# Server-Aided Revocable Identity-Based Encryption

Baodong Qin[1,2], Robert H. Deng[1(✉)], Yingjiu Li[1], and Shengli Liu[3]

[1] School of Information Systems, Singapore Management University, Singapore 178902, Singapore
{robertdeng,yjli,bdqin}@smu.edu.sg
[2] Southwest University of Science and Technology, Mianyang 621010, China
[3] Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China
slliu@sjtu.edu.cn

**Abstract.** Efficient user revocation in Identity-Based Encryption (IBE) has been a challenging problem and has been the subject of several research efforts in the literature. Among them, the tree-based revocation approach, due to Boldyreva, Goyal and Kumar, is probably the most efficient one. In this approach, a trusted Key Generation Center (KGC) periodically broadcasts a set of key updates to all (non-revoked) users through public channels, where the size of key updates is only $O(r \log \frac{N}{r})$, with $N$ being the number of users and $r$ the number of revoked users, respectively; however, every user needs to keep at least $O(\log N)$ long-term secret keys and all non-revoked users are required to communicate with the KGC regularly. These two drawbacks pose challenges to users who have limited resources to store their secret keys or cannot receive key updates in real-time.

To alleviate the above problems, we propose a novel system model called *server-aided* revocable IBE. In our model, almost all of the workloads on users are delegated to an *untrusted* server which manages users' public keys and key updates sent by a KGC periodically. The server is untrusted in the sense that it does not possess any secret information. Our system model requires each user to keep just one short secret key and *does not* require users to communicate with either the KGC or the server during key updating. In addition, the system supports delegation of users' decryption keys, namely it is secure against decryption key exposure attacks. We present a concrete construction of the system that is provably secure against adaptive-ID chosen plaintext attacks under the DBDH assumption in the standard model. One application of our server-aided revocable IBE is encrypted email supporting lightweight devices (e.g., mobile phones) in which an email server plays the role of the untrusted server so that only non-revoked users can read their email messages.

**Keywords:** IBE · Revocation · Decryption key exposure


G. Pernul et al. (Eds.): ESORICS 2015, Part I, LNCS 9326, pp. 286–304, 2015.
DOI: 10.1007/978-3-319-24174-6_15

## 1 Introduction

Identity-Based Encryption (IBE) [26] eliminates the need for a Public Key Infrastructure (PKI) as in the traditional Public-Key Encryption (PKE) systems. In an IBE system, each user is allowed to use an arbitrary string (e.g., email address or phone number) as his/her public key. The corresponding decryption key is computed by a trusted authority, called Key Generation Center (KGC). Identity-based encryption has been thoroughly studied using pairing, e.g., [5,7,23] or other mathematical tools [6,8]. IBE has also been generalized to hierarchical IBE [12], fuzzy IBE [22] and attribute-based encryption [11]. In the IBE setting, as well as in all its generalizations, it is important and necessary to provide a means to revoke (compromised) users from the system. In the PKI setting, efficient revocation (e.g., [1,10,19,20]) is achievable via publicly available certificate revocation lists. However, realizing efficient user revocation in the IBE setting has been quit challenging.

To address the challenge of key revocation in IBE, Boneh and Franklin (BF) [5] suggested that a sender encrypts a message using a recipient's identity concatenated with the current time period, i.e., $\mathsf{id}||t$, and the KGC issues a decryption key $\mathsf{DK}_{\mathsf{id}||\mathsf{t}}$ for every non-revoked user and over every time period. Unfortunately, the BF approach is inefficient: the KGC must generate $O(N-r)$ new decryption keys in each time period, where $N$ is the total number of users and $r$ is the number of revoked users in the time period $\mathsf{t}$. Hence, the workload on the KGC is proportional to $N$. Moreover, each non-revoked user need to maintain a secure channel with the KGC to get his/her new decryption key.

Boldyreva, Goyal and Kumar (BGK) [3] proposed and formalized the notion of revocable IBE. They presented an efficient R-IBE scheme based on the fuzzy IBE scheme of Sahai and Waters [22] and the tree-based revocation scheme of Naor et al. [19] in the selective-ID security model. In their scheme, each user keeps a tuple of long term secret keys. The KGC publicly broadcasts a set of key updates in each time period, so that only non-revoked users can compute new decryption keys from their long term secret keys and the key updates. Compared with the BF approach, the BGK approach significantly reduces the total size of key updates from linear to logarithmic (i.e., $O(r \log \frac{N}{r})$) in the number of users. Nevertheless, in practice, the BGK approach may suffer from the following two limitations: (1) all non-revoked users need to communicate with the KGC and update their decryption keys periodically; and (2) the sizes of both key updates and users' secret keys grow logarithmically in the number of users, i.e., $O(r \log \frac{N}{r})$ and $O(\log N)$, respectively. The first limitation cannot be avoided due to the system model of revocable IBE; while the second limitation, as explained in Lee et al. [21], is inherent in the tree-based revocation approach. Other revocable IBE schemes [13,17,18,21,24,25] that follow the BGK revocable IBE model also have such limitation(s). *A natural question that arises is whether the two limitations can be overcome in a new system model for revocable IBE?*

Our Contributions and Results. In this paper, we propose a novel revocable IBE system model to overcome the two limitations in the BGK approach. Our idea

is based on the observation that in the BGK approach, almost all of the workload on the user side can be delegated to an *untrusted* third party server. Our system model, referred to as Server-aided Revocable IBE (SR-IBE), is depicted in Fig. 1. Specifically, the SR-IBE system, consists of four types of parities: a KGC, senders, recipients and a server, and works as follows:

1. (Key Distribution: KGC ⟶ recipients and server) At the system setup phase, the KGC issues a long-term secret key and a corresponding tuple of long-term public keys for every recipient/user. The former is given to a recipient while the latter is given to the server.
2. (Encryption: sender ⟶ server) A sender encrypts a message for an identity and a time period. The resulting ciphertext is sent to the server.
3. (Partial Decryption: server ⟶ recipient) The server transforms the ciphertext to a partially decrypted ciphertext using a transformation key corresponding to the recipient's identity and the time period embedded in the ciphertext.
4. (Decryption: recipient) The recipient recovers the sender's message from the partially decrypted ciphertext using his/her long-term secret key or a delegated decryption key for the current time period.
5. (Key Updates: KGC ⟶ server) In each key updating period, the KGC delivers a set of key updates to the server rather than to all non-revoked users. The server combines the key updates and the stored users' public keys to generate the transformation keys in the current time period for all users.

As in the standard revocable IBE, the KGC in SR-IBE is assumed to be fully trusted and cannot be compromised. However, the server in our model is assumed to be *untrusted* in the sense that it does not keep any secret data and only performs public storage and computation operations according to the system specifications. This notion of untrusted server is much weaker than the notion of *semi-trusted* third party in the literature which is normally assumed

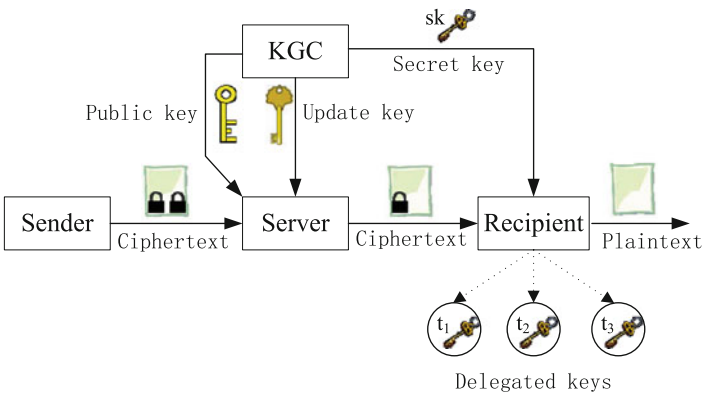



**Fig. 1.** System model of our server-aided revocable IBE

to hold some secret keys and cannot collude with other parties. We stress that in both cases, the server (or the third party) should perform correct operations and give correct results to the users (or the other parties). We will propose a formal security model (see Sect. 3) for RS-IBE, capturing all known threats as considered in the standard R-IBE model. We will also construct a concrete SR-IBE scheme. Remarkably, even assuming an untrusted server, our scheme achieves the following advantages simultaneously:

- It is provably secure against *both* adaptive-ID attacks and decryption exposure attacks under the Decisional Bilinear Diffie-Hellman (DBDH) assumption in the standard model, which is the refined security model for revocable IBE proposed by Seo and Emura.
- The size of every user's private key is *constant* (i.e., $O(1)$).
- *No communication* is required between users and the KGC during key update.
- The size of key updates from the KGC to the server is logarithmic (i.e., $O(r \log \frac{N}{r})$) in the number of users, as in the tree-based approach.

To show the advantages of our approach, we give a detailed comparison of our scheme with some representative non-server-aided revocable IBEs [3,5,17,18,24] and a server-aided revocable IBE [14] in Tables 1 and 2, respectively. Clearly, in our SR-IBE, non-revoked users do *not* need to communicate with the KGC or the server, while in all previous approaches, including the server-aided approach of [14], users must communicate with either the KGC or the server during every key update period. Additionally, almost all the workload on users in the previous approaches is taken over by the server in our SR-IBE while *without* sacrificing security (i.e., the scheme is still secure in the refined full security model of [24]). It is worth noting that the approach in [21] solved the second problem existed in the tree-based approach using multilinear maps, but the size of the public parameter linearly depends on the number of users and its security is proved in the selective revocation list model, which is weaker than the adaptive-ID model.

The authors of [14] showed how to delegate workload of the KGC to a semi-trusted server, which they referred to as outsourced KGC. In their approach, though the size of secret keys kept by each user is constant, the outsourced KGC must manage an outsourced master secret key and a large number of secret key shares (linear to the number of users) due to their revocation strategy of randomly splitting the master secret key for each user. Hence, the approach in [14] can not prevent collusion attacks between the outsourced KGC and revoked users, which is indicated in Table 2.

The above feature of our SR-IBE is especially attractive for lightweight user devices such as mobile phones. An excellent application scenario of the SR-IBE is secure email system in which the email server stores users' public keys, and performs key updates and partial decryptions; while email recipients only need to store their (constant size) secret keys and using them to recover email content from partially decrypted messages. In addition, the SR-IBE system supports delegation of decryption keys. When a user is away from office for a period of time, he can delegate his decryption keys over this period to his colleagues or assistants.

**Table 1.** Comparison with non-server-aided revocalbe IBE schemes

| Schemes | BF [5] | BGK [3] | LCFZZ [18] | LV [17] | SE [24] | PLL [21] | LLP [13] | Ours 4.2 | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | User | Server |
| PP Size | $O(1)$ | $O(1)$ | $O(1)$ | $O(\kappa)$ | $O(\kappa)$ | $O(N+\kappa)$ | $O(1)$ | $O(\kappa)$ | |
| PK Size | - | - | - | - | - | - | - | - | $O(N \log N)$ |
| SK Size | $O(1)$ | $O(\log N)$ | $O(1)$ | $O(\log N)$ | $O(\log N)$ | $O(1)$ | $O(\log^{1.5} N)$ | $O(1)$ | - |
| CT Size | $O(1)$ | $O(1)$ | $O(1)$ | $O(1)$ | $O(1)$ | $O(1)$ | $O(1)$ | $O(1)$ | $O(1)$ |
| KU Size | $O(N-r)$ | $O(r \log \frac{N}{r})$ | $O(r)$ | $O(r \log \frac{N}{r})$ | $O(r \log \frac{N}{r})$ | $O(1)$ | $O(r)$ | - | $O(r \log \frac{N}{r})$ |
| Dec. Cost | $O(1)$ | $O(1)$ | $O(r)$ | $O(1)$ | $O(1)$ | $O(1)$ | $O(1)$ | $O(1)$ | $O(1)$ |
| DKE Resis. | ✓ | × | × | × | ✓ | ✓ | ✓ | ✓ | |
| Model | Full | Selective | Selective | Full | Full | SelectiveRL | Full | Full | |
| Assumption | RO, BDH | DBDH | DBDH | DBDH | DBDH | MDHE | Static | DBDH | |

$N$ is the total number of users, $r$ is the number of revoked users and $\kappa$ is the security parameter. The meanings of those abbreviations can be followed easily or found in the paper. "-" means that the item does not exist in the corresponding scheme.

**Table 2.** Comparison with server-aided revocable IBE schemes

| Schemes | SK Size | | KU Size | | DKE Resis. | Model | Assumption | Collusion |
|---|---|---|---|---|---|---|---|---|
| | User | Server | KGC-Server | Server-User | | | | with User |
| LLCJL [14] | $O(1)$ | $O(N-r)$ | - | $O(N-r)$ | ✓ | Full | RO, DBDH | × |
| Ours Sect. 4.2 | $O(1)$ | - | $O(r \log \frac{N}{r})$ | - | ✓ | Full | DBDH | ✓ |

Other Related Works and Discussion. Revocation with mediator [2,4,9,16] has been studied in the IBE setting, where an online semi-trusted third party (i.e., mediator) holds shares of all users' secret keys and helps users to decrypt ciphertexts. User revocation is managed by the mediator by disabling the decryption service for revoked users. As a result, this approach is subject to collusion attack between the mediator and revoked users. Our SR-IBE system model seems similar to but is inherently different from the mediator approach. In SR-IBE, user revocation is controlled by the KGC, but not the server. The server simply functions as a publicly accessible computer. Without the server, users still can decrypt their ciphertexts as they can reconstruct their transformation keys from the public keys and public key updates.

Li et al. [14] proposed an efficient method to delegate the key update workload of the trusted KGC to an outsourced semi-trusted KGC. The functionality of the outsourced KGC is similar to that of the mediator discussed earlier. For each user, the outsourced KGC splits an outsourced master secret into two shares: one is used to compute key updates and the other is used to compute the secret key for the user. To revoke a user, instead of stopping decryption service as in the mediator approach, the outsourced KGC stops sending key updates to the revoked user. So, the outsourced KGC cannot collude with revoked users and the size of key updates is linear to the number of users. Recently, Liang et al. [15] proposed a cloud-based revocable IBE with ciphertext delegation. They employed a similar secret key split technique as in [14] to achieve revocation and hence the size of key updates grows linearly with the number of system users. Besides identity revocation, they also considered ciphertext delegation through a proxy re-encryption technique so that revoked users cannot decrypt old ciphertexts.

The work in [27] combined revocable encryption with the standard IBE to directly revoke users by specifying a receiver and a set of revoked users in a ciphertext. This approach requires a sender to know the set of revoked users and hence it does not follow the notion of revocable IBE considered in this paper.

Organization. The rest of this paper is organized as follows. Section 2 introduces some basic cryptographic notions. The formal definition and security model for SR-IBE are given in Sect. 3. The main construction and security proof of our scheme are presented in Sects. 4 and 5 respectively. Summary is given in Sect. 6.

## 2 Preliminaries

**Notations.** Throughout the paper, $\mathbb{N}$ denotes the set of natural numbers and $\kappa \in \mathbb{N}$ denotes the security parameter. If $S$ is a finite set, then $s \leftarrow_R S$ denotes the operation of picking an element $s$ from $S$ uniformly at random. If $X$ is a random variable over $S$, then we write $x \leftarrow X$ to denote the process of sampling a value $x \in S$ according to the distribution $X$. We call a function $\mathsf{negl}$ negligible in $\kappa$, if for every positive polynomial $\mathsf{poly}(\cdot)$ there exists an $N$ such that for all $\kappa > N$, $\mathsf{negl}(\kappa) < 1/\mathsf{poly}(\kappa)$. A probabilistic polynomial-time (PPT) algorithm $A$ is an algorithm that on input $x$, computes $A(x)$ using randomness and its running time is bounded by $\mathsf{poly}(\kappa)$.

**Bilinear Groups.** Let $\mathbb{G}, \mathbb{G}_T$ be groups of prime order $p$ and let $g$ be a generator of $\mathbb{G}$. An efficiently computable map $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is a (symmetric) pairing if it satisfies the following two conditions:

- (Bilinearity) For all $a, b \in \mathbb{Z}_p$, we have $\hat{e}(g^a, g^b) = \hat{e}(g, g)^{ab}$;
- (Non-degeneracy) For any generator $g$ of $\mathbb{G}$, $\hat{e}(g, g)$ is a generator of $\mathbb{G}_T$ (i.e., $\hat{e}(g, g) \neq 1$).

We denote by $\mathcal{BP}(\kappa)$ a bilinear group generator, which takes as input a security parameter $\kappa$, and outputs a description of bilinear groups $\mathcal{G} = (\mathbb{G}, \mathbb{G}_T, \hat{e}, p, g)$.

**The DBDH Assumption.** Let $\mathcal{G} = (\mathbb{G}, \mathbb{G}_T, \hat{e}, q, g) \leftarrow \mathcal{BP}(\kappa)$. The Decisional Bilinear Diffie-Hellman (DBDH) assumption states that, for any PPT algorithm, it is hard to distinguish the tuple $(\mathcal{G}, g^a, g^b, g^c, \hat{e}(g, g)^{abc})$ from the tuple $(\mathcal{G}, g^a, g^b, g^c, Z)$, where $a, b, c \leftarrow_R \mathbb{Z}_p$ and $Z \leftarrow_R \mathbb{G}_T$.

**The Waters IBE Scheme** [28]**.** Let $\mathbb{G}$ be a group of prime order $p$. For an identity $\mathsf{id} = (b_1, \ldots, b_n) \in \{0,1\}^n$ and $U = (u_0, u_1, \ldots, u_n) \in \mathbb{G}^{n+1}$, we denote by $F_{\mathsf{Wat},U}(\mathsf{id}) = u_0 \cdot \prod_{i=1}^{n} u_i^{b_i}$ the hash function used in the IBE scheme of Waters.

The Waters IBE scheme consists of the following five PPT algorithms:

$\mathsf{Sys}_{\mathsf{Wat}}(\kappa)$: On input $\kappa$, output a system parameter $\mathsf{pp}_{\mathsf{Wat}} = (\mathcal{G}, h, U)$, where $\mathcal{G} \leftarrow \mathcal{BP}(\kappa)$, $h \leftarrow_R \mathbb{G}$ and $U \leftarrow_R \mathbb{G}^{n+1}$.

$\mathsf{Setup}_{\mathsf{Wat}}(\mathsf{pp}_{\mathsf{Wat}})$: On input $\mathsf{pp}_{\mathsf{Wat}}$, output master public key $\mathsf{MPK}_{\mathsf{Wat}} = g_1 = g^\alpha$ and master secret key $\mathsf{MSK}_{\mathsf{Wat}} = h^\alpha$, where $\alpha \leftarrow_R \mathbb{Z}_p$.

$\mathsf{PrivKG}_{\mathsf{Wat}}(\mathsf{MSK}_{\mathsf{Wat}}, \mathsf{id})$: On input $\mathsf{MSK}_{\mathsf{Wat}}$ and an identity $\mathsf{id} \in \{0,1\}^n$, output user's secret key $\mathsf{SK}_{\mathsf{id}} = (h^\alpha \cdot F_{\mathsf{Wat},U}(\mathsf{id})^r, g^r)$, where $r \leftarrow_R \mathbb{Z}_p$.

$\mathsf{Enc}_{\mathsf{Wat}}(\mathsf{MPK}_{\mathsf{Wat}}, \mathsf{id}, M)$: On input $\mathsf{MPK}_{\mathsf{Wat}}$, $\mathsf{id} \in \{0,1\}^n$ and message $M \in \mathbb{G}$, choose $z \leftarrow_R \mathbb{Z}_p$ and output $CT_{\mathsf{id}} = (C_0, C_1, C_2)$ where $C_0 = \hat{e}(g_1, h)^z \cdot M$, $C_1 = g^z$, $C_2 = F_{\mathsf{Wat},U}(\mathsf{id})^z$.

$\mathsf{Dec}_{\mathsf{Wat}}(\mathsf{SK}_{\mathsf{id}}, CT_{\mathsf{id}})$: On input $\mathsf{SK}_{\mathsf{id}} = (d_1, d_2)$ and $CT_{\mathsf{id}} = (C_0, C_1, C_2)$, output $M = C_0 \cdot K^{-1}$, where $K = \hat{e}(d_1, C_1) \cdot \hat{e}(d_2, C_2)^{-1}$.

We adopt the standard (adaptive) ID-CPA security of IBE as defined, e.g., in [5]. From [28], we have the following theorem.

**Theorem 1 (Security of the Waters IBE** [28, Theorem 1]**).** *Under the DBDH assumption, the Waters IBE scheme is ID-CPA secure and the security proof induces to a factor of $O(nQ)$ reduction loss, where $Q$ is the number of private key queries.*

## 3 Definition and Security of SR-IBE

**Definition 1 (SR-IBE).** *A SR-IBE scheme involves four parties: a key generation center (KGC), sender, recipient and a third party (i.e., a server). Algorithms among these parties are defined as follows:*

$\mathsf{pp} \leftarrow \mathsf{Sys}(\kappa)$*: This is the system parameter generation algorithm run by the KGC. It takes as input a security parameter $\kappa$ and outputs a system parameter* $\mathsf{pp}$*, shared by all parities.*

$(\mathsf{MPK}, \mathsf{MSK}, \mathsf{RL}, \mathsf{ST}) \leftarrow \mathsf{Setup}(\mathsf{pp}, N)$*: This is the setup algorithm run by the KGC. It takes as input the system parameter* $\mathsf{pp}$ *and a maximal number of users $N$, and outputs a master public key* $\mathsf{MPK}$*, a master secret key* $\mathsf{MSK}$*, an initial revocation list* $\mathsf{RL}$ *and state* $\mathsf{ST}$*.*

$(\mathsf{PK}_{\mathsf{id}}, \mathsf{ST}) \leftarrow \mathsf{PubKG}(\mathsf{MSK}, \mathsf{id}, \mathsf{ST})$*: This is the public key generation algorithm run by the KGC. It takes as input a master secret key* $\mathsf{MSK}$*, the recipient's identity* $\mathsf{id}$ *and state* $\mathsf{ST}$*, and outputs a public key* $\mathsf{PK}_{\mathsf{id}}$ *for the recipient, and an updated state* $\mathsf{ST}$*. The public key* $\mathsf{PK}_{\mathsf{id}}$ *is sent to the server (through a public channel).*

$(\mathsf{KU}_{\mathsf{TK},\mathsf{t}}, \mathsf{ST}) \leftarrow \mathsf{TKeyUp}(\mathsf{MSK}, \mathsf{t}, \mathsf{RL}, \mathsf{ST})$*: This is the transformation key update generation algorithm run by the KGC. It takes as input a master secret key* $\mathsf{MSK}$*, a time period* $\mathsf{t}$*, a revocation list* $\mathsf{RL}$ *and a state* $\mathsf{ST}$*, and outputs a transformation key update* $\mathsf{KU}_{\mathsf{TK},\mathsf{t}}$ *and an updated state* $\mathsf{ST}$*. The key update* $\mathsf{KU}_{\mathsf{TK},\mathsf{t}}$ *is sent to the server (through a public channel).*

$\mathsf{TK}_{\mathsf{id},\mathsf{t}} \leftarrow \mathsf{TranKG}(\mathsf{PK}_{\mathsf{id}}, \mathsf{KU}_{\mathsf{TK},\mathsf{t}})$*: This is the transformation key generation algorithm run by the server. It takes as input a public key* $\mathsf{PK}_{\mathsf{id}}$ *for identity* $\mathsf{id}$ *and a transformation key update* $\mathsf{KU}_{\mathsf{TK},\mathsf{t}}$ *for time period* $\mathsf{t}$*, and outputs a transformation key* $\mathsf{TK}_{\mathsf{id},\mathsf{t}}$*.*

$\mathsf{SK}_{\mathsf{id}} \leftarrow \mathsf{PrivKG}(\mathsf{MSK}, \mathsf{id})$*: This is the private key generation algorithm run by the KGC. It takes as input a master secret key* $\mathsf{MSK}$ *and the recipient's identity* $\mathsf{id}$*, and outputs a private key* $\mathsf{SK}_{\mathsf{id}}$ *for the recipient. The private key must be sent to the recipient through a secure channel.*

$\mathsf{DK}_{\mathsf{id},\mathsf{t}} \leftarrow \mathsf{DecKG}(\mathsf{SK}_{\mathsf{id}}, \mathsf{t})$*: This is the decryption key generation algorithm run by the recipient himself. It takes as input his private key* $\mathsf{SK}_{\mathsf{id}}$ *and a time period* $\mathsf{t}$*, and outputs a decryption key* $\mathsf{DK}_{\mathsf{id},\mathsf{t}}$ *for time period* $\mathsf{t}$*.*

$CT_{\mathsf{id},\mathsf{t}} \leftarrow \mathsf{Enc}(\mathsf{MPK}, \mathsf{id}, \mathsf{t}, M)$*: This is the encryption algorithm run by the sender. It takes as input a master public key* $\mathsf{MPK}$*, the recipient's identity* $\mathsf{id}$*, a time period* $\mathsf{t}$ *and a message* $M$*, and outputs a ciphertext* $CT_{\mathsf{id},\mathsf{t}}$*. The ciphertext is sent into the server.*

$CT'_{\mathsf{id},\mathsf{t}} \leftarrow \mathsf{Transform}(\mathsf{TK}_{\mathsf{id},\mathsf{t}}, CT_{\mathsf{id},\mathsf{t}})$*: This is the ciphertext transformation algorithm run by the server. It takes as input a transformation key* $\mathsf{TK}_{\mathsf{id},\mathsf{t}}$ *and a ciphertext* $CT_{\mathsf{id},\mathsf{t}}$*, and outputs a partially decrypted ciphertext* $CT'_{\mathsf{id},\mathsf{t}}$*. The partially decrypted ciphertext* $CT'_{\mathsf{id},\mathsf{t}}$ *is publicly sent to the recipient.*

$M/\perp \leftarrow \mathsf{Dec}(\mathsf{DK}_{\mathsf{id},\mathsf{t}}, CT'_{\mathsf{id},\mathsf{t}})$*: This is the decryption algorithm run by the recipient. It takes as input a decryption key* $\mathsf{DK}_{\mathsf{id},\mathsf{t}}$ *and a partially decrypted ciphertext* $CT'_{\mathsf{id},\mathsf{t}}$*, and outputs a message* $M$ *or the special symbol* $\perp$*.*

$\mathsf{RL} \leftarrow \mathsf{Revoke}(\mathsf{id}, \mathsf{t}, \mathsf{RL}, \mathsf{ST})$*: This is the revocation algorithm run by the KGC. It takes as input an identity* $\mathsf{id}$*, a time period* $\mathsf{t}$*, a revocation list* $\mathsf{RL}$ *and state* $\mathsf{ST}$*, and outputs an updated revocation list* $\mathsf{RL}$*.*

**Correctness.** The *correctness* requires that for all security parameter $\kappa$ and all message $M$, if the recipient is not revoked at time period $\mathsf{t}$ and if all parties follow the prescribed algorithms, then we have $\mathsf{Dec}(\mathsf{DK}_{\mathsf{id},\mathsf{t}}, CT'_{\mathsf{id},\mathsf{t}}) = M$.

Next, we give the semantic security against adaptive IDentity Chosen Plaintext Attacks for Server-aided Revocable IBE scheme (shorted as SR-ID-CPA security). We begin by introducing the oracles that can be accessed adaptively and repeatedly by an adversary.

- (Public Key Oracle) $\mathcal{O}^{\mathrm{sr-ibe}}_{\mathsf{PubKG}}(\cdot)$: On input an identity $\mathsf{id}$, it outputs a public key $\mathsf{PK}_{\mathsf{id}}$ by running $\mathsf{PubKG}(\mathsf{MSK}, \mathsf{id}, \mathsf{ST})$.
- (Transformation Key Update Oracle) $\mathcal{O}^{\mathrm{sr-ibe}}_{\mathsf{TKeyUp}}(\cdot)$: On input a time period $\mathsf{t}$, it outputs $\mathsf{KU}_{\mathsf{TK},\mathsf{t}}$ by running $\mathsf{TKeyUp}(\mathsf{MSK}, \mathsf{t}, \mathsf{RL}, \mathsf{ST})$.
- (Private Key Oracle) $\mathcal{O}^{\mathrm{sr-ibe}}_{\mathsf{PrivKG}}(\cdot)$: On input an identity $\mathsf{id}$, it outputs a private key $\mathsf{SK}_{\mathsf{id}}$ through running $\mathsf{PrivKG}(\mathsf{MSK}, \mathsf{id})$.
- (Decryption Key Oracle) $\mathcal{O}^{\mathrm{sr-ibe}}_{\mathsf{DecKG}}(\cdot,\cdot)$: On input an identity $\mathsf{id}$ and a time period $\mathsf{t}$, it outputs $\mathsf{DK}_{\mathsf{id},\mathsf{t}}$ by running $\mathsf{DecKG}(\mathsf{SK}_{\mathsf{id}}, \mathsf{t})$, where $\mathsf{SK}_{\mathsf{id}}$ is obtained via $\mathsf{PrivKG}(\mathsf{MSK}, \mathsf{id})$.
- (Revocation Oracle) $\mathcal{O}^{\mathrm{sr-ibe}}_{\mathsf{Revoke}}(\cdot,\cdot)$: On input an identity $\mathsf{id}$ and a time period $\mathsf{t}$, it outputs an updated revocation list $\mathsf{RL}$ by running $\mathsf{Revoke}(\mathsf{id}, \mathsf{t}, \mathsf{RL}, \mathsf{ST})$.

**Definition 2 (SR-ID-CPA Security).** *Let* $\mathcal{O}^{\mathrm{ibe}}_{\mathrm{sr}}$ *denote the family of the oracles defined above. We say a SR-IBE scheme is SR-ID-CPA secure, if for any PPT adversary* $\mathcal{A}$*, the function* $\mathsf{Adv}^{\mathrm{sr-id-ibe}}_{\mathrm{SR-IBE},\mathcal{A}}(\kappa)$ *is negligible in* $\kappa$*, where*

$$\mathsf{Adv}^{\mathrm{sr-id-cpa}}_{\mathrm{SR-IBE},\mathcal{A}}(\kappa) := \left| \Pr\left[ b' = b : \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Sys}(\kappa) \\ (\mathsf{MPK}, \mathsf{MSK}, \mathsf{RL}, \mathsf{ST}) \leftarrow \mathsf{Setup}(\mathsf{pp}) \\ (\mathsf{id}^*, \mathsf{t}^*, M_0, M_1) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathrm{ibe}}_{\mathrm{sr}}}(\mathsf{MPK}) \\ b \leftarrow_R \{0,1\} \\ CT_{\mathsf{id}^*,\mathsf{t}^*} \leftarrow \mathsf{Enc}(\mathsf{MPK}, \mathsf{id}^*, \mathsf{t}^*, M_b) \\ b' \leftarrow \mathcal{A}^{\mathcal{O}^{\mathrm{ibe}}_{\mathrm{sr}}}(CT_{\mathsf{id}^*,\mathsf{t}^*}) \end{array} \right] - \frac{1}{2} \right|.$$

*In the above definition, the following conditions must hold:*

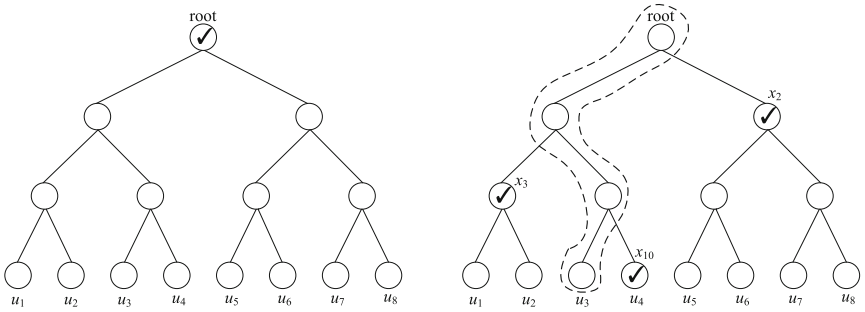
1. $M_0, M_1 \in \mathcal{M}$ *and* $|M_0| = |M_1|$, *where* $\mathcal{M}$ *is the message space.*
2. $\mathcal{O}^{\mathrm{sr-ibe}}_{\mathsf{TKeyUp}}(\cdot)$ *and* $\mathcal{O}^{\mathrm{sr-ibe}}_{\mathsf{Revoke}}(\cdot,\cdot)$ *can be queried only in non-decreasing order of time.*
3. $\mathcal{O}^{\mathrm{sr-ibe}}_{\mathsf{Revoke}}(\cdot,\cdot)$ *can not be queried on time* $\mathsf{t}$ *if* $\mathcal{O}^{\mathrm{sr-ibe}}_{\mathsf{TKeyUp}}(\cdot)$ *has been queried on time* $\mathsf{t}$.
4. *If the private key generation oracle* $\mathcal{O}^{\mathrm{sr-ibe}}_{\mathsf{PrivKG}}(\cdot)$ *is queried on the challenge identity* $\mathsf{id}^*$, *the revocation oracle* $\mathcal{O}^{\mathrm{sr-ibe}}_{\mathsf{Revoke}}(\cdot,\cdot)$ *must be queried on* $(\mathsf{id}^*, \mathsf{t})$ *for any* $\mathsf{t} \leq \mathsf{t}^*$.
5. *If* $\mathsf{id}^*$ *is not revoked at time* $\mathsf{t}^*$, $\mathcal{O}^{\mathrm{sr-ibe}}_{\mathsf{DecKG}}(\cdot,\cdot)$ *can not be queried on* $(\mathsf{id}^*, \mathsf{t}^*)$.

The above security notion essentially captures the following scenarios: (1) a revoked user cannot access ciphertexts encrypted under a future time period; (2) a compromised decryption key for $(\mathsf{id}, \mathsf{t})$ only endangers the privacy of ciphertexts encrypted under $(\mathsf{id}, \mathsf{t})$; (3) Except the KGC, all other parities can collude. A user's decryption key is updated by the user himself; hence, no communication is required between the user and the KGC once the user's private key was distributed. Moreover, all communications between the KGC and the server take place over a public channel which can be accessed by the adversary. The server does not hold any secret data, it simply functions as a computing device.

## 4 Construction of SR-IBE Scheme

### 4.1 The Node Selection Algorithm: KUNodes

In this subsection, we recall the node selection algorithm KUNodes as in previous revocable IBE systems [3,24]. This algorithm computes a minimal set $\mathsf{Y}$ of nodes for which transformation key updates have to be published so that the server can generate the transformation keys corresponding to non-revoked users.




(a) No user is revoked: $\mathsf{Y} = \{\text{root}\}$ (b) $u_3$ is revoked: $\mathsf{Y} = \{x_2, x_3, x_{10}\}$

**Fig. 2.** Illustration of KUNodes Algorithm

We employ similar notations as in [3]. For a binary tree $\mathsf{BT}$ with $N$ leaves, corresponding to $N$ users, we denote by $\mathsf{root}$ the root node of the tree $\mathsf{BT}$. If $\theta$ is a leaf node, we let $\mathsf{Path}(\theta)$ stand for the set of nodes on the path from $\theta$ to $\mathsf{root}$ (both $\theta$ and $\mathsf{root}$ are inclusive). If $\theta$ is a non-leaf node, then $\theta_l$ and $\theta_r$ denote left and right children of $\theta$. The node selection algorithm $\mathsf{KUNodes}$ takes as input the binary tree $\mathsf{BT}$, the revocation list $\mathsf{RL}$ and a revocation time $\mathsf{t}$, and works as follows: it first marks all ancestors of users that were revoked by revocation time $\mathsf{t}$ as revoked nodes. Then, it outputs all the non-revoked children of revoked nodes. A simple pictorial depiction of $\mathsf{KUNodes}$ is given in Fig. 2. Below is the formal definition.

$\mathsf{KUNodes}(\mathsf{BT}, \mathsf{RL}, \mathsf{t})$ //Node selection algorithm
$\mathsf{X}, \mathsf{Y} \leftarrow \emptyset$
$\forall(\theta_i, \mathsf{t}_i) \in \mathsf{RL}$, if $\mathsf{t}_i \leq \mathsf{t}$, then add $\mathsf{Path}(\theta_i)$ to $\mathsf{X}$
$\forall x \in \mathsf{X}$, if $x_l \notin \mathsf{X}$, then add $x_l$ to $\mathsf{Y}$; if $x_r \notin \mathsf{X}$, then add $x_r$ to $\mathsf{Y}$
If $\mathsf{Y} = \emptyset$, then add $\mathsf{root}$ to $\mathsf{Y}$
Return $\mathsf{Y}$

### 4.2 The Construction

We assume that the identity space is $\{0,1\}^n$ and the time space is $\mathcal{T}$. The message space $\mathcal{M}$ is the same as that of the underlying group. Our SR-IBE scheme consists of the following algorithms:

$\mathsf{Sys}(\kappa)$: On input a security parameter $\kappa$, the KGC does the following:

1. Choose $\mathcal{G} = (\mathbb{G}, \mathbb{G}_T, p, g, \hat{e}) \leftarrow \mathcal{BP}(\kappa)$.
2. Choose a random element $h \leftarrow_R \mathbb{G}$.
3. Choose a random $n+1$-dimensional vector $U = (u_0, u_1, \ldots, u_n) \leftarrow_R \mathbb{G}^{n+1}$ and a random 2-dimensional vector $(v_0, v_1) \leftarrow_R \mathbb{G}^2$.
4. Define and return $\mathsf{pp} = (\mathcal{G}, h, U, v_0, v_1)$.

$\mathsf{Setup}(\mathsf{pp}, N)$: On input $\mathsf{pp}$ and a maximal number of users $N$, the KGC does the following:

1. Choose two random exponents $\alpha, \beta \leftarrow_R \mathbb{Z}_p$ and set $g_1 = g^{\alpha+\beta}$.
2. Initialize the revocation list $\mathsf{RL} = \emptyset$ and the state $\mathsf{ST} := \mathsf{BT}$, where $\mathsf{BT}$ is a binary tree with $N$ leaves.
3. Define $\mathsf{MPK} = g_1$ and $\mathsf{MSK} = (h^\alpha, h^\beta)$.
4. Return $(\mathsf{MPK}, \mathsf{MSK}, \mathsf{RL}, \mathsf{ST})$.

$\mathsf{PubKG}(\mathsf{MSK}, \mathsf{id}, \mathsf{ST})$: Parse $\mathsf{MSK}$ as $(h^\alpha, h^\beta)$ and $\mathsf{ST}$ as $\mathsf{BT}$. The KGC does the following:

1. Pick an unassigned leaf note $\theta$ from $\mathsf{BT}$ and store $\mathsf{id}$ in this node.
2. For each node $x \in \mathsf{Path}(\theta)$, it does the following:
   (a) Recall $g_{x,1}$ from $\mathsf{BT}$. If it is undefined, choose $g_{x,2} \leftarrow_R \mathbb{G}$, set $g_{x,1} = h^\alpha / g_{x,2}$ and store the pair $(g_{x,1}, g_{x,2})$ in node $x$.

(b) Choose $r_x \leftarrow_R \mathbb{Z}_p$.
(c) Compute $(P_{x,1}, P_{x,2}) = (g_{x,1} \cdot F_{\mathsf{Wat},U}(\mathsf{id})^{r_x}, g^{r_x})$.

3. Return $\mathsf{PK}_{\mathsf{id}} = \{(x, P_{x,1}, P_{x,2})\}_{x \in \mathsf{Path}(\theta)}$ and an updated state $\mathsf{ST}$.

$\mathsf{TKeyUp}(\mathsf{MSK}, \mathsf{t}, \mathsf{RL}, \mathsf{ST})$: Parse $\mathsf{MSK}$ as $(h^\alpha, h^\beta)$ and $\mathsf{ST}$ as $\mathsf{BT}$. For all $x \in \mathsf{KUNodes}(\mathsf{BT}, \mathsf{RL}, \mathsf{t})$, the KGC does the following:

1. Fetch $g_{x,2}$ from $\mathsf{BT}$. If it is not defined, similar as in the public key generation algorithm, choose $(g_{x,1}, g_{x,2}) \in \mathbb{G} \times \mathbb{G}$ such that $g_{x,1} \cdot g_{x,2} = h^\alpha$ and store it in the node $x$.
2. Choose $s_x \leftarrow_R \mathbb{Z}_p$.
3. Compute $(Q_{x,1}, Q_{x,2}) = (g_{x,2} \cdot (v_0 v_1^{\mathsf{t}})^{s_x}, g^{s_x})$.
4. Return $\mathsf{KU}_{\mathsf{TK},\mathsf{t}} = \{(x, Q_{x,1}, Q_{x,2}))_{x \in \mathsf{KUNodes}(\mathsf{BT},\mathsf{RL},\mathsf{t})}$ to the server.

$\mathsf{TranKG}(\mathsf{PK}_{\mathsf{id}}, \mathsf{KU}_{\mathsf{TK},\mathsf{t}})$: On input $\mathsf{PK}_{\mathsf{id}}$ and $\mathsf{KU}_{\mathsf{TK},\mathsf{t}}$, the server generates a transformation key for $(\mathsf{id}, \mathsf{t})$ as follows: Parse $\mathsf{PK}_{\mathsf{id}}$ as $\{(x, P_{x,1}, P_{x,2})\}_{x \in I}$ and $\mathsf{KU}_{\mathsf{id},\mathsf{t}}$ as $\{(x, Q_{x,1}, Q_{x,2})\}_{x \in J}$ for some sets of nodes $I$, $J$. If $I \cap J = \emptyset$ (i.e., no pair $(i,j) \in I \times J$ such that $i = j$), return $\perp$; else choose an arbitrary $x \in I \cap J$ and $r'_x, s'_x \leftarrow_R \mathbb{Z}_p$, compute and return $\mathsf{TK}_{\mathsf{id},\mathsf{t}} = (TK_1, TK_2, TK_3)$, where

$$\begin{cases} TK_1 = P_{x,1} \cdot Q_{x,1} \cdot F_{\mathsf{Wat},U}(\mathsf{id})^{r'_x} \cdot (v_0 v_1^{\mathsf{t}})^{s'_x} & \left(= h^\alpha \cdot F_{\mathsf{Wat},U}(\mathsf{id})^{r_x + r'_x} \cdot (v_0 v_1^{\mathsf{t}})^{s_x + s'_x}\right) \\ TK_2 = P_{x,2} \cdot g^{r'_x} & \left(= g^{r_x + r'_x}\right) \\ TK_3 = Q_{x,2} \cdot g^{s'_x} & \left(= g^{s_x + s'_x}\right) \end{cases}.$$

$\mathsf{PrivKG}(\mathsf{MSK}, \mathsf{id})$: Parse $\mathsf{MSK}$ as $(h^\alpha, h^\beta)$, the KGC does the following:

1. Choose $r_{\mathsf{id}} \leftarrow_R \mathbb{Z}_p$.
2. Compute $(D_{\mathsf{id},1}, D_{\mathsf{id},2}) = (h^\beta \cdot F_{\mathsf{Wat},U}(\mathsf{id})^{r_{\mathsf{id}}}, g^{r_{\mathsf{id}}})$.
3. Return $\mathsf{SK}_{\mathsf{id}} = (D_{\mathsf{id},1}, D_{\mathsf{id},2})$.

$\mathsf{DecKG}(\mathsf{SK}_{\mathsf{id}}, \mathsf{t})$: Parse $\mathsf{SK}_{\mathsf{id}}$ as $(D_{\mathsf{id},1}, D_{\mathsf{id},2})$. The user chooses $r'_{\mathsf{id}}, s'_{\mathsf{t}} \leftarrow_R \mathbb{Z}_p$, and then computes and returns $\mathsf{DK}_{\mathsf{id},\mathsf{t}} = (D_1, D_2, D_3)$, where

$$\begin{cases} D_1 = D_{\mathsf{id},1} \cdot F_{\mathsf{Wat},U}(\mathsf{id})^{r'_{\mathsf{id}}} \cdot (v_0 v_1^{\mathsf{t}})^{s'_{\mathsf{t}}} & \left(= h^\beta \cdot F_{\mathsf{Wat},U}(\mathsf{id})^{r_{\mathsf{id}} + r'_{\mathsf{id}}} \cdot (v_0 v_1^{\mathsf{t}})^{s'_{\mathsf{t}}}\right) \\ D_2 = D_{\mathsf{id},2} \cdot g^{r'_{\mathsf{id}}} & \left(= g^{r_{\mathsf{id}} + r'_{\mathsf{id}}}\right) \\ D_3 = g^{s'_{\mathsf{t}}} & \left(= g^{s'_{\mathsf{t}}}\right) \end{cases}.$$

$\mathsf{Encrypt}(\mathsf{MPK}, \mathsf{id}, \mathsf{t}, M)$: To encrypt a message $M$ under identity $\mathsf{id}$ and time period $\mathsf{t}$, the sender chooses $z \leftarrow_R \mathbb{Z}_p$ and sets $CT = (C_0, C_1, C_2, C_3)$, where

$$C_0 = \hat{e}(g_1, h)^z \cdot M \qquad C_1 = g^z \qquad C_2 = F_{\mathsf{Wat},U}(\mathsf{id})^z \qquad C_3 = (v_0 v_1^{\mathsf{t}})^z.$$

It returns $CT_{\mathsf{id},\mathsf{t}} = (\mathsf{id}, \mathsf{t}, CT)$ to the server.

$\mathsf{Transform}(\mathsf{TK}_{\mathsf{id},\mathsf{t}}, CT_{\mathsf{id},\mathsf{t}})$: Parse $\mathsf{TK}_{\mathsf{id},\mathsf{t}}$ as $(TK_1, TK_2, TK_3)$ and $CT_{\mathsf{id},\mathsf{t}}$ as $(\mathsf{id}, \mathsf{t}, C_0, C_1, C_2, C_3)$. It computes

$$K_1 = \frac{\hat{e}(C_1, TK_1)}{\hat{e}(C_2, TK_2) \cdot \hat{e}(C_3, TK_3)} \quad \left(= \hat{e}(g^\alpha, h)^z\right)$$

Then, it sets $C'_0 = C_0 / K_1$ and returns $CT'_{\mathsf{id},\mathsf{t}} = (\mathsf{id}, \mathsf{t}, C'_0, C_1, C_2, C_3)$ to the recipient.

$\mathsf{Decrypt}(\mathsf{DK}_{\mathsf{id},\mathsf{t}}, CT'_{\mathsf{id},\mathsf{t}})$: Parse $\mathsf{DK}_{\mathsf{id},\mathsf{t}}$ as $(D_1, D_2, D_3)$ and $CT'_{\mathsf{id},\mathsf{t}}$ as $(\mathsf{id}, \mathsf{t}, (C'_0, C_1, C_2, C_3))$. It computes

$$K_2 = \frac{\hat{e}(C_1, D_1)}{\hat{e}(C_2, D_2) \cdot \hat{e}(C_3, D_3)} \left( = \frac{\hat{e}(g^z, h^\beta \cdot F_{\mathsf{Wat},U}(\mathsf{id})^{r_{\mathsf{id}}+r'_{id}} \cdot (v_0 v_1^{\mathsf{t}})^{s_{\mathsf{t}}+s'_{\mathsf{t}}})}{\hat{e}(F_{\mathsf{Wat},U}(\mathsf{id})^z, g^{r_{\mathsf{id}}+r'_{\mathsf{id}}}) \cdot \hat{e}((v_0 v_1^{\mathsf{t}})^z, g^{s_{\mathsf{t}}+s'_{\mathsf{t}}})} = \hat{e}(g^\beta, h)^z \right)$$

and returns $M = C'_0/K_2$.

## 5 Security Proof

Correctness of the scheme can be verified by direct calculation. We omit it here and only focus on its security proof below.

**Theorem 2.** *If there exists a PPT adversary $\mathcal{A}$ breaking the SR-ID-CPA security of the proposed SR-IBE scheme, then we can construct a PPT adversary $\mathcal{B}$ breaking the ID-CPA security of the Waters IBE scheme. Moreover,*

$$\mathsf{Adv}^{\mathrm{sr-id-cpa}}_{\mathrm{SR-IBE},\mathcal{A}}(\kappa) \leq 2Q|\mathcal{T}| \cdot \mathsf{Adv}^{\mathrm{id-cpa}}_{\mathrm{IBE}_{\mathsf{Wat}},\mathcal{B}}(\kappa)$$

*where $Q$ is the maximal number of oracle queries issued by the adversary $\mathcal{A}$ and $\mathcal{T}$ is the set of revocation time periods.*

**Proof Outline.** Here, we only highlight the center idea of proof. We refer the interested reader to Appendix A for the formal proof.

At a high level, we can view our scheme as a combination of a traditional revocable IBE scheme of Seo and Emura [24] (with master secret key $h^\alpha$) and a two-level HIBE scheme derived from the Waters IBE scheme [28] (with master secret key $h^\beta$). The first component is again built from the Waters IBE scheme. The long-term private keys, hold by users in RIBE, are now publicly delegated to the server. Each user actually holds one of the first level secret keys of the underlying two-level HIBE scheme as his long-term private key. In the proof, we divide the adversaries into the following two distinct types.

**Type I Adversary:** The adversary issues a query to the private key oracle $\mathcal{O}^{\mathrm{sr-ibe}}_{\mathsf{PrivKG}}(\cdot)$ with the challenge identity $\mathsf{id}^*$. So, the identity $\mathsf{id}^*$ must be revoked before the challenge time $\mathsf{t}^*$.

**Type II Adversary:** The adversary *never* issues a query to the private key oracle $\mathcal{O}^{\mathrm{sr-ibe}}_{\mathsf{PrivKG}}(\cdot)$ with the challenge identity $\mathsf{id}^*$, but it may query the decryption key oracle $\mathcal{O}^{\mathrm{sr-ibe}}_{\mathsf{DecKG}}(\cdot,\cdot)$ with $(\mathsf{id}^*, \mathsf{t})$ as long as $\mathsf{t} \neq \mathsf{t}^*$.

We can view our proof as a reduction to either the security of the underlying revocable IBE scheme or the security of the HIBE scheme according to which type of adversaries our scheme is faced with. For the first type of adversary, it can obtain the challenge first-level secrete key of the HIBE scheme, and hence any decryption key. So, we cannot reduce our security to the underlying HIBE. Instead, we reduce it to the security of the underlying RIBE. The RIBE oracle can answer all public key queries and transformation key update queries issued by

the adversary since the challenge identity must be revoked before the challenge time. For the second type of adversary, the adversary does not query the private key of the challenge identity, so it can query for long-term public keys and key updates even for challenge identity and time period. In this case, it is possible to reduce our security to that of the underlying HIBE since the adversary is forbidden to query the decryption key for challenge identity and time period.

## 6 Conclusion

In this paper, we proposed a new system model for revocable IBE, named server-aided revocable IBE (SR-IBE). The model has two desirable features which make it especially suitable for users with limited computation, communication, and storage capabilities. First, SR-IBE delegates almost all of the workload imposed on users in previous non-server aided revocable IBE systems to an untrusted third party server. Second, SR-IBE only requires each user to store a short long-term private key such that a user can update decryption keys all by himself, without having to communicate with either the KGC or the third party server. We also presented a concrete SR-IBE scheme and proved that it is secure against both adaptive-ID attacks and decryption exposure attacks under the decisional Bilinear Diffie-Hellman assumption in the standard model. An ideal application of the SR-IBE is secure Email systems supporting mobile users in which Email servers could naturally double as the untrusted third party server.

**Acknowledgments.** We thank the anonymous reviewers for their helpful comments. The work of Robert H. Deng was supported by Singapore Ministry of Education Academic Research Fund Tier 1 under the research grant 14-C220-SMU-06. The work of Shengli Liu was supported by the National Natural Science Foundation of China (NSFC Grant No. 61170229 and 61373153), the Specialized Research Fund for the Doctoral Program of Higher Education (Grant No. 20110073110016), and the Scientific innovation projects of Shanghai Education Committee (Grant No. 12ZZ021).

## A Proof of Theorem 2

*Proof.* Let $\mathcal{A}$ be the adversary that breaks the SR-ID-CPA security of the above SR-IBE scheme. We construct an adaptive ID-CPA adversary (simulator) of the Waters IBE scheme using $\mathcal{A}$ as a subroutine.

The simulator is given a challenge instance of the Waters IBE scheme, including a system parameter $\mathsf{pp}_{\mathsf{Wat}} = (\mathcal{G}, h, U)$, a master public key $\mathsf{MPK}_{\mathsf{Wat}} = g^{\gamma^*}$ for some unknown exponent $\gamma^*$, private key generation oracle $\mathcal{O}^{\mathrm{ibe}}_{\mathsf{PrivKG}_{\mathsf{Wat}}}(\cdot)$ and an encryption oracle $\mathcal{O}^{\mathrm{ibe}}_{\mathsf{Enc}_{\mathsf{Wat}}}(\cdot,\cdot,\cdot)$[1]. The simulator first randomly guesses the challenge revocation time period $\mathsf{t}^* \leftarrow_R \mathcal{T}$.

In the proof, the simulator has to randomly guess which type of adversaries (as described in Sect. 5) is going to be.

[1] The encryption oracle is defined as follows: on input $(\mathsf{id}, M_0, M_1)$, output $CT^{\mathsf{Wat}}_{\mathsf{id}} = (C_0, C_1, C_2) \leftarrow \mathsf{Enc}_{\mathsf{Wat}}(\mathsf{id}, M_b)$, where $b \leftarrow_R \{0,1\}$.

*Proof (Proof of Type I Adversary).* Let $Q$ be the maximal number of queries issued by the adversary. The simulator randomly guesses $i^* \leftarrow_R \{1, \ldots, Q\}$, assuming that $\mathsf{id}^*$ firstly appears in the $i^*$-th query among all queries issued by the adversary. We show at the end of the proof that the guess holds with probability $1/Q$. Additionally, the simulator randomly chooses an unassigned leaf node $\theta^*$ for storing the challenge identity $\mathsf{id}^*$.

The simulator simulates the SR-ID-CPA game as follows:

**System Parameter**: The simulator chooses random exponents $a, b \leftarrow_R \mathbb{Z}_p$ and sets $v_0 = h^{\mathsf{t}^*} \cdot g^a$, $v_1 = h^{-1} \cdot g^b$. The simulator returns $\mathsf{pp} = (\mathsf{pp}_{\mathsf{Wat}}, v_0, v_1)$ to the adversary and holds $(a, b)$.

**Setup:** The simulator chooses a random exponent $\beta \leftarrow_R \mathbb{Z}_p$. It sets $\mathsf{MPK} = g_1 = \mathsf{MPK}_{\mathsf{Wat}}$ and $\mathsf{MSK} = (h^{\gamma^* - \beta}, h^{\beta})$, where $\gamma^* - \beta = \alpha$ is unknown to the simulator. The simulator sends $\mathsf{MPK}$ to the adversary. It holds $h^{\beta}$, an empty revocation list $\mathsf{RL}$ and an initial state of a binary tree $\mathsf{ST} := \mathsf{BT}$.

**Private Key Oracle and Decryption Key Oracle:** Since the simulator knows the master secret key part $h^{\beta}$, it can answer all queries issued by the adversary to these two oracles. If the $i^*$-th query appears among the queries issued to these two oracles, the simulator knows the challenge identity $\mathsf{id}^*$ and stores it in the pre-assigned leaf node $\theta^*$.

**Transformation Key Update Oracle:** When $\mathcal{A}$ issues a transformation key update generation query on time period $\mathsf{t}$, the simulator does the following:

1. For all $x \in \mathsf{KUNodes}(\mathsf{BT}, \mathsf{RL}, \mathsf{t})$, fetch $\eta_x$ from node $x$ of $\mathsf{BT}$ if it is defined. Otherwise, randomly choose $\eta_x \in \mathbb{G}$ and store it in the node $x$.
2. Choose $s_x \leftarrow_R \mathbb{Z}_p$.
3. Compute $(x, Q_{x,1}, Q_{x,2})$ as follows:

$$(Q_{x,1}, Q_{x,2}) = \begin{cases} (\eta_x^{-1} \cdot h^{-\beta} \cdot (v_0 v_1^{\mathsf{t}})^{s_x}, g^{s_x}) & \text{If } x \notin \mathsf{Path}(\theta^*) \\ (\eta_x^{-1} \cdot h^{-\beta} \cdot h^{(\mathsf{t}^* - \mathsf{t}) s_x} \cdot g_1^{\frac{a+b\mathsf{t}}{\mathsf{t}-\mathsf{t}^*}} \cdot g^{(a+b\mathsf{t}) s_x}, g_1^{\frac{1}{\mathsf{t}-\mathsf{t}^*}} \cdot g^{s_x}) & \text{Otherwise} \end{cases}.$$

4. Return $\mathsf{KU}_{\mathsf{TK},\mathsf{t}} = \{(x, Q_{x,1}, Q_{x,2})\}_{x \in \mathsf{KUNodes}(\mathsf{BT},\mathsf{RL},\mathsf{t})}$ to the adversary.

*Remark 1.* For all $x \in \mathsf{Path}(\theta^*)$, if $\mathsf{t} \neq \mathsf{t}^*$, $(Q_{x,1}, Q_{x,2})$ can be rewritten in the form $(Q_{x,1}, Q_{x,2}) = \left(\eta_x^{-1} \cdot h^{\gamma^* - \beta} \cdot (v_0 v_1^{\mathsf{t}})^{s'_x}, g^{s'_x}\right)$ for some unknown exponent $s'_x = -\frac{\gamma^*}{\mathsf{t}^* - \mathsf{t}} + s_x$.

**Public Key Oracle:** Let $j_{\mathsf{PK}}$ denote $\mathcal{A}$'s $j_{\mathsf{PK}}$-th query issued to the public key oracle. When $\mathcal{A}$ issues a public key generation query on identity $\mathsf{id}$, if the identity $\mathsf{id}$ firstly appears in $\mathcal{A}$'s queries, the simulator randomly chooses an unassigned leaf node $\theta$ and stores $\mathsf{id}$ in $\theta$. We consider the following three cases:

**Case 1:** $j_{\mathsf{PK}} < i^*$. In this case, $\mathsf{id} \neq \mathsf{id}^*$. So, the simulator can query the Waters user key generation oracle $\mathcal{O}^{\mathrm{ibe}}_{\mathsf{PrivKG}_{\mathsf{Wat}}}(\cdot)$ on identity $\mathsf{id}$ and obtain a

Waters "private key" $(d_1, d_2)$ for identity $\mathsf{id}$. For each node $x \in \mathsf{Path}(\theta)$, the simulator recalls $\eta_x$ if it is defined. If not, it chooses $\eta_x \leftarrow_R \mathbb{G}$ and stores $\eta_x$ in node $x$. Then, it chooses $r_x \leftarrow_R \mathbb{Z}_p$ and computes $(x, P_{x,1}, P_{x,2})$ as follows

$$(P_{x,1}, P_{x,2}) = \begin{cases} (\eta_x \cdot d_1 \cdot F_{\mathsf{Wat},U}(\mathsf{id})^{r_x}, d_2 \cdot g^{r_x}) & \text{If } x \notin \mathsf{Path}(\theta^*) \\ (\eta_x \cdot F_{\mathsf{Wat},U}(\mathsf{id})^{r_x}, g^{r_x}) & \text{Otherwise} \end{cases}.$$

The simulator returns $\mathsf{PK}_{\mathsf{id}} = \{(x, P_{x,1}, P_{x,2})\}_{x \in \mathsf{KUNodes(BT,RL,t)}}$ to the adversary.

**Case 2:** $j_{\mathsf{PK}} = i^*$. In this case, the simulator knows the challenge identity $\mathsf{id}^*$ and stores it in the pre-assigned leaf node $\theta^*$. For each $x \in \mathsf{Path}(\theta^*)$, the simulator recalls $\eta_x$ if it is defined. If not, it chooses $\eta_x \leftarrow_R \mathbb{G}$ and stores $\eta_x$ in node $x$. It then chooses $r_x \leftarrow_R \mathbb{Z}_p$ and computes $(x, P_{x,1}, P_{x,2})$ as follows

$$(P_{x,1}, P_{x,2}) = (\eta_x \cdot F_{\mathsf{Wat},U}(\mathsf{id}^*)^{r_x}, g^{r_x}).$$

The simulator returns $\mathsf{PK}_{\mathsf{id}^*} = \{(x, P_{x,1}, P_{x,2})\}_{x \in \mathsf{Path}(\theta^*)}$ to the adversary.

**Case 3:** $j_{\mathsf{PK}} > i^*$. In this case, the simulator knows the challenge identity $\mathsf{id}^*$[2]. If $\mathsf{id} \neq \mathsf{id}^*$, the simulator does the same process as in Case 1. Otherwise, it does the same process as in Case 2.

*Remark 2.* Observe that for each node $x \in \mathsf{BT}$, the pair of values $(g_{x,1}, g_{x,2})$ is well defined. For example, if $x \notin \mathsf{Path}(\theta^*)$, the simulator in fact implicitly defined $g_{x,1} = \eta_x \cdot h^{\gamma^*}$ and $g_{x,2} = \eta_x^{-1} \cdot h^{-\beta}$ so that $g_{x,1} \cdot g_{x,2} = h^{\gamma^* - \beta}$. So, from the above constructions of transformation key updates and public keys, if an identity $\mathsf{id}$ is not revoked at time period $\mathsf{t}$, the transformation key $\mathsf{TK}_{\mathsf{id},\mathsf{t}}$ always has the form $\mathsf{TK}_{\mathsf{id},\mathsf{t}} = (h^{\gamma^* - \beta} \cdot F_{\mathsf{Wat},U}(\mathsf{id})^{r_x} \cdot (v_0 v_1^{\mathsf{t}})^{s_x}, g^{r_x}, g^{s_x})$ for some random exponents $r_x, s_x \leftarrow_R \mathbb{Z}_p$.

**Challenge Ciphertext:** When $\mathcal{A}$ issues a challenge ciphertext query $(\mathsf{id}^*, \mathsf{t}^*, M_0, M_1)$, the simulator submits $(\mathsf{id}^*, M_0, M_1)$ to the Waters encryption oracle $\mathcal{O}^{\mathrm{ibe}}_{\mathsf{Enc}_{\mathsf{Wat}}}(\cdot,\cdot,\cdot)$ and obtains the Waters challenge ciphertext $CT^{\mathsf{Wat}}_{\mathsf{id}^*} = (C_0^{\mathsf{Wat}}, C_1^{\mathsf{Wat}}, C_2^{\mathsf{Wat}})$ where

$$C_0^{\mathsf{Wat}} = \hat{e}(\mathsf{MPK}_{\mathsf{Wat}}, h)^z \cdot M_b \qquad C_1^{\mathsf{Wat}} = g^z \qquad C_2^{\mathsf{Wat}} = F_{\mathsf{Wat},U}(\mathsf{id}^*)^z$$

for some unknown random exponent $z \leftarrow_R \mathbb{Z}_p$. The simulator computes $C_3 = (C_1^{\mathsf{Wat}})^{a+b\mathsf{t}^*}$ and returns $CT_{\mathsf{id}^*,\mathsf{t}^*} = (C_0^{\mathsf{Wat}}, C_1^{\mathsf{Wat}}, C_2^{\mathsf{Wat}}, C_3)$ to $\mathcal{A}$.

**Guess:** Finally, $\mathcal{A}$ outputs a bit $b'$. The simulator forwards it to its own challenger.

**Discussion 1.** *Since the simulated master public key* $\mathsf{MPK} = \mathsf{MPK}_{\mathsf{Wat}}$ *and* $C_3 = C_1^{a+b\mathsf{t}^*} = (v_0 v_1^{\mathsf{t}^*})^z$*, the simulated challenge ciphertext has the right distribution as that in the original SR-ID-CPA game. Recall that all the oracles are also well*

[2] It may be obtained from previous queries issued to other oracles.

*simulated. So, if the simulator correctly guesses the challenge time period* $\mathsf{t}^*$ *and the index* $i^*$*, the simulator perfectly simulates Type I adversary* $\mathcal{A}$*'s environment in the SR-ID-CPA game. Recall that all guesses are randomly and independently chosen from the corresponding sets and the simulated SR-ID-CPA game only depends on the simulator's guesses, not depending on the adversary's behaviour. So, the simulator successfully simulates the SR-ID-CPA game with probability at least* $1/(Q|\mathcal{T}|)$*, i.e.,* $\mathsf{Adv}^{\mathrm{sr-id-cpa}}_{\mathrm{SR-IBE},\mathcal{A}}(\kappa) \leq Q|\mathcal{T}| \cdot \mathsf{Adv}^{\mathrm{id-cpa}}_{\mathsf{Wat},\mathcal{B}}(\kappa)$*, where* $\mathcal{B}$ *is an adversary (the simulator) attacking the Waters IBE scheme.*

*Proof (Proof of Type II Adversary).* The simulator simulates the SR-ID-CPA game for Type II adversary as follows:

**System Parameter:** The simulator chooses $a, b \leftarrow_R \mathbb{Z}_p$ and sets $v_0 = h^{\mathsf{t}^*} \cdot g^a$, $v_1 = h^{-1} \cdot g^b$. It returns $\mathsf{pp} = (\mathsf{pp}_{\mathsf{Wat}}, v_0, v_1)$ to the adversary and holds $(a, b)$.

**Setup:** The simulator chooses $\alpha \leftarrow_R \mathbb{Z}_p$ and implicitly sets $\beta = \gamma^* - \alpha$. It defines $\mathsf{MPK} = \mathsf{MPK}_{\mathsf{Wat}}$ and $\mathsf{MSK} = (h^\alpha, h^\beta)$, where $h^\beta = h^{\gamma^* - \alpha}$ is unknown. The simulator sends $\mathsf{MPK}$ to the adversary, and holds $h^\alpha$, an empty revocation list $\mathsf{RL}$ and a state of a binary tree $\mathsf{ST} := \mathsf{BT}$.

**Public Key Oracle and Transformation Key Update Oracle:** Note that, the simulator knows the master secret key part $h^\alpha$. So, it can answer all the queries issued by the adversary to these two oracles.

**Private Key Oracle:** Recall that a Type II adversary does not query the private key $\mathsf{SK}_{\mathsf{id}^*}$. When the adversary issues a private key generation query for identity $\mathsf{id}$, the simulator can answer it as follows:

1. Forward $\mathsf{id}$ to the Waters private key oracle $\mathcal{O}^{\mathrm{ibe}}_{\mathsf{PrivKG}_{\mathsf{Wat}}}(\cdot)$ and obtain a Waters private key $(d_1, d_2)$ for identity $\mathsf{id}$.
2. Set $D_{\mathsf{id},1} = h^{-\alpha} \cdot d_1$ and $D_{\mathsf{id},2} = d_2$.
3. Return $\mathsf{SK}_{\mathsf{id}} = (D_{\mathsf{id},1}, D_{\mathsf{id},2})$ to $\mathcal{A}$.

**Decryption Key Oracle:** When $\mathcal{A}$ issues a decryption key generation query on $(\mathsf{id}, \mathsf{t})$, the simulator first checks whether $\mathsf{t} = \mathsf{t}^*$. If so, we must have $\mathsf{id} \neq \mathsf{id}^*$. The simulator first involves private key oracle with $\mathsf{id}$ to obtain private key $\mathsf{SK}_{\mathsf{id}}$ and thereby the decryption key $\mathsf{DK}_{\mathsf{id},\mathsf{t}^*}$. Otherwise (i.e., $\mathsf{t} \neq \mathsf{t}^*$), the simulator does the following:

1. Choose $r_{\mathsf{id}}, s_{\mathsf{t}} \leftarrow_R \mathbb{Z}_p$.
2. Compute $D_1 = h^{-\alpha} \cdot F_{\mathsf{Wat},U}(\mathsf{id})^{r_{\mathsf{id}}} \cdot h^{(\mathsf{t}^* - \mathsf{t})s_{\mathsf{t}}} \cdot (g^{\beta^*})^{-\frac{a+b\mathsf{t}}{\mathsf{t}^* - \mathsf{t}}} \cdot g^{(a+b\mathsf{t})s_{\mathsf{t}}}$, $D_2 = g^{r_{\mathsf{id}}}$ and $D_3 = (g^{\beta^*})^{-\frac{1}{\mathsf{t}^* - \mathsf{t}}} \cdot g^{s_{\mathsf{t}}}$.
3. Return $\mathsf{DK}_{\mathsf{id},\mathsf{t}} = (D_1, D_2, D_3)$ to $\mathcal{A}$.

**Challenge Ciphertext Oracle:** When $\mathcal{A}$ issues a challenge ciphertext query with $(\mathsf{id}^*, \mathsf{t}^*, M_0, M_1)$, the simulator does the following:

1. Forward $(\mathsf{id}^*, M_0, M_1)$ to the Waters encryption oracle $\mathcal{O}^{\mathrm{ibe}}_{\mathsf{Enc}_{\mathsf{Wat}}}(\cdot, \cdot, \cdot)$, which will outputs a challenge ciphertext of the Waters IBE scheme $CT^{\mathsf{Wat}}_{\mathsf{id}^*,} = (C_0^{\mathsf{Wat}}, C_1^{\mathsf{Wat}}, C_2^{\mathsf{Wat}})$.
2. Compute $C_3 = (C_1^{\mathsf{Wat}})^{a+b\mathsf{t}^*}$.
3. Define $(C_0, C_1, C_2) = (C_0^{\mathsf{Wat}}, C_1^{\mathsf{Wat}}, C_2^{\mathsf{Wat}})$.

4. Return $CT_{\mathsf{id}^*,\mathsf{t}^*} = (C_0, C_1, C_2, C_3)$ to $\mathcal{A}$.

**Guess:** Finally, the adversary outputs a guess bit $b'$, which is also the guess bit of the simulator.

**Discussion 2** *Similar to previous analysis, the challenge ciphertext is well distributed. Recall that a valid Waters private key for identity* $\mathsf{id}$ *is of the form* $\mathsf{SK}_{\mathsf{id}}^{\mathsf{Wat}} = (d_1, d_2) = (h^{\gamma} \cdot F_{\mathsf{Wat},U}(\mathsf{id})^{r_{\mathsf{id}}}, g^{r_{\mathsf{id}}})$*, where* $r_{\mathsf{id}} \leftarrow_R \mathbb{Z}_p$*. So,*

$$(D_{\mathsf{id},1}, D_{\mathsf{id},2}) = (h^{-\alpha} \cdot d_1, d_2) = (h^{\beta} \cdot F_{\mathsf{Wat},U}(\mathsf{id})^{r_{\mathsf{id}}}, g^{r_{\mathsf{id}}})$$

*is a valid SR-IBE private key for identity* $\mathsf{id}$*. For decryption key generation oracle, if* $\mathsf{t} = \mathsf{t}^*$*, we must have* $\mathsf{id} \neq \mathsf{id}^*$ *and hence the decryption key is well defined. If* $\mathsf{t} \neq \mathsf{t}^*$*, the decryption key* $\mathsf{DK}_{\mathsf{id},\mathsf{t}}$ *can be rewritten as the form* $(D_1, D_2, D_3) = (h^{\gamma^*-\alpha} \cdot F_{\mathsf{Wat},U}(\mathsf{id})^{r_{\mathsf{id}}} \cdot (v_0 v_1^{\mathsf{t}})^{s'_{\mathsf{t}}}, g^{r_{\mathsf{id}}}, g^{s'_{\mathsf{t}}})$ *for some unknown exponent* $s'_{\mathsf{t}} = -\frac{\gamma^*}{\mathsf{t}^*-\mathsf{t}} + s_{\mathsf{t}}$*. So, in this case, the decryption key oracle is also well defined. To conclude, if the simulator correctly guesses the challenge revocation time, the simulator perfectly simulates the SR-ID-CPA game for the Type II adversary* $\mathcal{A}$*. Since the guess is independent of the adversary's behaviour, we have* $\mathsf{Adv}^{\mathrm{sr-id-cpa}}_{\mathrm{SR-IBE},\mathcal{A}}(\kappa) \leq |\mathcal{T}| \cdot \mathsf{Adv}^{\mathrm{id-cpa}}_{\mathrm{IBE}_{\mathsf{Wat}},\mathcal{B}}(\kappa)$*, where* $\mathcal{B}$ *is an adversary (i.e., the simulator) attacking the Waters IBE scheme.*

Finally, we discuss the probability of the events that the simulator correctly guess the type of adversary. Since the adversary's behaviour is independent of the simulator's guess, with probability exactly $1/2$ the guess is correct. So,

$$\mathsf{Adv}^{\mathrm{sr-id-cpa}}_{\mathrm{SR-IBE},\mathcal{A}}(\kappa) \leq 2 \cdot \max(Q \cdot |\mathcal{T}| + |\mathcal{T}|) \cdot \mathsf{Adv}^{\mathrm{id-cpa}}_{\mathrm{IBE}_{\mathsf{Wat}},\mathcal{B}}(\kappa) = 2Q|\mathcal{T}| \cdot \mathsf{Adv}^{\mathrm{id-cpa}}_{\mathrm{IBE}_{\mathsf{Wat}},\mathcal{B}}(\kappa).$$

So, the reduction loss in our security proof is $2Q|\mathcal{T}|$. Since the Waters IBE scheme has been proven secure under the DBDH assumption with reduction loss $O(nQ)$, our SR-IBE scheme is secure under the DBDH assumption with $O(nQ^2|\mathcal{T}|)$ reduction loss, which is the same as in previous revocable IBE schemes [17,24]. This completes the proof of Theorem 2. □

## References

1. Aiello, W., Lodha, S.P., Ostrovsky, R.: Fast digital identity revocation. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 137–152. Springer, Heidelberg (1998)
2. Baek, J., Zheng, Y.: Identity-based threshold decryption. In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 262–276. Springer, Heidelberg (2004)
3. Boldyreva, A., Goyal, V., Kumar, V.: Identity-based encryption with efficient revocation. In: Ning, P., Syverson, P.F., Jha, S. (eds.) CCS 2008, pp. 417–426. ACM (2008)
4. Boneh, D., Ding, X., Tsudik, G., Wong, C.: A method for fast revocation of public key certificates and security capabilities. In: Wallach, D.S. (ed.) 10th USENIX Security Symposium, Washington, D.C., USA, 13–17 August 2001. USENIX (2001)

5. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
6. Boneh, D., Gentry, C., Hamburg, M.: Space-efficient identity based encryption without pairings. In: FOCS 2007, pp. 647–657. IEEE Computer Society (2007)
7. Canetti, R., Halevi, S., Katz, J.: A forward-secure public-key encryption scheme. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 255–271. Springer, Heidelberg (2003)
8. Cocks, C.: An identity based encryption scheme based on quadratic residues. In: Honary, B. (ed.) Cryptography and Coding 2001. LNCS, vol. 2260, pp. 360–363. Springer, Heidelberg (2001)
9. Ding, X., Tsudik, G.: Simple identity-based cryptography with mediated RSA. In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612, pp. 193–210. Springer, Heidelberg (2003)
10. Gentry, C.: Certificate-based encryption and the certificate revocation problem. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 272–293. Springer, Heidelberg (2003)
11. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Juels, A., Wright, R.N., di Vimercati, S.D.C. (eds.) CCS 2006, pp. 89–98. ACM (2006)
12. Horwitz, J., Lynn, B.: Toward hierarchical identity-based encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 466–481. Springer, Heidelberg (2002)
13. Lee, K., Lee, D.H., Park, J.H.: Efficient revocable identity-based encryption via subset difference methods. IACR Cryptology ePrint Arch. **2014**, 132 (2014)
14. Li, J., Li, J., Chen, X., Jia, C., Lou, W.: Identity-based encryption with outsourced revocation in cloud computing. IEEE Trans. Comput. 99(PrePrints), 1 (2013)
15. Liang, K., Liu, J.K., Wong, D.S., Susilo, W.: An efficient cloud-based revocable identity-based proxy re-encryption scheme for public clouds data sharing. In: Kutyłowski, M., Vaidya, J. (eds.) ESORICS 2014, Part I. LNCS, vol. 8712, pp. 257–272. Springer, Heidelberg (2014)
16. Libert, B., Quisquater, J.: Efficient revocation and threshold pairing based cryptosystems. In: Borowsky, E., Rajsbaum, S. (eds.) PODC 2003, pp. 163–171. ACM (2003)
17. Libert, B., Vergnaud, D.: Adaptive-ID secure revocable identity-based encryption. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 1–15. Springer, Heidelberg (2009)
18. Lin, H., Cao, Z., Fang, Y., Zhou, M., Zhu, H.: How to design space efficient revocable IBE from non-monotonic ABE. In: Cheung, B.S.N., Hui, L.C.K., Sandhu, R.S., Wong, D.S. (eds.) ASIACCS 2011, pp. 381–385. ACM (2011)
19. Naor, D., Naor, M., Lotspiech, J.: Revocation and tracing schemes for stateless receivers. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 41–62. Springer, Heidelberg (2001)
20. Naor, M., Nissim, K.: Certificate revocation and certificate update. IEEE J. Sel. Areas Commun. **18**(4), 561–570 (2000)
21. Park, S., Lee, K., Lee, D.H.: New constructions of revocable identity-based encryption from multilinear maps. IACR Cryptology ePrint Arch. **2013**, 880 (2013)
22. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)

23. Sakai, R., Ohgishi, K., Kasahara, M.: Cryptosystems based on pairing. In: The 2000 Symposium on Cryptography and Information Security, Okinawa, Japan, pp. 135–148 (2000)
24. Seo, J.H., Emura, K.: Revocable identity-based encryption revisited: security model and construction. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 216–234. Springer, Heidelberg (2013)
25. Seo, J.H., Emura, K.: Revocable identity-based cryptosystem revisited: security models and constructions. IEEE Trans. Inf. Forensics Secur. **9**(7), 1193–1205 (2014)
26. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)
27. Su, L., Lim, H.W., Ling, S., Wang, H.: Revocable IBE systems with almost constant-size key update. In: Cao, Z., Zhang, F. (eds.) Pairing 2013. LNCS, vol. 8365, pp. 168–185. Springer, Heidelberg (2014)
28. Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)