Fully-Secure Lattice-Based IBE as Compact as PKE

Daniel Apon [*] Xiong Fan [†] Feng-Hao Liu [‡]

February 11, 2016

Abstract

We construct an identity-based encryption (IBE) scheme from the standard Learning with Errors (LWE) assumption, which both has a *compact* public-key (with size similar to known lattice-based PKE schemes) and also achieves *adaptive* security in the standard model. This improves over previous IBE schemes from lattices, which either have a public key that grows at least linearly with the length of the supported identities, or achieve a non-adaptive notion of security, or require a random oracle.

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.

1 Introduction

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, without managing a public key infrastructure, and without the online assistance of a third party. Instead, users initially register an arbitrary string id (such as an email or IP address) with a central key authority, who generates the corresponding private key $\mathsf{sk}_{\mathsf{id}}$ for each user using a master secret msk. Moreover, there is a single public key mpk that allows encrypting private messages to any identity.

Boneh and Franklin [BF01] proposed the first construction of IBE based on bilinear groups in the random oracle model, and Waters [Wat05] gave the first standard-model instantiation of IBE also using bilinear groups. For lattice-based constructions, Gentry, Peikert, and Vaikuntanathan [GPV08] proposed the first lattice-based IBE in the random oracle model, and the concurrent works of Cash et al. [CHKP10] and Agrawal et al. [ABB10] gave standard-model IBE from lattices.

A central open question is to construct *efficient* and *fully secure* identity-based encryption schemes from standard lattice assumptions, as measured by the size of the keys and ciphertexts in the system (cf. [Pei15, Question 9]). Indeed, the state-of-the-art IBE schemes from bilinear groups [Lew12, BKP14, Wee16] enjoy a constant-size public key and ciphertexts, whereas the most efficient and *fully-secure* IBE from lattices [ABB10] requires a public key consisting of a number of matrices *linear* in the dimension of the supported identities. By comparison, the public key of Dual Regev type public key encryption (PKE) [GPV08] has size dominated by a *single* matrix.

Historically speaking, a key technical barrier for progress on efficient IBE in the lattice world follows from a "one-line" observation of Boneh and Franklin [BF01]: identity-based encryption implies existentially

[*] University of Maryland, dapon@cs.umd.edu.

[†] Cornell University, xfan@cs.cornell.edu.

[‡] Florida Atlantic University, fenghao.liu@fau.edu.

unforgeable digital signatures in a generic, black-box manner. Moreover in the case of lattices, there is a non-black-box transformation [ABB10] that is *efficiency-preserving;* that is, constant-sized IBE from lattices (effectively) implies constant-sized signatures from lattices.

This latter question of lattice-based signature efficiency has remained wide open until a recent, breakthrough line of work: Ducas and Micciancio [DM14] first demonstrated how to reduce the public key of lattice-based signatures to a logarithmic number of matrices. Böhl et al. [BHJ$^+$15] showed how to reduce the size of the signatures to a logarithmic number of lattice vectors. And ultimately, Alperin-Sheriff [Alp15] proposed a lattice-based signature scheme with a *constant* number of matrices in the public key by extending the homomorphic trapdoor function technique of Gorbunov, Vaikuntanathan, and Wichs [GVW15] — originally used to construct leveled fully homomorphic signatures (but without the short public key of [Alp15]).

As such, the question of efficient identity-based encryption from lattices is ripe for additional progress. We then ask:

Is there a standard-model, fully-secure, lattice-based Identity-Based Encryption scheme that is comparably efficient to existing lattice-based Public Key Encryption schemes?

Or more concretely:

Is there a fully-secure, lattice-based IBE with a compact *public key?*

1.1 Our Contributions

In this work, we construct a fully-secure identity-based encryption scheme from the standard Learning with Errors [Reg05] assumption with a *compact* public key of bit-size (about) $2nm\log(q)$, where m and q are small polynomials in n. This asymptotically matches the public space complexity of known lattice-based PKE schemes, and points us toward real-world-efficient implementations of lattice-based IBE. We emphasize that prior fully-secure, lattice-based IBE schemes [CHKP10, ABB10] have a public key whose size grows at least linearly with ℓ, the length of identities, whereas our IBE's PK size is (at least) asymptotically sublinear in the identity length. For a succinct comparison to prior work, see Table 1 below.

Formally, we obtain the following result:

Theorem 1.1 (Main)**.** *Under the standard Learning with Errors (LWE) assumption, there is an identity-based encryption (IBE) scheme with full (i.e. adaptive) security, where*

- *the modulus q is a prime of size polynomial in the security parameter n,*
- *ciphertexts consist of a vector in $\mathbb{Z}_q^{2m+1}$, and*
- *the public key consists of* two *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 appendix for details):

Theorem 1.2. *Under the standard Short Integer Solution (SIS) assumption, there is an existentially unforgeable digital signature scheme, where*

- *the modulus q is a prime of size polynomial in the security parameter n,*
- *signatures consist of a vector in $\mathbb{Z}_q^{2m}$, and*
- *the (public) verification key consists of* two *matrices in $\mathbb{Z}_q^{n\times m}$.*

We note that this latter signature scheme, as induced by our IBE, is comparable in efficiency to the recent, compact signature scheme of Alperin-Sheriff [Alp15], but with somewhat better LWE/SIS parameters.

Scheme Type	# Matrices in PK	Security Type
PKE [GPV08]	1	IND-CPA
IBE [GPV08]	1	Adaptive (in the ROM)
IBE [CHKP10]	ℓ^2	Adaptive
IBE [ABB10]	2	Selective
IBE [ABB10]	ℓ	Adaptive
IBE – This paper	2	Adaptive

Table 1: Comparison of lattice PKE/IBE efficiency and security. Matrices are in $\mathbb{Z}_q^{n \times m}$, where $q = \mathsf{poly}(n)$ and – minimally – $\ell = \omega(1)$ (in n) is the identities' dimension. (Note that compressing the *number* of matrices to $O(1)$ with CRT is easy, but naively requires modulus $q = n^{\omega(1)}$.)

1.2 Our Techniques

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]. An initial observation is that the related notions of "admissible hash functions" and "abort-resistant hash functions" can be replaced generically by *pairwise independent hash functions* plus the *random isolation* technique of Valiant and Vazirani [VV85]. We use the freedom afforded by this insight to design a new encoding scheme for identities. To do so, we take a new conceptual view of the public parameters: 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* pairwise independent hash function $H_{\boldsymbol{h}}$ on their chosen identities during encryption.

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* $\mathbf{T_A}$ 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* $\mathbf{T_B}$ 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 $\boldsymbol{x} = (x_1, ..., x_\ell)$ according to [ABB10], the sender first constructs an identity-specific lattice, called the *target lattice* for $\boldsymbol{x}$,

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

by "mixing" a *long* public key $(\mathbf{A}, \mathbf{B}_1, ..., \mathbf{B}_\ell)$. 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 $\boldsymbol{h} = (h_1, ..., h_\ell)$. The identity encoding mechanism and hash are jointly designed so that the target lattice for identity $\boldsymbol{x}$, i.e. $[\mathbf{A} | \sum (-1)^{x_i} \mathbf{B}_i]$, becomes

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

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 SampleRight with $\mathbf{T_B}$ for every identity in the adversary's view, except the single challenge point $\boldsymbol{x}^*$.

In what follows, we will execute this same process, but with a *short* public key of only *two* matrices $(\mathbf{A}, \mathbf{B})$ (and a vector $\boldsymbol{u}$).

Getting to $\omega(\log(n))$ with a public CRHF. A first idea to compress the public key, due to folklore, is to compress any identity string $\boldsymbol{x} \in \{0,1\}^\ell$ (or even $\boldsymbol{x} \in \{0,1\}^*$) with a collision-resistant hash function. Since there is one matrix in the ABB public key for every bit of the identity, this asymptotically allows us to consider identities that look like $\boldsymbol{x} \in \{0,1\}^{\omega(\log(n))}$ for security parameter (and lattice row dimension) n. Therefore, the core of our task is to compress an ABB-like public key consisting of $\omega(\log(n))$ matrices in $\mathbb{Z}_q^{n\times m}$ into $O(1)$ matrices of same size.

Getting to $\omega(1)$ with algebraic identities. A simple idea in order to proceed further is to injectively map boolean identities $\boldsymbol{x} \in \{0,1\}^{\omega(\log(n))}$ to algebraic identities $\boldsymbol{x} \in \mathbb{Z}_{\mathsf{poly}(n)}^{\omega(1)}$ by simply interpreting the string $\boldsymbol{x}$ in a polynomial, rather than constant, base. If we do so, the corresponding target lattice of the real system becomes

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

for $x_i \in \mathbb{Z}_{\mathsf{poly}(n)}$, and the public key shrinks from $\omega(\log(n))$ matrices to $\omega(1)$ matrices.

Unfortunately, we cannot continue increasing the base to a superpolynomial (in order to achieve constant matrices in the public key), as this results in a *large* noise growth in the security proof causing SampleRight to fail. (Concretely, SampleRight requires that the matrix $\mathbf{R}$ contained in the target lattice's description has sufficiently small norm.) To see this, consider using this idea and then replacing the public key matrices $\mathbf{B}_i$ for $i \in [O(1)]$ with matrices $\mathbf{AR}_i + h_i\mathbf{B}$ as before. The target lattice in the security proof becomes

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

for $x_i \in \mathbb{Z}_{\mathsf{superpoly}(n)}$, which contains in its description the matrix $\mathbf{R} = \sum_{i\in[O(1)]} x_i \mathbf{R}_i$ of superpolynomial norm, which is too large.

A new perspective on the IBE public key from SIMD-style FHE. Our new observations start here. We begin by preparing the scheme for the transition to $O(1)$ matrices in the public key. As mentioned earlier, we draw a connection between the public key of our IBE system and the GSW-FHE scheme [GSW13]. To be more exact, we borrow an insight due to Hiromasa, Abe, and Okamoto [HAO15], who showed that by applying SIMD (Single Instruction Multiple Data) message-packing techniques to the GSW-FHE, the resulting scheme is *matrix-multiplication-homomorphic.* (Notably, their scheme requires a circular security assumption to enable publicly encrypting messages to this form; we will not require a public encryption mechanism, and so do not need to make the additional assumption.)

Ciphertexts in this SIMD GSW-FHE scheme can be viewed in the form $\mathbf{AR} + \mathbf{MG} \in \mathbb{Z}_q^{n\times m}$, where for modulus $q = \mathsf{poly}(n)$, $\mathbf{A} \in \mathbb{Z}_q^{n\times m}, \mathbf{R} \in \mathbb{Z}_q^{m\times m}$, where $\mathbf{M} \in \mathbb{Z}_q^{n\times n}$ contains a message vector $\mu \in \mathbb{Z}_q^n$ along the diagonal, and where $\mathbf{G} \in \mathbb{Z}_q^{n\times m}$ is the "gadget matrix" as first (explicitly) introduced in the work of Micciancio and Peikert [MP12]. The salient point is that there is an efficiently computable function $\mathbf{G}^{-1}$, so that

1. the desired homomorphism holds (roughly that $\mathsf{ct}_1 \cdot \mathbf{G}^{-1}(\mathsf{ct}_2) = \mathsf{ct}_\times$), and
2. $\mathbf{G}^{-1}(\mathbf{AR} + \mathbf{MG})$ is a matrix in $\{0,1\}^{m \times m}$, and thus has small norm.

This technique should be reminiscent of typical bit-decomposition techniques for readers who are familiar with recent FHE developments.

This scheme is also naturally equipped with a *matrix-multiplication-by-a-constant* operation of the following form:

$$(\mathbf{AR} + \mathbf{MG}) \cdot \mathbf{G}^{-1}(\mathbf{CG}) = \mathbf{AR} \cdot \mathit{small} + \mathbf{MCG}$$

for any diagonal matrices $\mathbf{C}, \mathbf{M} \in \mathbb{Z}_q^{n \times n}$. This leads us to consider a new, real-world IBE scheme where the (uniform) public key matrices $\mathbf{B}_i \in \mathbb{Z}_q^{n \times m}$ are "multiplied by a constant" $\mathbf{X}_i \in \mathbb{Z}_q^{n \times n}$, where the matrix $\mathbf{X}_i$ contains the i-th identity coordinate $x_i \in \mathbb{Z}_q$ on each coordinate of its diagonal, for $i = [\omega(1)]$. In other words, the target lattice in our real scheme becomes

$$\big[\mathbf{A} \bigm| \mathbf{Y}\big] = \left[\mathbf{A} \;\middle|\; \sum_{i=1}^{\omega(1)} \big(\mathbf{B}_i \cdot \mathbf{G}^{-1}\left(\mathbf{X}_i\mathbf{G}\right)\big)\right].$$

To match the above in the security proof, we would find a choice of hash key $\boldsymbol{h}$ and its matrix-encoding $\mathbf{H}_i \in \mathbb{Z}_q^{n \times n}$, then use the Leftover Hash Lemma to replace real-world public key matrices $\mathbf{B}_i$ with their ciphertext form $\mathbf{AR}_i + \mathbf{H}_i\mathbf{G}$. The target lattice in the security proof becomes

$$\begin{aligned}
\big[\mathbf{A} \bigm| \mathbf{Y}_{\mathsf{proof}}\big] &= \left[\mathbf{A} \;\middle|\; \sum_{i=1}^{\omega(1)} \big((\mathbf{AR}_i + \mathbf{H}_i\mathbf{G}) \cdot \mathbf{G}^{-1}\left(\mathbf{X}_i\mathbf{G}\right)\big)\right] \\
&= \left[\mathbf{A} \;\middle|\; \sum_{i=1}^{\omega(1)} \big(\mathbf{AR}_i \cdot \mathbf{G}^{-1}\left(\mathbf{X}_i\mathbf{G}\right) + \mathbf{H}_i\mathbf{X}_i\mathbf{G}\big)\right] \\
&= \left[\mathbf{A} \;\middle|\; \left(\sum_{i=1}^{\omega(1)} \mathbf{AR}_i \cdot \mathit{small}\right) + H_{\boldsymbol{h}}(\boldsymbol{x})\mathbf{G}\right].
\end{aligned}$$

A critical detail at this juncture is the observation that our *real* IBE system contains no homomorphic ciphertexts whatsoever. Indeed, these "ciphertexts" are introduced into the security proof by a purely *statistical* argument via the Leftover Hash Lemma. In order to ensure that a statistical indistinguishability argument via the Leftover Hash Lemma holds, it must be the case that the public key's "homomorphic ciphertexts" *cannot be decrypted — ever.* Otherwise, they will fail to be statistically close to the true public key of the real system, and will instead be only computationally close to the original public key, which is technically insufficient for our purposes. (We require that the adversary's view is *statistically* independent of the hash key used in the proof.)

Nonetheless by viewing these public keys as matrix-multiplication-homomorphic ciphertexts as above, we find that our security reduction can $\mathsf{SampleRight}$ using a trapdoor $\mathbf{T}_\mathbf{G}$ for the gadget matrix $\mathbf{G}$, whenever the hash output $H_{\boldsymbol{h}}(\boldsymbol{x}) := \sum \mathbf{H}_i\mathbf{X}_i$ is full-rank; i.e. whenever the gadget matrix $\mathbf{G}$ does not vanish.

Achieving a public key of two matrices. Our final step is to bring this $\omega(1)$ term down to $O(1)$. The intuition for the final step is that a pairwise independent hash function achieving the requirements of the ABB-type [ABB10] security proof can be computed by a "low-rank" matrix-multiplication operation of the form $\mathbf{H} \cdot \mathbf{X}$, where $\mathbf{H} \in \mathbb{Z}_q^{n \times n\ell}$ and $\mathbf{X} \in \mathbb{Z}_q^{\ell n \times n}$ for $\ell = \omega(1) \ll \log(n)$. (The reader may picture $\ell = \log\log(n)$.) The technical details of this hash function are found in Section 4.

In order to achieve a compact IBE public key, we will encode this computation in a single public key matrix $\mathbf{B}$ and a single multiply-by-constant $\mathbf{X}$ operation. Implementing this strategy (and aligning the matrices' dimensions) is somewhat subtle; we defer the details to Section 3. At a high level though, we will use a gadget matrix $\widehat{\mathbf{G}}$ and flattening function $\widehat{\mathbf{G}}^{-1}$ in *logarithmic base* plus the usual gadget matrix $\mathbf{G}$ so that:

$$\begin{aligned} \mathbf{B}_{\mathsf{real}} \cdot \widehat{\mathbf{G}}^{-1}(\mathbf{X} \cdot \mathbf{G}) &\overset{\mathsf{stat}}{\approx} \mathbf{B}_{\mathsf{proof}} \cdot \widehat{\mathbf{G}}^{-1}(\mathbf{X} \cdot \mathbf{G}) \\ &= (\mathbf{AR} + \mathbf{H} \cdot \widehat{\mathbf{G}}) \cdot \widehat{\mathbf{G}}^{-1}(\mathbf{X} \cdot \mathbf{G}) \\ &= \mathbf{AR} \cdot \widehat{\mathbf{G}}^{-1}(\mathbf{X} \cdot \mathbf{G}) + \mathbf{H} \cdot \mathbf{X} \cdot \mathbf{G} \\ &= \mathbf{AR} \cdot \mathit{small} + H_{\boldsymbol{h}}(\boldsymbol{x})\mathbf{G}, \end{aligned}$$

which implies that an ABB-type security proof will proceed as desired.

As such, our final real system contains only *two* matrices $(\mathbf{A}, \mathbf{B})$ (and a vector $\boldsymbol{u}$), and has target lattice

$$[\mathbf{A} \mid \mathbf{Y}] = \left[\mathbf{A} \,\middle|\, \mathbf{B} \cdot \widehat{\mathbf{G}}^{-1}(\mathbf{X} \cdot \mathbf{G})\right],$$

where $\mathbf{X} \in \mathbb{Z}_q^{\ell n \times n}$ encodes identity $\boldsymbol{x} \in \mathbb{Z}_q^{\ell}$ by n-fold repetition along ℓ diagonals.

This yields a fully-secure, compact IBE from LWE.

2 Preliminaries

Notation. 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 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.

2.1 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 the 4-tuple $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$:

The PPT Setup algorithm takes the security parameter λ, and outputs the master key pair $(\mathsf{mpk}, \mathsf{msk})$.

The PPT key generation algorithm KeyGen takes an identity id from identity space ID and the master secret key msk, then outputs a secret key $\mathsf{sk}_{\mathsf{id}}$ for the identity id.

The PPT encryption algorithm Enc encrypts message $\mu \in \mathcal{M}$ for some identity id under mpk.

The deterministic decryption algorithm Dec decrypts ciphertexts to messages μ' using the secret key $\mathsf{sk}_{\mathsf{id}}$.

Definition 2.1. *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* $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}$,

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

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 $\mathbf{Expt}_{\mathcal{A}}^{\mathsf{IBE}}(1^\lambda)$:

- **Setup**: A challenger runs the Setup algorithm, and sends the master public key mpk to the adversary.
- **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 $\mathcal{A}$.
- **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 $\mathcal{A}$.
- **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.
- **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

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

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

Definition 2.2. *We say an IBE scheme* Π *is* fully secure, *if for all* PPT *adversaries* $\mathcal{A}$, *we have*

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

2.2 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 $\mathbb{R}^{m+1}$ 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 2.3 ([ABB10]). *Regarding 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* $\mathsf{Pr}[||\mathbf{R}|| > \sigma\sqrt{m}] < e^{-2m}$.

2.3 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 2.4 ([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:*

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

2.4 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 2.5 (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 $\mathcal{O} \in \{\mathcal{O}_{\boldsymbol{s}}, \mathcal{O}_\$\}$):*

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

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

2.5 Two-Sided Trapdoors and Sampling Algorithms

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

Lemma 2.6 ([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*

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

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

Lemma 2.7 ([GPV08, CHKP10, ABB10])**.** *Let $q > 2, m > n$. There are two algorithms as follows:*

- *There is a* PPT *algorithm* $\mathsf{SampleLeft}(\mathbf{A}, \mathbf{B}, \mathbf{T_A}, \boldsymbol{u}, s)$*: It takes as input*
  - *a rank-n matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, and any matrix $\mathbf{B} \in \mathbb{Z}_q^{n\times m_1}$,*
  - *a "short" basis $\mathbf{T_A}$ for lattice $\Lambda_q^\perp(\mathbf{A})$, a vector $\boldsymbol{u} \in \mathbb{Z}_q^n$,*
  - *and 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 $\mathbf{F} := (\mathbf{A}|\mathbf{B})$.

- *There is a* PPT *algorithm* $\mathsf{SampleRight}(\mathbf{A}, \mathbf{B}, \mathbf{R}, \mathbf{T_B}, \boldsymbol{u}, s)$*: It takes as input*
  - *any matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, and a rank-$n$ matrix $\mathbf{B} \in \mathbb{Z}_q^{n\times m}$,*
  - *a matrix $\mathbf{R} \in \mathbb{Z}_q^{m\times m}$, where $s_\mathbf{R} := ||\mathbf{R}|| = \sup_{\boldsymbol{x}:||\boldsymbol{x}||=1} ||\mathbf{R}\boldsymbol{x}||$,*
  - *a "short" basis $\mathbf{T_B}$ for lattice $\Lambda_q^\perp(\mathbf{B})$, a vector $\boldsymbol{u} \in \mathbb{Z}_q^n$,*
  - *and a Gaussian parameter $s > ||\widetilde{\mathbf{T_B}}|| \cdot s_\mathbf{R} \cdot \omega(\sqrt{\log m})$,*

  then outputs a vector $\boldsymbol{r} \in \mathbb{Z}^{2m}$ distributed statistically close to $\mathcal{D}_{\Lambda_q^{\boldsymbol{u}}(\mathbf{F}),s}$ where $\mathbf{F} := (\mathbf{A}|\mathbf{AR} + \mathbf{B})$.

3 Gadget Matrices for Lattices, and Gadget-Based Matrix Operations

In this section, we first review the gadget matrix $\mathbf{G}$ technique proposed by Micciancio and Peikert [MP12], and show that this matrix $\mathbf{G}$ can be tweaked to support "invariant-preserving" pseudo-commutative matrix multiplication operations. Then, we generalize matrix $\mathbf{G}$ and its trapdoor to other integer powers or mixed-integer products. At the end of this section, we explore a new arrangement of matrix operations for the generalized gadget matrices that is critical to our main result.

3.1 The Gadget Matrix $\mathbf{G}$

Micciancio and Peikert [MP12] introduced a *universal* structured (primitive) matrix $\mathbf{G}$, typically in $\mathbb{Z}_q^{n\times\Omega(n\log q)}$, also known as the "gadget matrix." The (typical) gadget matrix $\mathbf{G}$ has an associated, *public* trapdoor basis $\mathbf{T_G}$ that is "short" – i.e. $||\widetilde{\mathbf{T_G}}|| \in (\sqrt{5}, 5)$, depending on the factorization of the modulus q.

Consider a basic matrix $\mathbf{G}_{basic} := \boldsymbol{g}^T \otimes \mathbf{I}_n \in \mathbb{Z}_q^{n\times nk}$, for row $\boldsymbol{g}^T = \left[1, 2, 2^2, ..., 2^{k-1}\right] \in \mathbb{Z}_q^{1\times k}$ and for $k = \lceil \log_2 q \rceil$. For arbitrary $m \geq nk$, we can use a padded matrix, i.e. $\mathbf{G} = [\mathbf{G}_{basic} \mid \mathbf{0}] \in \mathbb{Z}_q^{n\times m}$, where $\mathbf{0}$ is a zero matrix in $\mathbb{Z}_q^{n\times(m-nk)}$. Since the columns of $\mathbf{G}_{basic}$ generates all of $\mathbb{Z}_q^n$, so do the columns of $\mathbf{G}$. Therefore, by the extended basis technique of the work [CHKP10], we are able to derive an associated public trapdoor basis for the padded matrix $\mathbf{G}$ such that $||\widetilde{\mathbf{T_G}}|| = ||\widetilde{\mathbf{T}_{\mathbf{G}_{basic}}}|| \in (\sqrt{5}, 5)$ as well.

Notably, the shortness of $\mathbf{T_G}$ implies that it is a useful input to the SampleRight algorithm to generate short vectors $\boldsymbol{r}$ in the $\boldsymbol{u}$-coset of lattices of type

$$\left\{\Lambda_q^{\boldsymbol{u}}(\mathbf{A}|\mathbf{AR} + \mathbf{MG})\right\}_{\mathbf{A},\mathbf{R},\mathbf{M},\boldsymbol{u}}$$

for $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, $\boldsymbol{u} \in \mathbb{Z}_q^n$, "short" $\mathbf{R} \in \mathbb{Z}_q^{m\times m}$ (c.f. Lemma 2.7), and rank-n $\mathbf{M} \in \mathbb{Z}_q^{n\times n}$.

3.2 (Pseudo-Commutative) Matrix Operations with $\mathbf{G}$ via the Flattening Function $\mathbf{G}^{-1}$

Over general lattices, the only matrices that commute with $\mathbf{G} \in \mathbb{Z}_q^{n\times m}$ are scaled identity matrices $\alpha\mathbf{I}$, in the sense that

$$\mathbf{G} \cdot (\alpha\mathbf{I}_m) = (\alpha\mathbf{I}_n) \cdot \mathbf{G}.$$

Note that if the $\mathbf{G}$ here is padded (from the matrix $\mathbf{G}_{basic}$ as above), then $\alpha\mathbf{I}_m$ could alternatively be the block matrix $\mathbf{A}$ containing $\alpha\mathbf{I}_n$ in the appropriate quadrant with zeroes everywhere else.

The works of Xagawa [Xag13] and Alperin-Sheriff and Peikert [AP14], among others, describe a technique (also implicit in the earlier dimension-modulus trade-off ideas of homomorphic encryption, e.g. [BV11]) that resolves this non-commutativity problem for $\mathbf{G}$. In particular, there is an efficiently computable function $\mathbf{G}^{-1}$ so that for any matrix $\mathbf{B} \in \mathbb{Z}_q^{n\times m}$, so that $\mathbf{G}^{-1}(\mathbf{B}) = \mathbf{X} \in \{0,1\}^{m\times m}$ and $\mathbf{GX} = \mathbf{B}$. This allows for "pseudo-commutative" multiplication of the gadget matrix (resp. matrices) $\mathbf{G}$ by any square matrix $\mathbf{M}$ of dimension n, by observing that

$$\mathbf{G} \cdot \left(\mathbf{G}^{-1}(\mathbf{MG})\right) = \mathbf{M} \cdot \mathbf{G}.$$

We note that the matrix $\mathbf{X} = \mathbf{G}^{-1}(\mathbf{B})$ has small norm independent of $\mathbf{B}$, and refer the reader to the representative works for further details.

3.3 Non-Binary Gadgets $\mathbf{G}_{n,b,m}$, and Batch Change-of-Base $\mathbf{G}^{-1}_{n',b',m'}(\cdot)$

As mentioned by [MP12], their results for $\mathbf{G}$ 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 $2 \leq b \leq q$, let $\boldsymbol{g}_b^T := \left[1, b, b^2, ..., b^{k_b-1}\right] \in \mathbb{Z}_q^{1\times k_b}$ for $k_b = \lceil \log_b q \rceil$. (Note that the typical base-2 $\boldsymbol{g}^T$ is $\boldsymbol{g}_2^T$.) For row dimension n and b as before, we let $\mathbf{G}_{n,b} =$

$\mathbf{g}_b^T \otimes \mathbf{I}_n \in \mathbb{Z}_q^{n \times nk_b}$. The public trapdoor basis $\mathbf{T}_{\mathbf{G}_{n,b}}$ is given analogously. Similar to the above padding argument, $\mathbf{G}_{n,b} \in \mathbb{Z}_q^{n \times nk_b}$ can be padded into a matrix $\mathbf{G}_{n,b,m} \in \mathbb{Z}_q^{n \times m}$ for $m \geq nk_b$ without increasing the norm of $\widetilde{\mathbf{T}_{\mathbf{G}_{n,b,m}}}$ from that of $\widetilde{\mathbf{T}_{\mathbf{G}_{n,b}}}$.

For this paper, we do not need to use $\widetilde{\mathbf{T}_{\mathbf{G}_{n,b}}}$ or $\widetilde{\mathbf{T}_{\mathbf{G}_{n,b,m}}}$ at all, but we keep the discussion for exposition. We also mention that under this notation, the typical $\mathbf{G}$ in dimension n is either $\mathbf{G}_{n,2} \in \mathbb{Z}_q^{n \times n \log_2 q}$ or its padded version $\mathbf{G}_{n,2,m} \in \mathbb{Z}_q^{n \times m}$ depending on the setting.

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

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

3.4 Further Gadget-Based Matrix Multiplication Operations

In what follows, we explore a new arrangement of matrix operations (pseudo-commutative and non-commutative) that is critical to our main result in this paper. First, fix any integer n. Then, for some sufficiently small $\ell = \omega(1) \ll \log_2(q) \approx \log_2(n)$, define $\ell' := 2^\ell$ such that $\ell' = \omega(1) < \log_2(q) \approx \log_2(n)$. (We remark that in principle ℓ' may be made an arbitrarily small super-constant function of the security parameter n, e.g. $\ell' \approx \log^*(n)$.) Finally, fix any integer $m \geq n\ell k_{\ell'}$.

Then for any two matrices $\mathbf{H} \in \mathbb{Z}_q^{n \times n\ell}$, $\mathbf{X} \in \mathbb{Z}_q^{\ell n \times n}$, consider the following terms, in order:

1. First, consider the base-2, dimension-n "gadget-encoding" of $\mathbf{X} \in \mathbb{Z}_q^{\ell n \times n}$, i.e. the matrix

$$\mathbf{X} \cdot \mathbf{G}_{n,2,m} \in \mathbb{Z}_q^{\ell n \times m} = \mathbb{Z}_q^{n \log_2(\ell') \times m}.$$

2. Next, consider the base-ℓ', dimension-$(n\ell)$ flattening (with zero-row padding) of the above:

$$\mathbf{G}_{n\ell,\ell',m}^{-1}\left(\mathbf{X} \cdot \mathbf{G}_{n,2,m}\right) \in \{0, 1, ..., \ell'-1\}^{m \times m} \subsetneq \mathbb{Z}_q^{m \times m}.$$

3. Then, consider the base-ℓ', dimension-$(n\ell)$ gadget-encoding of $\mathbf{H} \in \mathbb{Z}_q^{n \times n\ell}$, i.e. the matrix

$$\mathbf{H} \cdot \mathbf{G}_{n\ell,\ell',m} \in \mathbb{Z}_q^{n \times m}.$$

4. Finally — for sufficiently large m, we find the following relationship holds:

$$\left(\mathbf{H} \cdot \mathbf{G}_{n\ell,\ell',m}\right) \cdot \left(\mathbf{G}_{n\ell,\ell',m}^{-1}\left(\mathbf{X} \cdot \mathbf{G}_{n,2,m}\right)\right) = \left(\mathbf{H} \cdot \mathbf{X}\right) \cdot \mathbf{G}_{n,2,m} \in \mathbb{Z}_q^{n \times m}$$

with $\left|\left|\mathbf{G}_{n\ell,\ell',m}^{-1}\left(\mathbf{X} \cdot \mathbf{G}_{n,2,m}\right)\right|\right| = \mathsf{small}$, conditioned on the sufficiently small choice of $\ell = \omega(1)$. We emphasize that only *public information*, n, m, ℓ, is required to perform this batch base-change-then-multiply operation, when given as input any $\mathbf{M} \in \mathbb{Z}_q^{n \times m}$ (equal to $\mathbf{H} \cdot \mathbf{G}_{n\ell,\ell',m} \in \mathbb{Z}_q^{n \times m}$) and any $\mathbf{X} \in \mathbb{Z}_q^{\ell n \times n}$.

Definition 3.1. *We refer to the matrix* $\mathbf{H} \cdot \mathbf{G}_{n\ell,\ell',m} \in \mathbb{Z}_q^{n \times m}$ *as the* predicate-encoding of $\mathbf{H} \in \mathbb{Z}_q^{n \times n\ell}$, *and to the matrix* $\mathbf{G}_{n\ell,\ell',m}^{-1}\left(\mathbf{X} \cdot \mathbf{G}_{n,2,m}\right) \in \{0, 1, ..., \ell'-1\}^{m \times m} \subsetneq \mathbb{Z}_q^{m \times m}$ *as the* input-encoding of $\mathbf{X} \in \mathbb{Z}_q^{\ell n \times n}$.

4 Hashing, Encoding, and Randomly Isolating Unique Solutions

In this section, we construct a hash function family $\mathcal{H}^*$ that will be used in our main security proof, and prove that it satisfies certain properties developed along the way. Toward this end, we first show that all pairwise independent hash functions allow for "nice, random isolations" following Valiant and Vazirani. Then we describe an intermediate, pairwise independent hash function family $\mathcal{H}$ used to build $\mathcal{H}^*$ later. Next, we recall a notion of embedding vectors into invertible matrices due to Cramer and Damgård (also used regularly in the lattice-based FE literature). Finally, we present the hash family $\mathcal{H}^*$, and prove that it both randomly isolates nicely, and has either invertible or zero output.

To begin, we recall the definition of pairwise independent hash function families.

Definition 4.1 (Pairwise Independent Hash Functions)**.** *A family of functions $\mathcal{H} = \{H : \mathcal{X} \to \mathcal{Y}\}$ is called a* family of pairwise independent hash functions *if for all $x_1 \neq x_2 \in \mathcal{X}$ and $y_1, y_2 \in \mathcal{Y}$, it holds that*

$$\mathsf{Pr}_{H\in\mathcal{H}}[H(x_1) = y_1 \wedge H(x_2) = y_2] = \frac{1}{|\mathcal{Y}|^2}$$

4.1 Isolations of Random Hash Functions

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 $\mathsf{NP} = \mathsf{RP}$. Their proof relies on a key technical lemma regarding 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 $2^{k-2} \leq |Q| \leq 2^{k-1}$, it holds that $\mathsf{Pr}_{H\in\mathcal{H}}\left[\left|H^{-1}(\mathbf{0}) \cap Q\right| = 1\right] \geq \frac{1}{8}$.

We show a similar lemma, but for $|Q|$ based on the range $\mathcal{Y}$ of an arbitrary, pairwise independent $\mathcal{H}$:

Lemma 4.2. *For any pairwise independent hash function family $\mathcal{H} : \mathcal{X} \to \mathcal{Y}$, for any $\delta \in (0,1)$, for any fixed set $Q \subseteq \mathcal{X}$ such that $|Q| \leq \delta|\mathcal{Y}|$, and any element $x \in \mathcal{X} \setminus Q$, we have*

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

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

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

where we get the second equation from the pairwise independent property of hash function H, and the inequality from the union bound.

For the upper bound, we have simply

$$\begin{aligned}&\mathsf{Pr}_{H\in\mathcal{H}}[H(x)=0\wedge H(x')\neq 0,\forall x'\in Q]\\ \leq&\mathsf{Pr}_{H\in\mathcal{H}}[H(x)=0]=\frac{1}{|\mathcal{Y}|}.\end{aligned}$$

This completes the proof. □

Remark 4.3. *We observe that in applications, the exact "probability gap" in the above lemma can be tuned smaller (or larger) by changing the size of the application's hash function's range $|\mathcal{Y}|$ (conditioned on fixed query set size $|Q|$). We give a concrete instantiation of this idea in our hash functions that follow.*

Remark 4.4. *In prior works, e.g. [Wat05, HK12, BR09, ABB10], the property of Lemma 4.2 is known as "abort-resistance," due to its usefulness in combination with the artificial abort proof technique. This is somewhat detacted from the idea of pairwise independent hashing. In the spirit of [VV85], we take an alternate perspective, and simply say that "pairwise independent hash functions* randomly isolate *nicely."*

4.2 An Intermediate Hash Function Family

We define a hash function family $\mathcal{H}':\mathcal{X}\to\mathcal{Y}$, where $\mathcal{X}=\mathbb{Z}_q^\ell,\mathcal{Y}=\mathbb{Z}_q^t$, which is either used by each of the representative works in the above remark, or is a variation thereof. Each function of $\mathcal{H}'$ is indexed by a set of integer vectors $\boldsymbol{h}=\{\boldsymbol{h}_i\}_{i\in\{0,\ldots,\ell\}}$ where each $\boldsymbol{h}_i\in\mathbb{Z}_q^t$. Then for $\boldsymbol{x}=(x_1,\ldots,x_\ell)\in\mathbb{Z}_q^\ell$, define

$$H'_{\boldsymbol{h}}(\boldsymbol{x})=\boldsymbol{h}_0+\sum_{i\in[\ell]}x_i\boldsymbol{h}_i.$$

We show that this hash function family is a pairwise independent hash function family.

Lemma 4.5. *Let q be a prime. For $\boldsymbol{h}=(\boldsymbol{h}_0,...,\boldsymbol{h}_\ell)\in(\mathbb{Z}_q^t)^{\ell+1}$, the collection $\mathcal{H}'_{\boldsymbol{h}}:\mathbb{Z}_q^\ell\to\mathbb{Z}_q^t$ defined above is a pairwise independent hash function family.*

Proof. Let $\boldsymbol{x}_1\neq\boldsymbol{x}_2\in\mathbb{Z}_q^\ell$ and $\boldsymbol{y}_1,\boldsymbol{y}_2\in\mathbb{Z}_q^t$. Writing each vector $\boldsymbol{h}_i$ of the hash key $\boldsymbol{h}$ as a row, we have

$$\begin{aligned}&\Pr_{\boldsymbol{h}=\boldsymbol{h}_0,\ldots,\boldsymbol{h}_\ell}\left[H'_{\boldsymbol{h}}(\boldsymbol{x}_1)=\boldsymbol{y}_1\wedge H'_{\boldsymbol{h}}(\boldsymbol{x}_2)=\boldsymbol{y}_2\right]\\ &=\Pr_{\boldsymbol{h}}\left[\begin{bmatrix}1 & x_{1,1} & ... & x_{1,\ell}\\ 1 & x_{2,1} & ... & x_{2,\ell}\end{bmatrix}\begin{bmatrix}h_{0,1} & ... & h_{0,t}\\ \vdots & \ddots & \vdots\\ h_{\ell,1} & ... & h_{\ell,t}\end{bmatrix}=\begin{bmatrix}y_{1,1} & ... & y_{1,t}\\ y_{2,1} & ... & y_{2,t}\end{bmatrix}\right]\\ &=\prod_{j\in[t]}\left(\Pr_{h_{0,j},\ldots,h_{\ell,j}}\left[\begin{bmatrix}1 & x_{1,1} & ... & x_{1,\ell}\\ 1 & x_{2,1} & ... & x_{2,\ell}\end{bmatrix}\begin{bmatrix}h_{0,j}\\ \vdots\\ h_{\ell,j}\end{bmatrix}=\begin{bmatrix}y_{1,j}\\ y_{2,j}\end{bmatrix}\right]\right)\end{aligned}$$

where the second equality follows from the facts that each coordinate of $\boldsymbol{h}$ is chosen i.i.d. and that the t sub-systems are distinct. Observe that each sub-system is a linear transformation of the $h_{0,j},...,h_{\ell,j}$.

Since $\mathsf{rank}\left(\begin{bmatrix}1 & x_{1,1} & ... & x_{1,\ell}\\ 1 & x_{2,1} & ... & x_{2,\ell}\end{bmatrix}\right)=2$, we have

$$\Pr_{h_{0,j},\ldots,h_{\ell,j}}\left[\begin{bmatrix}1 & x_{1,1} & ... & x_{1,\ell}\\ 1 & x_{2,1} & ... & x_{2,\ell}\end{bmatrix}\begin{bmatrix}h_{0,j}\\ \vdots\\ h_{\ell,j}\end{bmatrix}=\begin{bmatrix}y_{1,j}\\ y_{2,j}\end{bmatrix}\right]=1/q^2,$$

which gives

$$\Pr_{\boldsymbol{h}=\boldsymbol{h}_0,\ldots,\boldsymbol{h}_\ell}\left[H'_{\boldsymbol{h}}(\boldsymbol{x}_1)=\boldsymbol{y}_1 \wedge H'_{\boldsymbol{h}}(\boldsymbol{x}_2)=\boldsymbol{y}_2\right]=1/q^{2t}=1/|\mathcal{Y}|^2$$

as required. □

4.3 The Matrix Embedding M of $\mathbb{F}^t$ into $\mathbb{F}^{t\times t}$

In their work on the amortized complexity of zero knowledge, Cramer and Damgård [CD09] describe an encoding function that maps a superpolynomially-sized domain $\mathbb{F}^t$ to matrices in $\mathbb{F}^{t\times t}$ with certain, strongly injective properties. (We state the definition of M in the language of general fields, but will later always take $\mathbb{F}=\mathbb{Z}_q$.) This encoding notion has been repurposed (and updated) in works such as [ABB10, AFV11] under the name "Full-Rank Difference encoding," or more recently in [Xag13, Alp15] under the name "Invertible Difference encoding." We refer to this function as the Matrix Embedding M, describe it, and show the properties we need of it.

The goal in designing M is to construct an additive subgroup $\mathbb{G}$ of $\mathbb{F}^{t\times t}$ of size q^t with all non-zero matrices in $\mathbb{G}$ of full-rank. For a polynomial $g\in\mathbb{F}[X]$ of degree at most $t-1$, $\mathsf{coeff}(g)\in\mathbb{F}^{1\times t}$ is the t-row of coefficients of g (padding zeroes on the right, as needed). Let f be some polynomial of degree t, irreducible in $\mathbb{F}[X]$. Then for $g\in\mathbb{F}[X]$, the polynomial $g \bmod f$ has degree at most $t-1$, so $\mathsf{coeff}(g \bmod f)\in\mathbb{F}^t$.

For any field $\mathbb{F}$, any integer $t\geq 2$, and input $\boldsymbol{h}=(h_0,...,h_{t-1})\in\mathbb{F}^t$, define $g_{\boldsymbol{h}}(X)=\sum_{i=0}^{t-1}h_i x^i\in\mathbb{F}[X]$. Then let the matrix-embedding function $M_{t,q}:\mathbb{F}^t\to\mathbb{F}^{t\times t}$ be defined as follows:

$$M_{t,q}(\boldsymbol{h}):=\begin{bmatrix}\mathsf{coeff}(g_{\boldsymbol{h}} \bmod f)\\ \mathsf{coeff}(X\cdot g_{\boldsymbol{h}} \bmod f)\\ \mathsf{coeff}(X^2\cdot g_{\boldsymbol{h}} \bmod f)\\ \vdots\\ \mathsf{coeff}(X^{t-1}\cdot g_{\boldsymbol{h}} \bmod f)\end{bmatrix}\in\mathbb{F}^{t\times t}.$$

From here on, we will take $\mathbb{F}:=\mathbb{Z}_q$ for q prime. It is straightforward to verify that the matrix embedding function $M:=M_{t,q}:\mathbb{Z}_q^t\to\mathbb{Z}_q^{t\times t}$ obeys the following (expected) properties:

Fact 4.6 (M is linear)**.** *$M(a\boldsymbol{h}_1+b\boldsymbol{h}_2)=aM(\boldsymbol{h}_1)+bM(\boldsymbol{h}_2)$ for any $a,b\in\mathbb{Z}_q,\boldsymbol{h}_1,\boldsymbol{h}_2\in\mathbb{Z}_q^t$.*

Fact 4.7 (The image of M is invertible or zero)**.** *For any vector $\boldsymbol{h}\neq\mathbf{0}$, $M(\boldsymbol{h})$ is invertible, and $M(\mathbf{0})=\mathbf{0}$.*

4.4 Our Main Hash Function Family

We describe our main hash function family $\mathcal{H}^*_{\ell,t,n,q}:\mathbb{Z}_q^\ell\to\mathbb{Z}_q^{n\times n}$ for integers ℓ,t,n,q. Each function of $\mathcal{H}^*_{\ell,t,n,q}$ is indexed by a set of integer vectors $\boldsymbol{h}=\{\boldsymbol{h}_i\}_{i\in\{0,\ldots,\ell\}}$ where each $\boldsymbol{h}_i\in\mathbb{Z}_q^t$. For $\boldsymbol{x}=(x_1,\ldots,x_\ell)\in\mathbb{Z}_q^\ell$,

$$H^*_{\boldsymbol{h}}(\boldsymbol{x})=M_{t,q}(\boldsymbol{h}_0)\otimes\mathbf{I}_{n/t}+\sum_{i\in[\ell]}x_iM_{t,q}(\boldsymbol{h}_i)\otimes\mathbf{I}_{n/t}.$$

The hash function above is similar to one hash function used in the work [ABB], except that they use $\mathbf{I}_n$ for the first term, and we use $M_{t,q}(\boldsymbol{h}_0)\otimes\mathbf{I}_{n/t}$. While it is possible to directly prove that their hash function has the desired isolation property (as the work [ABB] did), we consider the modified construction that achieves a simpler and more modular analysis – we will use the intermediate pairwise independent hash function described in Section 4.2 and the embedding properties in Section 4.3 in a modular way.

In our subsequent IBE security proof, we will "encode" the family $\mathcal{H}^*$ in such a way that (ℓ,n,q) are public, but the exact choice of $t\in[n]$, and thus the *order* of the implicit additive subgroup $\mathbb{G}$ underlying

$\mathbb{F}^{t\times t}$ embedded in $\mathbb{F}^{n\times n}$, will be "hidden" by an external statistical argument. Indeed, our security proof (in the sequel) hinges on the ability to hide the calculation of the following theorem from IBE adversaries holding a given PK.

Now, we show that the (plain) hash function family $\mathcal{H}^*$ has the following, nice isolation property.

Theorem 4.8. *For any integers ℓ, t, n, and a prime q, let $\mathcal{H}^*_{\ell,t,n,q}$ be the hash function family defined above. Then for any $\delta \in (0,1)$, for any fixed set $Q \subseteq \mathbb{Z}_q^\ell$ such that $|Q| \leq \delta q^t$, and any $\boldsymbol{x} \in \mathbb{Z}_q^\ell \setminus Q$, we have*

$$\Pr_{\boldsymbol{h}\leftarrow(\mathbb{Z}_q^t)^{\ell+1}}\left[\begin{array}{c} H^*_{\boldsymbol{h}}(\boldsymbol{x}) = \boldsymbol{0} \ \bigwedge \\ \forall \boldsymbol{x}' \in Q : H^*_{\boldsymbol{h}}(\boldsymbol{x}') \textit{ is full rank} \end{array}\right] \in \left(\frac{1-\delta}{q^t}, \frac{1}{q^t}\right)$$

Proof of Theorem 4.8. We consider another intermediate hash family $\bar{\mathcal{H}}_{\ell,t,n,q} : \mathbb{Z}_q^\ell \to \mathbb{Z}_q^{t\times t}$, where each function of the family is indexed by $\boldsymbol{h}_i \in \mathbb{Z}_q^t$. For $\boldsymbol{x} = (x_1, \ldots, x_\ell) \in \mathbb{Z}_q^\ell$,

$$\bar{H}_{\boldsymbol{h}}(\boldsymbol{x}) = M_{t,q}(\boldsymbol{h}_0) + \sum_{i\in[\ell]} x_i M_{t,q}(\boldsymbol{h}_i) \in \mathbb{Z}_q^{t\times t}.$$

Basically, $\bar{\mathcal{H}}_{\ell,t,n,q}$ is the same as $\mathcal{H}^*_{\ell,t,n,q}$ except it does not compute the tensor product. By a simple observation, we have (1) $H^*_{\boldsymbol{h}}(\boldsymbol{x}) = \boldsymbol{0}$ if and only if $\bar{H}_{\boldsymbol{h}}(\boldsymbol{x}) = \boldsymbol{0}$, and (2) $H^*_{\boldsymbol{h}}(\boldsymbol{x})$ is full rank if and only if $\bar{H}_{\boldsymbol{h}}(\boldsymbol{x})$ is full rank. These facts imply that

$$\Pr_{\boldsymbol{h}\leftarrow(\mathbb{Z}_q^t)^{\ell+1}}\left[\begin{array}{c} H^*_{\boldsymbol{h}}(\boldsymbol{x}) = \boldsymbol{0} \ \bigwedge \\ \forall \boldsymbol{x}' \in Q : H^*_{\boldsymbol{h}}(\boldsymbol{x}') \text{ is full rank} \end{array}\right] = \Pr_{\boldsymbol{h}\leftarrow(\mathbb{Z}_q^t)^{\ell+1}}\left[\begin{array}{c} \bar{H}_{\boldsymbol{h}}(\boldsymbol{x}) = \boldsymbol{0} \ \bigwedge \\ \forall \boldsymbol{x}' \in Q : \bar{H}_{\boldsymbol{h}}(\boldsymbol{x}') \text{ is full rank} \end{array}\right].$$

Define $H'_{\boldsymbol{h}}(\boldsymbol{x}) = \boldsymbol{h}_0 + \sum_{i\in[\ell]} x_i \boldsymbol{h}_i$ as in Subsection 4.2. By the linearity of $M_{t,q}$ (Fact 4.6), we can rewrite $\bar{H}_{\boldsymbol{h}}(\boldsymbol{x})$ as $M_{t,q}\left(\boldsymbol{h}_0 + \sum_{i\in[\ell]} x_i \boldsymbol{h}_i\right) = M_{t,q}(H'_{\boldsymbol{h}}(\boldsymbol{x}))$. By Fact 4.7, we know that (1) $\bar{H}_{\boldsymbol{h}}(\boldsymbol{x}) = \boldsymbol{0}$ if and only if $H'_{\boldsymbol{h}}(\boldsymbol{x}) = 0$, and (2) $\bar{H}_{\boldsymbol{h}}(\boldsymbol{x})$ is full rank if and only if $H'_{\boldsymbol{h}}(\boldsymbol{x}) \neq 0$. Therefore, we have:

$$\Pr_{\boldsymbol{h}\leftarrow(\mathbb{Z}_q^t)^{\ell+1}}\left[\begin{array}{c} \bar{H}_{\boldsymbol{h}}(\boldsymbol{x}) = \boldsymbol{0} \ \bigwedge \\ \forall \boldsymbol{x}' \in Q : \bar{H}_{\boldsymbol{h}}(\boldsymbol{x}') \text{ is full rank} \end{array}\right] = \Pr_{\boldsymbol{h}\leftarrow(\mathbb{Z}_q^t)^{\ell+1}}\left[\begin{array}{c} H'_{\boldsymbol{h}}(\boldsymbol{x}) = 0 \ \bigwedge \\ \forall \boldsymbol{x}' \in Q : H'_{\boldsymbol{h}}(\boldsymbol{x}') \neq 0 \end{array}\right].$$

By Lemma 4.5, we know the hash functions $\{H'_{\boldsymbol{h}}\}_{\boldsymbol{h}\in(\mathbb{Z}_q^t)^{\ell+1}}$ form a pairwise independent family mapping from $\mathbb{Z}_q^\ell$ to $\mathbb{Z}_q^t$. Then finally we apply Lemma 4.2 to obtain:

$$\Pr_{\boldsymbol{h}\leftarrow(\mathbb{Z}_q^t)^{\ell+1}}\left[\begin{array}{c} H'_{\boldsymbol{h}}(\boldsymbol{x}) = 0 \ \bigwedge \\ \forall \boldsymbol{x}' \in Q : H'_{\boldsymbol{h}}(\boldsymbol{x}') \neq 0 \end{array}\right] \in \left(\frac{1-\delta}{q^t}, \frac{1}{q^t}\right).$$

This concludes the proof of the theorem. □

5 Compact Fully-Secure Identity-Based Encryption from LWE

Now, we present our compact IBE scheme, its correctness proof, and its parameter settings. The proof of full security follows in the sequel. Let the message space be $\mathcal{M} = \{0,1\}$, and the identity space be $\mathsf{ID} = \mathbb{Z}_q^\ell$ for $q = \mathsf{poly}(n)$ and slightly super-constant $\ell = \omega(1) \ll \log_2(n)$. As discussed in the introduction, we can use the technique of collision resistant hash functions, so that it is without loss of generality to consider ID spaces of super-polynomial-size. We write identities $\boldsymbol{x}$ and messages μ.

As discussed in the work [ABB], the message space $\mathcal{M}$ can easily be extended from bits to bit-strings. For technical reasons, we need to either set the first coordinate of each identity $\boldsymbol{x}$ to 1, or exclude the all-zeroes identity; we choose the former for ease of overall presentation.

5.1 Our Construction

Our identity-based encryption scheme $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is as follows:

$\mathsf{Setup}(1^\lambda)$: On input the security parameter λ, the setup algorithm does:

1. Set lattice parameters $n = n(\lambda), m = m(\lambda), q = q(\lambda)$, Gaussian parameter $s = s(\lambda)$, and the dimension ℓ of identity space $\mathsf{ID} = \{1\} \times \mathbb{Z}_q^{\ell-1}$ as specified in Section 5.3, and define $\ell' = 2^\ell$.
2. Generate a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ associated with its trapdoor $\mathbf{T_A}$, using algorithm $\mathsf{TrapGen}(q, n, m)$.
3. Sample a uniformly random matrix $\mathbf{B} \in \mathbb{Z}_q^{n\times m}$, and a random vector $\boldsymbol{u} \in \mathbb{Z}_q^n$.
4. Output master public key mpk and master secret key msk as

$$\mathsf{mpk} = (\mathbf{A}, \mathbf{B}, \boldsymbol{u}), \qquad \mathsf{msk} = \mathbf{T_A}$$

$\mathsf{KeyGen}(\mathsf{mpk}, \mathsf{msk}, \boldsymbol{x})$: On input the master secret key msk and identity $\boldsymbol{x} = (1, x_1, ..., x_{\ell-1}) \in \{1\} \times \mathbb{Z}_q^{\ell-1}$, the key generation algorithm does:

1. Define the input-encoding matrices

$$\mathbf{X}' := \begin{bmatrix} \mathbf{I}_n \\ x_1\mathbf{I}_n \\ x_2\mathbf{I}_n \\ \vdots \\ x_{\ell-1}\mathbf{I}_n \end{bmatrix} \in \mathbb{Z}_q^{\ell n\times n}, \quad \text{and} \quad \mathbf{X} := \mathbf{G}^{-1}_{n\ell,\ell',m}\left(\mathbf{X}' \cdot \mathbf{G}_{n,2,m}\right) \in [\ell']^{m\times m}.$$

2. Set the ID matrix for $\boldsymbol{x}$ to be

$$\mathbf{Y} := \mathbf{B} \cdot \mathbf{X} \in \mathbb{Z}_q^{n\times m}.$$

3. Compute short vector $\boldsymbol{r} \in \mathbb{Z}_q^{2m}$, using

$$\boldsymbol{r} \leftarrow \mathsf{SampleLeft}(\mathbf{A}, \mathbf{T_A}, \mathbf{Y}, \boldsymbol{u}, s)$$

such that $[\mathbf{A}|\mathbf{Y}] \cdot \boldsymbol{r} = \boldsymbol{u} \bmod q$.

4. Output $\mathsf{sk}_{\boldsymbol{x}} := \boldsymbol{r}$.

$\mathsf{Enc}(\mathsf{mpk}, \boldsymbol{x}, \mu)$: On input the master public key mpk, an identity $\boldsymbol{x} = (1, x_2, ..., x_\ell)$ and message μ, the encryption algorithm does:

1. Define the input-encoding matrix $\mathbf{X}$ and the ID matrix $\mathbf{Y}$ for $\boldsymbol{x}$ as in KeyGen.
2. Choose a uniformly random vector $\boldsymbol{s} \in \mathbb{Z}_q^n$, an error vector $\boldsymbol{e}_0 \leftarrow \mathcal{D}_{\mathbb{Z}_q^m,\alpha}$ and an error integer $e \leftarrow \mathcal{D}_{\mathbb{Z}_q,\alpha}$.
3. Choose a random rotation matrix $\mathbf{R} \in \{-1, 1\}^{m\times m}$, and let $\mathbf{R}' := \mathbf{R}\mathbf{X}$. Then define error vector $\boldsymbol{e}_1^T := \boldsymbol{e}_0^T\mathbf{R}'$.
4. Set ciphertext $(\boldsymbol{c}_0, c_1)$ as

$$\boldsymbol{c}_0 := \boldsymbol{s}^T[\mathbf{A}|\mathbf{Y}] + (\boldsymbol{e}_0^T|\boldsymbol{e}_1^T), \qquad c_1 := \boldsymbol{s}^T\boldsymbol{u} + e + \left\lfloor \frac{q}{2} \right\rfloor \mu$$

5. Output ciphertext $\mathsf{ct}_{\boldsymbol{x}} := (\boldsymbol{c}_0, c_1) \in \mathbb{Z}_q^{2m+1}$.

$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$: On input a secret key sk and ciphertext ct, the decryption algorithm does:

1. Parse secret key $\mathsf{sk} = \boldsymbol{r} \in \mathbb{Z}_q^{2m}$ and ciphertext $\mathsf{ct} = (\boldsymbol{c}_0, c_1) \in \mathbb{Z}_q^{2m+1}$.
2. Output $\mu' := \mathsf{Round}(c_1 - (\langle \boldsymbol{c}_0, \boldsymbol{r}\rangle \bmod q)) \in \{0, 1\}$.

5.2 Correctness

We prove correctness of our scheme as follows.

Lemma 5.1. *The identity-based encryption scheme* Π *is correct (cf. Definition 2.1).*

Proof. For any identity $\boldsymbol{x} \in \mathsf{ID}$, recall that the secret key $\mathsf{sk}_{\boldsymbol{x}} = \boldsymbol{r}$ is generated using $\mathsf{SampleLeft}(\mathbf{A}, \mathbf{T}_{\mathbf{A}}, \mathbf{Y}, \boldsymbol{u}, s)$, where $\mathbf{Y} = \mathbf{BX}$, and the input is encoded as

$$\mathbf{X}' := \begin{bmatrix} \mathbf{I}_n \\ x_1 \mathbf{I}_n \\ x_2 \mathbf{I}_n \\ \vdots \\ x_{\ell-1} \mathbf{I}_n \end{bmatrix} \in \mathbb{Z}_q^{\ell n \times n}, \quad \text{and} \quad \mathbf{X} := \mathbf{G}^{-1}_{n\ell,\ell',m}\left(\mathbf{X}' \cdot \mathbf{G}_{n,2,m}\right) \in [\ell']^{m \times m}.$$

Further recall that the ciphertext $\mathsf{ct}_{\boldsymbol{x}} = (\boldsymbol{c}_0, c_1)$ for identity $\boldsymbol{x}$ and message μ is set as

$$\boldsymbol{c}_0 := \boldsymbol{s}^T \left[\mathbf{A} | \mathbf{Y}\right] + (\boldsymbol{e}_0^T | \boldsymbol{e}_1^T), \qquad c_1 := \boldsymbol{s}^T \boldsymbol{u} + e + \left\lfloor \frac{q}{2} \right\rfloor \mu$$

where $\boldsymbol{s} \overset{\$}{\leftarrow} \mathbb{Z}_q^n$, $\boldsymbol{e}_0 \leftarrow \mathcal{D}_{\mathbb{Z}_q^n,\alpha}$, $e \leftarrow \mathcal{D}_{\mathbb{Z}_q,\alpha}$, and $\boldsymbol{e}_1^T := \boldsymbol{e}_0^T \mathbf{R}'$, where $\mathbf{R}' := \mathbf{RX}$ and $\mathbf{R} \leftarrow \{0,1\}^{m \times m}$.

Then the decryption procedure produces

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

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 in Section 5.3.

This completes the proof of correctness. □

5.3 Parameter Setting

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	$n = \lambda$
m	PK-lattice column dimension	$m = n^{1+\delta}$
q	modulus	$q = n^{4.5+4\delta} \log^{3.5+2\delta}(n)$
s	$\mathsf{SampleLeft}$ and $\mathsf{SampleRight}$ width	$s = n^{1.5+1.5\delta} \log^{1.5+\delta}(n)$
α	error width	$\alpha = \sqrt{n} \log^{1+\delta}(n)$
ℓ	identity-space dimension	$\ell = \log\log(n)$
ℓ'	integer-base parameter	$\ell' = \log(n)$

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:
$$|e_1^T\boldsymbol{r}| \leq ||e_0^T|| \cdot ||\mathbf{R}|| \cdot ||\mathbf{X}|| \cdot ||\boldsymbol{r}|| \approx \alpha\sqrt{m} \cdot \sqrt{m} \cdot \ell' m \cdot s\sqrt{m} = m^{2.5} s\alpha\ell' < q/4.$$
- For $\mathsf{SampleLeft}$, we know $||\widetilde{\mathbf{T_A}}|| = O(\sqrt{n\log(q)})$, so require that the sampling width s satisfies
$$s > \sqrt{n\log(q)} \cdot \omega(\sqrt{\log(m)}).$$
- For $\mathsf{SampleRight}$, we know $||\widetilde{\mathbf{T_G}}|| \leq 5$ and that
$$\begin{aligned} s_{\mathbf{R}} &\leq ||\mathbf{R}^*|| \cdot ||\mathbf{X}|| \\ &\leq 12\sqrt{2m} \cdot (\ell' m) \\ &= O(2^\ell m^{1.5}). \end{aligned}$$
Therefore, we need the (joint) sampling width s to, in fact, satisfy the stronger constraint
$$s > 2^\ell m^{1.5}\omega(\sqrt{\log(m)}).$$
- To apply the Leftover Hash Lemma, we need $m \geq (n+1)\log(q) + \omega(\log(n))$.
- To apply Regev's reduction, we need $\alpha > \sqrt{n}\omega(\log(n))$ (α here is an absolute value, not a ratio).

Lattice Approximation Factor. Regev [Reg05] showed that there exists an efficiently samplable B-bounded distribution χ for $B \geq \sqrt{n} \cdot \omega(\log n)$, so that if there is an efficient algorithm that solves the (average-case) $\mathsf{LWE}_{n,m,q,\chi}$ problem, then there is an efficient quantum algorithm that solves $\mathsf{GapSVP}_{\widetilde{O}(n\cdot q/B)}$ and $\mathsf{SIVP}_{\widetilde{O}(n\cdot q/B)}$ on any n-dimensional lattice. The work of Brakerski et al. [BLP+13] shows an analogous-but-classical result for any $\sqrt{n}$-dimensional lattice. For the case of our example parameter setting in Table 2, the resulting lattice problems' approximation factor is $\widetilde{O}\left(n^{5+\epsilon}\right)$, for $\epsilon > 0$.

Further Optimizations. Above, we upper bound the matrix $||\mathbf{X}||$ by $\ell' m$ using a trivial bound, as every entry of $\mathbf{X}$ is at most $\ell' - 1$. However, as $\mathbf{X}$ is a highly sparse matrix in our construction, it is possible to significantly optimize $||\mathbf{X}||$; e.g. about $\ell' \log_2(q)$. Assuming this tighter bound, the parameters s and q can be selected more aggressively, e.g. key width $s \approx \sqrt{m}$ and modulus $q \approx n^{2.5}$. This will result in a somewhat better approximation factor of about $\widetilde{O}(n^{3+\epsilon})$.

Using the tag-based sampling algorithms of the [MP12]-style trapdoor framework in place of [ABB]'s explicit $\mathsf{SampleRight}$ procedure leads to an approximation factor of $\widetilde{O}(n^{1.5})$, matching the (minimum) approximation factor of Dual Regev type PKE, up to polylogarithmic factors in the security parameter n.

6 Security Proof

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

Theorem 6.1. *Assuming the hardness of the standard LWE assumption, our IBE construction is fully secure (cf. Definition 2.2).*

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 $\mathbf{Expt}_{\mathcal{A}}^{\mathsf{IBE}}(1^\lambda)$ as in Definition 2.2. 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 2.2 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 the scheme's setup algorithm. Now this hybrid experiment selects $t = \lceil \log_q(2|Q|/\epsilon) \rceil$, so that we have $q^t \geq 2|Q|/\epsilon \geq q^{t-1}$. Then it choose a random hash function $H^*_{\boldsymbol{h}} \in \mathcal{H}_{\ell-1,t,n,q}$ by selecting ℓ random integer vectors $\boldsymbol{h}_i \in \mathbb{Z}_q^t$ for $i = 0, ..., \ell-1$, and passes it to the challenger. Recall that the hash function maps input $\boldsymbol{x} = (x_1, ..., x_{\ell-1}) \in \mathbb{Z}_q^{\ell-1}$ to:

$$H^*_{\boldsymbol{h}}(\boldsymbol{x}) = M_{t,q}(\boldsymbol{h}_0) \otimes \mathbf{I}_{n/t} + \sum_{i \in [\ell-1]} x_i M_{t,q}(\boldsymbol{h}_i) \otimes \mathbf{I}_{n/t}$$

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 $H^*_{\boldsymbol{h}}$ 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 $Q = \{(1, \boldsymbol{x}_i)\}$ to denote the query set, and by definition challenge identity $(1, \boldsymbol{x}^*) \cap Q = \emptyset$. Recall that we pad 1 to every ID.
- **Guess**: In the guess phase, the adversary outputs his guess $b' \in \{0, 1\}$ for b. The challenger now does the abort check: $H^*_{\boldsymbol{h}}(\boldsymbol{x}^*) = 0$ and $H^*_{\boldsymbol{h}}(\boldsymbol{x}_i) \neq 0$ for all $(1, \boldsymbol{x}_i) \in Q$. 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, the challenger selects ℓ random vectors $\boldsymbol{h}_i \in \mathbb{Z}_q^t$ for $i \in \{0, 1, \ldots, \ell-1\}$, and sets matrix

$$\mathbf{H} = [\mathbf{V}_0 | \mathbf{V}_1 | \mathbf{V}_2 | \cdots | \mathbf{V}_{\ell-1}] \in \mathbb{Z}_q^{n \times n\ell}$$

where each matrix $\mathbf{V}_i = M_{t,q}(\boldsymbol{h}_i) \otimes \mathbf{I}_{n/t} \in \mathbb{Z}_q^{n \times n}$ encodes $\boldsymbol{h}_i \in \mathbb{Z}_q^t$ using the Matrix Embedding function $M_{t,q}(\cdot)$ as described in Section 4. Then, the challenger sets

$$\mathbf{B} = \mathbf{A}\mathbf{R}^* + \mathbf{H}\mathbf{G}_{n\ell,\ell',m}$$

where $\mathbf{R}^* \in \{-1, 1\}^{m \times m}$ is randomly chosen. Additionally, the challenger uses the matrix $\mathbf{R}^*$ 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 $\boldsymbol{e}_0 \leftarrow \mathcal{D}_{\mathbb{Z}_q^m,s}$ and an error integer $e \leftarrow \mathcal{D}_{\mathbb{Z}_q,s}$.
- Set $\mathbf{R}^{*'} := \mathbf{R}^*\mathbf{X}$, and then define error vector $\boldsymbol{e}_1^T := \boldsymbol{e}_0^T \mathbf{R}^{*'}$. Recall that the real scheme uses a random $\mathbf{R} \in \{-1, 1\}^{m \times m}$ to generate the ciphertext.

- Set the challenge ciphertext $(\boldsymbol{c}_0^*, c_1^*)$ as

$$\boldsymbol{c}_0^* := \boldsymbol{s}^T \left[\mathbf{A}|\mathbf{Y}\right] + (\boldsymbol{e}_0|\boldsymbol{e}_1)^T, \qquad c_1^* := \boldsymbol{s}^T \boldsymbol{u} + e + \left\lfloor \frac{q}{2} \right\rfloor \mu$$

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 $\mathbf{T}_{\mathbf{A}}$ using algorithm $\mathsf{TrapGen}(q, n, m)$. To answer a a secret key query for an ID $(1, \boldsymbol{x}_i)$, the secret key is generated using algorithm $\mathsf{SampleLeft}$ associated with the trapdoor $\mathbf{T}_{\mathbf{A}}$.

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 $(1, \boldsymbol{x}^{(i)}) = (1, x_1^{(i)}, ..., x_{\ell-1}^{(i)}) \in \mathbb{Z}_q^\ell$ by first computing whether $H_{\boldsymbol{h}}^*(\boldsymbol{x}_i) = 0$. If it is 0, then the challenger aborts as the previous hybrid. Otherwise, the challenger uses the algorithm $\mathsf{SampleRight}$ with the trapdoor $\mathbf{T}_{\mathbf{G}_{n,2,m}}$ to reply as follows:

$$\boldsymbol{r} \leftarrow \mathsf{SampleRight}\left(\mathbf{A}, \mathbf{B}\cdot\mathbf{X},\ \mathbf{R}^*\mathbf{X},\ \mathbf{T}_{\mathbf{G}_{n,2,m}},\ \boldsymbol{u},\ s\right),$$

where $\mathbf{X}$ is the encoding of the ID $(1, \boldsymbol{x}^{(i)})$. Note that by unfolding the public matrix $\mathbf{B}$ as a predicate encoding, and the identity matrix $\mathbf{X}$ as an input encoding (cf. Definition 3.1), we have:

$$\begin{aligned}
\mathbf{Y} &= \mathbf{B}\cdot\mathbf{X} \\
&= (\mathbf{A}\mathbf{R}^* + \mathbf{H}\mathbf{G}_{n\ell,\ell',m})\cdot\mathbf{G}_{n\ell,\ell',m}^{-1}\left(\mathbf{X}'\cdot\mathbf{G}_{n,2,m}\right) \\
&= \mathbf{A}\mathbf{R}^*\mathbf{X} + \left(\mathbf{H}\cdot\mathbf{X}'\right)\cdot\mathbf{G}_{n,2,m} \\
&= \mathbf{A}\mathbf{R}^*\mathbf{X} + \left(([\mathbf{V}_0|\mathbf{V}_1|\cdots|\mathbf{V}_{\ell-1}]) \cdot \begin{bmatrix} \mathbf{I}_n \\ x_1^{(i)}\mathbf{I}_n \\ \vdots \\ x_{\ell-1}^{(i)}\mathbf{I}_n \end{bmatrix} \right)\cdot\mathbf{G}_{n,2,m} \\
&= \mathbf{A}\mathbf{R}^*\mathbf{X} + \left(\mathbf{V}_0 + \sum_{j=1}^{\ell-1} x_j^{(i)}\mathbf{V}_j\right)\cdot\mathbf{G}_{n,2,m} \\
&= \mathbf{A}\mathbf{R}^*\mathbf{X} + H_{\boldsymbol{h}}^*(\boldsymbol{x}^{(i)})\cdot\mathbf{G}_{n,2,m}
\end{aligned}$$

From the above computation, it is not hard to see that the challenger is able to answer the key query using $\mathsf{SampleRight}$ as long as $H_{\boldsymbol{h}}^*(\boldsymbol{x}^{(i)}) \neq 0$. We remark that (1) $\mathbf{X}$ has a short norm by the change of base argument, so $\mathbf{R}^*\mathbf{X}$ also has a short norm. This is an important property for the $\mathsf{SampleRight}$ algorithm (c.f. Lemma 2.7); and 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 $\mathsf{SampleRight}$ algorithm:

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

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 [ABB], which is described as follows.

Lemma 6.2. *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*

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

The lemma was analyzed by Bellare and Ristenpart [BR09], and further elaborated in the work [ABB]. 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 6.3. *Hybrid H_1 and H_2 are statistically indistinguishable.*

Proof. We show that hybrid H_2 is statistically close to H_1 using Lemma 2.4. Note that the only difference between the two hybrids is how the matrix $\mathbf{B}$ and the error vector $\boldsymbol{e}_1$ in the challenge ciphertext were generated. All the other matrices/vectors are generated identically. In H_1, $(\mathbf{B}, \boldsymbol{e}_1)$ together with the public matrix $\mathbf{A}$ look like $(\mathbf{A}, \mathbf{B}, \boldsymbol{e}_0^T\mathbf{R}^*\mathbf{X})$, yet in H_2, they look like $(\mathbf{A},\ \mathbf{A} \cdot \mathbf{R}^* + \mathbf{H}\mathbf{G}_{n\ell,\ell',m},\ \boldsymbol{e}_0^T\mathbf{R}^*\mathbf{X})$. Since the two hybrids only differ at the matrix/vector, it suffices to show:

$$(\mathbf{A}, \mathbf{B}, \boldsymbol{e}_0^T\mathbf{R}^*\mathbf{X}) \approx (\mathbf{A},\ \mathbf{A} \cdot \mathbf{R}^* + \mathbf{H}\mathbf{G}_{n\ell,\ell',m},\ \boldsymbol{e}_0^T\mathbf{R}^*\mathbf{X}).$$

By Lemma 2.4, we know that the following two distributions are statistically close:

$$(\mathbf{A}, \mathbf{B}, \boldsymbol{e}_0^T\mathbf{R}^*) \approx (\mathbf{A}, \mathbf{A} \cdot \mathbf{R}^*, \boldsymbol{e}_0^T\mathbf{R}^*)$$

where matrix $\mathbf{B}$ is sampled from uniform distribution over $\mathbb{Z}_q^{n\times m}$. Thus, we have

$$(\mathbf{A},\ \mathbf{B} + \mathbf{H}\mathbf{G}_{n\ell,\ell',m},\ \boldsymbol{e}_0^T\mathbf{R}^*\mathbf{X}) \approx (\mathbf{A},\ \mathbf{A} \cdot \mathbf{R}^* + \mathbf{H}\mathbf{G}_{n\ell,\ell',m},\ \boldsymbol{e}_0^T\mathbf{R}^*\mathbf{X}).$$

As $(\mathbf{A}, \mathbf{B} + \mathbf{H}\mathbf{G}_{n\ell,\ell',m}, \boldsymbol{e}_0^T\mathbf{R}^*\mathbf{X})$ is identically distributed as $(\mathbf{A}, \mathbf{B}, \boldsymbol{e}_0^T\mathbf{R}^*\mathbf{X})$, we have

$$(\mathbf{A},\ \mathbf{B},\ \boldsymbol{e}_0^T\mathbf{R}^*\mathbf{X}) \approx (\mathbf{A},\ \mathbf{A} \cdot \mathbf{R}^* + \mathbf{H}\mathbf{G}_{n\ell,\ell',m},\ \boldsymbol{e}_0^T\mathbf{R}^*\mathbf{X}).$$

This completes the proof of the claim. □

Lemma 6.4. *Hybrid H_2 and H_3 are statistically indistinguishable.*

Proof. We show that hybrid H_2 is statistically close to H_3 using Lemmas 2.6 and 2.7. 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 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 SampleRight.

By Lemma 2.6, 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 5.3, SampleLeft and SampleRight are distributed identically by Lemma 2.7. Therefore, hybrid H_2 and H_3 are statistically indistinguishable. □

Lemma 6.5. *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 2.5, 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 matrix $\mathbf{B}$ as in hybrid H_2.
4. Send $\mathsf{mpk} = (\mathbf{A}, \mathbf{B}, \boldsymbol{u})$ 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 identity $(1, \boldsymbol{x}^*) = (1, x_1^*, ..., x_{\ell-1}^*)$, 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

$$\boldsymbol{c}_0^* = [\boldsymbol{v}^{*T} | \boldsymbol{v}^{*T}\mathbf{R}^{*'}], \qquad c_1^* = b_{m+1} + \mu_{b^*}\left\lfloor \frac{q}{2} \right\rfloor$$

   where $\mathbf{R}^{*'} = \mathbf{R}^*\mathbf{X}^*$, and matrix $\mathbf{R}^*$ are chosen randomly from $\{-1,1\}^{m\times m}$ at setup phase.
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 $|Q| \le 0.5\epsilon q^t$. By setting $\delta := 0.5\epsilon$ in Theorem 4.8, we know that

$$\Pr_{\boldsymbol{h}\leftarrow(\mathbb{Z}_q^t)^{\ell+1}}\left[\begin{array}{c} H_{\boldsymbol{h}}^*(\boldsymbol{x}) = \mathbf{0} \bigwedge \\ \forall \boldsymbol{x}' \in Q : H_{\boldsymbol{h}}^*(\boldsymbol{x}') \text{ is full rank} \end{array}\right] \in \left(\frac{1-0.5\epsilon}{q^t}, \frac{1}{q^t}\right).$$

Thus, we know that for any $(Q+1)$-tuple I denoting a challenge id along with ID queries, we have $\epsilon(I) \in \left(\frac{1-0.5\epsilon}{q^t}, \frac{1}{q^t}\right)$. Then by setting $[\epsilon_{\min}, \epsilon_{\max}] = [\frac{1-0.5\epsilon}{q^t}, \frac{1}{q^t}]$ in Lemma 6.2, we have

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

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 $\frac{\epsilon}{4q^t}$ is still non-negligible (even though q^t might look large). Recall that we set $t = \lceil \log_q(2|Q|/\epsilon) \rceil$, so that we have $q^t \geq 2|Q|/\epsilon \geq q^{t-1}$. This implies $\frac{1}{q^t} \geq \frac{\epsilon}{2q|Q|}$, and further we can derive: $\epsilon/4q^t \geq \frac{\epsilon^2}{8q|Q|}$. This is non-negligible as long as ϵ is non-negligible, since q is polynomial for our setting of parameters, and $|Q|$ is polynomially bounded.

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 6.3 and 6.4, we know that the adjacent hybrids are indistinguishable, and thus we have

$$\mathsf{Pr}[\mathsf{E}_1] \approx \mathsf{Pr}[\mathsf{E}_2], \qquad \mathsf{Pr}[\mathsf{E}_2] \approx \mathsf{Pr}[\mathsf{E}_3]. \tag{2}$$

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 6.5, we know that

$$|\mathsf{Pr}[\mathsf{E}_3] - \mathsf{Pr}[\mathsf{E}_4]| = \left|\mathsf{Pr}[\mathsf{E}_3] - \frac{1}{2}\right| \leq \mathbf{Adv}_{\mathcal{B}}^{\mathsf{LWE}}(1^\lambda). \tag{3}$$

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

$$\frac{\epsilon^2}{8q|Q|} \leq \left|\mathsf{Pr}[\mathsf{E}_1] - \frac{1}{2}\right| \approx \left|\mathsf{Pr}[\mathsf{E}_2] - \frac{1}{2}\right| \approx \left|\mathsf{Pr}[\mathsf{E}_3] - \frac{1}{2}\right| \leq \mathbf{Adv}_{\mathcal{B}}^{\mathsf{LWE}}(1^\lambda),$$

which implies $\mathbf{Adv}_{\mathcal{B}}^{\mathsf{LWE}}(1^\lambda) \geq \frac{\epsilon^2}{8q|Q|} - \mathsf{negl}(\lambda)$. This means the reduction $\mathcal{B}$ defined in Lemma 6.5 breaks the LWE assumption with non-negligible probability. This reaches a contradiction, which completes the proof of Theorem 6.1. □

Acknowledgments. We thank Jonathan Katz for insightful comments on an early draft that considerably improved our presentation, and thank Chris Peikert for suggesting the IBE public-key size problem.

This work was performed under financial assistance award 70NANB15H328 from the U.S. Department of Commerce, National Institute of Standards and Technology, a DARPA SafeWare grant, NSF grants CNS-1314857, CNS-1453634, CNS-1518765, CNS-1514261, a Packard Fellowship, a Sloan Fellowship, two Google Faculty Research Awards, and a VMWare Research Award.

References

[ABB] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model (full version). `https://crypto.stanford.edu/~dabo/pubs/papers/latticebb.pdf`. Incorporates material from [ABB10] and [Boy10].

[ABB10] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 553–572. Springer, Heidelberg, May 2010.

[AFV11] Shweta Agrawal, David Mandell Freeman, and Vinod Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 21–40. Springer, Heidelberg, December 2011.

[Alp15] Jacob Alperin-Sheriff. Short signatures with short public keys from homomorphic trapdoor functions. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 236–255. Springer, Heidelberg, March / April 2015.

[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.

[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.

[BHJ$^+$15] Florian Böhl, Dennis Hofheinz, Tibor Jager, Jessica Koch, and Christoph Striecks. Confined guessing: New signatures from standard assumptions. *Journal of Cryptology*, 28(1):176–208, January 2015.

[BKP14] Olivier Blazy, Eike Kiltz, and Jiaxin Pan. (Hierarchical) identity-based encryption from affine message authentication. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 408–425. Springer, Heidelberg, August 2014.

[BLP$^+$13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 575–584. ACM Press, June 2013.

[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.

[CD09] Ronald Cramer and Ivan Damgård. On the amortized complexity of zero-knowledge protocols. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 177–191. Springer, Heidelberg, August 2009.

[CHKP10] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 523–552. Springer, Heidelberg, May 2010.

[DM14] Léo Ducas and Daniele Micciancio. Improved short lattice signatures in the standard model. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 335–352. Springer, Heidelberg, August 2014.

[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.

[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.

[GVW15] Sergey Gorbunov, Vinod Vaikuntanathan, and Daniel Wichs. Leveled fully homomorphic signatures from standard lattices. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 469–477. ACM Press, June 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.

[HK12] Dennis Hofheinz and Eike Kiltz. Programmable hash functions and their applications. *Journal of Cryptology*, 25(3):484–527, July 2012.

[Lew12] Allison B. Lewko. Tools for simulating features of composite order bilinear groups in the prime order setting. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 318–335. Springer, Heidelberg, April 2012.

[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.

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

[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.

[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.

[Wat05] Brent R. Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 114–127. Springer, Heidelberg, May 2005.

[Wee16] Hoeteck Wee. Déjà q: Encore! un petit ibe. Theory of Cryptography: 13th International Conference, *TCC 2016-A*, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part II, 2016.

[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.

A Simple, Compact Digital Signatures from SIS

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 $\mathcal{M} = \{1\} \times \mathbb{Z}_q^{\ell-1}$; that is, a 1-element is appended to the message. Our fully secure signature scheme $\Sigma_{\mathsf{sig}} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ is as follows:

$\mathsf{KeyGen}(1^\lambda)$: On input the security parameter λ, the key generation algorithm does:

1. Set lattice parameters $n = n(\lambda), m = m(\lambda), q = q(\lambda)$ and Gaussian parameter $s = s(\lambda)$ (according to the same constraints as our IBE Π; cf. Section 5.3).
2. Generate a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ associated with its trapdoor $\mathbf{T_A}$ using algorithm $\mathsf{TrapGen}(q, n, m)$.
3. Sample a uniformly random matrix $\mathbf{B} \in \mathbb{Z}_q^{n\times m}$.
4. Output verification key vk and signing key sk as:

$$\mathsf{vk} = (\mathbf{A}, \mathbf{B}), \qquad \mathsf{sk} = \mathbf{T_A}$$

$\mathsf{Sign}(\mathsf{sk}, \boldsymbol{\mu})$: On input a signing key sk and a message $\boldsymbol{\mu} = (1, \mu_1, ..., \mu_{\ell-1}) \in \{1\} \times \mathbb{Z}_q^{\ell-1}$, the signing algorithm does:

1. Define the input-encoding matrices

$$\mathbf{X}' := \begin{bmatrix} \mathbf{I}_n \\ \mu_1\mathbf{I}_n \\ \mu_2\mathbf{I}_n \\ \vdots \\ \mu_{\ell-1}\mathbf{I}_n \end{bmatrix} \in \mathbb{Z}_q^{\ell n\times n}, \quad \text{and} \quad \mathbf{X} := \mathbf{G}^{-1}_{n\ell,\ell',m}\left(\mathbf{X}' \cdot \mathbf{G}_{n,2,m}\right) \in [\ell']^{m\times m}.$$

2. Set the message matrix for $\boldsymbol{\mu}$ to be

$$\mathbf{Y} := \mathbf{B} \cdot \mathbf{X} \in \mathbb{Z}_q^{n\times m}.$$

3. Compute short vector $\boldsymbol{r} \in \mathbb{Z}_q^{2m}$, using

$$\boldsymbol{r} \leftarrow \mathsf{SampleLeft}(\mathbf{A}, \mathbf{T_A}, \mathbf{Y}, 0, s)$$

such that $[\mathbf{A}|\mathbf{Y}] \cdot \boldsymbol{r} = 0 \bmod q$.

4. Output the signature $\sigma_{\boldsymbol{\mu}} = \boldsymbol{r}$.

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

1. First check the message $\boldsymbol{\mu}$ is well formed in $\{1\} \times \mathbb{Z}_q^{\ell-1}$.
2. Then check the signature $\sigma_{\boldsymbol{\mu}} = \boldsymbol{r}$ is a small but non-zero vector, i.e. $0 \neq |\boldsymbol{r}| \leq \sqrt{2m}s$.
3. Check whether the following equation holds:

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

where $\mathbf{Y} = \mathbf{B} \cdot \mathbf{X}$, and matrix $\mathbf{X}$ is computed as in the signing algorithm.

4. If all the verification steps pass, then accept the signature; otherwise, reject.

We say a signature scheme is unforgeable if for any PPT adversary, he cannot forge a signature of a new message, which passes the verification algorithm, even if he can obtain any polynomial number of message/signature pairs of his choice. The correctness and security proofs (from Short Integer Solution) are the same as in [Boy10], with the modified hash function computation and simulation of matrix $\mathbf{B}$ as shown in the IBE Π's security proof (cf. Section 6). In what follows, we sketch the proof of unforgeability of the signature scheme Σ_{sig} as follows:

Theorem A.1. *Assuming the hardness of the standard SIS assumption, our signature scheme Σ_{sig} is unforgeable.*

Proof (sketch): Suppose there exists an adversary $\mathcal{A}$ that can break the unforgeability of the signature scheme Σ_{sig}, 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 $\mathbf{A}\boldsymbol{r} = 0 \bmod q$, such that $0 \neq ||\boldsymbol{r}|| \leq \beta$.

Setup. Reduction $\mathcal{B}$ sets matrix $\mathbf{B}$ in vk as follows: Select ℓ random vectors $\boldsymbol{h}_i \in \mathbb{Z}_q^t$ for $i \in \{0, 1, \ldots, \ell-1\}$ of hash function $H^*_{\boldsymbol{h}} \in \mathcal{H}_{\ell-1,t,n,q}$, and set matrix

$$\mathbf{H} = [\mathbf{V}_0|\mathbf{V}_1|\mathbf{V}_2|\cdots|\mathbf{V}_{\ell-1}] \in \mathbb{Z}_q^{n\times n\ell}$$

where each matrix $\mathbf{V}_i = M_{t,q}(\boldsymbol{h}_i) \otimes \mathbf{I}_{n/t} \in \mathbb{Z}_q^{n\times n}$ encodes $\boldsymbol{h}_i \in \mathbb{Z}_q^t$ using the Matrix Embedding function $M_{t,q}(\cdot)$ as described in Section 4. Then, $\mathcal{B}$ sets

$$\mathbf{B} = \mathbf{A}\mathbf{R}^* + \mathbf{H}\mathbf{G}_{n\ell,\ell',m}$$

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

Queries. Reduction $\mathcal{B}$ answers signature queries from $\mathcal{A}$ on message $\boldsymbol{\mu}^{(i)} = (\mu_1^{(i)}, ..., \mu_{\ell-1}^{(i)})$ as follows:

1. Abort the simulation if

$$H^*_{\boldsymbol{h}}(\boldsymbol{\mu}^{(i)}) = M_{t,q}(\boldsymbol{h}_0) \otimes \mathbf{I}_{n/t} + \sum_{i\in[\ell-1]} \mu_i^{(i)} M_{t,q}(\boldsymbol{h}_i) \otimes \mathbf{I}_{n/t} = 0$$

2. Otherwise, compute

$$\mathbf{B} \cdot \mathbf{X} = \mathbf{A}\mathbf{R}^*\mathbf{X} + H^*_{\boldsymbol{h}}(\boldsymbol{\mu}^{(i)}) \cdot \mathbf{G}_{n,2,m}$$

We omit the computation detail here, since the process is similar to computation in Hybrid H_3 in the IBE security proof. Then sample a short vector $\boldsymbol{r}_i$, using

$$\boldsymbol{r}_i \leftarrow \mathsf{SampleRight}\left(\mathbf{A}, \mathbf{B}\cdot\mathbf{X},\ \mathbf{R}^*\mathbf{X},\ \mathbf{T}_{\mathbf{G}_{n,2,m}},\ 0,\ s\right),$$

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 $H^*_{\boldsymbol{h}}(\boldsymbol{\mu}^*) \neq 0$.
2. Otherwise, we have

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

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 Theorem 4.8, that if an adversary successfully forges a signature with probability ϵ, by setting $\delta := \epsilon$ in Theorem 4.8, then reduction $\mathcal{B}$ solves SIS instance with probability

$$\epsilon' \geq \pi \frac{(1-\epsilon)}{q^t} \geq \pi \frac{(1-\epsilon)\epsilon}{2|Q|} \geq \frac{(1-\epsilon)\epsilon}{3|Q|}$$

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 $t = \lceil \log_q(2|Q|/\epsilon) \rceil$, so that we have $q^t \geq 2|Q|/\epsilon \geq q^{t-1}$, which implies $\frac{1}{q^t} \geq \frac{\epsilon}{2q|Q|}$.

This concludes the proof sketch. □