Theoretical Computer Science 863 (2021) 127–155

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Efficient revocable identity-based encryption with short public parameters ☆

Keita Emura [a], Jae Hong Seo [b], Yohei Watanabe [c,d,*]

[a] The National Institute of Information and Communication Technology, 4-2-1 Nukui-Kitamachi, Koganei, Tokyo, 184-8795, Japan
[b] Hanyang University, 222 Wangsimni-ro, Seongdong-gu, Seoul, 04763, Republic of Korea
[c] Graduate School of Informatics and Engineering, The University of Electro-Communications, 1-5-1 Chofugaoka, Chofu, 182-8585, Japan
[d] Cyber Physical Security Research Center (CPSEC), National Institute of Advanced Industrial Science and Technology (AIST), 2-4-7 Aomi, Koto-ku, Tokyo, 135-0064, Japan

ARTICLE INFO

Article history:
Received 16 October 2017
Received in revised form 8 February 2021
Accepted 15 February 2021
Available online 18 February 2021
Communicated by X. Deng

Keywords:
Revocable identity-based encryption
Decryption key exposure resistance
Static assumptions
Asymmetric pairings

ABSTRACT

Revocation functionality is vital to real-world cryptographic systems for managing their reliability. In the context of identity-based encryption (IBE), Boldyreva, Goyal, and Kumar (ACM CCS 2008) first showed an efficient revocation method for IBE, and such an IBE scheme with the scalable revocation method is called revocable IBE (RIBE). Seo and Emura (PKC 2013) introduced a new security notion, called *decryption key exposure resistance (DKER)*, which is a desirable security notion for RIBE. However, all existing RIBE schemes that achieve adaptive security with DKER require long public parameters or composite-order bilinear groups.

In this paper, we first show an RIBE scheme that (1) satisfies adaptive security; (2) achieves DKER; (3) realizes constant-size public parameters; and (4) is constructed over prime-order bilinear groups. Our core technique relies on Seo and Emura's one (PKC 2013), which transform the Waters IBE (EUROCRYPT 2005) to the corresponding RIBE scheme. Specifically, we construct an IBE scheme that satisfies constant-size public parameters over prime-order groups and some requirements for the Seo-Emura technique, and then transform the IBE scheme to an RIBE scheme. We also discuss how to extend the proposed RIBE scheme to a chosen-ciphertext secure one and server-aided one (ESORICS 2015).

1. Introduction

Identity-based encryption (IBE), which is public-key encryption (PKE) enabling one to use an arbitrary bit-string such as an e-mail address as their public key, was first introduced by Shamir [49], and the first efficient realization was proposed by Boneh and Franklin [5]. Since arbitrary strings can be used as public keys, IBE, unlike PKE, does not need certificates of public keys published by public-key infrastructures (PKIs). Although the absence of the certificates is one of the main advantages of IBE, it leads to another problem: how are malicious users *efficiently* revoked? For excluding malicious users and considering users withdrawal from a system, revocation functionality is indispensable if secure systems are launched in the real world. Actually, Boneh and Franklin only showed the following naïve revocation procedure: the lifetime of the

☆ A preliminary version appeared in the proceedings of Cryptograpers' Track on RSA conference 2017 (CT-RSA 2017) [52].

* Corresponding author at: Graduate School of Informatics and Engineering, The University of Electro-Communications, 1-5-1 Chofugaoka, Chofu, 182-8585, Japan.
E-mail address: watanabe@uec.ac.jp (Y. Watanabe).

https://doi.org/10.1016/j.tcs.2021.02.024

system is divided into (polynomially many) discrete time periods, a key generation center (KGC) regenerates secret keys for all of the non-revoked users at each time period. Specifically, for every identity $\mathtt{I}$ of non-revoked users at time period $\mathtt{T}$, the KGC generates a secret key for $\mathtt{I}\|\mathtt{T}$. This revocation procedure requires $O(\tilde{n})$ computational complexity of the KGC, where $\tilde{n}$ is the number of non-revoked users, and thus it is inefficient solution. There are many efficient IBE schemes (e.g., [54,7,13]), however such a revocation problem seems to become a barrier to practical use of IBE. To resolve the problem, Boldyreva et al. [2] proposed efficient revocation functionality by using the complete subtree (CS) method [33], which was originally used for broadcast encryption. They prepared two kinds of secret keys for each identity $\mathtt{I}$, a (long-term) secret key $sk_{\mathtt{I}}$ and (short-term) decryption keys $dk_{\mathtt{I},\mathtt{T}}$, which depend on not only $\mathtt{I}$ but time period $\mathtt{T}$. If $\mathtt{I}$ is not revoked at time period $\mathtt{T}$, the decryption key $dk_{\mathtt{I},\mathtt{T}}$ can be generated by the secret key $sk_{\mathtt{I}}$ and *key update* $ku_{\mathtt{T}}$, which is generated and broadcasted by the KGC at every $\mathtt{T}$. Key update $ku_{\mathtt{T}}$ is constructed by the CS method, and therefore the size of $ku_{\mathtt{T}}$ is $O(r\log(n/r))$, where n is the number of maximum users and r is the number of revoked users. IBE with such a scalable revocation method is called *revocable IBE (RIBE)*.

After the seminal work by Boldyreva et al. [2], several RIBE schemes have been proposed thus far. Almost all such subsequent works basically follow Boldyreva et al.'s revocation methodology. Libert and Vergnaud [30] proposed the first adaptively secure RIBE scheme. Seo and Emura defined an additional security notion to capture more realistic threats: *decryption key exposure resistance (DKER)* [42,44]. Intuitively, DKER is a security notion against adversaries who get "exposed" decryption keys of honest users. Namely, secure RIBE schemes with DKER guarantee security against not only maliciously revoked users but also such adversaries. DKER is important where the secret key is stored in physically secure devices such as USB pen drives to be isolated from the Internet but decryption keys are stored in weaker device such as smart phones. In fact, Boneh-Franklin's naïve solution satisfies DKER. Revocation functionality in RIBE should be efficient realization of the naïve solution, and in that sense, DKER is important. Seo and Emura [42,44] showed the first adaptively secure RIBE scheme with DKER.

Although several RIBE schemes with DKER have been proposed so far (e.g., [23,21,16,45]), there is still room for improvement in terms of efficiency. Specifically, with the exception of [21,23], there are no adaptively-secure RIBE schemes that simultaneously achieve DKER and constant-size public parameters.[1] As we know, the public-key size in usual PKI is important since it should be sent to an encryptor along with the recipient's public key in many applications. Although RIBE can remove a certificate of the recipient's public key (advantage of using IBE) and expensive revocation list of PKI (advantage of using 'R'IBE), an encryptor still needs global public parameters (as well as the recipient's ID) to encrypt a plaintext m. The encryptor needs to retrieve the public parameter from the KGC (or, receive it from the recipient) to encrypt m, and therefore minimizing the public-parameter size is important as well as the ciphertext size in the sense of communication complexity. Moreover, the public-parameter size affects the running time of all algorithms, especially an encryption algorithm. The only exceptions, RIBE schemes proposed in [21,23], are constructed over composite-order groups. Although a composite-order group has nice cryptographic features, cryptographic protocols in the composite-order setting usually require a large parameter in practice, compared to those in the prime-order setting. Furthermore, as reported in [14], there are several practical issues to generate pairing-friendly elliptic curves that contains a composite-order modulus. Therefore, it is important to realize an RIBE scheme that has constant-size public parameters over prime-order groups.

Thus, it is quite natural to ask:

Can we attain an adaptively secure efficient RIBE scheme with DKER from (relatively) simple assumptions?

In particular, we would like to ask:

Can we attain an RIBE scheme that (1) satisfies adaptive security under (relatively) simple assumptions; (2) achieves DKER; (3) realizes constant-size public parameter; and (4) is constructed over prime-order groups?

1.1. Our contribution

In this paper, we propose an RIBE scheme that achieves all the above properties (1)–(4). More specifically, we show the first adaptively secure RIBE scheme with DKER and constant-size public parameters in asymmetric bilinear groups of prime order. The security of our scheme is proved under static assumptions, which are mild variants of the symmetric external Diffie-Hellman (SXDH) assumption.

Difficulties in constructing RIBE with the above properties. Roughly speaking, only two construction methodologies for realizing constant-size public-parameter IBE are known: One from strong assumptions such as static ones in composite-order groups and q-type ones (e.g., [12,55]), and one from simple assumption via the dual system encryption methodology [54] in either prime-order or composite-order groups. Therefore, if we want to realize an RIBE scheme with constant-size public parameters under (relatively) simple and static assumptions in prime-order groups, it seems that we should apply the latter one.

[1] After submitting this paper in October, 2017, several subsequent works [9,11] proposed RIBE schemes that realize constant-size public parameters from standard assumptions while satisfying DKER. We here ignore them to make consistent with our original motivation and will list the above papers at Section 1.2.

However, in the dual system encryption framework, there exist difficulties achieving adaptively secure RIBE schemes with DKER and short parameters (in the sense of constant-size public parameters and prime-order groups). In fact, Lee observed such an obstacle [21], and pointed out a revocable hierarchical IBE (RHIBE) scheme in [46] has some security flaw. Let us briefly review such an obstacle. In the dual system encryption framework, ciphertexts and secret keys have two forms: Normal and semi-functional ones. Normal secret keys can decrypt not only normal but semi-functional ciphertexts, whereas semi-functional secret keys only decrypt normal ciphertexts. In the security proof, a normal challenge ciphertext and secret keys are transformed into their semi-functional forms one by one. In the process of changing some normal secret key, called *a target key*, into its semi-functional form, some pairwise independent function f has to be embedded into public parameters at the beginning of the transition. Randomness for the challenge ciphertext and the target key is computed by f. Specifically, the simulator generates randomness $r_C := f(\mathtt{I}^*)$ for the challenge ciphertext, as well as randomness $r_K := f(\mathtt{I})$ for the target key, where $\mathtt{I}^*$ is the target identity and $\mathtt{I}$ is an identity such that $\mathtt{I} \neq \mathtt{I}^*$. Since $\mathtt{I} \neq \mathtt{I}^*$, i.e., r_C is independent of r_K from an adversarial view in the information-theoretic sense, the proof works well. To the best of our knowledge, such a pairwise independent function f is necessary for proving security of all of the currently-known IBE schemes using the dual system encryption methodology. On the other hand, in the RIBE setting, the adversary can get not only a challenge ciphertext for $\mathtt{I}^*$ but a secret key for $\mathtt{I}^*$ for handling the case that the target user is revoked before the target time period (see Definition 1). Therefore, if a target key is a secret key for $\mathtt{I}^*$ and the adversary issues it to a secret key generation oracle, then randomness r_C for the challenge ciphertext and randomness r_K for the target key are no longer independent of each other from the view point of the adversary, since it holds $r_C = r_K = f(\mathtt{I}^*)$.

Lee [21] and Lee et al. [23] overcame the above obstacle by carefully designing hybrid games, and proposed adaptively secure R(H)IBE schemes with DKER via the dual system encryption methodology. Indeed, their schemes achieve adaptive security, DKER, and short public parameters (i.e., the desired properties (1)–(3)), however it is constructed over composite-order bilinear groups. Although other RHIBE schemes have been proposed [8,18,19,22,27,31,32,39,41,43,47,48,51], none of them satisfies all the properties (1)–(4).[2]

Our approach. We overcome the difficulty mentioned above by taking the Seo-Emura approach [42]. Seo and Emura proposed an adaptively secure RIBE scheme based on the Waters IBE [53], and showed a security reduction from the Waters IBE to their RIBE scheme. Namely, the approach does not depend on the proof methodology of the underlying Waters IBE scheme for proving security of their RIBE scheme. Therefore, the Seo-Emura technique is promising approach to avoiding the randomness correlation problem, which is specific to dual-system-encryption-based RIBE schemes. More precisely, our approach is three-fold: We first find (or construct) a "basic" IBE scheme that realizes constant-size public parameters via the dual system encryption; construct an RIBE scheme with constant-size public parameters in bilinear groups of prime order based on the basic IBE scheme; and prove the adaptive security with DKER by showing a security reduction from the basic IBE scheme to the constructed RIBE scheme.

However, the approach still has problems. Although the Seo-Emura technique essentially requires *the secret-key re-randomization*[3] of the underlying IBE scheme, unfortunately, almost all of the dual system encryption-based IBE schemes (e.g., [54,29,6]) do not have this property. Moreover, the Boneh-Boyen technique [3] is essential for the security proof taking the Seo-Emura approach. Therefore, the basic IBE scheme must satisfy (i) the secret-key re-randomization property and (ii) applicability of the Boneh-Boyen technique (for details, see Section 3).

Based on the above discussion, we employ the Jutla-Roy IBE [17] (and its variant [38]), which satisfies the property (i), as a promising candidate of our basic IBE scheme. However, it does not satisfy the property (ii). Concretely speaking, the Boneh-Boyen technique requires some group elements that contain the master key in the exponent in the public parameter of the underlying IBE, however the original Jutla-Roy IBE does not contain them. Hence, we modify the Jutla-Roy IBE so that it satisfies the property (ii), and we prove the security under the Augmented Decisional Diffie-Hellman on $\mathbb{G}_1$ (ADDH1), which is a static assumption newly introduced in this paper, and Decisional Diffie-Hellman on $\mathbb{G}_2$ (DDH2) assumptions. The ADDH1 assumption is similar to the previously used assumption in [36], and the security is proved in the generic bilinear group model.

We then construct an RIBE scheme based on the Jutla-Roy IBE, and show a security reduction from the modified Jutla-Roy IBE to the RIBE scheme. As a result, we obtain the first RIBE scheme that achieves the properties (1)–(4).

Efficiency comparison. We compare the efficiency among our scheme, the Seo-Emura scheme [42,44], the Lee scheme [21], and the Lee-Lee-Park (LLP) scheme [23] in Table 1. All the schemes meet adaptive security with DKER. Due to the underlying CS method, the sizes of secret keys and key updates in every scheme except for the LLP RIBE are $O(\log n)$ and $O(r \log(n/r))$, respectively. On the other hand, the LLP scheme achieves $O(\log^{1.5} n)$ secret-key sizes and $O(r)$ key-update sizes due to the underlying layered subset difference (LSD) method [15]. The Lee scheme and LLP scheme achieve asymptotically compact schemes in the sense that many of parameters consist of a smaller number of group elements. However, those schemes are based on the composite-order bilinear groups, where each group element requires much larger space to describe than that of in the prime-order bilinear group in practice for the same security level. We provide the detailed comparison for the 128-

[2] Though RHIBE schemes in [18,19,22,31,32,51] were proposed after the initial submission of this paper, they do not meet at least one desired property.
[3] It means that each secret key can be re-randomized with fresh randomness.

Table 1
Efficiency Comparison among adaptively secure RIBE schemes with DKER.
Let $|\mathbb{G}_1|$, $|\mathbb{G}_2|$, and $|\mathbb{G}_T^{\mathsf{asym}}|$ be the bit-length of an element of asymmetric bilinear groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ respectively. Let $|\mathbb{G}_p|$ and $|\mathbb{G}_T^{\mathsf{sym}}|$ be the bit-length of an element of symmetric bilinear groups $\mathbb{G}_p$ and $\mathbb{G}_T$ employed in [42,44], respectively. Let $|\mathbb{G}_N|$ and $|\mathbb{G}_T^{\mathsf{comp}}|$ be the bit-length of an element of symmetric bilinear groups $\mathbb{G}_N$ and $\mathbb{G}_T$ of composite order $N = p_1p_2p_3$, where p_1, p_2, and p_3 are distinct prime numbers, employed in [21], respectively. Let $|\mathbb{Z}_p|$ and $|\mathbb{Z}_N|$ be the bit-length of an element of $\mathbb{Z}_p$ and $\mathbb{Z}_N$, respectively. On 256-bit Barreto-Naehrig curve [1], $|\mathbb{G}_1| = 256$, $|\mathbb{G}_2| = |\mathbb{G}_p| = 512$, and $|\mathbb{G}_T^{\mathsf{asym}}| = |\mathbb{G}_T^{\mathsf{sym}}| = 3072$ due to [6]. Note that $|\mathbb{G}_N|$ and $|\mathbb{G}_T^{\mathsf{comp}}|$ should be much larger so that N cannot be factored. L is the hierarchy depth, n is the maximum number of users, r is the number of revoked users, and ℓ is the bit-length of identity. For example, if 32 byte e-mail address is regarded as identity, then $\ell = 256$.

Scheme	*#mpk*	*#msk*	*#C*	*#sk*
Seo-Emura [42,44]	$(6+\ell)\|\mathbb{G}_p\|$	$\|\mathbb{G}_p\|$	$3\|\mathbb{G}_p\| + \|\mathbb{G}_T^{\mathsf{sym}}\|$	$(2\log n)\|\mathbb{G}_p\|$
Lee [21] ($L = 1$)	$8\|\mathbb{G}_N\| + 3\|\mathbb{G}_T^{\mathsf{comp}}\|$	$\|\mathbb{G}_N\|$	$4\|\mathbb{G}_N\| + \|\mathbb{G}_T^{\mathsf{comp}}\|$	$(2\log n)\|\mathbb{G}_N\|$
LLP [23]	$6\|\mathbb{G}_N\| + 1\|\mathbb{G}_T^{\mathsf{comp}}\|$	$\|\mathbb{G}_N\|$	$4\|\mathbb{G}_N\|$	$(\log^{1.5} n)\|\mathbb{G}_N\|$
Proposed scheme	$7\|\mathbb{G}_1\| + 11\|\mathbb{G}_2\| + \|\mathbb{G}_T^{\mathsf{asym}}\|$	$2\|\mathbb{G}_2\|$	$4\|\mathbb{G}_1\| + \|\mathbb{G}_T^{\mathsf{asym}}\| + \|\mathbb{Z}_p\|$	$(5\log n)\|\mathbb{G}_2\|$

Scheme	*#ku*	*#dk*	Symmetric or Asymmetric	Prime or Composite	Assumption
Seo-Emura [42,44]	$(2r\log(n/r))\|\mathbb{G}_p\|$	$3\|\mathbb{G}_p\|$	Symmetric	Prime	DBDH
Lee [21] ($L = 1$)	$(2r\log(n/r))\|\mathbb{G}_N\| + 2\|\mathbb{Z}_N\|$	$4\|\mathbb{G}_N\|$	Symmetric	Composite	Static
LLP [23]	$4r\|\mathbb{G}_N\|$	$3\|\mathbb{G}_N\|$	Symmetric	Composite	Static
Proposed scheme	$(3r\log(n/r))\|\mathbb{G}_2\|$	$6\|\mathbb{G}_2\|$	Asymmetric	Prime	ADDH1 & DDH2

bit security in the caption of Table 1. Our scheme is more efficient than the Seo-Emura RIBE in terms of constant-size public parameters and asymmetric pairings, and other parameters are comparable to those of the Seo-Emura RIBE. In addition, our proof technique provides a better reduction loss than that of the Seo-Emura RIBE. More precisely, the reduction loss of our scheme is $O(q_1 q|\mathcal{T}|)$, whereas that of the Seo-Emura RIBE is $O(\ell q^2|\mathcal{T}|)$, where ℓ is the bit-length of identity, q is the maximum number of queries in the security game, q_1 is the maximum number of queries *before the challenge phase* in the security game, and $|\mathcal{T}|$ is the number of time periods in the schemes.

Extension to CCA-secure RIBE scheme. Roughly speaking, the Seo-Emura technique is originally a kind of combination technique of two specific IBE schemes, the Waters IBE [53] and Boneh-Boyen IBE [3], to obtain RIBE schemes. Ishida et al. [16] focused on the underlying Waters IBE in the Seo-Emura technique, and applied a transformation from a 2-level CPA-secure HIBE scheme to a CCA-secure IBE scheme (called *the BCHK transformation* [4]) to the underlying Waters IBE by extending it into a 2-level scheme. We can take a similar approach to our RIBE scheme (and hence the underlying modified Jutla-Roy IBE). As a result, we obtain the first CCA-secure RIBE scheme with short public parameters in prime-order groups. For details, see Section 5.1.

Extension to server-aided RIBE scheme. Basically, we use the Seo-Emura technique for enhancing both security and efficiency at the same time. There is another nice feature of using the Seo-Emura technique such that the technique well harmonizes with variants of IBE schemes since it is a kind of transformation technique from IBE to RIBE. In fact, Qin et al. [35] have already used the Seo-Emura technique to construct a variant scheme so-called Server-Aided RIBE (SRIBE) that reduces a burden on users with aids of untrusted server. Qin et al.'s scheme is built on the base of the Seo-Emura RIBE so that it still has long public parameters. We find that our scheme can be easily transformed to a server-aided scheme and satisfies all the security requirements introduced by Qin et al. That is, we achieve the first SRIBE scheme with short public parameter. The detailed construction is given in Section 5.2.

Differences from the conference version [52]. This paper is an extended version of the work published in [52]. In particular, this version corrects some minor bugs, clarifies some explanations, and contains full details of security proofs including the generic hardness of a new assumption, which are only sketched in the conference version. We additionally discuss how to extend the proposed scheme to a chosen-ciphertext secure one and a server-aided one in Section 5.

1.2. Other approaches and subsequent work

Other approaches towards RIBE with all the desired properties. As described earlier, the Lee RIBE [21] and the LLP RIBE [23] achieve the desired property (1)–(3). One might come up with an approach for achieving (4): applying the conversion [29] that converts cryptosystems constructed over composite-order groups into one constructed over prime-order groups. Since their RIBE schemes seem to satisfy applicable conditions for the conversion, we expect to get an RIBE scheme that meets all the properties via the approach. However, the conversion is not very efficient and the resultant RIBE scheme would be less efficient than our scheme (in the concrete sense).

Another approach is to devise dual-system-encryption-based IBE schemes that have the secret-key rerandomization property. Lee et al. [24–26] proposed digital signature and sequential aggregate signature schemes based on the Lewko-Waters IBE [28], and the underlying IBE scheme provides the key rerandomization, though the IBE itself was not explicitly described. Therefore, we might be able to construct an RIBE scheme that satisfies the properties (1)–(4). Note that the resulting RIBE

scheme would require non-standard but static complexity assumptions called LW1 and LW2, which were introduced in [28], as in our scheme relying on the ADDH1 assumption.

Subsequent works. After initial submission of this paper, several subsequent papers on RIBE, especially on RIBE schemes that achieve all the desired properties, were published. Ge and Wei [11] proposed an efficient revocable identity-based broadcast encryption (RIBBE) scheme, which includes an RIBE scheme with the properties (1)–(4) as a special case. Their RIBBE scheme is indeed based on our RIBE scheme, and therefore has the similar efficiency when we see it as RIBE. Very recently, Emura et al. [9] showed an adaptively-secure RHIBE scheme with DKER from the k-Lin assumption, and it implies a new adaptively-secure RIBE scheme with all the properties without relying on the non-standard assumption.

1.3. Paper organization

In Section 2, we describe notation and definitions throughout this paper. In Section 3, we propose an IBE scheme, which is used as the underlying IBE scheme of our RIBE scheme, based on the Jutla-Roy IBE. In Section 4, we show the first adaptively secure RIBE scheme with DKER and short public parameters in prime-order groups, and some extensions are discussed in Section 5. We give the proof of ADDH1 assumption in the generic bilinear group model in Section 6. We finally conclude in Section 7.

2. Preliminaries

Notation. In this paper, "probabilistic polynomial-time" is abbreviated as "PPT". For a prime p, let $\mathbb{Z}_p := \{0, 1, \ldots, p-1\}$ and $\mathbb{Z}_p^\times := \mathbb{Z}_p \setminus \{0\}$. If we write $(y_1, y_2, \ldots, y_m) \leftarrow \mathcal{A}(x_1, x_2, \ldots, x_n)$ for an algorithm $\mathcal{A}$ having n inputs and m outputs, it means to input $x_1, x_2, \ldots, x_n$ into $\mathcal{A}$ and to get the resulting output $y_1, y_2, \ldots, y_m$. We write $(y_1, y_2, \ldots, y_m) \leftarrow \mathcal{A}^{\mathcal{O}}(x_1, x_2, \ldots, x_n)$ to indicate that an algorithm $\mathcal{A}$ that is allowed to access an oracle $\mathcal{O}$ takes $x_1, x_2, \ldots, x_n$ as input and outputs $(y_1, y_2, \ldots, y_m)$. If $\mathcal{X}$ is a set, we write $x \xleftarrow{\$} \mathcal{X}$ to mean the operation of picking an element x of $\mathcal{X}$ uniformly at random. We use λ as a security parameter. $\mathcal{M}$, $\mathcal{I}$, and $\mathcal{T}$ denote sets of plaintexts, IDs, and time periods, respectively, which are determined by the security parameter λ.

Bilinear groups. A bilinear group generator $\mathcal{G}$ is an algorithm that takes a security parameter λ as input and outputs a bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$, where p is a prime, $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are multiplicative cyclic groups of order p, g_1 and g_2 are (random) generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively, and e is an efficiently computable and non-degenerate bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ with the following bilinear property: For any $u, u' \in \mathbb{G}_1$ and $v, v' \in \mathbb{G}_2$, $e(uu', v) = e(u, v)e(u', v)$ and $e(u, vv') = e(u, v)e(u, v')$.

A bilinear map e is called symmetric or a "Type-1" pairing if $\mathbb{G}_1 = \mathbb{G}_2$. Otherwise, it is called asymmetric. In the asymmetric setting, e is called a "Type-2" pairing if there is an efficiently computable isomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$. If no efficiently computable isomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$ is known, then it is called a "Type-3" pairing. Throughout this paper, we focus on the Type-3 pairing. Type-3 is the most efficient setting since compared to Type-1, the size of representation of $\mathbb{G}_1$ in the Type-3 setting is smaller and whole operations in the Type-3 setting are more efficient; and compared to Type-2, the size of representation of $\mathbb{G}_2$ in the Type-3 setting is smaller and group operations in $\mathbb{G}_2$ in the Type-3 are more efficient. For details, see [10].

KUNode algorithm. To reduce costs of a revocation process, we use a binary tree structure and apply the following KUNode algorithm as in the previous RIBE schemes [2,30,42]. $\mathsf{KUNode}(\mathtt{BT}, RL, \mathtt{T})$ takes as input a binary tree $\mathtt{BT}$, a revocation list RL, and a time period $\mathtt{T} \in \mathcal{T}$, and outputs a set of nodes. When η is a non-leaf node, then we write η_L and η_R as the left and right child of η, respectively. When η is a leaf node, $\mathsf{Path}(\mathtt{BT}, \eta)$ denotes the set of nodes on the path from η to the *root*. Each user is assigned to a leaf node. If a user who is assigned to η is revoked on a time period $\mathtt{T} \in \mathcal{T}$, then $(\eta, \mathtt{T}) \in RL$. $\mathsf{KUNode}(\mathtt{BT}, RL, \mathtt{T})$ is executed as follows. It sets $\mathcal{X} := \emptyset$ and $\mathcal{Y} := \emptyset$. For any $(\eta_i, \mathtt{T}_i) \in RL$, if $\mathtt{T}_i \leq \mathtt{T}$ then it adds $\mathsf{Path}(\mathtt{BT}, \eta_i)$ to $\mathcal{X}$ (i.e., $\mathcal{X} := \mathcal{X} \cup \mathsf{Path}(\mathtt{BT}, \eta_i)$). That is, KUNode adds at most $r \log n$ nodes to $\mathcal{X}$ where $r = |RL|$ and n is the number of leaves of $\mathtt{BT}$. Then, for any $\eta \in \mathcal{X}$, if $\eta_L \notin \mathcal{X}$, then it adds η_L to $\mathcal{Y}$. If $\eta_R \notin \mathcal{X}$, then it adds η_R to $\mathcal{Y}$. That is, KUNode adds at most $r \log n$ nodes to $\mathcal{Y}$. Actually, due to the result of [33], the size of $\mathcal{Y}$ is $O(r \log(n/r))$, and the time complexity is $O(\log \log n)$. Finally, it outputs $\mathcal{Y}$ if $\mathcal{Y} \neq \emptyset$. If $\mathcal{Y} = \emptyset$, then it adds *root* to $\mathcal{Y}$ and outputs $\mathcal{Y}$.

Revocable identity-based encryption. An RIBE scheme Π consists of seven-tuple algorithms (Setup, SKGen, KeyUp, DKGen, Enc, Dec, Revoke) defined as follows: For simplicity, we omit a public parameter in the input of all algorithms except for the Setup algorithm.

- $(mpk, msk, RL, st) \leftarrow \mathsf{Setup}(\lambda, N)$: A probabilistic algorithm for setup. It takes a security parameter λ and the maximum number of users N as input and outputs a public parameter mpk, a master secret key msk, an initial revocation list $RL = \emptyset$ and a state st.

- $(sk_{\mathtt{I}}, st) \leftarrow \mathsf{SKGen}(st, \mathtt{I})$: An algorithm for private key generation. It takes st and an identity $\mathtt{I} \in \mathcal{I}$ as input and outputs a secret key $sk_{\mathtt{I}}$ and updated state information st.[4]
- $ku_{\mathtt{T}} \leftarrow \mathsf{KeyUp}(msk, st, RL, \mathtt{T})$: An algorithm for key update generation. It takes msk, state st, a current revocation list RL, and a time period $\mathtt{T}$ as input, and then outputs key update $ku_{\mathtt{T}}$.
- $dk_{\mathtt{I},\mathtt{T}}$ or $\perp \leftarrow \mathsf{DKGen}(sk_{\mathtt{I}}, ku_{\mathtt{T}})$: A probabilistic algorithm for decryption key generation. It takes $sk_{\mathtt{I}}$ and $ku_{\mathtt{T}}$ as input and then outputs a decryption key $dk_{\mathtt{I},\mathtt{T}}$ at $\mathtt{T}$ or $\perp$ if $\mathtt{I}$ has been revoked by $\mathtt{T}$.
- $C_{\mathtt{I},\mathtt{T}} \leftarrow \mathsf{Enc}(M, \mathtt{I}, \mathtt{T})$: A probabilistic algorithm for encryption. It takes $M \in \mathcal{M}$, $\mathtt{I} \in \mathcal{I}$, and $\mathtt{T} \in \mathcal{T}$ as input and then outputs a ciphertext $C_{\mathtt{I},\mathtt{T}}$.
- M or $\perp \leftarrow \mathsf{Dec}(dk_{\mathtt{I},\mathtt{T}}, C_{\mathtt{I},\mathtt{T}})$: A deterministic algorithm for decryption. It takes $dk_{\mathtt{I},\mathtt{T}}$ and $C_{\mathtt{I},\mathtt{T}}$ as input and then outputs M or $\perp$.
- $RL \leftarrow \mathsf{Revoke}(\mathtt{I}, \mathtt{T}, RL, st)$: An algorithm for revocation. It takes $(\mathtt{I}, \mathtt{T}) \in \mathcal{I} \times \mathcal{T}$, the current revocation list RL, and a state st as input and then outputs an updated revocation list RL.

In the above model, we require that Π meets the following correctness property: For all security parameter $\lambda \in \mathbb{N}$, all $(mpk, msk, RL, st) \leftarrow \mathsf{Setup}(\lambda, N)$, all $M \in \mathcal{M}$, all $\mathtt{I} \in \mathcal{I}$, all $\mathtt{T} \in \mathcal{T}$, if $\mathtt{I}$ is not revoked on $\mathtt{T} \in \mathcal{T}$, it holds that $M = \mathsf{Dec}(dk_{\mathtt{I},\mathtt{T}}, \mathsf{Enc}(M, \mathtt{I}, \mathtt{T}))$, where $dk_{\mathtt{I},\mathtt{T}} \leftarrow \mathsf{DKGen}(\mathsf{SKGen}(st, \mathtt{I}), \mathsf{KeyUp}(msk, st, RL, \mathtt{T}))$.

We describe the notion of indistinguishability against chosen plaintext attack (IND-RID-CPA). Note that this notion also captures DKER, which was introduced by Seo and Emura [42], and this security model is the strongest known one. Let $\mathcal{A}$ be a PPT adversary, and $\mathcal{A}$'s advantage against IND-RID-CPA security is defined by

$$\mathsf{Adv}^{\mathsf{RID\text{-}CPA}}_{\Pi,\mathcal{A}}(\lambda, N) := \left| \Pr\left[b' = b \;\middle|\; \begin{array}{l} (mpk, msk, RL, st) \leftarrow \mathsf{Setup}(\lambda, N), \\ (M_0^*, M_1^*, \mathtt{I}^*, \mathtt{T}^*, state) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{find}, mpk), \\ b \xleftarrow{\$} \{0,1\}, \\ C^*_{\mathtt{I}^*,\mathtt{T}^*} \leftarrow \mathsf{Enc}(M_b^*, \mathtt{I}^*, \mathtt{T}^*), \\ b' \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{guess}, C^*_{\mathtt{I}^*,\mathtt{T}^*}, state) \end{array} \right] - \frac{1}{2} \right|.$$

Here, $\mathcal{O}$ is a set of oracles $\{SKGen(\cdot), KeyUp(\cdot), Revoke(\cdot,\cdot), DKGen(\cdot,\cdot)\}$ defined as follows.

SKGen$(\cdot)$**:** For a query $\mathtt{I} \in \mathcal{I}$, it stores and returns $\mathsf{SKGen}(st, \mathtt{I})$.
KeyUp$(\cdot)$**:** For a query $\mathtt{T} \in \mathcal{T}$, it stores and returns $\mathsf{KeyUp}(msk, RL, st, \mathtt{T})$.
Revoke$(\cdot,\cdot)$**:** For a query $(\mathtt{I}, \mathtt{T}) \in \mathcal{I} \times \mathcal{T}$, it updates a revocation list RL by running $\mathsf{Revoke}(\mathtt{I}, \mathtt{T}, RL, st)$.
DKGen$(\cdot,\cdot)$**:** For a query $(\mathtt{I}, \mathtt{T}) \in \mathcal{I} \times \mathcal{T}$, it finds $sk_{\mathtt{I}}$ and $ku_{\mathtt{T}}$ generated by the *SKGen* and *KeyUp* oracles, respectively (if $sk_{\mathtt{I}}$ has not been generated yet, *DKGen* executes $(sk_{\mathtt{I}}, st) \leftarrow \mathsf{SKGen}(st, \mathtt{I})$).[5] *DKGen* returns $\mathsf{DKGen}(sk_{\mathtt{I}}, ku_{\mathtt{T}})$ and stores it unless it is $\perp$.

The above oracles represent the following realistic threats and situations: *SKGen* represents the collusion among users as in ordinary IBE. $\mathcal{A}$ can access *KeyUp* since key updates are broadcasted by the KGC. The reason why $\mathcal{A}$ can access *Revoke* is an RIBE scheme should be secure against any situations in terms of the revocation list. *DKGen* represents decryption key exposure.

We then impose the following restrictions on $\mathcal{A}$. Specifically, the first three restrictions are placed to take into account practical situations, and we circumvent some trivial attacks by the other restrictions.

1. *KeyUp*$(\cdot)$ and *Revoke*$(\cdot,\cdot)$ can be queried at a time period which is later than or equal to that of all previous queries.
2. *Revoke*$(\cdot,\cdot)$ cannot be queried at a time period $\mathtt{T}$ after issuing $\mathtt{T}$ to $\mathsf{KeyUp}(\cdot)$.
3. *DKGen*$(\cdot,\cdot)$ cannot be queried at $\mathtt{T}$ before issuing $\mathtt{T}$ to $\mathsf{KeyUp}(\cdot)$.
4. $\mathcal{A}$ is allowed to issue $\mathtt{I}^*$ before $\mathtt{T}^*$. However, if $\mathtt{I}^*$ was issued to *SKGen*$(\cdot)$ at $\mathtt{T}' \leq \mathtt{T}^*$, then $(\mathtt{I}^*, \mathtt{T})$ must be issued to $\mathsf{Revoke}(\cdot,\cdot)$ such that $\mathtt{T}' \leq \mathtt{T} \leq \mathtt{T}^*$.
5. $(\mathtt{I}^*, \mathtt{T}^*)$ cannot be issued to *DKGen*$(\cdot,\cdot)$.

Definition 1 *(*IND-RID-CPA*).* An RIBE scheme Π is said to be IND-RID-CPA secure if for all PPT adversaries $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{RID\text{-}CPA}}_{\Pi,\mathcal{A}}(\lambda, N)$ is negligible in λ.

3. The basic IBE scheme

We begin with reviewing Seo and Emura's approach for transforming IBE to RIBE [42]. Although their approach is not generic, it seems quite broadly applicable to the other IBE schemes. We find some requirements for applying their technique.

[4] We consider the SKGen algorithm in the sense of history-free RHIBE [47,48], i.e., the algorithm takes st, rather than msk, as input.
[5] Contrary to $sk_{\mathtt{I}}$, $ku_{\mathtt{T}}$ is already stored by the *KeyUp* oracle due to the restrictions on the oracles.

Then, we propose an IBE scheme satisfying such the requirements, which has short public parameters and over prime-order bilinear groups.

Seo and Emura constructed an RIBE scheme based on the Waters IBE [53] and provided a security reduction to the Waters IBE. In the reduction, almost all queries can be easily simulated due to the adaptive security of the underlying Waters IBE. The most non-trivial part in the reduction is simulating decryption keys for $(\mathtt{I}^*, \mathtt{T})$, where $\mathtt{I}^*$ is the target identity, since the security of usual IBE schemes does not handle this case related to $\mathtt{I}^*$. To this end, Seo and Emura employed two techniques; the Boneh-Boyen technique [3] and secret-key re-randomization.

The Boneh-Boyen technique is originally for *selectively secure* scheme[6]; that is, if the simulator knows the target (time $\mathtt{T}^*$ in our case) in advance, then the simulator embeds it into public parameters so that the simulator can simulate all the other queries not related to $\mathtt{T}^*$.[7] The Boneh-Boyen technique enables the simulator to compute decryption keys for $(\mathtt{I}^*, \mathtt{T})$ with biased distribution, where $\mathtt{T}$ is not the target time. The secret-key re-randomization can resolve the biased distribution by forcing that all decryption keys have uniform randomness.

From the above interpretation, we find two requirements for the input IBE; (1) the secret-key re-randomization property and (2) applicability of the Boneh-Boyen technique. The latter requirement can be further segmentalized. (2-1) Each component of a secret key contains at most one component of a master key and (2-2) each component of the master-key is available in the public parameters in some form of elements in source-groups (of bilinear groups). The former is due to that the Boneh-Boyen technique can extract at most one master-key component from each secret-key component. The latter is due to that in the security reduction the master-key is embedded into key updates that consist of only elements in source-groups by using the master-key-related public parameters.[8]

The Waters IBE satisfies all the above requirements, but most of dual-system-encryption-based IBE schemes in prime-order groups do not. For example, the first scheme by Waters [54] and almost all of the IBE schemes using dual pairing vector spaces (DPVS) (e.g., [29,6]) do not satisfy any requirement, in particular, the public re-randomization requirement.

3.1. Modified Jutla-Roy IBE

We employ a modified version of the Jutla-Roy IBE [17] (and its variant [38]). The original scheme satisfies two requirements (1) and (2-1). In this subsection, we modify the Jutla-Roy IBE to additionally satisfy the requirement (2-2).

The master key of the Jutla-Roy IBE is $(y_0, x_0) \in \mathbb{Z}_p^2$. To get a basic IBE scheme for our RIBE scheme based on the Jutla-Roy IBE, we add the master key in the forms of elements in $\mathbb{G}_1$ and $\mathbb{G}_2$ with a random mask $\beta \in \mathbb{Z}_p^\times$, respectively, to the public parameters. Specifically, we add four group elements $(\chi_1 := g_1^{\beta(-x_0\alpha+y_0)}, g_2^{x_0\beta}, g_2^{y_0\beta}, g_2^{1/\beta})$ to the original public parameter. However, we then cannot apply the original security proof of the Jutla-Roy IBE, and so we add a new twist to the proof. The modified Jutla-Roy IBE $\Pi_{\mathrm{JR}} = (\mathsf{Init}, \mathsf{KeyGen}, \mathsf{IBEnc}, \mathsf{IBDec})$ is constructed as follows.[9]

- $\mathsf{Init}(\lambda)$: It runs $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e) \leftarrow \mathcal{G}$. It chooses $x_0, y_0, x_1, y_1, x_2, y_2, x_3, y_3 \xleftarrow{\$} \mathbb{Z}_p$ and $\alpha, \beta \xleftarrow{\$} \mathbb{Z}_p^\times$, and sets

$$z := e(g_1, g_2)^{-x_0\alpha+y_0},\ \ u_1 := g_1^{-x_1\alpha+y_1},\ \ w_1 := g_1^{-x_2\alpha+y_2},\ \ h_1 := g_1^{-x_3\alpha+y_3},\ \ \chi_1 := g_1^{\beta(-x_0\alpha+y_0)}.$$

  It outputs

$$PP := (g_1, g_1^\alpha, u_1, w_1, h_1, \chi_1, g_2, g_2^{x_1}, g_2^{x_2}, g_2^{x_3}, g_2^{y_1}, g_2^{y_2}, g_2^{y_3}, z, g_2^{x_0\beta}, g_2^{y_0\beta}, g_2^{\frac{1}{\beta}}),$$
$$MK := (g_2^{y_0}, g_2^{-x_0}).$$

- $\mathsf{KeyGen}(PP, MK, \mathtt{I})$: Parse MK as (d_1', d_2'). It chooses $r \xleftarrow{\$} \mathbb{Z}_p$ and computes

$$D_1 := (g_2^{y_2})^r,\ \ D_1' := d_1'\left((g_2^{y_1})^{\mathtt{I}} g_2^{y_3}\right)^r,\ \ D_2 := (g_2^{x_2})^{-r},\ \ D_2' := d_2'\left((g_2^{x_1})^{\mathtt{I}} g_2^{x_3}\right)^{-r},\ \ D_3 := g_2^r.$$

  It outputs $SK_{\mathtt{I}} := (D_1, D_1', D_2, D_2', D_3)$.

- $\mathsf{IBEnc}(PP, \mathtt{I}, M)$: It chooses $t, \mathtt{tag} \xleftarrow{\$} \mathbb{Z}_p$. For $M \in \mathbb{G}_T$, it computes

$$C_0 := Mz^t,\ \ C_1 := g_1^t,\ \ C_2 := (g_1^\alpha)^t,\ \ C_3 := \left(u_1^{\mathtt{I}} w_1^{\mathtt{tag}} h_1\right)^t.$$

  It outputs $C := (C_0, C_1, C_2, C_3, \mathtt{tag})$.

[6] Although our goal is adaptive security, the polynomial reduction loss enables one to use the selective security technique in terms of (polynomial-size) time period.

[7] Although the decryption key $(\mathtt{I}^*, \mathtt{T})$ is related to the target identity $\mathtt{I}^*$, it is not related to $\mathtt{T}^*$ so that the Boneh-Boyen technique is applicable.

[8] In (usual-but-not-all) pairing-based IBE schemes, private keys consist of elements in source-groups. Since both key updates and secret keys of RIBE are materials for decryption keys, they also should consist of source-group elements.

[9] The syntax of IBE is given in Appendix A.

– $\mathsf{IBDec}(PP, SK_{\mathtt{I}}, C)$: Parse $SK_{\mathtt{I}}$ and C as $(D_1, D_1', D_2, D_2', D_3)$ and $(C_0, C_1, C_2, C_3, \mathtt{tag})$, respectively. It computes

$$M = \frac{C_0 e(C_3, D_3)}{e(C_1, D_1^{\mathtt{tag}} D_1') e(C_2, D_2^{\mathtt{tag}} D_2')}.$$

We show the correctness of Π_{JR}. Suppose that $sk_{\mathtt{I}} = (D_1, D_1', D_2, D_2', D_3)$ and $C = (C_0, C_1, C_2, C_3,)$ are correctly generated. Then, we have

$$\begin{aligned}\frac{C_0 e(C_3, D_3)}{e(C_1, D_1^{\mathtt{tag}} D_1') e(C_2, D_2^{\mathtt{tag}} D_2')} &= \frac{M e(g_1, g_2)^{(-x_0\alpha + y_0)t}}{e(g_1^t, g_2^{y_2 r\mathtt{tag} + y_0 + r(y_1 \mathtt{I} + y_3)})} \cdot \frac{e(g_1^{t(\mathtt{I}(-x_1\alpha + y_1) + \mathtt{tag}(-x_2\alpha + y_2) - x_3\alpha + y_3)}, g_2^r)}{e(g_1^{\alpha t}, g_2^{-x_2 r\mathtt{tag} - x_0 - r(x_1 \mathtt{I} + x_3)})} \\ &= \frac{M e(g_1, g_2)^{(-x_0\alpha + y_0)t}}{e(g_1^t, g_2^{y_0}) e(g_1^{\alpha t}, g_2^{-x_0})} = M.\end{aligned}$$

3.2. Proof of security

We describe complexity assumptions used for proving the security proof of the modified Jutla-Roy IBE.

First, we give the definition of the decisional Diffie-Hellman (DDH) assumption in $\mathbb{G}_1$ and $\mathbb{G}_2$, which are called the DDH1 and DDH2 assumptions, respectively. We say that the SXDH assumption holds if both the DDH1 and DDH2 assumptions hold. Let $\mathcal{A}$ be a PPT adversary and we consider $\mathcal{A}$'s advantage against the DDHi problem $(i = 1, 2)$ as follows.

$$\mathsf{Adv}^{\mathsf{DDH}i}_{\mathcal{G},\mathcal{A}}(\lambda) := \left| \Pr\left[b' = b \;\middle|\; \begin{array}{l} D := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e) \leftarrow \mathcal{G}, \\ c_1, c_2 \xleftarrow{\$} \mathbb{Z}_p,\ b \xleftarrow{\$} \{0,1\}, \\ \text{if } b = 0 \text{ then } Z := g_i^{c_1 c_2},\ \text{else } Z \xleftarrow{\$} \mathbb{G}_i, \\ b' \leftarrow \mathcal{A}(\lambda, D, g_i^{c_1}, g_i^{c_2}, Z) \end{array} \right] - \frac{1}{2} \right|.$$

Definition 2 *(*DDHi *assumption).* The DDHi assumption relative to a generator $\mathcal{G}$ holds if for all PPT adversaries $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{DDH}i}_{\mathcal{G},\mathcal{A}}(\lambda)$ is negligible in λ.

Definition 3 *(*SXDH *assumption).* We say that the symmetric external Diffie-Hellman (SXDH) assumption relative to a generator $\mathcal{G}$ holds if both the DDH1 and DDH2 assumptions relative to $\mathcal{G}$ hold.

We then introduce a new assumption based on the DDH1 assumption, which is called *augmented decisional Diffie-Hellman assumption in* $\mathbb{G}_1$ (ADDH1 for short). Let $\mathcal{A}$ be a PPT adversary and we consider $\mathcal{A}$'s advantage against the ADDH1 problem as follows.

$$\mathsf{Adv}^{\mathsf{ADDH1}}_{\mathcal{G},\mathcal{A}}(\lambda) := \left| \Pr\left[b' = b \;\middle|\; \begin{array}{l} D := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e) \leftarrow \mathcal{G}(\lambda), \\ d, c_1, c_2 \xleftarrow{\$} \mathbb{Z}_p, c_3 \xleftarrow{\$} \mathbb{Z}_p^{\times},\ b \xleftarrow{\$} \{0,1\}, \\ \text{if } b = 0 \text{ then } Z := g_1^{c_1 c_2},\ \text{else } Z \xleftarrow{\$} \mathbb{G}_1, \\ b' \leftarrow \mathcal{A}(\lambda, D, g_1^{c_1}, g_1^{c_2}, g_1^{dc_3}, g_2^{d}, g_2^{c_2 c_3}, g_2^{dc_3}, g_2^{\frac{1}{c_3}}, Z) \end{array} \right] - \frac{1}{2} \right|.$$

Definition 4 *(*ADDH1 *assumption).* The ADDH1 assumption relative to a generator $\mathcal{G}$ holds if for all PPT adversaries $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{ADDH1}}_{\mathcal{G},\mathcal{A}}(\lambda)$ is negligible in λ.

This assumption is similar to the DDH2v assumption ("v" stands for "variant"), which was used for constructing the Lewko-Waters IBE [28] in prime-order groups in [36]. Similarly, we can also consider the DDH1v assumption.[10] The authors of [36] argued that the DDH2v (resp., DDH1v) assumption is the minimal assumption when one tries to put some information about c_1 or c_2 in an instance of DDH1 (resp., DDH2) while staying in the hardness of the problem. We define the ADDH1 problem by removing g_1^d from the DDH1v problem and adding $g_1^{dc_3}$ and g_2^{1/c_3}. Therefore, we may say this new assumption is also a not-so-strange one. Actually, we prove the security of this assumption in the generic bilinear group model as follows. We postpone the formal proof to Section 6.

Theorem 1 *(Informal). Let $\mathcal{A}$ be an algorithm that attempts to solve the* ADDH1 *problem in the generic group model. $\mathcal{A}$ makes at most q queries to the oracles computing the group actions in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$, and the bilinear map e. Then, the advantage ϵ of $\mathcal{A}$ in solving the problem is bounded by $\epsilon \leq 3(q+11)^2/4p$.*

[10] We give the formal definition of the DDH2v and DDH1v assumptions in Appendix A.2.

We prove the security of Π_{JR} under the above assumptions.

Theorem 2. *If the* ADDH1 *and* DDH2 *assumptions hold, then the resulting modified Jutla-Roy IBE* Π_{JR} *is* IND-ID-CPA *secure.*

Proof. Our security proof is the same as that of the Jutla-Roy IBE except that we have to care the extra terms $(\chi_1, g_2^{x_0\beta}, g_2^{y_0\beta}, g_2^{1/\beta})$ that were added to their scheme. We replace the DDH1 assumption of Jutla-Roy's proof with "DDH1 with the additional instance", the ADDH1 assumption, in order to treat these extra terms. More specifically, we need the ADDH1 assumption in the proof of indistinguishability of the semi-functional challenge ciphertext and the random element in the ciphertext space (see Lemma 3 for details).

We first describe how semi-functional ciphertexts and secret keys are generated as follows.

Semi-functional ciphertext: Parse a normal ciphertext C as $(C_0, C_1, C_2, C_3, \mathtt{tag})$. A semi-functional ciphertext $\widetilde{C} := (\tilde{C}_0, \tilde{C}_1, \tilde{C}_2, \tilde{C}_3, \widetilde{\mathtt{tag}})$ is computed as follows:

$$\begin{aligned}
\tilde{C}_0 &:= C_0 e(g_1, g_2)^{-x_0\mu} = M e(g_1, g_2)^{-x_0(\alpha t+\mu)+y_0 t},\\
\tilde{C}_1 &:= C_1,\\
\tilde{C}_2 &:= C_2 g_1^{\mu} = g_1^{\alpha t+\mu},\\
\tilde{C}_3 &:= C_3\Big((g_1^{x_1})^{\mathtt{I}}(g_1^{x_2})^{\mathtt{tag}} g_1^{x_3}\Big)^{-\mu}\\
&= C_3 g_1^{-\mu(x_1\mathtt{I}+x_2\mathtt{tag}+x_3)}\\
&= g_1^{-(\alpha t+\mu)(x_1\mathtt{I}+x_2\mathtt{tag}+x_3)} g_1^{t(y_1\mathtt{I}+y_2\mathtt{tag}+y_3)},
\end{aligned}$$

and $\widetilde{\mathtt{tag}} := \mathtt{tag}$, where $\mu \xleftarrow{\$} \mathbb{Z}_p^{\times}$. Note that the master key $g_2^{-x_0}$ is needed to generate the semi-functional ciphertext.

Semi-functional secret key: Parse a secret key $SK_{\mathtt{I}}$ as $(D_1, D_1', D_2, D_2', D_3)$. A semi-functional secret key $\widetilde{SK}_{\mathtt{I}} := (\tilde{D}_1, \tilde{D}_1', \tilde{D}_2, \tilde{D}_2', \tilde{D}_3)$ is computed as follows:

$$\begin{aligned}
\tilde{D}_1 &:= D_1 g_2^{\gamma} = g_2^{y_2 r+\gamma},\\
\tilde{D}_1' &:= D_1' g_2^{\gamma\phi} = g_2^{y_0+r(\mathtt{I}y_1+y_3)+\gamma\phi},\\
\tilde{D}_2 &:= D_2 g_2^{-\frac{\gamma}{\alpha}} = g_2^{-rx_2-\frac{\gamma}{\alpha}},\\
\tilde{D}_2' &:= D_2' g_2^{-\frac{\gamma\phi}{\alpha}} = g_2^{-x_0-r(\mathtt{I}x_1+x_3)-\frac{\gamma\phi}{\alpha}},\\
\tilde{D}_3 &:= D_3,
\end{aligned}$$

where $\phi \xleftarrow{\$} \mathbb{Z}_p$ and $\gamma \xleftarrow{\$} \mathbb{Z}_p^{\times}$. Note that in order to generate the semi-functional secret key, $g_2^{\frac{1}{\alpha}}$ is needed in addition to the public parameter.

A semi-functional ciphertext for $\mathtt{I}$ can be decrypted with a secret key for $\mathtt{I}$. This fact can be easily checked by

$$\frac{e(g_1, g_2)^{-x_0\mu} e(g_1^{-\mu(x_1\mathtt{I}+x_2\mathtt{tag}+x_3)}, D_3)}{e(g_1^{\mu}, D_2^{\mathtt{tag}} D_2')} = 1_{\mathbb{G}_T},$$

where $1_{\mathbb{G}_T}$ is an identity element of $\mathbb{G}_T$. Also, a normal ciphertext can be decrypted with a semi-functional secret key since it holds

$$e(C_1, g_2^{\gamma\mathtt{tag}} g_2^{\gamma\phi}) e(C_2, g_2^{-\frac{\gamma}{\alpha}\mathtt{tag}} g_2^{-\frac{\gamma\phi}{\alpha}}) = 1_{\mathbb{G}_T}.$$

We define the following games:

$\mathsf{Game}_{\mathsf{Real}}$: This is the same as the IND-ID-CPA game.

Game_0: This is the same as $\mathsf{Game}_{\mathsf{Real}}$ except that the challenge ciphertext is semi-functional.

Game_k $(1 \le k \le q)$: This is the same as Game_0 except for the following modification: Let q be the maximum number of identities issued to the *KeyGen* oracle, and $\mathtt{I}_i$ $(1 \le i \le q)$ be an i-th identity issued to the oracle. If queries regarding the first k identities $\mathtt{I}_1, \ldots, \mathtt{I}_k$ are issued, then semi-functional keys are returned. The rest of keys (i.e., keys for $\mathtt{I}_{k+1}, \ldots, \mathtt{I}_q$) are normal.

$\mathsf{Game}_{\mathsf{Final}}$: This is the same as Game_q except that the challenge ciphertext is a semi-functional one of a random element of $\mathbb{G}_T$.

Let S_{Real}, S_k $(0 \le k \le q)$, and S_{Final} be the probabilities that the event $b' = b$ occurs in $\mathsf{Game}_{\mathsf{Real}}$, Game_k, and $\mathsf{Game}_{\mathsf{Final}}$, respectively. We have

$$\mathsf{Adv}^{\mathsf{ID\text{-}CPA}}_{\Pi_{\mathrm{JR}},\mathcal{A}}(\lambda) \le |S_{\mathsf{Real}} - S_0| + \sum_{i=1}^{q} |S_{i-1} - S_i| + \left|S_q - S_{\mathsf{Final}}\right| + \left|S_{\mathsf{Final}} - \frac{1}{2}\right|.$$

Obviously, $|S_{\mathsf{Final}} - 1/2| = 0$. The rest of the proof follows from the following lemmas.

Lemma 1. $|S_{Real} - S_0| \le 2\mathsf{Adv}^{\mathsf{DDH1}}_{\mathcal{G},\mathcal{B}}(\lambda)$.

Proof. At the beginning, a PPT adversary $\mathcal{B}$ receives an instance $(g_1, g_1^{c_1}, g_1^{c_2}, g_2, Z)$ of the DDH1 problem. Then, $\mathcal{B}$ randomly chooses $x_0, y_0, x_1, y_1, x_2, y_2, x_3, y_3 \xleftarrow{\$} \mathbb{Z}_p$ and $\beta \xleftarrow{\$} \mathbb{Z}_p^{\times}$, and creates

$$z := e(g_1^{c_1}, g_2)^{-x_0} e(g_1, g_2)^{y_0},\ \ u_1 := (g_1^{c_1})^{-x_1} g_1^{y_1},\ \ w_1 := (g_1^{c_1})^{-x_2} g_1^{y_2},\ \ h_1 := (g_1^{c_1})^{-x_3} g_1^{y_3},\ \ \chi_1 := (g_1^{c_1})^{-x_0\beta} g_1^{y_0\beta}.$$

$\mathcal{B}$ sends $mpk := (g_1, g_1^{\alpha}, u_1, w_1, h_1, \chi_1, g_2, g_2^{x_1}, g_2^{x_2}, g_2^{x_3}, g_2^{y_1}, g_2^{y_2}, g_2^{y_3}, z, g_2^{\beta x_0}, g_2^{\beta y_0}, g_2^{\frac{1}{\beta}})$ to $\mathcal{A}$. Note that $\mathcal{B}$ knows a master key $msk := (g_2^{y_0}, g_2^{-x_0})$ and we implicitly set $\alpha := c_1$.

KeyGen oracle. $\mathcal{B}$ can simulate the oracle since $\mathcal{B}$ knows the master key.

Challenge. $\mathcal{B}$ receives $(M_0^*, M_1^*, \mathtt{I}^*)$ from $\mathcal{A}$. $\mathcal{B}$ chooses $d \xleftarrow{\$} \{0,1\}$. $\mathcal{B}$ chooses $\mathtt{tag}^* \xleftarrow{\$} \mathbb{Z}_p$ and computes

$$C_0^* := M_d^* e(Z, g_2)^{-x_0} e(g_1^{c_2}, g_2)^{y_0},\ \ C_1^* := g_1^{c_2},\ \ C_2^* := Z,\ \ C_3^* := Z^{-x_1\mathtt{I}^* - x_2\mathtt{tag}^* - x_3} (g_1^{c_2})^{y_1\mathtt{I}^* + y_2\mathtt{tag}^* + y_3}.$$

$\mathcal{B}$ sends $C^* := (C_0^*, C_1^*, C_2^*, C_3^*, \mathtt{tag}^*)$ to $\mathcal{A}$.

If $b = 0$, then the above ciphertext is normal by setting $t := c_2$. If $b = 1$, then the above ciphertext is semi-functional since it holds

$$\begin{aligned}
C_0^* &= M_d^* e(g_1, g_2)^{-x_0(c_1c_2+\mu)+y_0c_2} = M_d^* e(g_1, g_2)^{-x_0(\alpha t+\mu)+y_0 t},\\
C_2^* &= g_1^{c_1c_2+\mu} = g_1^{\alpha t+\mu},\\
C_3^* &= g_1^{-(c_1c_2+\mu)(x_1\mathtt{I}^* + x_2\mathtt{tag}^* + x_3)} g_1^{c_2(y_1\mathtt{I}^* + y_2\mathtt{tag}^* + y_3)} = g_1^{-(\alpha t+\mu)(x_1\mathtt{I}^* + x_2\mathtt{tag}^* + x_3)} g_1^{t(y_1\mathtt{I}^* + y_2\mathtt{tag}^* + y_3)}.
\end{aligned}$$

After receiving d' from $\mathcal{A}$, $\mathcal{B}$ sends $b' = 1$ to the challenger of the $DDH1$ problem if $d' = d$. Otherwise, $\mathcal{B}$ sends $b' = 0$ to the challenger. □

Lemma 2. $|S_{k-1} - S_k| \le 2\mathsf{Adv}^{\mathsf{DDH2}}_{\mathcal{G},\mathcal{B}}(\lambda)$ *for every* $k \in \{1, 2, \ldots, q\}$.

Proof. At the beginning, a PPT adversary $\mathcal{B}$ receives an instance $(g_1, g_2, g_2^{c_1}, g_2^{c_2}, Z)$ of the DDH2 problem. Then, $\mathcal{B}$ randomly chooses $x_0', y_0, x_1', y_1', y_1'', x_2', x_3', y_3', y_3'' \xleftarrow{\$} \mathbb{Z}_p$ and $\alpha, \beta \xleftarrow{\$} \mathbb{Z}_p^{\times}$, and (implicitly) sets

$$\begin{aligned}
x_0 &:= \frac{x_0' + y_0}{\alpha},\ \ x_1 := \frac{x_1' + y_1}{\alpha},\ \text{where } y_1 := y_1' + c_2 y_1'',\\
x_2 &:= \frac{x_2' + c_2}{\alpha},\ \ y_2 := c_2,\ \ x_3 := \frac{x_3' + y_3}{\alpha},\ \text{where } y_3 := y_3' + c_2 y_3''.
\end{aligned}$$

$\mathcal{B}$ creates

$$\begin{aligned}
&z := e(g_1, g_2)^{-x_0'},\ \ u_1 := g_1^{-x_1'},\ \ w_1 := g_1^{-x_2'},\ \ h_1 := g_1^{-x_3'},\ \ \chi_1 := g_1^{-x_0'\beta},\\
&g_2^{x_1} := g_2^{\frac{x_1'+y_1'}{\alpha}} (g_2^{c_2})^{\frac{y_1''}{\alpha}},\ \ g_2^{y_1} := g_2^{y_1'} (g_2^{c_2})^{y_1''},\ \ g_2^{x_2} := g_2^{\frac{x_2'}{\alpha}} (g_2^{c_2})^{\frac{1}{\alpha}},\ \ g_2^{y_2} := g_2^{c_2},\\
&g_2^{x_3} := g_2^{\frac{x_3'+y_3'}{\alpha}} (g_2^{c_2})^{\frac{y_3''}{\alpha}},\ \ g_2^{y_3} := g_2^{y_3'} (g_2^{c_2})^{y_3''}.
\end{aligned}$$

$\mathcal{B}$ sends $mpk := (g_1, g_1^{\alpha}, u_1, w_1, h_1, \chi_1, g_2, g_2^{x_1}, g_2^{x_2}, g_2^{x_3}, g_2^{y_1}, g_2^{y_2}, g_2^{y_3}, z, g_2^{\beta x_0}, g_2^{\beta y_0}, g_2^{\frac{1}{\beta}})$ to $\mathcal{A}$. Note that $\mathcal{B}$ knows a master key $msk := (g_2^{y_0}, g_2^{-x_0})$.

KeyGen oracle. Let $\mathtt{I}_i$ $(1 \le i \le q)$ be an i-th identity issued to the oracle. $\mathcal{B}$ creates $k-1$ semi-functional keys, and embeds Z into the k-th keys. The rest of keys are normal.

Case $i < k$**:** $\mathcal{B}$ creates and returns semi-functional keys. Since $\mathcal{B}$ knows the master key and α, $\mathcal{B}$ can create semi-functional keys.

Case $i = k$**:** $\mathcal{B}$ creates a semi-functional key by embedding Z as follows: $\mathcal{B}$ computes

$$\begin{aligned}
D_1 &:= Z,\\
D_1' &:= g_2^{y_0}(g_2^{c_1})^{\mathtt{I}_k y_1' + y_3'} Z^{\mathtt{I}_k y_1'' + y_3''},\\
D_2 &:= \left((g_2^{c_1})^{x_2'} Z\right)^{-\frac{1}{\alpha}},\\
D_2' &:= g_2^{-\frac{x_0'}{\alpha}}(g_2^{c_1})^{-\frac{\mathtt{I}_k(x_1'+y_1')+x_3'+y_3'}{\alpha}} g_2^{-\frac{y_0}{\alpha}} Z^{-\frac{\mathtt{I}_k y_1'' + y_3''}{\alpha}},\\
D_3 &:= g_2^{c_1}.
\end{aligned}$$

$\mathcal{B}$ sets $SK_{\mathtt{I}_k} := (D_1, D_1', D_2, D_2', D_3)$. If $b = 0$, then it is easy to see that the above keys are normal by setting $r := c_1$. If $b = 1$, then the above ciphertext is semi-functional since it holds

$$\begin{aligned}
D_1 :=& Z = g_2^{c_1 c_2 + \gamma} = g_2^{y_2 r + \gamma},\\
D_1' :=& g_2^{y_0}(g_2^{c_1})^{\mathtt{I}_k y_1' + y_3'} Z^{\mathtt{I}_k y_1'' + y_3''}\\
=& g_2^{y_0 + c_1(\mathtt{I}_k(y_1' + c_2 y_1'') + y_3' + c_2 y_3'')} g_2^{\gamma(\mathtt{I}_k y_1'' + y_3'')}\\
=& g_2^{y_0 + r(\mathtt{I}_k y_1 + y_3)} g_2^{\gamma\phi},\\
D_2 :=& \left((g_2^{c_1})^{x_2'} Z\right)^{-\frac{1}{\alpha}} = g_2^{-\frac{c_1(x_2'+c_2)}{\alpha}} g_2^{-\frac{\gamma}{\alpha}} = g_2^{-r x_2} g_2^{-\frac{\gamma}{\alpha}},\\
D_2' :=& g_2^{-\frac{x_0'}{\alpha}}(g_2^{c_1})^{-\frac{\mathtt{I}_k(x_1'+y_1')+x_3'+y_3'}{\alpha}} g_2^{-\frac{y_0}{\alpha}} Z^{-\frac{\mathtt{I}_k y_1'' + y_3''}{\alpha}}\\
=& g_2^{-\frac{(x_0'+y_0)+c_1(\mathtt{I}_k(x_1'+y_1'+c_2 y_1'')+(x_3'+y_3'+c_2 y_3''))}{\alpha}} \cdot g_2^{-\frac{\gamma(\mathtt{I}_k y_1'' + y_3'')}{\alpha}}\\
=& g_2^{-x_0 - r(\mathtt{I}_k x_1 + x_3)} g_2^{-\frac{\gamma\phi}{\alpha}},
\end{aligned}$$

where $Z := g_2^{c_1 c_2 + \gamma}$, $r := c_1$, and $\phi := \mathtt{I}_k y_1'' + y_3''$. Since y_1'' and y_3'' are chosen uniformly at random, ϕ is also uniformly distributed.

Case $i > k$**:** $\mathcal{B}$ creates and returns normal keys by using the master key.

Challenge. $\mathcal{B}$ receives $(M_0^*, M_1^*, \mathtt{I}^*)$ from $\mathcal{A}$. $\mathcal{B}$ chooses $d \xleftarrow{\$} \{0,1\}$. However, $\mathcal{B}$ cannot create a semi-functional ciphertext for $\mathtt{I}^*$ without knowledge of c_2 (and hence y_1 and y_3). To generate the semi-functional ciphertext without the knowledge, $\mathcal{B}$ sets

$$\widetilde{\mathtt{tag}}^* := -\mathtt{I}^* y_1'' - y_3''.$$

Since y_1'' and y_3'' are chosen uniformly at random, probability distribution of $\widetilde{\mathtt{tag}}^*$ is also uniformly at random from $\mathcal{A}$'s view. Then, $\mathcal{B}$ chooses $t \xleftarrow{\$} \mathbb{Z}_p$ and $\mu \xleftarrow{\$} \mathbb{Z}_p^\times$, and computes

$$\begin{aligned}
\tilde{C}_0^* :=& M_d^* z^t e(g_1, g_2)^{-x_0\mu} = M_d^* e(g_1, g_2)^{-x_0(\alpha t + \mu) + y_0 t},\\
\tilde{C}_1^* :=& g_1^t,\\
\tilde{C}_2^* :=& g_1^{\alpha t + \mu}\\
\tilde{C}_3^* :=& \left(u_1^{\mathtt{I}^*} w_1^{\widetilde{\mathtt{tag}}^*} h_1\right)^t g_1^{-\frac{\mu}{\alpha}(\mathtt{I}^*(x_1'+y_1')+x_2'\widetilde{\mathtt{tag}}^*+x_3'+y_3')}\\
=& \left(u_1^{\mathtt{I}^*} w_1^{\widetilde{\mathtt{tag}}^*} h_1\right)^t \cdot g_1^{-\frac{\mu}{\alpha}(\mathtt{I}^*(x_1'+y_1'+c_2 y_1'')+\widetilde{\mathtt{tag}}^*(x_2'+c_2)+x_3'+y_3'+c_2 y_3'')} \cdot g_1^{\frac{c_2\mu}{\alpha}(\mathtt{I}^* y_1'' + \widetilde{\mathtt{tag}}^* + y_3'')}\\
=& \left(u_1^{\mathtt{I}^*} w_1^{\widetilde{\mathtt{tag}}^*} h_1\right)^t g_1^{-\mu(\mathtt{I}^* x_1 + \widetilde{\mathtt{tag}}^* x_2 + x_3)}.
\end{aligned}$$

$\mathcal{B}$ sends $\tilde{C}^* := (\tilde{C}_0^*, \tilde{C}_1^*, \tilde{C}_2^*, \tilde{C}_3^*, \widetilde{\mathtt{tag}}^*)$ to $\mathcal{A}$.

After receiving d' from $\mathcal{A}$, $\mathcal{B}$ sends $b' = 1$ to the challenger of the $DDH2$ problem if $d' = d$. Otherwise, $\mathcal{B}$ sends $b' = 0$ to the challenger. □

Lemma 3. $|S_q - S_{Final}| \leq 2\mathsf{Adv}^{\mathsf{ADDH1}}_{\mathcal{G},\mathcal{B}}(\lambda)$.

Proof. At the beginning, a PPT adversary $\mathcal{B}$ receives an instance $(g_1, g_1^{dc_3}, g_1^{c_1}, g_1^{c_2}, g_2, g_2^{d}, g_2^{c_2c_3}, g_2^{dc_3}, g_2^{\frac{1}{c_3}}, Z)$ of the ADDH1 problem. Then, $\mathcal{B}$ randomly chooses $x_1, x_2, x_3, y'_1, y'_2, y'_3 \xleftarrow{\$} \mathbb{Z}_p$ and $\alpha \xleftarrow{\$} \mathbb{Z}_p^{\times}$, and (implicitly) sets

$$x_0 := c_2, \quad y'_0 := d, \quad y_0 := x_0\alpha + y'_0, \quad y_1 := x_1\alpha + y'_1,$$
$$y_2 := x_2\alpha + y'_2, \quad y_3 := x_3\alpha + y'_3,$$
$$\beta := c_3, \quad \beta x_0 := c_2c_3, \quad \beta y_0 := \beta(x_0\alpha + y'_0) = \alpha c_2c_3 + dc_3.$$

Then, $\mathcal{B}$ creates

$$z := e(g_1, g_2^d) = e(g_1, g_2)^{y'_0}, \quad u_1 := g_1^{y'_1}, \quad w_1 := g_1^{y'_2}, \quad h_1 := g_1^{y'_3},$$
$$\chi_1 := g_1^{dc_3} = g_1^{\beta y'_0}, \quad g_2^{\beta x_0} := g_2^{c_2c_3}, \quad g_2^{\beta y_0} := (g_2^{c_2c_3})^{\alpha} g_2^{dc_3}, \quad g_2^{\frac{1}{\beta}} := g_2^{\frac{1}{c_3}}.$$

$\mathcal{B}$ sends $mpk := (g_1, g_1^{\alpha}, u_1, w_1, h_1, \chi_1, g_2, g_2^{x_1}, g_2^{x_2}, g_2^{x_3}, g_2^{y_1}, g_2^{y_2}, g_2^{y_3}, z, g_2^{\beta x_0}, g_2^{\beta y_0}, g_2^{\frac{1}{\beta}})$ to $\mathcal{A}$. Note that $\mathcal{B}$ does not know a master key $msk := (g_2^{y_0}, g_2^{-x_0})$.

KeyGen oracle. When receiving a query $\mathtt{I}$, $\mathcal{B}$ chooses $r, \phi' \xleftarrow{\$} \mathbb{Z}_p$ and $\gamma \xleftarrow{\$} \mathbb{Z}_p^{\times}$, and (implicitly) sets

$$\phi := \frac{\alpha(\phi' - x_0 - (x_1\mathtt{I} + x_3)r)}{\gamma} \quad (\text{hence } \phi' = x_0 + (x_1\mathtt{I} + x_3)r + \frac{\gamma\phi}{\alpha}).$$

ϕ is randomly distributed from $\mathcal{A}$'s viewpoint due to ϕ', and note that $\mathcal{B}$ does not know the value of ϕ. Then $\mathcal{B}$ computes

$$D_1 := g_2^{y_2 r + \gamma},$$
$$D'_1 := g_2^{d} g_2^{(y'_1\mathtt{I} + y'_3)r + \alpha\phi'}$$
$$= g_2^{x_0\alpha + y'_0 + ((x_1\alpha + y'_1)\mathtt{I} + x_3\alpha + y'_3)r + \gamma\phi}$$
$$= g_2^{y_0 + (y_1\mathtt{I} + y_3)r + \gamma\phi},$$
$$D_2 := g_2^{-x_2 r - \frac{\gamma}{\alpha}},$$
$$D'_2 := g_2^{-\phi'} = g_2^{-x_0 - (x_1\mathtt{I} + x_3)r - \frac{\gamma\phi}{\alpha}},$$
$$D_3 := g_2^{r}.$$

$\mathcal{B}$ sends $SK_{\mathtt{I}} := (D_1, D'_1, D_2, D'_2, D_3)$ to $\mathcal{A}$.

Challenge. $\mathcal{B}$ receives $(M_0^*, M_1^*, \mathtt{I}^*)$ from $\mathcal{A}$. $\mathcal{B}$ chooses $d \xleftarrow{\$} \{0,1\}$. $\mathcal{B}$ chooses $t, \mathtt{tag}^* \xleftarrow{\$} \mathbb{Z}_p$ and computes

$$C_0^* := M_d^* \cdot e(g_1, g_2^d)^t e(Z, g_2)^{-1}, \quad C_1^* := g_1^t, \quad C_2^* := g_1^{\alpha t} g_1^{c_1},$$
$$C_3^* := (u_1^{\mathtt{I}^*} w_1^{\mathtt{tag}^*} h_1)^t (g_1^{c_1})^{-x_1\mathtt{I}^* - x_2\mathtt{tag}^* - x_3}.$$

$\mathcal{B}$ sends $C^* := (C_0^*, C_1^*, C_2^*, C_3^*, \mathtt{tag}^*)$ to $\mathcal{A}$.

If $b = 0$, then the above ciphertext is semi-functional one of M_d^* by setting $\mu := c_1$. If $b = 1$, then the above ciphertext is semi-functional one of a random element of $\mathbb{G}_T$ since it holds

$$C_0^* = M_d^* \cdot e(g_1, g_2)^{y'_0 t - x_0\mu - \eta}$$
$$= M_d^* \cdot e(g_1, g_2)^{-x_0\alpha t + y_0 t - x_0\mu - \eta}$$
$$= M_d^* \cdot e(g_1, g_2)^{-x_0(\alpha t + \mu) + y_0 t} e(g_1, g_2)^{-\eta}$$
$$= R \cdot e(g_1, g_2)^{-x_0(\alpha t + \mu) + y_0 t},$$

where $R = M_d^* e(g_1, g_2)^{-\eta}$.

After receiving d' from $\mathcal{A}$, $\mathcal{B}$ sends $b' = 1$ to the challenger of the ADDH1 problem if $d' = d$. Otherwise, $\mathcal{B}$ sends $b' = 0$ to the challenger. □

Proof of Theorem 2. From Lemmas 1–3, we have $\mathsf{Adv}^{\mathsf{ID\text{-}CPA}}_{\Pi_{\mathrm{JR}},\mathcal{A}}(\lambda) \leq 2\mathsf{Adv}^{\mathsf{DDH1}}_{\mathcal{G},\mathcal{B}}(\lambda) + 2q \cdot \mathsf{Adv}^{\mathsf{DDH2}}_{\mathcal{G},\mathcal{B}}(\lambda) + 2\mathsf{Adv}^{\mathsf{ADDH1}}_{\mathcal{G},\mathcal{B}}(\lambda) \leq 4\mathsf{Adv}^{\mathsf{ADDH1}}_{\mathcal{G},\mathcal{B}}(\lambda) + 2q \cdot \mathsf{Adv}^{\mathsf{DDH2}}_{\mathcal{G},\mathcal{B}}(\lambda)$. □

4. Our construction

We construct an RIBE scheme based on the original Jutla-Roy IBE, and prove that the security of the proposed scheme relies on that of the modified Jutla-Roy IBE. An RIBE scheme Π =(Setup, SKGen, KeyUp, DKGen, Enc, Dec, Revoke) is constructed as follows.

- Setup(λ, N): It runs $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e) \leftarrow \mathcal{G}$. It chooses $x_0, y_0, x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4, x_5, y_5 \xleftarrow{\$} \mathbb{Z}_p$ and $\alpha \xleftarrow{\$} \mathbb{Z}_p^{\times}$, and sets

$$z := e(g_1, g_2)^{-x_0\alpha+y_0},\ u_1 := g_1^{-x_1\alpha+y_1},\ \ w_1 := g_1^{-x_2\alpha+y_2},\ \ h_1 := g_1^{-x_3\alpha+y_3},$$
$$v_1 := g_1^{-x_4\alpha+y_4},\ \hat{v}_1 := g_1^{-x_5\alpha+y_5}.$$

  Let BT be a binary tree that has N leaves, where N is a power of two for simplicity. It outputs

$$mpk := (g_1, g_1^{\alpha}, u_1, w_1, h_1, v_1, \hat{v}_1, g_2, g_2^{x_1}, g_2^{x_2}, \ldots, g_2^{x_5}, g_2^{y_1}, g_2^{y_2}, \ldots, g_2^{y_5}, z),$$
$$msk := (g_2^{y_0}, g_2^{-x_0}),$$

  $st :=$ BT, and $RL := \emptyset$.
- SKGen(st, I): Parse st as BT. It randomly chooses an unassigned leaf η from BT, and stores I in the node η. For each node $\theta \in$ Path(BT, η), it recalls P_θ if it was defined. Otherwise, it chooses $P_\theta \xleftarrow{\$} \mathbb{G}_2$ and stores P_θ in the node θ. Then, it chooses $r_\theta \xleftarrow{\$} \mathbb{Z}_p$ and it computes

$$\mathtt{SK}_{1,\theta} := (g_2^{y_2})^{r_\theta},\ \mathtt{SK}'_{1,\theta} := P_\theta\Big((g_2^{y_1})^{\mathtt{I}} g_2^{y_3}\Big)^{r_\theta},$$
$$\mathtt{SK}_{2,\theta} := (g_2^{x_2})^{-r_\theta},\ \mathtt{SK}'_{2,\theta} := P_\theta\Big((g_2^{x_1})^{\mathtt{I}} g_2^{x_3}\Big)^{-r_\theta},\ \ \mathtt{SK}_{3,\theta} := g_2^{r_\theta}.$$

  It outputs $sk_{\mathtt{I}} := \{(\mathtt{SK}_{1,\theta}, \mathtt{SK}'_{1,\theta}, \mathtt{SK}_{2,\theta}, \mathtt{SK}'_{2,\theta}, \mathtt{SK}_{3,\theta})\}_{\theta \in \mathsf{Path}(\mathtt{BT},\eta)}$.
- KeyUp(msk, st, RL, T): Parse msk as $(\mathtt{MK}_1, \mathtt{MK}_2)$. For each node $\theta \in$ KUNode(BT, RL, T), it recalls P_θ if it was defined. Otherwise, it chooses $P_\theta \xleftarrow{\$} \mathbb{G}_2$ and stores P_θ in the node θ. It chooses $s_\theta \xleftarrow{\$} \mathbb{Z}_p$ and computes

$$\mathtt{KU}'_{1,\theta} := P_\theta^{-1}\mathtt{MK}_1\Big((g_2^{y_4})^{\mathtt{T}} g_2^{y_5}\Big)^{s_\theta},$$
$$\mathtt{KU}'_{2,\theta} := P_\theta^{-1}\mathtt{MK}_2\Big((g_2^{x_4})^{\mathtt{T}} g_2^{x_5}\Big)^{-s_\theta},\ \mathtt{KU}_{3,\theta} := g_2^{s_\theta}.$$

  It outputs $ku_{\mathtt{T}} := \{(\mathtt{KU}'_{1,\theta}, \mathtt{KU}'_{2,\theta}, \mathtt{KU}_{3,\theta})\}_{\theta \in \mathsf{KUNode}(\mathtt{BT}, RL, \mathtt{T})}$.
- DKGen($sk_{\mathtt{I}}, ku_{\mathtt{T}}$): Parse $sk_{\mathtt{I}}$ and $ku_{\mathtt{T}}$ as $\{(\mathtt{SK}_{1,\theta}, \mathtt{SK}'_{1,\theta}, \mathtt{SK}_{2,\theta}, \mathtt{SK}'_{2,\theta}, \mathtt{SK}_{3,\theta})\}_{\theta \in \Theta_{\mathtt{SK}}}$ and $\{(\mathtt{KU}'_{1,\theta}, \mathtt{KU}'_{2,\theta}, \mathtt{KU}_{3,\theta})\}_{\theta \in \Theta_{\mathtt{KU}}}$, respectively. It outputs $\perp$ if $\Theta_{\mathtt{SK}} \cap \Theta_{\mathtt{KU}} = \emptyset$. Otherwise, for some $\theta \in \Theta_{\mathtt{SK}} \cap \Theta_{\mathtt{KU}}$, it computes as follows. It chooses $R, S \xleftarrow{\$} \mathbb{Z}_p$ and computes

$$\mathtt{DK}_1 := \mathtt{SK}_{1,\theta}(g_2^{y_2})^R,$$
$$\mathtt{DK}'_1 := \mathtt{SK}'_{1,\theta}\mathtt{KU}'_{1,\theta}\Big((g_2^{y_1})^{\mathtt{I}} g_2^{y_3}\Big)^R\Big((g_2^{y_4})^{\mathtt{T}} g_2^{y_5}\Big)^S,$$
$$\mathtt{DK}_2 := \mathtt{SK}_{2,\theta}(g_2^{x_2})^{-R},$$
$$\mathtt{DK}'_2 := \mathtt{SK}'_{2,\theta}\mathtt{KU}'_{2,\theta}\Big((g_2^{x_1})^{\mathtt{I}} g_2^{x_3}\Big)^{-R}\Big((g_2^{x_4})^{\mathtt{T}} g_2^{x_5}\Big)^{-S},$$
$$\mathtt{DK}_3 := \mathtt{SK}_{3,\theta} g_2^R,\ \ \mathtt{DK}_4 := \mathtt{KU}_{3,\theta} g_2^S.$$

  It outputs $dk_{\mathtt{I},\mathtt{T}} := (\mathtt{DK}_1, \mathtt{DK}'_1, \mathtt{DK}_2, \mathtt{DK}'_2, \mathtt{DK}_3, \mathtt{DK}_4)$.
- Enc(M, I, T): It chooses $t, \mathtt{tag} \xleftarrow{\$} \mathbb{Z}_p$. For $M \in \mathbb{G}_T$, it computes

$$C_0 := Mz^t,\ \ C_1 := g_1^t,\ \ C_2 := (g_1^{\alpha})^t,\ \ C_3 := \Big(u_1^{\mathtt{I}} w_1^{\mathtt{tag}} h_1\Big)^t,\ \ C_4 := (v_1^{\mathtt{T}}\hat{v}_1)^t.$$

  It outputs $C_{\mathtt{I},\mathtt{T}} := (C_0, C_1, C_2, C_3, C_4, \mathtt{tag})$.
- Dec($dk_{\mathtt{I},\mathtt{T}}, C_{\mathtt{I},\mathtt{T}}$): Parse $dk_{\mathtt{I},\mathtt{T}}$ and $C_{\mathtt{I},\mathtt{T}}$ as $(\mathtt{DK}_1, \mathtt{DK}'_1, \mathtt{DK}_2, \mathtt{DK}'_2, \mathtt{DK}_3, \mathtt{DK}_4)$ and $(C_0, C_1, C_2, C_3, C_4, \mathtt{tag})$, respectively. It computes

$$M = \frac{C_0 e(C_3, \mathtt{DK}_3) e(C_4, \mathtt{DK}_4)}{e(C_1, \mathtt{DK}_1^{\mathtt{tag}}\mathtt{DK}'_1) e(C_2, \mathtt{DK}_2^{\mathtt{tag}}\mathtt{DK}'_2)}.$$

– Revoke$(\mathtt{I}, \mathtt{T}, RL, st)$: Output $RL := RL \cup \{(\mathtt{I}, \mathtt{T})\}$.

We show the correctness of our RIBE scheme Π.

First, we show the correctness of the DKGen algorithm. Parse $sk_{\mathtt{I}}$ and $ku_{\mathtt{T}}$ as $\{sk_\theta\}_{\theta\in\Theta_{\mathtt{SK}}} = \{(\mathtt{SK}_{1,\theta}, \mathtt{SK}'_{1,\theta}, \mathtt{SK}_{2,\theta}, \mathtt{SK}'_{2,\theta}, \mathtt{SK}_{3,\theta})\}_{\theta\in\Theta_{\mathtt{SK}}}$ and $\{ku_\theta\}_{\theta\in\Theta_{\mathtt{KU}}} = \{(\mathtt{KU}'_{1,\theta}, \mathtt{KU}'_{2,\theta}, \mathtt{KU}_{3,\theta})\}_{\theta\in\Theta_{\mathtt{KU}}}$, respectively. Let r_θ and s_θ be internal randomness of $sk_{\mathtt{I}}$ and $ku_{\mathtt{T}}$, respectively. For any $\theta \in \Theta_{\mathtt{SK}} \cap \Theta_{\mathtt{KU}}$, we have

$$
\begin{aligned}
\mathtt{DK}_1 &:= \mathtt{SK}_{1,\theta}(g_2^{y_2})^R = (g_2^{y_2})^{R+r_\theta} = (g_2^{y_2})^{\hat{R}},\\
\mathtt{DK}'_1 &:= \mathtt{SK}'_{1,\theta}\mathtt{KU}'_{1,\theta}\left((g_2^{y_1})^{\mathtt{I}}g_2^{y_3}\right)^R\left((g_2^{y_4})^{\mathtt{T}}g_2^{y_5}\right)^S\\
&= g_2^{y_0}\left((g_2^{y_1})^{\mathtt{I}}g_2^{y_3}\right)^{R+r_\theta}\left((g_2^{y_4})^{\mathtt{T}}g_2^{y_5}\right)^{S+s_\theta}\\
&= g_2^{y_0}\left((g_2^{y_1})^{\mathtt{I}}g_2^{y_3}\right)^{\hat{R}}\left((g_2^{y_4})^{\mathtt{T}}g_2^{y_5}\right)^{\hat{S}},\\
\mathtt{DK}_2 &:= \mathtt{SK}_{2,\theta}(g_2^{x_2})^{-R} = (g_2^{x_2})^{-(R+r_\theta)} = (g_2^{x_2})^{-\hat{R}},\\
\mathtt{DK}'_2 &:= \mathtt{SK}'_{2,\theta}\mathtt{KU}'_{2,\theta}\left((g_2^{x_1})^{\mathtt{I}}g_2^{x_3}\right)^{-R}\left((g_2^{x_4})^{\mathtt{T}}g_2^{x_5}\right)^{-S}\\
&= g_2^{-x_0}\left((g_2^{x_1})^{\mathtt{I}}g_2^{x_3}\right)^{-(R+r_\theta)}\left((g_2^{x_4})^{\mathtt{T}}g_2^{x_5}\right)^{-(S+s_\theta)}\\
&= g_2^{-x_0}\left((g_2^{x_1})^{\mathtt{I}}g_2^{x_3}\right)^{-\hat{R}}\left((g_2^{x_4})^{\mathtt{T}}g_2^{x_5}\right)^{-\hat{S}},\\
\mathtt{DK}_3 &:= \mathtt{SK}_{3,\theta}g_2^R = g_2^{R+r_\theta} = g_2^{\hat{R}},\\
\mathtt{DK}_4 &:= \mathtt{KU}_{3,\theta}g_2^S = g_2^{S+s_\theta} = g_2^{\hat{S}},
\end{aligned}
$$

where $R, S \xleftarrow{\$} \mathbb{Z}_p$, $\hat{R} := R + r_\theta$, and $\hat{S} := S + s_\theta$.

We then show the decryption correctness. Suppose that $dk_{\mathtt{I},\mathtt{T}}$ is correctly generated as above. Parse $dk_{\mathtt{I},\mathtt{T}}$ and $C_{\mathtt{I},\mathtt{T}}$ as $(\mathtt{DK}_1, \mathtt{DK}'_1, \mathtt{DK}_2, \mathtt{DK}'_2, \mathtt{DK}_3, \mathtt{DK}_4)$ and $(C_0, C_1, C_2, C_3, C_4, \mathtt{tag})$, respectively. Then, we have

$$
\begin{aligned}
&\frac{C_0 e(C_3, \mathtt{DK}_3)e(C_4, \mathtt{DK}_4)}{e(C_1, \mathtt{DK}_1^{\mathtt{tag}}\mathtt{DK}'_1)e(C_2, \mathtt{DK}_2^{\mathtt{tag}}\mathtt{DK}'_2)}\\
&= M\cdot e(g_1, g_2)^{(-x_0\alpha+y_0)t}\cdot\frac{e(g_1^{t(\mathtt{I}(-x_1\alpha+y_1)+\mathtt{tag}(-x_2\alpha+y_2)-x_3\alpha+y_3)}, g_2^{\hat{R}})}{e(g_1^t, g_2^{y_0+y_2\hat{R}\mathtt{tag}+y_0+\hat{R}(\mathtt{I}y_1+y_3)+\hat{S}(\mathtt{T}y_4+y_5)})}\\
&\qquad\cdot\frac{e(g_1^{t(\mathtt{T}(-x_4\alpha+y_4)-x_5\alpha+y_5)}, g_2^{\hat{S}})}{e(g_1^{\alpha t}, g_2^{-x_0-x_2\hat{R}\mathtt{tag}-x_0-\hat{R}(\mathtt{I}x_1+x_3)-\hat{S}(\mathtt{T}x_4+x_5)})}\\
&= \frac{M\cdot e(g_1, g_2)^{(-x_0\alpha+y_0)t}}{e(g_1^t, g_2^{y_0})e(g_1^{\alpha t}, g_2^{-x_0})} = M.
\end{aligned}
$$

The security of the above construction is given as follows.

Theorem 3. *If the* ADDH1 *and* DDH2 *assumptions hold, then the resulting RIBE scheme* Π *is* IND-RID-CPA *secure.*

We show the following lemma, and we obtain Theorem 3 as a corollary of the lemma.

Lemma 4. *The proposed RIBE scheme* Π *is* IND-RID-CPA *secure as long as the modified Jutla-Roy IBE* $\Pi_{\mathtt{JR}}$*, which is described in Section 3.1, is* IND-ID-CPA *secure.*

Proof. We construct a PPT algorithm $\mathcal{B}$ which breaks the IND-ID-CPA security of the modified Jutla-Roy IBE $\Pi_{\mathtt{JR}}$ using a PPT adversary $\mathcal{A}$ which breaks the IND-RID-CPA security of Π.

At the beginning, $\mathcal{B}$ receives a public parameter $PP = (g_1, g_1^\alpha, u_1, w_1, h_1, \chi_1, g_2, g_2^{x_1}, g_2^{y_1}, g_2^{x_2}, g_2^{x_3}, g_2^{y_3}, z, g_2^{x_0\beta}, g_2^{y_0\beta}, g_2^{\frac{1}{\beta}})$. $\mathcal{B}$ guesses what time period $\mathtt{T}^*$ will be submitted from $\mathcal{A}$ in the challenge phase, and it holds with probability $1/|\mathcal{T}|$. Once $\mathcal{B}$ finds the guess wrong, it terminates the simulation and outputs a random bit b'. We assume $\mathcal{B}$'s guess is right in the rest of the proof. $\mathcal{B}$ creates $\mathtt{BT}$ with N leaves. $\mathcal{B}$ chooses $\tilde{x}, \hat{x}, \tilde{y}, \hat{y} \xleftarrow{\$} \mathbb{Z}_p$ and (implicitly) sets

$$x_4 = \beta x_0 + \tilde{x},\ \ x_5 = -\mathtt{T}^*\beta x_0 + \hat{x},\ \ y_4 = \beta y_0 + \tilde{y},\ \ y_5 = -\mathtt{T}^*\beta y_0 + \hat{y},$$
$$-x_4\alpha + y_4 := \beta(-x_0\alpha + y_0) - \alpha\tilde{x} + \tilde{y},$$
$$-x_5\alpha + y_5 := -\mathtt{T}^*\beta(-x_0\alpha + y_0) - \alpha\hat{x} + \hat{y}.$$

Then, $\mathcal{B}$ computes

$$g_2^{x_4} := g_2^{\beta x_0} g_2^{\tilde{x}},\ \ g_2^{x_5} := (g_2^{\beta x_0})^{-\mathtt{T}^*} g_2^{\hat{x}},\ \ g_2^{y_4} := g_2^{\beta y_0} g_2^{\tilde{y}},\ \ g_2^{y_5} := (g_2^{\beta y_0})^{-\mathtt{T}^*} g_2^{\hat{y}},$$
$$v_1 := g_1^{-x_4\alpha + y_4} = \chi_1(g_1^{\alpha})^{-\tilde{x}} g_1^{\tilde{y}},\ \ \hat{v}_1 := g_1^{-x_5\alpha + y_5} = \chi_1^{-\mathtt{T}^*}(g_1^{\alpha})^{-\hat{x}} g_1^{\hat{y}}.$$

$\mathcal{B}$ sends $mpk := (g_1, g_1^{\alpha}, u_1, w_1, h_1, v_1, \hat{v}_1, g_2, g_2^{x_1}, g_2^{x_2}, \ldots, g_2^{x_5}, g_2^{y_1}, g_2^{y_2}, \ldots, g_2^{y_5}, z)$ to $\mathcal{A}$.

$\mathcal{B}$ guesses whether an adversary $\mathcal{A}$ will issue the target identity $\mathtt{I}^*$ to the *SKGen* oracle, and when it will issue $\mathtt{I}^*$ to the (*SKGen* and) *DKGen* oracle. More precisely, let q_1 be the maximum number of identities issued to the *SKGen* and *DKGen* oracles *before the challenge phase*. $\mathcal{B}$ randomly guesses $(k^*, i^*) \in \{1,2\} \times \{1,2,\ldots,q_1,q_1+1\}$. $k^* = 1$ denotes that $\mathcal{A}$ issues a query $\mathtt{I}^*$ for $sk_{\mathtt{I}^*}$. Note that when $k^* = 1$, $\mathtt{I}^*$ is revoked before the target time period $\mathtt{T}^*$. $k^* = 2$ denotes that $\mathcal{A}$ never issues a query $\mathtt{I}^*$ for $sk_{\mathtt{I}^*}$ during the game. $i^* \in \{1,2,\ldots,q_1\}$ denotes that $\mathcal{A}$ first issues $\mathtt{I}^*$ to $\mathcal{B}$ at the i^*-th identity in their queries (before the challenge phase). $i^* = q_1 + 1$ denotes that $\mathcal{A}$ issues a query $\mathtt{I}^*$ for $sk_{\mathtt{I}^*}$ after the challenge phase. In the following, for convenience we call a type-k^* adversary as in [42]. Furthermore, we classify these adversarial types more specifically according to the value of i^*: $\mathcal{A}$ is said to be a type-k^*-a adversary if $i^* \in \{1,2,\ldots,q_1\}$; and a type-$k^*$-b adversary if $i^* = q_1 + 1$. Once $\mathcal{B}$ finds the guess wrong, it terminates the simulation and outputs a random bit b'. In the rest of the proof, we assume $\mathcal{B}$'s guess is right. It holds with probability $1/2(q_1+1)$.

Type-1-a and type-1-b adversary. The difference of simulations between the type-1-a and type-1-b adversaries is just a way of simulating the *SKGen* and *DKGen* oracles. When $\mathcal{A}$ is the type-1-a or type-1-b adversaries, $\mathcal{B}$ simulates the oracles as follows. $\mathcal{B}$ first chooses a node η^* for a target identity $\mathtt{I}^*$ of $\mathtt{BT}$ uniformly at random in advance.

SKGen and *DKGen* oracles for the type-1-a adversary. Suppose that $\mathcal{B}$ receives a j-th identity $\mathtt{I}$ as a secret key query $\mathtt{I}$ or a decryption key query $(\mathtt{I}, \mathtt{T})$ from $\mathcal{A}$. $\mathcal{B}$ then returns a secret key $sk_{\mathtt{I}}$ or a decryption key $dk_{\mathtt{I},\mathtt{T}}$ as follows.

Case $j < i^*$: $\mathcal{B}$ first transfers $\mathtt{I}$ to the *KeyGen* oracle of the IND-ID-CPA game of Π_{JR}, and gets $SK_{\mathtt{I}} := (D_1, D_1', D_2, D_2', D_3)$, if $\mathcal{B}$ does not have it. $\mathcal{B}$ randomly chooses an unassigned leaf η ($\neq \eta^*$) from $\mathtt{BT}$ and stores $\mathtt{I}$ in the node η if it is not done.

***SKGen* oracle:** For each node $\theta \in \mathsf{Path}(\mathtt{BT}, \eta)$, $\mathcal{B}$ recalls P_θ if it was defined. Otherwise, it chooses $P_\theta \xleftarrow{\$} \mathbb{G}_2$ and stores P_θ in the node θ. For $\theta \in \mathsf{Path}(\mathtt{BT}, \eta)$, if $\theta \notin \mathsf{Path}(\mathtt{BT}, \eta^*)$, then $\mathcal{B}$ chooses $r_\theta \xleftarrow{\$} \mathbb{Z}_p$ and computes

$$\mathtt{SK}_{1,\theta} := D_1 (g_2^{y_2})^{r_\theta},\ \ \mathtt{SK}'_{1,\theta} := P_\theta D_1' \left((g_2^{y_1})^{\mathtt{I}} g_2^{y_3}\right)^{r_\theta},$$
$$\mathtt{SK}_{2,\theta} := D_2 (g_2^{x_2})^{-r_\theta},\ \ \mathtt{SK}'_{2,\theta} := P_\theta D_2' \left((g_2^{x_1})^{\mathtt{I}} g_2^{x_3}\right)^{-r_\theta},$$
$$\mathtt{SK}_{3,\theta} := D_3 g_2^{r_\theta}.$$

Otherwise, $\mathcal{B}$ chooses $r_\theta \xleftarrow{\$} \mathbb{Z}_p$ and computes

$$\mathtt{SK}_{1,\theta} := (g_2^{y_2})^{r_\theta},\ \mathtt{SK}'_{1,\theta} := P_\theta \left((g_2^{y_1})^{\mathtt{I}} g_2^{y_3}\right)^{r_\theta},$$
$$\mathtt{SK}_{2,\theta} := (g_2^{x_2})^{-r_\theta},\ \mathtt{SK}'_{2,\theta} := P_\theta \left((g_2^{x_1})^{\mathtt{I}} g_2^{x_3}\right)^{-r_\theta},$$
$$\mathtt{SK}_{3,\theta} := g_2^{r_\theta}.$$

It stores and outputs $sk_{\mathtt{I}} := \{(\mathtt{SK}_{1,\theta}, \mathtt{SK}'_{1,\theta}, \mathtt{SK}_{2,\theta}, \mathtt{SK}'_{2,\theta}, \mathtt{SK}_{3,\theta})\}_{\theta \in \mathsf{Path}(\mathtt{BT},\eta)}$.

***DKGen* oracle:** $\mathcal{B}$ creates and stores $sk_{\mathtt{I}}$ as above if $\mathtt{I}$ is first issued to the *SKGen* and *DKGen* oracles (otherwise, $\mathcal{B}$ uses the stored $sk_{\mathtt{I}}$), and runs DKGen algorithm. Note that $\mathcal{A}$ had to issue $\mathtt{T}$ to the *KeyUp* oracle before issuing the decryption query, and hence $ku_{\mathtt{T}}$ was already generated at that time. It outputs $dk_{\mathtt{I},\mathtt{T}}$.

Case $j = i^*$: Then, $\mathcal{B}$ regards the received identity $\mathtt{I}$ as a target identity, and creates a secret key for $\mathtt{I}^* := \mathtt{I}$ as follows. $\mathcal{B}$ first stores $\mathtt{I}^*$ in η^*.

***SKGen* oracle:** For each node $\theta \in \mathsf{Path}(\mathtt{BT}, \eta^*)$, $\mathcal{B}$ recalls P_θ if it was defined. Otherwise, it chooses $P_\theta \xleftarrow{\$} \mathbb{G}_2$ and stores P_θ in the node θ. For $\theta \in \mathsf{Path}(\mathtt{BT}, \eta^*)$, $\mathcal{B}$ chooses $r_\theta \xleftarrow{\$} \mathbb{Z}_p$ and computes

$$\mathrm{SK}_{1,\theta} := (g_2^{y_2})^{r_\theta},\ \mathrm{SK}'_{1,\theta} := P_\theta\left((g_2^{y_1})^{\mathtt{I}} g_2^{y_3}\right)^{r_\theta},$$
$$\mathrm{SK}_{2,\theta} := (g_2^{x_2})^{-r_\theta},\ \mathrm{SK}'_{2,\theta} := P_\theta\left((g_2^{x_1})^{\mathtt{I}} g_2^{x_3}\right)^{-r_\theta},$$
$$\mathrm{SK}_{3,\theta} := g_2^{r_\theta}.$$

It outputs $sk_{\mathtt{I}^*} := \{(\mathrm{SK}_{1,\theta}, \mathrm{SK}'_{1,\theta}, \mathrm{SK}_{2,\theta}, \mathrm{SK}'_{2,\theta}, \mathrm{SK}_{3,\theta})\}_{\theta\in\mathsf{Path}(\mathtt{BT},\eta)}$.

***DKGen* oracle:** $\mathcal{B}$ creates and stores $sk_{\mathtt{I}}$ as above if $\mathtt{I}$ is first issued to the *SKGen* and *DKGen* oracles (otherwise, $\mathcal{B}$ uses the stored $sk_{\mathtt{I}}$), and runs DKGen algorithm.

Case $j > i^*$: If $\mathtt{I} \neq \mathtt{I}^*$, then $\mathcal{B}$ performs the same procedure in the case $j < i^*$. Otherwise, $\mathcal{B}$ does the same process in the case $j = i^*$.

SKGen and *DKGen* oracles for the type-1-b adversary. The type-1-b adversary $\mathcal{A}$ issues the target identity $\mathtt{I}^*$ only after the challenge phase. Therefore, $\mathcal{B}$ already knows what identity is a target one when $\mathcal{A}$ sends the secret or decryption key query for $\mathtt{I}^*$ to $\mathcal{B}$. We show how $\mathcal{B}$ returns a secret key $sk_{\mathtt{I}}$ or a decryption key $dk_{\mathtt{I},\mathtt{T}}$ as follows.

Case $\mathtt{I} \neq \mathtt{I}^*$: $\mathcal{B}$ performs the same procedure in the case $j < i^*$ of the simulation for the type-1-a adversary.
Case $\mathtt{I} = \mathtt{I}^*$: $\mathcal{B}$ performs the same procedure in the case $j = i^*$ of the simulation for the type-1-a adversary.

The rest of the simulations is the same for the both of the type-1-a and type-1-b adversaries.

KeyUp oracle. When $\mathcal{B}$ receives a query $\mathtt{T}$ from $\mathcal{A}$, for each node $\theta \in \mathsf{KUNode}(\mathtt{BT}, RL, \mathtt{T})$, $\mathcal{B}$ recalls P_θ if it was defined. Otherwise, it chooses $P_\theta \xleftarrow{\$} \mathbb{G}_2$ and stores P_θ in the node θ. For $\theta \in \mathsf{KUNode}(\mathtt{BT}, RL, \mathtt{T})$, if $\theta \notin \mathsf{Path}(\mathtt{BT}, \eta^*)$, $\mathcal{B}$ then chooses $s_\theta \xleftarrow{\$} \mathbb{Z}_p$ and computes

$$\mathrm{KU}'_{1,\theta} := P_\theta^{-1}\left((g_2^{y_4})^{\mathtt{T}} g_2^{y_5}\right)^{s_\theta},\ \mathrm{KU}'_{2,\theta} := P_\theta^{-1}\left((g_2^{x_4})^{\mathtt{T}} g_2^{x_5}\right)^{-s_\theta},\ \mathrm{KU}_{3,\theta} := g_2^{s_\theta}.$$

Otherwise, $\mathcal{B}$ then chooses $s_\theta \xleftarrow{\$} \mathbb{Z}_p$ and computes

$$\mathrm{KU}'_{1,\theta} := P_\theta^{-1}((g_2^{y_4})^{\mathtt{T}} g_2^{y_5})^{s_\theta} (g_2^{\frac{1}{\beta}})^{-\frac{\mathtt{T}\tilde{y}+\hat{y}}{\mathtt{T}-\mathtt{T}^*}},$$
$$\mathrm{KU}'_{2,\theta} := P_\theta^{-1}((g_2^{x_4})^{\mathtt{T}} g_2^{x_5})^{-s_\theta} (g_2^{\frac{1}{\beta}})^{\frac{\mathtt{T}\tilde{x}+\hat{x}}{\mathtt{T}-\mathtt{T}^*}},\ \mathrm{KU}_{3,\theta} := g_2^{s_\theta} (g_2^{\frac{1}{\beta}})^{-\frac{1}{\mathtt{T}-\mathtt{T}^*}}.$$

Note that the above can be always computed since there exists no node θ such that $\theta \in \mathsf{Path}(\mathtt{BT}, \eta^*)$ and $\theta \in \mathsf{KUNode}(\mathtt{BT}, RL, \mathtt{T}^*)$ since $sk_{\mathtt{I}^*}$ is already revoked before $\mathtt{T}^*$. It finally outputs $ku_{\mathtt{T}} := \{(\mathrm{KU}'_{1,\theta}, \mathrm{KU}'_{2,\theta}, \mathrm{KU}_{3,\theta})\}_{\theta\in\mathsf{KUNode}(\mathtt{BT},RL,\mathtt{T})}$.

The simulation goes well since it holds that

$$\begin{aligned}
((g_2^{y_4})^{\mathtt{T}} g_2^{y_5})^{s_\theta} (g_2^{\frac{1}{\beta}})^{-\frac{\mathtt{T}\tilde{y}+\hat{y}}{\mathtt{T}-\mathtt{T}^*}} &= ((g_2^{\beta y_0+\tilde{y}})^{\mathtt{T}} g_2^{-\mathtt{T}^*\beta y_0+\hat{y}})^{s_\theta} g_2^{-\frac{\mathtt{T}\tilde{y}+\hat{y}}{(\mathtt{T}-\mathtt{T}^*)\beta}} \\
&= g_2^{y_0} (g_2^{(\mathtt{T}-\mathtt{T}^*)\beta y_0+\mathtt{T}\tilde{y}+\hat{y}})^{s_\theta} g_2^{-\frac{\mathtt{T}\tilde{y}+\hat{y}}{(\mathtt{T}-\mathtt{T}^*)\beta}} g_2^{-y_0} \\
&= g_2^{y_0} (g_2^{(\mathtt{T}-\mathtt{T}^*)\beta y_0+\mathtt{T}\tilde{y}+\hat{y}})^{s_\theta} \cdot (g_2^{(\mathtt{T}-\mathtt{T}^*)\beta y_0+\mathtt{T}\tilde{y}+\hat{y}})^{-\frac{1}{(\mathtt{T}-\mathtt{T}^*)\beta}} \\
&= g_2^{y_0} (g_2^{(\mathtt{T}-\mathtt{T}^*)\beta y_0+\mathtt{T}\tilde{y}+\hat{y}})^{s_\theta-\frac{1}{(\mathtt{T}-\mathtt{T}^*)\beta}} \\
&= g_2^{y_0} (g_2^{(\mathtt{T}-\mathtt{T}^*)\beta y_0+\mathtt{T}\tilde{y}+\hat{y}})^{s'_\theta} \\
&= g_2^{y_0} ((g_2^{y_4})^{\mathtt{T}} g_2^{y_5})^{s'_\theta},
\end{aligned}$$

$$\begin{aligned}
((g_2^{x_4})^{\mathtt{T}} g_2^{x_5})^{-s_\theta} (g_2^{\frac{1}{\beta}})^{\frac{\mathtt{T}\tilde{x}+\hat{x}}{\mathtt{T}-\mathtt{T}^*}} &= ((g_2^{\beta x_0+\tilde{x}})^{\mathtt{T}} g_2^{-\mathtt{T}^*\beta x_0+\hat{x}})^{-s_\theta} g_2^{\frac{\mathtt{T}\tilde{x}+\hat{x}}{(\mathtt{T}-\mathtt{T}^*)\beta}} \\
&= g_2^{-x_0} (g_2^{(\mathtt{T}-\mathtt{T}^*)\beta x_0+\mathtt{T}\tilde{x}+\hat{x}})^{-s_\theta} g_2^{\frac{\mathtt{T}\tilde{x}+\hat{x}}{(\mathtt{T}-\mathtt{T}^*)\beta}} g_2^{x_0} \\
&= g_2^{-x_0} (g_2^{(\mathtt{T}-\mathtt{T}^*)\beta x_0+\mathtt{T}\tilde{x}+\hat{x}})^{-s_\theta} \cdot (g_2^{(\mathtt{T}-\mathtt{T}^*)\beta x_0+\mathtt{T}\tilde{x}+\hat{x}})^{\frac{1}{(\mathtt{T}-\mathtt{T}^*)\beta}} \\
&= g_2^{-x_0} (g_2^{(\mathtt{T}-\mathtt{T}^*)\beta x_0+\mathtt{T}\tilde{x}+\hat{x}})^{-s_\theta+\frac{1}{(\mathtt{T}-\mathtt{T}^*)\beta}} \\
&= g_2^{-x_0} (g_2^{(\mathtt{T}-\mathtt{T}^*)\beta x_0+\mathtt{T}\tilde{x}+\hat{x}})^{-s'_\theta} \\
&= g_2^{-x_0} ((g_2^{x_4})^{\mathtt{T}} g_2^{x_5})^{-s'_\theta},
\end{aligned}$$

$$g_2^{s_\theta}(g_2^{\frac{1}{\beta}})^{-\frac{1}{\mathtt{T}-\mathtt{T}^*}} = g_2^{s_\theta-\frac{1}{(\mathtt{T}-\mathtt{T}^*)\beta}} = g_2^{s'_\theta},$$

where $s'_\theta = s_\theta - \frac{1}{(\mathtt{T}-\mathtt{T}^*)\beta}$.

Challenge. When $\mathcal{B}$ receives $(M_0^*, M_1^*, \mathtt{I}^*, \mathtt{T}^*)$ from $\mathcal{A}$, then it sends $(M_0^*, M_1^*, \mathtt{I}^*)$ to the challenger in the IND-ID-CPA game of Π_{JR}. After receiving $(C_0^*, C_1^*, C_2^*, C_3^*, \mathtt{tag}^*)$ from the challenger, $\mathcal{B}$ sets $C_4^* := (C_2^*)^{-(\mathtt{T}^*\tilde{x}+\hat{x})}(C_1^*)^{\mathtt{T}^*\tilde{y}+\hat{y}}$. This is well-formed since

$$C_4^* = (v_1^{\mathtt{T}^*}\hat{v}_1)^t = g_1^{t(-\mathtt{T}^*\tilde{x}\alpha+\mathtt{T}^*\tilde{y}-\hat{x}\alpha+\hat{y})} = g_1^{-t\alpha(\mathtt{T}^*\tilde{x}+\hat{x})+t(\mathtt{T}^*\tilde{y}+\hat{y})}.$$

$\mathcal{B}$ sends $(C_0^*, C_1^*, C_2^*, C_3^*, C_4^*, \mathtt{tag}^*)$ to $\mathcal{A}$.

When $\mathcal{A}$ outputs b', then $\mathcal{B}$ transfer it. We can show the distribution of all the above transcriptions between $\mathcal{A}$ and $\mathcal{B}$ is identical to the real experiment from the viewpoint of $\mathcal{A}$ as in [44, Claim 1], and therefore we omit it.

Type-2-a and type-2-b adversary. The difference of simulations between the type-2-a and type-2-b adversaries is also a way of simulating the *DKGen* oracle. Before describing the difference, we show how $\mathcal{B}$ simulates the *SKGen* and *KeyUp* oracles.

SKGen oracle. $\mathcal{B}$ first transfers $\mathtt{I}$ to the *KeyGen* oracle of the IND-ID-CPA game of Π_{JR}, and gets $SK_{\mathtt{I}} := (D_1, D'_1, D_2, D'_2, D_3)$ if $\mathcal{B}$ does not have it. $\mathcal{B}$ randomly chooses an unassigned leaf η from $\mathtt{BT}$ and stores $\mathtt{I}$ in the node η if it is not done. For each node $\theta \in \mathsf{Path}(\mathtt{BT}, \eta)$, $\mathcal{B}$ recalls P_θ if it was defined. Otherwise, it chooses $P_\theta \xleftarrow{\$} \mathbb{G}_2$ and stores P_θ in the node θ. For $\theta \in \mathsf{Path}(\mathtt{BT}, \eta)$, $\mathcal{B}$ chooses $r_\theta \xleftarrow{\$} \mathbb{Z}_p$ and computes

$$\mathtt{SK}_{1,\theta} := D_1(g_2^{y_2})^{r_\theta},\ \ \mathtt{SK}'_{1,\theta} := P_\theta D'_1\left((g_2^{y_1})^{\mathtt{I}}g_2^{y_3}\right)^{r_\theta},$$

$$\mathtt{SK}_{2,\theta} := D_2(g_2^{x_2})^{-r_\theta},\ \ \mathtt{SK}'_{2,\theta} := P_\theta D'_2\left((g_2^{x_1})^{\mathtt{I}}g_2^{x_3}\right)^{-r_\theta},\ \ \mathtt{SK}_{3,\theta} := D_3 g_2^{r_\theta}.$$

It stores and outputs $sk_{\mathtt{I}} := \{(\mathtt{SK}_{1,\theta}, \mathtt{SK}'_{1,\theta}, \mathtt{SK}_{2,\theta}, \mathtt{SK}'_{2,\theta}, \mathtt{SK}_{3,\theta})\}_{\theta\in\mathsf{Path}(\mathtt{BT},\eta)}$.

KeyUp oracle. When $\mathcal{B}$ receives a query $\mathtt{T}$ from $\mathcal{A}$, for each node $\theta \in \mathsf{KUNode}(\mathtt{BT}, RL, \mathtt{T})$, $\mathcal{B}$ recalls P_θ if it was defined. Otherwise, it chooses $P_\theta \xleftarrow{\$} \mathbb{G}_2$ and stores P_θ in the node θ. For $\theta \in \mathsf{KUNode}(\mathtt{BT}, RL, \mathtt{T})$, $\mathcal{B}$ chooses $s_\theta \xleftarrow{\$} \mathbb{Z}_p$ and computes

$$\mathtt{KU}'_{1,\theta} := P_\theta^{-1}\left((g_2^{y_4})^{\mathtt{T}}g_2^{y_5}\right)^{s_\theta},\ \ \mathtt{KU}'_{2,\theta} := P_\theta^{-1}\left((g_2^{x_4})^{\mathtt{T}}g_2^{x_5}\right)^{-s_\theta},\ \ \mathtt{KU}_{3,\theta} := g_2^{s_\theta}.$$

It outputs $ku_{\mathtt{T}} := \{(\mathtt{KU}'_{1,\theta}, \mathtt{KU}'_{2,\theta}, \mathtt{KU}_{3,\theta})\}_{\theta\in\mathsf{KUNode}(\mathtt{BT},RL,\mathtt{T})}$.

DKGen oracle for the type-2-a adversary. Let q_d ($\le q_1$) be the maximum number of identities made queries to the *DKGen* oracle before the challenge phase. Suppose that $\mathcal{B}$ receives a j-th identity $\mathtt{I}$ as the decryption key query $(\mathtt{I}, \mathtt{T})$ from $\mathcal{A}$. $\mathcal{B}$ then returns a decryption key $dk_{\mathtt{I},\mathtt{T}}$ as follows.

Case $j < i^*$**:** $\mathcal{B}$ creates and stores $sk_{\mathtt{I}}$ as above if $\mathtt{I}$ is first queried to the *SKGen* and *DKGen* oracles (otherwise, $\mathcal{B}$ uses the stored $sk_{\mathtt{I}}$), and runs DKGen algorithm.

Case $j = i^*$**:** Then, $\mathcal{B}$ regards the received identity $\mathtt{I}$ as a target identity, and creates a decryption key for $\mathtt{I}^* := \mathtt{I}$ as follows. $\mathcal{B}$ chooses $r, s \xleftarrow{\$} \mathbb{Z}_p$ and computes

$$\mathtt{DK}_1 := (g_2^{y_2})^r,\ \ \mathtt{DK}'_{1,\theta} := \left((g_2^{y_1})^{\mathtt{I}^*}g_2^{y_3}\right)^r((g_2^{y_4})^{\mathtt{T}}g_2^{y_5})^s(g_2^{\frac{1}{\beta}})^{-\frac{\mathtt{T}\tilde{y}+\hat{y}}{\mathtt{T}-\mathtt{T}^*}},$$

$$\mathtt{DK}_2 := (g_2^{x_2})^{-r},\ \ \mathtt{DK}'_2 := \left((g_2^{x_1})^{\mathtt{I}^*}g_2^{x_3}\right)^{-r}((g_2^{x_4})^{\mathtt{T}}g_2^{x_5})^{-s}(g_2^{\frac{1}{\beta}})^{\frac{\mathtt{T}\tilde{x}+\hat{x}}{\mathtt{T}-\mathtt{T}^*}},$$

$$\mathtt{DK}_3 := g_2^r,\ \ \mathtt{DK}_4 := g_2^s(g_2^{\frac{1}{\beta}})^{-\frac{1}{\mathtt{T}-\mathtt{T}^*}}.$$

Case $j > i^*$**:** If $\mathtt{I} \neq \mathtt{I}^*$, then $\mathcal{B}$ performs the same procedure in the case $j < i^*$. Otherwise, $\mathcal{B}$ does the same process in the case $j = i^*$.

DKGen oracle for the type-2-b adversary. The type-2-b adversary $\mathcal{A}$ issues the target identity $\mathtt{I}^*$ only after challenge phase. Therefore, $\mathcal{B}$ does not have to guess which identity issued to the oracle is a target one. We show how $\mathcal{B}$ returns a decryption key $dk_{\mathtt{I},\mathtt{T}}$ as follows.

Case $\mathtt{I} \neq \mathtt{I}^*$**:** $\mathcal{B}$ performs the same procedure in the case $j < i^*$ of the simulation for the type-2-a adversary.
Case $\mathtt{I} = \mathtt{I}^*$**:** $\mathcal{B}$ performs the same procedure in the case $j = i^*$ of the simulation for the type-2-a adversary.

Challenge. $\mathcal{B}$ creates the challenge ciphertext as in the challenge phase for the type-1-a and type-1-b adversary.

When $\mathcal{A}$ outputs b', then $\mathcal{B}$ transfer it. We can also show the distribution of all the above transcriptions between $\mathcal{A}$ and $\mathcal{B}$ is identical to the real experiment from the viewpoint of $\mathcal{A}$ as in [44, Claim 2], and therefore we omit it.

We estimate the reduction loss. Let $\mathcal{S}$ be an event that $\mathcal{B}$ wins the IND-ID-CPA game of Π_{JR} (i.e., $b' = b$), $\mathcal{E}_1$ be an event that $\mathcal{B}$ correctly guesses the target time period, and $\mathcal{E}_2$ be an event that $\mathcal{B}$'s guess (k^*, i^*) is right, respectively. We then have

$$\begin{aligned}
Adv^{\mathsf{ID\text{-}CPA}}_{\Pi_{\mathrm{JR}},\mathcal{B}}(\lambda) &= \left|\Pr[\mathcal{S}] - \frac{1}{2}\right| \\
&= \left|\Pr[\mathcal{S} \wedge \mathcal{E}_1] + \Pr[\mathcal{S} \wedge \neg\mathcal{E}_1] - \frac{1}{2}\right| \\
&= \frac{1}{|\mathcal{T}|}\left|\Pr[\mathcal{S} \mid \mathcal{E}_1] - \frac{1}{2}\right| \\
&= \frac{1}{|\mathcal{T}|}\left|\Pr[\mathcal{S} \wedge \mathcal{E}_2 \mid \mathcal{E}_1] + \Pr[\mathcal{S} \wedge \neg\mathcal{E}_2 \mid \mathcal{E}_1] - \frac{1}{2}\right| \\
&= \frac{1}{2|\mathcal{T}|(q_1+1)}\left|\Pr[\mathcal{S} \mid \mathcal{E}_1 \wedge \mathcal{E}_2] - \frac{1}{2}\right| \\
&= \frac{1}{2|\mathcal{T}|(q_1+1)}\mathsf{Adv}^{\mathsf{RID\text{-}CPA}}_{\Pi,\mathcal{A}}(\lambda).
\end{aligned}$$

Therefore, we have

$$\mathsf{Adv}^{\mathsf{RID\text{-}CPA}}_{\Pi,\mathcal{A}}(\lambda) \leq 8|\mathcal{T}|(q_1+1)\mathsf{Adv}^{\mathsf{ADDH1}}_{\mathcal{G},\mathcal{B}}(\lambda) + 2|\mathcal{T}|q(q_1+1)Adv^{\mathsf{DDH2}}_{\mathcal{G},\mathcal{B}}(\lambda),$$

where q is the maximum number of queries issued to the *KeyGen* oracle in the IND-ID-CPA game of Π_{JR}. □

5. Extensions

As we have seen in earlier sections, our RIBE construction employs the Seo-Emura technique as a core technique. In this section, we show that our RIBE scheme can be easily extended to a CCA-secure scheme and variants of RIBE thanks to the Seo-Emura technique.

5.1. CCA security

Remark that the Ishida-Watanabe-Shikata scheme [16] achieves not only adaptive security with DKER over prime-order groups but also CCA security. They proposed two schemes. The first one employs the BCHK transformation [4], and the second one is constructed via the KEM/DEM framework. We notice that still the size of public parameter depends on the length of identity. Although their second construction relies on the underlying Kiltz-Galindo identity-based KEM [20], we can employ their first construction to construct a CCA-secure RIBE scheme with constant-size public parameter based on our RIBE scheme. The detailed construction of an RIBE scheme Π using a one-time signature (OTS) scheme $\Pi_{\mathrm{OTS}} = (\mathsf{KG}, \mathsf{Sign}, \mathsf{Ver})$ is as follows.[11]

– $\mathsf{Setup}(\lambda, N)$: It runs $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e) \leftarrow \mathcal{G}$. It chooses $x_0, y_0, x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4, x_5, y_5, x_{vk}, y_{vk} \xleftarrow{\$} \mathbb{Z}_p$ and $\alpha \xleftarrow{\$} \mathbb{Z}_p^{\times}$, and sets

$$\begin{aligned}
&z := e(g_1, g_2)^{-x_0\alpha + y_0},\ \ u_1 := g_1^{-x_1\alpha + y_1},\ \ w_1 := g_1^{-x_2\alpha + y_2},\ \ h_1 := g_1^{-x_3\alpha + y_3}, \\
&v_1 := g_1^{-x_4\alpha + y_4},\ \ \hat{v}_1 := g_1^{-x_5\alpha + y_5},\ \ \hat{u}_1 := g_1^{-x_{vk}\alpha + y_{vk}}.
\end{aligned}$$

Let $\mathtt{BT}$ be a binary tree that has N leaves, where N is a power of two for simplicity. It outputs

$$\begin{aligned}
mpk &:= (g_1, g_1^{\alpha}, u_1, w_1, h_1, v_1, \hat{v}_1, \hat{u}_1, g_2, \{g_2^{x_i}, g_2^{y_i}\}_{i=1}^{5}, g_2^{x_{vk}}, g_2^{y_{vk}}, z), \\
msk &:= (g_2^{y_0}, g_2^{-x_0}),
\end{aligned}$$

$st := \mathtt{BT}$, and $RL := \emptyset$.

[11] Formal descriptions of CCA security and OTS are given in Appendix A.

– $\mathsf{SKGen}(st, \mathtt{I})$: Parse st as $\mathtt{BT}$. It randomly chooses an unassigned leaf η from $\mathtt{BT}$, and stores $\mathtt{I}$ in the node η. For each node $\theta \in \mathsf{Path}(\mathtt{BT}, \eta)$, it recalls P_θ if it was defined. Otherwise, it chooses $P_\theta \xleftarrow{\$} \mathbb{G}_2$ and stores P_θ in the node θ. Then, it chooses $r_\theta \xleftarrow{\$} \mathbb{Z}_p$ and it computes

$$\mathtt{SK}_{1,\theta} := (g_2^{y_2})^{r_\theta},\ \mathtt{SK}'_{1,\theta} := P_\theta\left((g_2^{y_1})^{\mathtt{I}} g_2^{y_3}\right)^{r_\theta},\ \mathtt{SK}''_{1,\theta} := (g_2^{y_{vk}})^{r_\theta},$$
$$\mathtt{SK}_{2,\theta} := (g_2^{x_2})^{-r_\theta},\ \mathtt{SK}'_{2,\theta} := P_\theta\left((g_2^{x_1})^{\mathtt{I}} g_2^{x_3}\right)^{-r_\theta},\ \mathtt{SK}''_{2,\theta} := (g_2^{x_{vk}})^{-r_\theta},$$
$$\mathtt{SK}_{3,\theta} := g_2^{r_\theta}.$$

It outputs $sk_{\mathtt{I}} := \{(\mathtt{SK}_{1,\theta}, \mathtt{SK}'_{1,\theta}, \mathtt{SK}''_{1,\theta}, \mathtt{SK}_{2,\theta}, \mathtt{SK}'_{2,\theta}, \mathtt{SK}''_{2,\theta}, \mathtt{SK}_{3,\theta})\}_{\theta \in \mathsf{Path}(\mathtt{BT},\eta)}$.

– $\mathsf{KeyUp}(msk, st, RL, \mathtt{T})$: Parse msk as $(\mathtt{MK}_1, \mathtt{MK}_2)$. For each node $\theta \in \mathsf{KUNode}(\mathtt{BT}, RL, \mathtt{T})$, it recalls P_θ if it was defined. Otherwise, it chooses $P_\theta \xleftarrow{\$} \mathbb{G}_2$ and stores P_θ in the node θ. It chooses $s_\theta \xleftarrow{\$} \mathbb{Z}_p$ and computes

$$\mathtt{KU}'_{1,\theta} := P_\theta^{-1}\mathtt{MK}_1\left((g_2^{y_4})^{\mathtt{T}} g_2^{y_5}\right)^{s_\theta},$$
$$\mathtt{KU}'_{2,\theta} := P_\theta^{-1}\mathtt{MK}_2\left((g_2^{x_4})^{\mathtt{T}} g_2^{x_5}\right)^{-s_\theta},\ \mathtt{KU}_{3,\theta} := g_2^{s_\theta}.$$

It outputs $ku_{\mathtt{T}} := \{(\mathtt{KU}'_{1,\theta}, \mathtt{KU}'_{2,\theta}, \mathtt{KU}_{3,\theta})\}_{\theta \in \mathsf{KUNode}(\mathtt{BT}, RL, \mathtt{T})}$.

– $\mathsf{DKGen}(sk_{\mathtt{I}}, ku_{\mathtt{T}})$: Parse $sk_{\mathtt{I}}$ and $ku_{\mathtt{T}}$ as $\{(\mathtt{SK}_{1,\theta}, \mathtt{SK}'_{1,\theta}, \mathtt{SK}_{2,\theta}, \mathtt{SK}'_{2,\theta}, \mathtt{SK}_{3,\theta})\}_{\theta \in \Theta_{\mathtt{SK}}}$ and $\{(\mathtt{KU}'_{1,\theta}, \mathtt{KU}'_{2,\theta}, \mathtt{KU}_{3,\theta})\}_{\theta \in \Theta_{\mathtt{KU}}}$, respectively. It outputs $\perp$ if $\Theta_{\mathtt{SK}} \cap \Theta_{\mathtt{KU}} = \emptyset$. Otherwise, for some $\theta \in \Theta_{\mathtt{SK}} \cap \Theta_{\mathtt{KU}}$, it computes as follows. It chooses $R, S \xleftarrow{\$} \mathbb{Z}_p$ and computes

$$\mathtt{DK}_1 := \mathtt{SK}_{1,\theta}(g_2^{y_2})^R,$$
$$\mathtt{DK}'_1 := \mathtt{SK}'_{1,\theta}\mathtt{KU}'_{1,\theta}\left((g_2^{y_1})^{\mathtt{I}} g_2^{y_3}\right)^R\left((g_2^{y_4})^{\mathtt{T}} g_2^{y_5}\right)^S,$$
$$\mathtt{DK}''_1 := \mathtt{SK}''_{1,\theta}(g_2^{y_{vk}})^R,$$
$$\mathtt{DK}_2 := \mathtt{SK}_{2,\theta}(g_2^{x_2})^{-R},$$
$$\mathtt{DK}'_2 := \mathtt{SK}'_{2,\theta}\mathtt{KU}'_{2,\theta}\left((g_2^{x_1})^{\mathtt{I}} g_2^{x_3}\right)^{-R}\left((g_2^{x_4})^{\mathtt{T}} g_2^{x_5}\right)^{-S},$$
$$\mathtt{DK}''_2 := \mathtt{SK}''_{2,\theta}(g_2^{x_{vk}})^{-R},$$
$$\mathtt{DK}_3 := \mathtt{SK}_{3,\theta} g_2^R,\ \mathtt{DK}_4 := \mathtt{KU}_{3,\theta} g_2^S.$$

It outputs $dk_{\mathtt{I},\mathtt{T}} := (\mathtt{DK}_1, \mathtt{DK}'_1, \mathtt{DK}''_1, \mathtt{DK}_2, \mathtt{DK}'_2, \mathtt{DK}''_2, \mathtt{DK}_3, \mathtt{DK}_4)$.

– $\mathsf{Enc}(M, \mathtt{I}, \mathtt{T})$: It generates $(osk, ovk) \leftarrow \mathsf{KG}(\lambda)$.[12] It chooses $t, \mathtt{tag} \xleftarrow{\$} \mathbb{Z}_p$. For $M \in \mathbb{G}_T$, it computes

$$C_0 := M \cdot z^t,\ C_1 := g_1^t,\ C_2 := (g_1^\alpha)^t,$$
$$C_3 := \left(u_1^{\mathtt{I}} w_1^{\mathtt{tag}} h_1 \hat{u}_1^{ovk}\right)^t,\ C_4 := (v_1^{\mathtt{T}} \hat{v}_1)^t.$$

It also computes $\sigma \leftarrow \mathsf{Sign}(osk, (C_0, C_1, C_2, C_3, C_4, \mathtt{tag}))$ It outputs $C_{\mathtt{I},\mathtt{T}} := (ovk, C_0, C_1, C_2, C_3, C_4, \mathtt{tag}, \sigma)$.

– $\mathsf{Dec}(dk_{\mathtt{I},\mathtt{T}}, C_{\mathtt{I},\mathtt{T}})$: Parse $dk_{\mathtt{I},\mathtt{T}}$ and $C_{\mathtt{I},\mathtt{T}}$ as $(\mathtt{DK}_1, \mathtt{DK}'_1, \mathtt{DK}_2, \mathtt{DK}'_2, \mathtt{DK}_3, \mathtt{DK}_4)$ and $(C_0, C_1, C_2, C_3, C_4, \mathtt{tag})$, respectively. If $\mathsf{Ver}(ovk, (C_0, C_1, C_2, C_3, C_4, \mathtt{tag}), \sigma) \to 1$, then it computes

$$M = \frac{C_0 e(C_3, \mathtt{DK}_3) e(C_4, \mathtt{DK}_4)}{e(C_1, \mathtt{DK}_1^{\mathtt{tag}}\mathtt{DK}'_1(\mathtt{DK}''_1)^{vk}) e(C_2, \mathtt{DK}_2^{\mathtt{tag}}\mathtt{DK}'_2(\mathtt{DK}''_2)^{vk})}.$$

– $\mathsf{Revoke}(\mathtt{I}, \mathtt{T}, RL, st)$: Output $RL := RL \cup \{(\mathtt{I}, \mathtt{T})\}$.

The correctness obviously holds, therefore we omit the description.

Theorem 4. *If the* ADDH1 *and* DDH2 *assumptions hold and the underlying OTS scheme* Π_{OTS} *is* sUF-OT *secure, then the resulting RIBE scheme* Π *is* IND-RID-CCA *secure.*

Proof sketch. This proof basically follows the proof of Theorem 3. More precisely, we construct a PPT algorithm $\mathcal{B}$ that breaks the IND-ID-CPA security of the 2-level modified Jutla-Roy HIBE scheme $\widehat{\Pi}_{\mathsf{JR}}$[13] by using a PPT algorithm $\mathcal{A}$ that

[12] We assume that ovk is appropriately encoded when used for group operations.
[13] The modified Jutla-Roy HIBE scheme is given in Appendix A.4.

breaks the IND-RID-CCA security of Π. Although the main task is to simulate a decryption oracle, it can be done by the technique in [4]. Specifically, let $\mathtt{I}^*$ and $\mathtt{T}^*$ be challenge identity and time period, and $(ovk^*, C_0^*, C_1^*, C_2^*, C_3^*, C_4^*, \mathtt{tag}^*, \sigma^*)$ be a challenge ciphertext of Π. If $\mathcal{A}$ issues a query $(\mathtt{I}^*, \mathtt{T}^*, C_{\mathtt{I}^*\mathtt{T}^*})$ such that $C_{\mathtt{I}^*\mathtt{T}^*}$ contains ovk $(\neq ovk^*)$ to the decryption oracle, $\mathcal{B}$ can answer it by issuing $(\mathtt{I}^*, ovk)$ to the *KeyGen* oracle and getting $SK_{\mathtt{I}^*,ovk}$. Otherwise, we can construct a PPT algorithm $\mathcal{F}$ that breaks the sUF-OT security of Π_{OTS} by using $\mathcal{A}$ that issues $(\mathtt{I}^*, \mathtt{T}^*, C_{\mathtt{I}^*\mathtt{T}^*})$ such that $C_{\mathtt{I}^*\mathtt{T}^*}$ contains ovk^* and $\mathsf{Ver}(ovk^*, C = (C_0, C_1, C_2, C_3, C_4, \mathtt{tag}), \sigma) \to 1$, since (C, σ) is a successful forgery pair of Π_{OTS}. □

5.2. Server-aided RIBE

Qin et al. [35] proposed server-aided RIBE (SRIBE). In their scheme, almost all of the workloads on users are delegated to an untrusted server who does not have any secret value. More specifically, the server partially decrypts ciphertexts for non-revoked users with *transformation keys* created from key update, and the users can decrypt the partially-decrypted ciphertexts with their secret keys. Therefore, non-revoked users do not need to pay attention to when key update is broadcasted, and Qin et al.'s scheme achieves constant-size secret keys.

Formally, SRIBE $\Pi_{\mathrm{SA}} = (\mathsf{Setup}, \mathsf{SKGen}, \mathsf{KeyUp}, \mathsf{TKGen}, \mathsf{DKGen}, \mathsf{Enc}, \mathsf{PartDec}, \mathsf{Dec}, \mathsf{Revoke})$ is defined as follows.[14] We omit a public parameter in the input of all algorithms except for the Setup algorithm for simplicity.

- $(mpk, msk, RL, st) \leftarrow \mathsf{Setup}(\lambda, N)$: Same as ordinary RIBE (see Section 2).
- $(pk_{\mathtt{I}}, sk_{\mathtt{I}}, st) \leftarrow \mathsf{SKGen}(msk, \mathtt{I}, st)$: An algorithm for users' key generation. It takes msk, an identity $\mathtt{I} \in \mathcal{I}$, and st as input and outputs a public/secret-key pair $(pk_{\mathtt{I}}, sk_{\mathtt{I}})$ and updated state information st.
- $ku_{\mathtt{T}} \leftarrow \mathsf{KeyUp}(msk, st, RL, \mathtt{T})$: Same as ordinary RIBE (see Section 2).
- $tk_{\mathtt{I},\mathtt{T}}$ or $\perp \leftarrow \mathsf{TKGen}(pk_{\mathtt{I}}, ku_{\mathtt{T}})$: A probabilistic algorithm for transformation key generation. It takes $pk_{\mathtt{I}}$ and $ku_{\mathtt{T}}$ as input and then outputs a transformation key $tk_{\mathtt{I},\mathtt{T}}$ at $\mathtt{T}$ or $\perp$ if $\mathtt{I}$ has been revoked by $\mathtt{T}$.
- $dk_{\mathtt{I},\mathtt{T}} \leftarrow \mathsf{DKGen}(sk_{\mathtt{I}}, \mathtt{T})$: A probabilistic algorithm for decryption key generation. It takes $sk_{\mathtt{I}}$ and $\mathtt{T}$ as input and then outputs a decryption key $dk_{\mathtt{I},\mathtt{T}}$ at $\mathtt{T}$.
- $C_{\mathtt{I},\mathtt{T}} \leftarrow \mathsf{Enc}(M, \mathtt{I}, \mathtt{T})$: Same as ordinary RIBE (see Section 2).
- $ct_{\mathtt{I},\mathtt{T}}$ or $\perp \leftarrow \mathsf{PartDec}(tk_{\mathtt{I},\mathtt{T}}, C_{\mathtt{I},\mathtt{T}})$: A deterministic algorithm for partial decryption. It takes $tk_{\mathtt{I},\mathtt{T}}$ and $C_{\mathtt{I},\mathtt{T}}$ as input and then outputs a partially-decrypted ciphertext $ct_{\mathtt{I},\mathtt{T}}$ or $\perp$.
- M or $\perp \leftarrow \mathsf{Dec}(dk_{\mathtt{I},\mathtt{T}}, ct_{\mathtt{I},\mathtt{T}})$: A deterministic algorithm for decryption. It takes $dk_{\mathtt{I},\mathtt{T}}$ and $ct_{\mathtt{I},\mathtt{T}}$ as input and then outputs M or $\perp$.
- $RL \leftarrow \mathsf{Revoke}(\mathtt{I}, \mathtt{T}, RL, st)$: Same as ordinary RIBE (see Section 2).

In the above model, we require that Π_{SA} meets the following correctness property: For all security parameter $\lambda \in \mathbb{N}$, all $(mpk, msk, RL, st) \leftarrow \mathsf{Setup}(\lambda, N)$, all $M \in \mathcal{M}$, all $\mathtt{I} \in \mathcal{I}$, all $\mathtt{T} \in \mathcal{T}$, if $\mathtt{I}$ is not revoked on $\mathtt{T} \in \mathcal{T}$, it holds that $M = \mathsf{Dec}(dk_{\mathtt{I},\mathtt{T}}, \mathsf{PartDec}(tk_{\mathtt{I},\mathtt{T}}, \mathsf{Enc}(M, \mathtt{I}, \mathtt{T})))$, where $(pk_{\mathtt{I}}, sk_{\mathtt{I}}, st) \leftarrow \mathsf{SKGen}(msk, \mathtt{I}, st)$, $dk_{\mathtt{I},\mathtt{T}} \leftarrow \mathsf{DKGen}(sk_{\mathtt{I}}, \mathtt{T})$, and $tk_{\mathtt{I},\mathtt{T}} \leftarrow \mathsf{TKGen}(pk_{\mathtt{I}}, \mathsf{KeyUp}(msk, st, RL, \mathtt{T}))$.

We describe an SRIBE version of IND-RID-CPA, which is called IND-SRID-CPA. Let $\mathcal{A}$ be a PPT adversary, and $\mathcal{A}$'s advantage against IND-SRID-CPA security is defined by

$$\mathsf{Adv}^{\mathsf{SRID\text{-}CPA}}_{\Pi_{\mathrm{SA}},\mathcal{A}}(\lambda, N) := \left| \Pr\left[b' = b \;\middle|\; \begin{array}{l} (mpk, msk, RL, st) \leftarrow \mathsf{Setup}(\lambda, N), \\ (M_0^*, M_1^*, \mathtt{I}^*, \mathtt{T}^*, state) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{find}, mpk), \\ b \xleftarrow{\$} \{0,1\}, \\ C^*_{\mathtt{I}^*,\mathtt{T}^*} \leftarrow \mathsf{Enc}(M_b^*, \mathtt{I}^*, \mathtt{T}^*), \\ b' \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{guess}, C^*_{\mathtt{I}^*,\mathtt{T}^*}, state) \end{array} \right] - \frac{1}{2} \right|.$$

Here, $\mathcal{O}$ is a set of oracles $\{PKGen(\cdot), SKGen(\cdot), KeyUp(\cdot), Revoke(\cdot,\cdot), DKGen(\cdot,\cdot)\}$ defined as follows.

PKGen$(\cdot)$**:** For a query $\mathtt{I} \in \mathcal{I}$, it returns $pk_{\mathtt{I}}$ if it is already generated. Otherwise, it stores and returns $pk_{\mathtt{I}}$ by running $\mathsf{SKGen}(msk, \mathtt{I}, st)$.

SKGen$(\cdot)$**:** For a query $\mathtt{I} \in \mathcal{I}$, it returns $sk_{\mathtt{I}}$ if it is already generated. Otherwise, it stores and returns $sk_{\mathtt{I}}$ by running $\mathsf{SKGen}(msk, \mathtt{I}, st)$.

KeyUp$(\cdot)$**:** For a query $\mathtt{T} \in \mathcal{T}$, it stores and returns $\mathsf{KeyUp}(msk, RL, st, \mathtt{T})$.

Revoke$(\cdot,\cdot)$**:** For a query $(\mathtt{I}, \mathtt{T}) \in \mathcal{I} \times \mathcal{T}$, it updates a revocation list RL by running $\mathsf{Revoke}(\mathtt{I}, \mathtt{T}, RL, st)$.

DKGen$(\cdot,\cdot)$**:** For a query $(\mathtt{I}, \mathtt{T}) \in \mathcal{I} \times \mathcal{T}$, it finds $sk_{\mathtt{I}}$ generated by the *SKGen* oracle (if it has not been generated yet, *DKGen* generates it by running $\mathsf{SKGen}(msk, \mathtt{I}, st)$). *DKGen* returns $\mathsf{DKGen}(sk_{\mathtt{I}}, \mathtt{T})$.

[14] We here simplify the original algorithms of SRIBE [35]. For instance, $pk_{\mathtt{I}}$ and $sk_{\mathtt{I}}$ are generated separately in the original model. Note that our construction, which will be shown later, is also secure in the original model.

$\mathcal{A}$ can access the above oracles under the same restrictions as the IND-RID-CPA game of ordinary RIBE.

Definition 5 *(IND-SRID-CPA).* An SRIBE scheme Π_{SA} is said to be IND-SRID-CPA secure if for all PPT adversaries $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{SRID\text{-}CPA}}_{\Pi_{\mathsf{SA}},\mathcal{A}}(\lambda, N)$ is negligible in λ.

Remark 1 *(On transformation key oracles).* As mentioned by Qin et al. [35], the server is untrusted in the sense that it does not possess any secret information. That is, transformation keys are generated by public values only. Thus, we do not need to prepare a transformation-key oracle in our security model, which is the same as ones in the original paper [35] and its subsequent work [34].

We briefly explain Qin et al.'s SRIBE scheme [35] as follows. A master secret key of the Seo-Emura RIBE scheme is divided to two values via two-out-of-two secret sharing, say α and β. A ciphertext has two blinding factors according to α and β such as $M \cdot e(g,g)^{\alpha} e(g,g)^{\beta}$. KGC computes a secret key of the Seo-Emura RIBE scheme for $\mathtt{I}$ by using the master secret α, and sends it to the server as $pk_{\mathtt{I}}$. Since $pk_{\mathtt{I}}$ is generated by employing the CS method, the size of $pk_{\mathtt{I}}$ is $O(r\log(N/r))$. Moreover, KGC issues a long-term secret key $sk_{\mathtt{I}}$ to a user $\mathtt{I}$ by using the master secret β. It is particular worth noting that the size of $sk_{\mathtt{I}}$ is constant, and $sk_{\mathtt{I}}$ is independent of time t. Moreover, the user can compute the decryption key, say $dk_{\mathtt{I},t}$ which removes the β-part blinding factor of a ciphertext, from $sk_{\mathtt{I}}$ and t regardless of whether he/she is revoked or not. At time $\mathtt{T}$, KGC computes key update information $ku_{\mathtt{T}}$, which is sent to the server *via public channels*. If a user $\mathtt{I}$ is not revoked at time $\mathtt{T}$, then the server can compute $tk_{\mathtt{I},\mathtt{T}}$, which removes the α-part blinding factor $e(g,g)^{\alpha}$ of a ciphertext, from $pk_{\mathtt{I}}$ and ku_t. The server partially decrypts a ciphertext by using $tk_{\mathtt{I},\mathtt{T}}$, and sends the result to the user $\mathtt{I}$. The user can obtain the plaintext by removing the β-part blinding factor $e(g,g)^{\beta}$ of the partially-decrypted ciphertext by using $dk_{\mathtt{I},\mathtt{T}}$.

This construction methodology can be employed to our RIBE scheme. Then, we can construct an SRIBE scheme with the same advantages of our RIBE scheme, i.e., constant-size public parameter and asymmetric pairing settings. More specifically, we modify our RIBE scheme in the following manner. In our SRIBE scheme, x and y are additionally chosen and (g_2^y, g_2^{-x}) are additionally contained in msk. This additional master keys are used for computing secret keys of users, and $-x\alpha + y$ has the role of β as above. The server partially decrypts a ciphertext by removing the blinding factor $e(g_1,g_2)^{(-x_0\alpha+y_0)t}$ as in our RIBE scheme, and the user can remove the remaining blind factor $e(g_1,g_2)^{(-x\alpha+y)t}$. The detailed construction is given below.

- $\mathsf{Setup}(\lambda, N)$: It runs $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e) \leftarrow \mathcal{G}$. It chooses $x, y, x_0, y_0, x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4, x_5, y_5 \xleftarrow{\$} \mathbb{Z}_p$ and $\alpha \xleftarrow{\$} \mathbb{Z}_p^{\times}$, and sets

$$z := e(g_1,g_2)^{-(x+x_0)\alpha+y+y_0},\ u_1 := g_1^{-x_1\alpha+y_1},\ w_1 := g_1^{-x_2\alpha+y_2},$$
$$h_1 := g_1^{-x_3\alpha+y_3},\ v_1 := g_1^{-x_4\alpha+y_4},\ \hat{v}_1 := g_1^{-x_5\alpha+y_5}.$$

  Let $\mathtt{BT}$ be a binary tree that has N leaves, where N is a power of two for simplicity. It outputs

$$mpk := (g_1, g_1^{\alpha}, u_1, w_1, h_1, v_1, \hat{v}_1, g_2, \{g_2^{x_i}, g_2^{y_i}\}_{i=1}^{5}, z),$$
$$msk := (g_2^{y_0}, g_2^{y}, g_2^{-x_0}, g_2^{-x}),$$

  $st := \mathtt{BT}$, and $RL := \emptyset$.
- $\mathsf{SKGen}(msk, \mathtt{I}, st)$: Parse st and msk as $\mathtt{BT}$ and $(\mathtt{MK}_1, \mathtt{MK}'_1, \mathtt{MK}_2, \mathtt{MK}'_2)$, respectively. It chooses an unassigned leaf η from $\mathtt{BT}$ uniformly at random, and stores $\mathtt{I}$ in the node η. For each node $\theta \in \mathsf{Path}(\mathtt{BT}, \eta)$, it recalls P_θ if it was defined. Otherwise, it chooses $P_\theta \xleftarrow{\$} \mathbb{G}_2$ and stores P_θ in the node θ. Then, it chooses $r_\theta \xleftarrow{\$} \mathbb{Z}_p$ and it computes

$$\mathtt{PK}_{1,\theta} := (g_2^{y_2})^{r_\theta},\ \mathtt{PK}'_{1,\theta} := P_\theta \left((g_2^{y_1})^{\mathtt{I}} g_2^{y_3}\right)^{r_\theta},$$
$$\mathtt{PK}_{2,\theta} := (g_2^{x_2})^{-r_\theta},\ \mathtt{PK}'_{2,\theta} := P_\theta \left((g_2^{x_1})^{\mathtt{I}} g_2^{x_3}\right)^{-r_\theta},\ \mathtt{PK}_{3,\theta} := g_2^{r_\theta}.$$

  It also chooses $r \xleftarrow{\$} \mathbb{Z}_p$ and computes

$$\mathtt{SK}_1 := (g_2^{y_2})^{r},\ \mathtt{SK}'_1 := \mathtt{MK}'_1 \left((g_2^{y_1})^{\mathtt{I}} g_2^{y_3}\right)^{r},$$
$$\mathtt{SK}_2 := (g_2^{x_2})^{-r},\ \mathtt{SK}'_2 := \mathtt{MK}'_2 \left((g_2^{x_1})^{\mathtt{I}} g_2^{x_3}\right)^{-r},\ \mathtt{SK}_3 := g_2^{r}.$$

  It outputs

$$pk_{\mathtt{I}} := \{(\mathtt{PK}_{1,\theta}, \mathtt{PK}'_{1,\theta}, \mathtt{PK}_{2,\theta}, \mathtt{PK}'_{2,\theta}, \mathtt{PK}_{3,\theta})\}_{\theta \in \mathsf{Path}(\mathtt{BT},\eta)},$$
$$sk_{\mathtt{I}} := (\mathtt{SK}_1, \mathtt{SK}'_1, \mathtt{SK}_2, \mathtt{SK}'_2, \mathtt{SK}_3).$$

– KeyUp($msk, st, RL, \mathtt{T}$): Parse msk as $(\mathtt{MK}_1, \mathtt{MK}'_1, \mathtt{MK}_2, \mathtt{MK}'_2)$. For each node $\theta \in \mathsf{KUNode}(\mathtt{BT}, RL, \mathtt{T})$, it recalls P_θ if it was defined. Otherwise, it chooses $P_\theta \xleftarrow{\$} \mathbb{G}_2$ and stores P_θ in the node θ. It chooses $s_\theta \xleftarrow{\$} \mathbb{Z}_p$ and computes

$$\mathtt{KU}'_{1,\theta} := P_\theta^{-1}\mathtt{MK}_1\left((g_2^{y_4})^{\mathtt{T}}g_2^{y_5}\right)^{s_\theta},$$
$$\mathtt{KU}'_{2,\theta} := P_\theta^{-1}\mathtt{MK}_2\left((g_2^{x_4})^{\mathtt{T}}g_2^{x_5}\right)^{-s_\theta}, \ \ \mathtt{KU}_{3,\theta} := g_2^{s_\theta}.$$

It outputs $ku_{\mathtt{T}} := \{(\mathtt{KU}'_{1,\theta}, \mathtt{KU}'_{2,\theta}, \mathtt{KU}_{3,\theta})\}_{\theta \in \mathsf{KUNode}(\mathtt{BT}, RL, \mathtt{T})}$.

– TKGen($pk_{\mathtt{I}}, ku_{\mathtt{T}}$): Parse $pk_{\mathtt{I}}$ and $ku_{\mathtt{T}}$ as $\{(\mathtt{PK}_{1,\theta}, \mathtt{PK}'_{1,\theta}, \mathtt{PK}_{2,\theta}, \mathtt{PK}'_{2,\theta}, \mathtt{PK}_{3,\theta})\}_{\theta \in \Theta_{\mathtt{SK}}}$ and $\{(\mathtt{KU}'_{1,\theta}, \mathtt{KU}'_{2,\theta}, \mathtt{KU}_{3,\theta})\}_{\theta \in \Theta_{\mathtt{KU}}}$, respectively. It outputs $\perp$ if $\Theta_{\mathtt{SK}} \cap \Theta_{\mathtt{KU}} = \emptyset$. Otherwise, for some $\theta \in \Theta_{\mathtt{SK}} \cap \Theta_{\mathtt{KU}}$, it computes as follows. It chooses $R, S \xleftarrow{\$} \mathbb{Z}_p$ and computes

$$\mathtt{TK}_1 := \mathtt{PK}_{1,\theta}(g_2^{y_2})^R, \ \ \mathtt{TK}'_1 := \mathtt{PK}'_{1,\theta}\mathtt{KU}'_{1,\theta}\left((g_2^{y_1})^{\mathtt{I}}g_2^{y_3}\right)^R\left((g_2^{y_4})^{\mathtt{T}}g_2^{y_5}\right)^S,$$
$$\mathtt{TK}_2 := \mathtt{PK}_{2,\theta}(g_2^{x_2})^{-R}, \ \ \mathtt{TK}'_2 := \mathtt{PK}'_{2,\theta}\mathtt{KU}'_{2,\theta}\left((g_2^{x_1})^{\mathtt{I}}g_2^{x_3}\right)^{-R}\left((g_2^{x_4})^{\mathtt{T}}g_2^{x_5}\right)^{-S},$$
$$\mathtt{TK}_3 := \mathtt{PK}_{3,\theta}g_2^R, \ \ \mathtt{TK}_4 := \mathtt{KU}_{3,\theta}g_2^S.$$

It outputs $tk_{\mathtt{I},\mathtt{T}} := (\mathtt{TK}_1, \mathtt{TK}'_1, \mathtt{TK}_2, \mathtt{TK}'_2, \mathtt{TK}_3, \mathtt{TK}_4)$.

– DKGen($sk_{\mathtt{I}}, \mathtt{T}$): Parse $sk_{\mathtt{I}}$ as $(\mathtt{SK}_1, \mathtt{SK}'_1, \mathtt{SK}_2, \mathtt{SK}'_2, \mathtt{SK}_3)$. It chooses $\hat{R}, \hat{S} \xleftarrow{\$} \mathbb{Z}_p$ and computes

$$\mathtt{DK}_1 := \mathtt{SK}_1(g_2^{y_2})^{\hat{R}}, \ \ \mathtt{DK}'_1 := \mathtt{SK}'_1\left((g_2^{y_1})^{\mathtt{I}}g_2^{y_3}\right)^{\hat{R}}\left((g_2^{y_4})^{\mathtt{T}}g_2^{y_5}\right)^{\hat{S}},$$
$$\mathtt{DK}_2 := \mathtt{SK}_2(g_2^{x_2})^{-\hat{R}}, \ \ \mathtt{DK}'_2 := \mathtt{SK}'_2\left((g_2^{x_1})^{\mathtt{I}}g_2^{x_3}\right)^{-\hat{R}}\left((g_2^{x_4})^{\mathtt{T}}g_2^{x_5}\right)^{-\hat{S}},$$
$$\mathtt{DK}_3 := \mathtt{SK}_3 g_2^{\hat{R}}, \ \ \mathtt{DK}_4 := g_2^{\hat{S}}.$$

It outputs $dk_{\mathtt{I},\mathtt{T}} := (\mathtt{DK}_1, \mathtt{DK}'_1, \mathtt{DK}_2, \mathtt{DK}'_2, \mathtt{DK}_3, \mathtt{DK}_4)$.

– Enc($M, \mathtt{I}, \mathtt{T}$): It chooses $t, \mathtt{tag} \xleftarrow{\$} \mathbb{Z}_p$. For $M \in \mathbb{G}_T$, it computes

$$C_0 := M \cdot z^t, \ \ C_1 := g_1^t, \ \ C_2 := (g_1^\alpha)^t,$$
$$C_3 := \left(u_1^{\mathtt{I}}w_1^{\mathtt{tag}}h_1\right)^t, \ \ C_4 := (v_1^{\mathtt{T}}\hat{v}_1)^t.$$

It outputs $C_{\mathtt{I},\mathtt{T}} := (C_0, C_1, C_2, C_3, C_4, \mathtt{tag})$.

– PartDec($tk_{\mathtt{I},\mathtt{T}}, C_{\mathtt{I},\mathtt{T}}$): Parse $tk_{\mathtt{I},\mathtt{T}}$ and $C_{\mathtt{I},\mathtt{T}}$ as $(\mathtt{TK}_1, \mathtt{TK}'_1, \mathtt{TK}_2, \mathtt{TK}'_2, \mathtt{TK}_3, \mathtt{TK}_4)$ and $(C_0, C_1, C_2, C_3, C_4, \mathtt{tag})$, respectively. It computes

$$C'_0 = \frac{C_0 e(C_3, \mathtt{TK}_3)e(C_4, \mathtt{TK}_4)}{e(C_1, \mathtt{TK}_1^{\mathtt{tag}}\mathtt{TK}'_1)e(C_2, \mathtt{TK}_2^{\mathtt{tag}}\mathtt{TK}'_2)}.$$

It outputs $ct_{\mathtt{I},\mathtt{T}} := (C'_0, C_1, C_2, C_3, C_4, \mathtt{tag})$

– Dec($dk_{\mathtt{I},\mathtt{T}}, ct_{\mathtt{I},\mathtt{T}}$): Parse $dk_{\mathtt{I},\mathtt{T}}$ and $C_{\mathtt{I},\mathtt{T}}$ as $(\mathtt{DK}_1, \mathtt{DK}'_1, \mathtt{DK}_2, \mathtt{DK}'_2, \mathtt{DK}_3, \mathtt{DK}_4)$ and $(C'_0, C_1, C_2, C_3, C_4, \mathtt{tag})$, respectively. It computes

$$M = \frac{C'_0 e(C_3, \mathtt{DK}_3)e(C_4, \mathtt{DK}_4)}{e(C_1, \mathtt{DK}_1^{\mathtt{tag}}\mathtt{DK}'_1)e(C_2, \mathtt{DK}_2^{\mathtt{tag}}\mathtt{DK}'_2)}.$$

– Revoke($\mathtt{I}, \mathtt{T}, RL, st$): Output $RL := RL \cup \{(\mathtt{I}, \mathtt{T})\}$.

If $tk_{\mathtt{I},\mathtt{T}}$ and $C_{\mathtt{I},\mathtt{T}}$ are correctly generated, we can easily check that it holds $C'_0 = M \cdot e(g_1, g_2)^{(-x\alpha+y)t}$ in a similar way to the proposed RIBE scheme as well as the correctness of Dec. We omit details.

Theorem 5. *If the* ADDH1 *and* DDH2 *assumptions hold, then the resulting SRIBE scheme* Π_{SA} *is* IND-SRID-CPA *secure.*

Proof sketch. We also construct a PPT algorithm $\mathcal{B}$ that breaks the IND-ID-CPA security of the modified Jutla-Roy IBE Π_{JR} using a PPT adversary $\mathcal{A}$ that breaks the IND-SRID-CPA security of Π_{SA}. When $\mathcal{B}$ receives a public parameter PP of Π_{JR}, $\mathcal{B}$ can create a public parameter mpk of Π_{SA} in the same way as the proof of Theorem 3 except for $z := e(g_1, g_2)^{-(x+x_0)\alpha+y+y_0}$. $\mathcal{B}$ computes $z := z_{\mathsf{JR}} \cdot e(\hat{g}_1, g_2)^{-x}e(g_1, g_2)^y$, where $\hat{g}_1 := g_1^\alpha$ and $z_{\mathsf{JR}} := e(g_1, g_2)^{-x_0\alpha+y_0}$ are components of PP, and $x, y \xleftarrow{\$} \mathbb{Z}_p$. The rest of the proof follows the proof of Theorem 3. □

6. ADDH1 problem in generic bilinear groups

We show a security proof of the ADDH1 assumption in the generic bilinear group model to provide confidence in the assumption. The generic group model is introduced by Shoup [50] to derive a lower bound on computational complexity of solving certain computational problems without looking into the actual groups structure used in a scheme. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be the Type-3 pairing. Elements of groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are encoded into uniform random strings so that equality of group elements can be only tested by the adversary. We assume four oracles. Three of them simulate the group actions in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$, respectively, and the fourth one simulates the bilinear map e. The encoding of group elements in $\mathbb{G}_1$ is modeled as an injective map $\sigma_1 : \mathbb{Z}_p \to \Sigma_1$, where $\Sigma_1 \subset \{0,1\}^*$. σ_1 maps all $x \in \mathbb{Z}_p$ to its string representation $\sigma(x)$ of $g_1^x \in \mathbb{G}_1$. Similarly, $\sigma_2 : \mathbb{Z}_p \to \Sigma_2$ and $\sigma_T : \mathbb{Z}_p \to \Sigma_T$ are defined, where $\Sigma_2, \Sigma_T \subset \{0,1\}^*$. An upper bound on the advantage of an adversary solving the ADDH1 problem in a generic bilinear group model is given by the following theorem.

Theorem 1. *Let $\mathcal{A}$ be an algorithm that attempts to solve the* ADDH1 *problem in the generic group model. Assume that σ_1, σ_2, and σ_T are random encoding functions for $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$, and $\mathcal{A}$ makes at most q queries to the oracles computing the group actions in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$, and the bilinear map e. If $d, c_1, c_2, c_3, c_4 \xleftarrow{\$} \mathbb{Z}_p^\times$ and $b \xleftarrow{\$} \{0,1\}$ with $z_b := c_1c_2$ and $z_{1-b} := c_4$, then given $\sigma_1(1)$, $\sigma_1(c_1)$, $\sigma_1(c_2)$, $\sigma_1(dc_3)$, $\sigma_1(z_0)$, $\sigma_1(z_1)$, $\sigma_2(1)$, $\sigma_2(d)$, $\sigma_2(c_2c_3)$, $\sigma_2(dc_3)$, $\sigma_2(1/c_3)$ the advantage ϵ of $\mathcal{A}$ in solving the problem is bounded by*

$$\epsilon \leq \frac{3(q+11)^2}{4p}.$$

Proof. We consider an algorithm $\mathcal{B}$ that simulates the generic bilinear group for $\mathcal{A}$. Let $F_{i,j}$ be polynomials over $\mathbb{Z}_p[C_1, C_2, C_3, D, Z_0, Z_1]$ with 6 variables $C_1, C_2, C_3, D, Z_0, Z_1$, and $\sigma_{i,j}$ be arbitrary distinct strings from $\{0,1\}$. $\mathcal{B}$ maintains three lists of pairs, $L_i := \{(F_{i,j}, \sigma_{i,j}) : j = 0, 1, \ldots, \tau_i - 1\}$ $(i \in \{1, 2, T\})$ such that at each step τ of the game the relation $\tau_1 + \tau_2 + \tau_T = \tau + 11$ holds. At the beginning of the game (i.e., $\tau = 0$), the lists are initialized by setting $\tau_1 = 6$, $\tau_2 = 5$, $\tau_T = 0$, $F_{1,0} = 1$, $F_{1,1} = C_1$, $F_{1,2} = C_2$, $F_{1,3} = DC_3$, $F_{1,4} = Z_0$, $F_{1,5} = Z_1$, $F_{2,0} = 1$, $F_{2,1} = D$, $F_{2,2} = C_2C_3$, $F_{2,3} = DC_3$, and $F_{2,4} = 1/C_3$. The corresponding strings are set to arbitrary distinct strings in $\{0,1\}^*$. We assume that $\mathcal{A}$ only queries the oracles on strings previously obtained from $\mathcal{B}$, and $\mathcal{B}$ can easily determine the index j of any given string $\sigma_{i,j}$ in the list L_i. $\mathcal{B}$ then starts the game by sending strings $\sigma_{1,0}, \sigma_{1,1}, \ldots, \sigma_{1,5}, \sigma_{2,0}, \sigma_{2,1}, \ldots, \sigma_{2,4}$, to $\mathcal{A}$. $\mathcal{B}$ simulates the oracles as follows.

Group actions in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$: First, we consider $\mathbb{G}_1$. After receiving two strings σ_{1,j_1} and σ_{1,j_2} with a selection bit indicating multiplication or division from $\mathcal{A}$, $\mathcal{B}$ computes $F_{1,\tau_1} := F_{1,j_1} \pm F_{1,j_2}$. If there exists an index i with $0 \leq i < \tau_1$ such that $F_{1,\tau_1} = F_{1,i}$, then $\mathcal{B}$ sets $\sigma_{1,\tau_1} := \sigma_{1,i}$. Otherwise, it sets σ_{1,τ_1} to a uniform random string from $\{0,1\}^* \setminus \{\sigma_{1,0}, \sigma_{1,1}, \ldots, \sigma_{1,\tau_1-1}\}$. $\mathcal{B}$ then adds the pair $(F_{1,\tau_1}, \sigma_{1,\tau_1})$ to L_1, returns σ_{1,τ_1} to $\mathcal{A}$, and increments τ_1 by one. $\mathcal{B}$ gives similar simulations of group actions in $\mathbb{G}_2$ and $\mathbb{G}_T$.

Pairing: After receiving σ_{1,j_1} and σ_{2,j_2} from $\mathcal{A}$, $\mathcal{B}$ computes $F_{T,\tau_T} := F_{1,j_1} \cdot F_{2,j_2}$. If there exists an index i with $0 \leq i < \tau_T$ such that $F_{T,\tau_T} = F_{T,i}$, then $\mathcal{B}$ sets $\sigma_{T,\tau_T} := \sigma_{T,i}$. Otherwise, it sets σ_{T,τ_T} to a uniform random string from $\{0,1\}^* \setminus \{\sigma_{T,0}, \sigma_{T,1}, \ldots, \sigma_{T,\tau_T-1}\}$. $\mathcal{B}$ then adds the pair $(F_{T,\tau_T}, \sigma_{T,\tau_T})$ to L_T, returns σ_{T,τ_T} to $\mathcal{A}$, and increments τ_T by one.

After at most q queries, $\mathcal{A}$ terminates and outputs a bit $b' \in \{0,1\}$. At this point, $\mathcal{B}$ chooses $d^*, c_1^*, c_2^*, c_3^*, c_4^* \xleftarrow{\$} \mathbb{Z}_p^\times$ and $b \xleftarrow{\$} \{0,1\}$, and sets $z_0^* := c_1^*c_2^*$ and $z_1^* := c_4^*$. $\mathcal{B}$ assigns $c_1^*, c_2^*, c_3^*, d^*, z_0^*, z_1^*$ to $C_1, C_2, C_3, D, Z_0, Z_1$. The simulation provided by $\mathcal{B}$ is perfect unless this assignment causes any of the following to hold.

1. $F_{1,j_1}(c_1^*, c_2^*, c_3^*, d^*, z_0^*, z_1^*) - F_{1,j_2}(c_1^*, c_2^*, c_3^*, d^*, z_0^*, z_1^*) = 0$ for some $j_1 \neq j_2$ and $F_{1,j_1} \neq F_{1,j_2}$.
2. $F_{2,j_1}(c_1^*, c_2^*, c_3^*, d^*, z_0^*, z_1^*) - F_{2,j_2}(c_1^*, c_2^*, c_3^*, d^*, z_0^*, z_1^*) = 0$ for some $j_1 \neq j_2$ and $F_{2,j_1} \neq F_{2,j_2}$.
3. $F_{T,j_1}(c_1^*, c_2^*, c_3^*, d^*, z_0^*, z_1^*) - F_{T,j_2}(c_1^*, c_2^*, c_3^*, d^*, z_0^*, z_1^*) = 0$ for some $j_1 \neq j_2$ and $F_{T,j_1} \neq F_{T,j_2}$.

Let F be an event that at least one of the above holds. As in the security proof of the DDH2v assumption in [37, Appendix B.1], which is the full version of [36], we use the result by Schwartz [40]: Let p be a prime number and $F(X_1, X_2, \ldots, X_k)$ be a non-zero polynomial in $\mathbb{Z}_p[X_1, X_2, \ldots, X_k]$ of degree m. Then, if $x_1, x_2, \ldots, x_k$ are chosen from $\mathbb{Z}_p$ uniformly at random, the probability that $F(x_1, x_2, \ldots, x_k) = 0$ is at most m/p.

We show that the simulation is perfect when F does not occur, and then b is information-theoretically hidden from the view point of $\mathcal{A}$. We note that all variables except for Z_b and Z_{1-b} are independent of b. Since Z_b is C_1C_2 which is a polynomial of degree 2, $\mathcal{A}$ will win it produces C_1C_2 using combinations of polynomials from L_1 and L_2. The only degree two polynomials that can be constructed are DC_1, DC_2, DC_3, C_2C_3 or a sum of these. $\mathcal{A}$ could also try to engineer degree three polynomials in L_T composed of C_1C_2. $\mathcal{A}$ can construct only $C_1C_2C_3$ from $\sigma_1(c_1)$ and $\sigma_2(c_2c_3)$, however, it does not have $\sigma_2(c_3)$, which is necessary for finding out b. Therefore, we have $\Pr[b = b' \mid \mathsf{F}] - 1/2$.

We then derive a bound on the probability that F occurs. For fixed j_1 and j_2, $F_{1,j_1} - F_{1,j_2}$ is a polynomial degree at most two and hence is zero at a random $d^*, c_1^*, c_2^*, c_3^*, z_0^*, z_1^*$ with probability at most $2/p$. Similarly, $F_{2,j_1} - F_{2,j_2}$ vanishes at

a random $d^*, c_1^*, c_2^*, c_3^*, z_0^*, z_1^*$ with probability at most $2/p$ for fixed j_1 and j_2. For fixed j_1 and j_2, $F_{T,j_1} - F_{T,j_2}$ vanishes at a random $d^*, c_1^*, c_2^*, c_3^*, z_0^*, z_1^*$ with probability at most $3/p$ since degree of the polynomial is at most three. There are totally $\binom{\tau_1}{2}$, $\binom{\tau_2}{2}$, and $\binom{\tau_T}{2}$ pairs of polynomials from L_1, L_2, and L_T, respectively. We have $\tau_1 + \tau_2 + \tau_T = \tau + 11 \leq q + 11$ since there are at most q queries. Thus, we have

$$\begin{aligned}\Pr[\mathsf{F}] &\leq \binom{\tau_1}{2}\frac{2}{p} + \binom{\tau_2}{2}\frac{2}{p} + \binom{\tau_T}{2}\frac{3}{p} \\ &\leq \epsilon \leq \frac{3(q+11)^2}{2p}.\end{aligned}$$

Since $\Pr[b' = b] \leq \Pr[b = b' \mid \neg\mathsf{F}](1 - \Pr[\mathsf{F}]) - \Pr[\mathsf{F}] \leq 1/2 + \Pr[\mathsf{F}]/2$ and $\Pr[b' = b] \geq \Pr[b = b' \mid \neg\mathsf{F}](1 - \Pr[\mathsf{F}]) - \Pr[\mathsf{F}] = 1/2 - \Pr[\mathsf{F}]/2$, we have

$$\left|\Pr[b = b'] - \frac{1}{2}\right| \leq \frac{\Pr[\mathsf{F}]}{2} \leq \epsilon \leq \frac{3(q+11)^2}{4p}.$$

We completed the proof. □

7. Concluding remarks

From a practical use perspective, both efficient revocation functionality and short key sizes are significantly important for cryptosystems. However, as aforementioned in the introduction, dual system encryption, which is a methodology for realizing constant-size IBE schemes, is hard to be used for constructing adaptively secure RIBE schemes with DKER and constant-size public parameters in prime-order groups. To realize such an RIBE scheme without the dual system encryption methodology, we took a similar approach to Seo and Emura's one [42]. Namely, we constructed a "basic" IBE scheme that satisfies important requirements for the Seo-Emura approach, based on the Jutla-Roy IBE [17], and then showed an RIBE construction based on the basic IBE. We proved the IND-RID-CPA security (with DKER) of the proposed scheme under mild variants of the SXDH assumption, which were newly introduced in this paper. We showed our RIBE scheme can be easily extended to a CCA-secure scheme. Moreover, our RIBE construction itself has the benefit of well harmonizing IBE variants, and SRIBE is a good example. The resulting SRIBE scheme becomes better than Qin et al.'s SRIBE scheme. We also proved the security of the assumption in the generic bilinear group model.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

We would like to thank anonymous reviewers of CT-RSA 2017 and TCS for valuable comments. Keita Emura was supported by JSPS KAKENHI Grant Number JP16K00198. Jae Hong Seo was supported by the Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No. 2016-6-00600, A Study on Functional Encryption: Construction, Security Analysis, and Implementation). Yohei Watanabe was supported by Grant-in-Aid for JSPS Fellows Grant Number JP16J10532 and Grant-in-Aid for Young Scientists Grant Number JP17K12697.

Appendix A. Omitted description

A.1. Hierarchical identity-based encryption

We give the syntax of hierarchical IBE (HIBE). Let $\mathtt{ID}|_j := (\mathtt{I}_1, \mathtt{I}_2, \ldots, \mathtt{I}_j)$ be a j-dimensional vector of IDs. An ℓ-level HIBE scheme Π_{IBE} consists of four-tuple algorithms (Init, KeyGen, IBEnc, IBDec) defined as follows.

- $(PP, MK) \leftarrow \mathsf{Init}(\lambda, \ell)$: A probabilistic algorithm for setup. It takes a security parameter λ as input and outputs a public parameter PP and a master secret key MK.
- $SK_{\mathtt{ID}|_{j+1}} \leftarrow \mathsf{KeyGen}(PP, SK_{\mathtt{ID}|_j}, \mathtt{I}_{j+1})$: An algorithm for private key generation. It takes PP, $SK_{\mathtt{ID}|_j}$, and an identity $\mathtt{I}_{j+1} \in \mathcal{I}$ as input and outputs a secret key $SK_{\mathtt{ID}|_{j+1}}$. Note that $SK_{\mathtt{ID}|_0}$ means MK.
- $C \leftarrow \mathsf{IBEnc}(PP, M, \mathtt{ID}|_j)$: A probabilistic algorithm for encryption. It takes PP, $M \in \mathcal{M}$, and $\mathtt{ID}|_j \in \mathcal{I}^j$ as input and then outputs a ciphertext C.
- M or $\perp \leftarrow \mathsf{IBDec}(PP, SK_{\mathtt{ID}|_j}, C)$: A deterministic algorithm for decryption. It takes PP, $SK_{\mathtt{ID}|_j}$, and C as input and then outputs M or $\perp$.

In the above model, we require that Π_{IBE} meets the following correctness property: For all security parameter $\lambda \in \mathbb{N}$, all $\ell := poly(\lambda)$, all $(PP, MK) \leftarrow \mathsf{Init}(\lambda, \ell)$, all $M \in \mathcal{M}$, all $\mathtt{ID}|_j \in \mathcal{I}^j$ for $j \in \{1, 2, \ldots, \ell\}$, it holds that $M \leftarrow \mathsf{IBDec}(PP, SK_{\mathtt{ID}|_j}, \mathsf{IBEnc}(PP, M, \mathtt{ID}|_j))$, where $SK_{\mathtt{ID}|_j}$ is a secret key for $\mathtt{ID}|_j$ correctly generated by KeyGen.

We describe the notion of indistinguishability against chosen plaintext attack (IND-ID-CPA). Let $\mathcal{A}$ be a PPT adversary, and $\mathcal{A}$'s advantage against IND-ID-CPA security is defined by

$$\mathsf{Adv}^{\mathsf{ID\text{-}CPA}}_{\Pi_{\mathrm{IBE}},\mathcal{A}}(\lambda, \ell) := \left| \Pr\left[b' = b \;\middle|\; \begin{array}{l} (PP, MK) \leftarrow \mathsf{Init}(\lambda, \ell), \\ (M_0^*, M_1^*, \mathtt{ID}|_j^*, state) \leftarrow \mathcal{A}^{KeyGen(\cdot)}(\mathsf{find}, PP), \\ b \xleftarrow{\$} \{0,1\}, \\ C^* \leftarrow \mathsf{IBEnc}(PP, M_b^*, \mathtt{ID}|_j^*), \\ b' \leftarrow \mathcal{A}^{KeyGen(\cdot)}(\mathsf{guess}, C^*, state) \end{array} \right] - \frac{1}{2} \right|.$$

KeyGen is an oracle that returns $SK_{\mathtt{ID}|_j}$ for a query $\mathtt{ID}|_j$. $\mathcal{A}$ cannot issue $\mathtt{ID}|_j^*$ and its prefix to *KeyGen*.

Definition 6 *(*IND-ID-CPA*).* An ℓ-level HIBE scheme Π_{IBE} is said to be IND-ID-CPA secure if for all PPT adversaries $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{ID\text{-}CPA}}_{\Pi_{\mathrm{IBE}},\mathcal{A}}(\lambda, \ell)$ is negligible in λ.

A.2. Complexity assumptions

We describe the DDH2v assumption, which was introduced in [36]. The authors proved the security of it in the generic bilinear group model. We furthermore describe the DDH1v assumption. This is analogous to the DDH2v assumption, and therefore its security can be proved in the same way as the DDH2v assumption.

The DDH2v **assumption.** Let $\mathcal{A}$ be a PPT adversary and we consider $\mathcal{A}$'s advantage against the DDH2v problem as follows.

$$\mathsf{Adv}^{\mathsf{DDH2v}}_{\mathcal{G},\mathcal{A}}(\lambda) := \left| \Pr\left[b' = b \;\middle|\; \begin{array}{l} D := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e) \leftarrow \mathcal{G}(\lambda), \\ d, c_1, c_2, c_3 \xleftarrow{\$} \mathbb{Z}_p,\ b \xleftarrow{\$} \{0,1\}, \\ \text{if } b = 0 \text{ then } Z := g_2^{c_1 c_2}, \text{ else } Z \xleftarrow{\$} \mathbb{G}_2, \\ b' \leftarrow \mathcal{A}(\lambda, D, g_1^d, g_1^{c_2 c_3}, g_1^{dc_3}, g_2^{c_1}, g_2^{c_2}, Z) \end{array} \right] - \frac{1}{2} \right|.$$

Definition 7 *(*DDH2v *assumption [36]).* The augmented DDH2v assumption relative to a generator $\mathcal{G}$ holds if for all PPT adversaries $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{DDH2v}}_{\mathcal{G},\mathcal{A}}(\lambda)$ is negligible in λ.

The DDH1v **assumption.** Let $\mathcal{A}$ be a PPT adversary and we consider $\mathcal{A}$'s advantage against the DDH1v problem as follows.

$$\mathsf{Adv}^{\mathsf{DDH1v}}_{\mathcal{G},\mathcal{A}}(\lambda) := \left| \Pr\left[b' = b \;\middle|\; \begin{array}{l} D := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e) \leftarrow \mathcal{G}(\lambda), \\ d, c_1, c_2, c_3 \xleftarrow{\$} \mathbb{Z}_p,\ b \xleftarrow{\$} \{0,1\}, \\ \text{if } b = 0 \text{ then } Z := g_1^{c_1 c_2}, \text{ else } Z \xleftarrow{\$} \mathbb{G}_1, \\ b' \leftarrow \mathcal{A}(\lambda, D, g_1^d, g_1^{c_1}, g_1^{c_2}, g_2^d, g_2^{c_2 c_3}, g_2^{dc_3}, Z) \end{array} \right] - \frac{1}{2} \right|.$$

Definition 8 *(*DDH1v *assumption).* The augmented DDH1v assumption relative to a generator $\mathcal{G}$ holds if for all PPT adversaries $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{DDH1v}}_{\mathcal{G},\mathcal{A}}(\lambda)$ is negligible in λ.

A.3. CCA *security*

We describe the notion of indistinguishability against chosen ciphertext attack for RIBE (IND-RID-CCA). Let $\mathcal{A}$ be a PPT adversary, and $\mathcal{A}$'s advantage against IND-RID-CCA security is defined by

$$\mathsf{Adv}^{\mathsf{RID\text{-}CCA}}_{\Pi,\mathcal{A}}(\lambda, N) := \left| \Pr\left[b' = b \;\middle|\; \begin{array}{l} (mpk, msk, RL, st) \leftarrow \mathsf{Setup}(\lambda, N), \\ (M_0^*, M_1^*, \mathtt{I}^*, \mathtt{T}^*, state) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{find}, mpk), \\ b \xleftarrow{\$} \{0,1\}, \\ C^*_{\mathtt{I}^*,\mathtt{T}^*} \leftarrow \mathsf{Enc}(M_b^*, \mathtt{I}^*, \mathtt{T}^*), \\ b' \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{guess}, C^*_{\mathtt{I}^*,\mathtt{T}^*}, state) \end{array} \right] - \frac{1}{2} \right|.$$

Here, $\mathcal{O}$ is a set of oracles $\{SKGen(\cdot), KeyUp(\cdot), Revoke(\cdot,\cdot), DKGen(\cdot,\cdot), Dec(\cdot,\cdot,\cdot)\}$. All the oracles except for *Dec* are the same as those of IND-RID-CPA game. *Dec* is defined as follows.

Dec$(\cdot,\cdot,\cdot)$**:** For a query $(\mathtt{I},\mathtt{T},C_{\mathtt{I},\mathtt{T}})$, it returns $\mathsf{Dec}(dk_{\mathtt{I},\mathtt{T}},C_{\mathtt{I},\mathtt{T}})$, where $dk_{\mathtt{I},\mathtt{T}}$ is a correctly-generated decryption key for $\mathtt{I}$ and $\mathcal{T}$.

In addition to the restrictions in IND-RID-CPA game, we consider the following restrictions on $\mathcal{A}$.

1. $Dec(\cdot,\cdot,\cdot)$ cannot be queried at $\mathtt{T}$ before issuing $\mathtt{T}$ to $\mathsf{KeyUp}(\cdot)$.
2. $(\mathtt{I}^*,\mathtt{T}^*,C^*_{\mathtt{I}^*,\mathtt{T}^*})$ cannot be issued to $Dec(\cdot,\cdot,\cdot)$.

Definition 9 *(*IND-RID-CCA*).* An RIBE scheme Π is said to be IND-RID-CCA secure if for all PPT adversaries $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{RID\text{-}CCA}}_{\Pi,\mathcal{A}}(\lambda,N)$ is negligible in λ.

A.4. Modified Jutla-Roy HIBE

We here give an HIBE version of the modified Jutla-Roy IBE. The ℓ-level modified Jutla-Roy IBE $\widehat{\Pi}_{\mathrm{JR}}$ =(Init, KeyGen, IBEnc, IBDec) is constructed as follows.

– $\mathsf{Init}(\lambda,\ell)$: It runs $(\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T,p,g_1,g_2,e)\leftarrow\mathcal{G}$. It chooses $x_0,y_0,\{x_{1,i},y_{1,i}\}_{i=1}^{\ell},x_2,y_2,x_3,y_3\xleftarrow{\$}\mathbb{Z}_p$ and $\alpha,\beta\xleftarrow{\$}\mathbb{Z}_p^{\times}$, and sets

$$z:=e(g_1,g_2)^{-x_0\alpha+y_0},\ \ u_{1,i}:=g_1^{-x_{1,i}\alpha+y_{1,i}}\text{ for every }i\in\{1,2,\dots\ell\},$$
$$w_1:=g_1^{-x_2\alpha+y_2},\ \ h_1:=g_1^{-x_3\alpha+y_3},\ \ \chi_1:=g_1^{\beta(-x_0\alpha+y_0)}.$$

It outputs

$$PP:=(g_1,g_1^{\alpha},\{u_{1,i}\}_{i=1}^{\ell},w_1,h_1,\chi_1,g_2,\{g_2^{x_{1,i}},g_2^{y_{1,i}}\}_{i=1}^{\ell},g_2^{x_2},g_2^{y_2},g_2^{x_3},g_2^{y_3},z,g_2^{x_0\beta},g_2^{y_0\beta},g_2^{\frac{1}{\beta}}),$$
$$MK:=(g_2^{y_0},g_2^{-x_0}).$$

– $\mathsf{KeyGen}(PP,SK_{\mathtt{ID}|_j},\mathtt{I}_{j+1})$:

Case of $j=0$ (MK): Parse MK as (d_1',d_2'). It chooses $r\xleftarrow{\$}\mathbb{Z}_p$ and computes

$$D_1:=(g_2^{y_2})^r,\ \ D_1':=d_1'\Big((g_2^{y_{1,1}})^{\mathtt{I}_1}g_2^{y_3}\Big)^r,$$
$$D_2:=(g_2^{x_2})^{-r},\ \ D_2':=d_2'\Big((g_2^{x_{1,1}})^{\mathtt{I}_1}g_2^{x_3}\Big)^{-r},\ \ D_3:=g_2^r,$$
$$K_{1,i}:=(g_2^{y_{1,i}})^r\text{ for every }i\in\{2,3,\dots,\ell\},$$
$$K_{2,i}:=(g_2^{x_{1,i}})^{-r}\text{ for every }i\in\{2,3,\dots,\ell\}.$$

It outputs

$$SK_{\mathtt{I}_1}:=(D_1,D_1',D_2,D_2',D_3,\{K_{1,i},K_{2,i}\}_{i=2}^{\ell}).$$

Case of $j\in\{1,2,\dots,\ell-1\}$: Parse $SK_{\mathtt{ID}|_j}$ as $(d_1,d_1',d_2,d_2',\{k_{1,i},k_{2,i}\}_{i=j+1}^{\ell})$. It chooses $r\xleftarrow{\$}\mathbb{Z}_p$ and computes

$$D_1:=d_1(g_2^{y_2})^r,\ \ D_1':=d_1'k_{1,j+1}^{\mathtt{I}_{j+1}}\Big(\prod_{i=1}^{j+1}(g_2^{y_{1,i}})^{\mathtt{I}_i}g_2^{y_3}\Big)^r,$$
$$D_2:=d_2(g_2^{x_2})^{-r},\ \ D_2':=d_2'k_{2,j+1}^{\mathtt{I}_{j+1}}\Big(\prod_{i=1}^{j+1}(g_2^{x_{1,i}})^{\mathtt{I}_i}g_2^{x_3}\Big)^{-r},\ \ D_3:=d_3g_2^r,$$
$$K_{1,i}:=k_{1,i}(g_2^{y_{1,i}})^r\text{ for every }i\in\{j+2,\dots,\ell\},$$
$$K_{2,i}:=k_{2,i}(g_2^{x_{1,i}})^{-r}\text{ for every }i\in\{j+2,\dots,\ell\}.$$

It outputs $SK_{\mathtt{ID}|_{j+1}}:=(D_1,D_1',D_2,D_2',D_3,\{K_{1,i},K_{2,i}\}_{i=j+2}^{\ell})$.

– $\mathsf{IBEnc}(PP,\mathtt{ID}|_j,M)$: It chooses $t,\mathtt{tag}\xleftarrow{\$}\mathbb{Z}_p$. For $M\in\mathbb{G}_T$, it computes

$$C_0:=Mz^t,\ \ C_1:=g_1^t,\ \ C_2:=(g_1^{\alpha})^t,\ \ C_3:=\Big(\prod_{i=1}^{j}u_{1,i}^{\mathtt{I}_i}w_1^{\mathtt{tag}}h_1\Big)^t.$$

It outputs $C:=(C_0,C_1,C_2,C_3,\mathtt{tag})$.

– $\mathsf{IBDec}(PP, SK_{\mathtt{ID}|_j}, C)$: Parse $SK_{\mathtt{I}}$ and C as $(D_1, D_1', D_2, D_2', D_3)$ and $(C_0, C_1, C_2, C_3, \mathtt{tag})$, respectively. It computes

$$M = \frac{C_0 e(C_3, D_3)}{e(C_1, D_1^{\mathtt{tag}} D_1') e(C_2, D_2^{\mathtt{tag}} D_2')}.$$

We show the correctness of Π_{JR}. Suppose that $sk_{\mathtt{ID}|_j} = (D_1, D_1', D_2, D_2', D_3)$ and $C = (C_0, C_1, C_2, C_3, \mathtt{tag})$ are correctly generated. Then, we have

$$\begin{aligned}
&\frac{C_0 e(C_3, D_3)}{e(C_1, D_1^{\mathtt{tag}} D_1') e(C_2, D_2^{\mathtt{tag}} D_2')} \\
&= \frac{M \cdot e(g_1, g_2)^{(-x_0\alpha + y_0)t}}{e(g_1^t, g_2^{y_2 r \mathtt{tag} + y_0 + r(\sum_{i=1}^{j} y_{1,i} \mathtt{I}_i + y_3)})} \cdot \frac{e(g_1^{t(\sum_{i=1}^{j} \mathtt{I}_i(-x_{1,i}\alpha + y_{1,i}) + \mathtt{tag}(-x_2\alpha + y_2) - x_3\alpha + y_3)}, g_2^r)}{e(g_1^{\alpha t}, g_2^{-x_2 r \mathtt{tag} - x_0 - r(\sum_{i=1}^{j} x_{1,i} \mathtt{I}_i + x_3)})} \\
&= \frac{M e(g_1, g_2)^{(-x_0\alpha + y_0)t}}{e(g_1^t, g_2^{y_0}) e(g_1^{\alpha t}, g_2^{-x_0})} = M.
\end{aligned}$$

Theorem 2. *If the* ADDH1 *and* DDH2 *assumptions hold, then the resulting ℓ-level modified Jutla-Roy IBE $\widehat{\Pi}_{\mathrm{JR}}$ is* IND-ID-CPA *secure.*

Since the above theorem can be proved in the same way as Theorem 2, we omit the proof.

A.5. One-time signature

An OTS scheme $\Pi_{\mathrm{OTS}} = (\mathsf{KG}, \mathsf{Sign}, \mathsf{Ver})$ is defined as follows.

- $(osk, ovk) \leftarrow \mathsf{KG}(\lambda)$: A probabilistic algorithm for setup. It takes a security parameter λ as input and outputs a signing/verification-key pair (osk, ovk).
- $\sigma \leftarrow \mathsf{Sign}(osk, m)$: An algorithm for signature generation. It takes osk and a message $m \in \mathcal{M}$ as input and outputs a signature σ.
- 1 or $0 \leftarrow \mathsf{Ver}(ovk, m, \sigma)$: A deterministic algorithm for verification. It takes ovk, m, and S as input and then outputs 1 (accept) or 0 (reject).

We require that for all $\lambda \in \mathbb{N}$, all $(osk, ovk) \leftarrow \mathsf{KG}(\lambda)$, all $m \in \mathcal{M}$, it holds that $\mathsf{Ver}(ovk, m, \mathsf{Sign}(osk, m)) \rightarrow 1$.

A notion of *strong unforgeability against one-time chosen message attack* (sUF-OT) is defined as follows. Let $\mathcal{A}$ be a PPT adversary, and $\mathcal{A}$'s advantage against sUF-OT security is defined by

$$\mathsf{Adv}^{\mathsf{sUF\text{-}OT}}_{\Pi_{\mathrm{OTS}}, \mathcal{A}}(\lambda) := \Pr\left[\mathsf{Ver}(ovk, m^*, \sigma^*) \rightarrow 1 \;\middle|\; \begin{array}{l} (osk, ovk) \leftarrow \mathsf{KG}(\lambda), \\ (m^*, \sigma^*) \leftarrow \mathcal{A}^{Sign(\cdot)}(ovk) \end{array} \right].$$

Sign is an oracle that returns $\mathsf{Sign}(osk, m)$ for a query m. $\mathcal{A}$ is allowed to access the *Sign* oracle only once, and we require $(m^*, \sigma^*) \neq (m, \sigma)$, where σ is a response of $Sign(m)$.

Definition 10 *(*sUF-OT*).* An OTS scheme Π_{OTS} is said to be sUF-OT secure if for all PPT adversaries $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{sUF\text{-}OT}}_{\Pi_{\mathrm{OTS}}, \mathcal{A}}(\lambda)$ is negligible in λ.

References

[1] P.S.L.M. Barreto, M. Naehrig, Pairing-friendly elliptic curves of prime order, in: B. Preneel, S. Tavares (Eds.), Selected Areas in Cryptography, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 319–331.

[2] A. Boldyreva, V. Goyal, V. Kumar, Identity-based encryption with efficient revocation, in: Proceedings of the 15th ACM Conference on Computer and Communications Security, ACM, New York, NY, USA, 2008, pp. 417–426.

[3] D. Boneh, X. Boyen, Efficient selective-id secure identity-based encryption without random oracles, in: C. Cachin, J. Camenisch (Eds.), Advances in Cryptology – EUROCRYPT 2004, Springer Berlin Heidelberg, 2004, pp. 223–238.

[4] D. Boneh, R. Canetti, S. Halevi, J. Katz, Chosen ciphertext security from identity based encryption, SIAM J. Comput. 36 (2007) 1301–1328.

[5] D. Boneh, M. Franklin, Identity-based encryption from the Weil pairing, in: J. Kilian (Ed.), Advances in Cryptology – CRYPTO 2001, Springer Berlin Heidelberg, 2001, pp. 213–229.

[6] J. Chen, H.W. Lim, S. Ling, H. Wang, H. Wee, Shorter identity-based encryption via asymmetric pairings, Des. Codes Cryptogr. 73 (2014) 911–947.

[7] J. Chen, H. Wee, Fully, (almost) tightly secure ibe and dual system groups, in: R. Canetti, J. Garay (Eds.), Advances in Cryptology – CRYPTO 2013, Springer Berlin Heidelberg, 2013, pp. 435–460.

[8] K. Emura, J.H. Seo, T. Youn, Semi-generic transformation of revocable hierarchical identity-based encryption and its DBDH instantiation, IEICE Trans. 99-A (2016) 83–91.

[9] K. Emura, A. Takayasu, Y. Watanabe, Adaptively secure revocable hierarchical ibe from k-linear assumption, Cryptology ePrint Archive, Report 2020/886, 2020.

[10] S.D. Galbraith, K.G. Paterson, N.P. Smart, Pairings for cryptographers, Discrete Appl. Math. 156 (2008) 3113–3121.
[11] A. Ge, P. Wei, Identity-based broadcast encryption with efficient revocation, in: D. Lin, K. Sako (Eds.), Public-Key Cryptography – PKC 2019, Springer International Publishing, Cham, 2019, pp. 405–435.
[12] C. Gentry, Practical identity-based encryption without random oracles, in: S. Vaudenay (Ed.), Advances in Cryptology – EUROCRYPT 2006, Springer Berlin Heidelberg, 2006, pp. 445–464.
[13] J. Gong, X. Dong, J. Chen, Z. Cao, Efficient IBE with tight reduction to standard assumption in the multi-challenge setting, in: J.H. Cheon, T. Takagi (Eds.), Advances in Cryptology – ASIACRYPT 2016, Springer Berlin Heidelberg, Berlin, Heidelberg, 2016, pp. 624–654.
[14] A. Guillevic, Comparing the pairing efficiency over composite-order and prime-order elliptic curves, in: ACNS 2013, Springer, 2013, pp. 357–372.
[15] D. Halevy, A. Shamir, The LSD broadcast encryption scheme, in: M. Yung (Ed.), Advances in Cryptology – CRYPTO 2002, Springer Berlin Heidelberg, 2002, pp. 47–60.
[16] Y. Ishida, Y. Watanabe, J. Shikata, Constructions of CCA-secure revocable identity-based encryption, in: E. Foo, D. Stebila (Eds.), Information Security and Privacy, ACISP 2015, Springer International Publishing, 2015, pp. 174–191.
[17] C.S. Jutla, A. Roy, Shorter quasi-adaptive NIZK proofs for linear subspaces, in: K. Sako, P. Sarkar (Eds.), Advances in Cryptology – ASIACRYPT 2013, Springer Berlin Heidelberg, 2013, pp. 1–20.
[18] S. Katsumata, T. Matsuda, A. Takayasu, Lattice-based revocable (hierarchical) IBE with decryption key exposure resistance, in: D. Lin, K. Sako (Eds.), Public-Key Cryptography – PKC 2019, Springer, 2019, pp. 441–471.
[19] S. Katsumata, T. Matsuda, A. Takayasu, Lattice-based revocable (hierarchical) IBE with decryption key exposure resistance, Theor. Comput. Sci. 809 (2020) 103–136.
[20] E. Kiltz, D. Galindo, Direct chosen-ciphertext secure identity-based key encapsulation without random oracles, in: L. Batten, R. Safavi-Naini (Eds.), Information Security and Privacy, Springer Berlin Heidelberg, 2006, pp. 336–347.
[21] K. Lee, Revocable hierarchical identity-based encryption with adaptive security, Cryptology ePrint Archive, Report 2016/749, 2016.
[22] K. Lee, A generic construction for revocable identity-based encryption with subset difference methods, Cryptology ePrint Archive, Report 2019/798, 2019.
[23] K. Lee, D.H. Lee, J.H. Park, Efficient revocable identity-based encryption via subset difference methods, Des. Codes Cryptogr. 85 (2017) 39–76.
[24] K. Lee, D.H. Lee, M. Yung, Sequential aggregate signatures made shorter, in: M. Jacobson, M. Locasto, P. Mohassel, R. Safavi-Naini (Eds.), Applied Cryptography and Network Security, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 202–217.
[25] K. Lee, D.H. Lee, M. Yung, Sequential aggregate signatures with short public keys: design, analysis and implementation studies, in: K. Kurosawa, G. Hanaoka (Eds.), Public-Key Cryptography – PKC 2013, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 423–442.
[26] K. Lee, D.H. Lee, M. Yung, Sequential aggregate signatures with short public keys without random oracles, Theor. Comput. Sci. 579 (2015) 100–125.
[27] K. Lee, S. Park, Revocable hierarchical identity-based encryption with shorter private keys and update keys, Des. Codes Cryptogr. 86 (2018) 2407–2440.
[28] A. Lewko, B. Waters, New techniques for dual system encryption and fully secure hibe with short ciphertexts, in: D. Micciancio (Ed.), Theory of Cryptography, TCC 2010, Springer Berlin Heidelberg, 2010, pp. 455–479.
[29] A.B. Lewko, Tools for simulating features of composite order bilinear groups in the prime order setting, in: D. Pointcheval, T. Johansson (Eds.), Advances in Cryptology – EUROCRYPT 2012, Springer Berlin Heidelberg, 2012, pp. 318–335.
[30] B. Libert, D. Vergnaud, Adaptive-id secure revocable identity-based encryption, in: M. Fischlin (Ed.), Topics in Cryptology – CT-RSA 2009, Springer Berlin Heidelberg, 2009, pp. 1–15.
[31] X. Ma, D. Lin, Generic constructions of revocable identity-based encryption, in: Z. Liu, M. Yung (Eds.), Information Security and Cryptology - 15th International Conference, Inscrypt 2019, Springer, 2019, pp. 381–396.
[32] X. Ma, D. Lin, Generic constructions of RIBE via subset difference method, Cryptology ePrint Archive, Report 2019/1376, 2019.
[33] D. Naor, M. Naor, J. Lotspiech, Revocation and tracing schemes for stateless receivers, in: J. Kilian (Ed.), Advances in Cryptology – CRYPTO 2001, Springer Berlin Heidelberg, 2001, pp. 41–62.
[34] K. Nguyen, H. Wang, J. Zhang, Server-aided revocable identity-based encryption from lattices, in: S. Foresti, G. Persiano (Eds.), Cryptology and Network Security, Springer International Publishing, Cham, 2016, pp. 107–123.
[35] B. Qin, R.H. Deng, Y. Li, S. Liu, Server-aided revocable identity-based encryption, in: G. Pernul, P.Y.A. Ryan, E. Weippl (Eds.), Computer Security – ESORICS 2015, Springer International Publishing, Cham, 2015, pp. 286–304.
[36] S.C. Ramanna, S. Chatterjee, P. Sarkar, Variants of Waters' dual system primitives using asymmetric pairings, in: M. Fischlin, J. Buchmann, M. Manulis (Eds.), Public Key Cryptography — PKC 2012, Springer Berlin Heidelberg, 2012, pp. 298–315.
[37] S.C. Ramanna, S. Chatterjee, P. Sarkar, Variants of Waters' dual system primitives using asymmetric pairings, Cryptology ePrint Archive, Report 2012/024, http://eprint.iacr.org/, 2012; The full version of [36].
[38] S.C. Ramanna, P. Sarkar, Efficient (anonymous) compact HIBE from standard assumptions, in: S. Chow, J. Liu, L. Hui, S. Yiu (Eds.), Provable Security, ProvSec 2014, Springer International Publishing, 2014, pp. 243–258.
[39] G. Ryu, K. Lee, S. Park, D.H. Lee, Unbounded hierarchical identity-based encryption with efficient revocation, in: H.w. Kim, D. Choi (Eds.), Information Security Applications, Springer International Publishing, Cham, 2016, pp. 122–133.
[40] J.T. Schwartz, Fast probabilistic algorithms for verification of polynomial identities, J. ACM 27 (1980) 701–717.
[41] J.H. Seo, K. Emura, Efficient delegation of key generation and revocation functionalities in identity-based encryption, in: E. Dawson (Ed.), Topics in Cryptology – CT-RSA 2013, Springer Berlin Heidelberg, 2013, pp. 343–358.
[42] J.H. Seo, K. Emura, Revocable identity-based encryption revisited: security model and construction, in: K. Kurosawa, G. Hanaoka (Eds.), Public-Key Cryptography – PKC 2013, Springer Berlin Heidelberg, 2013, pp. 216–234.
[43] J.H. Seo, K. Emura, Revocable hierarchical identity-based encryption, Theor. Comput. Sci. 542 (2014) 44–62.
[44] J.H. Seo, K. Emura, Revocable identity-based cryptosystem revisited: security models and constructions, IEEE Trans. Inf. Forensics Secur. 9 (2014) 1193–1205.
[45] J.H. Seo, K. Emura, Revocable identity-based encryption with rejoin functionality, IEICE Trans. 97-A (2014) 1806–1809.
[46] J.H. Seo, K. Emura, Adaptive-id secure revocable hierarchical identity-based encryption, in: K. Tanaka, Y. Suga (Eds.), Advances in Information and Computer Security, Springer International Publishing, 2015, pp. 21–38.
[47] J.H. Seo, K. Emura, Revocable hierarchical identity-based encryption: history-free update, security against insiders, and short ciphertexts, in: K. Nyberg (Ed.), Topics in Cryptology – CT-RSA 2015, Springer International Publishing, 2015, pp. 106–123.
[48] J.H. Seo, K. Emura, Revocable hierarchical identity-based encryption via history-free approach, Theor. Comput. Sci. 615 (2016) 45–60.
[49] A. Shamir, An efficient identification scheme based on permuted kernels (extended abstract), in: G. Brassard (Ed.), Advances in Cryptology – CRYPTO '89, Proceedings, Springer, New York, 1990, pp. 606–609.
[50] P. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, SIAM J. Comput. 26 (1997) 1484–1509, https://doi.org/10.1137/S0097539795293172.
[51] S. Wang, J. Zhang, J. He, H. Wang, C. Li, Simplified revocable hierarchical identity-based encryption from lattices, in: Y. Mu, R.H. Deng, X. Huang (Eds.), Cryptology and Network Security - Proceedings of the 18th International Conference, CANS 2019, Fuzhou, China, October 25–27, 2019, Springer, 2019, pp. 99–119.

[52] Y. Watanabe, K. Emura, J.H. Seo, New revocable IBE in prime-order groups: adaptively secure, decryption key exposure resistant, and with short public parameters, in: H. Handschuh (Ed.), Topics in Cryptology – CT-RSA 2017, Springer International Publishing, 2017, pp. 432–449.
[53] B. Waters, Efficient identity-based encryption without random oracles, in: R. Cramer (Ed.), Advances in Cryptology – EUROCRYPT 2005, Springer Berlin Heidelberg, 2005, pp. 114–127.
[54] B. Waters, Dual system encryption: realizing fully secure IBE and HIBE under simple assumptions, in: S. Halevi (Ed.), Advances in Cryptology – CRYPTO 2009, Springer Berlin Heidelberg, 2009, pp. 619–636.
[55] H. Wee, Déjà Q: Encore! un petit IBE, in: E. Kushilevitz, T. Malkin (Eds.), Theory of Cryptography, TCC 2016-A, Part II, Springer Berlin Heidelberg, Berlin, Heidelberg, 2016, pp. 237–258.