Identity-Based Encryption with Security against the KGC: A Formal Model and Its Instantiations[*]

Keita Emura[§], Shuichi Katsumata[¶], and Yohei Watanabe[†]

[§]National Institute of Information and Communications Technology (NICT), Japan.
[¶]National Institute of Advanced Industrial Science and Technology (AIST), Japan.
[†]The University of Electro-Communications, Japan.
k-emura@nict.go.jp, shuichi.katsumata@aist.go.jp, watanabe@uec.ac.jp

December 2, 2019

Abstract

The *key escrow problem* is one of the main barriers to the widespread real-world use of identity-based encryption (IBE). Specifically, a key generation center (KGC), which generates secret keys for a given identity, has the power to decrypt all ciphertexts. At PKC 2009, Chow defined a notion of security against the KGC, that relies on assuming that it cannot discover the underlying identities behind ciphertexts. However, this is not a realistic assumption since, in practice, the KGC manages an identity list, and hence it can easily guess the identities corresponding to given ciphertexts. Chow later amended this issue by introducing a new entity called an identity-certifying authority (ICA) and proposed an *anonymous key-issuing protocol.* Essentially, this allows the users, KGC, and ICA to interactively generate secret keys without users ever having to reveal their identities to the KGC. Unfortunately, since Chow separately defined the security of IBE and that of the anonymous key-issuing protocol, his IBE definition did not provide any formal treatment when the ICA is used to authenticate the users. Effectively, all of the subsequent works following Chow lack the formal proofs needed to determine whether or not it delivers a secure solution to the key escrow problem.

In this paper, based on Chow's work, we formally define an IBE scheme that resolves the key escrow problem and provide formal definitions of security against corrupted users, KGC, and ICA. Along the way, we observe that if we are allowed to assume a fully trusted ICA, as in Chow's work, then we can construct a trivial (and meaningless) IBE scheme that is secure against the KGC. Finally, we present two instantiations in our new security model: a lattice-based construction based on the Gentry–Peikert–Vaikuntanathan IBE scheme (STOC 2008) and Rückert's lattice-based blind signature scheme (ASIACRYPT 2010), and a pairing-based construction based on the Boneh–Franklin IBE scheme (CRYPTO 2001) and Boldyreva's blind signature scheme (PKC 2003).

1 Introduction

1.1 Identity-Based Encryption

Public key cryptography has long been in widespread real-world use, but it has the issue that public keys look like random strings. Consequently, public key infrastructure (PKI) has also been

[*]An extended abstract appears in the 24th European Symposium on Research in Computer Security (ESORICS 2019) [14].

developed to prove the validity of public keys. *Identity-based encryption* (IBE) [33] can reduce the costs associated with PKI systems by enabling users to select arbitrary strings (such as e-mail addresses or bio-information) as public keys. A special entity called the key-generation center (KGC) maintains a master public/secret key pair $(\mathsf{mpk}, \mathsf{msk})$. The KGC (implicitly) confirms the validity of each user ID and then issues an associated secret key $\mathsf{sk}_{\mathsf{ID}}$ using the master secret key msk. Once the master public key mpk has been downloaded from the KGC, anyone can encrypt messages to any user as long as they know the recipient's ID.

1.2 Key Escrow Problem and Current Solutions

The *key escrow problem* is a significant barrier to the widespread real-world use of IBE, and is a severe concern for communication privacy. Notably, the KGC potentially has the power to decrypt all ciphertexts, since it can generate secret keys for any ID. Several attempts have already been made to deal with this issue by reducing the amount of trust on the KGC.

One line of research is to make users participate in the secret key-generation process. In certificateless encryption (CE) [3], in addition to the secret key $\mathsf{sk}_{\mathsf{ID}}$ generated by the KGC, each users generate their own public/secret key pair $(\mathsf{pk}, \mathsf{sk})$. Here, both the ID and pk are required to encrypt messages, so decryption involves both $\mathsf{sk}_{\mathsf{ID}}$ and sk. Effectively, the KGC can no longer decrypt ciphertexts since it does not know sk. However, requiring additional individual public keys detracts one of the main merits of IBE since the size of public information grows linearly with the number of users in the system. Garg et al. [15, 16] proposed registration-based encryption (RBE), which improved the CE approach by having the KGC aggregate and succinctly compress all users' public keys into the master public key mpk. Instead of generating a secret key $\mathsf{sk}_{\mathsf{ID}}$ for each user, the KGC only needs to update and maintain mpk using the pair $(\mathsf{ID}, \mathsf{pk})$ of each user ID. As in standard IBE, encryption and decryption only require $(\mathsf{mpk}, \mathsf{ID})$ and sk, respectively. Recently, Goyal and Vusirikala extended Garg et al.'s works to consider the verifiability of the key accumulation process [19]. One drawback when implementing RBE in practice, however, is that mpk must be periodically updated by the KGC, and the users must fetch this updated information.

Another approach is to define an independent notion of security against the KGC for *standard* IBEs. Here, we call an IBE standard if encryption requires only a *static* mpk and ID, and decryption requires only $\mathsf{sk}_{\mathsf{ID}}$, as originally defined in [33]. We want to define a notion that captures some type of anonymity [1, 8] for the KGC. In other words, we want to guarantee that ciphertexts remain anonymous and that the KGC cannot determine the correct identity needed to decrypt a given ciphertext. Based on this high-level idea, Izabachène and Pointcheval [22] formalized anonymity concerning the KGC for identity-based key encapsulation mechanisms (IB-KEMs). However, as Chow [12] pointed out, their definition is incomplete since it considers the situation where an adversary can only obtain the challenge ciphertext, whereas, in a standard IB-KEM, adversaries can obtain both the challenge ciphertext and the corresponding session key. In order to define a more stringent notion of security against the KGC, Chow [12] introduced the notion of KGC anonymous ciphertext indistinguishability (ACI-KGC), which guarantees that the KGC cannot obtain any information about the corresponding plaintext from a ciphertext *if the user's identity is chosen randomly and is unknown to the KGC.* However, as he noted, requiring ACI-KGC is still insufficient in practice: the KGC typically manages a list of issued identity/secret key pairs, so it could decrypt any ciphertext via brute force by running the decryption algorithm against all the secret keys issued so far. In other words, even though ACI-KGC is a well-motivated security definition, it does not capture real-world scenarios. To resolve this gap between the security notion and practical implementation, Chow also introduced, in the same paper, a new entity called an *identity-certifying authority* (ICA) and defined an *anonymous key-issuing protocol.* In this protocol, the ICA authenticates the user's identity ID by providing them certificates. The user can then use this certificate to interact with the KGC and obtain $\mathsf{sk}_{\mathsf{ID}}$ without revealing their identity ID, an idea reminiscent of blind IBEs [10, 20].[1] Since the KGC is now outsourcing the work of authenticating users to the ICA, it will no longer know which identities it has generated secret keys for.

Chow's work [12] was a significant step toward defining a standard IBE scheme that can resolve the key escrow problem. However, in this research, we identify some deficiencies in this formulation and show that the definition must be refined. First, as explained above, Chow introduced the ICA and proposed an anonymous key-issuing protocol involving the user, the KGC, and the ICA to make the definition more practical. However, unfortunately, ACI-KGC and security of the anonymous key-issuing protocol were separately defined; Chow defined ACI-KGC only between the user and the KGC and did not provide any formal treatment when the ICA is used to authenticate the users. He provided some informal argument suggesting that something similar to ACI-KGC should hold for a standard IBE scheme in the presence of the ICA. However, on closer look, ACI-KGC is not a notion which can be naturally extended to such scenarios.[2] In fact, in case the ICA is fully trusted, as in Chow's work, we observe that there exists a trivial construction that meets the definition while such a construction should be deemed insecure in practice. Consider the following IBE scheme, defined between the users, the KGC', and the fully trusted ICA': the ICA' plays the role of the KGC in a standard IBE scheme, and the KGC' does nothing. It is easy to see that this construction achieves ACI-KGC security since the KGC holds no secret information, and standard anonymous IBE readily implies ACI-KGC. We have simply transferred all the trust from the KGC to the ICA and also deferred the key escrow problem to it. This shows that, for a well-defined and well-motivated security notion, we cannot fully trust the ICA. Considering the relevance of key escrow problems for IBE in practice, we must provide a formal treatment of ACI-KGC in the presence of an ICA, which avoids this insecure construction. We remark that all subsequent works [29, 34] have followed Chow's definition. Finally, since all prior schemes are based on pairing, constructing a *post-quantum* IBE under any sensible security notion that resolves the key escrow problem remains open.

1.3 Our Contribution

In this paper, we formalize a standard IBE scheme that captures the key escrow problem, and provide two instantiations based on lattices and pairings. Our formalization is inspired by Chow's original work and is based on the idea of creating an ICA to authenticate the system's users. We describe this scheme as *blind IBE with certified identities* to differentiate between prior formalizations. This terminology follows from the fact that the proposed secret key-generation process can be seen as a blind IBE combined with an ICA to certify user identities. Under our new definition, we propose a lattice-based and pairing-based constructions. At a high level, both of our constructions follow a similar template by carefully combining a standard anonymous IBE scheme (that is insecure under key escrows) with an appropriate blind signature scheme. Both constructions are secure in the random oracle model. We note that as far as we are aware, this is the first post-quantum IBE to resolve the key escrow problem based on Chow's work.[3] Our contributions are

[1] Note that without an ICA, this can never be secure since a malicious user can obtain $\mathsf{sk}_{\mathsf{ID}}$ for any ID without identification. Recall that in practice KGC implicitly authenticates the users to which it provides $\mathsf{sk}_{\mathsf{ID}}$.

[2] More precisely, no key issuing process is defined, and the ICA is not involved in ACI-KGC. Chow analyzed the anonymous key-issuing protocol in a 2-party computation fashion. Although he briefly claimed that the anonymous key-issuing protocol could be used securely with the IBE scheme, it is not clear whether the security preserves since the original ACI-KGC security restricts the adversary from participating in the key-issuing protocol.

[3] We note that another potential path to resolving the key escrow problem may be to consider distributed KGCs [6]. For example, Kumar and Chand [25] proposed a pairing-based scheme by considering multiple cloud privacy centers.

summarized in more detail in the following.

Formalization of a blind IBE with certified identities. We formalize a standard IBE scheme (which we call blind IBE with certified identities) that resolves the key escrow problem based on Chow's work [12]. Our definition involves three entities: the users, the KGC, and the ICA. The ICA is responsible for authenticating users by issuing certificates, while the KGC is responsible for (blindly) generating secret keys for users. We define three security notions, one for each of the three entities involved. Specifically, we define indistinguishability and anonymity against chosen plaintext attacks for the users (IND-ANON-CPA), the KGC (IND-ANON-KGC), and the ICA (IND-ANON-ICA). The first of these, IND-ANON-CPA, captures the standard concept of IBE security, while the second and third model cases where the KGC or the ICA is malicious. Our IBE formalization takes all the aforementioned issues into account: the syntax captures anonymous key-issuing via IND-ANON-KGC security, and the ICA is no longer fully trusted due to our additional definition of IND-ANON-ICA security. Our formalization can be seen as a natural and formal extension of Chow's idea of combining ACI-KGC security with an anonymous key-issuing protocol.

Lattice-based instantiation. We provide a concrete instantiation of a blind IBE with certified identities, based on the learning with errors (LWE) problem in the random oracle model. Our construction is based on the lattice-based IBE scheme by Gentry–Peikert–Vaikuntanathan (GPV-IBE) [18], which is arguably the most efficient IBE scheme based on standard lattice assumptions. The two main technical hurdles involved in our construction are as follows.

(a) *Realizing anonymous key issuing.* Unlike standard IBE, where the KGC knows (i.e., authorizes via some certification mechanism) which ID it is issuing secret keys to, IBE schemes that deal with the key escrow problem cannot allow the KGC to know the IDs corresponding to the secret keys. This is the main reason why the ICA was introduced: it authorizes users by providing them with certificates that do not leak their ID to the KGC, which they can then use to obtain secret keys from the KGC. The main problem here is thus figuring out what the certificate should look like and how to combine it with GPV-IBE's key-generation process.

Our main observation is that the secret keys of GPV-IBE are only a short vector over $\mathbb{Z}^m$, and the key-generation process of GPV-IBE can be viewed as a signing algorithm through the Naor transformation [6]. Concretely, a secret key for ID, which is a short vector over $\mathbb{Z}^m$, can be seen as a signature for some message (related to ID) over $\mathbb{Z}_q^n$. At a high level, the user in our construction will end up receiving two certificates: one from the ICA and another from the KGC, and then the user will combine them to form a secret key for the GPV-IBE. However, the two certificates must be related to one specific ID in some meaningful way, or otherwise, the user can simply mix-and-match different certificates together. To this end, we build on the lattice-based blind signature scheme proposed by Rückert [31] and use it in a way so that the KGC can blindly sign to a vector over $\mathbb{Z}_q^n$ which implicitly commits to an ID. We note that Rückert [32] later mentions that the blind signature scheme in [31] is vulnerable in some use cases; however, the way we use it avoids this problem.

(b) *Satisfying IND-ANON-KGC security.* Informally, IND-ANON-KGC security stipulates that even if the KGC receives polynomially many ciphertexts for the same ID, as long as the ID is sampled from a sufficiently high min-entropy source, then it cannot tell whether the ciphertexts are real or sampled uniformly at random from the ciphertext space. In other words, even though the KGC can construct secret keys for any ID, it should not be able to identify the right ID corresponding to the ciphertexts with more than negligible probability. Below we recall the ciphertext of GPV-IBE:

It may be possible to use the threshold variant of the GPV IBE scheme by Bendlin et al. [4] to obtain a lattice-based IBE scheme secure against the key escrow problem. We leave this as a potential future direction.

$c_0 = \mathbf{u}_{\mathsf{ID}}^{\top}\mathbf{s} + x + \mathsf{M}\lfloor q/2 \rceil$ and $\mathbf{c}_1 = \mathbf{A}^{\top}\mathbf{s} + \mathbf{x}$, where $\mathsf{M} \in \{0, 1\}$ is the plaintext, $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ is included in mpk, $\mathbf{u}_{\mathsf{ID}} = \mathsf{H}(\mathsf{ID}) \in \mathbb{Z}_q^n$ is a hash value (derived from the random oracle), $\mathbf{s} \in \mathbb{Z}_q^n$ is a uniformly random vector over $\mathbb{Z}_q^n$, and $\mathbf{x}, x$ are small "noise" terms.

At first glance, IND-ANON-KGC security seems highly related to the notion of multi-challenge IND-CPA security of IBE [21], where an adversary can obtain polynomially many challenge ciphertexts. Therefore, it may seem that the security proof of IND-ANON-KGC follows from a simple hybrid argument since so does standard multi-challenge security. However, this intuition is inaccurate. The key difference between the two security notions is that in IND-ANON-KGC the KGC holds the master secret key to the IBE scheme, i.e., the trapdoor for the lattice generated by $\mathbf{A}$. This prevents us from using a hybrid argument. Moreover, since the adversary for IND-ANON-KGC (which is the KGC) has the power to fully recover the randomness $\mathbf{s}$ from $\mathbf{c}_1$ in the ciphertext, this prevents us from using an entropy-based argument on the vector $\mathbf{s}$ to argue uniformness of c_0, as was done in previous proofs for GPV-IBE in the multi-challenge setting [24]. We, therefore, depart from previous proof techniques for GPV-IBE to prove IND-ANON-KGC of our proposed IBE scheme. We take advantage of the fact that the adversary does not know the ID corresponding to the challenge ciphertext. Note that to the contrary, in the multi-challenge setting, the adversary was able to specify the ID that is being encrypted. In our security proof, we use the fact that $\mathbf{u}_{\mathsf{ID}} = \mathsf{H}(\mathsf{ID}) \in \mathbb{Z}_q^n$ is distributed as a uniformly random vector from the view of the adversary in the random oracle model and view $\mathbf{u}_{\mathsf{ID}}$ as the LWE secret, rather than the encryption randomness $\mathbf{s}$ as in previous proofs.

Pairing-based instantiation. We also provide an instantiation of a blind IBE scheme with certified identities, based on the DBDH problem in the random oracle model. The high-level approach is similar to our lattice-based scheme: we combine an anonymous IBE scheme with a blind signature scheme where the blind signatures can be viewed as secret keys of the anonymous IBE scheme. As for the building blocks, we use the anonymous IBE scheme of Boneh-Franklin (BF-IBE) [6] and the blind signature of Boldyreva [5]. Since Boldyreva's blind signature scheme is based on the Boneh-Lynn-Shacham (BLS) signature scheme [7], the blind signature can be reused as a secret key of the BF-IBE scheme.

We note that Chow [12] constructs a blind IBE scheme with certified identities with a different design approach (albeit without a formal security proof as mentioned above). Unlike our construction using a specific blind signature scheme, he uses a commitment scheme with some additional property and combines it with the anonymous IBE scheme of Gentry [17]. In the same paper, he also briefly mentions the possibility of realizing a blind IBE scheme with certified identities based on the more efficient BF-IBE by combining it with Boldyreva's blind signature scheme. However, details on how to combine them are not provided.

2 Preliminaries

Notations. For a distribution or random variable X we write $x \leftarrow X$ to denote the operation of sampling a random x according to X. For a set S, we write $s \leftarrow S$ as a shorthand for $s \leftarrow U(S)$. For a vector $\mathbf{v} \in \mathbb{R}^n$, denote $\|\mathbf{v}\|$ as the standard Euclidean norm. For a matrix $\mathbf{R} \in \mathbb{R}^{n \times n}$, denote $\|\mathbf{R}\|$ as the length of the longest column and $\|\mathbf{R}\|_{\mathrm{GS}}$ as the longest column of the Gram-Schmidt orthogonalization of $\mathbf{R}$. We denote $s_1(\mathbf{R})$ as the largest singular value of $\mathbf{R}$. Finally, denote $\langle A, B \rangle$ as an interactive protocol between two PPT algorithms A and B.

2.1 Background on Lattices

Lattices and Gaussian measures. A (full-rank-integer) m-dimensional lattice Λ in $\mathbb{Z}^m$ is a set of the form $\{\sum_{i\in[m]} x_i\mathbf{b}_i | x_i \in \mathbb{Z}\}$, where $\mathbf{B} = \{\mathbf{b}_1, \cdots, \mathbf{b}_m\}$ are m linearly independent vectors in $\mathbb{Z}^m$. We call $\mathbf{B}$ the basis of the lattice Λ. For any positive integers n, m and $q \geq 2$, a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and a vector $\mathbf{u} \in \mathbb{Z}_q^n$, we define the lattices $\Lambda^{\perp}(\mathbf{A}) = \{\mathbf{z} \in \mathbb{Z}^m | \mathbf{A}\mathbf{z} = \mathbf{0} \mod q\}$ and $\Lambda^{\perp}_{\mathbf{u}}(\mathbf{A}) = \{\mathbf{z} \in \mathbb{Z}^m | \mathbf{A}\mathbf{z} = \mathbf{u} \mod q\}$.

Lemma 1 ([18]). Let n, m, q be positive integers such that $m \geq 2n\log q$. Let σ be any positive real such that $\sigma \geq \omega(\sqrt{\log n})$. Then for $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}$ and $\mathbf{e} \leftarrow D_{\mathbb{Z}^m,\sigma}$, the distribution of $\mathbf{u} = \mathbf{A}\mathbf{e} \mod q$ is statistically close to uniform over $\mathbb{Z}_q^n$.

Furthermore, fix $\mathbf{u} \in \mathbb{Z}_q^n$. Then the conditional distribution of $\mathbf{e} \leftarrow D_{\mathbb{Z}^m,\sigma}$ given $\mathbf{A}\mathbf{e} = \mathbf{u} \mod q$ for a uniformly random $\mathbf{A}$ in $\mathbb{Z}_q^{n\times m}$ is statistically close to $D_{\Lambda^{\perp}_{\mathbf{u}}(\mathbf{A}),\sigma}$.

Lemma 2 ([18]). Let n, m, q be positive integers with $m \geq 2n\log q$, $\sigma \geq \omega(\sqrt{\log m})$, and $\mathbf{u}$ be an arbitrary vector in $\mathbb{Z}_q^n$. Then, for all but a q^{-n} fraction of $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, if we sample a vector $\mathbf{x} \leftarrow D_{\Lambda^{\perp}_{\mathbf{u}}(\mathbf{A}),\sigma}$, we have $\Pr[\|\mathbf{x}\| > \sqrt{m}\sigma] < \mathsf{negl}(n)$.

Lemma 3 (Noise Rerandomization, [23]). Let q, ℓ, m be positive integers and r a positive real satisfying $r > \max\{\omega(\sqrt{\log m}), \omega(\sqrt{\log \ell})\}$. Let $\mathbf{b} \in \mathbb{Z}_q^m$ be arbitrary and $\mathbf{z}$ chosen from $D_{\mathbb{Z}^m,r}$. Then there exists a PPT algorithm ReRand such that for any $\mathbf{V} \in \mathbb{Z}^{m\times\ell}$ and positive real $\sigma > s_1(\mathbf{V})$, $\mathsf{ReRand}(\mathbf{V}, \mathbf{b}+\mathbf{z}, r, \sigma)$ outputs $\mathbf{b}'^{\top} = \mathbf{b}^{\top}\mathbf{V} + \mathbf{z}'^{\top} \in \mathbb{Z}_q^{\ell}$, where $\mathbf{z}'$ is distributed statistically close to $D_{\mathbb{Z}^{\ell},2r\sigma}$.

Sampling algorithms. The following lemma states useful algorithms for sampling short vectors from lattices.

Lemma 4. [([18, 26]] Let $n, m, q > 0$ be integers with $m \geq 2n\log q$.

- $\mathsf{TrapGen}(1^n, 1^m, q) \rightarrow (\mathbf{A}, \mathbf{T_A})$: There exists a randomized algorithm that outputs a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and a full-rank matrix $\mathbf{T_A} \in \mathbb{Z}^{m\times m}$, where $\mathbf{T_A}$ is a basis for $\Lambda^{\perp}(\mathbf{A})$, $\mathbf{A}$ is statistically close to uniform and $\|\mathbf{T_A}\|_{\mathsf{GS}} = O(\sqrt{n\log q})$.
- $\mathsf{SamplePre}(\mathbf{A}, \mathbf{u}, \mathbf{T_A}, \sigma) \rightarrow \mathbf{e}$: There exists a randomized algorithm that, given a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, a vector $\mathbf{u} \in \mathbb{Z}_q^n$, a basis $\mathbf{T_A}$ for $\Lambda^{\perp}(\mathbf{A})$, and a Gaussian parameter $\sigma > \|\mathbf{T_A}\|_{\mathsf{GS}} \cdot \omega(\sqrt{\log m})$, outputs a vector $\mathbf{e} \in \mathbb{Z}^m$ sampled from a distribution which is $\mathsf{negl}(n)$-close to $D_{\Lambda^{\perp}_{\mathbf{u}}(\mathbf{A}),\sigma}$.

Hardness assumption. We define the Learning with Errors (LWE) problem introduced by Regev [30].

Definition 1 (Learning with Errors). For integers $n = n(\lambda)$, $m = m(\lambda)$, $q = q(n) > 2$, an error distribution $\chi = \chi(n)$ over $\mathbb{Z}^m$, and a PPT algorithm $\mathcal{A}$, the advantage for the *learning with errors problem* $\mathsf{LWE}_{n,m,q,\chi}$ of $\mathcal{A}$ is defined as $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{LWE}_{n,m,q,\chi}} = \Big|\Pr\big[\mathcal{A}(\mathbf{A}, \mathbf{A}^{\top}\mathbf{s}+\mathbf{z}) = 1\big] - \Pr\big[\mathcal{A}(\mathbf{A}, \mathbf{w}+\mathbf{z}) = 1\big]\Big|$ where $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}$, $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, $\mathbf{w} \leftarrow \mathbb{Z}_q^m$, and $\mathbf{z} \leftarrow \chi$. We say that the LWE assumption holds if $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{LWE}_{n,m,q,\chi}}$ is negligible for all PPT algorithm $\mathcal{A}$.

We note adding the noise term $\mathbf{z} \leftarrow \chi$ to the uniform element in $\mathbf{w} \leftarrow \mathbb{Z}_q^m$ is done intentionally. This is only a syntactical change to make the proof using Lemma 3 during the security proof easier as done in prior works [23, 24].

For prime q and $\alpha \in (0,1)$, the (decisional) $\mathsf{LWE}_{n,m,q,D_{\mathbb{Z},\alpha q}}$ for $\alpha q > 2\sqrt{n}$ has been shown by Regev [30] via a quantum reduction to be as hard as approximating the worst-case SIVP and GapSVP problems to within $\tilde{O}(n/\alpha)$ factors in the ℓ_2-norm in the worst case. In the subsequent works, (partial) dequantumization of the reduction were achieved [27, 9].

2.2 Background on Pairings

Bilinear groups. We define bilinear groups as follows.

Definition 2 (Bilinear Groups). Let p be a λ-bit prime, $\mathbb{G}$ and $\mathbb{G}_T$ be groups of order p, $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a bilinear map (pairing), and g be a generator of $\mathbb{G}$. We require bilinearlity: for all $g_1, g_2 \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$, $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$, and non-degeneracy: $e(g, g) \neq 1$ hold. We say that $(\mathbb{G}, \mathbb{G}_T, e, p, g)$ is a bilinear group.

Hardness assumption. We define the Decision Bilinear Diffie-Hellman (DBDH) problem as follows.

Definition 3 (DBDH). For a PPT algorithm $\mathcal{A}$, the advantage of the DBDH problem of $\mathcal{A}$ is defined as $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{DBDH}} = \left| \Pr\left[\mathcal{A}(1^\lambda, g, g^a, g^b, g^c, Z) = 1\right] - \Pr\left[\mathcal{A}(1^\lambda, g, g^a, g^b, g^c, e(g,g)^{abc}) = 1\right] \right|$ where $a, b, c \leftarrow \mathbb{Z}_p$ and $(\mathbb{G}, \mathbb{G}_T, e, p, g)$ is a bilinear group with a λ-bit prime p. We say that the DBDH assumption holds if $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{DBDH}}$ is negligible for all PPT algorithm $\mathcal{A}$.

2.3 General Primitives

Pseudorandom function. We provide the standard definition of pseudorandom function.

Definition 4 (Pseudorandom Function). A pseudorandom function is defined by a PPT algorithm $\mathsf{PRF} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$, where $\mathcal{K}$, $\mathcal{X}$, and $\mathcal{Y}$ are sets (implicitly) parameterized by the security parameter λ, and we further assume $\mathcal{K}$ is an efficiently sampleable set. We say the PRF is *pseudorandom* if the advantage

$$\mathsf{Adv}_{\mathsf{PRF},\mathcal{A}}(\lambda) = \left| \Pr[\mathcal{A}^{\mathsf{PRF}(\mathsf{s},\cdot)}(1^\lambda) \to 1] \ - \ \Pr[\mathcal{A}^{\mathcal{O}(\cdot)}(1^\lambda) \to 1] \right|$$

is negligible for any PPT adversary $\mathcal{A}$ where $\mathsf{s} \leftarrow \mathcal{K}$. Here, $\mathcal{O} : \mathcal{X} \to \mathcal{Y}$ is a random function that returns uniformly random elements over $\mathcal{Y}$.

In the random oracle model, a hash function can be used as a PRF. In case we use multiple independent PRFs in a scheme, we can append the index of the PRF with the actual input and feed it to the hash function.

Digital signature scheme. We define a *deterministic* digital signature scheme where the randomness used to sign a message is derived deterministically from the signing key and the message. Using PRFs, any digital signature scheme can be derandomized.

Definition 5 (Digital Signatures). A digital signature scheme Π_{Sig} with message space $\{0,1\}^\ell$ is a triple of polynomial time algorithms ($\mathsf{Sig.KeyGen}$, $\mathsf{Sig.Sign}$, $\mathsf{Sig.Verify}$) of the following form:

$\mathsf{Sig.KeyGen}(1^\lambda) \to (\mathsf{vk}_{\mathsf{Sig}}, \mathsf{sk}_{\mathsf{Sig}})$: The key generation algorithm takes as input the security parameter 1^λ and outputs a verification key $\mathsf{vk}_{\mathsf{Sig}}$ and signing key $\mathsf{sk}_{\mathsf{Sig}}$.

$\mathsf{Sig.Sign}(\mathsf{sk}_{\mathsf{Sig}}, x) \to \sigma_{\mathsf{Sig}}$: The (deterministic) signing algorithm takes as inputs the signing key $\mathsf{sk}_{\mathsf{Sig}}$ and message $x \in \{0,1\}^\ell$, and outputs a signature σ_{Sig}.

$\mathsf{Sig.Verify}(\mathsf{vk}_{\mathsf{Sig}}, x, \sigma_{\mathsf{Sig}}) \to \top$ **or** $\perp$ **:** The verification algorithm takes as inputs the verification key $\mathsf{vk}_{\mathsf{Sig}}$, message $x \in \{0,1\}^{\ell}$ and signature σ_{Sig}, and outputs $\top$ if the signature is valid and outputs $\perp$ otherwise.

Correctness. We say a digital signature scheme is correct if for all λ, $\ell \in \mathsf{poly}(\lambda)$, messages $x \in \{0,1\}^{\ell}$, $(\mathsf{vk}_{\mathsf{Sig}}, \mathsf{sk}_{\mathsf{Sig}}) \in \mathsf{Sig.KeyGen}(1^{\lambda})$, $\sigma_{\mathsf{Sig}} \in \mathsf{Sig.Sign}(\mathsf{sk}_{\mathsf{Sig}}, x)$, we have $\mathsf{Sig.Verify}(\mathsf{vk}_{\mathsf{Sig}}, x, \sigma_{\mathsf{Sig}}) = \top$.

Eucma security. The security notion of existential unforgeability under an adaptive chosen message attack (eu-cma) is defined by the following game between an adversary $\mathcal{A}$ and a challenger.

- Setup: The challenger runs $(\mathsf{vk}_{\mathsf{Sig}}, \mathsf{sk}_{\mathsf{Sig}}) \leftarrow \mathsf{Sig.KeyGen}(1^{\lambda})$ and provides $\mathcal{A}$ the verification key $\mathsf{vk}_{\mathsf{Sig}}$.

- Signature Queries: When $\mathcal{A}$ submits a message $x \in \{0,1\}^{\ell}$, the challenger responds by returning $\sigma_{\mathsf{Sig}} \leftarrow \mathsf{Sig.Sign}(\mathsf{sk}_{\mathsf{Sig}}, x)$.

- Forgery: Finally, $\mathcal{A}$ outputs a pair $(x^*, \sigma^*_{\mathsf{Sig}})$. The adversary $\mathcal{A}$ wins if $\mathsf{Sig.Verify}(\mathsf{vk}_{\mathsf{Sig}}, x^*, \sigma^*_{\mathsf{Sig}}) = \top$ and x^* was not submitted by $\mathcal{A}$ as a signature query.

We say Π_{Sig} is eu-cma secure if the advantage $\mathsf{Adv}^{\mathsf{eu\text{-}cma}}_{\mathsf{Sig},\mathcal{A}}(\lambda) = \Pr[\mathcal{A} \text{ wins}]$ is negligible for any PPT $\mathcal{A}$.

3 Blind Identity-Based Encryption with Certified Identities

In this section, we present a new and secure IBE formalization that resolves the key escrow problem. As mentioned in the introduction, we refer to this primitive as *blind IBE with certified identities*, since the secret key-generation process can be seen as blind IBE [10, 20] with an ICA to certify users' identities. For simplicity, we occasionally call it "IBE" for simplicity.

In our scheme, users first authenticate themselves with the ICA to obtain certificates, which they then use to run an interactive protocol with the KGC and construct secret keys $\mathsf{sk}_{\mathsf{ID}}$ for use as in standard IBE. Here, the KGC never knows which user ID it is interacting with, and in particular, this implies that it does not know the $\mathsf{sk}_{\mathsf{ID}}$. We assume that users communicate with the ICA and the KGC via secure channels. Note that we use the same encryption and decryption algorithms as in standard IBE.

Definition 6 (Blind IBE with Certified Identities). A blind IBE scheme with certified identities Π_{IBE} consists of the following PPT algorithms:

$\mathsf{Setup}(1^{\lambda}) \to \mathsf{params}$**:** The setup algorithm takes as input a security parameter 1^{λ}, and outputs a public parameter params. We assume the identity space $\mathcal{ID}$ and the message space $\mathcal{M}$ are defined by params. Moreover, we assume params are implicitly provided as input to all algorithms.

$\mathsf{KGC.KeyGen}(\mathsf{params}) \to (\mathsf{mpk}, \mathsf{msk})$**:** The setup algorithm run by KGC takes as input params, and outputs a master public key mpk and a master secret key msk.

$\mathsf{ICA.KeyGen}(\mathsf{params}) \to (\mathsf{vk}, \mathsf{ik})$**:** The key-generation algorithm run by ICA takes as input params, and outputs a certificate verification key vk and a certificate-issuing key ik.

$\mathsf{ICA.Cert}(\mathsf{vk}, \mathsf{ik}, \mathsf{ID}) \to (\mathsf{cert}, \mathsf{td})$**:** The certificate-issuing algorithm run by ICA takes as inputs a certificate verification key vk, certificate-issuing key ik and an identity $\mathsf{ID} \in \mathcal{ID}$, and outputs a certificate cert and a trapdoor information td.

$\mathsf{IBE.Enc}(\mathsf{mpk}, \mathsf{ID}, \mathsf{M}) \to \mathsf{ct}$**:** The encryption algorithm run by a user takes as inputs the master public key mpk, an identity $\mathsf{ID} \in \mathcal{ID}$ and a message $\mathsf{M} \in \mathcal{M}$, and outputs a ciphertext ct.

$\mathsf{IBE.Dec}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{ID}}, \mathsf{ct}) \to \mathsf{M}$ **or** $\perp$**:** The decryption algorithm run by a user takes as input the master public key mpk, a secret key $\mathsf{sk}_{\mathsf{ID}}$ and a ciphertext ct, and outputs M or $\perp$.

$\langle \mathsf{ObtainKey}(\mathsf{mpk}, \mathsf{ID}, \mathsf{cert}, \mathsf{td}), \mathsf{IssueKey}(\mathsf{mpk}, \mathsf{msk}, \mathsf{vk}) \rangle$**:** The interactive key-issuing protocol between a user and the KGC involves two interactive algorithms $\mathsf{ObtainKey}$ and $\mathsf{IssueKey}$. The user and the KGC interactively run the $\mathsf{ObtainKey}$ algorithm and the $\mathsf{IssueKey}$ algorithm, respectively, as follows.

User: The user takes as input (mpk, ID, cert, td) as specified by the input of $\mathsf{ObtainKey}$, and sends a first-round message $\mathsf{M}_{\mathsf{user}}$ to KGC.

KGC: The KGC takes as input (mpk, msk, vk) as specified by the input of $\mathsf{IssueKey}$ along with the message $\mathsf{M}_{\mathsf{user}}$ sent by the user, and returns a second-round message $\mathsf{M}_{\mathsf{KGC}}$ to the user.

User: On input the message $\mathsf{M}_{\mathsf{KGC}}$ from the KGC, the user (locally) outputs either $\mathsf{sk}_{\mathsf{ID}}$ or $\perp$.

We denote $(\mathsf{sk}, \epsilon) \leftarrow \langle \mathsf{ObtainKey}(\mathsf{mpk}, \mathsf{ID}, \mathsf{cert}, \mathsf{td}), \mathsf{IssueKey}(\mathsf{mpk}, \mathsf{msk}, \mathsf{vk}) \rangle$ to indicate that the final output obtained by the user and the KGC are the secret key $\mathsf{sk}_{\mathsf{ID}}$ and an empty string ϵ, respectively. Note that depending on what the KGC responds as the second message $\mathsf{M}_{\mathsf{KGC}}$, $\mathsf{sk}_{\mathsf{ID}}$ may be set to $\perp$. Furthermore, we call $(\mathsf{M}_{\mathsf{user}}, \mathsf{M}_{\mathsf{KGC}})$ as the *transcript* of the protocol.

Correctness. For all $\lambda \in \mathbb{N}$, all $\mathsf{ID} \in \mathcal{ID}$, and all $\mathsf{M} \in \mathcal{M}$, $\mathsf{IBE.Dec}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{ID}}, \mathsf{ct}) = \mathsf{M}$ holds with overwhelming probability where it is taken over the randomness used in running $\mathsf{params} \leftarrow \mathsf{Setup}(1^\lambda)$, $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{KGC.KeyGen}(\mathsf{params})$, $(\mathsf{vk}, \mathsf{ik}) \leftarrow \mathsf{ICA.KeyGen}(\mathsf{params})$, $(\mathsf{cert}, \mathsf{td}) \leftarrow \mathsf{ICA.Cert}(\mathsf{vk}, \mathsf{ik}, \mathsf{ID})$, $(\mathsf{sk}_{\mathsf{ID}}, \epsilon) \leftarrow \langle \mathsf{ObtainKey}(\mathsf{mpk}, \mathsf{ID}, \mathsf{cert}, \mathsf{td}), \mathsf{IssueKey}(\mathsf{mpk}, \mathsf{msk}, \mathsf{vk}) \rangle$, and $\mathsf{ct} \leftarrow \mathsf{IBE.Enc}(\mathsf{mpk}, \mathsf{ID}, \mathsf{M})$.

Remark 1 (On the round complexity of key issuing). The above definition only considers a two-move key-issuing protocol. One can easily generalize the definition to a multi-move protocol, however, we restricted it to a two-move protocol for simplicity. Indeed, the instantiation we provide in Sections 4 and 5 will be two-move.

Security against users. As in standard IBE, we consider the notion of security against corrupted users and define indistinguishability against chosen plaintext attacks. We call this IND-ANON-CPA, to explicitly indicate that it implies anonymity. Broadly speaking, this differs from other similar definitions or standard IBE in that an adversary $\mathcal{A}$ can access the certifying oracle, that will output certificates for any ID (except the challenge identity ID^*), and can also supply the obtained certificates to the key-generation oracle. Note that we do not consider an adversary $\mathcal{A}$ that can obtain a certificate for ID^*, since this will allow $\mathcal{A}$ to trivially break security. This corresponds to the assumption that, in practice, an adversary cannot obtain a certificate for the challenge identity ID^*.

Definition 7 (IND-ANON-CPA). We define IND-ANON-CPA security by the following game between a challenger and a PPT adversary $\mathcal{A}$. Below, let CTSamp be a sampling algorithm that takes a master public key as input and outputs an element in the ciphertext space.

- Setup. At the outset of the game, the challenger runs $\mathsf{params} \leftarrow \mathsf{Setup}(1^\lambda)$, $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{KGC.KeyGen}(\mathsf{params})$, $(\mathsf{vk}, \mathsf{ik}) \leftarrow \mathsf{ICA.KeyGen}(\mathsf{params})$, and initializes an empty list $\mathsf{IDList} := \emptyset$. The challenger further picks a random coin $\mathsf{coin} \leftarrow \{0, 1\}$ and keeps it secret. The challenger gives $(\mathsf{params}, \mathsf{mpk}, \mathsf{vk})$ to $\mathcal{A}$. After this, $\mathcal{A}$ can adaptively make the following three types of queries to the challenger in arbitrary order: certificate, secret key, and challenge queries. $\mathcal{A}$ can query the first two arbitrarily polynomially many times and the third only once.

  Certificate Query: If $\mathcal{A}$ submits $\mathsf{ID} \in \mathcal{ID}$ to the challenger, the challenger computes $(\mathsf{cert}, \mathsf{td}) \leftarrow \mathsf{ICA.Cert}(\mathsf{vk}, \mathsf{ik}, \mathsf{ID})$ and returns $(\mathsf{cert}, \mathsf{td})$ to $\mathcal{A}$. It then stores ID to IDList.

  Secret Key Query: If $\mathcal{A}$ submits a first-round message $\mathsf{M}_{\mathsf{user}}$ to the challenger, the challenger runs the $\mathsf{IssueKey}$ algorithm taking as inputs (mpk, msk, vk) and the message $\mathsf{M}_{\mathsf{user}}$, and obtains a second-round message $\mathsf{M}_{\mathsf{KGC}}$. It then returns $\mathsf{M}_{\mathsf{KGC}}$ to $\mathcal{A}$.

  Challenge Query: If $\mathcal{A}$ submits $(\mathsf{ID}^*, \mathsf{M}^*)$ to the challenger where $\mathsf{ID}^* \in \mathcal{ID}$, $\mathsf{ID}^* \notin \mathsf{IDList}$, and $\mathsf{M}^* \in \mathcal{M}$, the challenger proceeds as follows: If $\mathsf{coin} = 0$, the challenger returns $\mathsf{ct}^* \leftarrow \mathsf{IBE.Enc}(\mathsf{mpk}, \mathsf{ID}^*, \mathsf{M}^*)$. Otherwise, if $\mathsf{coin} = 1$, the challenger returns $\mathsf{ct}^* \leftarrow \mathsf{CTSamp}(\mathsf{mpk})$.

- Guess. $\mathcal{A}$ outputs a guess $\widehat{\mathsf{coin}} \in \{0, 1\}$ for coin. We say that Π_{IBE} is IND-ANON-CPA secure if the advantage

$$\mathsf{Adv}^{\text{IND-ANON-CPA}}_{\mathsf{IBE},\mathcal{A}}(\lambda) = \left| \Pr[\mathsf{coin} = \widehat{\mathsf{coin}}] - 1/2 \right|$$

  is negligible for any PPT adversary $\mathcal{A}$.

Security against the KGC. We also consider the notion of security against the honest-but-curious KGC, which follows the protocol but attempts to obtain information about the underlying plaintexts from the observed ciphertexts. This is a more stringent and practical security notion than the corresponding notion informally stated in [12]. A more detailed explanation on the difference between prior works is provided in Remark 2. In brief, our definition guarantees that if the KGC runs, i.e., generates secret keys as specified, it cannot obtain any information about the corresponding identities or plaintexts from ciphertexts, even if it uses knowledge obtained via the key-issuing protocol.

At the start of the security game, the adversary $\mathcal{A}$ is given the master secret key msk along with all public information $(\mathsf{mpk}, \mathsf{params}, \mathsf{vk})$. In addition, $\mathcal{A}$ is allowed to access two oracles, namely, the key-generation and encryption oracles. First, $\mathcal{A}$ obtains the secret key $\mathsf{sk}_{\mathsf{ID}}$ for a randomly chosen identity ID from the key-generation oracle. This captures the scenario where an unknown user ID generates their secret key $\mathsf{sk}_{\mathsf{ID}}$ via executing $\langle \mathsf{ObtainKey}, \mathsf{IssueKey} \rangle$ with the KGC. The identities sent to the key-generation oracle are stored in an identity list IDList. In addition, for any plaintext M and any ID in IDList, $\mathcal{A}$ can ask for a ciphertext ct from the encryption oracle. This captures the scenario where the KGC can observe ciphertexts for all users to whom it has issued secret keys. In the challenge phase, $\mathcal{A}$ specifies the challenge identity ID^* from IDList (and submits an arbitrary message M^*) to obtain the challenge ciphertext ct^*. Note that $\mathcal{A}$ does not specify ID nor ID^* itself, but simply specifies the indices in IDList. It is clear that if ciphertexts reveal any information about the corresponding identities, $\mathcal{A}$ could easily win the game by creating $\mathsf{sk}_{\mathsf{ID}^*}$. In particular, our definition captures both indistinguishability and anonymity.

Definition 8 (IND-ANON-KGC). We define IND-ANON-KGC security by the following game between a challenger and a PPT adversary $\mathcal{A}$. Below, let CTSamp be a sampling algorithm that takes a master public key as input and outputs an element in the ciphertext space.[4]

- Setup. At the outset of the game, the challenger runs $\mathsf{params} \leftarrow \mathsf{Setup}(1^\lambda)$, $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{KGC.KeyGen}(\mathsf{params})$, $(\mathsf{vk}, \mathsf{ik}) \leftarrow \mathsf{ICA.KeyGen}(\mathsf{params})$ and initializes an empty set $\mathsf{IDList} := \emptyset$ and a counter $Q_{\mathsf{key}} := 0$. The challenger further picks a random coin $\mathsf{coin} \leftarrow \{0,1\}$ and keeps it secret. The challenger gives $(\mathsf{params}, \mathsf{mpk}, \mathsf{msk}, \mathsf{vk})$ to $\mathcal{A}$. After this, $\mathcal{A}$ can adaptively make the following three types of queries to the challenger in an arbitrary order: encryption, issue key, and challenge queries. $\mathcal{A}$ can query the first two arbitrarily polynomial many times and the third only once.

  Encryption Query: If $\mathcal{A}$ submits an index i and a message $\mathsf{M} \in \mathcal{M}$ to the challenger, the challenger first checks if $i \in [Q_{\mathsf{key}}]$ where $[0]$ is defined as the empty set. If not, the challenger forces $\mathcal{A}$ to output a random coin $\widehat{\mathsf{coin}}$ in $\{0,1\}$. Otherwise, the challenger retrieves the i-th entry ID of IDList and returns $\mathsf{ct} \leftarrow \mathsf{IBE.Enc}(\mathsf{mpk}, \mathsf{ID}, \mathsf{M})$.

  IssueKey Query: If $\mathcal{A}$ makes an IssueKey query, the challenger first randomly samples $\mathsf{ID} \leftarrow \mathcal{ID}$ and computes $(\mathsf{cert}, \mathsf{td}) \leftarrow \mathsf{ICA.Cert}(\mathsf{vk}, \mathsf{ik}, \mathsf{ID})$. It then runs $\mathsf{ObtainKey}$ on inputs (mpk, ID, cert, td) to generate the first-round message $\mathsf{M}_{\mathsf{user}}$ and returns $\mathsf{M}_{\mathsf{user}}$ to $\mathcal{A}$. Finally, the challenger stores ID to IDList and updates $Q_{\mathsf{key}} \leftarrow Q_{\mathsf{key}} + 1$.

  Challenge Query: If $\mathcal{A}$ submits (M^*, i^*) to the challenger where $\mathsf{M}^* \in \mathcal{M}$, the challenger first checks if $i^* \in [Q_{\mathsf{key}}]$. If not, the challenger forces $\mathcal{A}$ to output a random coin $\widehat{\mathsf{coin}}$ in $\{0,1\}$. Otherwise, the challenger proceeds as follows: The challenger first retrieves the i^*-th entry ID^* of IDList. Then, if $\mathsf{coin} = 0$, the challenger returns $\mathsf{ct}^* \leftarrow \mathsf{IBE.Enc}$ $(\mathsf{mpk}, \mathsf{ID}^*, \mathsf{M}^*)$. Otherwise, if $\mathsf{coin} = 1$, the challenger returns $\mathsf{ct}^* \leftarrow \mathsf{CTSamp}(\mathsf{mpk})$.

- Guess. $\mathcal{A}$ outputs a guess $\widehat{\mathsf{coin}} \in \{0,1\}$ for coin. We say that Π_{IBE} is IND-ANON-KGC secure if the advantage

$$\mathsf{Adv}^{\text{IND-ANON-KGC}}_{\mathsf{IBE},\mathcal{A}}(\lambda) = \left|\Pr[\mathsf{coin} = \widehat{\mathsf{coin}}] - 1/2\right|$$

  is negligible for any PPT adversary $\mathcal{A}$.

Remark 2 (Differences from the existing definition). As described above, our idea is based on Chow's work [12]. However, Chow's notion of security against the KGC (ACI-KGC) is defined for standard IBE under the assumption that the KGC does not know the identities used in the system. However, this assumption is generally invalid since, in practice, the KGC manages identity lists in order to identify users before generating their keys.[5] By contrast, IND-ANON-KGC is defined for a version of IBE where the KGC generates secret keys without having access to such an identity list, meaning that the key-generation process does not violate the real-world usage.

Security against the ICA. Unlike Chow's work [12] that only considered a fully trusted ICA, we aim to define security against a potentially malicious ICA. However, such a definition is difficult. A malicious ICA can generate certificates for any identity ID and thereby obtain the corresponding secret keys by impersonating the user and interacting with the KGC. Therefore, in principle, we cannot allow the ICA to have arbitrary access to the key-generation oracle (i.e., interacting with the

[4]We note that the sampling algorithm CTSamp does not necessarily have to be identical to the one we defined in the IND-ANON-CPA security. This is true for the subsequent IND-ANON-ICA security.

[5]In the same paper [12], Chow acknowledged this issue and introduced an anonymous key-issuing protocol to compensate for the gap between the assumption and reality, however, he did not give any formal definition of ACI-KGC for IBE where secret keys are generated with this protocol.

KGC). Given this, we model the malicious ICA to have the capability of generating a potentially malicious key pair $(\mathsf{vk}, \mathsf{ik})$[6] while disallowing it to have access to the key-generation oracle. Unlike Chow's definition, our definition prevents to construct a trivial IBE scheme secure against the KGC mentioned in the introduction; the ICA plays the role of the KGC in a standard IBE scheme and the KGC does nothing.

Definition 9 (IND-ANON-ICA). We define IND-ANON-ICA security by the following game between a challenger and a PPT adversary $\mathcal{A}$. Below, let CTSamp be a sampling algorithm that takes a master public key as input and outputs an element in the ciphertext space.

- Setup. At the outset of the game, the challenger runs $\mathsf{params} \leftarrow \mathsf{Setup}(1^\lambda)$ and $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{KGC.KeyGen}(\mathsf{params})$. The challenger picks a random coin $\mathsf{coin} \leftarrow \{0,1\}$ and keeps it secret. The challenger gives $(\mathsf{params}, \mathsf{mpk})$ to $\mathcal{A}$. Then, $\mathcal{A}$ can make the following challenge query once.

  Challenge Query: If $\mathcal{A}$ submits $(\mathsf{ID}^*, \mathsf{M}^*)$ to the challenger where $\mathsf{ID}^* \in \mathcal{ID}$ and $\mathsf{M}^* \in \mathcal{M}$, the challenger proceeds as follows: If $\mathsf{coin} = 0$, the challenger returns $\mathsf{ct}^* \leftarrow \mathsf{IBE.Enc}(\mathsf{mpk}, \mathsf{ID}^*, \mathsf{M}^*)$. Otherwise, if $\mathsf{coin} = 1$, the challenger returns $\mathsf{ct}^* \leftarrow \mathsf{CTSamp}(\mathsf{mpk})$.

- Guess. $\mathcal{A}$ outputs a guess $\widehat{\mathsf{coin}} \in \{0,1\}$ for coin. We say that Π_{IBE} is IND-ANON-ICA secure if the advantage

$$\mathsf{Adv}^{\text{IND-ANON-ICA}}_{\mathsf{IBE},\mathcal{A}}(\lambda) = \left|\Pr[\mathsf{coin} = \widehat{\mathsf{coin}}] - 1/2\right|$$

  is negligible for any PPT adversary $\mathcal{A}$.

Remark 3. One can consider a stronger definition than what we define above; the malicious ICA is allowed to obtain secret keys for any ID $(\neq \mathsf{ID}^*)$ during the game. The reason why we do not define this stronger notion is that, compared to our weaker definition, it seems to only capture some additional unnatural scenarios. In practice, if the ICA can impersonate any user ID $(\neq \mathsf{ID}^*)$ and interact with the KGC, it is only fair to assume that it is also able to impersonate ID^*, and hence, obtain a secret key for ID^* by interacting with the KGC. Nonetheless, we like to point out that our construction appearing in the next sections can in fact be proven to satisfy such a stronger definition.

4 Lattice-based Construction

4.1 Proposed IBE scheme from Lattices

In this section, we present our lattice-based scheme. This combines the GPV-IBE scheme [18] with Rückert's full-domain-hash style lattice-based blind signature scheme [31]. Although Rückert [32] later found out [31] that his signature scheme is vulnerable (to an attack we will explain later), fortunately, this vulnerability does not affect our scheme. Informally, this is because we can guarantee that the KGC only issues a secret key when it sees a *valid* certificate (presumably signed by the ICA).

Construction. Let the identity space $\mathcal{ID}$ of the IBE scheme Π_{IBE} be $\mathcal{ID} = \{0,1\}^*$. In practice, by using collision resistant hash functions, we can set $\mathcal{ID} = \{0,1\}^\ell$ for $\ell = O(\lambda)$. Here, we occasionally treat elements in $\mathbb{Z}_q^n$ as binary strings over $\{0,1\}^{n \log q}$ through some fixed canonical

[6]Looking ahead, this will be implicit in our definition since the $\mathsf{IBE.Enc}$ algorithm is independent of $(\mathsf{vk}, \mathsf{ik})$.

embedding. Let $\mathsf{PRF} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be any pseudorandom function with appropriate domain $\mathcal{X}$ and range $\mathcal{Y}$. I.e., let $\mathcal{X}$ include $\mathcal{ID}$ and the set of all the first-round messages $\mathsf{M}_{\mathsf{user}}$, and let range $\mathcal{Y}$ include an appropriate length of randomness used by algorithms $\mathsf{ICA.Cert}$ and $\mathsf{IssueKey}$. Finally, let Π_{Sig} : ($\mathsf{Sig.KeyGen}$, $\mathsf{Sig.Sign}$, $\mathsf{Sig.Verify}$) be a deterministic digital signature scheme with message space $\{0,1\}^{n \log q}$ where the randomness used to sign a message is derived deterministically from the signing key and the message. Using PRFs, any digital signature scheme can be derandomized. We assume that Π_{Sig} provides the standard security notion of existential unforgeability under an adaptive chosen message attack (eu-cma).

$\mathsf{Setup}(1^\lambda)$: Choose positive integers n, m and prime q, and output $\mathsf{params} = (1^\lambda, 1^n, 1^m, q, \alpha', \sigma, \mathsf{H})$, where $\mathsf{H} : \{0,1\}^* \to \mathbb{Z}_q^n$ is a hash function modeled as a random oracle.

$\mathsf{KGC.KeyGen}(\mathsf{params})$: Run $(\mathbf{A}, \mathbf{T}_{\mathbf{A}}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$ and sample a PRF key $\mathsf{s}_{\mathsf{KGC}} \leftarrow \mathcal{K}$. Then, output a master pubic key $\mathsf{mpk} = \mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and a master secret key $\mathsf{msk} = (\mathbf{T}_{\mathbf{A}}, \mathsf{s}_{\mathsf{KGC}})$.

$\mathsf{ICA.KeyGen}(\mathsf{params})$: Run $(\mathsf{vk}_{\mathsf{Sig}}, \mathsf{sk}_{\mathsf{Sig}}) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$ and sample a PRF key $\mathsf{s}_{\mathsf{ICA}} \leftarrow \mathcal{K}$. Then, output a certificate verification key $\mathsf{vk} = \mathsf{vk}_{\mathsf{Sig}}$ and a certificate issuing key $\mathsf{ik} = (\mathsf{sk}_{\mathsf{Sig}}, \mathsf{s}_{\mathsf{ICA}})$.

$\mathsf{ICA.Cert}(\mathsf{vk}, \mathsf{ik}, \mathsf{ID})$: Parse $\mathsf{ik} = (\mathsf{sk}_{\mathsf{Sig}}, \mathsf{s}_{\mathsf{ICA}})$ and compute $\mathbf{u}_{\mathsf{ID}} = \mathsf{H}(\mathsf{ID})$. Then, sample a short vector $\mathbf{y}_{\mathsf{ID},1} \leftarrow \{0,1\}^m$ and compute $\mathbf{u}_{\mathsf{ID},1} = \mathbf{A}\mathbf{y}_{\mathsf{ID},1}$. Furthermore, compute $\mathbf{u}_{\mathsf{ID},2} = \mathbf{u}_{\mathsf{ID}} - \mathbf{u}_{\mathsf{ID},1} \in \mathbb{Z}_q^n$ and $\sigma_{\mathsf{Sig}} \leftarrow \mathsf{Sig.Sign}(\mathsf{sk}_{\mathsf{Sig}}, \mathbf{u}_{\mathsf{ID},2})$. Finally, output a certificate $\mathsf{cert} = (\mathbf{u}_{\mathsf{ID},2}, \sigma_{\mathsf{Sig}})$ and trapdoor information $\mathsf{td} = \mathbf{y}_{\mathsf{ID},1}$. Here, we assume all the randomness used in this algorithm is derived from $r_{\mathsf{ID}} \leftarrow \mathsf{PRF}(\mathsf{s}_{\mathsf{ICA}}, \mathsf{ID})$.

$\mathsf{IBE.Enc}(\mathsf{mpk}, \mathsf{ID}, \mathsf{M})$: Compute $\mathbf{u}_{\mathsf{ID}} = \mathsf{H}(\mathsf{ID})$. To encrypt a message $\mathsf{M} \in \{0,1\}$, sample $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, $\mathbf{x} \leftarrow D_{\mathbb{Z}^m, \alpha' q}$, and $x \leftarrow D_{\mathbb{Z}, \alpha' q}$, and compute $c_0 = \mathbf{u}_{\mathsf{ID}}^\top \mathbf{s} + x + \mathsf{M}\lfloor q/2 \rceil$ and $\mathbf{c}_1 = \mathbf{A}^\top \mathbf{s} + \mathbf{x}$. Finally, output a ciphertext $\mathsf{ct} = (c_0, \mathbf{c}_1)$.

$\mathsf{IBE.Dec}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{ID}}, \mathsf{ct})$: Parse $\mathsf{sk}_{\mathsf{ID}} = \mathbf{e}_{\mathsf{ID}}$ and $\mathsf{ct} = (c_0, \mathbf{c}_1)$. Compute $w = c_0 - \mathbf{e}_{\mathsf{ID}}^\top \mathbf{c}_1$. Output 0 if w is closer to 0 than to $\lfloor q/2 \rfloor$ modulo q, and 1, otherwise.

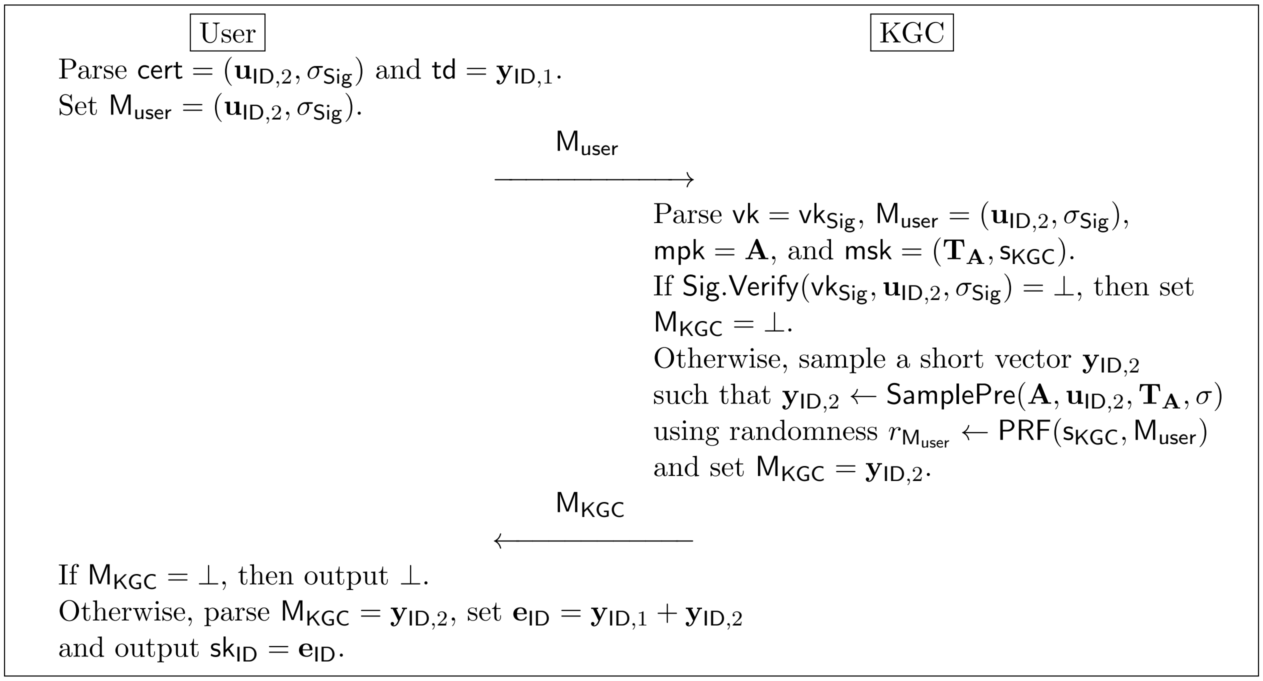
$\langle \mathsf{ObtainKey}(\mathsf{mpk}, \mathsf{ID}, \mathsf{cert}, \mathsf{td}), \mathsf{IssueKey}(\mathsf{mpk}, \mathsf{msk}, \mathsf{vk}) \rangle$: The user and the KGC interactively runs $\mathsf{ObtainKey}$ and $\mathsf{IssueKey}$, respectively.

User: On input (mpk, ID, $\mathsf{cert}, \mathsf{td}$), set the first-round message $\mathsf{M}_{\mathsf{user}} = \mathsf{cert}$ and send $\mathsf{M}_{\mathsf{user}}$ to the KGC. Here, $\mathsf{cert} = (\mathbf{u}_{\mathsf{ID},2}, \sigma_{\mathsf{Sig}})$.

KGC: On input (mpk, msk, vk) and the first-round message $\mathsf{M}_{\mathsf{user}}$, parse $\mathsf{vk} = \mathsf{vk}_{\mathsf{Sig}}$ and $\mathsf{M}_{\mathsf{user}} = (\mathbf{u}_{\mathsf{ID},2}, \sigma_{\mathsf{Sig}})$. If $\mathsf{Sig.Verify}(\mathsf{vk}_{\mathsf{Sig}}, \mathbf{u}_{\mathsf{ID},2}, \sigma_{\mathsf{Sig}}) = \perp$, then set $\mathsf{M}_{\mathsf{KGC}} = \perp$ and send $\mathsf{M}_{\mathsf{KGC}}$ to the user. Otherwise, parse $\mathsf{mpk} = \mathbf{A}$ and $\mathsf{msk} = (\mathbf{T}_{\mathbf{A}}, \mathsf{s}_{\mathsf{KGC}})$. Then, sample a short vector $\mathbf{y}_{\mathsf{ID},2} \leftarrow \mathsf{SamplePre}(\mathbf{A}, \mathbf{u}_{\mathsf{ID},2}, \mathbf{T}_{\mathbf{A}}, \sigma)$, set $\mathsf{M}_{\mathsf{KGC}} = \mathbf{y}_{\mathsf{ID},2}$, and send $\mathsf{M}_{\mathsf{KGC}}$ to the user. Here, we assume all the randomness used in this algorithm is derived from $r_{\mathsf{M}_{\mathsf{user}}} \leftarrow \mathsf{PRF}(\mathsf{s}_{\mathsf{KGC}}, \mathsf{M}_{\mathsf{user}})$.

User: If $\mathsf{M}_{\mathsf{KGC}} = \perp$, then output $\perp$. Otherwise, parse $\mathsf{td} = \mathbf{y}_{\mathsf{ID},1}$ and $\mathsf{M}_{\mathsf{KGC}} = \mathbf{y}_{\mathsf{ID},2}$, set $\mathbf{e}_{\mathsf{ID}} = \mathbf{y}_{\mathsf{ID},1} + \mathbf{y}_{\mathsf{ID},2}$ and (locally) output the secret key $\mathsf{sk}_{\mathsf{ID}} = \mathbf{e}_{\mathsf{ID}}$.

Remark 4 (Generating randomness via PRFs). Here, we generate the randomness used by the $\mathsf{ICA.Cert}$ and $\mathsf{IssueKey}$ algorithms via PRFs. This has the effect of only allowing the adversary to obtain one valid certificate cert per identity ID and one valid second-round message $\mathsf{M}_{\mathsf{KGC}}$ per first-round message $\mathsf{M}_{\mathsf{user}}(= \mathsf{cert})$. We require this condition during the security proof for reasons similar to those for other lattice-based IBE schemes [18, 2, 11].



$\boxed{\text{User}}$ $\boxed{\text{KGC}}$

Parse $\mathsf{cert} = (\mathbf{u}_{\mathsf{ID},2}, \sigma_{\mathsf{Sig}})$ and $\mathsf{td} = \mathbf{y}_{\mathsf{ID},1}$.
Set $\mathsf{M}_{\mathsf{user}} = (\mathbf{u}_{\mathsf{ID},2}, \sigma_{\mathsf{Sig}})$.

$\xrightarrow{\mathsf{M}_{\mathsf{user}}}$

Parse $\mathsf{vk} = \mathsf{vk}_{\mathsf{Sig}}$, $\mathsf{M}_{\mathsf{user}} = (\mathbf{u}_{\mathsf{ID},2}, \sigma_{\mathsf{Sig}})$,
$\mathsf{mpk} = \mathbf{A}$, and $\mathsf{msk} = (\mathbf{T}_{\mathbf{A}}, \mathsf{s}_{\mathsf{KGC}})$.
If $\mathsf{Sig.Verify}(\mathsf{vk}_{\mathsf{Sig}}, \mathbf{u}_{\mathsf{ID},2}, \sigma_{\mathsf{Sig}}) = \perp$, then set
$\mathsf{M}_{\mathsf{KGC}} = \perp$.
Otherwise, sample a short vector $\mathbf{y}_{\mathsf{ID},2}$
such that $\mathbf{y}_{\mathsf{ID},2} \leftarrow \mathsf{SamplePre}(\mathbf{A}, \mathbf{u}_{\mathsf{ID},2}, \mathbf{T}_{\mathbf{A}}, \sigma)$
using randomness $r_{\mathsf{M}_{\mathsf{user}}} \leftarrow \mathsf{PRF}(\mathsf{s}_{\mathsf{KGC}}, \mathsf{M}_{\mathsf{user}})$
and set $\mathsf{M}_{\mathsf{KGC}} = \mathbf{y}_{\mathsf{ID},2}$.

$\xleftarrow{\mathsf{M}_{\mathsf{KGC}}}$

If $\mathsf{M}_{\mathsf{KGC}} = \perp$, then output $\perp$.
Otherwise, parse $\mathsf{M}_{\mathsf{KGC}} = \mathbf{y}_{\mathsf{ID},2}$, set $\mathbf{e}_{\mathsf{ID}} = \mathbf{y}_{\mathsf{ID},1} + \mathbf{y}_{\mathsf{ID},2}$
and output $\mathsf{sk}_{\mathsf{ID}} = \mathbf{e}_{\mathsf{ID}}$.

Figure 1: Flow of the Key-issuing Protocol (Lattice-based)

Remark 5 (Role of certificates). If the validity of cert is not checked, then users can obtain information about the master secret key as follows. First, the user samples $\mathbf{y}_1$ from $\mathbb{Z}_q^m$, computes $\mathbf{u}' = \mathbf{A}\mathbf{y}_1$, and then sends $\mathbf{u}'$ directly to the KGC, which returns $\mathbf{e}_{\mathsf{ID}}$ such that $\mathbf{A}\mathbf{e}_{\mathsf{ID}} = \mathbf{u}'$. If we let $\mathbf{e} = \mathbf{y}_1 - \mathbf{e}_{\mathsf{ID}}$, then $\mathbf{A}\mathbf{e} = \mathbf{A}(\mathbf{y}_1 - \mathbf{e}_{\mathsf{ID}}) = \mathbf{0}$. This means that the user has obtained an $\mathbf{e}$ satisfying $\mathbf{A}\mathbf{e} = \mathbf{0}$. If enough users collude together, then we can recover a trapdoor $\mathbf{T}_{\mathbf{A}}$ for $\mathbf{A}$ such that $\mathbf{A}\mathbf{T}_{\mathbf{A}} = \mathbf{0}$. Thus, for the security proof, we must require that users cannot obtain such an $\mathbf{e}$. This attack has been identified by Rückert [32] as an issue in constructing a full-domain-hash style lattice-based blind signature scheme. In our scheme, users have no choice but to use the $\mathbf{u}'_{\mathsf{ID}}$ issued by the ICA unless they can forge cert, and this issue does not appear.

Correctness. The following lemma states the correctness of our blind IBE scheme with certified identity.

Lemma 5 (Correctness). Suppose the parameters q, σ and α' satisfy $\sigma > \omega(\sqrt{n \log m \log q})$ and $\alpha' < 1/(8\sigma\sqrt{m})$. Then our scheme is correct with overwhelming probability.

Proof. If the key issuing protocol between the user and the KGC is run correctly, then any user ID will obtain a secret key $\mathsf{sk}_{\mathsf{ID}} = \mathbf{e}_{\mathsf{ID}}$ such that $\mathbf{A}\mathbf{e}_{\mathsf{ID}} = \mathbf{A}(\mathbf{y}_{\mathsf{ID},1} + \mathbf{y}_{\mathsf{ID},2}) = \mathbf{u}_{\mathsf{ID},1} + \mathbf{u}_{\mathsf{ID},2} = \mathbf{u}_{\mathsf{ID}}$. Let $\mathsf{ct} \leftarrow \mathsf{IBE.Enc}(\mathsf{mpk}, \mathsf{ID}, \mathsf{M})$. Then when we run $\mathsf{IBE.Dec}$ with $\mathsf{sk}_{\mathsf{ID}}$, we obtain $w = c_0 - \mathbf{e}_{\mathsf{ID}}^\top \mathbf{c}_1 = \mathsf{M}\lfloor q/2 \rceil + x + (\mathbf{y}_{\mathsf{ID},1} + \mathbf{y}_{\mathsf{ID},2})^\top \mathbf{x}$. By Lemma 4, we have that $\mathbf{y}_{\mathsf{ID},2}$ is distributed negligibly close to $D_{\Lambda^{\perp}_{\mathbf{u}_{\mathsf{ID},2}}(\mathbf{A}),\sigma}$. Then, by Lemma 2, we have $\|\mathbf{y}_{\mathsf{ID},2}\| \le \sigma\sqrt{m}$ with overwhelming probability. Since, $x \leftarrow D_{\mathbb{Z},\alpha' q}$ and $\mathbf{x} \leftarrow D_{\mathbb{Z}^m,\alpha' q}$ and $\mathbf{y}_{\mathsf{ID},1} \in \{0,1\}^m$ the error term w can be bounded by $\|x + (\mathbf{y}_{\mathsf{ID},1} + \mathbf{y}_{\mathsf{ID},2})^\top \mathbf{x}\| \le 2\alpha' q \sigma \sqrt{m}$, where we used the subgaussian property to make a finer analysis of the noise bound on $\mathbf{y}_{\mathsf{ID},1}^\top \mathbf{x}$ (See for example [18, 28]). Hence, for the error term to have absolute value less than $q/4$, it suffices to choose the parameters as in the statement. $\square$

Parameter Selection. For the system to satisfy correctness and make the security proof work, we need the following restrictions. Note that we will prove the security of the scheme under the LWE assumption whose noise rate is α, which is lower than α' that is used in the encryption algorithm.

- The error term is less than $q/4$ (i.e., $\alpha' < 1/8\sigma\sqrt{m}$ by Lemma 5)
- $\mathsf{TrapGen}$ operates properly (i.e., $m > 3n \log q$ by Lemma 4)
- Samplable from $D_{\Lambda^{\perp}_{\mathbf{u}}(\mathbf{A}),\sigma}$ (i.e., $\sigma > \|\mathbf{T_A}\|_{\text{GS}} \cdot \omega(\sqrt{\log m}) = O(\sqrt{n \log m \log q})$ by Lemma 4),
- σ is sufficiently large so that we can apply Lemma 1 (i.e., $\sigma > \omega(\log n)$),
- We can apply Lemma 3 (i.e., $\alpha'/2\alpha > \sqrt{(\sigma+1)^2 m + 1}$),
- $\mathsf{LWE}_{n,m,q,D_{\mathbb{Z},\alpha q}}$ is hard (i.e., $\alpha q > 2\sqrt{n}$ for prime q).

To satisfy these requirements, for example, we can set the parameters $m, q, \sigma, \alpha, \alpha'$ as: $m = n^{1+\kappa}$, $q = n^{2+3.5\kappa}$, $\sigma = n^{0.5+\kappa}$, $\alpha' q = n^{1.5+2\kappa}$, and $\alpha q = 2 \cdot n^{0.5}$, where $\kappa > 0$ is a constant that can be set arbitrarily small. In the above, we round up m to the nearest integer and q to the nearest largest prime.

Multi-bit Encryption. Although the above scheme only provides 1-bit encryption, we can extend it to provide k-bit encryption without incurring a factor k blow up of the ciphertext size. We change the range of the hash function as $\mathsf{H} : \{0,1\}^* \to \mathbb{Z}_q^{n \times k}$. Then, a plaintext vector $\mathsf{M} \in \mathbb{Z}_q^k$ is encrypted as $\mathbf{c}_0 = \mathbf{u}_{\mathsf{ID}}^{\top}\mathbf{s} + \mathbf{x}' + \mathsf{M}\lfloor q/2 \rceil \in \mathbb{Z}_q^k$ where $\mathbf{s} \leftarrow \mathbb{Z}_q^{n \times k}$ and $\mathbf{x}' \leftarrow D_{\mathbb{Z}^k,\alpha' q}$ (where $\mathbf{c}_1$ is the same). Note that the secret key is now required to be a matrix $\mathbf{e}_{\mathsf{ID}} \in \mathbb{Z}^{m \times k}$.

4.2 Security Analysis

Theorem 1. Our blind IBE scheme with certified identity Π_{IBE} is IND-ANON-CPA secure in the random oracle model if the underlying signature scheme Π_{Sig} is eu-cma secure, the PRF is pseudorandom, and assuming the hardness of $\mathsf{LWE}_{n,m,q,D_{\mathbb{Z},\alpha q}}$. Alternatively, we can get rid of the second requirement by replacing the PRF by the random oracle.

Proof Overview. The high level structure of the proof follows the original GPV-IBE security proof. That is, for a random oracle query ID, the simulator first samples $\mathbf{e}_{\mathsf{ID}}$ from $D_{\mathbb{Z}^m,\sigma}$ and sets $\mathbf{u}_{\mathsf{ID}} = \mathbf{A}\mathbf{e}_{\mathsf{ID}}$, instead of sampling $\mathbf{e}_{\mathsf{ID}}$ from $D_{\Lambda^{\perp}_{\mathbf{u}_{\mathsf{ID}}}(\mathbf{A}),\sigma}$. Since our key-issuing protocol is 2-move, and $\mathbf{u}_{\mathsf{ID},2}$ contained in $\mathbf{u}_{\mathsf{ID},2}$ depends on the key issuing, we employ this idea twice: the simulator samples $\mathbf{y}_{\mathsf{ID},1}$ from $\{0,1\}^m$ and $\mathbf{y}_{\mathsf{ID},2}$ from $D_{\Lambda^{\perp}_{\mathbf{u}}(\mathbf{A}),\sigma}$, and sets $\mathbf{u}_{\mathsf{ID}} = \mathbf{A}(\mathbf{y}_{\mathsf{ID},1} + \mathbf{y}_{\mathsf{ID},2})$, and also sets $\mathbf{u}_{\mathsf{ID},2} = \mathbf{u}_{\mathsf{ID}} - \mathbf{A}\mathbf{y}_{\mathsf{ID},1}$. We remark that the distribution of $\mathbf{A}\mathbf{y}_{\mathsf{ID},2}$ is statistically close to uniform over $\mathbb{Z}_q^n$ [18], and thus $\mathbf{u}_{\mathsf{ID}} \leftarrow \mathbb{Z}_q^n$ and $\mathbf{u}_{\mathsf{ID}} = \mathbf{A}(\mathbf{y}_{\mathsf{ID},1} + \mathbf{y}_{\mathsf{ID},2})$ are identical from $\mathcal{A}$'s view. We also remark that we adopt the proof strategy of Katsumata et al. [24]. In the original GPV-IBE scheme the so-called partitioning technique was used to prove security; the simulator divides the identity space into two in such a way that for one partition it can only construct secret keys and for the other it can only construct challenge ciphertexts. However, this proof strategy was notorious for having a loose reduction. Recently, Katsumata et al. [24] provided a much tighter reduction by following the proof technique of the Cramer-Shoup encryption scheme [13]. Since our proof follows the strategy of [24], it enjoys tight security as well.

Proof. Let $\mathsf{CTSamp}(\mathsf{mpk})$ be an algorithm that outputs a random element from $\mathbb{Z}_q \times \mathbb{Z}_q^m$ and $\mathcal{A}$ a PPT algorithm which breaks the IND-ANON-CPA security of our blind IBE scheme with certified identity. We make some assumptions on $\mathcal{A}$ to make our proof simpler without loss of generality. First, we assume that $\mathcal{A}$ never queries the random oracle on the same input. Next, we assume that whenever $\mathcal{A}$ queries for a certificate or a challenge ciphertext, the corresponding ID has already been queried to the random oracle H. In the following let X_i denote the event that $\mathcal{A}$ wins in Game_i. We modify the games so that in the final game, the adversary will have no winning advantage.

Game_0 : This is the original security game. At the beginning of the game the challenger prepares params, $(\mathsf{mpk}, \mathsf{msk})$, $(\mathsf{vk}, \mathsf{ik})$ as specified by the game and gives $(\mathsf{params}, \mathsf{mpk}, \mathsf{vk})$ to $\mathcal{A}$. The challenger also prepares three empty lists IDList, CList, and HList. Here, the lists CList and HList are absent in the security definition and only introduced to be used throughout this proof. Then, the challenger picks a random coin $\mathsf{coin} \leftarrow \{0, 1\}$ and answers to the queries made by the adversary $\mathcal{A}$ as follows:

- When $\mathcal{A}$ makes a random oracle query on ID, the challenger samples a random $\mathbf{u}_{\mathsf{ID}} \leftarrow \mathbb{Z}_q^n$ and updates $\mathsf{HList} \leftarrow \mathsf{HList} \cup \{(\mathsf{ID}, \mathbf{u}_{\mathsf{ID}}, \perp)\}$. Then, it returns $\mathbf{u}_{\mathsf{ID}}$ to $\mathcal{A}$.

- When $\mathcal{A}$ queries for a certificate corresponding to ID, the challenger runs $(\mathsf{cert}, \mathsf{td}) \leftarrow \mathsf{ICA.Cert}$ $(\mathsf{vk}, \mathsf{ik}, \mathsf{ID})$ and returns $(\mathsf{cert}, \mathsf{td})$ to $\mathcal{A}$. It further updates $\mathsf{IDList} \leftarrow \mathsf{IDList} \cup \{\mathsf{ID}\}$ and $\mathsf{CList} \leftarrow \mathsf{CList} \cup \{(\mathsf{cert}, \mathsf{td}, \mathsf{ID})\}$. Here, as in the real scheme, the randomness used to run $\mathsf{ICA.Cert}$ is generated by $r_{\mathsf{ID}} \leftarrow \mathsf{PRF}(\mathsf{s}_{\mathsf{ICA}}, \mathsf{ID})$, where the PRF key $\mathsf{s}_{\mathsf{ICA}}$ is included in the certificate issuing key ik.

- When $\mathcal{A}$ queries for a secret key with a first-round message $\mathsf{M}_{\mathsf{user}}$, the challenger parses $\mathsf{M}_{\mathsf{user}} = (\mathbf{u}_{\mathsf{ID},2}, \sigma_{\mathsf{Sig}})$ and returns the second-round message $\mathsf{M}_{\mathsf{KGC}} = \mathbf{y}_{\mathsf{ID},2}$ or $\perp$ to $\mathcal{A}$ depending on $\mathsf{M}_{\mathsf{user}}$. Here, the randomness used to run $\mathsf{IssueKey}$ is generated by $r_{\mathsf{M}_{\mathsf{user}}} \leftarrow \mathsf{PRF}(\mathsf{s}_{\mathsf{KGC}}, \mathsf{M}_{\mathsf{user}})$, where the PRF key $\mathsf{s}_{\mathsf{KGC}}$ is included in the master secret key msk.

- When $\mathcal{A}$ queries for a challenge ciphertext on ID^* and message M^*, the challenger returns $\mathsf{ct}^* \leftarrow \mathsf{IBE.Enc}(\mathsf{mpk}, \mathsf{ID}^*, \mathsf{M}^*)$ if $\mathsf{coin} = 0$ and $\mathsf{ct}^* \leftarrow \mathsf{CTSamp}(\mathsf{mpk})$ if $\mathsf{coin} = 1$.

At the end of the game, $\mathcal{A}$ outputs a guess $\widehat{\mathsf{coin}}$ for coin. Finally, the challenger outputs $\widehat{\mathsf{coin}}$. By definition, we have $\left|\Pr[X_0] - \frac{1}{2}\right| = \left|\Pr[\widehat{\mathsf{coin}} = \mathsf{coin}] - \frac{1}{2}\right| = \mathsf{Adv}_{\mathsf{IBE},\mathcal{A}}^{\text{IND-ANON-CPA}}(\lambda)$.

Game_1 : In this game, we change how the challenger generates the randomness used for $\mathsf{ICA.Cert}$ and $\mathsf{IssueKey}$. In the previous game, the challenger generated PRF keys $\mathsf{s}_{\mathsf{ICA}}$ and $\mathsf{s}_{\mathsf{KGC}}$ and created randomnesses r_{ID} and $r_{\mathsf{M}_{\mathsf{user}}}$ for inputs ID and $\mathsf{M}_{\mathsf{user}}$, respectively. In this game, the challenger uniformly randomly samples r_{ID} and $r_{\mathsf{M}_{\mathsf{user}}}$ from the appropriate domains for each inputs ID and $\mathsf{M}_{\mathsf{user}}$, and responds with the same randomness for same inputs. Due to the pseudorandomness of the underlying PRF, this makes negligible difference. Therefore, we have $|\Pr[X_0] - \Pr[X_1]| = \mathsf{negl}(\lambda)$. In the following games, we will no longer explicitly mention the used randomness for simplicity and assume $\mathsf{msk} = \mathbf{T}_{\mathbf{A}}$ and $\mathsf{ik} = \mathsf{sk}_{\mathsf{Sig}}$.

Game_2 : In this game, we change how the challenger answers the secret key queries. In particular, the challenger responds as follows for a secret key query:

- When $\mathcal{A}$ queries for a secret key with input a first-round message $\mathsf{M}_{\mathsf{user}} = (\mathbf{u}_{\mathsf{ID},2}, \sigma_{\mathsf{Sig}})$, the challenger first checks whether $\mathsf{Sig.Verify}(\mathsf{vk}_{\mathsf{Sig}}, \mathbf{u}_{\mathsf{ID},2}, \sigma_{\mathsf{Sig}}) = \top$. If not, it returns the second-round message $\mathsf{M}_{\mathsf{KGC}} = \perp$ to $\mathcal{A}$. Next, it sets $\mathsf{cert} = (\mathbf{u}_{\mathsf{ID},2}, \sigma_{\mathsf{Sig}})$ and checks whether $(\mathsf{cert}, \star, \star) \in \mathsf{CList}$, where $\star$ represents an arbitrary value. If not, the challenger aborts the game and forces $\mathcal{A}$ to output a random coin. Otherwise, the challenger proceeds as in the previous game.

As we show in Lemma 6, we have $|\Pr[X_1] - \Pr[X_2]| = \mathsf{negl}(\lambda)$ by the $\mathsf{eu\text{-}cma}$ security of the underlying signatures scheme. We postpone the proof to the end of the game sequence so as not to interrupt the proof.

Game_3 : In this game, we change how the random oracle queries are answered. In particular, the challenger responds as follows for a random oracle query:

- When $\mathcal{A}$ makes a random oracle query on ID, the challenger first samples $\mathbf{y}_{\mathsf{ID},1} \leftarrow \{0,1\}^m$ and $\mathbf{y}_{\mathsf{ID},2} \leftarrow D_{\mathbb{Z}^m,\alpha' q}$. Then, it sets $\mathbf{u}_{\mathsf{ID}} = \mathbf{A}(\mathbf{y}_{\mathsf{ID},1} + \mathbf{y}_{\mathsf{ID},2}) \in \mathbb{Z}_q^n$ and updates $\mathsf{HList} \leftarrow \mathsf{HList} \cup \{(\mathsf{ID}, \mathbf{u}_{\mathsf{ID}}, (\mathbf{y}_{\mathsf{ID},1}, \mathbf{y}_{\mathsf{ID},2}))\}$. Finally, it returns $\mathbf{u}_{\mathsf{ID}}$ to $\mathcal{A}$.

Here, the challenger responds to the other queries exactly as in the previous game. In particular, $\mathbf{y}_{\mathsf{ID},1}, \mathbf{y}_{\mathsf{ID},2}$ are *not* used anywhere and the only difference between the previous game is how $\mathbf{u}_{\mathsf{ID}}$ is created. Then, due to Lemma 1, the distribution of $\mathbf{u}_{\mathsf{ID}}$ in Game_3 is statistically close to that of Game_2. Therefore, we have $|\Pr[X_2] - \Pr[X_3]| = \mathsf{negl}(\lambda)$.

Game_4 : In this game, we change how the challenger answers the certificate queries. In particular, the challenger responds as follows for a certificate query:

- When $\mathcal{A}$ queries for a certificate corresponding to ID, the challenger retrieves the unique entry $(\mathsf{ID}, \mathbf{u}_{\mathsf{ID}}, (\mathbf{y}_{\mathsf{ID},1}, \mathbf{y}_{\mathsf{ID},2})) \in \mathsf{HList}$, which is guaranteed by the assumption we made on $\mathcal{A}$. Then, it sets $\mathbf{u}_{\mathsf{ID},1} = \mathbf{A}\mathbf{y}_{\mathsf{ID},1}$ and computes $\mathbf{u}_{\mathsf{ID},2} = \mathbf{u}_{\mathsf{ID}} - \mathbf{u}_{\mathsf{ID},1}$. The challenger further runs $\sigma_{\mathsf{Sig}} \leftarrow \mathsf{Sig.Sign}(\mathsf{sk}_{\mathsf{Sig}}, \mathbf{u}_{\mathsf{ID},2})$. Finally, it returns $(\mathsf{cert}, \mathsf{td}) = ((\mathbf{u}_{\mathsf{ID},2}, \sigma_{\mathsf{Sig}}), \mathbf{y}_{\mathsf{ID},1})$ and updates $\mathsf{CList} \leftarrow \mathsf{CList} \cup \{(\mathsf{cert}, \mathsf{td}, \mathsf{ID})\}$.

The only difference from the previous game is how $(\mathsf{cert}, \mathsf{td})$ is generated. Here, note that $\mathsf{cert} = (\mathbf{u}_{\mathsf{ID},2}, \sigma_{\mathsf{Sig}})$ is uniquely determined once $\mathbf{u}_{\mathsf{ID}}$ and $\mathsf{td} = \mathbf{y}_{\mathsf{ID},1}$ is fixed since we use deterministic signature schemes. Therefore, for any fixed ID, we consider the random variables $\mathbf{y}^{(3)}_{\mathsf{ID},1}$ and $\mathbf{y}^{(4)}_{\mathsf{ID},1}$ who are distributed according to the distribution of td conditioned on $\mathsf{H}(\mathsf{ID}) = \mathbf{u}_{\mathsf{ID}}$ in Game_3 and Game_4, respectively. It is easy to see that $\mathbf{y}^{(3)}_{\mathsf{ID},1}$ is distributed uniformly random over $\{0,1\}^m$ when the challenger runs $(\mathsf{cert}, \mathsf{td} = \mathbf{y}^{(3)}_{\mathsf{ID},1}) \leftarrow \mathsf{ICA.Cert}(\mathsf{vk}, \mathsf{ik}, \mathsf{ID})$, since $\mathbf{y}^{(3)}_{\mathsf{ID},1}$ is chosen independently from $\mathbf{u}_{\mathsf{ID}}$. Next we see how $\mathbf{y}^{(4)}_{\mathsf{ID},1}$ is distributed. Recall that owing to the change we made in Game_3, we have $\mathbf{u}_{\mathsf{ID}} = \mathbf{A}\mathbf{y}^{(4)}_{\mathsf{ID},1} + \mathbf{A}\mathbf{y}^{(4)}_{\mathsf{ID},2}$. Therefore, due to Lemma 1 and the fact that $\mathbf{y}^{(4)}_{\mathsf{ID},2}$ is information theoretically hidden from $\mathcal{A}$, $\mathbf{u}_{\mathsf{ID}}$ and $\mathbf{A}\mathbf{y}^{(4)}_{\mathsf{ID},1}$ are independently distributed according to the uniform distribution over $\mathbb{Z}_q^n$ with all but negligible probability. In other words, $\mathbf{u}_{\mathsf{ID}}$ is distributed independently from $\mathbf{y}^{(4)}_{\mathsf{ID},1}$ with all but negligible probability. Therefore, from the view of the adversary $\mathcal{A}$, $\mathbf{y}^{(4)}_{\mathsf{ID},1}$ is distributed negligibly close to a uniformly distribution over $\{0,1\}^m$. Hence, we have $|\Pr[X_3] - \Pr[X_4]| = \mathsf{negl}(\lambda)$.

Game_5 : In this game, we change how the challenger answers the secret key queries. In particular, the challenger responds as follows for a secret key query:

- When $\mathcal{A}$ queries for a secret key with input a first-round message $\mathsf{M}_{\mathsf{user}} = (\mathbf{u}_{\mathsf{ID},2}, \sigma_{\mathsf{Sig}})$, the challenger runs identically as in Game_4 up to the checking of $(\mathsf{cert}, \star, \star) \in \mathsf{CList}$. In case $(\mathsf{cert}, \star, \star) \notin \mathsf{CList}$, the challenger aborts as in the previous game. Otherwise, if $(\mathsf{cert}, \mathsf{td}, \mathsf{ID}) \in \mathsf{CList}$ for some td and ID, the challenger retrieves the unique entry $(\mathsf{ID}, \mathbf{u}_{\mathsf{ID}}, (\mathbf{y}_{\mathsf{ID},1}, \mathbf{y}_{\mathsf{ID},2})) \in \mathsf{HList}$, which is guaranteed from the way the challenger answers certificate queries. Finally, the challenge sets $\mathsf{M}_{\mathsf{KGC}} = \mathbf{y}_{\mathsf{ID},2}$ and returns the second-round message $\mathsf{M}_{\mathsf{KGC}}$ to $\mathcal{A}$.

Note that in this game, the challenger does not run algorithm $\mathsf{SamplePre}$ to sample the vector $\mathbf{y}_{\mathsf{ID},2}$ anymore. Specifically, the challenger no longer requires $\mathsf{msk} = \mathbf{T}_{\mathbf{A}}$ to play the role of the KGC in this game. Recall, we got rid of the PRF key inside msk in Game_1. The only difference from the previous game is how the vector $\mathbf{y}_{\mathsf{ID},2}$ is generated. For any fixed ID, let $\mathbf{y}^{(4)}_{\mathsf{ID},2}$ and $\mathbf{y}^{(5)}_{\mathsf{ID},2}$

be random variables distributed according to the distribution of the vector $\mathbf{y}_{\mathsf{ID},2}$ conditioned on $\mathsf{H}(\mathsf{ID}) = \mathbf{u}_{\mathsf{ID}}$ and $(\mathsf{cert}, \mathsf{td}) = ((\mathbf{u}_{\mathsf{ID},2}, \sigma_{\mathsf{Sig}}), \mathbf{y}_{\mathsf{ID},1})$ in Game_4 and Game_5, respectively. Since in Game_4, the challenger used algorithm $\mathsf{SamplePre}$ to sample $\mathbf{y}^{(4)}_{\mathsf{ID},2}$, due to Lemma 4, $\mathbf{y}^{(4)}_{\mathsf{ID},2}$ is distributed negligibly close to $D_{\Lambda^{\perp}_{\mathbf{u}_{\mathsf{ID},2}}(\mathbf{A}),\sigma}$. On the other hand, due to Lemma 1, $\mathbf{y}^{(5)}_{\mathsf{ID},2}$ is distributed negligibly close to $D_{\Lambda^{\perp}_{\mathbf{u}_{\mathsf{ID},2}}(\mathbf{A}),\sigma}$. Therefore, we have $|\Pr[X_4] - \Pr[X_5]| = \mathsf{negl}(\lambda)$.

Game_6 : In this game, we change how $(\mathsf{mpk}, \mathsf{msk}) = (\mathbf{A}, \mathbf{T_A})$ are created. Namely, the challenger chooses $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}$ without generating the associated trapdoor $\mathbf{T_A}$. By Lemma 4, this makes negligible difference. Since the challenger can answer all secret key queries without msk due to the change we made in Game_5, the view of $\mathcal{A}$ is altered only negligibly. Therefore, we have $|\Pr[X_5] - \Pr[X_6]| = \mathsf{negl}(\lambda)$.

Game_7 : In this game, we change the way the challenge ciphertext is created when $\mathsf{coin} = 0$. Recall in the previous games, when $\mathsf{coin} = 0$, the challenge ciphertext was created as in the real scheme. In this game, to create the challenge ciphertext for identity ID^* and message bit M^*, the challenger first retrieves the unique tuple $(\mathsf{ID}^*, \mathbf{u}_{\mathsf{ID}^*}, (\mathbf{y}_{\mathsf{ID}^*,1}, \mathbf{y}_{\mathsf{ID}^*,2})) \in \mathsf{HList}$ (which is guaranteed to exist by the assumption we made on $\mathcal{A}$). It then sets $\mathbf{e}_{\mathsf{ID}^*} = \mathbf{y}_{\mathsf{ID}^*,1} + \mathbf{y}_{\mathsf{ID}^*,2}$. It further samples $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, $\bar{\mathbf{x}} \leftarrow D_{\mathbb{Z}^m,\alpha q}$ and computes $\mathbf{v} = \mathbf{A}^\top\mathbf{s} + \bar{\mathbf{x}}$. The challenger then runs the following algorithm from Lemma 3:

$$\mathsf{ReRand}([\mathbf{e}_{\mathsf{ID}^*}|\mathbf{I}_{m\times m}], \mathbf{v}, \alpha q, \frac{\alpha'}{2\alpha}) \to \mathbf{c}' \in \mathbb{Z}_q^{m+1},$$

where $\mathbf{I}_{m\times m}$ is an identity matrix in $\mathbb{Z}_q^{m\times m}$. Let $c'_0 \in \mathbb{Z}_q$ denote the first entry of $\mathbf{c}'$ and let $\mathbf{c}_1 \in \mathbb{Z}_q^m$ denote the remaining entries. Finally, the challenger outputs the challenge ciphertext as

$$\mathsf{ct} = (c_0 = c'_0 + \mathsf{M}\lfloor q/2\rceil, \mathbf{c}_1). \tag{1}$$

We show that the view of $\mathcal{A}$ is changed only negligibly. Let $\mathbf{V} = [\mathbf{e}_{\mathsf{ID}^*}|\mathbf{I}_{m\times m}]$, $\mathbf{b} = \mathbf{A}^\top\mathbf{s}$, and $\mathbf{z} = \bar{\mathbf{x}}$. Then by the noise rerandomization lemma (i.e., Lemma 3), we see that

$$\mathbf{c}' = \mathbf{V}^\top\mathbf{b} + \mathbf{z}' = \big(\mathbf{A}\cdot[\mathbf{e}_{\mathsf{ID}^*}|\mathbf{I}_{m\times m}]\big)^\top\mathbf{s} + \mathbf{z}' = [\mathbf{u}_{\mathsf{ID}^*}|\mathbf{A}]^\top\mathbf{s} + \mathbf{z}',$$

where the distribution of $\mathbf{z}'$ is negligibly close to $D_{\mathbb{Z}^{m+1},\alpha' q}$. Here, the last equality follows from $\mathbf{A}\mathbf{e}_{\mathsf{ID}^*} = \mathbf{A}\mathbf{y}_{\mathsf{ID}^*,1} + \mathbf{A}\mathbf{y}_{\mathsf{ID}^*,2} = \mathbf{u}_{\mathsf{ID},1} + \mathbf{u}_{\mathsf{ID},2} = \mathbf{u}_{\mathsf{ID}}$. Furthermore, we are able to apply Lemma 3 since we have the following for our parameter selection:

$$\alpha'/2\alpha > \sqrt{(\sigma\sqrt{m} + \sqrt{m})^2 + 1} \geq s_1([\mathbf{e}_{\mathsf{ID}^*}|\mathbf{I}_{m\times m}]).$$

Therefore, we have $|\Pr[X_6] - \Pr[X_7]| = \mathsf{negl}(\lambda)$.

Game_8 : In this game, we further change the way the challenge ciphertext is created when $\mathsf{coin} = 0$. To create the challenge ciphertext when $\mathsf{coin} = 0$, the challenger first picks $\mathbf{b} \leftarrow \mathbb{Z}_q^m$, $\bar{\mathbf{x}} \leftarrow D_{\mathbb{Z}^m,\alpha q}$ and sets $\mathbf{v} = \mathbf{b} + \bar{\mathbf{x}}$. It then runs algorithm ReRand as in the previous Game_7 and sets the challenge ciphertext as in Equation (1). As we show in Lemma 7, we have $|\Pr[X_7] - \Pr[X_8]| = \mathsf{negl}(\lambda)$ by assuming the hardness of the LWE problem. We postpone the proof to the end of the game sequence so as not to interrupt the proof.

Game_9 : In this game, we further change the way the challenge ciphertext is created. When $\mathsf{coin} = 0$, the challenger first picks $\mathbf{b} \leftarrow \mathbb{Z}_q^m$, $\mathbf{x}' \leftarrow D_{\mathbb{Z}^m,\alpha' q}$ and computes

$$\mathbf{c}' = [\mathbf{e}_{\mathsf{ID}^*}|\mathbf{I}_{m\times m}]^\top\mathbf{b} + \mathbf{x}'.$$

It then parses $\mathbf{c}'$ into c'_0 and $\mathbf{c}_1$ and sets the challenge ciphertext as in Equation (1). Following the same argument we made to move from Game_6 to Game_7, we have $|\Pr[X_8] - \Pr[X_9]| = \mathsf{negl}(\lambda)$.

Game_{10} : In this game, we change the way the challenge ciphertext is created once more. Regardless of the value of $\mathsf{coin} \in \{0, 1\}$, the challenger samples $\mathsf{ct}^* \leftarrow \mathsf{CTSamp}(\mathsf{mpk})$ and returns ct^* to $\mathcal{A}$. We show that this alters the view of $\mathcal{A}$ only negligibly. Noice that in the previous game when $\mathsf{coin} = 0$, the challenge ciphertext could be written as follows:

$$c_0 = \mathbf{e}_{\mathsf{ID}^*}^\top \mathbf{b} + x'_0 + \mathsf{M}\lfloor q/2 \rceil, \quad \mathbf{c}_1 = \mathbf{b} + \mathbf{x}'_0,$$

where x'_0 is the first entry of $\mathbf{x}'$ and $\mathbf{x}'_0$ is the remaining entries. It suffices to show that the joint distribution of $(\mathbf{b}, \mathbf{e}_{\mathsf{ID}^*}^\top \mathbf{b})$ is negligible close to the uniform distribution over $\mathbb{Z}_q^m \times \mathbb{Z}_q$, conditioned on $\mathbf{u}_{\mathsf{ID}^*}$. Since $\mathbf{e}_{\mathsf{ID}^*} = \mathbf{y}_{\mathsf{ID}^*,1} + \mathbf{y}_{\mathsf{ID}^*,2}$ and $\mathbf{y}_{\mathsf{ID}^*,1}, \mathbf{y}_{\mathsf{ID}^*,2}$ are independent, we have $\mathbf{H}_\infty(\mathbf{e}_{\mathsf{ID}^*}) \geq \mathbf{H}_\infty(\mathbf{y}_{\mathsf{ID}^*,1})$. Moreover, since $\mathcal{A}$ never makes a certificate query for ID^*, $\mathbf{y}_{\mathsf{ID}^*,1} \in \{0, 1\}^m$ is distributed uniformly random over $\{0, 1\}^m$ from the view of $\mathcal{A}$; $\mathbf{H}_\infty(\mathbf{y}_{\mathsf{ID}^*,1}) \geq m$. Hence, using the left over hash lemma, we conclude that $\mathbf{e}_{\mathsf{ID}^*}^\top \mathbf{b}$ is statistically close to uniform over $\mathbb{Z}_q$. Therefore, we have $|\Pr[X_9] - \Pr[X_{10}]| = \mathsf{negl}(\lambda)$.

In this game, the adversary has no winning advantage since the challenge ciphertext is distributed identically for both $\mathsf{coin} = 0$ and 1. Namely, we have $|\Pr[X_{10}] - 1/2| = 0$.

Combining everything together, we conclude

$$\mathsf{Adv}_{\mathsf{IBE},\mathcal{A}}^{\text{IND-ANON-CPA}}(\lambda) = |\Pr[X_0] - 1/2| = \mathsf{negl}(\lambda)$$

.

To finish the proof of Theorem 1, it remains to prove the following two lemmas 6 and 7.

Lemma 6. If the underlying signature scheme Π_{Sig} is $\mathsf{eu\text{-}cma}$ secure, then $|\Pr[X_1] - \Pr[X_2]| = \mathsf{negl}(\lambda)$.

Proof. We observe that Game_1 and Game_2 are the same unless the adversary $\mathcal{A}$ makes a secret key query with input a second-round message $\mathsf{M}_{\mathsf{user}} = (\mathbf{u}_{\mathsf{ID},2}, \sigma_{\mathsf{Sig}})$ such that $\mathsf{Sig.Verify}(\mathsf{vk}_{\mathsf{Sig}}, \mathbf{u}_{\mathsf{ID},2}, \sigma_{\mathsf{Sig}}) = \top$ and $(\mathsf{cert} = (\mathbf{u}_{\mathsf{ID},2}, \sigma_{\mathsf{Sig}}), \star, \star) \notin \mathsf{CList}$, We denote this event F and define ϵ as the probability that F occurs in Game_1. Since $|\Pr[X_1] - \Pr[X_2]| \leq \Pr[\mathsf{F}]$, it suffices to show ϵ is negligible. To show this, we prove that there exists an adversary $\mathcal{B}$ that has advantage ϵ in breaking the $\mathsf{eu\text{-}cma}$ security game of Π_{Sig}. We give the description of $\mathcal{B}$ in the following.

At the outset of the $\mathsf{eu\text{-}cma}$ game, $\mathcal{B}$ is provided the verification key $\mathsf{vk}_{\mathsf{Sig}}$ by the Π_{Sig}-challenger and $\mathcal{B}$ sets the certificate verification key as $\mathsf{vk} = \mathsf{vk}_{\mathsf{Sig}}$. $\mathcal{B}$ further prepares params, $(\mathsf{mpk}, \mathsf{msk})$ as the Game_1-challenger and provides $\mathcal{A}$ with $(\mathsf{params}, \mathsf{mpk}, \mathsf{vk})$. $\mathcal{B}$ answers the random oracle and challenge ciphertext queries as the Game_1-challenger, This can be done since $\mathcal{B}$ does not require the signing key $\mathsf{sk}_{\mathsf{Sig}}$ corresponding to the verification key $\mathsf{vk}_{\mathsf{Sig}}$ to answer any of these queries. When $\mathcal{A}$ queries for a certificate corresponding to ID, $\mathcal{B}$ queries its Π_{Sig}-challenger on message $\mathbf{u}_{\mathsf{ID},2}$ and obtains σ_{Sig}. It then sets $\mathsf{cert} = (\mathbf{u}_{\mathsf{ID},2}, \sigma_{\mathsf{Sig}})$ and $\mathsf{td} = \mathbf{y}_{\mathsf{ID},1}$ and returns $(\mathsf{cert}, \mathsf{td})$ to $\mathcal{A}$. Moreover, when $\mathcal{A}$ queries for a secret key with a first-round message $\mathsf{M}_{\mathsf{user}} = (\mathbf{u}_{\mathsf{ID},2}, \sigma_{\mathsf{Sig}})$, if F does not occur, then it proceeds as in the Game_1-challenger (which can be done without knowledge of $\mathsf{sk}_{\mathsf{Sig}}$). Otherwise, if F occurs, $\mathcal{B}$ outputs $(\mathbf{u}_{\mathsf{ID},2}, \sigma_{\mathsf{Sig}})$ as its forgery to the Π_{Sig}-challenger and terminates. In case $\mathcal{A}$ terminates without triggering F, then $\mathcal{B}$ aborts the $\mathsf{eu\text{-}cma}$ game.

It is easy to check that unless F occurs, $\mathcal{B}$ completely simulates the view of Game_1 to $\mathcal{A}$. Furthermore, when F occurs, by the definition of F, we have $\mathsf{Sig.Verify}(\mathsf{vk}_{\mathsf{Sig}}, \mathbf{u}_{\mathsf{ID},2}, \sigma_{\mathsf{Sig}}) = \top$ and $(\mathsf{cert} = (\mathbf{u}_{\mathsf{ID},2}, \sigma_{\mathsf{Sig}}), \star, \star) \notin \mathsf{CList}$. In particular, since all the signature queries made by $\mathcal{B}$ are stored in CList by the way $\mathcal{B}$ answers $\mathcal{A}$'s certificate queries, this means that $\mathcal{B}$ has never queried a signature query on message $\mathbf{u}_{\mathsf{ID},2}$. Therefore, $(\mathbf{u}_{\mathsf{ID},2}, \sigma_{\mathsf{Sig}})$ is a valid forgery. This completes the proof of Lemma 6. □

Lemma 7. If the $\mathsf{LWE}_{n,q,D_{\mathbb{Z},\alpha q}}$ assumption holds, then $|\Pr[X_7] - \Pr[X_8]| = \mathsf{negl}(\lambda)$.

Proof. To prove the lemma, we use $\mathcal{A}$ to construct an LWE adversary $\mathcal{B}$ as follows:

At the beginning of the game, $\mathcal{B}$ samples m times the LWE oracle and forms the LWE instance $(\mathbf{A}, \mathbf{v} = \mathbf{b} + \bar{\mathbf{x}}) \in \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^m$, where $\bar{\mathbf{x}} \leftarrow D_{\mathbb{Z}^m,\alpha q}$. The task of $\mathcal{B}$ is to distinguish whether $\mathbf{b} = \mathbf{A}^\top \mathbf{s}$ for some $\mathbf{s} \leftarrow \mathbb{Z}_q^n$ or $\mathbf{b} \leftarrow \mathbb{Z}_q^m$. $\mathcal{B}$ sets the master public key as $\mathsf{mpk} = \mathbf{A}$, prepares params and $(\mathsf{vk}, \mathsf{ik})$, and samples a random coin $\mathsf{coin} \leftarrow \{0, 1\}$ as in the Game_7-challenger. Here, we assume that $\mathcal{B}$ is given an LWE instance which is consistent with the params (that is, n, q) output by algorithm Setup. Note that due to the modification we made in Game_6, $\mathcal{B}$ does not require $\mathbf{T_A}$ to answer any of $\mathcal{A}$'s queries. To generate the challenge ciphertext, if $\mathsf{coin} = 0$, it generates the challenge ciphertext as in Equation (1). If $\mathsf{coin} = 1$, $\mathcal{B}$ returns a random ciphertext using $\mathsf{CTSamp}(\mathsf{msk})$. At the end of the game, $\mathcal{A}$ outputs $\widehat{\mathsf{coin}}$. Finally, $\mathcal{B}$ outputs 1 if $\mathsf{coin} = \widehat{\mathsf{coin}}$ and 0 otherwise. It can be seen that if $\mathbf{A}$ and $\mathbf{v}$ are valid LWE instances (i.e., $\mathbf{b} = \mathbf{A}^\top \mathbf{s}$), then the view of $\mathcal{A}$ corresponds to Game_7. Otherwise (i.e., $\mathbf{b} \leftarrow \mathbb{Z}_q^m$), it corresponds to Game_8. We therefore conclude that assuming the hardness of the $\mathsf{LWE}_{n,q,D_{\mathbb{Z},\alpha q}}$ problem, we have $|\Pr[X_7] - \Pr[X_8]| = \mathsf{negl}(\lambda)$. □

□

Theorem 2. Our blind IBE scheme with certified identity Π_{IBE} is IND-ANON-KGC secure in the random oracle model if the PRF is pseudorandom and assuming the hardness of the $\mathsf{LWE}_{n,m,q,D_{\mathbb{Z},\alpha q}}$ problem. Alternatively, we can get rid of the first requirement by replacing the PRF by the random oracle.

Proof Overview. In our security proof, we use the fact that $\mathbf{u}_{\mathsf{ID}} = \mathsf{H}(\mathsf{ID}) \in \mathbb{Z}_q^n$ is distributed as a uniformly random vector from the view of the adversary in the random oracle model and embed $\mathbf{u}_{\mathsf{ID}}$ as the LWE secret, rather than embedding the encryption randomness $\mathbf{s}$ as in previous proofs. Recall that the LWE assumption informally states the following: given a uniformly random matrix $\mathbf{B} \leftarrow \mathbb{Z}_q^{n\times \ell}$ and some vector $\mathbf{v} \in \mathbb{Z}_q^\ell$, there is no PPT algorithm that can decide with non-negligible probability whether $\mathbf{v}$ is of the form $\mathbf{B}^\top \mathbf{d} + \mathbf{x}$ for some secret vector $\mathbf{d} \leftarrow \mathbb{Z}_q^n$ and noise $\mathbf{x}$, or a uniformly random vector over $\mathbb{Z}_q^\ell$. To restate, while in prior proofs encryption randomness $\mathbf{s}$ was set as the LWE secret $\mathbf{d}$, during our security proof, we set $\mathbf{u}_{\mathsf{ID}} = \mathsf{H}(\mathsf{ID})$ as $\mathbf{d}$ instead. Moreover, since the encryption randomness $\mathbf{s}$ of each ciphertext is completely known to the adversary in the IND-ANON-KGC setting, we set each of the $\mathbf{s}$ as the columns of the LWE matrix $\mathbf{B}$.

Proof. Let $\mathcal{A}$ be a PPT adversary against the IND-ANON-KGC security game with advantage ϵ. We assume $\mathcal{A}$ makes at most Q IssueKey queries and N encryption queries, where $Q(\lambda)$ and $N(\lambda)$ can be arbitrary large polynomials. We also define the sampling algorithm CTSamp as an algorithm which takes any $\mathsf{mpk} = \mathbf{A} \in \mathbb{Z}_q^{n\times m}$ as input and outputs $(t, \mathbf{A}^\top \mathbf{s} + \mathbf{x}) \in \mathbb{Z}_q \times \mathbb{Z}_q^m$, where $t \leftarrow \mathbb{Z}_q$, $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, and $\mathbf{x} \leftarrow D_{\mathbb{Z}^m,\alpha' q}$. In the following let X_i denote the event that $\mathcal{A}$ wins in Game_i. We modify the games so that in the final game, the adversary will have no winning advantage.

Game_0 : This is the original security game. At the beginning of the game the challenger prepares params, $(\mathsf{mpk}, \mathsf{msk})$, and $(\mathsf{vk}, \mathsf{ik})$ as specified by the game and gives $(\mathsf{params}, \mathsf{mpk}, \mathsf{msk}, \mathsf{vk})$ to $\mathcal{A}$. The challenger also prepares two empty lists IDList and HList, and an integer $Q_{\mathsf{key}} := 0$. Here, the list HList is absent in the security definition and only introduced to be used throughout this proof. Moreover, throughout this proof, for clarity we adopt the notation $\mathsf{IDList}[i] = \mathsf{ID}$ to indicate that the i-th index of the list IDList is set to ID. Initially, we have $\mathsf{IDList}[i] = \perp$ for all $i \in [Q]$. The challenger also picks a random coin $\mathsf{coin} \leftarrow \{0, 1\}$ which it keeps secret. Finally, the challenger answers to the queries made by the adversary $\mathcal{A}$ as follows:

- When $\mathcal{A}$ makes a random oracle query on ID, the challenger first checks if $(\mathsf{ID}, \star) \in \mathsf{HList}$. If so, it retrieves the (unique) tuple $(\mathsf{ID}, \mathbf{u}_{\mathsf{ID}})$ and returns $\mathbf{u}_{\mathsf{ID}}$ to $\mathcal{A}$. Otherwise, it samples a random $\mathbf{u}_{\mathsf{ID}} \leftarrow \mathbb{Z}_q^n$ and updates $\mathsf{HList} \leftarrow \mathsf{HList} \cup \{(\mathsf{ID}, \mathbf{u}_{\mathsf{ID}})\}$. Then, it returns $\mathbf{u}_{\mathsf{ID}}$ to $\mathcal{A}$. Here, the challenger can query the random oracle similarly to $\mathcal{A}$ (See the following item on IssueKey query).

- When $\mathcal{A}$ makes the j-th ($j \in [N]$) encryption query on index i and a message M_j, the challenger checks $i \in [Q_{\mathsf{key}}]$. If not, the challenger forces $\mathcal{A}$ to output a random coin $\widehat{\mathsf{coin}} \leftarrow \{0,1\}$. Otherwise, it retrieves $\mathsf{ID}_i = \mathsf{IDList}[i]$ and the unique tuple $(\mathsf{ID}_i, \mathbf{u}_{\mathsf{ID}_i}) \in \mathsf{HList}$ (which is guaranteed to exist). Then, it computes $c_0^{(j)} = \mathbf{u}_{\mathsf{ID}_i}^\top \mathbf{s}_j + x_j + \mathsf{M}_j \lfloor q/2 \rceil$ and $\mathbf{c}_1^{(j)} = \mathbf{A}^\top \mathbf{s}_j + \mathbf{x}_j$ as specified by the scheme and returns the ciphertext $\mathsf{ct} = (c_0^{(j)}, \mathbf{c}_1^{(j)})$ to $\mathcal{A}$.

- When $\mathcal{A}$ makes an IssueKey query, the challenger first samples $\mathsf{ID} \leftarrow \mathcal{ID}$. It then queries the random oracle on input ID and receives back $\mathbf{u}_{\mathsf{ID}}$. Then, it proceeds with generating $(\mathsf{cert}, \mathsf{td})$ as specified by algorithm $\mathsf{ICA.Cert}(\mathsf{vk}, \mathsf{ik}, \mathsf{ID})$ (i.e., runs the algorithm after the receiving $\mathbf{u}_{\mathsf{ID}} = \mathsf{H}(\mathsf{ID})$ back from the random oracle), and sets the first-round message $\mathsf{M}_{\mathsf{user}} = \mathsf{cert}$. It then returns $\mathsf{M}_{\mathsf{user}}$ to $\mathcal{A}$. Finally, the challenger updates $Q_{\mathsf{key}} \leftarrow Q_{\mathsf{key}} + 1$ and then sets $\mathsf{IDList}[Q_{\mathsf{key}}] = \mathsf{ID}$. Here, as in the real scheme, the randomness used to compute $\mathsf{M}_{\mathsf{user}}$ is generated by $r_{\mathsf{ID}} \leftarrow \mathsf{PRF}(\mathsf{s}_{\mathsf{ICA}}, \mathsf{ID})$, where the PRF key $\mathsf{s}_{\mathsf{ICA}}$ is included in the certificate issuing key ik which the challenger generates at the beginning of the game.

- When $\mathcal{A}$ queries for a challenge ciphertext on index i^* and a message M^*, the challenger checks $i^* \in [Q_{\mathsf{key}}]$. If not, the challenger forces $\mathcal{A}$ to output a random coin $\widehat{\mathsf{coin}} \leftarrow \{0,1\}$. Otherwise, it retrieves $\mathsf{ID}_{i^*} = \mathsf{IDList}[i^*]$ and the unique tuple $(\mathsf{ID}_{i^*}, \mathbf{u}_{\mathsf{ID}_{i^*}}) \in \mathsf{HList}$. It returns $\mathsf{ct}^* \leftarrow \mathsf{CTSamp}(\mathsf{mpk})$ if $\mathsf{coin} = 1$. Otherwise, if $\mathsf{coin} = 0$, it proceeds the same as it does for the encryption queries and returns $\mathsf{ct}^* = (c_0, \mathbf{c}_1) = (\mathbf{u}_{\mathsf{ID}_{i^*}}^\top \mathbf{s} + x + \mathsf{M}^* \lfloor q/2 \rceil, \mathbf{A}^\top \mathbf{s} + \mathbf{x})$ to $\mathcal{A}$.

At the end of the game, $\mathcal{A}$ outputs a guess $\widehat{\mathsf{coin}}$ for coin. Finally, the challenger outputs $\widehat{\mathsf{coin}}$. By definition, we have $\left|\Pr[X_0] - \frac{1}{2}\right| = \left|\Pr[\widehat{\mathsf{coin}} = \mathsf{coin}] - \frac{1}{2}\right| = \mathsf{Adv}_{\mathsf{IBE},\mathcal{A}}^{\text{IND-ANON-KGC}}(\lambda)$.

Game_1 : In this game, we change how the challenger generates the randomness used for answering the IssueKey query. In particular, the challenger always samples fresh randomness to be used when made an IssueKey query. Notably, even if the challenger happens to sample the same ID on two different IssueKey queries, it will use an independently sampled randomness. The only difference in the view of the adversary $\mathcal{A}$ in Game_0 and Game_1, is how the randomness are generated to answer the IssueKey query. Since the identity space $\mathcal{ID}$ is exponentially large and $\mathcal{A}$ only makes $Q = \mathsf{poly}(\lambda)$ many queries, the probability of sampling the same ID for two different IssueKey queries is negligible. Conditioned on this fact, the view of $\mathcal{A}$ is negligibly close assuming the pseudorandomness of the PRF. Therefore, combing the two arguments, we have $|\Pr[X_0] - \Pr[X_1]| = \mathsf{negl}(\lambda)$. In the following games, we will no longer explicitly mention the used randomness for simplicity and assume $\mathsf{ik} = \mathsf{sk}_{\mathsf{Sig}}$.

Game_2: In this game, we change how the challenger responds to the IssueKey queries. In particular, the challenger responds as follows for an IssueKey query:

- When $\mathcal{A}$ queries for an IssueKey query, the challenger first samples $\mathsf{ID} \leftarrow \mathcal{ID}$. The challenger then queries the random oracle on input ID and receives back $\mathbf{u}_{\mathsf{ID}}$. Then, it samples $\mathbf{u}_{\mathsf{ID},2} \leftarrow \mathbb{Z}_q^n$ (independent of ID) and signs $\sigma_{\mathsf{Sig}} \leftarrow \mathsf{Sig.Sign}(\mathsf{sk}_{\mathsf{Sig}}, \mathbf{u}_{\mathsf{ID},2})$, where $\mathsf{sk}_{\mathsf{Sig}}$ is included in the certificate issuing key ik. It then sets $\mathsf{M}_{\mathsf{user}} = \mathsf{cert} = (\mathbf{u}_{\mathsf{ID},2}, \sigma_{\mathsf{Sig}})$, and returns $\mathsf{M}_{\mathsf{user}}$ to $\mathcal{A}$. Finally, the challenger updates $Q_{\mathsf{key}} \leftarrow Q_{\mathsf{key}} + 1$ and sets $\mathsf{IDList}[Q_{\mathsf{key}}] = \mathsf{ID}$.

From the view of the adversary $\mathcal{A}$, the only difference between the two games are in how the vector $\mathbf{u}_{\mathsf{ID},2}$ is created. In the previous game, the challenger first sampled $\mathbf{y}_{\mathsf{ID},1} \leftarrow \{0,1\}^m$ and set $\mathbf{u}_{\mathsf{ID},2} = \mathbf{u}_{\mathsf{ID}} - \mathbf{A}\mathbf{y}_{\mathsf{ID},1}$ where $\mathbf{u}_{\mathsf{ID}} = \mathsf{H}(\mathsf{ID})$ was obtained via a random oracle query. Here, combining the three facts: $\mathsf{td} = \mathbf{y}_{\mathsf{ID},1} \leftarrow \{0,1\}^m$ is information theoretically hidden from $\mathcal{A}$; $\mathbf{A}$ is statistically close to uniform (Lemma 4); and by the left-over-hash lemma, we have that $\mathbf{u}_{\mathsf{ID},2}$ is distributed uniformly close to random over $\mathbb{Z}_q^n$ regardless of the value taken by $\mathbf{u}_{\mathsf{ID}}$. Therefore, since the distribution of $\mathbf{u}_{\mathsf{ID},2}$ is statistically close in Game_1 and Game_2, we have $|\Pr[X_1] - \Pr[X_2]| = \mathsf{negl}(\lambda)$.

Game_3: In this game, we change when the challenger queries each identity ID sampled during the IssueKey query to the random oracle. Namely, we make the following changes:

- When $\mathcal{A}$ queries for an IssueKey query the challenger directly samples $\mathbf{u} \leftarrow \mathbb{Z}_q^n$ and signs $\sigma_{\mathsf{Sig}} \leftarrow \mathsf{Sig.Sign}(\mathsf{sk}_{\mathsf{Sig}}, \mathbf{u})$. It then sets $\mathsf{M}_{\mathsf{user}} = \mathsf{cert} = (\mathbf{u}, \sigma_{\mathsf{Sig}})$, and returns $\mathsf{M}_{\mathsf{user}}$ to $\mathcal{A}$. Finally, the challenger updates $Q_{\mathsf{key}} \leftarrow Q_{\mathsf{key}} + 1$.
- When $\mathcal{A}$ makes the j-th ($j \in [N]$) encryption query on index i and a message M_j, the challenger first checks $i \in [Q_{\mathsf{key}}]$. If not, the challenger forces $\mathcal{A}$ to output a random coin $\widehat{\mathsf{coin}} \leftarrow \{0,1\}$. It then further checks if $\mathsf{IDList}[i] = \perp$. If so, the challenger samples $\mathsf{ID}_i \leftarrow \mathcal{ID}$ and sets $\mathsf{IDList}[i] = \mathsf{ID}_i$. Otherwise, it retrieves $\mathsf{ID}_i = \mathsf{IDList}[i]$. Then, the challenger queries the random oracle on input ID_i and receives back $\mathbf{u}_{\mathsf{ID}_i}$. Finally, it computes $c_0^{(j)} = \mathbf{u}_{\mathsf{ID}_i}^\top \mathbf{s}_j + x_j + \mathsf{M}_j \lfloor q/2 \rceil$ and $\mathbf{c}_1^{(j)} = \mathbf{A}^\top \mathbf{s}_j + \mathbf{x}_j$ as specified by the scheme and returns the ciphertext $\mathsf{ct} = (c_0^{(j)}, \mathbf{c}_1^{(j)})$ to $\mathcal{A}$.
- When $\mathcal{A}$ queries for a challenge ciphertext on index i^* and message M^*, it proceeds as it did to answer the encryption query.

The only difference between the previous game is the timing on which ID is sampled by the challenger; in particular, the timing when the random oracle is queried on input ID sampled by the challenger. However, since in Game_2 the challenger never required ID to answer the IssueKey query anymore due to the modification we made, it is easy to see that the view of $\mathcal{A}$ is identical in both games. Here, note that the randomness used by the challenger to answer the IssueKey queries were no longer tied to ID due to the modification we made in Game_1. In particular, the challenger is only required to check whether the i-th identity ID_i was sampled or not when it is queried on the i-th index for the encryption or challenge ciphertext query. Therefore we have $\Pr[X_2] = \Pr[X_3]$.

Game_4: In this game, at the outset of the game, the challenger samples a random index $I \leftarrow [Q]$ and keeps it secret. It then checks whether the index i^* submitted by $\mathcal{A}$ as the challenge ciphertext satisfies $i^* = I$. We call the event which this does not occur as abort. If event abort occurs, the challenger forces $\mathcal{A}$ to output a random coin $\widehat{\mathsf{coin}} \leftarrow \{0,1\}$. Otherwise, it proceeds in the same was as in the previous game. We have

$$\begin{aligned}
|\Pr[X_4] - 1/2| &= |\Pr[X_4|\neg\mathsf{abort}] \cdot \Pr[\neg\mathsf{abort}] + \Pr[X_4|\mathsf{abort}] \cdot \Pr[\mathsf{abort}] - 1/2| \\
&= |\Pr[X_3|\neg\mathsf{abort}] \cdot \Pr[\neg\mathsf{abort}] + (1/2) \cdot \Pr[\mathsf{abort}] - 1/2| \\
&= (1/Q) \cdot |\Pr[X_3] - 1/2|.
\end{aligned}$$

Here, the second line follows from the fact that conditioned on event abort not occurring Game_3 and Game_4 are identical. The final line follows from the fact that X_3 occurs independently of event abort, and when abort occurs the challenger outputs a random coin $\widehat{\mathsf{coin}}$ on behalf of $\mathcal{A}$.

Game_5: In this game, we modify how the challenger answers the encryption and challenge ciphertext query on $I \in [Q]$. In particular, we make the following modification:

- When $\mathcal{A}$ makes the j-th ($j \in [N]$) encryption query on index i and a message M_j, if $i \neq I$, then it proceeds as in the previous game. Otherwise, if $i = I$, then the challenger samples $\mathsf{ct}^{(j)} \leftarrow \mathsf{CTSamp}(\mathsf{mpk})$ and returns $\mathsf{ct}^{(j)}$ to $\mathcal{A}$.

- When $\mathcal{A}$ queries for a challenge ciphertext on index i^* and message M^*, it checks whether $i^* = I$ and forces $\mathcal{A}$ to output a random coin $\widehat{\mathsf{coin}} \leftarrow \{0,1\}$ if not satisfied. Otherwise, it returns $\mathsf{ct}^* \leftarrow \mathsf{CTSamp}(\mathsf{mpk})$ to $\mathcal{A}$ regardless of the value of $\mathsf{coin} \in \{0,1\}$.

We prove in Lemma 8 that $|\Pr[X_4] - \Pr[X_5]| = \mathsf{negl}(\lambda)$ assuming the hardness of the LWE problem.

Before we provide the proof of the lemma, we conclude our proof of Theorem 2. Observe that since the challenge ciphertext is sampled in the same way for both $\mathsf{coin} = 0$ and 1, we have $\Pr[X_5] = 1/2$. Therefore, combining everything together, we have

$$\mathsf{Adv}^{\text{IND-ANON-KGC}}_{\mathsf{IBE},\mathcal{A}}(\lambda) = |\Pr[X_0] - 1/2| = Q \cdot \mathsf{negl}(\lambda).$$

Thus, since $Q = \mathsf{poly}(\lambda)$, we conclude that $\mathsf{Adv}^{\text{IND-ANON-KGC}}_{\mathsf{IBE},\mathcal{A}}(\lambda)$ is negligible for all PPT adversary $\mathcal{A}$.

To finish the proof of Theorem 2, it remains to prove the following Lemma 8.

Lemma 8. If the $\mathsf{LWE}_{n,q,D_{\mathbb{Z},\alpha' q}}$ assumption holds, then $|\Pr[X_4] - \Pr[X_5]| = \mathsf{negl}(\lambda)$.

Proof. To prove the lemma, we use $\mathcal{A}$ to construct an LWE adversary $\mathcal{B}$ as follows: $\mathcal{B}$ is given access to an LWE oracle $\mathcal{O}_{\mathsf{LWE}}$ which returns (a fresh) $(\mathbf{v}, u) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ upon each invocation. The task of $\mathcal{B}$ is to distinguish whether $\mathcal{O}_{\mathsf{LWE}} = \mathcal{O}_{\mathbf{d}}$ in which case $u = \mathbf{v}^\top \mathbf{d} + x$ for $x \leftarrow D_{\mathbb{Z},\alpha' q}$, or $\mathcal{O}_{\mathsf{LWE}} = \mathcal{O}_{\$}$ in which case $u = u' + x$ for $u' \leftarrow \mathbb{Z}_q$ and $x \leftarrow D_{\mathbb{Z},\alpha' q}$. Looking ahead, $\mathcal{B}$ implicitly sets $\mathsf{H}(\mathsf{ID}_I) = \mathbf{d}$; if $\mathcal{O}_{\mathsf{LWE}} = \mathcal{O}_{\mathbf{d}}$, then $\mathcal{B}$ perfectly simulates Game_4 and otherwise $\mathcal{B}$ perfectly simulates Game_5.

At the beginning of the game, $\mathcal{B}$ prepares all params, $(\mathsf{mpk}, \mathsf{msk})$, $(\mathsf{vk}, \mathsf{ik})$ and samples a random coin $\mathsf{coin} \leftarrow \{0,1\}$. It also samples $I \leftarrow [Q]$ as in Game_4. It then provides $\mathcal{A}$ with $(\mathsf{params}, \mathsf{mpk}, \mathsf{msk}, \mathsf{vk}, \mathsf{ik})$ and answers to $\mathcal{A}$'s queries as follows:

- When $\mathcal{A}$ makes a random oracle query on ID, $\mathcal{B}$ first checks if $\mathsf{ID} = \mathsf{IDList}[I]$. If not, then $\mathcal{B}$ replies in the same way as the Game_4-challenger. Otherwise, $\mathcal{B}$ aborts and outputs a random $\mathsf{coin} \leftarrow \{0,1\}$ and terminates.

- When $\mathcal{A}$ queries for an IssueKey query, $\mathcal{B}$ replies in the same way as the Game_4-challenger. This can be done since $\mathcal{B}$ has ik.

- When $\mathcal{A}$ either makes the j-th ($j \in [N]$) encryption query on index i or a challenge ciphertext on index i^* with a corresponding message M, $\mathcal{B}$ first checks if $i = I$ or $i^* = I$. If not, it proceeds in the same way as the Game_4-challenger. Otherwise, $\mathcal{B}$ queries the oracle $\mathcal{O}_{\mathsf{LWE}}$ and receives $(\mathbf{v}, u) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$. It then samples $\mathbf{x} \leftarrow D_{\mathbb{Z}^m,\alpha' q}$ and sets

$$c_0 = u + \mathsf{M}\lfloor q/2 \rceil \quad \text{and} \quad \mathbf{c}_1 = \mathbf{A}^\top \mathbf{v} + \mathbf{x}.$$

  Finally, it returns $(c_0, \mathbf{c}_1)$ as $\mathsf{ct}^{(j)}$ or ct^* to $\mathcal{A}$. Here, note that $\mathcal{B}$ never queries the random oracle on input $\mathsf{ID}_I = \mathsf{IDList}[I]$.

At the end of the game, unless $\mathcal{B}$ aborts, $\mathcal{A}$ outputs $\widehat{\mathsf{coin}}$. In this case, $\mathcal{B}$ outputs 1 if $\mathsf{coin} = \widehat{\mathsf{coin}}$ and 0 otherwise. We now analyze the success probability of $\mathcal{B}$. In case $\mathcal{O}_{\mathsf{LWE}} = \mathcal{O}_{\mathbf{d}}$ for some $\mathbf{d} \leftarrow \mathbb{Z}_q^n$, then the above perfectly simulates Game_4 perfectly conditioned on $\mathcal{B}$ not aborting. Specifically, we have $\mathsf{H}(\mathsf{ID}) = \mathbf{d}$. On the other hand, when $\mathcal{O}_{\mathsf{LWE}} = \mathcal{O}_{\$}$, the above perfectly simulates Game_5 condition on $\mathcal{B}$ not aborting. This is because every c_0 is distributed randomly over $\mathbb{Z}_q$ in case u' is uniform random over $\mathbb{Z}_q$. Finally, we observe that the probability that $\mathcal{B}$ aborts is negligible. First, ID_I is information theoretically hidden from $\mathcal{A}$ since we are in the random oracle model. Therefore, since the number of IssueKey query Q is polynomially bounded and the identity space $\mathcal{ID}$ is exponentially large, the probability that $\mathcal{A}$ queries ID_I is negligible. We therefore conclude that assuming the hardness of the $\mathsf{LWE}_{n,q,D_{\mathbb{Z},\alpha' q}}$ problem, we have $|\Pr[X_4] - \Pr[X_5]| = \mathsf{negl}(\lambda)$. □

□

Theorem 3. Our blind IBE scheme with certified identity Π_{IBE} is IND-ANON-ICA secure in the random oracle model if the PRF is pseudorandom and assuming the hardness of the $\mathsf{LWE}_{n,m,q,D_{\mathbb{Z},\alpha q}}$ problem. Alternatively, we can get rid of the first requirement by replacing the PRF by the random oracle.

Proof. The IND-ANON-ICA game played between the adversary and the challenger is a strictly weaker variant of the IND-ANON-CPA game. In other words, any adversary with non-negligible advantage against the IND-ANON-ICA game also has non-negligible advantage against the IND-ANON-CPA game. Therefore, Theorem 1 proves Theorem 3. We point out that since the challenger never requires to answer a certificate query in the IND-ANON-ICA game, we do not additionally require the $\mathsf{eu\text{-}cma}$ security for the signature scheme Π_{Sig}. □

5 Pairing-based Construction

5.1 Proposed IBE scheme from Pairings

In this section, we present our pairing-based scheme. This combines the BF-IBE scheme [6] with the Boldyreva's blind signature scheme [5]. At a high level, this is similar to our lattice-based scheme where we combined the GPV-IBE with Rückert's blind signature scheme. Roughly, we are able to mix the BF-IBE scheme with Boldyreva's blind signature scheme since the signature generated by Boldyreva's scheme is a BLS signature [7] which can be interpreted as a secret key of the BF-IBE scheme.

Construction. Let $\mathbb{G}$ and $\mathbb{G}_T$ be groups with prime order p, $g \in \mathbb{G}$ be a generator, and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a pairing. Let the identity space $\mathcal{ID}$ of the IBE scheme Π_{IBE} be $\mathcal{ID} = \mathbb{Z}_p$. Finally, let $\Pi_{\mathsf{Sig}} : (\mathsf{Sig.KeyGen}, \mathsf{Sig.Sign}, \mathsf{Sig.Verify})$ be a digital signature scheme with message space $\{0,1\}^n$ for some n. Unlike the lattice-based construction, we do not assume that Π_{Sig} is deterministic. We assume that Π_{Sig} provides the standard security notion of existential unforgeability under an adaptive chosen message attack ($\mathsf{eu\text{-}cma}$).

$\mathsf{Setup}(1^\lambda)$**:** Choose $(\mathbb{G}, \mathbb{G}_T, g, p, e)$ where p be a λ-bit prime number. Output $\mathsf{params} = (1^\lambda, (\mathbb{G}, \mathbb{G}_T, g, p, e), \mathsf{H})$ where $\mathsf{H} : \{0,1\}^* \to \mathbb{G}$ is a hash function modeled as random oracle.

$\mathsf{KGC.KeyGen}(\mathsf{params})$**:** Choose $x \leftarrow \mathbb{Z}_p$ and compute $Y = g^x$. Then, output a master pubic key $\mathsf{mpk} = Y$ and a master secret key $\mathsf{msk} = x$.

$\mathsf{ICA.KeyGen}(\mathsf{params})$**:** Run $(\mathsf{vk}_{\mathsf{Sig}}, \mathsf{sk}_{\mathsf{Sig}}) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$. Then, output a certificate verification key $\mathsf{vk} = \mathsf{vk}_{\mathsf{Sig}}$ and a certificate issuing key $\mathsf{ik} = \mathsf{sk}_{\mathsf{Sig}}$.

$\mathsf{ICA.Cert}(\mathsf{vk}, \mathsf{ik}, \mathsf{ID})$**:** Parse $\mathsf{ik} = \mathsf{sk}_{\mathsf{Sig}}$ and compute $u_{\mathsf{ID}} = \mathsf{H}(\mathsf{ID})$. Then, choose $y_{\mathsf{ID},1} \leftarrow \mathbb{Z}_p$ and compute $u_{\mathsf{ID},1} = g^{y_{\mathsf{ID},1}}$. Furthermore, compute $u_{\mathsf{ID},2} = u_{\mathsf{ID}} u_{\mathsf{ID},1} \in \mathbb{G}$ and $\sigma_{\mathsf{Sig}} \leftarrow \mathsf{Sig.Sign}(\mathsf{sk}_{\mathsf{Sig}}, u_{\mathsf{ID},2})$. Finally, output a certificate $\mathsf{cert} = (u_{\mathsf{ID},2}, \sigma_{\mathsf{Sig}})$ and trapdoor information $\mathsf{td} = y_{\mathsf{ID},1}$.

$\boxed{\text{User}}$ $\boxed{\text{KGC}}$

Parse $\mathsf{cert} = (\mathbf{u}_{\mathsf{ID},2}, \sigma_{\mathsf{Sig}})$ and $\mathsf{td} = y_{\mathsf{ID},1}$.
Set $\mathsf{M}_{\mathsf{user}} = (u_{\mathsf{ID},2}, \sigma_{\mathsf{Sig}})$.

$$\xrightarrow{\mathsf{M}_{\mathsf{user}}}$$

Parse $\mathsf{vk} = \mathsf{vk}_{\mathsf{Sig}}$, $\mathsf{M}_{\mathsf{user}} = (u_{\mathsf{ID},2}, \sigma_{\mathsf{Sig}})$,
and $\mathsf{msk} = x$.
If $\mathsf{Sig.Verify}(\mathsf{vk}_{\mathsf{Sig}}, u_{\mathsf{ID},2}, \sigma_{\mathsf{Sig}}) = \bot$, then set
$\mathsf{M}_{\mathsf{KGC}} = \bot$.
Otherwise, compute $y_{\mathsf{ID},2} = u_{\mathsf{ID},2}^x$
and set $\mathsf{M}_{\mathsf{KGC}} = y_{\mathsf{ID},2}$.

$$\xleftarrow{\mathsf{M}_{\mathsf{KGC}}}$$

If $\mathsf{M}_{\mathsf{KGC}} = \bot$, then output $\bot$.
Otherwise, parse $\mathsf{M}_{\mathsf{KGC}} = y_{\mathsf{ID},2}$,
set $e_{\mathsf{ID}} = y_{\mathsf{ID},2} \cdot Y^{-y_{\mathsf{ID},1}}$,
and output $\mathsf{sk}_{\mathsf{ID}} = e_{\mathsf{ID}}$.

Figure 2: Flow of the Key-issuing Protocol (Pairing-based)

$\mathsf{IBE.Enc}(\mathsf{mpk}, \mathsf{ID}, \mathsf{M})$**:** Compute $u_{\mathsf{ID}} = \mathsf{H}(\mathsf{ID})$. To encrypt a message $\mathsf{M} \in \mathbb{G}_T$, sample $s \leftarrow \mathbb{Z}_p$, and compute $c_0 = g^s$ and $c_1 = \mathsf{M} \cdot e(u_{\mathsf{ID}}, Y)^s$. Finally, output a ciphertext $\mathsf{ct} = (c_0, c_1)$.

$\mathsf{IBE.Dec}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{ID}}, \mathsf{ct})$**:** Parse $\mathsf{sk}_{\mathsf{ID}} = \mathsf{H}(\mathsf{ID})^x$ and $\mathsf{ct} = (c_0, c_1)$. Compute $\mathsf{M} = c_1 / e(\mathsf{sk}_{\mathsf{ID}}, c_0)$ and output M.

$\langle \mathsf{ObtainKey}(\mathsf{mpk}, \mathsf{ID}, \mathsf{cert}, \mathsf{td}), \mathsf{IssueKey}(\mathsf{mpk}, \mathsf{msk}, \mathsf{vk}) \rangle$**:** The user and the KGC interactively runs $\mathsf{ObtainKey}$ and $\mathsf{IssueKey}$, respectively.

User: On input (mpk, ID, $\mathsf{cert}, \mathsf{td}$), set the first-round message $\mathsf{M}_{\mathsf{user}} = \mathsf{cert}$ and send $\mathsf{M}_{\mathsf{user}}$ to the KGC. Here, $\mathsf{cert} = (u_{\mathsf{ID},2}, \sigma_{\mathsf{Sig}})$.

KGC: On input (mpk, msk, vk) and the first-round message $\mathsf{M}_{\mathsf{user}}$, parse $\mathsf{vk} = \mathsf{vk}_{\mathsf{Sig}}$ and $\mathsf{M}_{\mathsf{user}} = (u_{\mathsf{ID},2}, \sigma_{\mathsf{Sig}})$. If $\mathsf{Sig.Verify}(\mathsf{vk}_{\mathsf{Sig}}, u_{\mathsf{ID},2}, \sigma_{\mathsf{Sig}}) = \bot$, then set $\mathsf{M}_{\mathsf{KGC}} = \bot$ and send $\mathsf{M}_{\mathsf{KGC}}$ to the user. Otherwise, parse $\mathsf{mpk} = Y$ and $\mathsf{msk} = x$. Then, compute $y_{\mathsf{ID},2} = u_{\mathsf{ID},2}^x$, set $\mathsf{M}_{\mathsf{KGC}} = y_{\mathsf{ID},2}$, and send $\mathsf{M}_{\mathsf{KGC}}$ to the user.

User: If $\mathsf{M}_{\mathsf{KGC}} = \bot$, then output $\bot$. Otherwise, parse $\mathsf{td} = y_{\mathsf{ID},1}$ and $\mathsf{M}_{\mathsf{KGC}} = y_{\mathsf{ID},2}$, compute $e_{\mathsf{ID}} = y_{\mathsf{ID},2} \cdot Y^{-y_{\mathsf{ID},1}}$ and (locally) output the secret key $\mathsf{sk}_{\mathsf{ID}} = e_{\mathsf{ID}}$.

Correctness. $\mathsf{sk}_{\mathsf{ID}} = u_{\mathsf{ID},2}^x \cdot Y^{-y_{\mathsf{ID},1}} = (\mathsf{H}(\mathsf{ID}) \cdot g^{y_{\mathsf{ID},1}})^x \cdot Y^{-y_{\mathsf{ID},1}} = \mathsf{H}(\mathsf{ID})^x \cdot Y^{y_{\mathsf{ID},1}} \cdot Y^{-y_{\mathsf{ID},1}} = \mathsf{H}(\mathsf{ID})^x$ holds. Then, $c_1 / e(\mathsf{sk}_{\mathsf{ID}}, c_0) = \mathsf{M} \cdot e(u_{\mathsf{ID}}, Y)^s / e(\mathsf{H}(\mathsf{ID})^x, g^s) = \mathsf{M}$ holds.

5.2 Security Analysis

Theorem 4. Our blind IBE scheme with certified identity Π_{IBE} is IND-ANON-CPA secure in the random oracle model if the underlying signature scheme Π_{Sig} is $\mathsf{eu\text{-}cma}$ secure, and assuming the hardness of the DBDH problem.

Proof. Let $\mathsf{CTSamp}(\mathsf{mpk})$ be an algorithm that outputs a random element from $\mathbb{G} \times \mathbb{G}_T$ and $\mathcal{A}$ a PPT algorithm which breaks the IND-ANON-CPA security of our blind IBE scheme with certified identity. We make some assumptions on $\mathcal{A}$ to make our proof simpler without loss of generality. First, we assume that $\mathcal{A}$ never queries the random oracle on the same input. Next, we assume that whenever $\mathcal{A}$ queries for a certificate or a challenge ciphertext, the corresponding ID has already been queried to the random oracle H. In the following let X_i denote the event that $\mathcal{A}$ wins in Game_i. We modify the games so that in the final game, the adversary will have no winning advantage.

Game_0 : This is the original security game. At the beginning of the game the challenger prepares params, $(\mathsf{mpk}, \mathsf{msk})$, $(\mathsf{vk}, \mathsf{ik})$ as specified by the game and gives $(\mathsf{params}, \mathsf{mpk}, \mathsf{vk})$ to $\mathcal{A}$. The challenger also prepares three empty lists IDList, CList, and HList. Here, the lists CList and HList are absent in the security definition and only introduced to be used throughout this proof. Then, the challenger picks a random coin $\mathsf{coin} \leftarrow \{0, 1\}$ and answers to the queries made by the adversary $\mathcal{A}$ as follows:

- When $\mathcal{A}$ makes a random oracle query on ID, the challenger samples a random $u_{\mathsf{ID}} \leftarrow \mathbb{G}$ and updates $\mathsf{HList} \leftarrow \mathsf{HList} \cup \{(\mathsf{ID}, u_{\mathsf{ID}}, \perp, \perp)\}$. Then, it returns u_{ID} to $\mathcal{A}$.

- When $\mathcal{A}$ queries for a certificate corresponding to ID, the challenger runs $(\mathsf{cert}, \mathsf{td}) \leftarrow \mathsf{ICA.Cert}$ $(\mathsf{vk}, \mathsf{ik}, \mathsf{ID})$ and returns $(\mathsf{cert}, \mathsf{td})$ to $\mathcal{A}$. It further updates $\mathsf{IDList} \leftarrow \mathsf{IDList} \cup \{\mathsf{ID}\}$ and $\mathsf{CList} \leftarrow \mathsf{CList} \cup \{(\mathsf{cert}, \mathsf{td}, \mathsf{ID})\}$.

- When $\mathcal{A}$ queries for a secret key with a first-round message $\mathsf{M}_{\mathsf{user}}$, the challenger parses $\mathsf{M}_{\mathsf{user}} = (u_{\mathsf{ID},2}, \sigma_{\mathsf{Sig}})$ and returns the second-round message $\mathsf{M}_{\mathsf{KGC}} = y_{\mathsf{ID},2}$ or $\perp$ to $\mathcal{A}$ depending on $\mathsf{M}_{\mathsf{user}}$.

- When $\mathcal{A}$ queries for a challenge ciphertext on ID^* and message M^*, the challenger returns $\mathsf{ct}^* \leftarrow \mathsf{IBE.Enc}(\mathsf{mpk}, \mathsf{ID}^*, \mathsf{M}^*)$ if $\mathsf{coin} = 0$ and $\mathsf{ct}^* \leftarrow \mathsf{CTSamp}(\mathsf{mpk})$ if $\mathsf{coin} = 1$.

At the end of the game, $\mathcal{A}$ outputs a guess $\widehat{\mathsf{coin}}$ for coin. Finally, the challenger outputs $\widehat{\mathsf{coin}}$. By definition, we have $\left|\Pr[X_0] - \frac{1}{2}\right| = \left|\Pr[\widehat{\mathsf{coin}} = \mathsf{coin}] - \frac{1}{2}\right| = \mathsf{Adv}^{\text{IND-ANON-CPA}}_{\mathsf{IBE},\mathcal{A}}(\lambda)$.

Game_1 : In this game, we change how the challenger answers the secret key queries. In particular, the challenger responds as follows for a secret key query:

- When $\mathcal{A}$ queries for a secret key with input a first-round message $\mathsf{M}_{\mathsf{user}} = (u_{\mathsf{ID},2}, \sigma_{\mathsf{Sig}})$, the challenger first checks whether $\mathsf{Sig.Verify}(\mathsf{vk}_{\mathsf{Sig}}, u_{\mathsf{ID},2}, \sigma_{\mathsf{Sig}}) = \top$. If not, it returns the second-round message $\mathsf{M}_{\mathsf{KGC}} = \perp$ to $\mathcal{A}$. Next, it sets $\mathsf{cert} = (u_{\mathsf{ID},2}, \sigma_{\mathsf{Sig}})$ and checks whether $(\mathsf{cert}, \star, \star) \in \mathsf{CList}$, where $\star$ represents an arbitrary value. If not, the challenger aborts the game and forces $\mathcal{A}$ to output a random coin. Otherwise, the challenger proceeds as in the previous game.

As in Lemma 6, we have $|\Pr[X_0] - \Pr[X_1]| = \mathsf{negl}(\lambda)$ by the $\mathsf{eu\text{-}cma}$ security of the underlying signatures scheme. We omit the proof since it is the same as that of Lemma 6.

Game_2: In this game, we change how the challenger generates the challenge ciphertext. When $\mathcal{A}$ queries for a challenge ciphertext on ID^* and message M^*, the challenger returns $\mathsf{ct}^* \leftarrow \mathsf{CTSamp}(\mathsf{mpk})$ regardless of coin. In this game, the adversary has no winning advantage since the challenge ciphertext is distributed identically for both $\mathsf{coin} = 0$ and 1. Namely, we have $|\Pr[X_2] - 1/2| = 0$. Finally, we show that $|\Pr[X_0] - \Pr[X_1]| = \mathsf{negl}(\lambda)$ by the DBDH assumption.

Lemma 9. If the DBDH assumption holds, then $|\Pr[X_1] - \Pr[X_2]| = \mathsf{negl}(\lambda)$.

Proof. Let (g, g^a, g^b, g^c, Z) be a DBDH instance. We use $\mathcal{A}$ to construct a DBDH adversary $\mathcal{B}$ as follows: At the begining of game, $\mathcal{B}$ sets $Y := g^a$, i.e., implicitly sets $x := a$. Moreover, $\mathcal{B}$ sets δ to be some real in $(0, 1)$ which we defined later. Then, $\mathcal{B}$ answers to the queries made by the adversary $\mathcal{A}$ as follows:

- When $\mathcal{A}$ makes a random oracle query on ID, $\mathcal{B}$ samples a random $\bar{b} \leftarrow \mathbb{Z}_p$. With the probability $1-\delta$, $\mathcal{B}$ sets $u_{\mathsf{ID}} := g^{\bar{b}}$ and updates $\mathsf{HList} \leftarrow \mathsf{HList} \cup \{(\mathsf{ID}, u_{\mathsf{ID}}, \bar{b}, 0)\}$. Otherwise, $\mathcal{B}$ sets $u_{\mathsf{ID}} := (g^b)^{\bar{b}}$ and updates $\mathsf{HList} \leftarrow \mathsf{HList} \cup \{(\mathsf{ID}, u_{\mathsf{ID}}, \bar{b}, 1)\}$. Then, $\mathcal{B}$ returns u_{ID} to $\mathcal{A}$.

- When $\mathcal{A}$ queries for a certificate corresponding to ID, $\mathcal{B}$ samples $y_{\mathsf{ID},1} \leftarrow \mathbb{Z}_p$, computes $u_{\mathsf{ID},1} = g^{y_{\mathsf{ID},1}}$, $u_{\mathsf{ID},2} = u_{\mathsf{ID}} u_{\mathsf{ID},1}$, and $\sigma_{\mathsf{Sig}} \leftarrow \mathsf{Sig.Sign}(\mathsf{sk}_{\mathsf{Sig}}, u_{\mathsf{ID},2})$. $\mathcal{B}$ returns $(\mathsf{cert}, \mathsf{td})$ to $\mathcal{A}$ where $\mathsf{cert} = (u_{\mathsf{ID},2}, \sigma_{\mathsf{Sig}})$ and $\mathsf{td} = y_{\mathsf{ID},1}$. $\mathcal{B}$ further updates $\mathsf{IDList} \leftarrow \mathsf{IDList} \cup \{\mathsf{ID}\}$ and $\mathsf{CList} \leftarrow \mathsf{CList} \cup \{(\mathsf{cert}, \mathsf{td}, \mathsf{ID})\}$.

- When $\mathcal{A}$ queries for a secret key with a first-round message $\mathsf{M}_{\mathsf{user}}$, $\mathcal{B}$ parses $\mathsf{M}_{\mathsf{user}} = (u_{\mathsf{ID},2}, \sigma_{\mathsf{Sig}})$. $\mathcal{B}$ returns $\mathsf{M}_{\mathsf{KGC}} = \perp$ if $\mathsf{Sig.Verify}(\mathsf{vk}_{\mathsf{Sig}}, u_{\mathsf{ID},2}, \sigma_{\mathsf{Sig}}) = \perp$. Otherwise, $\mathcal{B}$ extracts $y_{\mathsf{ID},1}$ from $((u_{\mathsf{ID},2}, \sigma_{\mathsf{Sig}}), y_{\mathsf{ID},1}, \mathsf{ID}) \in \mathsf{CList}$. We remark that such an entry always exists due to the modification we made in Game_1. $\mathcal{B}$ extracts $\bar{b}$ from $(\mathsf{ID}, u_{\mathsf{ID}}, \bar{b}, \bar{c}) \in \mathsf{HList}$. If $\bar{c} = 1$, then $\mathcal{B}$ aborts the game and forces $\mathcal{A}$ to output a random coin. Othwerise, if $\bar{c} = 0$, then $\mathcal{B}$ computes $y_{\mathsf{ID},2} = (g^a)^{y_{\mathsf{ID},1}+\bar{b}}$. Here, $u_{\mathsf{ID},2}^x = (u_{\mathsf{ID}} u_{\mathsf{ID},1})^x = (g^{\bar{b}} g^{y_{\mathsf{ID},1}})^a = (g^a)^{y_{\mathsf{ID},1}+\bar{b}}$ holds (since $x = a$). $\mathcal{B}$ returns the second-round message $\mathsf{M}_{\mathsf{KGC}} = y_{\mathsf{ID},2}$ to $\mathcal{A}$.

- When $\mathcal{A}$ queries for a challenge ciphertext on ID^* and message M^*, $\mathcal{B}$ extracts the entry $(\mathsf{ID}^*, u_{\mathsf{ID}^*}, \bar{b}^*, \bar{c}^*) \in \mathsf{HList}$. If $\bar{c}^* = 0$, then $\mathcal{B}$ aborts the game and forces $\mathcal{A}$ to output a random coin. Othwerise, if $\bar{c}^* = 1$, then $\mathcal{B}$ sets $c_0^* := g^c$ and computes $c_1^* = \mathsf{M}^* \cdot Z^{\bar{b}^*}$.

It is clear that if $Z = e(g,g)^{abc}$, then $\mathcal{B}$ perfectly simulates Game_1 to $\mathcal{A}$ conditioned on not aborting since $c_1^* = \mathsf{M}^* \cdot (e(g,g)^{abc})^{\bar{b}^*} = \mathsf{M}^* \cdot e((g^b)^{\bar{b}^*}, g^a)^c = \mathsf{M}^* \cdot e(\mathsf{H}(\mathsf{ID}^*), Y)^c$. On the other hand, if Z is random, then $\mathcal{B}$ perfectly simulates Game_2 condition on not aborting. Now, the probability that $\mathcal{B}$ does not abort is $\delta(1-\delta)^Q$ where Q is the number of random oracle queries. We set $\delta_{\mathsf{opt}} = 1/(1+Q)$ to maximize this probability, and in this case, the probability that $\mathcal{B}$ does not abort is at least $(\hat{e} \cdot (1+Q))^{-1}$ where $\hat{e}$ is the base of the natural logarithm. Therefore, since $\mathcal{B}$ does not abort with non-negligible probability, assuming the hardness of DBDH, no adversary can distinguish Game_1 and Game_2 with non-negligible probability. □

□

Theorem 5. Our blind IBE scheme with certified identity Π_{IBE} is IND-ANON-KGC secure in the random oracle model assuming the hardness of the DBDH problem.

Proof. Let $\mathcal{A}$ be a PPT adversary against the IND-ANON-KGC security game with advantage ϵ. We assume $\mathcal{A}$ makes at most Q IssueKey queries, and N encryption queries, where $Q(\lambda)$ and $N(\lambda)$ can be arbitrary large polynomials. We also define the sampling algorithm CTSamp as an algorithm which outputs a random element from $\mathbb{G} \times \mathbb{G}_T$. In the following let X_i denote the event that $\mathcal{A}$ wins in Game_i. We modify the games so that in the final game, the adversary will have no winning advantage.

Game_0 : This is the original security game. At the beginning of the game the challenger prepares params, $(\mathsf{mpk}, \mathsf{msk}) = (Y, x)$, and $(\mathsf{vk}, \mathsf{ik})$ as specified by the game and gives $(\mathsf{params}, \mathsf{mpk}, \mathsf{msk}, \mathsf{vk})$ to $\mathcal{A}$. The challenger also prepares two empty lists IDList and HList, and an integer $Q_{\mathsf{key}} := 0$. Here, the list HList is absent in the security definition and only introduced to be used throughout this proof. Moreover, throughout this proof, for clarity we adopt the notation $\mathsf{IDList}[i] = \mathsf{ID}$ to indicate that the i-th index of the list IDList is set to ID. Initially, we have $\mathsf{IDList}[i] = \perp$ for all $i \in [Q]$. The challenger also picks a random coin $\mathsf{coin} \leftarrow \{0,1\}$ which it keeps secret. Finally, the challenger answers to the queries made by the adversary $\mathcal{A}$ as follows:

- When $\mathcal{A}$ makes a random oracle query on ID, the challenger first checks if $(\mathsf{ID}, \star) \in \mathsf{HList}$. If so, it retrieves the (unique) tuple $(\mathsf{ID}, u_{\mathsf{ID}})$ and returns u_{ID} to $\mathcal{A}$. Otherwise, it samples a random $u_{\mathsf{ID}} \leftarrow \mathbb{G}$ and updates $\mathsf{HList} \leftarrow \mathsf{HList} \cup \{(\mathsf{ID}, u_{\mathsf{ID}})\}$. Then, it returns u_{ID} to $\mathcal{A}$. Here, the challenger can query the random oracle similarly to $\mathcal{A}$ (See the following item on IssueKey query).

- When $\mathcal{A}$ makes the j-th ($j \in [N]$) encryption query on index i and a message M_j, the challenger checks $i \in [Q_{\mathsf{key}}]$. If not, the challenger forces $\mathcal{A}$ to output a random coin $\widehat{\mathsf{coin}} \leftarrow \{0, 1\}$. Otherwise, it retrieves $\mathsf{ID}_i = \mathsf{IDList}[i]$ and the unique tuple $(\mathsf{ID}_i, u_{\mathsf{ID}_i}) \in \mathsf{HList}$ (which is guaranteed to exist). Then, it computes $c_0^{(j)} = g^s$ and and $c_1^{(j)} = \mathsf{M} \cdot e(u_{\mathsf{ID}}, Y)^s$ as specified by the scheme and returns the ciphertext $\mathsf{ct} = (c_0^{(j)}, c_1^{(j)})$ to $\mathcal{A}$.

- When $\mathcal{A}$ makes an IssueKey query, the challenger first samples $\mathsf{ID} \leftarrow \mathcal{ID}$. It then queries the random oracle on input ID and receives back u_{ID}. Then, it proceeds with generating $(\mathsf{cert}, \mathsf{td})$ as specified by algorithm $\mathsf{ICA.Cert}(\mathsf{vk}, \mathsf{ik}, \mathsf{ID})$ (i.e., runs the algorithm after the receiving $u_{\mathsf{ID}} = \mathsf{H}(\mathsf{ID})$ back from the random oracle), and sets the first-round message $\mathsf{M}_{\mathsf{user}} = \mathsf{cert}$. It then returns $\mathsf{M}_{\mathsf{user}}$ to $\mathcal{A}$. Finally, the challenger updates $Q_{\mathsf{key}} \leftarrow Q_{\mathsf{key}} + 1$ and then sets $\mathsf{IDList}[Q_{\mathsf{key}}] = \mathsf{ID}$.

- When $\mathcal{A}$ queries for a challenge ciphertext on index i^* and a message M^*, the challenger checks $i^* \in [Q_{\mathsf{key}}]$. If not, the challenger forces $\mathcal{A}$ to output a random coin $\widehat{\mathsf{coin}} \leftarrow \{0, 1\}$. Otherwise, it retrieves $\mathsf{ID}_{i^*} = \mathsf{IDList}[i^*]$ and the unique tuple $(\mathsf{ID}_{i^*}, u_{\mathsf{ID}_{i^*}}) \in \mathsf{HList}$. It returns $\mathsf{ct}^* \leftarrow \mathsf{CTSamp}(\mathsf{mpk})$ if $\mathsf{coin} = 1$. Otherwise, if $\mathsf{coin} = 0$, it proceeds the same as it does for the encryption queries and returns $\mathsf{ct}^* = (c_0, c_1) = (g^s, \mathsf{M}^* \cdot e(u_{\mathsf{ID}_{i^*}}, Y)^s)$ to $\mathcal{A}$.

At the end of the game, $\mathcal{A}$ outputs a guess $\widehat{\mathsf{coin}}$ for coin. Finally, the challenger outputs $\widehat{\mathsf{coin}}$. By definition, we have $\left|\Pr[X_0] - \frac{1}{2}\right| = \left|\Pr[\widehat{\mathsf{coin}} = \mathsf{coin}] - \frac{1}{2}\right| = \mathsf{Adv}_{\mathsf{IBE},\mathcal{A}}^{\text{IND-ANON-KGC}}(\lambda)$.

Game_1: In this game, we change how the challenger responds to the IssueKey queries. In particular, the challenger responds as follows for an IssueKey query:

- When $\mathcal{A}$ queries for an IssueKey query, the challenger first samples $\mathsf{ID} \leftarrow \mathcal{ID}$. The challenger then queries the random oracle on input ID and receives back u_{ID}. Then, it randomly samples $u_{\mathsf{ID},2} \leftarrow \mathbb{G}$ (independent of ID) and signs $\sigma_{\mathsf{Sig}} \leftarrow \mathsf{Sig.Sign}(\mathsf{sk}_{\mathsf{Sig}}, u_{\mathsf{ID},2})$, where $\mathsf{ik} = \mathsf{sk}_{\mathsf{Sig}}$. It then sets $\mathsf{M}_{\mathsf{user}} = \mathsf{cert} = (u_{\mathsf{ID},2}, \sigma_{\mathsf{Sig}})$, and returns $\mathsf{M}_{\mathsf{user}}$ to $\mathcal{A}$. Finally, the challenger updates $Q_{\mathsf{key}} \leftarrow Q_{\mathsf{key}} + 1$ and sets $\mathsf{IDList}[Q_{\mathsf{key}}] = \mathsf{ID}$.

From the view of the adversary $\mathcal{A}$, the only difference between the two games are in how the element $u_{\mathsf{ID},2}$ is created. In the previous game, the challenger first sampled $y_{\mathsf{ID},1} \leftarrow \mathbb{Z}_p$ and set $u_{\mathsf{ID},2} = u_{\mathsf{ID}} \cdot g^{y_{\mathsf{ID},1}}$ where $u_{\mathsf{ID}} = \mathsf{H}(\mathsf{ID})$ was obtained via a random oracle query. Since $y_{\mathsf{ID},1}$ is information theoretically hidden from $\mathcal{A}$, $u_{\mathsf{ID},2}$ is distributed uniformly close to random over $\mathbb{G}$ regardless of the value taken by u_{ID}. Therefore, the distribution of $u_{\mathsf{ID},2}$ is the same in Game_0 and Game_1. Specifically, we have $\Pr[X_0] = \Pr[X_1]$. Due to the modification we made in this game, the challenger no longer requires to know $\mathsf{H}(\mathsf{ID}) = u_{\mathsf{ID}}$ to answer $\mathcal{A}$'s IssueKey query.

Game_2: In this game, we change when the challenger queries each identity ID sampled during the IssueKey query to the random oracle. Namely, we make the following changes:

- When $\mathcal{A}$ queries for an IssueKey query the challenger directly samples $u \leftarrow \mathbb{G}$ and signs $\sigma_{\mathsf{Sig}} \leftarrow \mathsf{Sig.Sign}(\mathsf{sk}_{\mathsf{Sig}}, u)$. It then sets $\mathsf{M}_{\mathsf{user}} = \mathsf{cert} = (u, \sigma_{\mathsf{Sig}})$, and returns $\mathsf{M}_{\mathsf{user}}$ to $\mathcal{A}$. Finally, the challenger updates $Q_{\mathsf{key}} \leftarrow Q_{\mathsf{key}} + 1$.

- When $\mathcal{A}$ makes the j-th ($j \in [N]$) encryption query on index i and a message M_j, the challenger first checks $i \in [Q_{\mathsf{key}}]$. If not, the challenger forces $\mathcal{A}$ to output a random coin $\widehat{\mathsf{coin}} \leftarrow \{0,1\}$. It then further checks if $\mathsf{IDList}[i] = \perp$. If so, the challenger samples $\mathsf{ID}_i \leftarrow \mathcal{ID}$ and sets $\mathsf{IDList}[i] = \mathsf{ID}_i$. Otherwise, it retrieves $\mathsf{ID}_i = \mathsf{IDList}[i]$. Then, the challenger queries the random oracle on input ID_i and receives back u_{ID_i}. Finally, it computes $c_0^{(j)} = g^s$ and $c_1^{(j)} = \mathsf{M} \cdot e(u_{\mathsf{ID}_i}, Y)^s$ as specified by the scheme and returns the ciphertext $\mathsf{ct} = (c_0^{(j)}, c_1^{(j)})$ to $\mathcal{A}$.

- When $\mathcal{A}$ queries for a challenge ciphertext on index i^* and message M^*, it proceeds as it did to answer the encryption query.

The only difference between the previous game is the timing on which ID is sampled by the challenger; in particular, the timing when the random oracle is queried on input ID sampled by the challenger. However, since in Game_1 the challenger never required ID to answer the IssueKey query anymore due to the modification we made, it is easy to see that the view of $\mathcal{A}$ is identical in both games. Therefore we have $\Pr[X_1] = \Pr[X_2]$.

Game_3: In this game, at the outset of the game, the challenger samples a random index $I \leftarrow [Q]$ and keeps it secret. It then checks whether the index i^* submitted by $\mathcal{A}$ as the challenge ciphertext satisfies $i^* = I$. We call the event which this does not occur as abort. If event abort occurs, the challenger forces $\mathcal{A}$ to output a random coin $\widehat{\mathsf{coin}} \leftarrow \{0,1\}$. Otherwise, it proceeds in the same was as in the previous game. Using the same argument used in Theorem 2, we have $|\Pr[X_3] - 1/2| = (1/Q) \cdot |\Pr[X_2] - 1/2|$.

Game_4: In this game, we modify how the challenger answers the challenge ciphertext query. Specifically, we make the following modification:

- When $\mathcal{A}$ queries for a challenge ciphertext on index i^* and message M^*, it checks whether $i^* = I$ and forces $\mathcal{A}$ to output a random coin $\widehat{\mathsf{coin}} \leftarrow \{0,1\}$ if not satisfied. Otherwise, it returns $\mathsf{ct}^* \leftarrow \mathsf{CTSamp}(\mathsf{mpk})$ to $\mathcal{A}$ regardless of the value of $\mathsf{coin} \in \{0,1\}$.

We prove in Lemma 10 that $|\Pr[X_3] - \Pr[X_4]| = \mathsf{negl}(\lambda)$ assuming the hardness of the DBDH problem.

Before we provide the proof of the lemma, we conclude are proof of Theorem 5. Observe that since the challenge ciphertext is sampled in the same way for both $\mathsf{coin} = 0$ and 1, we have $\Pr[X_4] = 1/2$. Therefore, combining everything together, we have

$$\mathsf{Adv}_{\mathsf{IBE},\mathcal{A}}^{\text{IND-ANON-KGC}}(\lambda) = |\Pr[X_0] - 1/2| = Q \cdot \mathsf{negl}(\lambda).$$

Thus, since $Q = \mathsf{poly}(\lambda)$, we conclude that $\mathsf{Adv}_{\mathsf{IBE},\mathcal{A}}^{\text{IND-ANON-KGC}}(\lambda)$ is negligible for all PPT adversary $\mathcal{A}$.

To finish the proof of Theorem 5, it remains to prove the following Lemma 10.

Lemma 10. If the DBDH assumption holds, then $|\Pr[X_3] - \Pr[X_4]| = \mathsf{negl}(\lambda)$.

Proof. To prove the lemma, we use $\mathcal{A}$ to construct an DBDH adversary $\mathcal{B}$ as follows: Let (g, g^a, g^b, g^c, Z) be an DBDH instance. The task of $\mathcal{B}$ is to distinguish whether $Z = e(g,g)^{abc}$, or random. Looking ahead, $\mathcal{B}$ implicitly sets $\mathsf{H}(\mathsf{ID}_I) = g^{ab}$. If $Z = e(g,g)^{abc}$, then $\mathcal{B}$ perfectly simulates Game_3 and if Z is random, then $\mathcal{B}$ perfectly simulates Game_4.

At the beginning of the game, $\mathcal{B}$ prepares all params, $(\mathsf{mpk}, \mathsf{msk}) = (Y, x)$, $(\mathsf{vk}, \mathsf{ik})$ and samples a random coin $\mathsf{coin} \leftarrow \{0,1\}$ and $I \leftarrow [Q]$. It then provides $\mathcal{A}$ with $(\mathsf{params}, \mathsf{mpk}, \mathsf{msk}, \mathsf{vk}, \mathsf{ik})$ and answers to $\mathcal{A}$'s queries as follows:

- When $\mathcal{A}$ makes a random oracle query on ID, $\mathcal{B}$ first checks if $\mathsf{ID} = \mathsf{IDList}[I]$. If not, then $\mathcal{B}$ replies in the same way as the Game_3-challenger. If $\mathsf{ID} = \mathsf{IDList}[I]$, then $\mathcal{B}$ aborts and outputs a random $\mathsf{coin} \leftarrow \{0, 1\}$ and terminates.

- When $\mathcal{A}$ queries for an IssueKey query, $\mathcal{B}$ replies in the same way as the Game_3-challenger. This can be done since $\mathcal{B}$ has ik.

- When $\mathcal{A}$ makes the j-th ($j \in [N]$) encryption query on index i with a corresponding message M_j, $\mathcal{B}$ first checks if $i = I$. If so, $\mathcal{B}$ chooses $s_j \leftarrow \mathbb{Z}_p$ and computes the ciphertext as

$$c_0 = g^{s_j} \quad \text{and} \quad c_1 = \mathsf{M}_j \cdot e(g^a, g^b)^{x s_j}, \tag{2}$$

  Otherwise, $\mathcal{B}$ computes (c_0, c_1) in the same way as the Game_3-challenger. Finally, it returns (c_0, c_1) as $\mathsf{ct}^{(j)}$ to $\mathcal{A}$. Here, note that $\mathcal{B}$ never queries the random oracle on input $\mathsf{ID}_I = \mathsf{IDList}[I]$.

- When $\mathcal{A}$ queries for a challenge ciphertext on index i^* and message M^*, $\mathcal{B}$ first checks $i^* = I$. If not, it forces $\mathcal{A}$ to output a random coin $\widehat{\mathsf{coin}} \leftarrow \{0, 1\}$. If $i^* = I$, $\mathcal{B}$ then computes the challenge ciphertext as follows.

$$c_0^* = g^c \quad \text{and} \quad c_1^* = \mathsf{M}^* \cdot Z^x.$$

  Finally, it returns (c_0^*, c_1^*) as ct^* to $\mathcal{A}$.

At the end of the game, unless $\mathcal{B}$ aborts, $\mathcal{A}$ outputs $\widehat{\mathsf{coin}}$. In this case, $\mathcal{B}$ outputs 1 if $\mathsf{coin} = \widehat{\mathsf{coin}}$ and 0 otherwise. We now analyze the success probability of $\mathcal{B}$. Recall that $\mathcal{B}$ implicitly sets $\mathsf{H}(\mathsf{ID}_I) = g^{ab}$. First, we observe that encryption queries for the case $i = I$ are answered perfectly regardless of the value Z; c_1 in Equation (2) satisfies $c_1 = \mathsf{M}_j \cdot e(g^{ab}, g^x)^{s_j} = \mathsf{M}_j \cdot e(\mathsf{H}(\mathsf{ID}_I), Y)^{s_j}$. Now, in case $Z = e(g, g)^{abc}$, then the above perfectly simulates Game_3 conditioned on $\mathcal{B}$ not aborting since $Z^x = e(g, g)^{abcx} = e(\mathsf{H}(\mathsf{ID}_I), Y)^c$ holds and g^c is randomness used nowhere else. On the other hand, when Z is random, the above perfectly simulates Game_4 conditioned on $\mathcal{B}$ not aborting. This follows since g^c and Z are independently random and not used anywhere else. Finally, we observe that the probability that $\mathcal{B}$ aborts is negligible. First, ID_I is information theoretically hidden from $\mathcal{A}$ since we are in the random oracle model. Therefore, since the number of IssueKey query Q is polynomially bounded and the identity space $\mathcal{ID}$ is exponentially large, the probability that $\mathcal{A}$ queries ID_I is negligible. We therefore conclude that assuming the hardness of the DBDH problem, we have $|\Pr[X_3] - \Pr[X_4]| = \mathsf{negl}(\lambda)$. $\square$

This completes the proof of Theorem 5. $\square$

As in the lattice-based one, the following theorem holds.

Theorem 6. Our blind IBE scheme with certified identity Π_{IBE} is IND-ANON-ICA secure in the random oracle model assuming the hardness of the DBDH problem.

Acknowledgments: This work was supported by JSPS KAKENHI Grant Numbers JP16K00198, JP17K12697, and JP17J05603, and was supported by JST CREST Grant Number JPMJCR19F6, Japan. We would like to thank Prof. Sherman S. M. Chow for his invaluable comment.

References

[1] M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi. Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. *J. Cryptology*, 21(3):350–391, 2008. 2

[2] S. Agrawal, D. Boneh, and X. Boyen. Efficient lattice (H)IBE in the standard model. In *EUROCRYPT*, pages 553–572, 2010. 13

[3] S. S. Al-Riyami and K. G. Paterson. Certificateless public key cryptography. In *ASIACRYPT*, pages 452–473, 2003. 2

[4] R. Bendlin, S. Krehbiel, and C. Peikert. How to share a lattice trapdoor: Threshold protocols for signatures and (H)IBE. In *ACNS*, pages 218–236, 2013. 4

[5] A. Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In *Public Key Cryptography*, pages 31–46, 2003. 5, 24

[6] D. Boneh and M. K. Franklin. Identity-based encryption from the weil pairing. In *CRYPTO*, pages 213–229, 2001. 3, 4, 5, 24

[7] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the weil pairing. In *ASIACRYPT*, pages 514–532, 2001. 5, 24

[8] X. Boyen and B. Waters. Anonymous hierarchical identity-based encryption (without random oracles). In *CRYPTO*, pages 290–307, 2006. 2

[9] Z. Brakerski, A. Langlois, C. Peikert, O. Regev, and D. Stehlé. Classical hardness of learning with errors. In *STOC*, pages 575–584, 2013. 7

[10] J. Camenisch, M. Kohlweiss, A. Rial, and C. Sheedy. Blind and anonymous identity-based encryption and authorised private searches on public key encrypted data. In *PKC*, pages 196–214, 2009. 3, 8

[11] D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert. Bonsai trees, or how to delegate a lattice basis. In *EUROCRYPT*, pages 523–552, 2010. 13

[12] S. S. M. Chow. Removing escrow from identity-based encryption. In *PKC*, pages 256–276, 2009. 2, 3, 4, 5, 10, 11

[13] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *CRYPTO*, pages 13–25, 1998. 15

[14] K. Emura, S. Katsumata, and Y. Watanabe. Identity-based encryption with security against the KGC: A formal model and its instantiation from lattices. In *ESORICS*, pages 113–133, 2019. 1

[15] S. Garg, M. Hajiabadi, M. Mahmoody, and A. Rahimi. Registration-based encryption: Removing private-key generator from IBE. In *TCC*, pages 689–718, 2018. 2

[16] S. Garg, M. Hajiabadi, M. Mahmoody, A. Rahimi, and S. Sekar. Registration-based encryption from standard assumptions. In *PKC*, pages 63–93, 2019. 2

[17] C. Gentry. Practical identity-based encryption without random oracles. In *EUROCRYPT*, pages 445–464, 2006. 5

[18] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206, 2008. 4, 6, 12, 13, 14, 15

[19] R. Goyal and S. Vusirikala. Verifiable registration-based encryption. Cryptology ePrint Archive, Report 2019/1044, 2019. https://eprint.iacr.org/2019/1044. 2

[20] M. Green and S. Hohenberger. Blind identity-based encryption and simulatable oblivious transfer. In *ASIACRYPT*, pages 265–282, 2007. 3, 8

[21] D. Hofheinz, J. Koch, and C. Striecks. Identity-based encryption with (almost) tight security in the multi-instance, multi-ciphertext setting. In *PKC*, pages 799–822. Springer, 2015. 5

[22] M. Izabachène and D. Pointcheval. New anonymity notions for identity-based encryption. In *SCN*, pages 375–391, 2008. 2

[23] S. Katsumata and S. Yamada. Partitioning via non-linear polynomial functions: More compact IBEs from ideal lattices and bilinear maps. In *ASIACRYPT*, pages 682–712, 2016. 6

[24] S. Katsumata, S. Yamada, and T. Yamakawa. Tighter security proofs for GPV-IBE in the quantum random oracle model. In *ASIACRYPT*, pages 253–282, 2018. 5, 6, 15

[25] M. Kumar and S. Chand. ESKI-IBE: efficient and secure key issuing identity-based encryption with cloud privacy centers. *Multimedia Tools Appl.*, 78(14):19753–19786, 2019. 3

[26] D. Micciancio and C. Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT*, pages 700–718, 2012. 6

[27] C. Peikert. Public-key cryptosystems from the worst-case shortest vector problem. In *STOC*, pages 333–342. ACM, 2009. 7

[28] C. Peikert. An efficient and parallel gaussian sampler for lattices. In *CRYPTO*, pages 80–97. Springer, 2010. 14

[29] F. Qi, X. Tang, and Q. Wei. New escrow-free scheme for hierarchical identity-based encryption. In *ICA3PP*, pages 701–713, 2015. 3

[30] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, pages 84–93. ACM Press, 2005. 6, 7

[31] M. Rückert. Lattice-based blind signatures – preliminary version –. In *Algorithms and Number Theory*, 2009. 4, 12

[32] M. Rückert. Lattice-based blind signatures. In *ASIACRYPT*, pages 413–430, 2010. 4, 12, 14

[33] A. Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO*, pages 47–53, 1984. 2

[34] Q. Wei, F. Qi, and Z. Tang. Remove key escrow from the BF and Gentry identity-based encryption with non-interactive key generation. *Telecommunication Systems*, 69(2):253–262, 2018. 3