada [Yam17] proposes a novel, LWE-based IBE with slightly superlogarithmic $\omega(\log(\lambda))$ MPK blow-up, with *some* polynomial modulus (but which seems hard to determine precisely).

This brings us to the following open question:

Question #2: *Is there a standard-model, adaptively secure lattice-based IBE scheme that has a* completely short *public parameter – i.e. a* constant number *of $\mathbb{Z}_q^{n \times m}$ matrices in its MPK and uses a fixed, polynomial modulus?*

Digital Signatures. Digital Signatures are a standard cryptographic tool for securely authenticating digital messages and documents. In the lattice world, there is a standard transformation that converts any identity-based encryption scheme into a comparable digital signature scheme [Boy10]. We will use this connection in the natural way to obtain a signature scheme with a similarly compact, public verification key out of the identity-based encryption scheme that we explicitly construct first.

[1] Whether this assumption is realistic or not depends on context, but in many scenarios, it will be insufficient.

1.1 Our Contributions

In this work, we positively answer the two questions by constructing a FHE scheme supporting SIMD-style message packing, and an IBE scheme that is (essentially) as compact as lattice-based PKE scheme, under the standard Learning With Errors assumption. Our contributions can be summarized in the following three informal theorems.

Theorem 1.1 (FHE)**.** *Under the standard LWE assumption, there is a FHE scheme supporting SIMD-style message packing up to $O(\log q)$ messages without increasing the sizes of the public-key and ciphertext, where q is the modulus of LWE.*

Theorem 1.2 (IBE)**.** *Under the standard LWE assumption, there is an IBE scheme with adaptive security supporting identities of length n, where (1) the modulus q is a prime of size polynomial in the security parameter n, (2) ciphertexts consist of a vector in $\mathbb{Z}_q^{2m+1}$, and (3) the public key consists of* three *matrices in $\mathbb{Z}_q^{n\times m}$ and one vector in $\mathbb{Z}_q^n$.*

Additionally, following Boneh and Franklin [BF01], Agrawal et al. [ABB10] and Boyen [Boy10], we show how to map our new IBE scheme into a similarly efficient digital signature scheme (see the Section 7 for details):

Theorem 1.3 (Signature)**.** *Under the standard Short Integer Solution (SIS) assumption, there is an existentially unforgeable digital signature scheme supporting messages of length n, where (1) the modulus q is a prime of size polynomial in the security parameter n, (2) signatures consist of a vector in $\mathbb{Z}_q^{2m}$, and (3) the verification key consists of* three *matrices in $\mathbb{Z}_q^{n\times m}$.*

1.2 Related Work

In this section, we provide a detailed comparison with the first adaptively-secure IBE construction [ABB10] and several recent follow-up work [Yam16, ZCZ16, BL16, KY16, Yam17] on lattice-based IBE schemes. This work improves and thus subsumes the prior draft in [AFL16], which achieves an adaptively secure IBE scheme with a somewhat large MPK of size $\lambda/\log^2\lambda$ matrices.

We start our discussions on the following two works: first, Zhang et al. [ZCZ16] (Crypto '16) constructed an IBE scheme with polylogarithmic number of matrices for the public parameters, while their security only holds for Q-bounded adversary where the adversary can only obtain a prior-bounded Q number of private keys. Moreover, this restriction cannot be removed even by using a complexity leverage argument, e.g. setting Q to be super-polynomail, because the running time of the encryption algorithm in their scheme is at least linear in Q. In our detailed comparison presented later, we only consider schemes that allow any polynomially many key queries, and whose encryption running time is independent of the number of queries allowed. Thus, we do not include the work [ZCZ16]. Another recent work by Boyen et al. [BL16] (Asiacrypt '16) focused on reducing the security loss of the security reduction, yet their scheme does not try to optimize the size of the public parameters. Their parameters (public-key size) are roughly the same as the work of [ABB10], which we will provide a detailed comparison in Table 1.

In another recent work, Yamada [Yam16] (Eurocrypt '16) proposed an interesting primitive, namely the parameterized identity based encryption (PIBE), and the IBE construction are derived by encoding the identities using multiple independent PIBE. However, the IBE construction has to rely on a stronger LWE assumption where the modulus-to-error ratio is $\lambda^{\omega(1)}$, while all the other IBE constructions only rely on LWE with modulus-to-error ratio a fixed polynomial in the security parameter λ. In a follow-up work, Katsumata et al. [KY16] (Asiacrypt '16) built an IBE scheme based on ring-LWE, which is essentially the same as the ring analogue of the Yamada IBE [Yam16]. Their new security proof relies crucially on the underlying ring structure and require a stronger assumption, i.e. ring-LWE with a fixed polynomial approximation factor.

In a very recent work [Yam17], Yamada proposes new lattice IBEs with poly-logarithmic master public key sizes. This improvement is based on new design of partitioning functions. Of his IBE constructions, the best size of master public key he can achieve consists of $\omega(\log \lambda)$ matrices. However, the approximation factor of underlying LWE assumption is set to be some fixed polynomial $\mathsf{poly}(n)$, but was not determined explicitly in their work.

Table 1: Comparison of Adptively Secure Lattice-based IBE Scheme

Schemes	# of $\mathbb{Z}_q^{n\times m}$ mat. in $\lvert\mathsf{mpk}\rvert$	# of $\mathbb{Z}_q^m$ vec. in $\lvert\mathsf{ct}\rvert$	LWE param $1/\alpha$	Reduction Loss	Remarks
[CHKP10][1]	$O(\lambda)$	$O(\lambda)$	$\tilde{O}(n^{1.5})$	$O(\epsilon^{\nu+1}/Q^{\nu})$	
[ABB10]	$O(\lambda)$	$O(\lambda)$	$\tilde{O}(n^{5.5})$	$O(\epsilon^2\, qQ)$	
[Yam16]	$O(\lambda^{1/\mu})$	$O(1)$	$n^{\omega(1)}$	$O(\epsilon^{\mu+1}/nQ^{\mu})$	
[ZCZ16]	$O(\log Q)$	$O(1)$	$\tilde{O}(Q^2 \cdot n^{6.5})$	$O(\epsilon/kQ^2)$	Q-bounded
[AFL16]	$O(\lambda/\log^2 \lambda)$	$O(1)$	$\tilde{O}(n^6)$	$O(\epsilon^2/qQ)$	
[KY16][2]	$O(\lambda^{1/\mu})$	$O(1)$	$O(n^{2.5+2\mu})$	$O((\lambda^{\mu-1}\epsilon^{\mu}/Q^{\mu})^{\mu+1})$	ring-based
[BL16]	$O(\lambda)$	$O(1)$	$\mathsf{superpoly}(n)$	$O(\lambda)$	
[Yam17][3]	$\omega(\log^2 \lambda)$	$O(1)$	$O(n^{11})$	$O(\epsilon^2/n^2Q)$	
[Yam17][4]	$\omega(\log \lambda)$	$O(1)$	$\mathsf{poly}(n)$	$O(\epsilon^2/n^2Q)$	Need [BCH86, Bar89]
Ours[5]	3	$O(1)$	$\tilde{O}(n^{15.5})$	$O(\epsilon^2/\lvert Q\rvert q)$	

[1] $\nu > 1$ is the constant satisfying $c = 1 - 2^{-1/\nu}$, where c is the relative distance of the underlying error correcting code $C : \{0,1\}^k \to \{0,1\}^{\ell}$. We can take ν as close to 1 as one wants.

[2] μ is an arbitrary constant that typically is set to be 2 or 3.

[3] IBE construction Based on Modified Admissible Hash Function.

[4] IBE construction Based on Affine Functions.

[5] Using the optimization specified in Section 6.7 to improve the reduction loss.

We compare with adaptively secure IBE schemes under the LWE assumption in the standard model. $\lvert\mathsf{mpk}\rvert, \lvert\mathsf{ct}\rvert$ show the size of the master public keys, ciphertexts respectively. To measure the space efficiency, we count the number of basic components. Q and ϵ denote the number of key extraction queries and the advantage, respectively. $\mathsf{poly}(n)$ (resp. $\mathsf{superpoly}(n)$) represents fixed but large polynomial (super-polynomial) that does not depend Q and ϵ. To measure the reduction cost, we show the advantage of the LWE algorithm constructed from the adversary against the corresponding IBE scheme. To be fair, we calculate the reduction cost by employing the technique of Bellare and Ristenpart [BR09] for all schemes.

Roadmap. We present an overview of our technical approach in Section 2.3 and the necessary preliminary background in Appendix A. We recall the extended gadget matrix and its (modified) Gaussian sampling algorithm in Section 3. Our vector encoding scheme and its supported operations are described in Section 4. The applications of vector encoding scheme in FHE, IBE and signature scenarios are presented in Section 5, 6.5 and 7 respectively.

Notations. Let λ be the security parameter, and let PPT denote probabilistic polynomial time. We use bold uppercase letters to denote matrices $\mathbf{M}$, and bold lowercase letters to denote vectors $\boldsymbol{v}$. We write $\widetilde{\mathbf{M}}$ to denote the Gram-Schmidt orthogonalization of $\mathbf{M}$. We write $[n]$ to denote the set $\{1, ..., n\}$, and $|\boldsymbol{t}|$ to denote the number of bits in the string $\boldsymbol{t}$. We denote the i-th bit $\boldsymbol{s}$ by $\boldsymbol{s}[i]$. We use $\mathbf{I}_n$ to denote the $n \times n$

identity matrix and $\mathbf{0}_n$ to for $n \times n$ zero matrix. We say a function $\mathsf{negl}(\cdot) : \mathbb{N} \to (0,1)$ is negligible, if for every constant $c \in \mathbb{N}$, $\mathsf{negl}(n) < n^{-c}$ for sufficiently large n.

2 Overview of Our Techniques

In this work, we design a new lattice encoding structure, and then show its applications to the settings of FHE, IBE, and Signatures. Our new encoding uses the extended gadget defined in the prior draft of this work [AFL16]. To demonstrate the techniques, we first recall the gadget matrix defined by Micciancio and Peikert [MP12]: let $\mathbf{G} = \boldsymbol{g} \otimes \mathbf{I}_n$ where $\boldsymbol{g} = (1, 2, 4, ..., 2^{\lfloor \log q \rfloor})$. Micciancio and Peikert [MP12] developed a $\mathbf{G}$-trapdoor technique that given a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, a *short* matrix $\mathbf{R} \in \mathbb{Z}_q^{m \times m}$, and an invertible tag matrix $\mathbf{H} \in \mathbb{Z}_q^{n \times n}$, one can efficiently generate samples from the Gaussian distribution $\mathcal{D}_{\Lambda_q^{\boldsymbol{u}}(\mathbf{F}),s}$ where $\mathbf{F} := (\mathbf{A}|\mathbf{AR} + \mathbf{HG})$. This technique can be used to improve the analysis of the SampleRight algorithm used in the IBE/Signatures of the work [ABB10, Boy10], and their follow up work.

In a later work, Alperin and Peikert [AP14] simplified the FHE construction by Gentry, Sahai, and Waters [GSW13] using the $\mathbf{G}$ notion. In particular, an encryption of x can be expressed as $\mathbf{C}_x = \mathbf{AR} + x\mathbf{G}$ for some *short* $\mathbf{R}$ (similar to the syntax above). Alperin and Peikert [AP14] further showed how to compute an encryption of $x + y$, $x \cdot y$ given $\mathbf{C}_x$ and $\mathbf{C}_y$ in a surprisingly simple way: $\mathbf{C}_{x+y} = \mathbf{C}_x + \mathbf{C}_y$, and $\mathbf{C}_{xy} = \mathbf{C}_x \cdot \mathbf{G}^{-1}(\mathbf{C}_y)$. Since the $\mathbf{G}^{-1}$ operation produces a small-norm matrix, the ciphertexts of $\mathbf{C}_{x+y}$ and $\mathbf{C}_{xy}$ can be expressed as $\mathbf{AR}' + (x+y)\mathbf{G}$, and $\mathbf{AR}'' + xy\mathbf{G}$ respectively, for some small $\mathbf{R}'$ and $\mathbf{R}''$.

The prior draft of this work [AFL16] and a series of concurrent work [Yam16, KY16] have observed that the GSW structure above can be applied to the IBE setting, which was implicitly used in the work [ABB10]. At a high level, a central step of designing lattice-based scheme in these works [Yam16, AFL16, KY16] is to design a hash function (with "partitioning" properties) that can be homomorphically evaluated. To achieve a shorter key size, the prior work [AFL16] introduces a packed-GSW structure that allows to shrink the number of public matrices. The essential technique uses a new GSW encoding structure with an extended gadget matrix $\mathbf{G}_{n,\ell,m} = \boldsymbol{g}_\ell \otimes \mathbf{I}_n$ where $\boldsymbol{g} = (1, \ell, \ell^2, ..., \ell^{\lfloor \log_\ell q \rfloor})$ for some $\ell > 2$. Using the extended matrix, the prior work proposed a new encoding structure $\mathbf{H} = \left[\mathbf{H}_1 | \cdots | \mathbf{H}_d\right] \in \mathbb{Z}_q^{n \times dn}$ for matrices $\mathbf{H}_i$'s. In the application [AFL16], a hash function is embedded into these matrices. Then the prior draft defined a packed-GSW ciphertext as $\mathbf{B} = \mathbf{AR} + \mathbf{HG}_{dn,\ell,m}$. Using this packed structure, one can homomorphically compute a hash function applied to an input embedded in $(\mathbf{X}_1, \ldots, \mathbf{X}_d)$ as follows:

$$\mathbf{B}\mathbf{G}^{-1}_{dn,\ell,m}\left(\begin{bmatrix} \mathbf{X}_1 \\ \mathbf{X}_2 \\ \vdots \\ \mathbf{X}_d \end{bmatrix} \cdot \mathbf{G}\right) = \mathbf{AR}' + \left(\sum_{i \in d} \mathbf{H}_i \mathbf{X}_i\right)\mathbf{G}, \text{ for some small } \mathbf{R}'.$$

The prior draft [AFL16] showed this packed structure can be used to design IBE with smaller public-key size by a factor of $\log^2 n$.

However, the encoding method above can only support evaluation of an affine function. After the computation, the resulting ciphertext becomes $\mathbf{AR}' + \left(\sum_{i \in d} \mathbf{H}_i \mathbf{X}_i\right)\mathbf{G}$. We are not sure how to further compute on ciphertexts with $\mathbf{AR}' + \mathbf{MG}$ for $\mathbf{M} \neq \alpha \mathbf{I}_n$. (Note: a prior work [HAO15] needs to use a circular assumption in order to allow further computation on the structure $\mathbf{AR}' + \mathbf{MG}$ for $\mathbf{M} \neq \alpha \mathbf{I}_n$. In this paper, we get rid of the dependency on the assumption.) This is the main reason why the technique in our prior draft [AFL16] is not compatible with the designs in another series of work [Yam16, KY16], which required computation on non-linear functions, such as low-degree polynomials or low-depth circuits.

2.1 New Lattice Encoding Method

To tackle the challenge, this work introduces an even simpler structure that allows a significantly richer homomorphic operations. Our encoding can be used to achieve a packed FHE scheme that allows some SIMD operations and can be used to improve all the prior IBE schemes in the series [Yam16, AFL16, ZCZ16, KY16], and signatures [AS15]. Before demonstrating how to improve these applications, we first overview our new encoding structure.

Let $\boldsymbol{v} = (v_1, \ldots, v_d) \in \mathbb{Z}_q^d$ be a vector. To encode the vector, we design the structure $\mathsf{encode}(\boldsymbol{v}) = \mathbf{E}_{\boldsymbol{v}} = \left[v_1\mathbf{I}_n | \cdots | v_d\mathbf{I}_n\right] \cdot \mathbf{G}_{dn,\ell,m}$. This supports vector operations such as addition and scalar multiplication in a natural way. Let matrix $\mathbf{E}_{\boldsymbol{v}_1}$ and $\mathbf{E}_{\boldsymbol{v}_2}$ be the encoding of vector $\boldsymbol{v}_1, \boldsymbol{v}_2$. Apparently, $\mathbf{E}_{\boldsymbol{v}_1} + \mathbf{E}_{\boldsymbol{v}_2}$ is the encoding of vector $\boldsymbol{v}_1 + \boldsymbol{v}_2$. For scalar multiplication, i.e. $a \cdot \boldsymbol{v}$, where $a \in \mathbb{Z}_q$, we compute

$$\mathbf{E}_{\boldsymbol{v}} \cdot \mathbf{G}_{dn,\ell,m}^{-1}(\mathbf{E}_a) = \mathbf{E}_{\boldsymbol{v}} \cdot \mathbf{G}_{dn,\ell,m}^{-1}\left(a\mathbf{G}_{dn,\ell,m}\right) = \left[av_1\mathbf{I}_n | \cdots | av_d\mathbf{I}_n\right]\mathbf{G}_{dn,\ell,m} = \mathsf{encode}(a\boldsymbol{v}).$$

In addition to the vector operations, the encoding also supports packing and unpacking! That is, given an vector encoding $\mathbf{E}_{\boldsymbol{v}}$, for every $i \in [d]$ one can unpack the i-th coordinate to extract an encoding of v_i. Similarly, given encodings of $v_1, \ldots, v_d$, one can obtain a packed vector encoding $\mathbf{E}_{\boldsymbol{v}}$. The operations are quite simple and we refer the readers to Section 4.1 for details. The encoding structure is compatible with the GSW ciphertexts: $\mathbf{C} = \mathbf{AR} + \mathbf{E}_{\boldsymbol{v}}$. By the packing and unpacking methods, we can perform homomorphic operations not only for vector operations, but also the component-wise operations. This supports a significantly richer class of homomorphic evaluations. From here, it is not hard to derive a packed GSW scheme without using a circular assumption.

Further optimization over packed bits. For many applications, we are performing bit operations over ciphertexts, such as evaluating a low-depth boolean circuit (e.g. integer multiplications). Using the above packing methods, we can pack d bits into one vector encoding by setting $\boldsymbol{v} \in \{0,1\}^d$. If we use a polynomial modulus $q = \mathsf{poly}(n)$, then we can set $d = O(\log n)$, which means we can pack $O(\log n)$ bits in a GSW ciphertext. In our application of IBE, however, we need to be able to pack $\omega(\log n)$ bits in a single ciphertext in order to get a scheme that has a constant number of matrices in the public key. To achieve this, we need to further optimize the packing structure.

To tackle this challenge, we observe that actually an encoding $x\mathbf{G}$ can pack ν bits if ξ belongs to a larger space $[\xi]$ for $\xi = 2^\nu$, instead just $\{0,1\}$. If ξ is not too large yet super-constant, then we can afford to do a homomorphic equality test functionality $\mathsf{eq}_c(x) = \begin{cases} 1 & \text{if } x = c \\ 0 & \text{otherwise} \end{cases}$ by homomorphically evaluating the Lagrange polynomial. Then by using the equality test functionality, we can design a method that given $\mathbf{E}_x$ outputs $\mathbf{E}_{x_i}$, where x_i is the i-th bit of x for $i \in [\nu]$.

Using the above two structure, we can design a recursive packing structure for bits: let $\boldsymbol{v} \in [\xi]^d$, and

$$\mathsf{encode}(\boldsymbol{v}) = \mathbf{E}_{\boldsymbol{v}} = \left[v_1\mathbf{I}_n | \cdots | v_d\mathbf{I}_n\right] \cdot \mathbf{G}_{dn,\ell,m}.$$

From the encoding, we can unpack $\mathbf{E}_{v_i}$ first and then extract the j-th bit of v_i to obtain an encoding of the bit v_{ij}, i.e. $\mathbf{E}_{v_{ij}}$. Putting things together, we are allowed to pack $\omega(\log n)$ bits in a GSW ciphertext for polynomial modulus, i.e. $q = \mathsf{poly}(n)$. Note that if we can pack more bits by using a super-polynomial modulus, yet at the cost of a stronger underlying computational assumption (LWE with super-poly q).

2.2 Application to Fully Homomorphic Encryption

Our new encoding can be easily used to design packed GSW FHE schemes as we mentioned above. The ciphertext has the structure $\mathbf{AR} + \mathbf{E}_{\boldsymbol{v}}$ encrypting a $\boldsymbol{v}$. Then naturally, we can apply the SIMD operations

over vector space such as vector addition and scalar multiplication. Moreover, we can use the packing and unpacking method to extract each single element, so that the homomorphic operation supports element-wise and cross-element-wise computation as well. Our method improves over the prior packed GSW structure [HAO15] as we do not rely on a circular assumption as required in their work. On the other hand, the prior scheme [HAO15] is able to pack more plaintexts (i.e. $r \times r$ for some $r = \mathsf{poly}(n)$) for any modulus q at the cost of blowing up the public key and the ciphertexts, i.e. their sizes grow explicitly with r. Our scheme our scheme can pack $\omega(\log n)$ bits for $q = \mathsf{poly}(n)$, and require super-polynomial q if we want to pack more, but our public key and ciphertext remain the same size. Nevertheless, the $\omega(\log n)$ parameter for $q = \mathsf{poly}(n)$ is sufficient for us to derive an IBE scheme with 3 matrices in the public parameter. This is a significant implication of the new encoding structure.

2.3 Application to Identity-based Encryption and Signatures

Our high-level approach to compact identity-based encryption from LWE begins by revisiting the lattice-mixing and vanishing trapdoor techniques of Boyen [Boy10] and their use in Agrawal et al. [ABB10]. Our prior draft [AFL16] and concurrent work [Yam16] have developed a new conceptual view of the public parameters of the prior schemes [ABB10, Boy10]: Each matrix is (potentially) a fully homomorphic ciphertext, modeled after the Gentry-Sahai-Waters FHE scheme [GSW13], and our main goal will be to allow senders to homomorphically evaluate a *totally hidden* hash function $H_{\boldsymbol{h}}$ on their chosen identities during encryption. While prior work [AFL16] defined requirements of the hash function, such as "admissible hash functions" and "abort-resistant hash functions", our prior draft [AFL16] has observed that actually we only need *(almost) pairwise independent hash functions* plus the *random isolation* technique of Valiant and Vazirani [VV85]. We use the freedom afforded by this insight to obtain a conceptually simpler design.

The Agrawal-Boneh-Boyen IBE. We first review the construction and proof techniques of [ABB10]. Lattices in this type of system are built from "left" and "right" (sub)lattices, denoted $\mathbf{A}$ and $\mathbf{B}$ respectively. Each of these two systems are associated with a distinct trapdoor. As was the case in [ABB10], the *left trapdoor* $\mathbf{T}_{\mathbf{A}}$ serves as the true master secret msk of the real system. This left trapdoor is a "complete" trapdoor, which enables generating secret keys $\mathsf{sk}_{\boldsymbol{x}}$ for *every* string $\boldsymbol{x}$ allowed by the system. In contrast, the *right trapdoor* $\mathbf{T}_{\mathbf{B}}$ is a "partially faulty" trapdoor used only in the security proof, which enables generating secret keys $\mathsf{sk}_{\boldsymbol{x}}$ for every string $\boldsymbol{x}$ except some adaptively chosen challenge identity $\boldsymbol{x}^*$.

To encode identities $\boldsymbol{x} = (x_1, ..., x_n)$ according to [ABB10], the sender first constructs an identity-specific lattice, called the *target lattice* for $\boldsymbol{x}$,

$$\left[\mathbf{A} \mid \mathbf{Y}\right] = \left[\mathbf{A} \;\middle|\; \sum_{i=1}^{n} (-1)^{x_i} \mathbf{B}_i\right]$$

by "mixing" a *long* public key $(\mathbf{A}, \mathbf{B}_1, ..., \mathbf{B}_n)$. That is, if the i-th bit of the identity x_i is 0, the sender adds $\mathbf{B}_i$; otherwise, the sender subtracts $\mathbf{B}_i$. (Encryption then proceeds according to usual Dual Regev PKE [GPV08] using "this" target lattice as the PK.)

In the security proof, a hash function $H_{\boldsymbol{h}}$ is embedded in the computation by using the the Leftover Hash Lemma to replace each $\mathbf{B}_i$ with another matrix $\mathbf{A}\mathbf{R}_i + h_i\mathbf{B}$ for "short" $\mathbf{R}_i$, random "polluter" $\mathbf{B}$, and hash key $\boldsymbol{h} = (h_1, ..., h_n)$. The identity encoding mechanism and hash are jointly designed so that the target lattice for identity $\boldsymbol{x}$, i.e. $\left[\mathbf{A} \middle| \sum (-1)^{x_i} \mathbf{B}_i\right]$, becomes

$$\left[\mathbf{A} \mid \mathbf{Y}_{\mathsf{proof}}\right] = \left[\mathbf{A} \;\middle|\; \left(\sum_{i=1}^{n} (-1)^{x_i} \mathbf{A}\mathbf{R}_i\right) + H_{\boldsymbol{h}}(\boldsymbol{x})\mathbf{B}\right]$$

in the proof of security.

Intuitively, for challenge identity $\boldsymbol{x}^*$ and adaptively queried identities $\{\boldsymbol{x}_1, ..., \boldsymbol{x}_Q\}$, the security reduction will proceed whenever $H_{\boldsymbol{h}}(\boldsymbol{x}^*) = 0$ and for $i \in [Q]$, $H_{\boldsymbol{h}}(\boldsymbol{x}_i) \neq 0$. This is due to the fact that the matrix $\mathbf{B}$ *survives* in the target lattices for all of $\{\boldsymbol{x}_1, ..., \boldsymbol{x}_Q\}$, but the matrix $\mathbf{B}$ *vanishes* on the target lattice for the challenge identity $\boldsymbol{x}^*$, and therefore, the security reduction can $\mathsf{SampleRight}$ with $\mathbf{T}_{\mathbf{B}}$ for every identity in the adversary's view, except the single challenge point $\boldsymbol{x}^*$.

Later Improvements. The ABB approach was refined and improved in several ways. Micciancio and Peikert [MP12] developed the $\mathbf{G}$-trapdoor technique and then replaced the above $\mathbf{B}$ with $\mathbf{G}$ to get a tighter analysis (with smaller parameters). Later on, a series of work [Yam16, AFL16, ZCZ16, KY16, Yam17] observed that actually the structure of $\mathbf{AR} + x\mathbf{G}$ allows homomorphic evaluations, so we can actually design better hash function using less number of matrices. In this way, we can design a public homomorphic evaluation procedure $\mathsf{PubEval}$ and set $\mathbf{Y} = \mathsf{PubEval}(\mathbf{B}_1, \ldots, \mathbf{B}_t)$. In the security proof, $\mathbf{Y}_{\mathsf{proof}}$ would become $\mathbf{AR}^* + H_{\boldsymbol{h}}(\boldsymbol{x})\mathbf{G}$ for some bounded $||\mathbf{R}^*||$, so the above trapdoor vanishing technique can be applied. To achieve better efficiency, the series of work [Yam16, AFL16, ZCZ16, KY16, Yam17] designed hash functions with shorter "keys" and therefore smaller number of matrices, i.e. smaller t, using low-degree polynomials or integer multiplications plus modulo reduction. On the other hand, our prior draft considered a packed GSW structure, yet the prior structure can only support affine operations, so the two approaches cannot be combined to further improve the parameters. This work develops a new encoding (as we discussed in the prior subsection) so that we can achieve best of the both. Next we highlight the insights.

Our New Insights. Using our packing/unpacking structure gives a simple improvement over all previous schemes [Yam16, KY16, Yam17] that follows the design blueprint as stated above. At a high level, we can start with a single matrix $\mathbf{B}$ and then unpack many matrices $\mathbf{B}_1, \ldots, \mathbf{B}_n$. Then we apply the public evaluation to obtain the matrix $\mathbf{Y} = \mathsf{PubEval}(\mathbf{B}_1, \ldots, \mathbf{B}_n)$. This can reduce the number of matrices by a factor of $\omega(\log n)$, and in particular, we can simply combine a construction of Yamada [Yam17] and our new encoding to get an IBE scheme that only needs a constant number of matrices in the public parameter. However, the hash function (more efficient one) in the work of Yamada [Yam17] does not have an "explicit" computation procedure, so it is not easy to determine the concrete parameters needed in the scheme. The procedure needs to first transform the integer multiplication and modulo reduction into a boolean circuit (in NC1) using the method of a classic result [BCH86], and then transform the circuit into a branching program using the Barrington Theorem [Bar89]. Finally they can apply the computation of [BV15] to obtain a "norm-bounded" computation that does not blow up $||\mathbf{R}^*||$ by too much. This zigzag route is still unsatisfactory as the process might introduce extra overhead and complications, and it is not straight-forward to determine the modulus q we need: we know there exists a polynomial modulus q such that the scheme works, but it is hard to determine what that is.

To tackle this challenge, we revisit the trapdoor vanishing paradigm and design a hash function that can be explicitly computed homomorphically without using these transformations. Therefore, the concrete polynomial modulus q can be calculated in a straight-forward way. More specifically, we first start from an insight from our prior draft [AFL16] that actually we only need *(almost) pairwise independent hash functions* plus the *random isolation* technique of Valiant and Vazirani [VV85]. Therefore, the task is reduced to designing a "short-key" almost pairwise independent hash function, so we can embedded the hash key to the public matrixces $\mathbf{B}_i$'s. In particular, we design a new hash function in the following way.

We map an ID $\boldsymbol{a} \in \{0,1\}^n$ as a degree $n-1$ polynomial in $\mathbb{Z}_q[x]$ with the natural coefficient embedding (i.e. the coefficient of x^{i-1} is the i-th bit of the vector $\boldsymbol{a}$), and let $f_{\boldsymbol{a}}$ denote the polynomial. Then we consider the following hash function $h_{z,\alpha,\beta}(\boldsymbol{a}) = \alpha \cdot f_{\boldsymbol{a}}(z) + \beta$, where $z \in [n^2]$, $\alpha \in \mathbb{Z}_q$ and $\beta \in \mathbb{Z}_q$. We can prove that actually this hash function is an almost pairwise independent hash function from $\{0,1\}^n$ to $\mathbb{Z}_q$. However, if we take a closer look at the isolation technique of Valiant and Vazirani [VV85], it requires

that the range of the hash function has roughly the same size of $|Q|$, which is not true in this design. In fact, this is the reason why some of the prior work [ABB10] requires $q > 2Q$. See further details in Section 6.1.

While prior work [KY16, Yam17] developed different techniques to handle this issue, this work uses an even conceptually simpler approach – *parallel repetition*. That is, we consider a hash function

$$H_{\boldsymbol{z},\boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{a}) = [h_{z_1,\alpha_1,\beta_1}(\boldsymbol{a}), h_{z_2,\alpha_2,\beta_2}(\boldsymbol{a}), \ldots, h_{z_d,\alpha_d,\beta_d}(\boldsymbol{a})].$$

We can easily show that the parallel repeated function H remains almost pairwise independent, and we can set d such that the size of the range, i.e. q^d, approximates $|Q|$. Finally, we just need to compute $\mathbf{AR} + \mathsf{encode}(H(\boldsymbol{a}))\mathbf{G}_{dn,\ell,m}$ for the appropriate parameter d.

The overall procedures has the following structure: first we unpack $\mathbf{B_1}, \ldots, \mathbf{B}_n$ from a given matrix $\mathbf{B}$. Then we homomorphically compute the parallel repeated hash function H coordinate-by-coordinate to obtain $(\mathbf{B}_1^*, \ldots, \mathbf{B}_d^*)$. Finally, we pack these matrices together and obtain $\mathbf{B}^*$, which will be equal to $\mathbf{AR} + \mathsf{encode}(H(\boldsymbol{a}))\mathbf{G}_{dn,\ell,m}$. Finally, we observe that the Gaussian sampling algorithm in the work [MP12] carries directly to the extended matrix $\mathbf{G}_{dn,\ell,m}$. That is, if we have a short matrix $\mathbf{R}^*$ and a full-ranked (column) tag $\mathbf{H}' \in \mathbb{Z}_q^{n \times dn}$, then the original Gaussian sampling algorithm in the work [MP12] can be applied directly generate Gaussian samples from the Gaussian distribution $\mathcal{D}_{\Lambda_q^{\boldsymbol{u}}(\mathbf{F}),s}$ where $\mathbf{F} := (\mathbf{A}|\mathbf{AR}' + \mathbf{H}'\mathbf{G}_{dn,\ell,m})$. This suffices to implement the $\mathsf{SampleRight}$ algorithm used by the trapdoor vanishing framework as we discussed above.

Improving Signature Schemes. Boyen [Boy10] presented a signature scheme that uses a similar structure of the IBE scheme [ABB10]. Our improvement of the IBE scheme can be easily carried to the setting of signature schemes in the same way. We present the construction in Section 7.

3 Gadget Matrices and Generalized Gaussian Sampling

We first recall the gadget matrix [MP12, AP14], and the extended gadget matrix technique appeared in [AFL16], that are important to our construction.

Definition 3.1. *Let $m = n \cdot \lceil \log q \rceil$, and define the gadget matrix*

$$\mathbf{G}_{n,2,m} = \boldsymbol{g} \otimes \mathbf{I}_n \in \mathbb{Z}_q^{n \times m}$$

where vector $\boldsymbol{g} = (1, 2, 4, ..., 2^{\lfloor \log q \rfloor}) \in \mathbb{Z}_q^{\lceil \log q \rceil}$. We will also refer to this gadget matrix as "powers-of-two" matrix. We define the inverse function $\mathbf{G}_{n,2,m}^{-1} : \mathbb{Z}_q^{n \times m} \to \{0,1\}^{m \times m}$ which expands each entry $a \in \mathbb{Z}_q$ of the input matrix into a column of size $\lceil \log q \rceil$ consisting of the bits of binary representations. We have the property that for any matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, it holds that $\mathbf{G}_{n,2,m} \cdot \mathbf{G}_{n,2,m}^{-1}(\mathbf{A}) = \mathbf{A}$.

As mentioned by [MP12] and explicitly described in [AFL16], the results for $\mathbf{G}_{n,2,m}$ and its trapdoor can be extended to other integer powers or mixed-integer products. In this direction, we give a generalized notation for gadget matrices as follows:

For any modulus $q \geq 2$, for integer base $2 \leq \ell \leq q$, let $\boldsymbol{g}_\ell^\mathsf{T} := \left[1, \ell, \ell^2, ..., \ell^{k_\ell - 1}\right] \in \mathbb{Z}_q^{1 \times k_b}$ for $k_\ell = \lceil \log_\ell q \rceil$. (Note that the typical base-2 $\boldsymbol{g}^\mathsf{T}$ is $\boldsymbol{g}_2^\mathsf{T}$.) For row dimension n and ℓ as before, we let $\mathbf{G}_{n,\ell,nk_\ell} = \boldsymbol{g}_\ell^\mathsf{T} \otimes \mathbf{I}_n \in \mathbb{Z}_q^{n \times nk_\ell}$. The public trapdoor basis $\mathbf{T}_{\mathbf{G}_{n,\ell,nk_\ell}}$ is given analogously. Similar to the above padding argument, $\mathbf{G}_{n,\ell,nk_\ell} \in \mathbb{Z}_q^{n \times nk_\ell}$ can be padded into a matrix $\mathbf{G}_{n,\ell,m} \in \mathbb{Z}_q^{n \times m}$ for $m \geq nk_\ell$ without increasing the norm of $\widetilde{\mathbf{T}_{\mathbf{G}_{n,\ell,m}}}$ from that of $\widetilde{\mathbf{T}_{\mathbf{G}_{n,\ell,nk_\ell}}}$.

Following [Xag13] and [AP14], we now introduce a related function — the Batch Change-of-Base function $\mathbf{G}_{n',\ell',m'}^{-1}(\cdot)$ — as follows:

For any modulus $q \geq 2$, and for any integer base $2 \leq \ell' \leq q$, let integer $k_{\ell'} := \lceil \log_{\ell'}(q) \rceil$. For any integers $n' \geq 2$ and $m' \geq n' k_{\ell'}$ the function $\mathbf{G}^{-1}_{n',\ell',m'}(\cdot)$ takes as input a matrix from $\mathbb{Z}_q^{n' \times m'}$, first computes a matrix in $\{0, 1, ..., \ell' - 1\}^{n' \log_{\ell'}(q) \times m'}$ using the typical $\mathbf{G}^{-1}$ procedure (except with base-ℓ' output), then pads with rows of zeroes as needed to form a matrix in $\{0, 1, ..., \ell' - 1\}^{m' \times m'}$. For example, the typical base-2 $\mathbf{G}^{-1} = \mathbf{G}^{-1}_{n,2,m}$ takes $\mathbb{Z}_q^{n \times m}$ to $\{0,1\}^{m \times m}$ as expected.

Gaussian sampling using gadget matrix. Here we tweak the algorithm proposed in [MP12] of using gadget matrix $\mathbf{G}$ to sample from a discrete Gaussian over a desired coset of $\Lambda^{\perp}(\mathbf{A})$. We first recall the definition of generalized $\mathbf{G}_{dn,\ell,m}$-trapdoor in [MP12] as follows:

Definition 3.2 (Generalized $\mathbf{G}$-trapdoor)**.** *Let $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{G}_{dn,\ell,m} \in \mathbb{Z}_q^{dn \times m}$ be matrices with $m \geq n$. A $\mathbf{G}_{dn,\ell,m}$-trapdoor for $\mathbf{A}$ is a matrix $\mathbf{R} \in \mathbb{Z}^{m \times m}$ such that*

$$\mathbf{A} \begin{bmatrix} \mathbf{R} \\ \mathbf{I}_m \end{bmatrix} = \mathbf{H}\mathbf{G}_{dn,\ell,m}$$

for some full column rank matrix $\mathbf{H} \in \mathbb{Z}_q^{n \times dn}$. We refer to $\mathbf{H}$ as the tag of the trapdoor.

In the above definition, we only need the tag matrix $\mathbf{H}$ to be full column rank, instead of being invertible as required in [MP12]. The precise goal of sampling is, given a $\mathbf{G}_{dn,\ell,m}$-trapdoor $\mathbf{R}$ (with tag $\mathbf{H}$) for matrix $\mathbf{A}$ and a syndrome $u \in \mathbb{Z}_q^n$, to sample from the spherical discrete Gaussian distribution $D_{\Lambda^{\perp}_{\boldsymbol{u}}(\mathbf{A}),s}$ for a relatively small parameter s.

Lemma 3.3. *There is an efficient algorithm $\mathsf{SampleD}^{\mathcal{O}}(\mathbf{R}, \bar{\mathbf{A}}, \mathbf{H}, \boldsymbol{u}, s)$, where $\mathbf{R}$ is a $\mathbf{G}_{dn,\ell,m}$-trapdoor for matrix $\bar{\mathbf{A}}$ with full-rank tag matrix $\mathbf{H}$, a vector $\boldsymbol{u} \in \mathbb{Z}_q^n$ and an oracle $\mathcal{O}$ for Gaussian sampling over a desired coset $\Lambda^{\boldsymbol{v}}_q(\mathbf{G}_{dn,\ell,m})$. It will output a vector drawn from a distribution within negligible statistical distance of $\mathcal{D}_{\Lambda^{\boldsymbol{u}}(\mathbf{A}),s}$, where $\mathbf{A} = [\bar{\mathbf{A}}| - \bar{\mathbf{A}}\mathbf{R} + \mathbf{H}\mathbf{G}_{dn,\ell,m}]$.*

The following description of algorithm $\mathsf{SampleD}^{\mathcal{O}}(\mathbf{R}, \bar{\mathbf{A}}, \mathbf{H}, \boldsymbol{u}, s)$ is adapted from [MP12]:

- **Input**: An oracle $\mathcal{O}$ for Gaussian sampling over a desired coset $\Lambda^{\perp}_{\boldsymbol{v}}(\mathbf{G}_{dn,\ell,m})$ with fixed parameter $r\sqrt{\Sigma_{\mathbf{G}_{dn,\ell,m}}} \geq \eta_{\epsilon}(\Lambda^{\perp}(\mathbf{G}_{dn,\ell,m}))$, matrix $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n \times m}$, trapdoor matrix $\mathbf{R} \in \mathbb{Z}^{m \times m}$, full-rank tag matrix $\mathbf{H} \in \mathbb{Z}_q^{n \times dn}$ defining $\mathbf{A} = [\bar{\mathbf{A}}| - \bar{\mathbf{A}}\mathbf{R} + \mathbf{H}\mathbf{G}_{dn,\ell,m}]$ and a syndrome $\boldsymbol{u} \in \mathbb{Z}_q^n$.
- **Offline phase**: Choose a fresh perturbation $\boldsymbol{p} \leftarrow \mathcal{D}_{\mathbb{Z}^{2m}, r\sqrt{\Sigma_{\boldsymbol{p}}}}$, where $\Sigma_{\boldsymbol{p}} = \Sigma - \left[\begin{smallmatrix} \mathbf{R} \\ \mathbf{I} \end{smallmatrix}\right] \Sigma_{\mathbf{G}} [\mathbf{R}^{\mathsf{T}} | \mathbf{I}]$. Let $\boldsymbol{p} = (\boldsymbol{p}_1, \boldsymbol{p}_2)$ and compute $\bar{\boldsymbol{w}} = \bar{\mathbf{A}}(\boldsymbol{p}_1 - \mathbf{R}\boldsymbol{p}_2)$ and $\boldsymbol{w} = \mathbf{G}_{dn,\ell,m} \cdot \boldsymbol{p}_2$.
- **Online phase**: Compute the right-inverse matrix of $\mathbf{H}$ to be $\bar{\mathbf{H}} \in \mathbb{Z}_q^{dn \times n}$, such that $\mathbf{H}\bar{\mathbf{H}} = \mathbf{I}_n$. Let $\boldsymbol{v} = \bar{\mathbf{H}}(u - \bar{\boldsymbol{w}}) - \boldsymbol{w}$ and sample $\boldsymbol{z} \leftarrow \mathcal{D}_{\Lambda^{\perp}_{\boldsymbol{v}}(\mathbf{G}_{dn,\ell,m})}$ by calling oracle $\mathcal{O}(\boldsymbol{v})$.
- **Output**: Return $\boldsymbol{x} = \boldsymbol{p} + \left[\begin{smallmatrix} \mathbf{R} \\ \mathbf{I} \end{smallmatrix}\right] \boldsymbol{z}$.

We note that in comparison to the algorithm $\mathsf{SampleD}^{\mathcal{O}}$ in [MP12], the only difference is in the online phase, we use the right-inverse matrix of $\mathbf{H}$. The correctness proof and analysis remain the same as in [MP12], thus we omit the proof here.

4 Compact Encoding for Vectors

In this section, we first present our new compact encoding for vectors, which substaintially generalizes the encoding from the prior draft of this work [AFL16]. This new idea leads to significant improvements in the settings of fully homomorphic encryption (FHE), identity-based encryption (IBE), and signature schemes described in later sections.

4.1 Encoding for Vectors in $\mathbb{Z}_q^d$

Consider the vector space $\mathbb{Z}_q^d$. For vector $\boldsymbol{v} = (v_1, ..., v_d) \in \mathbb{Z}_q^d$, we define the following encoding algorithm which maps a d-dimensional vector to an $n \times m$ matrix.

$$\mathsf{encode}(\boldsymbol{v}) = \mathbf{E}_{\boldsymbol{v}} = \left[v_1\mathbf{I}_n | \cdots | v_d\mathbf{I}_n\right] \cdot \mathbf{G}_{dn,\ell,m} \tag{1}$$

Similarly, we also define the encoding for an integer $a \in \mathbb{Z}_q$ as: $\mathsf{encode}(a) = \mathbf{E}_a = a\mathbf{G}_{n,2,m}$.

Remark. Here the encoding structure $\mathbf{E}_{\boldsymbol{v}}$ requires parameters n, d, ℓ, m, q specified in the application. For notational simplicity, we do not include these parameters as indices in the notation $\mathbf{E}_{\boldsymbol{v}}$. We assume that these parameters are known to all the following algorithms $(\mathsf{Pack}, \mathsf{Unpack}, \mathsf{RecPack}, \mathsf{RecUnpack})$. Concrete setting of these parameters should be specified for each individual application.

The above encoding supports the following two operations naturally: (1) vector space operations, and (2) packing and unpacking. Below we define these procedures.

Vector space operations. Let matrix $\mathbf{E}_{\boldsymbol{v}_1}$ and $\mathbf{E}_{\boldsymbol{v}_2}$ be the encoding of vector $\boldsymbol{v}_1, \boldsymbol{v}_2$. Apparently, $\mathbf{E}_{\boldsymbol{v}_1} + \mathbf{E}_{\boldsymbol{v}_2}$ is the encoding of vector $\boldsymbol{v}_1 + \boldsymbol{v}_2$. For scalar multiplication, i.e. $a \cdot \boldsymbol{v}_1$, where $a \in \mathbb{Z}_q$, we compute

$$\mathbf{E}_{\boldsymbol{v}} \cdot \mathbf{G}_{dn,\ell,m}^{-1}\left(a\mathbf{G}_{dn,\ell,m}\right) = \left[av_{11}\mathbf{I}_n | \cdots | av_{1d}\mathbf{I}_n\right]\mathbf{G}_{dn,\ell,m} = \mathsf{encode}(a\boldsymbol{v})$$

For computing the inner product between $\mathbf{E}_{\boldsymbol{v}_1}$ and $\boldsymbol{v}_2 = (v_{21}, ..., v_{2d})$, the procedure is as follows

$$\mathbf{E}_{\boldsymbol{v}_1} \cdot \mathbf{G}_{dn,\ell,m}^{-1}\left(\begin{bmatrix} \boldsymbol{v}_{21}\mathbf{I}_n \\ \vdots \\ \boldsymbol{v}_{2d}\mathbf{I}_n \end{bmatrix} \mathbf{G}_{n,2,m}\right) = \langle \boldsymbol{v}_1, \boldsymbol{v}_2 \rangle \cdot \mathbf{G}_{n,2,m}$$

Packing and Unpacking. Intuitively, the packing procedure transforms d encodings of integers to an encoding of a d-dimensional vector, while the unpacking procedure unpacks the vector into d encodings of integers. The formal syntax is the following: let d be a parameter implicitly included in the algorithms.

- $\mathsf{Pack}(\{\mathbf{E}_{a_i}\}_{i=1}^d)$: On input d encodings $\mathbf{E}_{a_i}$ for $a_i \in \mathbb{Z}_q$, the procedure outputs an encoding $\mathbf{E}_{\boldsymbol{a}}$ for vector $\boldsymbol{a} = (a_1, ..., a_d)$.
- $\mathsf{Unpack}(\mathbf{E}_{\boldsymbol{v}}, i)$: On input an encoding $\mathbf{E}_{\boldsymbol{v}}$ for a vector $\boldsymbol{v}$ and an index $i \leq d$, the procedure outputs an encoding $\mathbf{E}_{v_i}$ for i-th coordinate v_i.

We instantiate the above two algorithms as follows. For $i \in [d]$, define matrix $\mathbf{U}_i$ as the $dn \times n$ extended unit matrix:

$$\mathbf{U}_i^\mathsf{T} = [\mathbf{0}_n | \cdots | \mathbf{0}_n | \mathbf{I}_n | \mathbf{0}_n | \cdots | \mathbf{0}_n] \tag{2}$$

i.e. only the i-th block is the identity matrix $\mathbf{I}_n$ and other blocks are zero matrices $\mathbf{0}_n$.

- $\mathsf{Pack}(\{\mathbf{E}_{a_i}\}_{i=1}^d)$: It outputs $\sum_{i\in[d]} \mathbf{E}_{a_i} \cdot \mathbf{G}_{n,2,m}^{-1}\left(\mathbf{U}_i^\mathsf{T} \cdot \mathbf{G}_{dn,\ell,m}\right) = [a_1\mathbf{I}_n | \cdots | a_d\mathbf{I}_n] \cdot \mathbf{G}_{dn,\ell,m}$.
- $\mathsf{Unpack}(\mathbf{E}_{\boldsymbol{v}}, i)$: It outputs $\mathbf{E}_{v_i} = \mathbf{E}_{\boldsymbol{v}} \cdot \mathbf{G}_{dn,\ell,m}^{-1}(\mathbf{U}_i\mathbf{G}_{n,2,m}) = v_i\mathbf{G}_{n,2,m}$.

4.2 Further Optimization of Packing/Unpacking over Bits

For many applications, we need to perform operations over (the GSW) encoding of bits. Using the above packing methods, we can pack d bits into one vector encoding by setting $\boldsymbol{v} \in \{0,1\}^d$. In this section, we propose an optimization that can pack $d \cdot \nu$ bits into one vector encoding for some $\nu = \omega(1)$. This method is the key step leading towards a lattice-based IBE scheme with a constant number of matrices in the public key as we will demonstrate in Section 6.5.

To pack more bits, we are going to use a vector of larger integers instead of just bits, i.e. $\boldsymbol{v} = (v_1, \ldots, v_d)$ in $[\xi]^d$ for some small $\xi = 2^\nu$, not just $\{0,1\}^d$. In this way, each element v_i can further encode ν bits, and therefore in total we can pack $d\nu$ bits. Then we show an unpacking algorithm that given an integer-encoding of v_i can unpack the j-th bit of v_i for $j \in [\nu]$. We note that the "noise" (might appear in different forms/names in different settings) of the unpacking algorithm grows with ξ^ξ (see Lemma 5.4 for further references), so this gives us a constraint that ν cannot be too large. In our IBE setting we can set $\nu = \omega(1)$, and for convenience we can think of both ξ and ν as some small $\omega(1)$ for the rest of the paper.

Recursive Packing Algorithm. We first show a recursive packing algorithm. Intuitively, the recursive packing algorithm $\mathsf{RecPack}$ takes encoding $\mathbf{E}_{v_{ij}} = v_{ij}\mathbf{G}_{n,2,m}$ for $i \in [d], j \in [\nu]$, where $v_{ij} \in \{0,1\}$, as input, and outputs $\mathbf{E}_{\boldsymbol{v}}$ as encoding of $\boldsymbol{v} = (v_1, ..., v_d)$. The description of $\mathsf{RecPack}$ is as follows: let $d, \xi = 2^\nu$ be parameters implicitly included in the algorithm.

$\mathsf{RecPack}(\{\mathbf{E}_{v_{ij}}\}_{i\in[d],j\in[\nu]})$: Given the encoding $\{\mathbf{E}_{v_{ij}}\}$ for $v_{ij} \in \{0,1\}$, the recursive packing does:

1. For $i \in [d]$, compute $\mathbf{E}_{v_i} = \sum_{j=1}^{\nu} \mathbf{E}_{v_{ij}} \mathbf{G}_{n,2,m}^{-1}(2^j \mathbf{G}_{n,2,m})$.
2. Output $\mathbf{E}_{\boldsymbol{v}} = \mathsf{Pack}(\{\mathbf{E}_{\boldsymbol{v}_i}\}_{i\in[d]}) = \left[v_1\mathbf{I}_n| \cdots |v_d\mathbf{I}_n\right] \cdot \mathbf{G}_{dn,\ell,m}$, where v_i is the decimal representation of $\{v_{i1}, \ldots, v_{i\nu}\}$.

Next, we present a recursive unpacking algorithm for the above encoding. We first present a method to unpack an integer encoding $\mathbf{E}_x = x\mathbf{G}_{n,2,m}$ for $x \in \{0, 1, \ldots, \xi\}$ for $\xi = 2^\nu$. Recall that we will set $\nu = \omega(1)$ small enough (yet super-constant) so that $2^\nu = \omega(1)$ is also small enough (yet still super-constant). Then we present a natural recursive unpacking algorithm that can unpack $d\nu$ bits. Our first step relies on the equality test functionality as elaborated below.

Equality Test Algorithm. Given an integer encoding $\mathbf{E}_x = x\mathbf{G}_{n,2,m}$ for $x \in \{0, 1, \ldots, \xi\}$ for $\xi = 2^\nu$, we can homomorphically evaluate the encoding regarding the following function:

$$\mathsf{eq}_c(x) = \begin{cases} 1 & \text{if } x = c \\ 0 & \text{otherwise} \end{cases}$$

Using this equality-test functionality, we can extract each bit of x as presented below, and therefore unpack ν bits out of the encoding $\mathbf{E}_{\boldsymbol{v}}$. This gives a natural recursive unpacking method, i.e., one can first unpack d integer encodings $\mathbf{E}_{v_1}, \ldots, \mathbf{E}_{v_d}$ from a vector encoding $\mathbf{E}_{\boldsymbol{v}}$, and from each $\mathbf{E}_{v_i}$ we can further unpack ν of bit encodings.

Next we describe the algorithm $\mathsf{EqTest}(c, \mathbf{E}_x)$ that achieves the functionality $\mathsf{eq}_c(x)$. For simplicity of notation, we abbreviate the subscript of $\mathbf{G}_{n,2,m}$ as $\mathbf{G}$ below, and let z be a parameter implicitly included in the algorithm.

$\mathsf{EqTest}(c, \mathbf{E}_x)$: Given input an integer $c \in \{0, ..., \xi\}$ (note $\xi = 2^\nu$) and an integer encoding $\mathbf{E}_x$,

1. Construct the Lagrange polynomial for integer c as

$$L_c(y) = \prod_{0\le i\le \xi, i\neq c} \frac{y-i}{c-i} = \prod_{0\le i\le \xi, i\neq c} \frac{1}{c-i} \cdot \prod_{0\le i\le \xi, i\neq c} (y-i) = \ell_c^{-1} \cdot \prod_{0\le i\le \xi, i\neq c} (y-i) \quad (3)$$

   where $\ell_c = \prod_{0\le i\le \xi, i\neq c}(c-i) \in \mathbb{Z}_q$.
2. Compute the result as

$$\mathbf{E}_{L_c(x)} = \mathbf{E}_x\mathbf{G}^{-1}(\mathbf{E}_x - 1\cdot\mathbf{G})\cdots\mathbf{G}^{-1}(\mathbf{E}_x - \xi\mathbf{G})\mathbf{G}^{-1}(\ell_c^{-1}\mathbf{G})$$

It is clear that Lagrange polynomial implements the functionality as it outputs 1 if and only if $x = c$, and otherwise 0. To figure out the j-th bit of x, one can compute $\sum_{y\in S_{ij}} \mathsf{EqTest}(y, \mathbf{E}_x)$ where S_{ij} is the set of all numbers less than z such that their j-th bits are 1. If the j-th bit of x is 0, then all the equality tests will be 0, so the sum is 0. If it is 1, then there is exactly one element, i.e. x in the set S_{ij} such that $\mathsf{EqTest}(y, \mathbf{E}_x)$ outputs 1. Therefore, this correctly implements the unpacking algorithm for the integer encoding. Next we formally define the procedure of the recursive unpacking that uses idea of the integer unpacking above.

Recursive Unpacking Algorithm. Using the above EqTest method, we can implement a recursive unpacking algorithm that extracts $d\cdot\nu$ bit-encodings from a vector encoding $\mathbf{E}_{\boldsymbol{v}}$ for $\boldsymbol{v} \in [\xi]^d$. Here for each $i \in [d]$, we denote $v_i = \sum_{i=1}^{\nu} v_{ij}2^j$ (note $\nu = \log\xi$), i.e. $\{v_{ij}\}$ is the bit-decomposition of v_i. Still, we let $d, \xi = 2^\nu$ be parameters implicitly included in the algorithm.

$\mathsf{RecUnpack}(\mathbf{E}_{\boldsymbol{v}}, (i,j))$: Given the vector encoding $\mathbf{E}_{\boldsymbol{v}}$ and index (i,j), the recursive unpacking runs the following:

1. Run $\mathbf{E}_{v_i} = \mathsf{Unpack}(\mathbf{E}_{\boldsymbol{v}}, i)$.
2. Construct a set $S_{ik} = \{\boldsymbol{v}_i \le \xi | \mathsf{BD}(\boldsymbol{v}_i)_k = 1\}$, where $\mathsf{BD}(\boldsymbol{v}_i)_k$ denotes the k-th bit of bit-decomposition of integer $\boldsymbol{v}_i$.
3. Compute and output $\mathbf{E}_{v_{ij}} = \sum_{x\in S_{ij}} \mathsf{EqTest}(x, \mathbf{E}_{v_i})$.

5 Application in GSW-FHE Setting

Gentry, Sahai, and Waters [GSW13] proposed a novel homomorphic encryption scheme (GSW), whose multiplication between ciphertexts are simply matrix multiplication. Alperin and Peikert [AP14] simplified the original GSW scheme using the gadget matrix defined by the work of Micciancio and Peikert [MP12]. Throughout this paper, we refer the GSW scheme as the simplified version of Alperin and Peikert [AP14]. In this section, we present a natural application of our new encoding scheme as in Section 4.1 to the FHE setting, which supports several Single-Instruction-Multiple-Data (SIMD) operation on ciphertexts. Our framework does not rely on a circular assumption as required in the prior work of Hiromasa et al. [HAO15]. We first recall algorithms $(\mathsf{GSW.KeyGen}, \mathsf{GSW.Enc}, \mathsf{GSW.Dec})$ as follows:

- $\mathsf{GSW.KeyGen}(1^\lambda)$: On input the security parameter λ, set parameters $n = n(\lambda), m = O(n\log q)$ and choose a B-bounded error distribution $\chi(\lambda)$. Then sample a random vector $\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_q^{n-1}$ and a random matrix $\mathbf{B} \xleftarrow{\$} \mathbb{Z}_q^{(n-1)\times m}$. Output the secret key sk and public key pk as

$$\mathsf{sk} = \boldsymbol{t} = (-\boldsymbol{s}, 1), \qquad \mathsf{pk} = \mathbf{A} = \begin{bmatrix}\mathbf{B}\\ \boldsymbol{b}\end{bmatrix}$$

  where $\boldsymbol{b} = \boldsymbol{s}^\mathsf{T}\mathbf{B} + \boldsymbol{e}$ and $\boldsymbol{e} \leftarrow \chi^m$.

- $\mathsf{GSW.Enc}(\mathsf{pk}, \mu)$: Choose a short random matrix as the randomness $\mathbf{R} \xleftarrow{\$} \{0,1\}^{m\times m}$. Then output the ciphertext of message μ as
$$\mathbf{C} = \mathbf{AR} + \mu\mathbf{G}_{n,2,m}$$
- $\mathsf{GSW.Dec}(\mathsf{sk}, \mathbf{C})$: First define a vector $\boldsymbol{w} = (0, ..., 0, \lceil q/2 \rceil) \in \mathbb{Z}_q^n$, and then compute
$$v = \boldsymbol{t}\mathbf{C}\mathbf{G}_{n,2,m}^{-1}(\boldsymbol{w})$$
Output $\mu' = \lfloor \frac{v}{q/2} \rceil$ as the decrypted message.

Our new encoding method can be applied directly in the GSW scheme so that a single matrix can encrypt a vector of d integers. The compact encryption can be described as:

- $\mathsf{Comp.Enc}(\mathsf{pk}, \boldsymbol{v})$: Choose a short random matrix as the randomness $\mathbf{R} \xleftarrow{\$} \{0,1\}^{m\times m}$. Then output the ciphertext of message vector $\boldsymbol{v}$ as $\mathbf{C} = \mathbf{AR} + \mathbf{E}_{\boldsymbol{v}}$, where $\mathbf{E}_{\boldsymbol{v}} = \left[v_1\mathbf{I}_n | \cdots | v_d\mathbf{I}_n\right] \cdot \mathbf{G}_{dn,\ell,m}$. We note that parameters d, ℓ will be implicit in all encoding structures $\mathbf{E}_{\boldsymbol{v}}$ later in this section. See the next remark for how to set these parameters.

As we discussed in Section 4.1 and 4.2, the vector encoding can perform the packing/unpacking operations. These operations can be easily applied to the GSW ciphertexts. In particular, we have the following operations:

- Pack d GSW ciphertexts into one packed ciphertext of a vector, i.e. $\mathsf{Pack}(\{\mathbf{C}_i\}_{i\in[d]})$ outputs $\mathbf{C}^*$;
- Unpack a packed ciphertext into d GSW ciphertexts, i.e. $\mathsf{Unpack}(\mathbf{C}^*, i)$ outputs $\mathbf{C}_i$ for $i \in [d]$;
- Recursively pack $d \cdot \nu$ GSW bit-encryptions into one packed ciphertexts for parameter $\nu = \omega(1)$, i.e. $\mathsf{RecPack}(\{\mathbf{C}_{ij}\}_{i\in[d],j\in[\nu]})$ outputs $\mathbf{C}^*$. Here parameters $d, z = 2^\nu$ are implicitly included in both of the algorithms $\mathsf{RecPack}$ and $\mathsf{RecUnpack}$.
- Recursively unpack a packed ciphertext into $d \cdot \nu$ GSW bit-encryptions , i.e. $\mathsf{RecUnpack}(\mathbf{C}^*, (i, j))$ outputs $\mathbf{C}_{ij}$ for $i \in [d], j \in [\nu]$.

Parameter Selection: Here d, ν, ℓ are parameters depending on n, q and can be set differently according to the application. In the FHE setting, we can set $q = \mathsf{poly}(n)$, $\ell = n$, $d = O(\log n)$ and $\nu = \omega(1)$, and it is possible to set d larger if we use a super polynomial q. In our IBE setting, we can set $q = \mathsf{poly}(n)$, $\ell = n$, $d = O(\log n)$ and $\nu = \omega(1)$. This suffices to give us an IBE scheme that only contains a constant number of matrices in the public parameters. Details are presented in Section 6.5.

To bound the noise growth after homomorphic evaluating a function f, the concept of δ-expanding below is implicitly used in various constructions, such as attribute-based encryption [BGG+14, GV15] and predicate encryption [GVW15]. This definition was formalized in [Yam17] for the partitioning functions (e.g. a particular type of hash functions) used in their IBE constructions. Here, we extend the definition to generally circuits, e.g., $f : \mathcal{X}^u \to \mathcal{Y}$. Intuitively, the parameter δ is a bound of noise growth after evaluating a function f.

Definition 5.1 (δ-expanding evaluation)**.** *Deterministic algorithms* $(\mathsf{PubEval}, \mathsf{TrapEval})$ *are δ-expanding with a function (a circuit with u inputs) $f : \mathcal{X}^u \to \mathcal{Y}$ if they are efficient and satisfy the follow properties:*

- $\mathsf{PubEval}(\{\mathbf{B}_i \in \mathbb{Z}_q^{n\times m}\}_{i\in[u]}, f)$*: On input matrices $\{\mathbf{B}_i\}_{i\in[u]}$ that are GSW-encryption of $\boldsymbol{x} = \{x_i\}_{i\in[u]}$ and a function $f \in \mathcal{F}$, the public evaluation algorithm outputs $\mathbf{B}_{f(\boldsymbol{x})} \in \mathbb{Z}_q^{n\times m}$ as the result.*

- $\mathsf{TrapEval}(\boldsymbol{x} \in \mathcal{X}^u, \mathbf{A} \in \mathbb{Z}_q^{n\times m}, \{\mathbf{R}_i\}_{i\in[u]}, f)$*: the trapdoor evaluation algorithm output* $\mathbf{R}_f$*, such that*

$$\mathsf{PubEval}(\{\mathbf{A}\mathbf{R}_i + x_i\mathbf{G}\}_{i\in[u]}, f) = \mathbf{A}\mathbf{R}_f + f(\boldsymbol{x})\mathbf{G}.$$

  Furthermore, we have $||\mathbf{R}_f|| \leq \delta \cdot \max_{i\in[u]} ||\mathbf{R}_i||$.

The definition can be extended to δ-expanding with a family of functions/circuits $\mathcal{F}$. I.e., $(\mathsf{PubEval}, \mathsf{TrapEval})$ *are δ-expanding with $\mathcal{F}$ if and only if for any $f \in \mathcal{F}$, the algorithms are δ-expanding with f.*

Next, for some simple function such as addition and multiplication, we already know the following facts:

Lemma 5.2 (Facts from [GSW13, BGG$^+$14, AP14, BV15, GV15])**.** *For an integer u, let $\boldsymbol{x} = (x_1, \ldots, x_u)$ be an input vector. There exist algorithms* $(\mathsf{PubEval}, \mathsf{TrapEval})$ *such that the following holds.*

- *Let function* $\mathsf{Add}_u : \mathbb{Z}_q^u \to \mathbb{Z}_q$ *be defined as* $\mathsf{Add}_u(\boldsymbol{x}) = \sum_{i=1}^u x_i$*. The algorithms are u-expanding with* Add_u*.*
- *Let function* $\mathsf{Mult}_{\mathsf{bool}} : \{0,1\} \times \mathbb{Z}_q \to \mathbb{Z}_q$ *be defined as* $\mathsf{Mult}_{\mathsf{bool}}(b, x) = b \cdot x$*.The algorithms are m-expanding with* $\mathsf{Mult}_{\mathsf{bool}}$*.*
- *Let function* $\mathsf{Mult}_{u,p} : [0, p-1]^u \to \mathbb{Z}_q$ *(each $x_i \in [0, p-1]$ for some bounded $p \leq q$) be defined as* $\mathsf{Mult}_u(\boldsymbol{x}_i) = \prod_{i=1}^u x_i$*. The algorithms are $mu(p-1)^u$-expanding with* Mult_u *.*

We next analyze our new (recursive) packing/unpacking algorithms using the notion of δ-expanding. Before the analysis, we present a useful lemma for a norm upper bound on $||\mathbf{G}_{n,2,m}^{-1}(\mathbf{U}_i^\mathsf{T} \cdot \mathbf{G}_{dn,\ell,m})||$.

Lemma 5.3. *Let $\mathbf{U}_i$ be the $dn \times n$ extended unit matrix as defined in Equation (2) for $i \in [d]$, then we have $||\mathbf{G}_{dn,\ell,m}^{-1}(\mathbf{U}_i\mathbf{G}_{n,2,m})|| \leq \ell \log_\ell q$.*

Proof. This bound can be proved by unfolding the formula as

$$\begin{aligned}\mathbf{G}_{dn,\ell,m}^{-1}(\mathbf{U}_i\mathbf{G}_{n,2,m}) &= \mathbf{G}_{dn,\ell,m}^{-1}([\mathbf{0}_{n\times m}|\cdots|\mathbf{0}_{n\times m}|\boldsymbol{g}\otimes\mathbf{I}_n|\mathbf{0}_{n\times m}|\cdots|\mathbf{0}_{n\times m}]^\mathsf{T}) \\ &= [\mathbf{0}_m|\cdots|\mathbf{0}_m|\mathbf{X}\otimes\mathbf{I}_n|\mathbf{0}_m|\cdots|\mathbf{0}_m]^\mathsf{T} \in \mathbb{Z}_q^{m\times m}\end{aligned}$$

where $\mathbf{X} \in [\ell]^{\log_\ell q \times \log q}$. By worst-case analysis on $||\mathbf{X}||$, we have $||\mathbf{G}_{dn,\ell,m}^{-1}(\mathbf{U}_i\mathbf{G}_{n,2,m})|| \leq \ell \log_\ell q$. □

In the following we show the exact δ-expanding for algorithms $(\mathsf{Pack}, \mathsf{Unpack}, \mathsf{RecPack}, \mathsf{RecUnpack})$ respectively:

Lemma 5.4. *Let d, ℓ be the parameters of the encoding structure, and $\xi = 2^\nu$ be the additional parameter used in the recursive packing/unpacking algorithms. Then there exist deterministic algorithms* $(\mathsf{PubEval}, \mathsf{TrapEval})$ *that are $(d \log q)$-expanding with the function* Pack*, $(\ell \log_\ell q)$-expanding with the function* Unpack*, $(d\nu \log^2 q)$-expanding with the function* RecPack*, and $(m\xi^{O(\xi)}\ell \log q \log_\ell q)$-expanding with the function* RecUnpack*.*

Remark: the recursive unpacking has an expanding factor that depends on ξ^ξ. This factor can be set small, (e.g. polynomially-bounded) for $\xi = \omega(1)$ for some small $\omega(1)$ function, e.g., $\xi = \log\log n$. This also explains the constraint on how much we can pack further by the recursive packing as discussed in Section 4.2.

Proof. We calculate the exact δ-expanding for algorithms $(\mathsf{Pack}, \mathsf{Unpack}, \mathsf{RecPack}, \mathsf{RecUnpack})$ respectively as:

- $\mathsf{Pack}(\{\mathbf{C}_{a_i}\}_{i=1}^{d})$: Given d GSW ciphertexts, unfold the equation as

$$\mathbf{C}_{\boldsymbol{a}} = \sum_{i\in[d]}(\mathbf{A}\mathbf{R}_i + \mathbf{E}_{a_i}) \cdot \mathbf{G}_{n,2,m}^{-1}(\mathbf{U}_i^{\mathsf{T}} \cdot \mathbf{G}_{dn,\ell,m}) = \sum_{i\in[d]} \mathbf{A}\mathbf{R}_i\mathbf{G}_{n,2,m}^{-1}(\mathbf{U}_i^{\mathsf{T}} \cdot \mathbf{G}_{dn,\ell,m}) + \mathbf{E}_{\boldsymbol{a}}$$

  The expanding term

$$||\sum_{i\in[d]} \mathbf{R}_i\mathbf{G}_{n,2,m}^{-1}(\mathbf{U}_i^{\mathsf{T}} \cdot \mathbf{G}_{dn,\ell,m})|| \le d\max_i ||\mathbf{R}_i\mathbf{G}_{n,2,m}^{-1}(\mathbf{U}_i^{\mathsf{T}} \cdot \mathbf{G}_{dn,\ell,m})|| \le d\log q \cdot \max_i ||\mathbf{R}_i||$$

  where the first inequality is by union bound, and we obtain the second inequality by worst-case estimate on $||\mathbf{G}_{n,2,m}^{-1}(\mathbf{U}_i^{\mathsf{T}} \cdot \mathbf{G}_{dn,\ell,m})||$.

- $\mathsf{Unpack}(\mathbf{C}_{\boldsymbol{v}}, i)$: Given packed ciphertext $\mathbf{C}_{\boldsymbol{v}}$ and index i, unfold the equation as

$$\mathbf{C}_{v_i} = \mathbf{C}_{\boldsymbol{v}}\mathbf{G}_{dn,\ell,m}^{-1}(\mathbf{U}_i\mathbf{G}_{n,2,m}) = \mathbf{A}\mathbf{R}\mathbf{G}_{dn,\ell,m}^{-1}(\mathbf{U}_i\mathbf{G}_{n,2,m}) + \mathbf{E}_{v_i}$$

  The expanding term $\mathbf{R}\mathbf{G}_{dn,\ell,m}^{-1}(\mathbf{U}_i\mathbf{G}_{n,2,m})$ can be bounded by worst-case estimates on $\mathbf{G}_{dn,\ell,m}^{-1}(\mathbf{U}_i\mathbf{G}_{n,2,m})$ as shown in Lemma 5.3, i.e. $||\mathbf{R}\mathbf{G}_{dn,\ell,m}^{-1}(\mathbf{U}_i\mathbf{G}_{n,2,m})|| \le \ell\log_\ell q \cdot ||\mathbf{R}||$.

- $\mathsf{RecPack}(\{\mathbf{C}_{v_{ij}}\}_{i\in[d],j\in[\nu]})$: Given ciphertext $\mathbf{C}_{v_{ij}}$ for bits, unfold the first step as

$$\mathbf{C}_{v_i} = \sum_{j\in[\nu]} \mathbf{C}_{v_{ij}}\mathbf{G}_{n,2,m}^{-1}(2^j\mathbf{G}_{n,2,m}) = \sum_{j\in[\nu]} \mathbf{A}\mathbf{R}_{ij}\mathbf{G}_{n,2,m}^{-1}(2^j\mathbf{G}_{n,2,m}) + \mathbf{E}_{v_i}$$

  Similarly, the expanding term $\sum_{j\in[\nu]} \mathbf{R}_{ij}\mathbf{G}_{n,2,m}^{-1}(2^j\mathbf{G}_{n,2,m})$ can be bounded as

$$||\sum_{j\in[\nu]} \mathbf{R}_{ij}\mathbf{G}_{n,2,m}^{-1}(2^j\mathbf{G}_{n,2,m})|| \le \nu\max_j ||\mathbf{R}_{ij}\mathbf{G}_{n,2,m}^{-1}(2^j\mathbf{G}_{n,2,m})|| \le \nu\log q \cdot \max_j ||\mathbf{R}_{ij}||$$

  By plugin $d\log q$-expanding of algorithm Pack to the second step of $\mathsf{RecPack}$, we can get $d\nu\log^2 q$-expanding for $\mathsf{RecPack}$.

- $\mathsf{RecUnpack}(\mathbf{C}_{\boldsymbol{v}}, (i,j))$: Given a packed GSW ciphertext $\mathbf{C}_{\boldsymbol{v}}$ and index (i,j), we first show how to instantiate the algorithms $\mathsf{TrapEval}$ for the function of $\mathsf{RecUnpack}_{ij}$. We use the notation $f_{\mathsf{RecUnpack}_{ij}}$ to denote the function. For $i \in [d], j \in [\nu]$, the function $f_{\mathsf{RecUnpack}_{ij}}$ takes $\boldsymbol{v} = (v_1, ..., v_d) \in [\xi]^d$ as input and outputs v_{ij}, such that $v_i = \sum_{j=1}^{\nu} v_{ij}2^j$.

  $\mathsf{TrapEval}(\boldsymbol{v}, \mathbf{A}, \mathbf{R}, f_{\mathsf{RecUnpack}_{ij}})$: The trapdoor evaluation for recursive unpacking function $f_{\mathsf{RecUnpack}_{ij}}$ runs the following:

  1. Compute $\mathbf{R}_i = \mathbf{R} \cdot \mathbf{G}_{dn,\ell,m}^{-1}(\mathbf{U}_i\mathbf{G}_{n,2,m})$, where $\mathbf{U}_i$ is the extended unit matrix defined in Equation (2)
  2. Construct a set $S_{ik} = \{\boldsymbol{v}_i \le \xi | \mathsf{BD}(\boldsymbol{v}_i)_k = 1\}$, where $\mathsf{BD}(x)_k$ denotes the k-th bit of bit-decomposition of integer x.
  3. Compute and output

$$\mathbf{R}_{ij} = \sum_{x\in S_{ij}} (\mathbf{R}_i \prod_{k=1}^{\xi} \mathbf{G}_{n,2,m}^{-1}((x-k)\cdot\mathbf{G}_{n,2,m})\mathbf{G}_{n,2,m}^{-1}(\ell_x^{-1}\mathbf{G}_{n,2,m}))$$

  where ℓ_x is the denominator of Lagrange polynomial $L_x(y) = \prod_{0\le i\le\xi, i\ne x} \frac{y-i}{x-i}$ defined in Equation (3).

The first step is the same as Unpack algorithm, which incurs $(\ell \log_\ell q)$-expanding. We unfold the third step as

$$\begin{aligned}||\mathbf{R}_{ij}|| &= \sum_{x \in S_{ij}} ||(\mathbf{R}_i \prod_{k=1}^{\xi} \mathbf{G}_{n,2,m}^{-1}((x-k)\cdot \mathbf{G}_{n,2,m})\mathbf{G}_{n,2,m}^{-1}(\ell_x^{-1}\mathbf{G}_{n,2,m}))|| \\ &\leq \xi \max_{x \in S_{ij}} ||(\mathbf{R}_i \prod_{k=1}^{\xi} \mathbf{G}_{n,2,m}^{-1}((x-k)\cdot \mathbf{G}_{n,2,m})\mathbf{G}_{n,2,m}^{-1}(\ell_x^{-1}\mathbf{G}_{n,2,m}))|| \\ &\leq \xi \cdot m\xi(\xi-1)^{\xi} \log q ||\mathbf{R}_i|| \leq m\xi^{O(\xi)} \log q \cdot \ell \log_\ell q \cdot ||\mathbf{R}||\end{aligned}$$

where the second inequality is obtained by union bound, the third inequality is by Lemma 5.2 and the last step is by combing the $(\ell \log_\ell q)$-expanding in algorithm Unpack.

□

Additionally, we can also perform vector space operations over packed ciphertexts, such as vector addition, scalar multiplication, and inner product as described in Section 4.1. An interesting and useful setting for encryption of vectors would be encrypting polynomials. If we view the coordinates of vector $\boldsymbol{v}$ as the coefficients of a polynomial f, then we can use the method above to encode a degree-$(d-1)$ polynomial, i.e. $f(x) = \sum_{i=0}^{d-1} a_i x^i \in \mathbb{Z}_q[x]$.

6 Application in IBE Setting

In this section, we apply our new encoding in the setting of Identity-Based Encryption. In particular, we construct an IBE scheme with 3 matrices (independent of the ID length n) in the public parameter. Our presentation follows the flow. (1) We first revisit the partitioning strategy required by the trapdoor vanishing framework as discussed in the overview. We show a conceptually simpler approach that actually almost pairwise independent hash functions suffice for the framework. (2) We next present our new design of pairwise independent hash family in two steps: first we present a simple hash function as a building block while a conditional probability might be too large; then we show how to adjust the probability by using a parallel repetition. (3) We then show that our almost pairwise independent hash function can be homomorphically evaluated with δ-expanding for some polynomial δ. (Recall Definition 5.1 for δ-expanding). (4) Using the tools from (1-3), we are able to design the desired IBE and prove security.

6.1 Partitioning Strategy

We first revisit the partitioning strategy required to implement the trapdoor vanishing technique as discussed in Section 2.3. At a high level, we need to design a hash function that isolates the challenge ID from all the other ID queries. That is, given a set of ID queries Q and a challenge ID $\boldsymbol{a}^* \notin Q$, our hash function should separate them with a noticeable probability, i.e. $H(\boldsymbol{a}^*) = 0$ but $H(\boldsymbol{a}) \neq 0$ for all $\boldsymbol{a} \in Q$. While prior work [BB04] defined this type of requirements as "admissible hash functions" and "abort-resistant hash functions", our prior draft [AFL16] has observed that actually it suffices for *(almost) pairwise independent hash functions* plus the *random isolation* technique of Valiant and Vazirani [VV85]. This will conceptually simplify our design of hash functions presented in the coming section.

More specifically, recall that Unambiguous Satisfiability (USAT) is the promise problem of deciding if a given boolean formula F is unsatisfiable, or has exactly one satisfying assignment. In [VV85], Valiant and Vazirani show that a polynomial-time algorithm for USAT implies NP = RP. Their proof relies on a key technical lemma about isolations of random hash functions. The idea is that by intersecting a formula

F with sufficiently many *random* hyperplanes (each of which are affine in F), the resulting formula F' will have a unique solution x' (also satisfying F) with constant probability. In more detail, [VV85] shows that for any pairwise independent hash function family $\mathcal{H} : \{0,1\}^n \to \{0,1\}^k$ and any set $Q \subset \{0,1\}^n$ such that $2^{k-2} \leq |Q| \leq 2^{k-1}$ (i.e. the range of H approximates $|Q|$), it holds that $\mathsf{Pr}_{H\in\mathcal{H}}\left[\left|H^{-1}(\mathbf{0}) \cap Q\right| = 1\right] \geq \frac{1}{8}$.

In this work, we show a similar lemma which captures the intuition: almost pairwise independent hash functions have the isolation property as long as a conditional probability defined as below approximates $1/|Q|$

Lemma 6.1. *Let $Q \subseteq \{0,1\}^n$, A, B be integers such that $B \leq A$, $|Q| \leq \delta B$ for some $\delta \in (0,1)$, and let $\mathcal{H} : \{0,1\}^n \to \mathcal{Y}$ be an almost pairwise independent hash function family which has the following parameters:*

- $\forall \boldsymbol{a} \in \{0,1\}^n$, $\Pr_{H\leftarrow\mathcal{H}}[H(\boldsymbol{a}) = 0] = 1/A$;
- $\forall \boldsymbol{a} \neq \boldsymbol{b} \in \{0,1\}^n$, $\Pr_{H\leftarrow\mathcal{H}}[H(\boldsymbol{a}) = 0 | H(\boldsymbol{b}) = 0] \leq 1/B$.

Then for any element $\boldsymbol{a} \notin Q$, we have

$$\mathsf{Pr}_{H\in\mathcal{H}}[H(\boldsymbol{a}) = 0 \wedge H(\boldsymbol{a}') \neq 0, \forall \boldsymbol{a}' \in Q] \in \left(\frac{1-\delta}{A}, \frac{1}{A}\right).$$

Proof. For the lower bound, we can unfold the probability formula as follows:

$$\begin{aligned}
&\mathsf{Pr}_{H\in\mathcal{H}}[H(\boldsymbol{a}) = 0 \wedge H(\boldsymbol{a}') \neq 0, \forall x' \in Q] \\
=&\mathsf{Pr}_{H\in\mathcal{H}}[\forall \boldsymbol{a}' \in Q, H(\boldsymbol{a}') \neq 0 | H(\boldsymbol{a}) = 0] \cdot \mathsf{Pr}_{H\in\mathcal{H}}[H(\boldsymbol{a}) = 0] \\
=&\mathsf{Pr}_{H\in\mathcal{H}}[\forall \boldsymbol{a}' \in Q, H(\boldsymbol{a}') \neq 0 | H(\boldsymbol{a}) = 0] \cdot \frac{1}{A} \\
=&\Big(1 - \mathsf{Pr}_{H\in\mathcal{H}}[\exists \boldsymbol{a}' \in Q, H(\boldsymbol{a}') = 0 | H(\boldsymbol{a}) = 0]\Big) \cdot \frac{1}{A} \\
\geq&\Big(1 - \sum_{\boldsymbol{a}'\in Q} \mathsf{Pr}_{H\in\mathcal{H}}[H(\boldsymbol{a}') = 0 | H(\boldsymbol{a}) = 0]\Big) \cdot \frac{1}{A} = \left(1 - \frac{|Q|}{B}\right) \cdot \frac{1}{A} \geq \frac{1-\delta}{A}.
\end{aligned}$$

The first equality comes from the definition of conditional probability; the second equality comes from the first property of the hash function; the third equality comes from the complement event; the first inequality comes from a union bound; the next equality comes from the second property of the hash function; the last inequality comes from the constraint that $|Q| \leq \delta B$.

For the upper bound, we have simply obtain the following:

$$\mathsf{Pr}_{H\leftarrow\mathcal{H}}[H(\boldsymbol{a}) = 0 \wedge H(\boldsymbol{a}') \neq 0, \forall \boldsymbol{a}' \in Q] \leq \mathsf{Pr}_{H\leftarrow\mathcal{H}}[H(\boldsymbol{a}) = 0] = 1/A.$$

This completes the proof. □

6.2 Our New Partitioning Function

As argued above, now our task is to design an almost pairwise independent hash family to serve as the partitioning function in the IBE scheme. To achieve this, we consider two steps: first we define a basic hash function, yet the conditional probability (i.e. $1/B$) might not approximate an arbitrary $|Q|^{-1}$ as required by Lemma 6.1. Then we can simply use a parallel repetition to adjust the probabilities appropriately. In our IBE security proof, we can adjust the number of repetition to adjust the conditional probability to one that approximates $|Q|^{-1}$ as required by Lemma 6.1.

Step 1. We define a basic partition family $\mathcal{H} : \{0,1\}^n \to \mathbb{Z}_q$ where each function $h_{z,\alpha,\beta} \in \mathcal{H}$ is indexed by an element $z \in [2n^2]$, $\alpha \in \mathbb{Z}_q$ and $\beta \in \mathbb{Z}_q$. The function maps a boolean string $\boldsymbol{a} = (a_0, ..., a_{n-1}) \in \{0,1\}^n$ to an integer in $\mathbb{Z}_q$ as

$$h_{z,\alpha,\beta}(\boldsymbol{a}) = \alpha \cdot f_{\boldsymbol{a}}(z) + \beta, \tag{4}$$

where $f_{\boldsymbol{a}}(x) = \sum_{i=0}^{n-1} a_i x^i$ is the polynomial indexed by $\boldsymbol{a}$. We then show that this partition function has the following properties:

Lemma 6.2. *Let q be a prime and $q \geq 2n$. For a random hash function from the family $\mathcal{H}$, (i.e. we randomly choose integers $z \in [2n^2], \alpha, \beta \in \mathbb{Z}_q$), we have:*

- $\forall \boldsymbol{a} \in \{0,1\}^n$, $\mathsf{Pr}[h_{z,\alpha,\beta}(\boldsymbol{a}) = 0] = 1/q$.
- $\forall \boldsymbol{a} \neq \boldsymbol{b} \in \{0,1\}^n$, $\mathsf{Pr}[h_{z,\alpha,\beta}(\boldsymbol{b}) = 0 | h_{z,\alpha,\beta}(\boldsymbol{a}) = 0] \leq 1/n$

Proof. For the first property, we can unfold the probability formula as follows:

$$\begin{aligned}
&\mathsf{Pr}[h_{z,\alpha,\beta}(\boldsymbol{a}) = 0] \\
=&\mathsf{Pr}[h_{z,\alpha,\beta}(\boldsymbol{a}) = 0 | f_{\boldsymbol{a}}(z) = 0]\mathsf{Pr}[f_{\boldsymbol{a}}(z) = 0] + \mathsf{Pr}[h_{z,\alpha,\beta}(\boldsymbol{a}) = 0 | f_{\boldsymbol{a}}(z) \neq 0]\mathsf{Pr}[f_{\boldsymbol{a}}(z) \neq 0] \\
=&\mathsf{Pr}[\beta = 0]\mathsf{Pr}[f_{\boldsymbol{a}}(z) = 0] + \mathsf{Pr}[\beta = -\alpha f_{\boldsymbol{a}}(z)]\mathsf{Pr}[f_{\boldsymbol{a}}(z) \neq 0] \\
=&\frac{1}{q}(\mathsf{Pr}[f_{\boldsymbol{a}}(z) = 0] + \mathsf{Pr}[f_{\boldsymbol{a}}(z) \neq 0]) = \frac{1}{q}.
\end{aligned}$$

The first equality is from the law of total probability; the second equality is from a direct computation; the third equality is from the uniform distribution of β, so $\mathsf{Pr}[\beta = w] = 1/q$ for any element $w \in \mathbb{Z}_q$.

For the second property, the condition can derive:

$$h_{z,\alpha,\beta}(\boldsymbol{a}) = 0 \iff \beta = -\alpha f_{\boldsymbol{a}}(z)$$

To calculate the conditional probability, we plugin the above equation into $h_{z,\alpha,\beta}(\boldsymbol{b}) = 0$ and obtain:

$$\alpha(f_{\boldsymbol{\alpha}}(z) - f_{\boldsymbol{\beta}}(z)) = 0 \iff \alpha(f_\alpha - f_\beta)(z) = 0$$

Therefore, we can unfold the probability formula as

$$\begin{aligned}
&\mathsf{Pr}[h_{z,\alpha,\beta}(\boldsymbol{b}) = 0 | h_{z,\alpha,\beta}(\boldsymbol{a}) = 0] = \mathsf{Pr}[\alpha(f_\alpha - f_\beta)(z) = 0] \\
\leq&\mathsf{Pr}[\alpha = 0] + \mathsf{Pr}[(f_\alpha - f_\beta)(z) = 0] \leq \frac{1}{q} + \frac{n}{2n^2} \leq \frac{1}{n}
\end{aligned}$$

The first inequality is by the union bound; the second inequality is because $\deg(f_\alpha - f_\beta) \leq n$ so the polynomial has at most n roots; the third inequality is because $q \geq 2n$. This completes the proof. □

Clearly, the conditional probability in the second property is a fixed $1/n$ and thus cannot approximate $1/|Q|$ for all different polynomial-sized set $|Q|$ as required by Lemma 6.1. To tackle this issue, we consider to adjust the probabilities using a parallel repetition.

Step 2. Next, we propose a parallel repetition version of the partition function defined as Equation (4). Let $\nu \geq u$ be parameters, and we consider a family $\mathcal{H}^{(u,\nu)}$ as the u-parallel repetition of the basic hash family $\mathcal{H}$ padded with $(\nu - u)$ 0's. That is, $\mathcal{H}^{(u,\nu)} : \{0,1\}^n \to \mathbb{Z}_q^\nu$ where each hash function $h_{\boldsymbol{z},\boldsymbol{\alpha},\boldsymbol{\beta}} \in \mathcal{H}^{(u,\nu)}$ is indexed by vectors $\boldsymbol{z} \in [2n^2]^u$ and $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{Z}_q^u$. The function $h_{\boldsymbol{z},\boldsymbol{\alpha},\boldsymbol{\beta}} : \{0,1\}^n \to \mathbb{Z}_q^\nu$ is defined as

$$h_{\boldsymbol{z},\boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{a}) = (h_{z_1,\alpha_1,\beta_1}(\boldsymbol{a}), ..., h_{z_u,\alpha_u,\beta_u}(\boldsymbol{a}), 0, ..., 0). \tag{5}$$

Similarly, we can prove the following lemma.

Lemma 6.3. *For a random hash function from the family $\mathcal{H}^{(u,\nu)}$, (i.e. we randomly choose integer vectors $\boldsymbol{z} \in [1, 2n^2]^u, \boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{Z}_q^u$), we have:*

- $\forall \boldsymbol{a} \in \{0,1\}^n$, $\mathsf{Pr}[h_{\boldsymbol{z},\boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{a}) = 0] = (1/q)^u$.
- $\forall \boldsymbol{a} \neq \boldsymbol{b} \in \{0,1\}^n$, $\mathsf{Pr}[h_{\boldsymbol{z},\boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{b}) = 0 | h_{\boldsymbol{z},\boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{a}) = 0] \leq (1/n)^u$

Proof. By the independence of each coordinate of $\boldsymbol{z}, \boldsymbol{\alpha}, \boldsymbol{\beta}$, obviously $\forall \boldsymbol{a} \in \{0,1\}^n$ we can get

$$\mathsf{Pr}[h_{\boldsymbol{z},\boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{a}) = 0] = (1/q)^u,$$

Similarly, by independence we can unfold the second probability formula as

$$\mathsf{Pr}[h_{\boldsymbol{z},\boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{b}) = 0 | h_{\boldsymbol{z},\boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{a}) = 0] = \prod_{i=1}^{u} \mathsf{Pr}[h_{z_i,\alpha_i,\beta_i}(\boldsymbol{b}) = 0 | h_{z_i,\alpha_i,\beta_i}(\boldsymbol{a}) = 0] \leq \frac{1}{n^u}.$$

□

In the scheme and security proof of our IBE scheme, we are evaluating the ν-folded hash function homomorphically. In the security proof, we can have the freedom to set u appropriately such that the conditional probability approximates $1/|Q|$. Thus, we can apply the prior Lemma 6.1 to analyze the success probability of the reduction in the proof.

6.3 Evaluate the Partitioning Function, Homomorphically

In this part, we show how to homomorphically evaluate the partitioning function, both the basic and parallel repetition version. The description of $(\mathsf{PubEval}^{(\boldsymbol{a},t)}, \mathsf{TrapEval}^{(\boldsymbol{a},t)})$ for function $h'_{\boldsymbol{a}}(z, \alpha, \beta) = h_{z,\alpha,\beta}(\boldsymbol{a}) = \alpha \cdot f_{\boldsymbol{a}}(z) + \beta : \{0,1\}^n \to \mathbb{Z}_q$ is as follows, where algorithms $(\mathsf{PubEval}^{(\boldsymbol{a},t)}, \mathsf{TrapEval}^{(\boldsymbol{a},t)})$ are indexed by a pre-specified identity $\boldsymbol{a}$ and parameters $t = 2\log(2n)$. Our task is to homomorphically compute $\alpha f_{\boldsymbol{a}}(z) + \beta$.

To compute this homomorphically (with small δ expanding factor) is not straight-forward. First we observe that the identity vector $\boldsymbol{a}$ has bit-length n, and therefore the polynomial $f_{\boldsymbol{a}}$ has degree $n-1$. To compute $f_{\boldsymbol{a}}(z)$, it seems that we must be able to compute z^{n-1} homomorphically. However, to our knowledge, this computation might not be in NC1, so it is not clear how to compute it under a polynomial modulus q. Even though computing z^{n-1} can be done as a tree of integer multiplications of depth $\log n$, an integer multiplication however, might not be computed in a constant depth. Thus, a direct homomorphic computation using known techniques might not work.

To tackle this challenge, we use the fact that $\boldsymbol{a}$ is a public know ID, and one can compute in the clear $f(1), f(2), \ldots, f(2n^2)$. We know that $f(z)$ must be one of them, so the computation can be done by the summation $\sum_{i \in [2n^2]} (z? = i) f(i)$, where $(z? = i)$ outputs the bit whether z equals i. We can show that this procedure is δ-expanding for some bounded polynomial δ. Therefore, if we have the encoding of the bit-decomposition of z, then we can homomorphically compute this quantity. In what follows, we modify the idea above and show a way to homomorphically compute the desired function $\alpha f_{\boldsymbol{a}}(z) + \beta$.

In the following context, we set $t = 2\log(2n)$, let $\mathbf{B}_1, \ldots, \mathbf{B}_t$ be the matrices that encode each bit of z. We let $\mathbf{B}_{t+1}$ and $\mathbf{B}_{t+2}$ as the matrices that encode α and β, and let $h'_{\boldsymbol{a}}$ be the function described above. Now we are ready to present the procedure.

$\mathsf{PubEval}^{(\boldsymbol{a},t)}(\{\mathbf{B}_i\}_{i \in [t+2]}, h'_{\boldsymbol{a}})$: The public evaluation does:

1. For each $i \in [2n^2]$, first compute $f_{\boldsymbol{a}}(i) = \sum_{j=1}^{n} a_j i^j$ and bit-decompose $f_{\boldsymbol{a}}(i)$ to get $\{f_{\boldsymbol{a}}(i)_j\}_{j \in [\log q]}$, then compute

$$\mathbf{B}'_{\alpha f_{\boldsymbol{a}}(i)} = \sum_{j=1}^{\log q} f_{\boldsymbol{a}}(i)_j \cdot \mathbf{B}_{t+1} \mathbf{G}^{-1}_{n,2,m}(2^j \mathbf{G}_{n,2,m}).$$

2. For $i \in [2n^2], j \in [t]$, calculate

$$\mathbf{B}_{i,j} = \mathsf{PubEval}(i_j \mathbf{G}_{n,2,m}, \mathbf{B}_j, \mathsf{Mult}_{\mathsf{bool}}) + \mathsf{PubEval}((1-i_j)\mathbf{G}_{n,2,m}, \mathbf{G}_{n,2,m} - \mathbf{B}_j, \mathsf{Mult}_{\mathsf{bool}}).$$

   Then for $i \in [2n^2]$, compute $\bar{\mathbf{B}}_i = \mathsf{PubEval}(\{\mathbf{B}_{i,j}\}_{j\in[t]}, \mathsf{Mult}_t)$.
3. Compute and output $\mathbf{B}^* = \mathbf{B}_{t+2} + \sum_{i\in[2n^2]} \mathsf{PubEval}(\bar{\mathbf{B}}_i, \mathbf{B}'_{f_{\boldsymbol{a}}(i)}, \mathsf{Mult}_2)$.

$\mathsf{TrapEval}^{(\boldsymbol{a},t)}((z,\alpha,\beta), \mathbf{A}, \{\mathbf{R}_i\}_{i\in[t+2]}, h'_{\boldsymbol{a}})$: The trapdoor evaluation algorithm does:

1. Parse the matrices $\{\mathbf{R}_i\}_{i\in[t+2]}$ as encoding of (z,α,β), i.e. $\{\mathbf{R}_i\}_{i\in[t]}$ is used to encode $(z_1,...,z_t) \in \{0,1\}^t$, $\mathbf{R}_{t+1}$ is for α and $\mathbf{R}_{t+2}$ is for β, where $\{z_i\}_{i\in[t]}$ is the bit representation of z.
2. For $i \in [2n^2]$, first compute $f_{\boldsymbol{a}}(i) = \sum_{j=1}^{n} a_j i^j$ and bit-decompose $f_{\boldsymbol{a}}(i)$ to get $\{f_{\boldsymbol{a}}(i)_j\}_{j\in[\log q]}$, then compute

$$\mathbf{R}'_{\alpha f_{\boldsymbol{a}}(i)} = \sum_{j=1}^{\log q} \mathsf{TrapEval}(\mathbf{A}, (\mathbf{I}_m, f_{\boldsymbol{a}}(i)_j), (\mathbf{R}_{t+1}\mathbf{G}^{-1}_{n,2,m}(2^j\mathbf{G}_{n,2,m}), \alpha\cdot 2^j), \mathsf{Mult}_{\mathsf{bool}})$$

3. Define an equality test circuit $\mathsf{eq}_y(y^*)$ to be

$$\mathsf{eq}_y(y^*) = \begin{cases} 1 & \text{if } y^* = y \\ 0 & \text{otherwise} \end{cases}$$

   For $i \in [2n^2]$, instantiate the equality test circuit $\mathsf{eq}_i(z)$ as $z_{?i} = \prod_{j\in[t]} \mathsf{xnor}(z_j, i_j)$, calculate

$$\mathbf{R}_{z_j,i_j} = \mathsf{TrapEval}(\mathbf{A}, (\mathbf{R}_j, z_j), (\mathbf{I}_m, i_j), \mathsf{Mult}_{\mathsf{bool}}) + \mathsf{TrapEval}(\mathbf{A}, (-\mathbf{R}_j, 1-z_j), (\mathbf{I}_m, -i_j), \mathsf{Mult}_{\mathsf{bool}})$$

   as the matrix denoting $\mathsf{xnor}(z_j, i_j)$. Next, compute

$$\bar{\mathbf{R}}_i = \mathsf{TrapEval}(\{\mathsf{xnor}(z_j, i_j)\}_{i\in[t]}, \mathbf{A}, \{\mathbf{R}_{z_j,i_j}\}_{i\in[t]}, \mathsf{Mult}_t)$$

4. Compute and output

$$\mathbf{R}^* = \mathbf{R}_{t+2} + \sum_{i\in[2n^2]} \mathsf{TrapEval}(\mathbf{A}, (\bar{\mathbf{R}}_i, z_{?i}), (\mathbf{R}'_{\alpha f_{\boldsymbol{a}}(i)}, h_{i,\alpha,\beta}(\boldsymbol{a})), \mathsf{Mult}_{\mathsf{bool}})$$

Here the first step computes an encoding for $\alpha f_{\boldsymbol{a}}(i)$ for each i, i.e. matrix $\mathbf{B}'_{\alpha f_{\boldsymbol{a}}(i)}$. Then in the second step, we do an integer comparison check whether $z? = i$, stored in the matrix $\bar{\mathbf{B}}_i$. Finally we do the homomorphic summation of $\sum_{i\in[2n^2]}(z? = i)\alpha f(i) + \beta$. Next, we prove that the procedure is δ-expanding for some bounded polynomial δ.

Lemma 6.4. $(\mathsf{PubEval}^{(\boldsymbol{a},t)}, \mathsf{TrapEval}^{(\boldsymbol{a},t)})$ *defined above is* $O(n^2m^2\log q)$*-expanding for the function* $h'_{\boldsymbol{a}} : \{0,1\}^n \to \mathbb{Z}_q$ *as defined above.*

Proof. By the m-expanding property of algorithm $\mathsf{Mult}_{\mathsf{bool}}$ stated in Lemma 5.2, the matrix $||\mathbf{R}'_{\alpha f_{\boldsymbol{a}}(i)}|| \le \log^2 q \cdot ||\mathbf{R}_{t+1}||$. Similarly, we have $||\bar{\mathbf{R}}_i|| \le mt||\mathbf{R}_{z_j,i_j}|| \le 2m\log q \max_{i\in[t]} ||\mathbf{R}_i||$. Then, we can bound the norm of $\mathbf{R}^*$ as

$$\begin{aligned} ||\mathbf{R}^*| &\le 2n^2 \max_{i\in[2n^2]} \mathsf{TrapEval}(\mathbf{A}, (\mathbf{R}_{z_{?i}}, z_{?i}), (\mathbf{R}'_{\alpha f_{\boldsymbol{a}}(i)}, h_{i,\alpha,\beta}(\boldsymbol{a})), \mathsf{Mult}_{\mathsf{bool}}) \\ &\le 2n^2 \cdot 2m\log q \cdot m \cdot \max_{i\in[t]} ||\mathbf{R}_i|| = O(n^2m^2\log q) \cdot \max_{i\in[t]} ||\mathbf{R}_i||. \end{aligned}$$

□

Next we describe the algorithms $(\mathsf{PubEval}_{||}^{(\boldsymbol{a},\nu,t)}, \mathsf{TrapEval}_{||}^{(\boldsymbol{a},\nu,t)}))$ for the parallel-repetition counterpart of the partitioning function. The function is defined as

$$\boldsymbol{h}'_{\boldsymbol{a}}(\boldsymbol{z}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = (h_{z_1,\alpha_1,\beta_1}(\boldsymbol{a}), ..., h_{z_\nu,\alpha_\nu,\beta_\nu}(\boldsymbol{a})).$$

We consider that the algorithms receive two *packed* matrices $\mathbf{B}$ and $\mathbf{B}'$. Then they can (recursively) unpack the matrices into $\nu \times t$ matrices (with t matrices in a group) and run ν-repetitions of the prior algorithms. Finally the algorithms can pack the ν results into a single matrix.

Remark. Here we note that if the input vectors $(\boldsymbol{z}, \boldsymbol{\alpha}, \boldsymbol{\beta})$ have the following format: the last $(\nu - u)$ components of all the vectors are 0, then the function $\boldsymbol{h}'_{\boldsymbol{a}}(\boldsymbol{z}, \boldsymbol{\alpha}, \boldsymbol{\beta})$ is consistent with the function $(h_{z_1,\alpha_1,\beta_1}(\boldsymbol{a}), ..., h_{z_u,\alpha_u,\beta_u}(\boldsymbol{a}), 0, \ldots, 0)$ as $h_{0,0,0}(\boldsymbol{a}) = 0$. This computes the family $\mathcal{H}^{u,\nu}$ as required in Lemma 6.3.

Let t, ℓ be the parameters used for the encoding structure i.e. $\mathbf{G}_{tn,\ell,m}$, and $z = 2^\nu$ be the parameter used by the recursive packing and unpacking algorithms. Then we describe the following algorithms. Here the matrix $\mathbf{B}$ recursively encodes $\{z_{ij}\}_{i\in[t],j\in[\nu]}$, and the matrix $\mathbf{B}'$ encodes $(\alpha_1, \beta_1, \alpha_2, \beta_2, \ldots, \alpha_\nu, \beta_\nu)$.

$\mathsf{PubEval}_{||}^{(\boldsymbol{a},\nu,t)}(\mathbf{B}, \mathbf{B}', \boldsymbol{h}'_{\boldsymbol{a}})$: Given two matrices $\mathbf{B}, \mathbf{B}' \in \mathbb{Z}_q^{n\times m}$, the public evaluation algorithm does:

1. For $i \in [t], j \in [\nu]$, recursively unpack matrix $\mathbf{B}$ as

$$\mathbf{B}_{ij} \leftarrow \mathsf{RecUnpack}(\mathbf{B}, (i, j));$$

   For $j \in [2\nu]$, unpack matrix $\mathbf{B}'$ as

$$\mathbf{B}'_j \leftarrow \mathsf{Unpack}(\mathbf{B}', i).$$

2. For $j \in [\nu]$, compute $\mathbf{B}^*_j \leftarrow \mathsf{PubEval}^{(\boldsymbol{a},t)}(\{\mathbf{B}_{ij}\}_{i\in[t]}, \mathbf{B}'_{2j-1}, \mathbf{B}'_{2j}, h'_{\boldsymbol{a}})$.
3. Output $\mathbf{B}^* \leftarrow \mathsf{Pack}(\{\mathbf{B}^*_j\}_{j\in[\nu]})$.

$\mathsf{TrapEval}_{||}^{(\boldsymbol{a},\nu,t)}((\boldsymbol{z}, \boldsymbol{\alpha}, \boldsymbol{\beta}), \mathbf{A}, \mathbf{R}, \mathbf{R}', \boldsymbol{h}'_{\boldsymbol{a}})$: The trapdoor evaluation algorithm does:

1. For $i \in [t], j \in [\nu]$, recursively unpack matrix $\mathbf{R}$ as

$$\mathbf{R}_{ij} \leftarrow \mathsf{RecUnpack}(\mathbf{R}, (i, j));$$

   For $j \in [2\nu]$, unpack matrix $\mathbf{R}'$ as

$$\mathbf{R}'_j \leftarrow \mathsf{Unpack}(\mathbf{R}', i).$$

2. For $j \in [\nu]$, compute $\mathbf{R}^*_j \leftarrow \mathsf{TrapEval}^{(\boldsymbol{a},t)}((z_j, \alpha_j, \beta_j), \mathbf{A}, \{\mathbf{R}_{ij}\}_{j\in[t]}, \mathbf{R}'_{2j-1}, \mathbf{R}'_{2j}, h'_{\boldsymbol{a}})$.
3. Output $\mathbf{R}^* \leftarrow \mathsf{Pack}(\{\mathbf{R}^*_j\}_{j\in[\nu]})$.

Lemma 6.5. *Let ℓ, t, ξ be parameters as defined above. Then the algorithms* $(\mathsf{PubEval}_{||}^{(\boldsymbol{a},\nu,t)}, \mathsf{TrapEval}_{||}^{(\boldsymbol{a},\nu,t)})$ *are* $\max(\delta_1, \delta_2)\delta_3\delta_4 = (\nu n^2 m^3 \xi^{O(\xi)} \ell \log^2 q \log_\ell q)$*-expanding for partitioning function $\boldsymbol{h}'_{\boldsymbol{a}}$ be defined above, where $\delta_1 = \ell \log_\ell q, \delta_2 = m\xi^{O(\xi)} \ell \log q \log_\ell q, \delta_3 = O(n^2 m^2 \log q), \delta_4 = \nu \log q$.*

Proof. By Lemma 5.4, the algorithm Unpack and RecUnpack in the first step incur $(\ell \log_\ell q) = \delta_1$-expanding and $(m\xi^{O(\xi)}\ell \log q \log_\ell q) = \delta_2$-expanding respectively. The second step, running TrapEval on partitioning function $h'_{\boldsymbol{a}}$ results in $O(n^2m^2 \log q) = \delta_3$-expanding as shown in Lemma 6.4. Lastly, the third step Pack incurs $(\nu \log q) = \delta_4$-expanding. Therefore, in total, the $(\mathsf{PubEval}_{||}^{(\boldsymbol{a},\nu,t)}, \mathsf{TrapEval}_{||}^{(\boldsymbol{a},\nu,t)})$ is $\max(\delta_1, \delta_2)\delta_3\delta_4$-expanding, where $\max(\delta_1, \delta_2)\delta_3\delta_4 = \nu n^2 m^3 z^{O(z)} \ell \log^2 q \log_\ell q$. □

The expanding factor might look large, yet it can be set to some fixed polynomial according to our IBE parameter selection: $\nu = \omega(1), \xi = \omega(1), \ell = n$. See details in the next section.

6.4 IBE Construction

We construct an IBE scheme based on the partition function $\boldsymbol{h}'_{\boldsymbol{a}} : \{0,1\}^n \to \mathbb{Z}_q^\nu$ with its associated evaluating algorithm $(\mathsf{PubEval}_{||}^{(\cdot,\nu,t)}, \mathsf{TrapEval}_{||}^{(\cdot,\nu,t)})$. We consider the identity space $\mathsf{ID} = \{0,1\}^n$ and boolean message space $\{0,1\}$. The description of IBE construction $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is as follows:

- $\mathsf{Setup}(1^\lambda)$: On input the security parameter λ, the setup algorithm generates a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ associated with its trapdoor $\mathbf{T}_{\mathbf{A}}$ using algorithm $\mathsf{TrapGen}(q, n, m)$. Then the algorithm picks two random matrices $\mathbf{B}, \mathbf{B}' \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$ and a random vector $\boldsymbol{u} \xleftarrow{\$} \mathbb{Z}_q^n$

  $$\mathsf{mpk} = (\mathbf{A}, \mathbf{B}, \mathbf{B}', \boldsymbol{u}), \qquad \mathsf{msk} = \mathbf{T}_{\mathbf{A}}.$$

  The algorithm sets global parameters ν, ℓ, t where we present in the parameter selection section later.

- $\mathsf{KeyGen}(\mathsf{mpk}, \mathsf{msk}, \mathsf{id})$: On input mpk, msk and an identity $\boldsymbol{a} = \mathsf{id} \in \mathsf{ID}$, the key generation algorithm first computes

  $$\mathbf{B}_{\mathsf{id}} \leftarrow \mathsf{PubEval}_{||}^{(\boldsymbol{a},\nu,t)}(\mathbf{B}, \mathbf{B}', \boldsymbol{h}'_{\boldsymbol{a}})$$

  Sample a short vector $\boldsymbol{r} \in \mathbb{Z}^{2m}$ using

  $$\boldsymbol{r} \leftarrow \mathsf{SampleLeft}(\mathbf{A}, \mathbf{B}_{\mathsf{id}}, \mathbf{T}_{\mathbf{A}}, \boldsymbol{u}, s_1)$$

  such that $[\mathbf{A}|\mathbf{B}_{\mathsf{id}}] \cdot \boldsymbol{r} = \boldsymbol{u} \bmod q$. Output $\mathsf{sk}_{\mathsf{id}} = \boldsymbol{r}$.

- $\mathsf{Enc}(\mathsf{mpk}, \mathsf{id}, \mu)$: On input the mpk, an identity $\boldsymbol{a} = \mathsf{id} \in \mathsf{ID}$ and a message $\mu \in \{0,1\}$, the encryption algorithm first computes

  $$\mathbf{B}_{\mathsf{id}} \leftarrow \mathsf{PubEval}_{||}^{(\boldsymbol{a},\nu,t)}(\mathbf{B}, \mathbf{B}', \boldsymbol{h}'_{\boldsymbol{a}})$$

  Then sample a random vector $\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_q^n$, noise vectors $\boldsymbol{e}_0, \boldsymbol{e}_1 \leftarrow \mathcal{D}_{\mathbb{Z}^m, s_2}$ and integer $e_1 \leftarrow \mathcal{D}_{\mathbb{Z}, s_3}$. Next, compute and output ciphertext $\mathsf{ct} = (\boldsymbol{c}_0, c_1)$

  $$\boldsymbol{c}_0 = \boldsymbol{s}^\mathsf{T}[\mathbf{A}|\mathbf{B}_{\mathsf{id}}] + (\boldsymbol{e}_0^\mathsf{T}, \boldsymbol{e}_1^\mathsf{T}), \qquad c_1 = \boldsymbol{s}^\mathsf{T}\boldsymbol{u} + e_1 + \mu \cdot \lfloor q/2 \rfloor$$

- $\mathsf{Dec}(\mathsf{sk}_{\mathsf{id}}, \mathsf{ct})$: On input a secret key $\mathsf{sk}_{\mathsf{id}} = \boldsymbol{r}$ and ciphertext $\mathsf{ct} = (\boldsymbol{c}_0, c_1)$, the decryption algorithm outputs

  $$\mu' = \mathsf{Round}(c_1 - \langle \boldsymbol{c}_0, \boldsymbol{r} \rangle \bmod q) \in \{0,1\}$$

6.5 Correctness Proof and Parameters Selection

We prove the correctness of IBE scheme as follows:

Lemma 6.6. *The identity-based encryption scheme* Π *is correct (c.f. Definition A.2).*

Proof. When the cryptosystem is operated as specified, we have during decryption,

$$\begin{aligned}\mu' &= \mathsf{Round}\left(c_1 - (\langle \boldsymbol{c}_0, \boldsymbol{r}\rangle \bmod q)\right)\\ &= \mathsf{Round}\left(\left\lfloor \frac{q}{2}\right\rfloor \mu + \underbrace{e - (\boldsymbol{e}_0^T|\boldsymbol{e}_1^T)\boldsymbol{r}}_{\text{small}}\right)\\ &= \mu \in \{0,1\}\end{aligned}$$

where the second equality follows from the definitions of $\boldsymbol{c}_0, c_1$, and $\boldsymbol{r}$, and the third equality follows if $e - (\boldsymbol{e}_0^T|\boldsymbol{e}_1^T)\boldsymbol{r}$ is indeed small, which holds w.h.p. by setting parameters appropriately as below.

This completes the proof of correctness. □

Parameter Selection. For arbitrarily small constant $\delta > 0$, we set the system parameters according to Table 2.

Parameters	Description	Setting
λ	security parameter	
n	PK-lattice row dimension	λ
m	PK-lattice column dimension	$n^{1+\delta}$
q	modulus	$n^{7.5+\epsilon}m^8$
s_1	$\mathsf{SampleLeft}$ and $\mathsf{SampleD}^{\mathcal{O}}$ width	$n^{4+\epsilon}m^{3.5}$
s_2	vector error width	$n^{3.5+\epsilon}m^{3.5}$
s_3	integer error width	$\sqrt{n}\log^{1+\epsilon}(n)$
s_4	ReRand width	$n^{3+\epsilon}m^{3.5}$
ℓ	integer-base parameter	n
ν	number of repetitions	$\log\log n$
t	The z range in partitioning function	$2\log 2n$

Table 2: IBE Parameters and Example Setting

These values are chosen in order to satisfy the following constraints:

- To ensure correctness, we require $|e - (\boldsymbol{e}_0^T|\boldsymbol{e}_1^T)\boldsymbol{r}| < q/4$; here we bound the dominating term:
$$|e_1^T\boldsymbol{r}| \le ||e_1^T|| \cdot ||\boldsymbol{r}|| \approx s_2\sqrt{m}\cdot s_1\sqrt{m} = ms_1s_2 < q/4.$$
- For $\mathsf{SampleLeft}$, we know $||\widetilde{\mathbf{T}_{\mathbf{A}}}|| = O(\sqrt{n\log(q)})$, so require that the sampling width s satisfies
$$s_1 > \sqrt{n\log(q)}\cdot\omega(\sqrt{\log(m)}).$$
- For $\mathsf{SampleD}^{\mathcal{O}}$, we know $||\widetilde{\mathbf{T}_{\mathbf{G}_{\nu n,\ell,m}}}|| \le \sqrt{\ell^2+1}$ and that
$$\begin{aligned}s_1 &\ge \sqrt{||\mathbf{R}^*||^2+1}\sqrt{|\mathbf{T}_{\mathbf{G}_{\nu n,\ell,m}}|+2}\\ &\ge \sqrt{m}\nu n^2m^3z^{O(z)}\ell\log^2 q\log_\ell q\cdot\sqrt{\ell^2+1}\\ &= O(n^{4+\epsilon}m^{3.5})\end{aligned}$$

Therefore, we need the (joint) sampling width s to, in fact, satisfy the stronger constraint

$$s_1 \geq n^{4+\epsilon} m^{4.5}$$

- To apply the Leftover Hash Lemma, we need $m \geq (n+1)\log(q) + \omega(\log(n))$.
- To apply Regev's reduction, we need $s_3 > \sqrt{n}\omega(\log(n))$ (s_3 here is an absolute value, not a ratio).
- To use algorithms ReRand in security proof (c.f. Lemma A.7), we need $s_4 > \sqrt{||\mathbf{R}^*||^2 + 1}$ and $s_2 = 2s_3 s_4$.

6.6 Security Proof

In this part, we show the security proof of our IBE construction as follows:

Theorem 6.7. *Assuming the hardness of the standard LWE assumption, our IBE construction is fully secure (cf. Definition A.3).*

Proof. We prove the security of our IBE construction by a sequence of hybrids, where the first hybrid is identical to the original security experiment $\mathbf{Expt}^{\mathsf{IBE}}_{\mathcal{A}}(1^\lambda)$ as in Definition A.3. We show that if a PPT adversary $\mathcal{A}$ that makes at most $|Q|$ secret key queries, can break the IBE scheme described above with non-negligible advantage ϵ (i.e. success probability $\frac{1}{2} + \epsilon$), then there exists a reduction that can break the LWE assumption with advantage $\mathsf{poly}(\epsilon) - \mathsf{negl}(\lambda)$. Given such an adversary $\mathcal{A}$, we consider the following hybrids.

The Sequence of Hybrids $(\mathsf{H}_0, \mathsf{H}_1, \mathsf{H}_2, \mathsf{H}_3, \mathsf{H}_4)$:

Hybrid H_0: This is the original security experiment $\mathbf{Expt}^{\mathsf{IBE}}_{\mathcal{A}}(1^\lambda)$ from Definition A.3 between the adversary $\mathcal{A}$ and the challenger.

Hybrid H_1: Hybrid H_1 is identical to hybrid H_0 except that we add an abort event that is independent of the adversary's view. Let $|Q|$ be the maximum size of that $\mathcal{A}$ can query, ϵ be the advantage of $\mathcal{A}$ in H_0, and n, ℓ, q be the parameters specified in Section 6.5. Now this hybrid experiment selects $u = \lceil \log_n(2|Q|/\epsilon) \rceil$, so that we have $n^u \geq 2|Q|/\epsilon \geq n^{u-1}$. Then it choose a random partitioning function $h^*_{\boldsymbol{z},\boldsymbol{\alpha},\boldsymbol{\beta}}$ by randomly choosing $\boldsymbol{z} \in [2n^2]^u, \boldsymbol{\alpha} \in \mathbb{Z}_q^u, \boldsymbol{\beta} \in \mathbb{Z}_q^u$, and passes it to the challenger. Recall that the hash function maps input $\boldsymbol{a} \in \{0,1\}^n$ to:

$$h^*_{\boldsymbol{z},\boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{a}) = (h^*_{z_1,\alpha_1,\beta_1}(\boldsymbol{a}), ..., h^*_{z_u,\alpha_u,\beta_u}(\boldsymbol{a}))$$

where $h^*_{z_i,\alpha_i,\beta_i}(\boldsymbol{a}) = \alpha_i \cdot f_{\boldsymbol{a}}(z_i) + \beta_i$ and $f_{\boldsymbol{a}}(x) = \sum_{i=0}^{n-1} a_i x^i$.

We then describe how the challenger behaves in hybrid H_1 as follows:

- **Setup**: The same as hybrid H_0 except the challenger keeps the hash function $h^*_{\boldsymbol{z},\boldsymbol{\alpha},\boldsymbol{\beta}}$ passed from the experiment.
- **Secret key and ciphertext query**: The challenger responds to identity queries and challenge ciphertext query (with a random choice of $b \in \{0,1\}$). We use set $Q = \{\boldsymbol{a}_i\}$ to denote the query set, and by definition challenge identity $\boldsymbol{a}^* \cap Q = \emptyset$.
- **Guess**: In the guess phase, the adversary outputs his guess $b' \in \{0,1\}$ for b. The challenger now does the abort check: $h^*_{\boldsymbol{z},\boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{a}^*) = 0$ and $h^*_{\boldsymbol{z},\boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{a}_i) \neq 0$ for all $\boldsymbol{a}_i \in Q$. If the condition does not hold, the challenger overwrites b' with a freshly random bit in $\{0,1\}$, and we say the challenge aborts the game.

Note that the adversary never sees the random hash function, and has no idea if an abort event took place. While it is convenient to describe the abort action at the end of the game, nothing would change if the challenger aborted the game as soon as the abort condition becomes true.

Hybrid H_2: In hybrid H_2, we change the method of generating matrix $\mathbf{B}$ in master public key. Recall that in hybrid H_1, matrix $\mathbf{B}$ is chosen at random from $\mathbb{Z}_q^{n\times m}$. In hybrid H_2, for $i \in [u]$, the challenger randomly chooses integers $z_i \in [2n^2], \alpha_i \in \mathbb{Z}_q, \beta_i \in \mathbb{Z}_q$. For $i \in [t], j \in [\nu]$, set matrix $\mathbf{E}_{ij}$ as

$$\mathbf{E}_{ij} = \begin{cases} z_{ij}\mathbf{G}_{n,2,m} & \text{if } j \le u \\ \mathbf{0}_{n\times m} & \text{otherwise} \end{cases}$$

then compute

$$\mathbf{E} = \mathsf{RecPack}(\{\mathbf{E}_{ij}\}_{i\in[t],j\in[\nu]}, \underbrace{\mathbf{0}_{n\times m}, ..., \mathbf{0}_{n\times m}}_{(\nu-u)t})$$

Next, set matrix $\mathbf{E}'$ for encoding $\{\alpha_i\}$ and $\{\beta_i\}$ as

$$\mathbf{E}' = [\alpha_1\mathbf{I}_n|\beta_1\mathbf{I}_n|\cdots|\alpha_u\mathbf{I}_n|\beta_u\mathbf{I}_n|\underbrace{\mathbf{0}_n|\cdots|\mathbf{0}_n}_{\nu t-2u}] \cdot \mathbf{G}_{t\nu n,\ell,m}$$

Then, the challenger sets

$$\mathbf{B} = \mathbf{A}\mathbf{R} + \mathbf{E}, \quad \mathbf{B}' = \mathbf{A}\mathbf{R}' + \mathbf{E}'$$

where $\mathbf{R}, \mathbf{R}' \in \{-1, 1\}^{m\times m}$ is randomly chosen. Additionally, the challenger uses the matrix $\mathbf{R}, \mathbf{R}'$ to generate the challenge ciphertext $(\boldsymbol{c}_0^*, c_1^*)$ in the following way:

- Choose a uniformly random vector $\boldsymbol{s} \in \mathbb{Z}_q^n$, an error vector $\boldsymbol{e} \leftarrow \mathcal{D}_{\mathbb{Z}^m,s_3}$ and an error integer $e \leftarrow \mathcal{D}_{\mathbb{Z},s_3}$.
- Compute $\mathbf{R}^* \leftarrow \mathsf{TrapEval}_{||}^{(\boldsymbol{a}^*,\nu,t)}((\boldsymbol{z},\boldsymbol{\alpha},\boldsymbol{\beta}), \mathbf{A}, \mathbf{R}, \mathbf{R}', h^*_{\boldsymbol{z},\boldsymbol{\alpha},\boldsymbol{\beta}})$. Then set challenge ciphertext

$$\boldsymbol{c}_0^* \leftarrow \mathsf{ReRand}([\mathbf{I}_m|\mathbf{R}^*], \boldsymbol{s}^\mathsf{T}\mathbf{A} + \boldsymbol{e}^\mathsf{T}, s_3, s_4), \quad c_1^* := \boldsymbol{s}^\mathsf{T}\boldsymbol{u} + e + \left\lfloor \frac{q}{2} \right\rfloor \mu$$

The rest of the hybrid is unchanged.

Hybrid H_3: In hybrid H_3, we change how matrix $\mathbf{A}$ in the master public key mpk is generated, and as well the method of answering secret key queries. Recall that in hybrid H_2, matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ is sampled with its trapdoor $\mathsf{T}_\mathbf{A}$ using algorithm $\mathsf{TrapGen}(q, n, m)$. To answer a secret key query for an identity $\boldsymbol{a}_i \in \{0,1\}^n$, the secret key is generated using algorithm $\mathsf{SampleLeft}$ associated with the trapdoor $\mathsf{T}_\mathbf{A}$.

In hybrid H_3, the challenger first samples a random matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ (without any trapdoor), and then sets matrix $\mathbf{B}$ in the same way as hybrid H_1. The challenger responds to the secret key query of ID $\boldsymbol{a}_i = (a_{i1}, ..., a_{in}) \in \{0,1\}^n$ by first computing whether $h^*_{\boldsymbol{z},\boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{a}_i) = 0$. If it is 0, then the challenger aborts as the previous hybrid. Otherwise, the challenger uses the algorithm $\mathsf{SampleD}^\mathcal{O}$ (c.f. Lemma 3.3) with the trapdoor $\mathbf{T}_{\mathbf{G}_{un,\ell,m}}$ to reply as follows:

$$\boldsymbol{r} \leftarrow \mathsf{SampleD}^\mathcal{O}\left(\mathbf{R}_{\boldsymbol{a}_i}, \mathbf{A}, \mathbf{E}'_{\boldsymbol{a}_i}, \boldsymbol{u}, s_1\right)$$

where the matrices $\mathbf{E}_{\boldsymbol{a}_i}$ and $\mathbf{R}_{\boldsymbol{a}_i}$ are

$$\mathbf{E}'_{\boldsymbol{a}_i} = [h^*_{z_1,\alpha_1,\beta_1}(\boldsymbol{a}_i)\mathbf{I}_n|\cdots|h^*_{z_u,\alpha_u,\beta_u}(\boldsymbol{a}_i)\mathbf{I}_n|\mathbf{0}_n|\cdots|\mathbf{0}_n] \cdot \mathbf{G}_{\nu n,\ell,m}$$
$$\mathbf{R}_{\boldsymbol{a}_i} = \mathsf{TrapEval}_{||}^{(\boldsymbol{a}_i,\nu,t)}((\boldsymbol{z},\boldsymbol{\alpha},\boldsymbol{\beta}), \mathbf{A}, \mathbf{R}, \mathbf{R}', h^*_{\boldsymbol{z},\boldsymbol{\alpha},\boldsymbol{\beta}})$$

By the δ-expanding of $(\mathsf{PubEval}_{||}^{(\boldsymbol{a}_i,\nu,t)}, \mathsf{TrapEval}_{||}^{(\boldsymbol{a}_i,\nu,t)})$, we have

$$\mathsf{PubEval}_{||}^{(\boldsymbol{a}_i,\nu,t)}(\mathbf{B},\mathbf{B}') = \mathbf{B}_{\boldsymbol{a}_i} = \mathbf{A}\mathbf{R}_{\boldsymbol{a}_i} + \mathbf{E}'_{\boldsymbol{a}_i}\mathbf{G}_{\nu n,\ell,m}$$

From the above computation, it is not hard to see that the challenger is able to answer the key query using $\mathsf{SampleD}^{\mathcal{O}}$ as long as $h^*_{\boldsymbol{z},\boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{a}_i) \neq 0$. We remark that (1) the δ-compatibility guarantees that the norm $|\mathbf{R}_{\boldsymbol{a}_i}| \leq \delta|\mathbf{R}|$ that (2) if the challenger does not abort, the relational invariant between the secret key and public key is the same as the original experiment (real scheme), as guaranteed by the $\mathsf{SampleD}^{\mathcal{O}}$ algorithm:

$$\left[\mathbf{A}\middle|\mathbf{B}_{\boldsymbol{a}_i}\right]\cdot\boldsymbol{r} = \boldsymbol{u} \bmod q$$

Hybrid H_4: Hybrid H_4 is identical to hybrid H_3 except that the challenge ciphertext $(\boldsymbol{c}_0^*, c_1^*)$ is chosen as a random independent element in $\mathbb{Z}_q^{2m} \times \mathbb{Z}_q$.

Analysis of Hybrids. The only difference between hybrids H_0 and H_1 is the abort event. We argue that the adversary still has non-negligible advantage in H_1 even though the abort event can happen. More formally, we will use Lemma 28 in the full version of the work [ABB10], which is described as follows.

Lemma 6.8. *Let I be a $Q+1$- ID tuple $(\mathsf{id}^*, \mathsf{id}_1, \ldots, \mathsf{id}_{|Q|})$ denoted the challenge ID along with the queried ID's, and $\epsilon(I)$ define the probability that an abort does not happen in hybrid H_1. For $i = 0, 1$, we set E_i be the event that $b = b'$ at the end of hybrid H_i. Assuming $\epsilon(I) \in [\epsilon_{\min}, \epsilon_{\max}]$, then we have*

$$\left|\mathsf{Pr}[\mathsf{E}_1] - \frac{1}{2}\right| \geq \epsilon_{\min}\left|\mathsf{Pr}[\mathsf{E}_0] - \frac{1}{2}\right| - \frac{1}{2}(\epsilon_{\max} - \epsilon_{\min})$$

The lemma was analyzed by Bellare and Ristenpart [BR09], and further elaborated in the work [ABB10]. As our overall proof just uses this lemma in a "black-box" way, we do not include its proof for simplicity of presentation. Next, we show indistinguishability between all the remaining consecutive hybrids.

Lemma 6.9. *Hybrid H_1 and H_2 are statistically indistinguishable.*

Proof. We show that hybrid H_2 is statistically close to H_1 using Lemma A.8. Note that the first difference between the two hybrids is how the matrices $\mathbf{B}, \mathbf{B}'$ and the error vector $\boldsymbol{e}_1$ in the challenge ciphertext were generated. All the other matrices/vectors are generated identically. In H_1, $(\mathbf{B}, \mathbf{B}', \boldsymbol{e}_1)$ together with the public matrix $\mathbf{A}$ look like $(\mathbf{A}, \mathbf{B}, \mathbf{B}')$, yet in H_2, they look like $(\mathbf{A}, \mathbf{B} = \mathbf{A}\mathbf{R} + \mathbf{E}, \mathbf{B}' = \mathbf{A}\mathbf{R}' + \mathbf{E}')$.

By Lemma A.8, we know that the following two distributions are statistically close:

$$(\mathbf{A}, \mathbf{B}, \mathbf{B}') \approx (\mathbf{A}, \mathbf{B} = \mathbf{A}\mathbf{R} + \mathbf{E}, \mathbf{B}' = \mathbf{A}\mathbf{R}' + \mathbf{E}')$$

where matrices $\mathbf{B}, \mathbf{B}'$ is sampled from uniform distribution over $\mathbb{Z}_q^{n\times m}$.The second difference lies in the generation of challenge ciphertext. The $\boldsymbol{c}_0^*$ component in hybrid H_1 and H_2 are generated respectively as,

$$\boldsymbol{c}_0^* = \boldsymbol{s}^\mathsf{T}[\mathbf{A}|\mathbf{B}_{\mathsf{id}^*}] + (\boldsymbol{e}_0^\mathsf{T}, \boldsymbol{e}_1^\mathsf{T}), \quad \boldsymbol{c}_0^* \leftarrow \mathsf{ReRand}([\mathbf{I}_m|\mathbf{R}^*], \boldsymbol{s}^\mathsf{T}\mathbf{A} + \boldsymbol{e}, s_3, s_4)$$

where $\mathbf{R}^* \leftarrow \mathsf{TrapEval}_{||}^{(\boldsymbol{a}^*,\nu,t)}((\boldsymbol{z},\boldsymbol{\alpha},\boldsymbol{\beta}), \mathbf{A}, \mathbf{R}, \mathbf{R}', h^*_{\boldsymbol{z},\boldsymbol{\alpha},\boldsymbol{\beta}})$, the error vectors $(\boldsymbol{e}_0, \boldsymbol{e}_1)$ is sampled from $\mathcal{D}_{\mathbb{Z}^m,s_2}$ and $\boldsymbol{e}$ is sampled from $\mathcal{D}_{\mathbb{Z}^m,s_3}$. By Lemma A.7, the distribution of $\boldsymbol{c}_0^*$ in hybrid H_2 is negligible close to the following:

$$\boldsymbol{c}_0^* = \boldsymbol{s}^\mathsf{T}\mathbf{A}[\mathbf{I}_m|\mathbf{R}^*] + (\boldsymbol{e}_0^\mathsf{T}, \boldsymbol{e}_1^\mathsf{T}) = \boldsymbol{s}^\mathsf{T}[\mathbf{A}|\mathbf{B}_{\mathsf{id}^*}] + (\boldsymbol{e}_0^\mathsf{T}, \boldsymbol{e}_1^\mathsf{T})$$

By setting the parameters as in Section 6.5, we can apply Lemma A.7. This completes the proof of the claim. □

Lemma 6.10. *Hybrid* H_2 *and* H_3 *are statistically indistinguishable.*

Proof. We show that hybrid H_2 is statistically close to H_3 using Lemmas A.11 and A.12. Note that in hybrid H_3, the public matrix $\mathbf{A}$ is sampled from $\mathsf{TrapGen}(q, n, m)$ along with a trapdoor, and secret key queries are answered using algorithm $\mathsf{SampleLeft}$; in hybrid H_3, $\mathbf{A}$ is sampled directly from the uniform distribution over $\mathbb{Z}_q^{n\times m}$, and these secret key queries are answered using algorithm $\mathsf{SampleD}^{\mathcal{O}}$.

By Lemma A.11, matrix $\mathbf{A}$ in hybrid H_2 is distributed close to uniform distribution over $\mathbb{Z}_q^{n\times m}$ as in hybrid H_3. For Gaussian parameter s set appropriately as in Section 6.5, $\mathsf{SampleLeft}$ and $\mathsf{SampleD}^{\mathcal{O}}$ are distributed identically by Lemma A.12. Therefore, hybrid H_2 and H_3 are statistically indistinguishable. □

Lemma 6.11. *Assuming the hardness of LWE assumption, hybrid* H_3 *and* H_4 *are computationally indistinguishable.*

Proof. Suppose there exists an adversary who has non-negligible advantage in distinguishing hybrid H_2 and H_3, then we can construct a reduction $\mathcal{B}$ that breaks the LWE assumption using the adversary $\mathcal{A}$. Recall in Definition A.9, an LWE instance is provided as a sampling oracle $\mathcal{O}$ that can be either uniformly random $\mathcal{O}_\$$ or a pseudorandom $\mathcal{O}_{\boldsymbol{s}}$ for some secret random $\boldsymbol{s} \in \mathbb{Z}_q^n$. The reduction $\mathcal{B}$ uses adversary $\mathcal{A}$ to distinguish the two oracles as follows:

Invocation. Reduction $\mathcal{B}$ requests $m+1$ instances from oracle $\mathcal{O}$, i.e. pair $(\boldsymbol{a}_i, b_i)$ for $i = 0, ..., m$.

Setup. Reduction $\mathcal{B}$ constructs master public key mpk as follows:

1. Set matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ to be the first m vectors $\boldsymbol{a}_i$ in pairs $(\boldsymbol{a}_i, b_i)$ for $i = 0, ..., m-1$.
2. Assign the $(m+1)$-th LWE instance $\boldsymbol{a}_{m+1}$ to be vector $\boldsymbol{u} \in \mathbb{Z}_q^n$.
3. Construct the reminder of master public key, namely matrices $\mathbf{B}, \mathbf{B}'$ as in hybrid H_2.
4. Send $\mathsf{mpk} = (\mathbf{A}, \mathbf{B}, \mathbf{B}', \boldsymbol{u})$ to $\mathcal{A}$.

Queries. Reduction $\mathcal{B}$ answers identity queries as in hybrid H_3, including aborting the simulation if needed.

Challenge ciphertext. When adversary $\mathcal{A}$ sends message (μ_0, μ_1) and challenge identity $\boldsymbol{a}^*$, reduction $\mathcal{B}$ does the following:

1. Set $\boldsymbol{v} \in \mathbb{Z}_q^m$ the first m integers b_i in LWE pairs $(\boldsymbol{a}_i, b_i)$, for $i = 0, ..., m-1$.
2. Set challenge ciphertext $(\boldsymbol{c}_0^*, c_1^*)$ as

$$\boldsymbol{c}_0^* = \boldsymbol{v}^*, \qquad c_1^* = b_{m+1} + \mu_{b^*} \left\lfloor \frac{q}{2} \right\rfloor$$

   where $\boldsymbol{v}^* \leftarrow \mathsf{ReRand}([\mathbf{I}_m | \mathbf{R}^*], \boldsymbol{v}, s_3, s_4)$, and $\mathbf{R}^* \leftarrow \mathsf{TrapEval}_{||}^{(\boldsymbol{a}^*, \nu, t)}((\boldsymbol{z}, \boldsymbol{\alpha}, \boldsymbol{\beta}), \mathbf{A}, \mathbf{R}, \mathbf{R}', h^*_{\boldsymbol{z},\boldsymbol{\alpha},\boldsymbol{\beta}})$.
3. Send challenge ciphertext $(\boldsymbol{c}_0^*, c_1^*)$ to adversary $\mathcal{A}$.

Guess. After being allowed to make additional queries, $\mathcal{A}$ guesses if it is interacting with a hybrid H_3 or H_4 challenger. Our simulator outputs the final guess as the answer to the LWE challenge it is trying to solve.

We can see that when $\mathcal{O} = \mathcal{O}_{\boldsymbol{s}}$, the adversary's view is as in hybrid H_3; when $\mathcal{O} = \mathcal{O}_\$$, the adversary's view is as in hybrid H_4. Hence, $\mathcal{B}$'s advantage in solving LWE is the same as $\mathcal{A}$'s advantage in distinguishing hybrids H_3 and H_4. □

Completing the Proof. Recall that $|Q|$ is the upper bound of the number of the adversary's key queries, and ϵ is the advantage of the adversary in H_0. By our setting of parameters in H_1, we have $|Q| \leq 0.5\epsilon n^u$. By setting $\delta := 0.5\epsilon$ in Lemma 6.1, we know that

$$\Pr_{h^*_{\boldsymbol{z},\boldsymbol{\alpha},\boldsymbol{\beta}}}\left[\begin{array}{c} h^*_{\boldsymbol{z},\boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{a}^*) = 0 \bigwedge \\ \forall \boldsymbol{a}_i \in Q : h^*_{\boldsymbol{z},\boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{a}_i) \neq 0 \end{array}\right] \in \left(\frac{1-0.5\epsilon}{q^u}, \frac{1}{q^u}\right).$$

Thus, we know that for any $(Q+1)$-tuple I denoting a challenge id along with ID queries, we have $\epsilon(I) \in \left(\frac{1-0.5\epsilon}{q^u}, \frac{1}{q^u}\right)$. Then by setting $[\epsilon_{\min}, \epsilon_{\max}] = [\frac{1-0.5\epsilon}{q^u}, \frac{1}{q^u}]$ in Lemma 6.8, we have

$$\left|\mathsf{Pr}[\mathsf{E}_1] - \frac{1}{2}\right| \geq \frac{1-0.5\epsilon}{q^u}\left|\mathsf{Pr}[\mathsf{E}_0] - \frac{1}{2}\right| - \frac{0.5\epsilon}{2q^u} = \frac{(1.5-\epsilon)\cdot\epsilon}{2q^u} \geq \frac{\epsilon}{4q^u}. \tag{6}$$

Note that $\left|\mathsf{Pr}[\mathsf{E}_0] - \frac{1}{2}\right| = \epsilon$ is the advantage of $\mathcal{A}$ in H_0, and the second inequality follows from the fact that $1.5 - \epsilon \geq 0.5$.

Next we show that the quantity $\frac{\epsilon}{4q^u}$ is still non-negligible (even though q^u might look large). Recall that we set $u = \lceil \log_n(2|Q|/\epsilon) \rceil$, so that we have $n^u \geq 2|Q|/\epsilon \geq n^{u-1}$. This implies $\frac{1}{n^u} \geq \frac{\epsilon}{2n|Q|}$. Since q is polynomial in our setting, so we have $q = O(n^c)$ for some constant c. Then we can further derive: $\epsilon/4q^u \geq \frac{\epsilon^2}{4\cdot 2^c |Q|^c q}$. This quantity is non-negligible as long as ϵ is non-negligible, q is polynomial for our setting of parameters, $|Q|$ is polynomially bounded, which implies $|Q|^c$ is polynomially bounded as well.

Then for $i = 1, 2, 3, 4$ we denote E_i as the event that the adversary successfully guesses the challenge bit, i.e. $b = b'$, at the end of hybrids $\mathsf{H}_1, \mathsf{H}_2, \mathsf{H}_3$ and H_4, respectively. From Lemmas 6.9 and 6.10, we know that the adjacent hybrids are indistinguishable, and thus we have

$$\mathsf{Pr}[\mathsf{E}_1] \approx \mathsf{Pr}[\mathsf{E}_2], \qquad \mathsf{Pr}[\mathsf{E}_2] \approx \mathsf{Pr}[\mathsf{E}_3]. \tag{7}$$

It is obvious that $\Pr[\mathsf{E}_4] = \frac{1}{2}$, as in this hybrid the challenge bit is independent of the adversary's view. From Lemma 6.11, we know that

$$|\mathsf{Pr}[\mathsf{E}_3] - \mathsf{Pr}[\mathsf{E}_4]| = \left|\mathsf{Pr}[\mathsf{E}_3] - \frac{1}{2}\right| \leq \mathbf{Adv}^{\mathsf{LWE}}_{\mathcal{B}}(1^\lambda). \tag{8}$$

Suppose $\mathcal{A}$ has non-negligible advantage ϵ in H_0. By the above computation, we know that

$$\frac{\epsilon^2}{4\cdot 2^c |Q|^c q} \leq \left|\mathsf{Pr}[\mathsf{E}_1] - \frac{1}{2}\right| \approx \left|\mathsf{Pr}[\mathsf{E}_2] - \frac{1}{2}\right| \approx \left|\mathsf{Pr}[\mathsf{E}_3] - \frac{1}{2}\right| \leq \mathbf{Adv}^{\mathsf{LWE}}_{\mathcal{B}}(1^\lambda),$$

which implies $\mathbf{Adv}^{\mathsf{LWE}}_{\mathcal{B}}(1^\lambda) \geq \frac{\epsilon^2}{4\cdot 2^c |Q|^c q} - \mathsf{negl}(\lambda)$. This means the reduction $\mathcal{B}$ defined in Lemma 6.11 breaks the LWE assumption with non-negligible probability. This reaches a contradiction, which completes the proof of Theorem 6.7. □

6.7 Optimizing the Security Analysis

Our security analysis can be optimized so that the reduction $\mathcal{B}$'s advantage can be lower bounded better, i.e., $\mathbf{Adv}^{\mathsf{LWE}}_{\mathcal{B}}(1^\lambda) \geq O\left(\frac{\epsilon^2}{|Q|q}\right)$ where ϵ is the advantage of the adversary $\mathcal{A}$ in attacking the IBE scheme. To achieve this, we need to slightly modify the basic hash function. It is not hard to see that the analysis carries almost the same as above, so we do not repeat the analysis for simplicity of presentation.

We recall that our basic partitioning function in Section 6.2: given an identity $\boldsymbol{a} \in \{0,1\}^n$, the hash function works as follows: $h_{z,\alpha,\beta}(\boldsymbol{a}) = \alpha \cdot f_{\boldsymbol{a}}(z) + \beta$ for $\alpha, \beta \in \mathbb{Z}_q$, $z \in [2n^2]$. Then we show that

- $\forall \boldsymbol{a} \in \{0,1\}^n$, $\mathsf{Pr}[h_{z,\alpha,\beta}(\boldsymbol{a}) = 0] = 1/q$.
- $\forall \boldsymbol{a} \neq \boldsymbol{b} \in \{0,1\}^n$, $\mathsf{Pr}[h_{z,\alpha,\beta}(\boldsymbol{b}) = 0 | h_{z,\alpha,\beta}(\boldsymbol{a}) = 0] \leq 1/n$,

and with a parallel repetition we can adjust the probability to $1/q^u$ and $1/n^u$ respectively.

After a more careful examination of the proof, we observed that the security loss comes from the gap between $1/q$ and $1/n$, and the reduction loss can be reduced to $O\left(\frac{\epsilon^2}{|Q|q}\right)$ if we can design a hash function where the conditional probability is roughly $1/q$. To achieve this, we still consider using multiple α's and z's with only one single β. That is, we consider the following hash family $\mathcal{H}'$ that contains the following functions:

$$h'_{z_1,z_2,\ldots,z_c,\alpha_1,\ldots,\alpha_c,\beta}(\boldsymbol{a}) = \sum_{i\in[c]} \alpha_i \cdot f_{\boldsymbol{a}}(z_i) + \beta,$$

for $c = \lceil \log_n q \rceil$, $z_i \in [n^2]$, $\alpha_i \in \mathbb{Z}_q$, $\beta \in \mathbb{Z}_q$ for all $i \in [c]$. With a similar analysis, we can prove that a random function from this family has the following parameters:

- $\forall \boldsymbol{a} \in \{0,1\}^n$, $\mathsf{Pr}_{h' \leftarrow \mathcal{H}'}[h'(\boldsymbol{a}) = 0] = 1/q$.
- $\forall \boldsymbol{a} \neq \boldsymbol{b} \in \{0,1\}^n$, $\mathsf{Pr}_{h' \leftarrow H'}[h'(\boldsymbol{b}) = 0 | h'(\boldsymbol{a}) = 0] \leq 2/q$.

The first property is exactly the same as Lemma 6.4, and the conditional probability can be upper bounded by $\mathsf{Pr}[\alpha_j = 0 | \exists j \in [c] : (f_{\boldsymbol{a}} - f_{\boldsymbol{b}})(z_j) \neq 0] + \mathsf{Pr}[(f_{\boldsymbol{a}} - f_{\boldsymbol{b}})(z_i) = 0 \, \forall i \in [c]] \leq 1/q + 1/n^c \leq 2/q$. From here, the same parallel repetition technique can adjust the parameters to $1/q^u$ and $(2/q)^u$, respectively. We note that c is a fixed constant as q is a fixed polynomial. This will only incur a constant blowup on the number of z's we need, and this can be packed by the recursive packing algorithm within the same matrix. (In our IBE scheme, the matrix is $\mathbf{B}$.) The expanding factor of the homomorphic computation will only blow up by a constant factor, so it will not affect the other parameters.

Now we are ready to revisit the security loss in the proof with these new parameters. We can set $u = \log_{q/2}[2|Q|/\epsilon]$ so that $(q/2)^u \geq 2|Q|/\epsilon \geq (q/2)^{u-1}$. The security analysis remains the same, except we can derive a better lower bound for the success probability in Equation 6. We observe that $\frac{2^{u-1}}{q^{u-1}} \geq \frac{\epsilon}{2|Q|}$, and therefore $\frac{1}{q^u} \geq \frac{\epsilon}{2^u|Q|q}$. Putting things together, we have $\frac{\epsilon}{4q^u} \geq \frac{\epsilon^2}{2^{u+2}|Q|q} = O(\frac{\epsilon}{|Q|q})$. The last equality comes from the fact that u is a constant (note that $|Q|$ is a polynomial, so $\log_{q/2}|Q|$ is a constant). This proves what we desired.

7 Application in Signature Setting

We can apply the technique in our IBE scheme to optimize the fully secure signature scheme proposed by Boyen in [Boy10], where we can obtain a fully secure signature scheme with constant size verification key. We assume the message space is $\mathcal{M} = \{0,1\}^n$. Our fully secure signature scheme $\Sigma = (\mathsf{Setup}, \mathsf{Sign}, \mathsf{Verify})$ is as follows:

- $\mathsf{Setup}(1^\lambda)$: On input the security parameter λ, the setup algorithm generates a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ associated with its trapdoor $\mathbf{T}_{\mathbf{A}}$ using algorithm $\mathsf{TrapGen}(q,n,m)$. Pick two random matrices $\mathbf{B}, \mathbf{B}' \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$ and a random vector $\boldsymbol{u} \xleftarrow{\$} \mathbb{Z}_q^n$

$$\mathsf{vk} = (\mathbf{A}, \mathbf{B}, \mathbf{B}', \boldsymbol{u}), \qquad \mathsf{sk} = \mathbf{T}_{\mathbf{A}}$$

- $\mathsf{Sign}(\mathsf{sk}, \boldsymbol{\mu})$: On input vk, sk and a message $\boldsymbol{\mu} \in \mathcal{M}$, the key generation algorithm first computes

$$\mathbf{B}_{\boldsymbol{\mu}} \leftarrow \mathsf{PubEval}_{||}^{(\boldsymbol{\mu},\nu,t)}(\mathbf{B}, \mathbf{B}')$$

Sample a short vector $\boldsymbol{r} \in \mathbb{Z}^{2m}$ using

$$\boldsymbol{r} \leftarrow \mathsf{SampleLeft}(\mathbf{A}, \mathbf{B}_{\boldsymbol{\mu}}, \mathbf{T}_{\mathbf{A}}, \boldsymbol{u}, s)$$

such that $[\mathbf{A}|\mathbf{B}_{\boldsymbol{\mu}}] \cdot \boldsymbol{r} = \boldsymbol{u}$. Output $\sigma_{\boldsymbol{\mu}} = \boldsymbol{r}$.

- $\mathsf{Verify}(\mathsf{vk}, \boldsymbol{\mu}, \sigma_{\boldsymbol{\mu}})$: On input a verification key vk, a message $\boldsymbol{\mu}$, and a signature $\sigma_{\boldsymbol{\mu}}$, the verification algorithm does:

  1. First check the signature $\sigma_{\boldsymbol{\mu}} = \boldsymbol{r}$ is a small but non-zero vector, i.e. $0 \neq |\boldsymbol{r}| \leq \sqrt{2m}s$.
  2. Check whether the following equation holds:

$$[\mathbf{A}|\mathbf{B}_{\boldsymbol{\mu}}] \cdot \boldsymbol{r} = 0 \bmod q$$

  where $\mathbf{B}_{\boldsymbol{\mu}} \leftarrow \mathsf{PubEval}_{||}^{(\boldsymbol{\mu},u,t)}(\mathbf{B}, \mathbf{B}')$.
  3. If both the verification steps pass, then output 1 (accept); otherwise output 0 (reject).

The correctness and security proofs (from Short Integer Solution) are the same as in [Boy10], with the modified partitioning function computation and simulation of matrices $\mathbf{B}, \mathbf{B}'$ as shown in the IBE's security proof (cf. Section 6.6). In what follows, we sketch the proof of unforgeability of the signature scheme Σ as follows:

Theorem 7.1. *Assuming the hardness of the standard SIS assumption, our signature scheme Σ is unforgeable (c.f. Definition A.5).*

Proof (sketch): Suppose there exists an adversary $\mathcal{A}$ that can break the unforgeability of the signature scheme Σ, then we can construct a reduction $\mathcal{B}$ that can simulate the the attack environment and use the forgery of adversary $\mathcal{A}$ to solve the SIS problem.

Invocation. Reduction $\mathcal{B}$ receives a random $\mathsf{SIS}_{q,n,m,\beta}$ instance $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, and is asked to output a solution to $\mathbf{A} \cdot \boldsymbol{r} = 0 \bmod q$, such that $0 \neq ||\boldsymbol{r}|| \leq \beta$.

Setup. Reduction $\mathcal{B}$ sets matrices $\mathbf{B}, \mathbf{B}'$ in vk as follows: randomly chooses integers $z_i \in [2n^2], \alpha_i \in \mathbb{Z}_q, \beta_i \in \mathbb{Z}_q$. For $i \in [t], j \in [\nu]$, set matrix $\mathbf{E}_{ij}$ as

$$\mathbf{E}_{ij} = \begin{cases} z_{ij}\mathbf{G}_{n,2,m} & \text{if } j \leq u \\ \mathbf{0}_{n \times m} & \text{otherwise} \end{cases}$$

then compute

$$\mathbf{E} = \mathsf{RecPack}(\{\mathbf{E}_{ij}\}_{i \in [t], j \in [\nu]}, \underbrace{\mathbf{0}_{n \times m}, ..., \mathbf{0}_{n \times m}}_{(\nu - u)t})$$

Next, set matrix $\mathbf{E}'$ for encoding $\{\alpha_i\}$ and $\{\beta_i\}$ as

$$\mathbf{E}' = [\alpha_1 \mathbf{I}_n | \beta_1 \mathbf{I}_n | \cdots | \alpha_u \mathbf{I}_n | \beta_u \mathbf{I}_n | \underbrace{\mathbf{0}_n | \cdots | \mathbf{0}_n}_{\nu t - 2u}] \cdot \mathbf{G}_{t\nu n, \ell, m}$$

Then, the challenger sets

$$\mathbf{B} = \mathbf{A}\mathbf{R} + \mathbf{E}, \quad \mathbf{B}' = \mathbf{A}\mathbf{R}' + \mathbf{E}'$$

where $\mathbf{R}, \mathbf{R}' \in \{-1, 1\}^{m \times m}$ is randomly chosen. Then $\mathcal{B}$ sends $\mathsf{vk} = (\mathbf{A}, \mathbf{B}, \mathbf{B}')$ to adversary $\mathcal{A}$.

Queries. Reduction $\mathcal{B}$ answers signature queries from $\mathcal{A}$ on message $\boldsymbol{\mu}_i \in \{0,1\}^n$ as follows:

1. Abort the simulation if

$$h^*_{\boldsymbol{z},\boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{\mu}_i) = (h^*_{z_1,\alpha_1,\beta_1}(\boldsymbol{\mu}_i), ..., h^*_{z_u,\alpha_u,\beta_u}(\boldsymbol{\mu}_i)) = \mathbf{0}$$

2. Otherwise, the challenger uses the algorithm $\mathsf{SampleD}^{\mathcal{O}}$ (c.f. Lemma 3.3) with the trapdoor $\mathbf{T}_{\mathbf{G}_{tn,\ell,m}}$ to reply as follows:

$$\boldsymbol{r}_i \leftarrow \mathsf{SampleD}^{\mathcal{O}}\left(\mathbf{R}_{\boldsymbol{\mu}_i}, \mathbf{A}, \mathbf{E}_{\boldsymbol{\mu}_i}, \boldsymbol{u},\ s\right)$$

   where the matrices $\mathbf{E}_{\boldsymbol{\mu}_i}$ and $\mathbf{R}_{\boldsymbol{\mu}_i}$ are

$$\mathbf{E}'_{\boldsymbol{\mu}_i} = [h^*_{z_1,\alpha_1,\beta_1}(\boldsymbol{\mu}_i)\mathbf{I}_n| \cdots |h^*_{z_u,\alpha_u,\beta_u}(\boldsymbol{\mu}_i)\mathbf{I}_n|\mathbf{0}_n| \cdots |\mathbf{0}_n]$$
$$\mathbf{R}_{\boldsymbol{\mu}_i} = \mathsf{TrapEval}_{||}^{(\boldsymbol{\mu}_i,\nu,t)}((\boldsymbol{z},\boldsymbol{\alpha},\boldsymbol{\beta}), \mathbf{A}, \mathbf{R}, \mathbf{R}', h^*_{\boldsymbol{z},\boldsymbol{\alpha},\boldsymbol{\beta}})$$

3. Output the signature $\boldsymbol{r}_i$ for message $\boldsymbol{\mu}_i$ to adversary $\mathcal{A}$.

Forgery. Reduction $\mathcal{B}$ receives a forged signature $\boldsymbol{r}^* = (\boldsymbol{r}^*_0, \boldsymbol{r}^*_1)$ on a new message $\boldsymbol{\mu}^*$, and does:

1. Abort the reduction if

$$h^*_{\boldsymbol{z},\boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{\mu}^*) = (h^*_{z_1,\alpha_1,\beta_1}(\boldsymbol{\mu}^*), ..., h^*_{z_u,\alpha_u,\beta_u}(\boldsymbol{\mu}^*)) \neq \mathbf{0}$$

2. Otherwise, we have

$$[\mathbf{A}|\mathbf{A}\mathbf{R}^*] \cdot \begin{pmatrix} \boldsymbol{r}^*_0 \\ \boldsymbol{r}^*_1 \end{pmatrix} = 0$$

   where $\mathbf{R}^* \leftarrow \mathsf{TrapEval}_{||}^{\boldsymbol{\mu}^*}((\boldsymbol{z},\boldsymbol{\alpha},\boldsymbol{\beta}), \mathbf{A}, \mathbf{R}, \mathbf{R}', h^*_{\boldsymbol{z},\boldsymbol{\alpha},\boldsymbol{\beta}})$.
3. Output $\boldsymbol{r} = \boldsymbol{r}^*_0 + \mathbf{R}^*\boldsymbol{r}^*_1$ as the SIS solution of matrix $\mathbf{A}$.

Analysis of Proof. The reduction is valid if $\mathcal{B}$ can complete the simulation without aborting with a substantial probability that is independent of the view of adversary $\mathcal{A}$ and the queries it makes. It follows from the bound of hash function in Lemma 6.1, that if an adversary successfully forges a signature with probability ϵ, by setting $\delta := 0.5\epsilon$ in Lemma 6.1, then reduction $\mathcal{B}$ solves SIS instance with probability

$$\epsilon' \geq \pi\frac{(1-\epsilon)}{n^u} \geq \pi\frac{(1-\epsilon)\epsilon}{2|Q|} \geq \frac{(1-\epsilon)\epsilon}{3|Q|}$$

where π is the probability that $\boldsymbol{r} = \boldsymbol{r}^*_0 + \mathbf{R}^*\boldsymbol{r}^*_1$ is a non-zero solution to SIS instance $\mathbf{A}$ given $\mathcal{B}$ does not abort the simulation, and $\pi \geq 2/3$. Recall that we set $u = \lceil \log_q(2|Q|/\epsilon) \rceil$, so that we have $n^u \geq 2|Q|/\epsilon \geq n^{u-1}$, which implies $\frac{1}{n^u} \geq \frac{\epsilon}{2n|Q|}$.

This concludes the proof sketch. □

References

[ABB10] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In Gilbert [Gil10], pages 553–572.

[AFL16] Daniel Apon, Xiong Fan, and Feng-Hao Liu. Compact identity based encryption from LWE. Cryptology ePrint Archive, Report 2016/125, 2016. `http://eprint.iacr.org/2016/125`.

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *28th ACM STOC*, pages 99–108. ACM Press, May 1996.

[Ajt99] Miklós Ajtai. Determinism versus non-determinism for linear time RAMs (extended abstract). In *31st ACM STOC*, pages 632–641. ACM Press, May 1999.

[AP10] Joël Alwen and Chris Peikert. Generating shorter bases for hard random lattices. *Theory of Computing Systems*, 48(3):535–553, 2010.

[AP14] Jacob Alperin-Sheriff and Chris Peikert. Faster bootstrapping with polynomial error. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 297–314. Springer, Heidelberg, August 2014.

[AS15] Jacob Alperin-Sheriff. Short signatures with short public keys from homomorphic trapdoor functions. In *IACR International Workshop on Public Key Cryptography*, pages 236–255. Springer, 2015.

[Bar89] David A Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in nc1. *Journal of Computer and System Sciences*, 38(1):150–164, 1989.

[BB04] Dan Boneh and Xavier Boyen. Secure identity based encryption without random oracles. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 443–459. Springer, Heidelberg, August 2004.

[BCH86] Paul W Beame, Stephen A Cook, and H James Hoover. Log depth circuits for division and related problems. *SIAM Journal on Computing*, 15(4):994–1003, 1986.

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

[BGG+14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 533–556. Springer, Heidelberg, May 2014.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS 2012*, pages 309–325. ACM, January 2012.

[BL16] Xavier Boyen and Qinyi Li. Towards tightly secure lattice short signature and id-based encryption. In Cheon and Takagi [CT16], pages 404–434.

[Boy10] Xavier Boyen. Lattice mixing and vanishing trapdoors: A framework for fully secure short signatures and more. In Phong Q. Nguyen and David Pointcheval, editors, *PKC 2010*, volume 6056 of *LNCS*, pages 499–517. Springer, Heidelberg, May 2010.

[BR09] Mihir Bellare and Thomas Ristenpart. Simulation without the artificial abort: Simplified proof and improved concrete security for Waters' IBE scheme. In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 407–424. Springer, Heidelberg, April 2009.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *52nd FOCS*, pages 97–106. IEEE Computer Society Press, October 2011.

[BV15] Zvika Brakerski and Vinod Vaikuntanathan. Constrained key-homomorphic PRFs from standard lattice assumptions - or: How to secretly embed a circuit in your PRF. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 1–30. Springer, Heidelberg, March 2015.

[CHKP10] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In Gilbert [Gil10], pages 523–552.

[CT16] Jung Hee Cheon and Tsuyoshi Takagi, editors. *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*. Springer, Heidelberg, December 2016.

[DRS04] Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 523–540. Springer, Heidelberg, May 2004.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 169–178. ACM Press, May / June 2009.

[Gil10] Henri Gilbert, editor. *EUROCRYPT 2010*, volume 6110 of *LNCS*. Springer, Heidelberg, May 2010.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 75–92. Springer, Heidelberg, August 2013.

[GV15] Sergey Gorbunov and Dhinakaran Vinayagamurthy. Riding on asymmetry: Efficient ABE for branching programs. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 550–574. Springer, Heidelberg, November / December 2015.

[GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 545–554. ACM Press, June 2013.

[GVW15] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits from LWE. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 503–523. Springer, Heidelberg, August 2015.

[HAO15] Ryo Hiromasa, Masayuki Abe, and Tatsuaki Okamoto. Packing messages and optimizing bootstrapping in GSW-FHE. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 699–715. Springer, Heidelberg, March / April 2015.

[KW03] Jonathan Katz and Nan Wang. Efficiency improvements for signature schemes with tight security reductions. In Sushil Jajodia, Vijayalakshmi Atluri, and Trent Jaeger, editors, *ACM CCS 03*, pages 155–164. ACM Press, October 2003.

[KY16] Shuichi Katsumata and Shota Yamada. Partitioning via non-linear polynomial functions: More compact IBEs from ideal lattices and bilinear maps. In Cheon and Takagi [CT16], pages 682–712.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 700–718. Springer, Heidelberg, April 2012.

[MR04] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on Gaussian measures. In *45th FOCS*, pages 372–381. IEEE Computer Society Press, October 2004.

[Pei15] Chris Peikert. A decade of lattice cryptography. Cryptology ePrint Archive, Report 2015/939, 2015. `http://eprint.iacr.org/2015/939`.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005.

[Sha84] Adi Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 47–53. Springer, Heidelberg, August 1984.

[vDGHV10] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In Gilbert [Gil10], pages 24–43.

[VV85] Leslie G Valiant and Vijay V Vazirani. Np is as easy as detecting unique solutions. In *Proceedings of the seventeenth annual ACM symposium on Theory of computing*, pages 458–463. ACM, 1985.

[Xag13] Keita Xagawa. Improved (hierarchical) inner-product encryption from lattices. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 235–252. Springer, Heidelberg, February / March 2013.

[Yam16] Shota Yamada. Adaptively secure identity-based encryption from lattices with asymptotically shorter public parameters. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 32–62. Springer, Heidelberg, May 2016.

[Yam17] Shota Yamada. Asymptotically compact adaptively secure lattice ibes and verifiable random functions via generalized partitioning techniques. Cryptology ePrint Archive, Report 2017/096, 2017. `http://eprint.iacr.org/2017/096`.

[ZCZ16] Jiang Zhang, Yu Chen, and Zhenfeng Zhang. Programmable hash functions from lattices: Short signatures and IBEs with small key sizes. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 303–332. Springer, Heidelberg, August 2016.

A Preliminaries

A.1 Fully Homomorphic Encryption

We recall the definition of (leveled) fully homomorphic encryption in the following. A (leveled) FHE is a tuple of algorithms $\Pi = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ described as follows:

- $\mathsf{Setup}(1^\lambda, 1^d)$: Given input the security parameter λ and maximum supported depth d, the setup algorithm outputs secret key sk and public key pk.
- $\mathsf{Enc}(\mathsf{pk}, \mu)$: On input pk and a message μ, the encryption algorithm outputs a ciphertext ct.
- $\mathsf{Eval}(\mathsf{pk}, \mathcal{C}, (\mathsf{ct}_1, ..., \mathsf{ct}_\ell))$: On input a boolean circuit $\mathcal{C}$ of depth $\leq d$ along with ℓ ciphertexts $(\mathsf{ct}_1, ..., \mathsf{ct}_\ell)$, the evaluation algorithm outputs an evaluated ciphertext ct.
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$: On input some ciphertext ct and a secret key sk, the decryption algorithm outputs a message μ.

Definition A.1 ((leveled) FHE)**.** *We call a scheme* $\Pi = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ *described above a (leveled) FHE scheme, if it satisfies:*

Correctness: *Let* $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^d)$ *and* $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}, \mu_i)$*, for* $i \in [\ell]$*. Let* $\mathcal{C}$ *be any boolean circuit of depth* $\leq d$ *and* $\mathsf{ct} \leftarrow \mathsf{Eval}(\mathsf{pk}, \mathcal{C}, (\mathsf{ct}_1, ..., \mathsf{ct}_\ell))$*. Then we have* $\mathsf{Dec}(\mathsf{ct}, \mathsf{sk}) = \mathcal{C}(\mu_1, ..., \mu_\ell)$*.*

Semantic security: *For any polynomial* $d = d(\lambda)$ *and any two messages* μ_0, μ_1*, the following distributions are computationally indistinguishable*

$$(\mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, \mu_0)) \approx (\mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, \mu_1))$$

where $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^d)$*.*

Compactness: *The size of the evaluated ciphertext, i.e.* $\mathsf{ct} \leftarrow \mathsf{Eval}(\mathsf{pk}, \mathcal{C}, (\mathsf{ct}_1, ..., \mathsf{ct}_\ell))$*, should be independent of circuit* $\mathcal{C}$ *and* ℓ*, but can depend on* λ *and* d*.*

A.2 Identity Based Encryption

We recall that *identity based encryption* (IBE) was introduced by Shamir [Sha84], and that Boneh and Franklin [BF01] proposed the first construction based on bilinear groups. An IBE scheme Π consists of a tuple of algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$:

- $\mathsf{Setup}(1^\lambda)$: Given the security parameter λ, the setup algorithm outputs the master key pair $(\mathsf{mpk}, \mathsf{msk})$.
- $\mathsf{KeyGen}(\mathsf{msk}, \mathsf{id} \in \mathsf{ID})$: Given the master secret key msk and an identity id, the key generation algorithm then outputs a secret key $\mathsf{sk}_{\mathsf{id}}$ for the identity id.
- $\mathsf{Enc}(\mathsf{mpk}, \mathsf{id}, \mu)$: Given mpk, an identity id and a message μ, the encryption algorithm outputs a ciphertext ct.
- $\mathsf{Dec}(\mathsf{sk}_{\mathsf{id}}, \mathsf{ct})$: Given secret key $\mathsf{sk}_{\mathsf{id}}$ and a ciphertext ct, the decryption algorithm decrypts ciphertexts to messages μ' using the secret key $\mathsf{sk}_{\mathsf{id}}$.

Definition A.2 (Correctness)**.** *We say an IBE scheme* Π *is* correct *if for every identity* $\mathsf{id} \in \mathsf{ID}$*, every message* $\mu \in \mathcal{M}$*, and every* $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$*,*

$$\Pr[\mathsf{Dec}(\mathsf{sk}_{\mathsf{id}}, \mathsf{Enc}(\mathsf{mpk}, \mathsf{id}, \mu)) = \mu] \geq 1 - \mathsf{negl}(\lambda).$$

For the security definition of IBE, we use the following experiment to describe it. Formally, for any PPT adversary $\mathcal{A}$, we consider the experiment $\mathbf{Expt}^{\mathsf{IBE}}_{\mathcal{A}}(1^\lambda)$:

1. **Setup**: A challenger runs the $\mathsf{Setup}(1^\lambda)$ algorithm, and sends the master public key mpk to the adversary.

2. **Query Phase I**: Proceeding adaptively, the adversary $\mathcal{A}$ queries a sequence of identities $(\mathsf{id}_1, ..., \mathsf{id}_m)$. On the i-th query, the challenger runs $\mathsf{KeyGen}(\mathsf{msk}, \mathsf{id}_i)$, and sends the result $\mathsf{sk}_{\mathsf{id}_i}$ to $\mathcal{A}$.

3. **Challenge**: Once adversary $\mathcal{A}$ decides that Query Phase I is over, it outputs the challenge identity id^* and two length-equal messages (μ_0^*, μ_1^*), under the constraint that the challenge identity id^* has never been queried before. In response, the challenger selects random $b \in \{0, 1\}$, and sends the ciphertext $\mathsf{Enc}(\mathsf{mpk}, \mathsf{id}^*, \mu_b^*)$ to $\mathcal{A}$.

4. **Query Phase II**: Adversary $\mathcal{A}$ continues to issue identity queries $(\mathsf{id}_{m+1}, ..., \mathsf{id}_n)$ adaptively, under the restriction that $\mathsf{id}_i \neq \mathsf{id}^*$. The challenger responds by issuing keys $\mathsf{sk}_{\mathsf{id}_i}$ as in Query Phase I.

5. **Guess**: Adversary $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$.

We define the advantage of adversary $\mathcal{A}$ in attacking an IBE scheme Π as

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

where the probability is over the randomness of the challenger and adversary.

Definition A.3 (IBE security)**.** *We say an IBE scheme* Π *is fully secure, if for all* PPT *adversaries* $\mathcal{A}$*, we have*

$$\mathbf{Adv}^{\mathsf{IBE}}_{\mathcal{A}}(1^\lambda) \leq \mathsf{negl}(\lambda)$$

A.3 Signature Scheme

A signature scheme for message space $\mathcal{M}$ consist of three algorithms $\Sigma = (\mathsf{Setup}, \mathsf{Sign}, \mathsf{Verify})$ with details as follows:

- $\mathsf{Setup}(1^\lambda)$: Given security parameter λ, the setup algorithm outputs signing key sk and verification key vk.
- $\mathsf{Sign}(\mathsf{sk}, \mu \in \mathcal{M})$: Given secret key sk and message $\mu \in \mathcal{M}$, the signing algorithm outputs a signature σ for the message.
- $\mathsf{Verify}(\mathsf{vk}, \mu, \sigma)$: Given verification key vk, a message μ and a signature σ, the verification algorithm outputs 1 (accept) or 0 (reject).

Definition A.4 (Correctness)**.** *We say a signature scheme* Σ *is correct, if for any message* $\mu \in \mathcal{M}$ *and any* $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\lambda)$*, we have*

$$\mathsf{Pr}[\mathsf{Verify}(\mathsf{vk}, \mu, \mathsf{Sign}(\mathsf{sk}, \mu)) = 1] = 1$$

For the unforgeability of signature scheme Σ, we use the following experiment to describe it. Formally, for any PPT adversary $\mathcal{A}$, we consider the experiment $\mathbf{Expt}^{\mathsf{sig}}_{\mathcal{A}}(1^\lambda)$:

1. **Setup**: A challenger runs the $\mathsf{Setup}(1^\lambda)$ algorithm, and sends the verification key vk to the adversary.

2. **Query Phase**: Proceeding adaptively, the adversary $\mathcal{A}$ queries a sequence of messages $(\mu_1, ..., \mu_m)$. On the i-th query, the challenger runs $\sigma_i \leftarrow \mathsf{Sign}(\mathsf{sk}, \mu_i)$, and sends the result σ_i to $\mathcal{A}$.

3. **Forgery**: Once adversary $\mathcal{A}$ decides that Query Phase is over, it outputs a message/signature pair (μ^*, σ^*), where message μ^* is not queried before.

We define the advantage of adversary $\mathcal{A}$ in attacking an IBE scheme Π as

$$\mathbf{Adv}^{\mathsf{sig}}_{\mathcal{A}}(1^\lambda) = \Pr[\mathsf{Verify}(\mathsf{vk}, \mu^*, \sigma^*) = 1]$$

where the probability is over the randomness of the challenger and adversary.

Definition A.5 (Unforgeability)**.** *We say a signature scheme Σ is unforgeable, if for all* PPT *adversaries $\mathcal{A}$, we have*

$$\mathbf{Adv}^{\mathsf{sig}}_{\mathcal{A}}(1^\lambda) \leq \mathsf{negl}(\lambda)$$

A.4 Lattice Background

A full-rank m-dimensional integer lattice $\Lambda \subset \mathbb{Z}^m$ is a discrete additive subgroup whose linear span is $\mathbb{R}^m$. The basis of Λ is a linearly independent set of vectors whose linear combinations are exactly Λ. Every integer lattice is generated as the $\mathbb{Z}$-linear combination of linearly independent vectors $\mathbf{B} = \{\boldsymbol{b}_1, ..., \boldsymbol{b}_m\} \subset \mathbb{Z}^m$. For a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, we define the "$q$-ary" integer lattices:

$$\Lambda_q^\perp = \{\boldsymbol{e} \in \mathbb{Z}^m | \mathbf{A}\boldsymbol{e} = \mathbf{0} \bmod q\}, \qquad \Lambda_q^{\mathbf{u}} = \{\boldsymbol{e} \in \mathbb{Z}^m | \mathbf{A}\boldsymbol{e} = \boldsymbol{u} \bmod q\}$$

It is obvious that $\Lambda_q^{\boldsymbol{u}}$ is a coset of $\Lambda_q^\perp$.

Let Λ be a discrete subset of $\mathbb{Z}^m$. For any vector $\boldsymbol{c} \in \mathbb{R}^m$, and any positive parameter $\sigma \in \mathbb{R}$, let $\rho_{\sigma,\boldsymbol{c}}(\boldsymbol{x}) = \exp(-\pi||\boldsymbol{x} - \boldsymbol{c}||^2/\sigma^2)$ be the Gaussian function on $\mathbb{R}^m$ with center $\boldsymbol{c}$ and parameter σ. Next, we let $\rho_{\sigma,\boldsymbol{c}}(\Lambda) = \sum_{\boldsymbol{x}\in\Lambda} \rho_{\sigma,\boldsymbol{c}}(\boldsymbol{x})$ be the discrete integral of $\rho_{\sigma,\boldsymbol{x}}$ over Λ, and let $\mathcal{D}_{\Lambda,\sigma,\boldsymbol{c}}(\boldsymbol{y}) := \frac{\rho_{\sigma,\boldsymbol{c}}(\boldsymbol{y})}{\rho_{\sigma,\boldsymbol{c}}(\Lambda)}$. We abbreviate this as $\mathcal{D}_{\Lambda,\sigma}$ when $\boldsymbol{c} = \mathbf{0}$.

Let S^m denote the set of vectors in $\mathbb{R}^m$ whose length is 1. Then the norm of a matrix $\mathbf{R} \in \mathbb{R}^{m\times m}$ is defined to be $\mathsf{sup}_{\boldsymbol{x}\in S^m}||\mathbf{R}\boldsymbol{x}||$. Then we have the following lemma, which bounds the norm for some specified distributions.

Lemma A.6 ([ABB10])**.** *With respect to the norm defined above, we have the following bounds:*

- *Let $\mathbf{R} \in \{-1, 1\}^{m\times m}$ be chosen at random, then we have $\mathsf{Pr}[||\mathbf{R}|| > 12\sqrt{2m}] < e^{-2m}$.*
- *Let $\mathbf{R}$ be sampled from $\mathcal{D}_{\mathbb{Z}^{m\times m},\sigma}$, then we have $\mathsf{Pr}[||\mathbf{R}|| > \sigma\sqrt{m}] < e^{-2m}$.*

The following lemma, proposed in [KY16], on noise rerandomization plays an important role in the security proof of our IBE construction.

Lemma A.7 (Noise rerandomization)**.** *Let q, m, m' be positive integers and r be a positive real number satisfying $r > \max\{\omega(\sqrt{\log m}), \omega(\sqrt{\log m'})\}$. Let $\boldsymbol{b}_1 \in \mathbb{Z}_q^m$ be arbitrary vector and $\boldsymbol{e}_1$ chosen from $\mathcal{D}_{\mathbb{Z}^m,r}$. Then for any $\mathbf{V} \in \mathbb{Z}^{m+m'}$ and positive real $s > |\mathbf{V}|$, there exists a* PPT *algorithm $\mathsf{ReRand}(\mathbf{V}, \boldsymbol{b}_1 + \boldsymbol{e}_1, r, s)$ that outputs $\boldsymbol{b}_2$ such $\boldsymbol{b}_2^\mathsf{T} = \boldsymbol{b}_1^\mathsf{T}\mathbf{V} + \boldsymbol{e}_2^\mathsf{T}$ where $\boldsymbol{e}_2$ is distributed statistically close to $\mathcal{D}_{\mathbf{Z}^{m'},2rs}$.*

Randomness Extraction. We will use the following lemma to argue the indistinghishability of two different distributions, which is a generalization of the leftover hash lemma proposed by Dodis et al. [DRS04].

Lemma A.8 ([ABB10])**.** *Suppose that $m > (n + 1)\log q + \omega(\log n)$. Let $\mathbf{R} \in \{-1, 1\}^{m\times k}$ be chosen uniformly at random for some polynomial $k = k(n)$. Let $\mathbf{A}, \mathbf{B}$ be matrix chosen randomly from $\mathbb{Z}_q^{n\times m}, \mathbb{Z}_q^{n\times k}$ respectively. Then, for all vectors $\boldsymbol{w} \in \mathbb{Z}^m$, the two following distributions are statistically close:*

$$(\mathbf{A}, \mathbf{AR}, \mathbf{R}^\mathsf{T}\boldsymbol{w}) \approx (\mathbf{A}, \mathbf{B}, \mathbf{R}^\mathsf{T}\boldsymbol{w})$$

Learning With Errors. The LWE problem was introduced by Regev [Reg05], who showed that solving it *on average* is as hard as (quantumly) solving several standard lattice problems *in the worst case*.

Definition A.9 (LWE)**.** *For an integer $q = q(n) \geq 2$, and an error distribution $\chi = \chi(n)$ over $\mathbb{Z}_q$, the* Learning With Errors problem $\mathsf{LWE}_{n,m,q,\chi}$ *is to distinguish between the following pairs of distributions (e.g. as given by a sampling oracle $\mathcal{O} \in \{\mathcal{O}_{\boldsymbol{s}}, \mathcal{O}_{\$}\}$):*

$$\{\mathbf{A}, \boldsymbol{s}^{\mathsf{T}}\mathbf{A} + \boldsymbol{x}\} \text{ and } \{\mathbf{A}, \boldsymbol{u}\}$$

where $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$, $\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_q^n$, $\boldsymbol{u} \xleftarrow{\$} \mathbb{Z}_q^m$, and $\boldsymbol{x} \xleftarrow{\$} \chi^m$.

Short Integer Solution. The SIS problem was first suggested to be hard on average by Ajtai [Ajt99] and then formalized by Micciancio and Regev [MR04].

Definition A.10 (SIS)**.** *For any $n \in \mathbb{Z}$, and any functions $m = m(n), q = q(n), \beta = \beta(n)$, the average-case Short Integer Solution problem ($\mathsf{SIS}_{q,n,m,\beta}$) is: Given an integer q, a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ chosen uniformly at random and a real $\beta \in \mathbb{R}$, find a non-zero integer vector $\boldsymbol{z} \in \mathbb{Z}^m \setminus \{\mathbf{0}\}$, such that $\mathbf{A}\boldsymbol{z} = 0 \mod q$ and $||\boldsymbol{z}|| \leq \beta$.*

Micciancio and Regev [MR04] showed that solving the average-case $\mathsf{SIS}_{q,n,m,\beta}$ problem for certain parameters is as hard as approximating the Shortest Independent Vector Problem in the worst case to within certain $\gamma = \beta \cdot \tilde{O}(\sqrt{n})$ factors.

Sampling Algorithms. We will use the following algorithms to sample short vectors from specified lattices.

Lemma A.11 ([GPV08, AP10])**.** *Let q, n, m be positive integers with $q \geq 2$ and sufficiently large $m = \Omega(n \log q)$. There exists a* PPT *algorithm $\mathsf{TrapGen}(q, n, m)$ that with overwhelming probability outputs a pair $(\mathbf{A} \in \mathbb{Z}_q^{n\times m}, \mathbf{T_A} \in \mathbb{Z}^{m\times m})$ such that $\mathbf{A}$ is statistically close to uniform in $\mathbb{Z}_q^{n\times m}$ and $\mathbf{T_A}$ is a basis for $\Lambda_q^{\perp}(\mathbf{A})$ satisfying*

$$||\mathbf{T_A}|| \leq O(n \log q) \quad \textit{and} \quad ||\widetilde{\mathbf{T_A}}|| \leq O(\sqrt{n \log q})$$

except with $\mathsf{negl}(n)$ probability.

Lemma A.12 ([GPV08, CHKP10, ABB10])**.** *Let $q > 2, m > n$. There is an algorithm $\mathsf{SampleLeft}(\mathbf{A}, \mathbf{B}, \mathbf{T_A}, \boldsymbol{u}, s)$: It takes as input: (1) a rank-n matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, and any matrix $\mathbf{B} \in \mathbb{Z}_q^{n\times m_1}$, (2) a "short" basis $\mathbf{T_A}$ for lattice $\Lambda_q^{\perp}(\mathbf{A})$, a vector $\boldsymbol{u} \in \mathbb{Z}_q^n$,, (3) a Gaussian parameter $s > ||\widetilde{\mathbf{T_A}}|| \cdot \omega(\sqrt{\log(m + m_1)})$. Then outputs a vector $\boldsymbol{r} \in \mathbb{Z}^{m+m_1}$ distributed statistically close to $\mathcal{D}_{\Lambda_q^{\boldsymbol{u}}(\mathbf{F}),s}$ where $\mathbf{F} := (\mathbf{A}|\mathbf{B})$.*