Efficient Batched IBE from Lattices in the Standard Model

Saisi Xiong[1] Yijian Zhang[2] and Jie Chen[3]

[1] East China Normal University, Shanghai, China.
saisixiong@163.com
[2] University of Wollongong, Wollongong NSW, Australia.
yz016@uowmail.edu.au
[3] Wuhan University, Wuhan, China.
s080001@e.ntu.edu.sg

Abstract. In this work, we present the first lattice-based construction of batched IBE in the standard model, whose security is proven under the succinct LWE assumption. Prior batched IBE schemes are only known either based on pairing-based assumptions or in the random oracle model. Moreover, our scheme is shown to be highly efficient, as the master public key, decryption key, and ciphertext are independent of the batch size B. Technically, we mainly rely on an insightful observation: batched IBE can be obtained solely from Inner-Product Encryption (IPE). To satisfy the efficiency requirements of batched IBE, we require an IPE scheme that owns two important features: *decomposable* key generation and *compact* components. Finally, we show how to construct such an IPE scheme from the well-known BGG+14 IPE scheme via careful modification.

1 Introduction

Identity-Based Encryption (IBE), which was introduced by [17], enables the encryption of messages using a master public key and an identity, thus removing the need for *Public Key Infrastructure* (PKI). In an IBE scheme, any user with a decryption key associated with the matched identity can decrypt and recover the message.

Batched IBE. In blockchain application scenarios, when encrypting a block, the encryptor will use the block's identity or publication time as the identity. Traditional IBE schemes require generating a large number of keys proportional to the number of identities. To address this problem, Agarwal et al. [1] proposed the batched IBE primitive. Batched IBE enables the batch decryption of multiple ciphertexts using a single secret key. Specifically, in a batched IBE scheme, a single secret key is generated regarding a short digest that binds a set of B block identities along with a label. This key can be used to decrypt data encrypted under any identity within the bound set and a matched label, as opposed to generating many keys for all these identities, thereby eliminating the linear dependency on B.

Initially, Agarwal et al. [1] constructed the first batched IBE scheme by utilizing a KZG polynomial commitment scheme along with an efficient special-purpose witness encryption scheme. However, the security of their scheme is only proven in the generic group model [18], which is pretty weak. Very recently, Gong et al. [11] presented a new batched IBE scheme without this drawback, which is based on the q-type assumption over pairings in the standard model. However, since all of these schemes are vulnerable to future quantum attacks, it is always of interest to construct batched IBE schemes based on assumptions beyond pairings, such as lattices. Compared with pairing-based cryptographic schemes, lattice-based schemes have gained broader popularity owing to the well-established post-quantum security guarantee. Therefore, Boneh et al. recently [6] proposed a lattice-based batched decryption scheme, which is based on the Learning with Error (LWE) assumption. Then, a batched IBE scheme can be obtained from this batched decryption scheme via a slight modification. However, a major drawback of this scheme is that it relies on random oracles. Therefore, in a spirit similar to that of Gong et al. [11], we aim to provide a new batched IBE scheme from lattices in the standard model.

1.1 Our Results

In this work, we propose the first lattice-based batched IBE scheme in the standard model, which is selectively secure based on the ℓ-succinct LWE assumption [20]. The ℓ-succinct LWE assumption is a falsifiable assumption and is implied by the (public-coin) evasive LWE assumption and the standard LWE assumption. This assumption, as well as its various variants, has been used in recent constructions of succinct functional commitments [22,23,10], distributed broadcast encryption [8,24], and registered attribute-based encryption [7]. In this work, we assume $\ell = \mathsf{poly}(\lambda, \log B)$, where B denotes the size of the batch set.

We summarize and compare prior batched IBE schemes in Table 1, and claim several advantages of our scheme as follows:

- **Security in the standard model.** Our scheme does not rely on any non-standard models such as the generic group model and the random oracle model. Prior to this work, only the scheme very recently proposed by Gong et al. [11] is known to be secure in the standard model, while it has to rely on a q-type assumption over pairings.
- **Compactness.** For any batched IBE schemes, it is desired to obtain a smaller master public key mpk, secret key sk, and ciphertext ct. Our result exactly achieves $|\mathsf{mpk}| = \mathsf{poly}(\log B)$, $|\mathsf{sk}| = \mathsf{poly}(\log B)$ and $|\mathsf{ct}| = \mathsf{poly}(\log B)$. Prior pairing-based schemes [1,11] must endure a long mpk scaling with B. On the other hand, although the scheme in [6] also owns these components that are comparable to our result, it relies on random oracles.

Scheme	$\|\mathsf{mpk}\|$	$\|\mathsf{sk}\|$	$\|\mathsf{ct}\|$	Assumption
AFP25 [1]	$O(B)$	$O(1)$	$O(1)$	GGM+ROM
GWWW25 [11]	$O(B)$	$O(1)$	$O(1)$	GGM+ROM
GWWW25 [11]	$O(B)$	$O(1)$	$O(1)$	q-type
BLT25 [6]	$O(\log^3 B)$	$O(\log B)$	$O(\log^3 B)$	LWE + ROM
This work	$O(\log^6 B)$	$O(\log^2 B)$	$O(\log^2 B)$	ℓ-succinct LWE

Table 1. Comparison with prior batched IBE schemes. For each scheme, we report the size of the master public key mpk, decryption key sk, ciphertext ct, and the underlying assmuption. Here B stands for the batch size, "GGM" stands for the generic bilinear group model, and "ROM" stands for the random oracle model. We hide λ-related factors in the parameter size and unify the identity space into $\mathbb{Z}_q$, and it always has $q = O(B)$ and $\ell = \mathsf{poly}(\lambda, \log B)$.

1.2 Technical Overview

We first consider the syntax of a batched IBE scheme. Let N be the number of users in the system. Each user is indexed by an unique identity $\mathsf{id} \in [N]$, where $[N]$ denotes the set $\{1, \cdots, N\}$. Then, a simplified batched IBE runs as follows:

- **Setup.** The setup algorithm takes the security parameter 1^λ and the maximum batch size 1^B, then outputs the master public key mpk and the master secret key msk.
- **Digest generation.** The digest-generation algorithm takes the master public key mpk, a batch set of identities $S = \{\mathsf{id}_1, \cdots, \mathsf{id}_B\}$, then outputs a digest dig.
- **Key generation.** The key-generation algorithm takes the master secret key msk, a digest dig, and a batch label t, then outputs a secret key sk.
- **Encryption.** The encryption algorithm takes the master public key mpk, a message m, an identity id, and a batch label t', then outputs a ciphertext ct.
- **Decryption.** The decryption algorithm takes the master public key mpk, a ciphertext ct, a secret key sk, a digest dig, a batch set of identities S, and an identity id, then outputs a message m or an empty symbol $\perp$.

For correctness, it says that the decryption algorithm will recover the message m if $\mathsf{id} \in S$ and $t = t'$ hold simultaneously. The security requires that ct reveals nothing about m when either $\mathsf{id} \notin S$ or $t \neq t'$. For efficiency, it critically requires that the runtime of the key generation algorithm is polylogarithmic in the batch size B. This implies that the digest dig and the secret key sk are also polylogarithmic in B.

In the following, we focus on constructing a simplified version of batched IBE, in which the labels are ignored and the decryption succeeds as long as $\mathsf{id} \in S$. Since the difference is minor, we still call it batched IBE for simplicity of expression.

Observation: a naive relation between batched IBE and IPE. We start with an interesting observation that can connect a batched IBE scheme to a well-studied primitive called *Inner-Product Encryption* [14]. An IPE scheme necessitates four algorithms of setup, key generation, encryption, and decryption. In particular, the setup algorithm takes 1^λ along with the vector length 1^ℓ, then outputs the master public key $\mathsf{mpk}_{\mathsf{IPE}}$ and the master secret key $\mathsf{msk}_{\mathsf{IPE}}$. The key-generation algorithm just takes $\mathsf{msk}_{\mathsf{IPE}}$ and a vector $\mathbf{y} \in \mathbb{Z}_q^{1\times\ell}$, then outputs a secret key $\mathsf{sk}_{\mathbf{y}}$. The encryption algorithm takes $\mathsf{mpk}_{\mathsf{IPE}}$, a message m, and a vector $\mathbf{x} \in \mathbb{Z}_q^{1\times\ell}$, then outputs a ciphertext $\mathsf{ct}_{\mathbf{x}}$. The decryption algorithm can employ $\mathsf{sk}_{\mathbf{y}}$ together with $(\mathbf{x}, \mathbf{y})$ to recover m from $\mathsf{ct}_{\mathbf{x}}$, as long as $\mathbf{x}\mathbf{y}^\top = 0$; otherwise, it outputs $\perp$ when $\mathbf{x}\mathbf{y}^\top \neq 0$.

A naive relation between batched IBE and IPE is that the former is actually a special case of the latter in terms of functionality. For a batch set $S = \{\mathsf{id}_1, \cdots, \mathsf{id}_B\}$ and a ciphertext identity id, we can define a special polynomial $f(x) = \prod_{i=1}^{B}(x - \mathsf{id}_i)$ to express the function of batched IBE:

$$\mathsf{id} \in S \iff f(\mathsf{id}) = 0 \text{ and } \mathsf{id} \notin S \iff f(\mathsf{id}) \neq 0 \tag{1}$$

A general but inefficient compiler. According to (1), we can therefore generically derive a batched IBE scheme (without satisfying the efficiency requirements) from any IPE. To this end, we can rewrite the polynomial f as the form $f(x) := \sum_{j=0}^{B} y_j x^j$, where y_j denotes the coefficient of f. The general compiler from IPE to batched IBE works as follows:

– **Setup.** It generates the master public key $\mathsf{mpk} = \mathsf{mpk}_{\mathsf{IPE}}$ and the master secret key $\mathsf{msk} = \mathsf{msk}_{\mathsf{IPE}}$, where it sets the vector length $\ell = B + 1$.
– **Digest generation.** It directly outputs $\mathsf{dig} = S$.
– **Key generation.** It generates $\mathsf{sk} = \mathsf{sk}_{\mathbf{y}}$, which is associated with a vector $\mathbf{y} = (y_0, y_1, \cdots, y_B)$.
– **Encryption.** It generates $\mathsf{ct} = \mathsf{ct}_{\mathbf{x}}$, which is associated with a vector $\mathbf{x} = (1, \mathsf{id}, \cdots, \mathsf{id}^B)$.
– **Decryption.** It just performs the IPE decryption algorithm. Note that if $f(\mathsf{id}) = \sum_{j=0}^{B} y_j \mathsf{id}^j = \mathbf{x}\mathbf{y}^\top = 0$, the message m can be recovered properly; otherwise, it reveals nothing. This is compliant with the relation (1).

Though supporting the function of batched IBE, the above scheme does not satisfy the efficiency requirements. Indeed, constructing inefficient batched IBE schemes can be trivially achieved by relying on traditional IBE, while we stick to using IPE and will next show how to get rid of such inefficiency.

Upgrading the efficiency via decomposable IPE. As noted in the first work of batched IBE [1], the essential challenge of constructing batched IBE schemes is to support an efficient key generation, which should be independent of the batch size B. To address this issue, our idea is to find (lattice-based) IPE schemes with good efficiency features as well. The most desirable feature is *succinct* key generation, i.e., the runtime of key generation in an IPE scheme is independent of the vector; however, such a scheme does not exist, as the key-generation algorithm must access the entire vector $\mathbf{y}$. To get around this limitation, our solution is to find an IPE scheme with the feature of *decomposable* key generation. In more detail, it means that the key-generation algorithm can be decomposed into two sub-algorithms that deal with the vector part and the key part, respectively.

(1) **Vector-specific sub-algorithm.** It takes the master public key $\mathsf{mpk}_{\mathsf{IPE}}$ and a vector $\mathbf{y}$, then outputs an internal state st.
(2) **Key-specific sub-algorithm.** It takes the master secret key $\mathsf{msk}_{\mathsf{IPE}}$ and st, then outputs $\mathsf{sk}_{\mathbf{y}}$ that is indistinguishable from the secret key normally generated from the key-generation algorithm in IPE.

Along this way, we can treat the vector-specific sub-algorithm as the digest-generation algorithm and the key-specific sub-algorithm as the key-generation algorithm, thus yielding a batch IBE scheme as desired! Of course, the internal state st and $\mathsf{sk}_{\mathbf{y}}$ should be of size $\mathsf{poly}(\log B)$ to guarantee the efficiency of our final scheme. On the other hand, despite not being clearly stated, the ciphertext of a batched IBE scheme should also be independent of the batch size B to facilitate real-world applications. Putting all these together, constructing an efficient batched IBE scheme requires at least a decomposable IPE that supports both compact secret keys and compact ciphertexts, i.e., independent of the vector length ℓ. This is quite a challenge in the study of IPE [3,25,5,12], as most existing schemes just support one of the compact components. To our knowledge, known compact IPE schemes either place restrictions on inner-product vectors [19] or rely on pairings [4,9,15]. On the other hand, many expressive *Attribute-Based Encryption* (ABE) schemes may already imply compact IPE, but they will not be considered in this work due to their extremely impractical efficiency.

The scheme adapted from BGG+14 IPE. Here, we show how to construct a concrete decomposable IPE scheme with compact parameters from lattices. Our starting point is the IPE scheme proposed by Boneh, Gentry, Gorbunov, Halevi, Nikolaenkok, Segev, Vaikuntanathan, and Vinayagamurthy in EUROCRYPT 2014 [5], which we refer to as BGG+14 IPE for short. The readers familiar with BGG+14 may see that it actually proposed a more powerful ABE scheme for general circuits, while the efficiency of this ABE scheme is still impractical. The IPE scheme we require is exactly a special case of the ABE scheme, so it is concerning that the impracticability reflects on the IPE scheme as well. However, we observe that the impracticality of ABE is mainly caused by the homomorphic multiplication between ciphertexts, which is unnecessary for the function of IPE. We therefore want to adapt the BGG+14 IPE scheme to a decomposable IPE scheme with compact components.

Given lattice parameters n, m, q, let $\mathbf{A} \otimes \mathbf{B}$ denote the the Kronecker product between matrices $\mathbf{A}$ and $\mathbf{B}$, let $\mathbf{G}$ denote the gadget matrix [16] and $\mathbf{G}^{-1}$ denote the binary decomposition operator, i.e., $\mathbf{G} \cdot \mathbf{G}^{-1}(\mathbf{A}) = \mathbf{A}$ for all $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$. Then we present an overview of BGG+14 IPE over lattice:

$$\begin{aligned}
\mathsf{mpk}_{\mathsf{IPE}} &: \mathbf{A}_0 \in \mathbb{Z}_q^{n\times m},\ \mathbf{A} = (\mathbf{A}_1, \cdots, \mathbf{A}_\ell) \in \mathbb{Z}_q^{n\times \ell m},\ \mathbf{p} \in \mathbb{Z}_q^n \\
\mathsf{ct}_{\mathbf{x}} &: \mathbf{s}\mathbf{A}_0 + \mathsf{noise} \in \mathbb{Z}_q^{1\times m},\ \mathbf{s}\mathbf{p} + \mathsf{m}\cdot\lfloor q/2\rfloor + \mathsf{noise} \in \mathbb{Z}_q, \\
&\quad \mathbf{s}(\mathbf{A} - \mathbf{x}\otimes\mathbf{G}) + \mathsf{noise} \in \mathbb{Z}_q^{1\times \ell m} \\
\mathsf{sk}_{\mathbf{y}} &: \mathbf{k} \in \mathbb{Z}^{2m} \text{ s.t. } [\mathbf{A}_0|\mathbf{A}(\mathbf{y}^\top \otimes \mathbf{I}_m)]\cdot\mathbf{k} = \mathbf{p}
\end{aligned}$$

where $\mathbf{s} \leftarrow \mathbb{Z}_q^{1\times n}$ is the ciphertext randomness, and the master secret key $\mathsf{msk}_{\mathsf{IPE}}$ is the trapdoor for matrix $\mathbf{A}_0$. With $\mathsf{msk}_{\mathsf{IPE}}$, it is easy to derive a low-norm vector $\mathbf{k}$ for any given matrix $\mathbf{B}$ such that $[\mathbf{A}_0|\mathbf{B}]\cdot\mathbf{k} = \mathbf{p}$. To decrypt the ciphertext $\mathsf{ct}_{\mathbf{x}}$, we only rely on the mechanism of homomorphic addition and scalar multiplication proposed by Boneh et al.. Concretely, it has

$$\begin{aligned}
\Big(\mathbf{s}(\mathbf{A} - \mathbf{x}\otimes\mathbf{G}) + \mathsf{noise}\Big)(\mathbf{y}^\top \otimes \mathbf{I}_m) &= \mathbf{s}\mathbf{A}(\mathbf{y}^\top \otimes \mathbf{I}_m) - \mathbf{s}(\mathbf{x}\otimes\mathbf{G})(\mathbf{y}^\top \otimes \mathbf{I}_m) + \mathsf{noise} \\
&= \mathbf{s}\mathbf{A}(\mathbf{y}^\top \otimes \mathbf{I}_m) - \mathbf{x}\mathbf{y}^\top \cdot \mathbf{s}\mathbf{G} + \mathsf{noise} \\
&\approx \mathbf{s}\mathbf{A}(\mathbf{y}^\top \otimes \mathbf{I}_m)
\end{aligned}$$

where the second $=$ follows the tensor-product property $(\mathbf{A}\otimes\mathbf{B})(\mathbf{C}\otimes\mathbf{D}) = (\mathbf{A}\mathbf{C})\otimes(\mathbf{B}\mathbf{D})$. Notably, $\mathbf{s}\mathbf{A}(\mathbf{y}^\top \otimes \mathbf{I}_m)$ can be estimated if $\mathbf{x}\mathbf{y}^\top = 0$. With $\mathsf{sk}_{\mathbf{y}}$, it can therefore recover

$$\begin{aligned}
\mathbf{s}\mathbf{p} + \mathsf{m}\cdot\lfloor q/2\rfloor + \mathsf{noise} - [\mathbf{s}\mathbf{A}_0|\mathbf{s}\mathbf{A}(\mathbf{y}^\top \otimes \mathbf{I}_m)]\cdot\mathbf{k} &= \mathbf{s}\mathbf{p} + \mathsf{m}\cdot\lfloor q/2\rfloor + \mathsf{noise} - \mathbf{s}\mathbf{p} \\
&= \mathsf{m}\cdot\lfloor q/2\rfloor + \mathsf{noise} \\
&\approx \mathsf{m}\cdot\lfloor q/2\rfloor
\end{aligned}$$

We show how to build a decomposable IPE with compact components from the above IPE scheme, in which we mainly analyze the efficiency of key generation and encryption:

- The key generation of the above IPE scheme is inherently decomposable. In the vector-specific sub-algorithm, it just outputs a vector commitment to $\mathbf{y}$, i.e., $\mathsf{st} = \mathbf{A}(\mathbf{y}^\top \otimes \mathbf{I}_m) \in \mathbb{Z}_q^{n\times m}$. In the key-specific sub-algorithm, it can generate the secret key $\mathsf{sk}_{\mathbf{y}} = \mathbf{k} \in \mathbb{Z}^{2m}$ just by taking msk and st. Since n, m and q are fixed system parameters, st and sk seem to be independent of the vector length ℓ.
- The ciphertext of the above IPE scheme is not compact, while we can compress it from $O(\ell)$ to $O(\log \ell)$ by employing the recent matrix-commitment scheme of Wee [21]. Specifically, such long ciphertext is caused by the component $\mathbf{s}(\mathbf{A} - \mathbf{x}\otimes\mathbf{G})$. To address this issue, the matrix-commitment scheme of Wee shows that there is an efficient and deterministic algorithm to compute a commitment $\mathbf{C}_{\mathbf{x}} \in \mathbb{Z}_q^{n\times m}$ and a low-norm opening $\mathbf{Z} \in \mathbb{Z}_q^{\ell m}$ such that

$$\mathbf{C}_{\mathbf{x}} \cdot \mathbf{V}_\ell = \mathbf{x}\otimes\mathbf{G} - \mathbf{B}\cdot\mathbf{Z} \in \mathbb{Z}_q^{n\times \ell m} \tag{2}$$

where $\mathbf{B}$ comes from the public parameters $\mathsf{pp}_{\mathsf{com}}$ (independent of the vector length ℓ) of the matrix-commitment scheme and $\mathbf{V}_\ell \in \mathbb{Z}_q^{m\times \ell m}$ is a fixed low-norm verification matrix that is publicly derived from $\mathsf{pp}_{\mathsf{com}}$ and the width ℓm. Thus, in the adapted IPE scheme, it encrypts using a succinct commitment $\mathbf{C}_{\mathbf{x}}$, instead of $\mathbf{x}\otimes\mathbf{G}$. This yields a short ciphertext independent of the vector length ℓ. On the other hand, since the implementation of the matrix-commitment scheme requires the ℓ-succinct LWE assumption [20], our final scheme will also rely on this assumption.

Above all, we present our new decomposable IPE scheme as follows:

$$\begin{aligned} \mathsf{mpk}_{\mathsf{IPE}} &: \mathsf{pp}_{\mathsf{com}},\ \mathbf{B}_1 \in \mathbb{Z}_q^{n\times m},\ \mathbf{p} \in \mathbb{Z}_q^n \\ \mathsf{ct}_{\mathbf{x}} &: \mathbf{sB} + \mathsf{noise} \in \mathbb{Z}_q^{1\times m},\ \mathbf{sp} + \mathsf{m}\cdot \lfloor q/2 \rceil + \mathsf{noise} \in \mathbb{Z}_q, \\ &\quad \mathbf{s}(\mathbf{B}_1 + \mathbf{C}_{\mathbf{x}}) + \mathsf{noise} \in \mathbb{Z}_q^{1\times m} \\ \mathsf{sk}_{\mathbf{y}} &: \mathbf{k} \in \mathbb{Z}^{2m} \text{ s.t. } [\mathbf{B} | -\mathbf{B}\mathbf{V}_\ell(\mathbf{y}^\top \otimes \mathbf{I}_m)] \cdot \mathbf{k} = \mathbf{p} \end{aligned}$$

where $\mathbf{B} \in \mathbb{Z}_q^{n\times m}$ in $\mathsf{pp}_{\mathsf{com}}$ serves as $\mathbf{A}_0$ and $-\mathbf{B}\mathbf{V}_\ell$ serves as $\mathbf{A}$ in the BGG+14 IPE scheme. The decryption can first recover $\mathbf{x}\otimes\mathbf{G}$ in the component $\mathbf{s}(\mathbf{B}_1 + \mathbf{C}_{\mathbf{x}}) + \mathsf{noise}$ via the matrix commitment scheme (2), then proceed to obtain m as before.

Compiling into batched IBE. The final step is to compile our decomposable IPE scheme into a concrete batched IBE scheme. The compilation is natural as claimed before, and recall that we assume the identity space $[N]$, so the underlying decomposable IPE scheme should take the vectors $\mathbf{y} \in [N^B]^{B+1}$ and $\mathbf{x} \in [N^B]^{B+1}$. However, a new problem is that the lattice modulus q must be greater than N^B and the ciphertext size is therefore linear in B. This is because the noise magnitude is enlarged by a factor of $(B+1)N^B$ when multiplying $\mathbf{y}^\top \otimes \mathbf{I}_m$ with ciphertext components when decrypting. Our solution is to replace $\mathbf{y}$ with a low-norm vector $\hat{\mathbf{y}}$, while not affecting the function of our batched IBE scheme. This can be done by further breaking $\mathbf{y} \in [N^B]^{B+1}$ into a binary decomposition form. In more detail, instead of $(\mathbf{y}, \mathbf{x})$, the decomposable IPE scheme takes new vectors $\hat{\mathbf{y}}^\top = \mathbf{G}^{-1}(\mathbf{y}^\top) \in \{0,1\}^{B(B+1)\lceil \log N\rceil}$ and $\hat{\mathbf{x}} = \mathbf{x}\otimes\mathbf{g}'$ where $\mathbf{g}' = (1, 2^1, \cdots, 2^{B\lceil \log N\rceil - 1})$. It can be seen that

$$\hat{\mathbf{x}}\hat{\mathbf{y}}^\top = (\mathbf{x}\otimes\mathbf{g}')\cdot\mathbf{G}^{-1}(\mathbf{y}^\top) = \mathbf{x}\mathbf{y}^\top = f(\mathsf{id})$$

where the polynomial f is as defined in (1). As a result, the noise growth factor will be at most $\mathsf{poly}(B)$, thus leading to a small modulus q that is bounded by $\mathsf{poly}(B)$. Therefore, we can obtain an efficient batched IBE scheme from lattices, where the master public key, secret key, and ciphertext are rather compact in size $\mathsf{poly}(\log B)$. By the way, a disadvantage of our scheme is that the encryption runtime scales with B, as it must access the entire vector $\mathbf{x}$ to generate a matrix commitment to $\mathbf{x}\otimes\mathbf{G}$. This issue can be easily resolved in practice by considering an offline/online encryption paradigm, where the (deterministic) computation of the matrix commitment can be executed in the offline phase.

Note that the above batched IBE scheme follows a simplified syntax. The original definition of batched IBE requires the secret key associated with an extra label t and the ciphertext associated with a label t'. The decryption fails if $t \neq t'$. We remark that this actually models the function of normal IBE schemes, so we decide to incorporate our simplified batched IBE with the lattice-based IBE scheme in [2]. This is feasible as the underlying BGG+14 IPE scheme and the IBE scheme share the same structure. Finally, we derive the first efficient batched IBE scheme from lattices, and more details are presented in Section 3.

2 Preliminaries

Notations. For a finite set S, we write $s \leftarrow S$ to denote that s is picked uniformly from S. Then, we use $|S|$ to denote the size of S. Let $\approx_s$ denote two distributions being statistically indistinguishable, $\approx_c$ denote two distributions being computationally indistinguishable, $\Delta(A, B)$ denote the statistical distance between the two distributions A and B. For any $x \in \{0,1\}^n$, we use $x[w]$ to denote the w-th bit of x. For a matrix $\mathbf{A}$, we use $\widetilde{\mathbf{A}}$ to denote the Gram-Schmidt orthogonalization of $\mathbf{A}$, $\|\mathbf{A}\|_2$ to denote the l_2-norm of $\mathbf{A}$, $\|\mathbf{A}\|$ to denote the l_∞-norm of $\mathbf{A}$. We use $\mathbb{Z}_p^*$ to denote $\mathbb{Z}_p \backslash \{0\}$ and use $\mathbf{G} = \mathbf{I}_n \otimes \mathbf{g} \in \mathbb{Z}^{n \times m}$ to denote gadget matrix where $\mathbf{g} = (1, 2^1, \cdots, 2^{\lceil \log q \rceil - 1}, 0, \cdots, 0) \in \mathbb{Z}^{1 \times m}$
Tensor Product (Kronecker Product). The tensor product is a binary operation acting on two vector spaces that maps them into a new vector space while preserving bilinearity. The tensor product for matrices $\mathbf{A} = (a_{i,j}) \in \mathbb{Z}^{n \times m}$, $\mathbf{B} \in \mathbb{Z}^{\ell \times \lambda}$ is defined as:

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{1,1}\mathbf{B} & \cdots & a_{1,m}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{n,1}\mathbf{B} & \cdots & a_{n,m}\mathbf{B} \end{bmatrix} \in \mathbb{Z}^{n\ell \times m\lambda}$$

We have the following properties:

$$(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) = (\mathbf{AC}) \otimes (\mathbf{BD})$$

2.1 Lattice Background

Lemma 1 (Lattice Trapdoor [5]). *Let λ be a security parameter and let q, ℓ, m be lattice parameters. There are efficient algorithms with the properties below:*

$\mathsf{TrapGen}(1^n, 1^m, q) \rightarrow (\mathbf{A}, \mathbf{T_A})$**.** *A randomized algorithm that receives parameters $1^n, 1^m, q$ where $m = \Theta(n \log q)$. It then outputs a full-rank matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and a basis $\mathbf{T_A} \in \mathbb{Z}^{m \times m}$ for $\Lambda_q^{\perp}(\mathbf{A})$ such that $\mathbf{A}$ is $\mathsf{negl}(n)$-close to uniform and $\|\widetilde{\mathbf{T}}_\mathbf{A}\| \leq O(\sqrt{n \log q})$, with all but negligible probability in n.*

$\mathsf{SamplePre}(\mathbf{A}, \mathbf{T_A}, \mathbf{B}, \sigma) \rightarrow \mathbf{R}$**.** *A randomized algorithm that receives a matrix $\mathbf{A}$, its trapdoor $\mathbf{T_A}$, a target matrix $\mathbf{B}$ and a gaussian parameter σ that satisfies $\sigma \geq \|\widetilde{\mathbf{T}}_\mathbf{A}\| \omega(\sqrt{\log m})$. It then outputs a matrix $\mathbf{R}$ distributed statistically close to $\mathcal{D}_{\Lambda_q^{\mathbf{B}}(\mathbf{A}), \sigma}$.*

$\mathsf{ExtendRight}(\mathbf{A}, \mathbf{T_A}, \mathbf{B}) \to \mathbf{T}_{[\mathbf{A}|\mathbf{B}]}$. *A deterministic algorithm that receives full-rank matrices* $\mathbf{A}, \mathbf{B} \in \mathbb{Z}_q^{n\times m}$, *a basis* $\mathbf{T_A}$ *of* $\Lambda_q^\perp(\mathbf{A})$ *and a gaussian parameter* $\sigma \geq \|\widetilde{\mathbf{T}}_\mathbf{A}\|\omega(\log n)$. *It then outputs a matrix* $\mathbf{T}_{[\mathbf{A}|\mathbf{B}]}$ *distributed statistically close to* $(\mathcal{D}_{\Lambda_q^\perp([\mathbf{A}|\mathbf{B}]),\sigma})^{2m}$ *such that* $\|\widetilde{\mathbf{T}}_{[\mathbf{A}|\mathbf{B}]}\| = \|\widetilde{\mathbf{T}}_\mathbf{A}\|$.

$\mathsf{ExtendLeft}(\mathbf{A}, \mathbf{B}, \mathbf{T_B}, \mathbf{R}) \to \mathbf{T}_{[\mathbf{A}|\mathbf{AR}+\mathbf{B}]}$. *A deterministic algorithm that receives full-rank matrices* $\mathbf{A}, \mathbf{B}, \mathbf{R} \in \mathbb{Z}_q^{n\times m}$, *a basis* $\mathbf{T_B}$ *of* $\Lambda_q^\perp(\mathbf{B})$ *and a gaussian parameter* $\sigma \geq \|\widetilde{\mathbf{T}}_\mathbf{A}\|\omega(\sqrt{\log n})$. *It then outputs a matrix* $\mathbf{T}_{[\mathbf{A}|\mathbf{AR}+\mathbf{B}]}$ *distributed statistically close to* $(\mathcal{D}_{\Lambda_q^\perp([\mathbf{A}|\mathbf{AR}+\mathbf{B}]),\sigma})^{2m}$ *such that* $\|\widetilde{\mathbf{T}}_{[\mathbf{A}|\mathbf{AR}+\mathbf{B}]}\| = \|\widetilde{\mathbf{T}}_\mathbf{B}\|(1 + \|\mathbf{R}\|_2)$.

Lemma 2 (Matrix Commitment [21]). *Let* λ *be a security parameter and let* q, ℓ, m *be lattice parameters. For* $\mathsf{pp} := (\mathbf{B}, \mathbf{W}, \mathbf{T})$, *where* $\mathbf{B} \in \mathbb{Z}_q^{n\times m}, \mathbf{W} \in \mathbb{Z}_q^{2m^2 n\times m}$, $\mathbf{T} \in \mathbb{Z}_q^{(2m^2+1)m\times 2m^3}$, $[\mathbf{I}_{2m^2} \otimes \mathbf{B}|\mathbf{W}]\mathbf{T} = \mathbf{I}_{2m^2} \otimes \mathbf{G}$, *there exists follow efficient algorithms* $(\mathsf{Com}, \mathsf{Ver}, \mathsf{Open})$ *where*

$\mathsf{Com}(\mathsf{pp}, \mathbf{x}) \to \mathbf{C_x}$. *It receives the public parameter* pp *and a vector* $\mathbf{x} \in \mathbb{Z}_q^\ell$ *and outputs the commitment* $\mathbf{C_x} \in \mathbb{Z}_q^{n\times m}$.

$\mathsf{Ver}(\mathsf{pp}, 1^\ell) \to \mathbf{V}$. *It receives the public parameter* pp *and the vector length* ℓ *and outputs a verification matrix* $\mathbf{V} \in \mathbb{Z}_q^{m\times \ell m}$.

$\mathsf{Open}(\mathsf{pp}, \mathbf{x}) \to \mathbf{Z_x}$. *It receives the public parameter* pp *and the vector* $\mathbf{x} \in \mathbb{Z}_q^\ell$ *and outputs an opening* $\mathbf{Z_x} \in \mathbb{Z}_q^{m\times \ell m}$.

For all $\ell \in \mathbb{N}, \mathbf{x} \in \mathbb{Z}_q^\ell$, *the matrices* $\mathbf{C_x} \leftarrow \mathsf{Com}(\mathsf{pp}, \mathbf{x}), \mathbf{V} \leftarrow \mathsf{Ver}(\mathsf{pp}, 1^\ell), \mathbf{Z_x} \leftarrow \mathsf{Open}(\mathsf{pp}, \mathbf{x})$ *satisfy:*

$$\mathbf{C_x} \cdot \mathbf{V} = \mathbf{x} \otimes \mathbf{G} - \mathbf{B} \cdot \mathbf{Z_x}$$
$$\|\mathbf{V}\| \leq O(\|\mathbf{T}\| \cdot m^4 \log q)$$
$$\|\mathbf{Z_x}\| \leq O(\|\mathbf{T}\| \cdot \log \ell \cdot m^7 \log q)$$

Lemma 3 (Gaussian Tail Bound). *Let* $\mathcal{D}_{\mathbb{Z},\sigma}$ *denote the discrete Gaussian distribution over* $\mathbb{Z}$ *with parameter* $\sigma > 0$. *For any* $\lambda \in \mathbb{N}$, *we have*

$$\Pr[\|\mathbf{x}\| \geq \sqrt{\lambda}\sigma \mid \mathbf{x} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}] \leq 2^{-\lambda}$$

Lemma 4 (Left-over Hash Lemma). *Let* λ *be a security parameter and let* q, ℓ, m *be lattice parameters. Let* P *be a probability distribution over* $\mathbb{Z}^m$ *where* $H_\infty(P) \geq 2n \log q$. *Then we have:*

$$(\mathbf{A}, \mathbf{Ar} \pmod q) \approx_s (\mathbf{A}, \mathbf{u})$$

where $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}$, $\mathbf{r} \leftarrow P$, $\mathbf{u} \leftarrow \mathbb{Z}_q^n$ *and* $H_\infty(P)$ *refers to the min-entropy of* P.

Lemma 5 (Noise Rerandomization [13]). *Let* λ *be a security parameter and let* q, ℓ, m *be lattice parameters and* $r > \max\{\omega(\sqrt{\log m}), \omega(\sqrt{\log \ell})\}$. *There exists an efficient algorithm* ReRand *that*

$$(\mathbf{A}, \mathbf{x} := \mathbf{v} + \mathbf{e}, \mathsf{ReRand}(\mathbf{A}, \mathbf{x}, \sigma)) \approx_s (\mathbf{A}, \mathbf{x}, \mathbf{y} := \mathbf{vA} + \mathbf{e}')$$

where $\mathbf{v} \in \mathbb{Z}_q^{1\times m}, \mathbf{e} \leftarrow \mathcal{D}_{\mathbb{Z}^{1\times m},r}$, $\mathbf{A} \in \mathbb{Z}^{m\times \ell}$ *and* $\sigma > \|\mathbf{A}\|$, $\mathbf{e}' \leftarrow \mathcal{D}_{\mathbb{Z}^{1\times \ell},2r\sigma}$.

Assumption 1 (ℓ-succinct LWE [20]) *Let λ be a security parameter and let m, q, σ be lattice parameters, where $m \geq 2n \log q$. Then we have*

$$(\mathbf{B}, \mathbf{sB} + \mathbf{e}, \mathbf{W}, \mathbf{T}) \approx_c (\mathbf{B}, \mathbf{u}, \mathbf{W}, \mathbf{T})$$

where $\mathbf{B} \leftarrow \mathbb{Z}_q^{n \times m}, \mathbf{s} \leftarrow \mathbb{Z}_q^{1 \times n}, \mathbf{e} \leftarrow \mathcal{D}_{\mathbb{Z}^{1 \times m}, \sigma}, \mathbf{u} \leftarrow \mathbb{Z}_q^{1 \times m}, \mathbf{W} \leftarrow \mathbb{Z}_q^{\ell n \times m}, \mathbf{T} \leftarrow [\mathbf{I}_\ell \otimes \mathbf{B} | \mathbf{W}]^{-1}(\mathbf{I}_\ell \otimes \mathbf{G})$.

2.2 Batched Identity Based Encryption

A batched IBE scheme consists of following five algorithms:

$\mathsf{Setup}(1^\lambda, 1^B) \rightarrow (\mathsf{mpk}, \mathsf{msk})$**.** The setup algorithm receives the security parameter 1^λ and the batch size 1^B. It outputs the master public key mpk and the master secret key msk.

$\mathsf{Digest}(\mathsf{mpk}, \{\mathsf{id}_1, \cdots, \mathsf{id}_B\}) \rightarrow \mathsf{dig}$**.** The digest-generation algorithm receives master public key mpk and a list of identities $\mathsf{id}_1, \cdots, \mathsf{id}_B$, then it returns a digest dig.

$\mathsf{Gen}(\mathsf{msk}, \mathsf{dig}, t) \rightarrow \mathsf{sk}$**.** The key-generation algorithm receives the master secret key msk, a digest dig, and a batch label t, then it returns a secret key sk.

$\mathsf{Enc}(\mathsf{mpk}, \mathsf{m}, \mathsf{id}, t) \rightarrow \mathsf{ct}$**.** The encryption algorithm receives the master public key mpk, a message m, an identity id and a batch label t, then it returns a ciphertext ct.

$\mathsf{Dec}(\mathsf{mpk}, \mathsf{ct}, \mathsf{sk}, \mathsf{dig}, \{\mathsf{id}_1, \cdots, \mathsf{id}_B\}, \mathsf{id}, t) \rightarrow \mathsf{m}$**.** The decryption algorithm receives the master public key mpk, a ciphertext ct, a secret key sk, a digest dig, a list of identities $\mathsf{id}_1, \cdots, \mathsf{id}_B$, an identity id and a batch label t, then it returns a message m.

Correctness. For all m, we require the following probability is $1 - \mathsf{negl}(\lambda)$, where $|S| = B$ and $\mathsf{id} \in S$.

$$\Pr\left[\mathsf{Dec}(\mathsf{mpk}, \mathsf{ct}, \mathsf{sk}, \mathsf{dig}, S, \mathsf{id}, t) = \mathsf{m} \;\middle|\; \begin{array}{c} (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^B) \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{m}, \mathsf{id}, t) \\ \mathsf{dig} \leftarrow \mathsf{Digest}(\mathsf{mpk}, S) \\ \mathsf{sk} \leftarrow \mathsf{Gen}(\mathsf{msk}, \mathsf{dig}, t) \end{array}\right]$$

Succinctness. We require the runtime of Gen to be at most $\mathsf{poly}(\log B)$ where B stands for the batch size. This also implies $|\mathsf{dig}| = \mathsf{poly}(\log B)$ and $|\mathsf{sk}| = \mathsf{poly}(\log B)$.

Security. The adaptive security is defined by the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$:

Init: The adversary $\mathcal{A}$ submits a target identity id^* and a batch label t^*.

Setup: The challenger $\mathcal{C}$ runs the setup algorithm to generate master public key mpk and master secret key msk, then sends mpk to adversary $\mathcal{A}$.

Query Phase 1: The adversary $\mathcal{A}$ may adaptively make secret key queries:

- $\mathcal{A}$ sends a list S of B identities along with a batch label t to $\mathcal{C}$. If $\mathsf{id}^* \in S$ and $t = t^*$, then $\mathcal{A}$ loses the game.
- Otherwise, $\mathcal{C}$ computes $\mathsf{dig} \leftarrow \mathsf{Digest}(\mathsf{mpk}, S)$ and $\mathsf{sk} \leftarrow \mathsf{Gen}(\mathsf{msk}, \mathsf{dig}, t)$, then sends sk to $\mathcal{A}$.

Challenge: The adversary $\mathcal{A}$ submits a pair of challenge messages $(\mathsf{m}_0, \mathsf{m}_1)$. The challenger $\mathcal{C}$ picks a random challenge bit $b \in \{0,1\}$ and replies with $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{m}_b, \mathsf{id}^*, t^*)$.

Query Phase 2: $\mathcal{A}$ may continue to request a polynomial number of queries like those in Query Phase 1, with the same restriction that $\mathcal{A}$ cannot request a secret key for a list S and batch label t where $\mathsf{id}^* \in S$ and $t = t^*$.

Guess: The adversary $\mathcal{A}$ outputs a bit b' and wins if $b' = b$.

We say a batched IBE scheme is selective security if for all efficient adversary $\mathcal{A}$, the advantage $\mathsf{Adv}_{\mathcal{A}}^{\mathrm{BIBE}}(\lambda) := \left|\Pr[\mathcal{A} \text{ wins}] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda)$.

3 Batched IBE from succinct LWE

We set identity space $\mathcal{ID} = \mathbb{Z}_q$, $\mathbf{g}' = (1, 2^1, \cdots, 2^{\lceil \log q \rceil - 1}) \in \mathbb{Z}^{1 \times \lceil \log q \rceil}$ and define an algorithm $\mathsf{GenCoeff}$, which takes as input a set of identities $S = \{\mathsf{id}_1 \in \mathbb{Z}_q, \cdots\}$ and outputs a coefficient vector.

$\mathsf{GenCoeff}(S) \to \mathbf{y}$**:** Suppose $|S| = B$, the algorithm first compute polynomial $P_S(X) := \prod_{\mathsf{id} \in S}(X - \mathsf{id}) = \sum_{i=0}^{B} y_i X^i$. Then, it set $\mathbf{y}' = (y_0, y_1, \cdots, y_B) \in \mathbb{Z}_q^{1 \times (B+1)}$ proceeds to compute and output the binary decomposition of $\mathbf{y}'$: $\mathbf{y} := (\mathbf{G}^{-1}((\mathbf{y}')^\top))^\top \in \{0,1\}^{1 \times (B+1)\lceil \log q \rceil}$.

3.1 Construction

$\mathsf{Setup}(1^\lambda, 1^B):$ Initialize the lattice parameters n, m, q. Sample

$$
\begin{aligned}
(\mathbf{B}, \mathbf{T}_{\mathbf{B}}) &\leftarrow \mathsf{TrapGen}(1^n, 1^m, q), \mathbf{W} \leftarrow \mathbb{Z}_q^{2m^2 n \times m}, \\
&\mathbf{T} \leftarrow \mathsf{SamplePre}([\mathbf{I}_{2m^2} \otimes \mathbf{B} \mid \mathbf{W}], \mathbf{I}_{2m^2} \otimes \mathbf{T}_{\mathbf{B}}, \mathbf{I}_{2m^2} \otimes \mathbf{G}, \sigma_0) \\
\mathbf{B}_1, \mathbf{B}_2, \mathbf{B}_3 &\leftarrow \mathbb{Z}_q^{n \times m}, \mathbf{P} \leftarrow \mathbb{Z}_q^{n \times \lambda}
\end{aligned}
$$

Output

$$
\begin{aligned}
\mathsf{mpk} &:= \big(\overbrace{\mathbf{B}, \mathbf{W}, \mathbf{T}}^{:=\mathsf{pp}}, \mathbf{B}_1, \mathbf{B}_2, \mathbf{B}_3, \mathbf{P}\big) \\
\mathsf{msk} &:= (\mathbf{T}_{\mathbf{B}})
\end{aligned}
$$

$\mathsf{Digest}(\mathsf{mpk}, S):$ Compute $\mathbf{y} \leftarrow \mathsf{GenCoeff}(S)$ and $\mathbf{V} \leftarrow \mathsf{Ver}(\mathsf{pp}, 1^{k(B+1)})$ where $k = \lceil \log q \rceil$. Output

$$
\mathsf{dig}_S := -\mathbf{B}_1 \mathbf{V}(\mathbf{y}^\top \otimes \mathbf{I}_m) \in \mathbb{Z}_q^{n \times m}
$$

$\mathsf{Gen}(\mathsf{msk}, \mathsf{dig}_S, t)$**:** Compute $\mathbf{T}_{[\mathbf{B}|\mathsf{dig}_S|\mathbf{B}_2 + t\mathbf{B}_3]} \leftarrow \mathsf{ExtendRight}(\mathbf{B}, \mathbf{T}_{\mathbf{B}}, [\mathsf{dig}_S|\mathbf{B}_2 + t\mathbf{B}_3])$

$$
\mathsf{sk} \leftarrow \mathsf{SamplePre}([\mathbf{B}|\mathsf{dig}_S|\mathbf{B}_2 + t\mathbf{B}_3], \mathbf{T}_{[\mathbf{B}|\mathsf{dig}_S|\mathbf{B}_2+t\mathbf{B}_3]}, \mathbf{P}, \sigma_1)
$$

$\mathsf{Enc}(\mathsf{mpk}, \mathsf{id}, t, \mathsf{m})$**:** Sample

$$\mathbf{s} \leftarrow \mathbb{Z}_q^{1\times n}, \mathbf{e}_0 \leftarrow \mathcal{D}_{\mathbb{Z}^{1\times m},\chi}, \mathbf{e}_1, \mathbf{e}_2 \leftarrow \mathcal{D}_{\mathbb{Z}^{1\times m},\chi'}, \mathbf{e}_3 \leftarrow \mathcal{D}_{\mathbb{Z}^{1\times \lambda},\chi'}$$

Set $\mathsf{ID} := (1, \mathsf{id}, \mathsf{id}^2, \cdots, \mathsf{id}^B) \otimes \mathbf{g}' \in \mathbb{Z}_q^{1\times k(B+1)}$ where $k = \lceil \log q \rceil$. Compute $\mathbf{C}_{\mathsf{ID}} \leftarrow \mathsf{Com}(\mathsf{pp}, \mathsf{ID})$. Output

$$\mathsf{ct} := \begin{pmatrix} \mathbf{c}_0 := \mathbf{sB} + \mathbf{e}_0 \in \mathbb{Z}_q^{1\times m}, & \mathbf{c}_1 := \mathbf{s}(\mathbf{B}_1 + \mathbf{C}_{\mathsf{ID}}) + \mathbf{e}_1 \in \mathbb{Z}_q^{1\times m} \\ \mathbf{c}_2 := \mathbf{s}(\mathbf{B}_2 + t\mathbf{B}_3) + \mathbf{e}_2 \in \mathbb{Z}_q^{1\times m}, & \mathbf{c}_3 := \mathbf{sP} + \mathbf{e}_3 + \mathsf{m}\lfloor \frac{q}{2} \rfloor \in \mathbb{Z}_q^{1\times \lambda} \end{pmatrix}$$

$\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}, S, \mathsf{id}, t, \mathsf{ct})$**:** Set $\mathsf{ID} := (1, \mathsf{id}^1, \cdots, \mathsf{id}^B) \otimes \mathbf{g}'$ where $k = \lceil \log q \rceil$. Compute $\mathbf{y} \leftarrow \mathsf{GenCoeff}(S)$, $\mathbf{V} \leftarrow \mathsf{Ver}(\mathsf{pp}, 1^{k(B+1)})$, $\mathbf{Z}_{\mathsf{ID}} \leftarrow \mathsf{Open}(\mathsf{pp}, \mathsf{ID})$ and

$$\mathbf{c}_4 = [\mathbf{c}_0|\mathbf{c}_1] \begin{pmatrix} -\mathbf{Z}_{\mathsf{ID}} \\ -\mathbf{V} \end{pmatrix} \cdot (\mathbf{y}^\top \otimes \mathbf{I}_m)$$

Output

$$\lfloor \frac{2}{q} \cdot (\mathbf{c}_3 - [\mathbf{c}_0|\mathbf{c}_4|\mathbf{c}_2] \cdot \mathsf{sk} \mod q) \rceil$$

Correctness. If $\mathsf{id} \in S$, then we have $\mathsf{ID} \cdot \mathbf{y}^\top = P_S(\mathsf{id}) = 0$, so

$$\begin{aligned} \mathbf{c}_4 =& [\mathbf{c}_0|\mathbf{c}_1] \begin{pmatrix} -\mathbf{Z}_{\mathsf{ID}} \\ -\mathbf{V} \end{pmatrix} \cdot (\mathbf{y}^\top \otimes \mathbf{I}_m) \\ =& \mathbf{s}[\mathbf{B}|\mathbf{B}_1 + \mathbf{C}_{\mathsf{ID}}] \begin{pmatrix} -\mathbf{Z}_{\mathsf{ID}} \\ -\mathbf{V} \end{pmatrix} \cdot (\mathbf{y}^\top \otimes \mathbf{I}_m) + \mathsf{noise}' \\ =& \mathbf{s}(-\mathbf{B}_1\mathbf{V} - \mathsf{ID} \otimes \mathbf{G}) \cdot (\mathbf{y}^\top \otimes \mathbf{I}_m) + \mathsf{noise}' \\ =& \mathbf{s} \cdot \mathsf{dig}_S - \mathbf{s}(\mathsf{ID} \cdot \mathbf{y}^\top) \otimes \mathbf{G} + \mathsf{noise}' \\ =& \mathbf{s} \cdot \mathsf{dig}_S + \mathsf{noise}' \end{aligned}$$

Then

$$\mathbf{c}_3 - [\mathbf{c}_0|\mathbf{c}_4|\mathbf{c}_2] \cdot \mathsf{sk} = \mathsf{m}\lfloor \frac{q}{2} \rfloor + \mathbf{sP} - \mathbf{s}[\mathbf{B}|\mathsf{dig}_S|\mathbf{B}_2 + t\mathbf{B}_3] \cdot \mathsf{sk} + \mathsf{noise} \approx \mathsf{m}\lfloor \frac{q}{2} \rfloor$$

The correctness follows from that

$$\begin{aligned} \|\mathsf{noise}\| =& \|\mathbf{e}_3 + [\mathbf{e}_0|\mathsf{noise}'] \cdot \mathsf{sk}\| \\ \leq& \|\mathbf{e}_3\| + \|\mathsf{sk}\| \cdot \left(\|\mathbf{e}_0\| + \left\| [\mathbf{e}_0|\mathbf{e}_1] \begin{pmatrix} -\mathbf{Z}_{\mathsf{ID}} \\ -\mathbf{V} \end{pmatrix} \cdot (\mathbf{y}^\top \otimes \mathbf{I}_m) \right\| + \|\mathbf{e}_2\| \right) \\ \leq& (\chi + \chi')\sigma_1 \cdot O(\|\mathbf{T}\| \cdot m^8 B \log B \log^2 q) \leq \frac{q}{4} \end{aligned}$$

Parameters. We set LWE parameters $m = n \cdot \mathsf{poly}(\lambda), q = \mathsf{poly}(n, B, \lambda), \chi = \mathsf{poly}(n, \lambda)$ to satisfy the following conditions:

$$\begin{aligned}
&\frac{q}{4} > (\chi + \chi') \cdot \sigma_0 \cdot \sigma_1 \cdot B \log B \log^2 q \cdot \mathsf{poly}(m, \lambda) && \text{(correctness)}\\
&m > 2(n+1)\log q && \text{(LHL)}\\
&\sigma_0 = \mathsf{poly}(m, \lambda) && (2m^2\text{-succinct LWE})\\
&\sigma_1 > \sigma_0 B \log B \log^2 q \cdot \mathsf{poly}(m, \lambda) && \text{(Trapdoor Sampling)}\\
&\chi > \omega(\sqrt{m}) && \text{(ReRand)}\\
&\chi' > 2m\chi && \text{(ReRand)}
\end{aligned}$$

3.2 Security

Theorem 1. *Let λ be the security parameter, and the lattice parameters described in Section 3.1. Then, under $2m^2$-succinct LWE assumption, the above batched IBE construction is selectively secure.*

Proof. We prove the theorem via a sequence of games.

– $\mathsf{Game}_{\mathsf{real}}$: The real game. We have $\mathsf{mpk} = \{\mathbf{B}, \mathbf{W}, \mathbf{T}, \mathbf{B}_1, \mathbf{B}_2, \mathbf{B}_3, \mathbf{P}\}$, where:

$$\mathbf{W} \leftarrow \mathbb{Z}_q^{2m^2 n \times m}, \mathbf{B}_1, \mathbf{B}_2, \mathbf{B}_3 \leftarrow \mathbb{Z}_q^{n \times m}, \mathbf{P} \leftarrow \mathbb{Z}_q^{n \times \lambda}$$

and

$$\begin{aligned}
(\mathbf{B}, \mathbf{T}_{\mathbf{B}}) \leftarrow &\mathsf{TrapGen}(1^n, 1^m, q)\\
\mathbf{T} \leftarrow &\mathsf{SamplePre}([\mathbf{I}_{2m^2} \otimes \mathbf{B} | \mathbf{W}], \mathbf{I}_{2m^2} \otimes \mathbf{T}_{\mathbf{B}}, \mathbf{I}_{2m^2} \otimes \mathbf{G}, \sigma_0)
\end{aligned}$$

The challenge ciphertext is constructed as follows:

$$\mathsf{ct} := (\mathbf{s}\mathbf{B} + \mathbf{e}_0, \mathbf{s}(\mathbf{B}_1 + \mathbf{C}_{\mathsf{ID}^*}) + \mathbf{e}_1, \mathbf{s}(\mathbf{B}_2 + t^*\mathbf{B}_3) + \mathbf{e}_2, \mathbf{s}\mathbf{P} + \mathsf{m}_b\lfloor \frac{q}{2} \rfloor + \mathbf{e}_3)$$

where $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, $\mathsf{ID}^* := (1, \mathsf{id}^*, \cdots, (\mathsf{id}^*)^B) \otimes \mathbf{g}'$, $\mathbf{C}_{\mathsf{ID}^*} \leftarrow \mathsf{Com}(\mathsf{pp}, \mathsf{ID}^*)$. The secret key sk for identity set S and batch label t as constructed as follows:

$$\mathsf{sk} \leftarrow \mathsf{SamplePre}([\mathbf{B} | \mathsf{dig}_S | \mathbf{B}_2 + t\mathbf{B}_3], \mathbf{T}_{[\mathbf{B}|\mathsf{dig}_S|\mathbf{B}_2 + t\mathbf{B}_3]}, \mathbf{P}, \sigma_1)$$

where $k = \lceil \log q \rceil$ and

$$\begin{aligned}
\mathbf{T}_{[\mathbf{B}|\mathsf{dig}_S|\mathbf{B}_2 + t\mathbf{B}_3]} \leftarrow &\mathsf{ExtendRight}(\mathbf{B}, \mathbf{T}_{\mathbf{B}}, [\mathsf{dig}_S | \mathbf{B}_2 + t\mathbf{B}_3]),\\
\mathbf{y} \leftarrow &\mathsf{GenCoeff}(S),\\
\mathsf{dig}_S := &-\mathbf{B}_1 \mathbf{V}(\mathbf{y}^\top \otimes \mathbf{I}_m),\\
\mathbf{V} \leftarrow &\mathsf{Ver}(\mathsf{pp}, 1^{k(B+1)})
\end{aligned}$$

- Game_0: This game is the same as $\mathsf{Game}_{\mathsf{real}}$ except that

$$\begin{aligned}(\mathbf{B}_3, \mathbf{T}_{\mathbf{B}_3}) \leftarrow & \mathsf{TrapGen}(1^n, 1^m, q),\\ \mathbf{B}_1 := & \mathbf{B}\mathbf{U}_1 - \mathbf{C}_{\mathsf{ID}^*},\\ \mathbf{B}_2 := & \mathbf{B}\mathbf{U}_2 - t^*\mathbf{B}_3,\\ \mathbf{P} := & \mathbf{B}\mathbf{U}\end{aligned}$$

  where $\mathbf{U}_1, \mathbf{U}_2 \leftarrow \{0,1\}^{m\times m}, \mathbf{U} \leftarrow \{0,1\}^{m\times\lambda}$. $\mathcal{C}$ keeps $\mathbf{T}_{\mathbf{B}_3}$ in secret.

- Game_1: This game is the same as Game_0 except that

$$\begin{aligned}\mathbf{c}_1 := & \mathsf{ReRand}(\mathbf{U}_1, \mathbf{c}_0, \chi'/(2\chi)),\\ \mathbf{c}_2 := & \mathsf{ReRand}(\mathbf{U}_2, \mathbf{c}_0, \chi'/(2\chi)),\\ \mathbf{c}_3 := & \mathsf{ReRand}(\mathbf{U}, \mathbf{c}_0, \chi'/(2\chi)) + \mathsf{m}_b\lfloor\frac{q}{2}\rfloor\end{aligned}$$

- Game_2: This game is the same as Game_1 except that the way sk for identity set S and batched label $t = t^*$ is generated, where $\mathsf{id}^* \notin S$. To generate such sk, $\mathcal{C}$ computes

$$\mathsf{sk} \leftarrow \mathsf{SamplePre}([\mathbf{B}|\mathsf{dig}_S|\mathbf{B}_2 + t\mathbf{B}_3], \begin{pmatrix}(\mathbf{Z}_{\mathsf{ID}^*} + \mathbf{U}_1\mathbf{V})(\mathbf{y}^\top \otimes \mathbf{I}_m)\\ \mathbf{I}_m\\ \mathbf{0}^{m\times m}\end{pmatrix}, \mathbf{P}, \sigma_1)$$

- Game_3: This game is the same as Game_2 except that the way sk for identity set S and batched label $t \neq t^*$ generated, where $\mathsf{id}^* \in S$. To generate such sk, $\mathcal{C}$ will do as follows:
  - Generate trapdoor by

$$\mathbf{T}' \leftarrow \mathsf{ExtendLeft}([\mathbf{B}|\mathsf{dig}_S], (t - t^*)\mathbf{B}_3, \mathbf{T}_{\mathbf{B}_3}, [\mathbf{U}_2^\top|\mathbf{0}^{m\times m}]^\top)$$

  - Compute $\mathsf{sk} \leftarrow \mathsf{SamplePre}([\mathbf{B}|\mathsf{dig}_S|\mathbf{B}_2 + t\mathbf{B}_3], \mathbf{T}', \mathbf{P}, \sigma_1)$
- Game_4: This game is the same as Game_3 except that $\mathbf{c}_0 \leftarrow \mathbb{Z}_q^{1\times m}$.
- Game_5: This game is the same as Game_4 except that $\mathbf{c}_3 \leftarrow \mathbb{Z}_q^{1\times\lambda}$.

Lemma 6. *Suppose $m > 2n\log q$, we have $\mathsf{Game}_{\mathit{real}} \approx_s \mathsf{Game}_0$.*

Proof. This follows from Left-over Hash Lemma and the properties of $\mathsf{TrapGen}$ function.

Lemma 7. *Suppose $\chi > \max\{\omega(\sqrt{m}), \omega(\sqrt{\lambda})\}$, $\chi' > 2\chi\max\{m, \lambda\}$, we have* $\mathsf{Game}_0 \approx_s \mathsf{Game}_1$

Proof. This follows from the properties of the ReRand function, as

$$\begin{aligned}\chi > & \max\{\omega(\sqrt{m}), \omega(\sqrt{\lambda})\},\\ \chi'/(2\chi) > & \max\{\|\mathbf{U}_1\|, \|\mathbf{U}_2\|, \|\mathbf{U}\|\}\end{aligned}$$

where $\mathbf{c}_0 = \mathbf{s}\mathbf{B} + \mathbf{e}_0$, $\mathbf{e}_0 \leftarrow \mathcal{D}_{\mathbb{Z}^{1\times m},\chi}$.

Lemma 8. *Suppose $\sigma_1 > \sigma_0 B \log B \log^2 q \cdot \mathit{poly}(m,\lambda)$, we have* $\mathsf{Game}_1 \approx_s \mathsf{Game}_2$.

Proof. Since $\mathsf{id}^* \notin S$, we have that

$$[\mathbf{B}|\mathbf{B}\mathbf{U}_1]\begin{pmatrix}-\mathbf{Z}_{\mathsf{ID}^*}\\-\mathbf{V}\end{pmatrix}(\mathbf{y}^\top \otimes \mathbf{I}_m) = (-\mathbf{B}_1\mathbf{V} - \mathsf{ID}^* \otimes \mathbf{G})(\mathbf{y}^\top \otimes \mathbf{I}_m) = -\mathsf{dig}_S - \mathsf{ID}^*\mathbf{y}^\top\mathbf{G}$$

which means that

$$[\mathbf{B}|\mathsf{dig}_S|\mathbf{B}_2 + t\mathbf{B}_3]\begin{pmatrix}(\mathbf{Z}_{\mathsf{ID}^*} + \mathbf{U}_1\mathbf{V})(\mathbf{y}^\top \otimes \mathbf{I}_m)\\ \mathbf{I}_m \\ \mathbf{0}^{m\times m}\end{pmatrix} = \mathsf{ID}^*\mathbf{y}^\top\mathbf{G}$$

where $\mathsf{ID}^*\mathbf{y}^\top \neq 0$, $\mathbf{y} = \mathsf{GenCoeff}(S)$. Moreover, trapdoor sampling requires that $\sigma_1 > \sigma_0 B \log B \log^2 q \cdot \mathsf{poly}(m,\lambda)$ because

$$\left\|\begin{pmatrix}(\mathbf{Z}_{\mathsf{ID}^*} + \mathbf{U}_1\mathbf{V})(\mathbf{y}^\top \otimes \mathbf{I}_m)\\ \mathbf{I}_m \\ \mathbf{0}^{m\times m}\end{pmatrix}\right\| = \sigma_0 B \log B \log^2 q \cdot \mathsf{poly}(m,\lambda)$$

Lemma 9. *Suppose $\sigma_1 > \omega(\sqrt{n \log q})$, we have* $\mathsf{Game}_2 \approx_s \mathsf{Game}_3$.

Proof. Since $t \neq t^*$, we have

$$[\mathbf{B}|\mathbf{B}_2 + t\mathbf{B}_3] = [\mathbf{B}|\mathbf{B}\mathbf{U}_2 + (t - t^*)\mathbf{B}_3]$$

So $\mathcal{C}$ can use $\mathbf{T}_{\mathbf{B}_3}$ to generate the trapdoor for $[\mathbf{B}|\mathbf{B}_2 + t\mathbf{B}_3]$.

Lemma 10. *Under $2m^2$-succinct LWE assumption, we have* $\mathsf{Game}_3 \approx_c \mathsf{Game}_4$.

Proof. Suppose there exists a bit $b \in \{0,1\}$ and an efficient adversary $\mathcal{A}$ that can distinguish between Game_3 and Game_4 with non-negligible advantage $\epsilon > 0$. We construct an algorithm $\mathcal{C}$ that uses $\mathcal{A}$ as a subroutine to solve the $2m^2$-succinct LWE problem with non-negligible advantage:

Init: The adversary $\mathcal{A}$ submits a target identity id^* and a target batched label t^*.

Setup: $\mathcal{C}$ uses $2m^2$ to get $(\mathbf{B}, \mathbf{u}, \mathbf{W}, \mathbf{T})$ as an instance of the $2m^2$-succinct LWE problem. $\mathcal{C}$ samples $\mathbf{U}_1, \mathbf{U}_2 \leftarrow \{0,1\}^{m\times m}, \mathbf{U} \leftarrow \{0,1\}^{m\times\lambda}$ and $(\mathbf{B}_3, \mathbf{T}_{\mathbf{B}_3}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$. Then, $\mathcal{C}$ sets $\mathsf{ID}^* := (1, \mathsf{id}^*, (\mathsf{id}^*)^2, \cdots, (\mathsf{id}^*)^B) \otimes \mathbf{g}'$. Finally, $\mathcal{C}$ gives mpk to $\mathcal{A}$ where

$$\mathsf{mpk} := (\overbrace{\mathbf{B}, \mathbf{W}, \mathbf{T}}^{:=\mathsf{pp}}, \mathbf{B}_1 := \mathbf{B}\mathbf{U}_1 - \mathbf{C}_{\mathsf{ID}^*}, \mathbf{B}_2 := \mathbf{B}\mathbf{U}_2 - t^*\mathbf{B}_3, \mathbf{P} := \mathbf{B}\mathbf{U})$$

among which $\mathbf{C}_{\mathsf{ID}^*} := \mathsf{Com}(\mathsf{pp}, \mathsf{ID}^*)$.

Query Phase 1: The adversary $\mathcal{A}$ may adaptively make secret key queries for identity set S and batched label t which satisfy $\mathsf{id} \notin S$ or $t \neq t^*$:

– If $\mathsf{id}^* \notin S$, $\mathcal{C}$ computes

$$\mathsf{sk} \leftarrow \mathsf{SamplePre}([\mathbf{B}|\mathsf{dig}_S|\mathbf{B}_2 + t\mathbf{B}_3], \begin{pmatrix} (\mathbf{Z}_{\mathsf{ID}^*} + \mathbf{U}_1\mathbf{V})(\mathbf{y}^\top \otimes \mathbf{I}_m) \\ \mathbf{I}_m \\ \mathbf{0}^{m\times m} \end{pmatrix}, \mathbf{P}, \sigma_1)$$

where

$$\mathbb{Z}_{\mathsf{ID}^*} := \mathsf{Open}(\mathsf{pp}, \mathsf{ID}^*), \mathbf{V} \leftarrow \mathsf{Ver}(\mathsf{pp}, 1^{k(B+1)}), \mathsf{dig}_S \leftarrow \mathsf{Digest}(\mathsf{mpk}, S)$$

– If $\mathsf{id}^* \in S$, $\mathcal{C}$ will do as follows:
  - Generate trapdoor by

$$\mathbf{T}' \leftarrow \mathsf{ExtendLeft}([\mathbf{B}|\mathsf{dig}_S], (t - t^*)\mathbf{B}_3, \mathbf{T}_{\mathbf{B}_3}, [\mathbf{U}_2^\top|\mathbf{0}^{m\times m}]^\top)$$

  - Compute $\mathsf{sk} \leftarrow \mathsf{SamplePre}([\mathbf{B}|\mathsf{dig}_S|\mathbf{B}_2 + t\mathbf{B}_3], \mathbf{T}', \mathbf{P}, \sigma_1)$

Challenge: The adversary $\mathcal{A}$ submits a pair of challenge messages $(\mathsf{m}_0, \mathsf{m}_1)$. $\mathcal{C}$ outputs challenge ciphertext ct as

$$\mathsf{ct} := \begin{Bmatrix} \mathbf{c}_0 := \mathbf{u}, & \mathbf{c}_1 := \mathsf{ReRand}(\mathbf{U}_1, \mathbf{u}, \chi'/(2\chi)), \\ \mathbf{c}_2 := \mathsf{ReRand}(\mathbf{U}_2, \mathbf{u}, \chi'/(2\chi)), & \mathbf{c}_3 := \mathsf{ReRand}(\mathbf{U}, \mathbf{u}, \chi'/(2\chi)) + \mathsf{m}_b\lfloor \frac{q}{2} \rfloor \end{Bmatrix}$$

Query Phase 2: $\mathcal{A}$ may continue to request a polynomial number of queries like those in Query Phase 1, $\mathcal{C}$ replies $\mathcal{A}$ as in Query Phase 1.
Guess: The adversary $\mathcal{A}$ outputs a guess b' and $\mathcal{C}$ outputs $b = b'$.

It is clear that $\mathcal{C}$ correctly simulates an execution of Game_3 and Game_4 for $\mathcal{A}$:

– The way for selecting public parameters is the same as in Game_3 and Game_4.
– The way for generating sk is the same as in Game_3 and Game_4.
– Consider the challenge ciphertext:
  - If $\mathbf{u} = \mathbf{sB} + \mathbf{e}_0$, then $\mathbf{c}_0 = \mathbf{sB} + \mathbf{e}_0$ which is the same as in Game_3.
  - If $\mathbf{u} \leftarrow \mathbb{Z}_q^{1\times m}$, then $\mathbf{c}_0 = \mathbf{u}$ assumes a uniform distribution on $\mathbb{Z}_q^{1\times m}$ which is the same as in Game_4.

Lemma 11. *Suppose* $m > 2(n+1)\log q$, *we have* $\mathsf{Game}_4 \approx_s \mathsf{Game}_5$.

Proof. This follows from Left-over Hash Lemma.

Acknowledgements. We are grateful to Hoeteck Wee for his insightful advice and helpful discussions.

References

1. Amit Agarwal, Rex Fernando, and Benny Pinkas. Efficiently-thresholdizable batched identity based encryption, with applications. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part III*, volume 16002 of *LNCS*, pages 69–100. Springer, Cham, August 2025. 1, 2, 3, 4

2. Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 553–572. Springer, Berlin, Heidelberg, May / June 2010. 8
3. 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, Berlin, Heidelberg, December 2011. 5
4. Allison Bishop, Abhishek Jain, and Lucas Kowalczyk. Function-hiding inner product encryption. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 470–491. Springer, Berlin, Heidelberg, November / December 2015. 5
5. Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 533–556. Springer, Berlin, Heidelberg, May 2014. 5, 8
6. Dan Boneh, Evan Laufer, and Ertem Nusret Tas. Batch decryption without epochs and its application to encrypted mempools. *Cryptology ePrint Archive*, 2025. 2, 3
7. Jeffrey Champion, Yao-Ching Hsieh, and David J. Wu. Registered ABE and adaptively-secure broadcast encryption from succinct LWE. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part III*, volume 16002 of *LNCS*, pages 3–34. Springer, Cham, August 2025. 2
8. Jeffrey Champion and David J. Wu. Distributed broadcast encryption from lattices. In Elette Boyle and Mohammad Mahmoody, editors, *TCC 2024, Part III*, volume 15366 of *LNCS*, pages 156–189. Springer, Cham, December 2024. 2
9. Jie Chen, Benoît Libert, and Somindu C. Ramanna. Non-zero inner product encryption with short ciphertexts and private keys. In Vassilis Zikas and Roberto De Prisco, editors, *SCN 16*, volume 9841 of *LNCS*, pages 23–41. Springer, Cham, August / September 2016. 5
10. Valerio Cini, Russell W. F. Lai, and Giulio Malavolta. Lattice-based succinct arguments from vanishing polynomials - (extended abstract). In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part II*, volume 14082 of *LNCS*, pages 72–105. Springer, Cham, August 2023. 2
11. Junqing Gong, Brent Waters, Hoeteck Wee, and David J. Wu. Threshold batched identity-based encryption from pairings in the plain model. Cryptology ePrint Archive, Paper 2025/2103, 2025. 2, 3
12. Shuichi Katsumata, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Adaptively secure inner product encryption from LWE. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part III*, volume 12493 of *LNCS*, pages 375–404. Springer, Cham, December 2020. 5
13. Shuichi Katsumata and Shota Yamada. Partitioning via non-linear polynomial functions: More compact IBEs from ideal lattices and bilinear maps. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 682–712. Springer, Berlin, Heidelberg, December 2016. 9
14. Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 146–162. Springer, Berlin, Heidelberg, April 2008. 4
15. Sam Kim, Kevin Lewi, Avradip Mandal, Hart Montgomery, Arnab Roy, and David J. Wu. Function-hiding inner product encryption is practical. In Dario

Catalano and Roberto De Prisco, editors, *SCN 18*, volume 11035 of *LNCS*, pages 544–562. Springer, Cham, September 2018. 5

16. Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 700–718. Springer, Berlin, Heidelberg, April 2012. 5
17. 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, Berlin, Heidelberg, August 1984. 1
18. Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Berlin, Heidelberg, May 1997. 2
19. Zhedong Wang, Xiong Fan, and Mingsheng Wang. Compact inner product encryption from LWE. In Sihan Qing, Chris Mitchell, Liqun Chen, and Dongmei Liu, editors, *ICICS 17*, volume 10631 of *LNCS*, pages 141–153. Springer, Cham, December 2017. 5
20. Hoeteck Wee. Circuit ABE with $\mathsf{poly}(\text{depth}, \lambda)$-sized ciphertexts and keys from lattices. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part III*, volume 14922 of *LNCS*, pages 178–209. Springer, Cham, August 2024. 2, 7, 10
21. Hoeteck Wee. Almost optimal KP and CP-ABE for circuits from succinct LWE. In Serge Fehr and Pierre-Alain Fouque, editors, *EUROCRYPT 2025, Part III*, volume 15603 of *LNCS*, pages 34–62. Springer, Cham, May 2025. 6, 9
22. Hoeteck Wee and David J. Wu. Lattice-based functional commitments: Fast verification and cryptanalysis. In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT 2023, Part V*, volume 14442 of *LNCS*, pages 201–235. Springer, Singapore, December 2023. 2
23. Hoeteck Wee and David J. Wu. Succinct vector, polynomial, and functional commitments from lattices. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part III*, volume 14006 of *LNCS*, pages 385–416. Springer, Cham, April 2023. 2
24. Hoeteck Wee and David J. Wu. Unbounded distributed broadcast encryption and registered ABE from succinct LWE. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part III*, volume 16002 of *LNCS*, pages 204–235. Springer, Cham, August 2025. 2
25. 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, Berlin, Heidelberg, February / March 2013. 5