IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-6, Issue-7, July 2018 E-ISSN: 2347-2693

Secure Data Sharing in Cloud Computing Using Revocable-Storage Identity-Based Encryption

Maadala Chandra Sekhar[1]*, Keerthi Kethineni[2]

1*Computer Science and Engineering, Qis College Engineering and Technology, Ongole
2 Computer Science and Engineering, Qis College Engineering and Technology, Ongole

Available online at: www.ijcseonline.org

Accepted: 06/Jul/2018, Published: 31/Jul/2018

Abstract—Cloud computing provides a flexible and convenient way for data sharing, which brings various benefits for both the society and individuals. But there exists a natural resistance for users to directly outsource the shared data to the cloud server since the data often contain valuable information. Thus, it is necessary to place cryptographically enhanced access control on the shared data. Identity-based encryption is a promising cryptographically primitive to build a practical data sharing system. However, access control is not static. That is, when some user's authorization is expired, there should be a mechanism that can remove him/her from the system. Consequently, the revoked user cannot access both the previously and subsequently shared data. To this end, we propose a notion called revocable-storage identity-based encryption (RS-IBE), which can provide the forward/backward security of cipher text by introducing the functionalities of user revocation and cipher text update simultaneously. Furthermore, we present a concrete construction of RS-IBE, and prove its security in the defined security model. The performance comparisons indicate that the proposed RS-IBE scheme has advantages in terms of functionality and efficiency, and thus is feasible for a practical and cost-effective data-sharing system. Finally, we provide implementation results of the proposed scheme to demonstrate its practicability.

***Keywords*—**Cloud computing, data sharing, revocation, Identity-based encryption, ciphertext update, decryption key exposure.

I. INTRODUCTION

CLOUD computing is a paradigm that provides massive computation capacity and huge memory space at a low cost [1]. It enables users to get intended services irrespective of time and location across multiple platforms (e.g., mobiled evices, personal computers), and thus brings great convenience to cloud users. Among numerous services provided by cloud computing, cloud storage service, such as Apple's iCloud [2], Microsoft's Azure [3] and Amazon's S3 [4], can offer a more flexible and easy way to share data over the Internet, which provides various benefits for our society [5], [6]. However, it also suffers from several security threats,which are the primary concerns of cloud users [7].

Firstly, outsourcing data to cloud server implies that data is out control of users. This may cause users' hesitation since the outsourced data usually contain valuable and sensitive information. Secondly, data sharing is often implemented in an open and hostile environment, and cloud server would become a target of attacks. Even worse, cloud server itself may reveal users' data for illegal profit. Thirdly, data sharing is not static. That is, when a user's authorization gets expired, he/she should no longer possess the privilege of accessing the previously and subsequently shared data. Therefore, while outsourcing data to cloud server, users also want to control access to these data such that only those currently authorized users can share the outsourced data.

A natural solution to conquer the aforementioned problem is to use cryptographically enforced access control such as identity-based encryption (IBE). Furthermore, to overcome the above security threats, such kind of identity-based access control placed on the shared data should meet the following security goals:

- **Data confidentiality**: Unauthorized users should be prevented from accessing the plaintext of the shared data stored in the cloud server. In addition, the cloud server, which is supposed to be honest but curious, should also be deterred from knowing plaintext ofthe shared data.
- **Backward secrecy**: Backward secrecy means that, when a user's authorization is expired, or a user's secret key is compromised, he/she should be prevented from accessing the plaintext of the *subsequently* shared data that are still encrypted under his/her identity.
- **Forward secrecy**: Forward secrecy means that, when a user's authority is expired, or a user's secret key is compromised, he/she should be prevented from accessing the plaintext of the shared data that can be *previously* accessed by him/her.

The specific problem addressed in this paper is how to construct a fundamental identity-based cryptographical tool to achieve the above security goals. We also note that there exist other security issues that are equally important for a practical system of data sharing, such as the authenticity and availability of the shared data [8], [9], [10], [11], [12]. But the research on these issues is beyond the scope of this paper.

1.1 Motivation

It seems that the concept of revocable identity-based encryption (RIBE) might be a promising approach that fulfills the aforementioned security requirements for data sharing. RIBE features a mechanism that enables a sender to append the current time period to the cipher text such that the receiver can decrypt the cipher text only under the condition that he/she is not revoked at that time period. As indicated in Figure 1, a RIBE-based data sharing system works as follows: **Step 1:** The data provider (e.g., David) first decides the users (e.g., Alice and Bob) who can share the data. Then, David encrypts the data under the identities Alice and Bob, and uploads the cipher text of the shared data to the cloud server.

Step 2: When either Alice or Bob wants to get the shared data, she or he can download and decrypt the corresponding cipher text. However, for an unauthorized user and the cloud server, the plaintext of the shared data is not available.

Step 3: In some cases, e.g., Alice's authorization gets expired, David can download the cipher text of the shared data, and then decrypt-then-re-encrypt the shared data such that Alice is prevented from accessing the plaintext of the shared data, and then upload the re-encrypted data to the cloud server again.

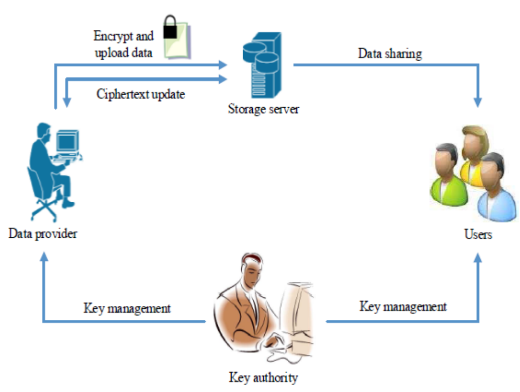


Fig. 1. A natural RIBE-based data sharing system

Key authority Data provider Storage server Users
Encrypt and upload data sharing Key management Key management Cipher text update Fig. 1. A natural RIBE-based data sharing system

Obviously, such a data sharing system can provide confidentiality and backward secrecy. Furthermore, the method of decrypting and re-encrypting all the shared data can ensure forward secrecy. However, this brings new challenges. Note that the process of decrypt-then-re-encrypt necessarily involves users' secret key information, which makes the overall data sharing system vulnerable to new attacks. In general, the use of secret key should be limited to only usual decryption, and it is inadvisable to update the cipher text periodically by using secret key.

Another challenge comes from efficiency. To update the cipher text of the shared data, the data provider has to frequently carry out the procedure of download-decrypt-encrypt- upload. This process brings great communication and computation cost, and thus is cumbersome and undesirable for cloud users with low capacity of computation and storage. One method to avoid this problem is to require the cloud server to directly re-encrypt the cipher text of the shared data. However, this may introduce cipher text extension, namely, the size of the cipher text of the shared data is linear in the number of times the shared data have been updated. In addition, the technique of proxy re-encryption can also be used to conquer the aforementioned problem of efficiency. Unfortunately, it also requires users to interact with the cloud server in order to update the cipher text of the shared data.

1.2 Related work

1.2.1 Revocable identity-based encryption The concept of identity-based encryption was introduced by Shamir [13], and conveniently instantiated by Boneh and Franklin [14]. IBE eliminates the need for providing a public
key infrastructure (PKI). Regardless of the setting of IBE or PKI, there must be an approach to revoke users from the system when necessary, e.g., the authority of some user is expired or the secret key of some user is disclosed. In the traditional PKI setting, the problem of revocation has been well studied [15], [16], [17], [18], [19], and several techniques are widely approved, such as certificate revocation list or appending validity periods to certificates. However, there are only a few studies on revocation in the setting of IBE. Boneh and Franklin [14] first proposed a natural revocation way for IBE. They appended the current time period to
the ciphertext, and non-revoked users periodically received private keys for each time period from the key authority. Unfortunately, such a solution is not scalable, since it requires the key authority to perform linear work in the number of non-revoked users. In addition, a secure channel is essential for the key authority and non-revoked users to transmit new keys. To conquer this problem, Boldyreva, Goyal and Kumar [20] introduced a novel approach to achieve efficient revocation. They used a binary tree to manage identity such that their RIBE scheme reduces the complexity of key revocation to logarithmic (instead of linear) in the maximum number of system users. However, this scheme only achieves selective security. Subsequently,

by using the aforementioned revocation technique, Libert and Vergnaud [21] proposed an adaptively secure RIBE scheme based on a variant ofWater's IBE scheme [22], Chen et al. [23] constructed a RIBE scheme from lattices. Recently, Seo and Emura [24] proposed an efficient RIBE scheme resistant to a realistic threat called decryption key exposure, whichmeans that the disclosure of decryption key for current time period has no effect on the security of decryption keys for other time periods. Inspired by the above work and [25], Liang et al. [26] introduced a cloud-based revocable identity-based proxy re-encryption that supports user revocation and ciphertext update. To reduce the complexity of revocation, they utilized a broadcast encryption
Scheme [27] to encrypt the cipher text of the update key, which is independent of users, such that only non-revoked users can decrypt the update key. However, this kind of revocation method cannot resist the collusion of revoked users and malicious non-revoked users as malicious non revoked users can share the update key with those revoked users. Furthermore, to update the cipher text, the key authority in their scheme needs to maintain a table for each user produces the re-encryption key for each time period, which significantly increases the key authority's workload.

1.2.2 Forward-secure cryptosystems

In 1997, Anderson [28] introduced the notion of forward security in the setting of signature to limit the damage of key exposure. The core idea is dividing the whole lifetime of a private key into T discrete time periods, such that the compromise of the private key for current time period cannot enable an adversary to produce valid signatures for previous time periods. Subsequently, Bellare and Miner provided formal definitions of forward-secure signature and presented practical solutions. Since then, a large number of forward-secure signature schemes [29], [30], [31], [32], [33] has been proposed.

In the context of encryption, Canetti, Halevi and Katz [34] proposed the first forward-secure public-key encryption scheme. Specifically, they firstly constructed a binary tree encryption, and then transformed it into a forward-secure encryption with provable security in the random oracle model. Based on Canetti et al's approach, Yao et al. [35] proposed a forward-secure hierarchical IBE by employing two hierarchical IBE schemes, and Nieto et al. [36] designed a forward-secure hierarchical predicate encryption.

Particularly, by combining Boldyreva et al.'s [20] revocation technique and the aforementioned idea of forward security1, in CRYPTO 2012 Sahai, Seyalioglu and Waters [37] proposed a generic construction of so-called revocablestorage attribute-based encryption, which supports user revocation and ciphertext update simultaneously. In other words, their construction provides both forward and backward secrecy. What must be pointed out is that the process of ciphertext update of this construction only needs public information. However, their construction cannot be resistant to decryption key exposure, since the decryption is a matching result of private key and update key.

1.3 Our contributions

In this paper, we introduce a notion called revocable storage identity-based encryption (RS-IBE) for building a cost-effective data sharing system that fulfills the three security goals. More precisely, the following achievements are captured in this paper: We provide formal definitions for RS-IBE and its
corresponding security model;
• We present a concrete construction of RS-IBE. The proposed scheme can provide confidentiality and backward/forward2 secrecy simultaneously;
• We prove the security of the proposed scheme in the standard model, under the decisional ℓ-Bilinear Diffie-Hellman Exponent (ℓ-BDHE) assumption. In addition, the proposed scheme can withstand decryption key exposure;
• The proposed scheme is efficient in the following ways:
1. They utilized the idea to provide the forward secrecy of ciphertext, rather than secret key as in the original case.
2. As in [37], our scheme achieves forward security under the assumption that the encrypted data is stored in the cloud and users do not store the encrypted/decrypted data locally.

– The procedure of ciphertext update only needs *public information*. Note that no previous identity-based encryption schemes in the literature can provide this feature;

– The additional computation and storage complexity, which are brought in by the forward secrecy, is all upper bounded by O(log(T)2), where T is the total number of time periods.

Outline. The remainder of this paper is structured as follows: In section 2, we introduce the preliminaries involved in our construction. Then we present the definitions of RSIBE in section 3, and provide the concrete construction in section 4, followed with the corresponding security analysis, performance discussions, and the implementation results of the scheme. Finally, we conclude in section 5.

II. PRELIMINARIES

In this section, we first briefly present the basic concepts on bilinear pairing and decisional ℓ-BDHE assumption. Then, an algorithm used to perform efficient revocation is introduced.

2.1 Bilinear pairing and complexity assumption

Definition 1 (**Bilinear pairing**). *Let* G1 *and* G2 *be two cyclic groups with prime order* q, *and* g *be a generator of* G1. *A bilinear pairing is a map* e : G1×G1 → G2 *with the following properties:*

• *Bilinearity:* e(ua, hb) = e(u, h)ab *for all* u, h ∈ G1, a, b ∈ Z∗q . • *Non-degeneracy:* e(g, g) 6= 1.

• *Computability: There exists an efficient algorithm to compute* e(u, h) *for any* u, h ∈ G1.

Definition 2 (**Decisional ℓ-BDHE Assumption**). *The decisional ℓ-BDHE problem is formalized as follows. Choose a group* G1 *with prime order* p *according to the security parameter* λ. *Select a generator* g *of* G1 *and* a, s R ←− Zp, *and let* fi = gai *Provide the vector* ***f*** = (g, gs, f1, ..., fℓ, fℓ+2, ..., f2ℓ) *and an element* D ∈ G2 *to a probabilistic polynomial-time (PPT) algorithm* C, *it outputs 0 to indicate that* D = e(gs, gaℓ+1), *and outputs 1 to indicate that* D *is a random element from* G2. *The advantage of* C *solving the decisional ℓ-BDHE problem in* G1 *is defined as follows:*

Advℓ−dBDHE C (λ) =___ Pr_C(***f*** ,D = e(gs, gaℓ+1)) = 0_ − Pr_C(***f*** ,D R ←− G2) = 0____

We say that the decisional ℓ-BDHE assumption holds in G1 *provided that no PPT algorithm can solve the decisional ℓ-BDHE problem with a non-negligible advantage.*

2.2 KUNodes algorithm

Our RS-IBE scheme uses the same binary tree structure introduced by Boldyreva, Goyal and Kumar [20] to achieve efficient revocation. To describe the revocation mechanism, we first present several notations. Denote by ε the root node of the binary tree BT , and Path(η) the set of nodes on the path from ε to the leaf node η (including ε and η). For a non-leaf node θ, we let all and θr stand for its left and right child, respectively. Given a time period t and revocations list RL, which is comprised of the tuples (ηi, ti) indicating that the node ηi was revoked at time period ti, the algorithm KUNodes(BT ,RL, t) outputs the smallest subset Y of nodes of BT such that Y contains an ancestor for each node that is not revoked before the time period t.

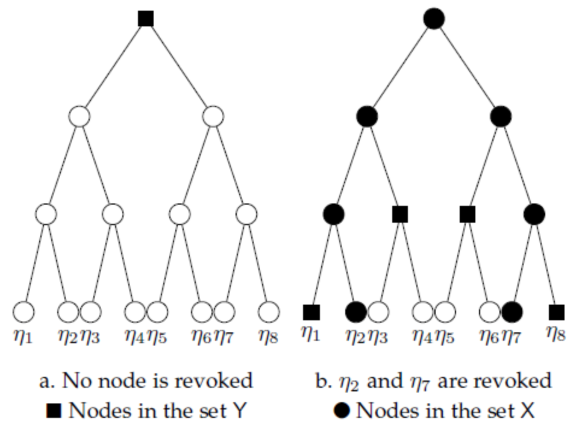


Fig. 2. An instance of the algorithm KUNodes

Informally, to identify the set Y, the algorithm first marks all the ancestors of revoked nodes as revoked, then outputs all the non-revoked children of revoked nodes. As an example,we present two instances of the algorithm KUNodes in Figure 2. The formal description is given below.

Algorithm 1 KUNodes(BT ,RL, t)
X,Y←− ∅
for all (ηi, ti) ∈ RL **do**
if ti ≤ t **then**
Add Path(ηi) to X
end if
end for
for all θ ∈ X **do**
if θl /∈ X **then**
Add θl to Y
end if
if θr /∈ X **then**
Add θr to Y
end if
end for
if Y = ∅ **then**
Add the root node ε to Y
end if
return Y

III. DEFINITION AND SECURITY MODEL OF RS-IBE

In this section, we first provide the formal definition of RSIBE, and then give the corresponding security model.

3.1 Syntax of RS-IBE

Definition 3 (**Revocable-Storage Identity-Based Encryption**).
A revocable-storage identity-based encryption scheme with message space M, *identity space* I *and total number of time*

periods T *is comprised of the following seven polynomial time algorithms:*

• **Setup**(1λ, T,N)*: The setup algorithm takes as input the security parameter* λ*, the time bound* T *and the maximum number of system users* N*, and it outputs the public parameter* PP *and the master secret key* MSK*,associated with the initial revocation list* RL = ∅ *and state* st*.*

• **PKGen**(PP,MSK, ID)*: The private key generation algorithm takes as input* PP, MSK *and an identity* ID ∈ I*, and it generates a private key* SKID *for* ID *and an updated state* st*.*

• **KeyUpdate**(PP,MSK,RL, t, st)*: The key update algorithm takes as input* PP,MSK*, the current revocation list* RL*, the key update time* t ≤ T *and the state* st*, it outputs the key update* KUt*.*

• **DKGen**(PP, SKID,KUt)*: The decryption key generation algorithm takes as input* PP, SKID *and* KUt*, and it generates a decryption key* DKID,t *for* ID *with time period* t *or a symbol* ⊥ *to illustrate that* ID *has been previously revoked.*

• **Encrypt**(PP, ID, t,M)*: The encryption algorithm takes as input* PP*, an identity* ID*, a time period* t ≤ T *, and a message* M ∈ M *to be encrypted, and outputs a ciphertext* CTID,t*.*

• **CTUpdate**(PP,CTID,t, t′)*: The ciphertext update algorithm takes as input* PP, CTID,t *and a new time period* t′ ≥ t*, and it outputs an updated ciphertext* CTID,t′ *.*

• **Decrypt**(PP,CTID,t,DKID,t′)*: The decryption algorithm takes as input* PP, CTID,t, DKID,t′ *, and it Recovers the encrypted message* M *or a distinguished symbol* ⊥ *indicating that* CTID,t *is an invalid ciphertext.*

• **Revoke**(PP, ID,RL, t, st)*: The revocation algorithm takes as input* PP*, an identity* ID ∈ I *to be revoked, the current revocation list* RL*, a state* st *and revocation time period* t ≤ T *, and it updates* RL *to a new one.*

Definition 4 (**Correctness of RS-IBE**)*. We say that a RS-IBE scheme is correct provided that, for any* (PP,MSK,RL, st) ← **Setup**(1λ, T,N), ID ∈ I, t ≤ T , M ∈ M*, all possible states* st *and a revocation lists* RL*, if* ID *is not revoked at time period* t*, then for* (SKID, sk)← **PKGen**(PP,MSK,ID),KUt← **Key Update**(PP,MSK,RL,t,sk),DKID,t←**DKGen**(PP,SKID,KU t)*, and* CTID,t′ ←**Encrypt**(PP, ID, t′ ,M)*, it is required that:*

• *If* t ≥ t′*, then* **Decrypt**(PP,CTID,t′ ,DKID,t) = M*. Otherwise,* **Decrypt** (PP,CTID,t′ ,DKID,t) =⊥ *with all but negligible probability.*

• *For* t ≥ t′*, it holds that* **CTUpdate**(PP,CTID,t′ , t) ≡ **Encrypt**(PP, ID, t,M) *holds.*

Here, ≡ *indicates equality in statistical distribution.*

3.2 Security model

Definition 5 (**IND-RID-CPA**)**.** *A RS-IBE scheme is INDRID- CPA secure provided that for any PPT adversary* A*, it has negligible advantage in the following security game between a challenger* C *and the adversary* A*:*

• *Setup.* C *performs* **Setup**(1λ, T,N) → (PP,MSK) *and sends* PP *to* A*;*

• *Phase 1.* A *makes the following queries in an adaptive way:*

a. OSK(ID)*:* C *performs* **PKGen**(PP,MSK, ID) → (SKID, st) *and returns* SKID *to* A*;*

b. OKU(t)*:* C *runs* **KeyUpdate**(PP,MSK,RL, t, st) → KUt *and returns* KUt*;*

c. ODK(ID, t)*:* C *runs* **PKGen**(PP,MSK, ID) → (SKID, st) *and* **DKGen**(PP, SKID,KUt) → DKID,t*, then forwards* DKID,t *to* A*;*

d. ORV (ID, t)*:* C *updates the current revocation list* RL *by running* **Revoke**(PP, ID,RL, t, st) → RL *and returns the updated* RL *to* A*;*

• *Challenge.* A *signals the query is over and sends* (M0,M1, ID∗, t∗, st) *to* C *subject to the restriction that* M0,M1 ∈ M *and* |M0| = |M1|*. Then,* C *chooses a random bit* β R ←− {0, 1} *and returns the challenged ciphertext* CTID∗,t∗ ← **Encrypt**(PP, ID∗, t∗,Mb) *to* A*;*

• *Phase 2.* A *begins another query phase as phase 1;* • *Guess.* A *outputs a bit* β′ *as a guess of* β*; The restrictions explicitly made on* A*'s queries in the game are as follows:*

1) OKU(t) *and* ORV (ID, t) *can only be queried in a sequential order in time. That is, the time* t *should be greater thanor equal to the time of all previous queries;*

2) ORV (ID, t) *cannot be queried if* OKU(t) *was queried;*

3) If OSK(ID∗) *was queried then* ORV (ID∗, t) *must be queried, where* t ≤ t∗*;*

4) ODK(ID, t) *cannot be queried before* OKU(t) *was queried;*

5) ODK(ID∗, t∗) *cannot be queried. The adversary* A*'s advantage in the above game is defined as*

AdvIND-RID-CPA RS-IBE,A (λ, T,N) =___ Pr_β′ = β_ − 12___

Remark 1. *In the security game above, we do not provide aquery oracle for the ciphertext update algorithm* **CTUpdate** *sincethe adversary can run it to a ciphertext just by using the public parameter* PP.

IV. RS-IBE RESISTANT TO DECRYPTION KEY EXPOSURE

In this section, we first present a concrete construction of RSIBE resistant to decryption key exposure, and then discuss its security and performance.

4.1 Construction

Our construction involves two binary trees BT and T to manage identity and time period, respectively. More precisely, for identity revocation, we follow Boldyreva et al.'s [20] strategy. That is, given an identity ID, we randomly store it in a leaf node η of BT , and generate the corresponding secret key SKID = {(θ, SKID,θ)}θ∈Path(η) as in previous RIBE schemes [20], [24]. If the user ID is not revoked at time period t, there exists a node θ ∈ Path(η) ∩ KUNodes(BT ,RL, t). Consequently, given the update key KUt = {(θ,KUt,θ)}θ∈KUNodes(BT ,RL,t), the user ID can obtain the decryption key for time period t by re-randomizing and combining (θ, SKID,θ) and (θ,KUt,θ). However, for a user that is revoked at time period t, there is no such node. As a result, the user cannot decrypt the ciphertext that is produced under its identity after the time period t (including t). Let T = 2ℓ be the total number of system time periods. For each 1 ≤ i ≤ T , the time period ti ∈ {0, 1}ℓ is associated with the i-th leaf node vti of T . Here, we arrange all leaf nodes of T in numerical order from left to right. Given a node v of T , let bv ∈ {0, 1}≤ℓ be the binary sequence corresponding to the path from the root node of T to v, where 0 and 1 indicate that the path passes through the left and right child of the parent node, respectively. Conversely, given a string b ∈ {0, 1}≤ℓ, let vb be the node that has a path b from the root node to it. Furthermore, denote by bv[j] and ti[j] the j-th bit of bv and ti respectively, and |bv| the length of bv. In addition, for each leaf node vt of the binary T , we define the following set: Tt = {v|Parent(v) ∈ Path(vt), v /∈ Path(vt)} ∪ {vt}, where Parent(v) denotes by the parent node of v. As presented in [34], such a set features the following property

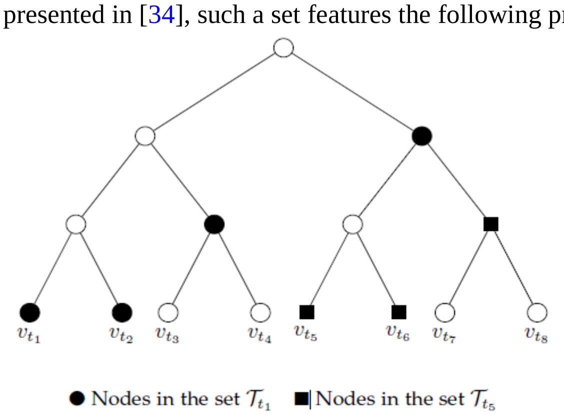


Fig. 3. An example of $\mathcal{T}$ with depth 3

Property 1. *Given two time periods* t *and* t′ *such that* t < t′, *for each node* v′ ∈ Tt′ , *there exists a node* v ∈ Tt *such that* bv *is a prefix of* bv′ . *In other words, there exists a string* b′ ∈ {0, 1}l *such that* bv′ = bv||b′, *where* l = |bv′ | − |bv|.

As presented subsequently, the above property enablesus to update the ciphertext from time periods t to t′. For explanatory purpose, we give a binary tree with depth 3, which means that the total number of time period is 8. As shown in Figure 3, the leftmost and rightmost leaf nodes vt1 and vt8 correspond to the strings 03 and 13 respectively. The sets Tt1 = {v1, v01, vt2 , vt1} and Tt5 = {v11, vt6 , vt5}. We present the concrete construction below.

• **Setup**(1λ, T,N): Given a security parameter λ, the total number of time periods T = 2ℓ, the maximum number of users N, this algorithm performs as follows:
1) Choose bilinear groups (G1,G2) with prime order p > 2λ and the corresponding bilinear map e : G1 × G1 →G2.
2) Select group elements g, g2R ←− G1 and an integerα R ←− Z∗ p, and set g1 = gα. Furthermore, pick two random vectors ***u*** = (u0, u1, ..., un) ∈ Gn+1 1 and ***h*** = (h0, h1, ..., hℓ) ∈ Gℓ+1 1 , and for each ID ∈ I = {0, 1}n and t ∈ {0, 1}ℓ, define the following two functions3:

Fu(ID) =u0 Y i=1 uID[i] i , Fh(t) = h0 ℓ Y j=1 ht[j] j . Here, ID[i] is the ith bit of ID.
3) Let the master secret key be MSK = gα2 , and initializethe state st with a binary tree BT of depth log(N) and the revocation list RL = ∅.
4) Publish the public parameter as PP = {G1,G2, e, g, g1, g2, ***u***, ***h***}.

• **PKGen**(PP,MSK, ID): For an identity ID ∈ I, this algorithm generates its secret key according to the following way:
1) Randomly choose and assign a leaf node η of BT to ID.
2) For each node θ ∈ Path(η),
a. Recall gθ,0 from BT if it has been defined previously4. Otherwise, pick gθ,0 R ←− G1, and store the pair (gθ,0, gθ,1 = g2/gθ,0) in the node θ.
b. Choose rθ,0 R ←− Z∗ p, and compute: SKID,θ = (SKθ,0, SKθ,1) = (gα θ,0Fu(ID)rθ,0 , grθ,0).
3) Update the state as st = BT , and then return the secret key SKID = {(θ, SKID,θ)}θ∈Path(η).

• **KeyUpdate**(PP,MSK,RL, t, sk): For each node θ ∈ KUNodes(BT ,RL, t), do the following.
1) Extract the pre-defined value gθ,1 from BT .
2) Choose rθ,1 R ←− Z∗ p and set KUt,θ = (KUθ,0,KUθ,1) = (gα θ,1 · Fh(t)rθ,1 , grθ,1).
3) Return KUt = {(θ,KUt,θ)}θ∈KUNodes(BT ,RL,t).

• **DKGen**(PP, SKID,KUt): If ID was revoked during time period t, return ⊥. Otherwise, there exists some node θ ∈ Path(η) ∩ KUNodes(BT ,RL, t). For this node θ, parse SKID,θ = (SKθ,0, SKθ,1) and KUt,θ =(KUθ,0,KUθ,1). Then, choose r0, r1 R ←− Z∗ p, compute and return DKID,t = (DKt,1,DKt,2,DKt,3) = (SKθ,0 · KUθ,0 · Fu(ID)r0 · Fh(t)r1 , SKθ,1 · gr0 ,KUθ,1 · gr1).

• **Encrypt**(PP, ID, t,M): To encrypt M ∈ G2 with IDat time period t, choose stR ←− Z∗ p, select sv R ←− Z∗p for each node v ∈ Tt. Particularly, let svt = st. Then, set the ciphertext as CTID,t = □ID, t,C0,C1,C2, {Cv}v∈Tt_, where

C0=M.e(g1,g2)stC1=g−st,C2=Fu(ID)st,Cv=(Cv,0,Cv,|bv|+1,

Cv,|bv|+2, ...,Cv,ℓ)= _□h0 |bv| Y j=1 hbv[j] j _sv , hsv |bv|+1, hsv |bv|+2..., hsv ℓ _.

3. We naturally require that n _ log(N)

4. As indicated in [20], a pseudo-random function can be used to recomputed g_,0 when necessary instead of having to store it in the node θ

• **CTUpdate**(PP,CTID,t, t′): Parse the ciphertext as CTID,t = □ID, t,C0,C1,C2, {Cv}v∈Tt_,where Cv = (Cv,0,Cv,|bv|+1,Cv,|bv|+2, ...,Cv,ℓ). To update the ciphertext from time period t to t′ ≥ t, do the following:

1) For each node v′ ∈ Tt′ , find a node v ∈ Tt such that bv ,is a prefix of bv′ .

2) Choose st′ R←−Z∗ p, select sv′ R ←− Z∗ p for each node v′ ∈ Tt. Particularly, let svt′ = st′

3) Compute C′0 = C0 · e(g1, g2)st′ , C′1 = C1 · g−st′ , C′2 = C2 · Fu(ID)st′ , Cv′ = (Cv′,0,Cv′,|bv′ |+1,Cv′,|bv′ |+2, ...,Cv′,ℓ) = _Cv,0 · |bv′ | Y j=|bv|+1 Cv,j · □h0 |bv′ | Y j=1 hbv′ [j] j _sv′ , Cv,|bv′ |+1 · hsv′ |bv′ |+1, ...,Cv,ℓ · hsv′ ℓ _.

4) Return CTID,t′ = □ID, t′,C′0 ,C′1 ,C′2 , {Cv′}v′∈Tt′ _.

• **Decrypt**(PP,CTID,t,DKID,t′): If t′ < t return ⊥. Otherwise, update the ciphertext CTID,t to get CTID,t′ , and parse CTID,t′ = □ID, t′,C′0,C′1,C′2, {Cv′}v′∈Tt′ _ and DKID,t′ = (DKt′,1,DKt′,2,DKt′,3). Then, return M = C′0 · e(C′1 ,DKt′,1) · e(C′2 ,DKt′,2) · e(Cvt′ ,0,DKt′,3).

• **Revoke**(PP, ID,RL, t, st): To revoke ID at time period t, update the revocation list by RL ← RL ∪ {(ID, t)} and return the updated RL.

CORRECTNESS. We verify the correctness of our scheme as follows: Firstly, given a node θ ∈ KUNodes(BT ,RL, t) ∩ Path(η), we have that

DKt,θ=(SKθ,0·KUθ,0·Fu(ID)r0·Fh(t)r1 , SKθ,1gr0,KUθ,1·gr1)=(gα2Fu(ID)rθ,0+r0Fh(t)rθ,1+r1 , grθ,0+r0 , grθ,1+r1).

Then, for a ciphertext CTID,t =□ID, t,C0,C1,C2, {Cv}v∈Tt_,we note that Cvt,0 = Fh(t)st . Thus, it holds that C0·e(C1,DKt,1)·e(C2,DKt,2)· e(Cvt,0,DKt,3)
= M · e(g1, g2)st · e(g−st , gα 2 · Fu(ID)rθ,0+r0 · Fh(t)rθ,1+r1) · e(Fu(ID)st , grθ,0+r0) · e(Fh(t)st , grθ,1+r1)
= M · e(g1, g2)st · e(g−st , gα 2)
= M.

Furthermore, note that the algorithm **CTUpdate** choosesfresh random exponents {sv′}v′∈Tt′ to update the original ciphertext CTID,t. Thus, CTID,t′ is not only a valid ciphertext under time period t′, but also is statistically indistinguishable from the output of **Encrypt**(PP, ID, t′,M).

4.2 Security analysis

Theorem 1. *If there exists a PPT adversary* A *breaking the INDRID- CPA security of the proposed RS-IBE scheme, then there exists an algorithm* C *solving the decisional* ℓ*-BDHE problem such that*

Advℓ−dBDHE C (λ) ≥ 1 32T q2(n + 1) · **Adv**IND-RID-CPA RS-IBE,A (λ, T,N),

where q *is the maximum number*5 *of secret key queries and decryption key queries, and* T = 2ℓ *is the total number of time periods*

Proof. Given a PPT adversary A breaking the IND-RIDCPA security of the proposed RS-IBE scheme, we will construct an algorithm C to solve the decisional ℓ-BDHE problem. More precisely, given a random instance of ℓ-BDHE problem in the form of a turple (G1, G2, e, p, ***f***, D) where ***f*** = (g, gs, f1, ..fℓ, fℓ+2, ..., f2ℓ) and fi = gai ∈ G1 for 1 ≤ i ≤ 2ℓ, the algorithm C can decide if D = e(fℓ+1, gs) by simulating the experiment according to the following steps.

Setup. The algorithm C randomly guesses a time period the adversary A will choose to be challenged. Denote it by t∗. To generate public parameter PP, the algorithm C proceeds as follows:

1) Choose α′ R ←− Z∗p and let g1 = f1gα′ and g2 = fℓ, which implicitly sets as α = (a + α′) and the master secret as MSK = gα 2 = g(a+α′), an unknown value to C.

2) Let m = 4q and select an integer ρ R ←− {0, 1, ..., n}. Furthermore, choose n+1 random integers x0, x1, ..., xn ∈ {0, 1, ...,m − 1} and another n + 1 random integers y0, y1, ..., yn ∈ Zp. For an identity ID ∈ {0, 1}n, define the following two functions:

J(ID) = (p − mρ) + x0 + n X i=1 ID[i]xi,

K(ID) = y0 + n X i=1 ID[i]yi.

3) Assign u0 = gx0−mρ 2 gy0 and ui = gxi 2 gyi for each 1 ≤ i ≤ n, and let ***u*** = (u0, u1, ..., un). Note that the above assignment implies that Fu(ID) = gJ(ID) 2 gK(ID).

4) Choose random integers γ0, γ1, ..., γℓ ∈ Zp and set h0 = gγ0 Qℓ j=1 ft∗[j] j and hj = gγjf−1 ℓ−j+1 for 1 ≤ j ≤ ℓ. Let ***h*** = (h0, h1, ..., hℓ).

5) Publish the public parameter as

PP = {G1,G2, e, g, g1, g2, ***u***, ***h***}.

Before starting to answer the adversary A's queries, the algorithm C flips a coin ctype R ←− { 0, 1} to guess that A belongs to which type of adversaries, which are distinguished as follows: -

• Type-1 adversaries (i.e., ctype = 0) choose to query OSK(ID∗) at some point, but ID∗ is revoked before the challenged time period t∗.

• Type-2 adversaries (i.e., ctype = 1) do not query OSK(ID∗) at any time.

Depending on A's type (i.e., the bit ctype), the algorithm C deals with A's behaviors by using different strategies separately.

• The case ctype = 0.

Denote by IDk the input of C's kth query on OSK(·). The algorithm C initially chooses k∗ R ←− {1, ..., q} as a guess that A's k∗th key query happens to be OSK(ID∗) (i.e., IDk∗ =

5. For simplicity, we assume that the maximum number of secret key queries is equal to the one of decryption key queries. =ID∗) with probability 1/q, and randomly selects a leaf node η∗ of BT to store ID∗.

Phase1. C responds to A's queries according to the following way:

• OKU(t): For each node θ ∈ KUNodes(BT ,RL, t), the algorithm C does the following:

1) Retrieve Yθ from θ if it was defined. Otherwise, it chooses Yθ R ←− G1 and stores it in θ.

2) If t = t∗, it must be that θ /∈ Paht(η∗), since ID∗ is revoked before the time period t∗. The algorithm C chooses rθ,1 R ←− Zp, and computes KUt,θ = (KUθ,0,KUθ,1) = (Y −1 θ Fh(t)rθ,1 , grθ,1).

Observe that KUt,θ is correctly formed as its assignment implicitly sets gα θ,1 = Y −1 θ .

3) If t 6= t∗, there exists an integer 1 ≤ l ≤ ℓ such that t∗[l] 6= t[l]. Without loss of generality, assume l is the smallest such integer. If θ /∈ Paht(η∗), the algorithm C does the same as in the case t = t∗. Otherwise, C performs as follows: Choose r′θ ,1
R ←− Zp and let rθ,1 = al t[l]−t∗[l] + r′θ,1, and set KUt,θ = (KUθ,0,KUθ,1) = (Y −1 θ gα 2 Fh(t)rθ,1 , grθ,1).
Observe that KUt, θ is well-defined in this case as its assignment implicitly sets gα θ,1 = Y −1
θ · gα 2 . As shown in Figure 4, KUt,θ is computable for C.

4) Return KUt = {(θ,KUt,θ)}θ∈KUNodes(BT ,RL,t) • OSK(IDk): Let IDk be the input of the k-th private key query.
– For k 6= k∗, if J(IDk) = 0, the algorithm C aborts. Otherwise, C randomly chooses a node η from BT and stores IDk in η, and then for each θ ∈ Path(η) does the following:

1) Recall Yθ if it was defined. Otherwise,
choose Yθ R ←− G1.

2) If θ ∈ Path(η∗), choose rθ,0 R ←− Zp, and Compute SKIDk,θ = (SKθ,0, SKθ,1) = (Yθ · Fu(IDk)rθ,0 , grθ,0).
Observe that KUt,θ is well-defined in this case as its assignment sets gα θ,0 = Yθ

3) If θ /∈ Path(η∗), choose r′θ,0 R ←− Zp and set rθ,0 = −a J(IDk) + r′θ,0, and let
SKIDk,θ = (SKθ,0, SKθ,1)= (Yθ · gα2 · Fu(IDk)rθ,0 , grθ,0). Observe that KUt,θ is well-defined in this case as its assignment implicitly sets gα θ,0 = Yθ·gα 2.In addition, we can see that SKIDk,θ is also computable for C:
SKθ,0 = Yθ · ga+α′2 · (gJ(IDk)m 2 gK(IDk))rθ,0
= Yθ · gα′ 2 · (gJ(IDk) 2 gK(IDk))r′ θ,0 · f −K(IDk) J(IDk)
1

KUθ,0 = Y −1θ · gα2 · Fh(t)rθ,1 = Y −1θ · gα′2 · fℓ+1·□gγ0ℓYj=1ft∗[j]ℓ−j+1ℓYj=1(gγj f−1
ℓ−j+1)t[j]_rθ,1
= Y −1θ · gα′· fℓ+1 · grθ,1(γ0+Pℓj=1 t[j]γj) ·□
l−1Yj=1ft∗[j]−t[j]ℓ−j+1 _rθ,1 · (ft∗[l]−t[l]
ℓ−l+1)rθ,1 · □ℓYj=l+1ft∗[j]−t[j]ℓ−j+1 _rθ,1
= Y −1θ · gα′2 · fℓ+1 · grθ,1(γ0+Pℓj=1 t[j]γj) · f−1ℓ+1 · fr′θ,1(t∗[l]−t[l])ℓ−l+1 □ℓYj=l+1ft∗[j]−t[j]
ℓ−j+1 _alt[l]−t∗[l] □ℓYj=l+1ft∗[j]−t[j]ℓ−j+1 _r′θ,1
= Y −1θ · gα′2 gr′θ,1(γ0+Pℓj=1 t[j]γj) · f1
t[l]−t∗[l] (γ0+Pℓj=1 t[j]γj)l · fr′θ,1(t∗[l]−t[l])ℓ−l+1
·
ℓYj=l+1(fℓ−j+l+1)t∗[j]−t[j]t[l]−t∗[l] ·ℓY
j=l+1fr′θ,1(t∗[j]−t[j])ℓ−j+1 .
KUθ,1 = grθ,1 = gr′θ,1 · (fl)1t[l]−t∗[l] .

Fig. 4. KUt,_ is computable for C.

and SKθ,1 = grθ,0 = gr′θ,0 · f−1J(IDk)1 .

– For k = k∗ (i.e., IDk = ID∗), if J(IDk∗) 6= 0, the algorithm C aborts. Otherwise, for each θ ∈ Path(η∗), the algorithm C proceeds as follows:

1) Retrieve Yθ from BT if it was defined. Otherwise, choose Yθ R ←− G1.

2) Select rθ,0 R ←− Zp, and compute

SKID∗,θ=(SKθ,0, SKθ,1)=(Yθ·Fu(ID∗)rθ,0,grθ,0).

Observe that KUt,θ is well-defined in this case as its assignment sets gα θ,0 = Yθ.

• ODK(ID, t): The algorithm C conducts a query OSK(ID), and then runs **DKGen**(PP, SKID,KUt). Now we show that by the definitions of OKU(·) and OSK(·), the output of ODK(·, ·) is also correctly formed. Given the input (ID, t), we distinguish them into the following three sub-cases:

– In case ID = ID∗, it must be that t 6= t∗. By the definitions, for each θ ∈ Path(η∗) ∩KUNodes(PP,RL, t), we have that

KUt,θ = (Y −1θ · gα2 · Fh(t)rθ,1 , grθ,1),

SKID∗,θ = (Yθ · Fu(ID∗)rθ,0 , grθ,0),
gαθ,1 = Y −1θ · gα2 , gαθ,0 = Yθ.

This implies that gαθ,0 · gαθ,1 = gα2 . Thus, The decryption key DKID∗,t is correctly formed.

– In case ID 6= ID∗ and t = t∗, it must be that
KU Nodes(PP,RL, t∗) ∩ Path(η∗) = ∅. Thus, for each node θ ∈ KU Nodes(PP,RL, t∗) ∩ Path(η), it holds that θ /∈ Path(η∗). Furthermore, we have that
KUt∗,θ = (Y −1θ · Fh(t)rθ,1 , grθ,1),
SKID∗,θ = (Yθ · gα2 · Fu(ID∗)rθ,0 , grθ,0),
gαθ,1 = Y −1θ , gαθ,0 = Yθ · gα2 .
This also implies that gαθ,0 · gαθ,1 = gα2 . Thus, the decryption key DKID,t∗ is also correctly formed.

– In the case ID 6= ID∗ and t 6= t∗, for each node θ ∈ KUNodes(PP,RL, t) ∩ Path(η), if θ ∈ Path(η∗) then the case is the same as the case ID = ID∗. Otherwise, the case is the same as the case ID 6= ID∗ and t = t∗. Therefore, the decryption key DKID,t is also correctly formed in this case.

Challenge. Given (M0,M1) and (ID∗, t′∗) on which the adversary A wishes to be challenged, if either ID∗ 6= IDk∗or t∗ 6= t′∗, the algorithm C aborts. Otherwise, it randomly flips a coin β ∈ {0, 1}, and generates the challenged ciphertextas follows:

1) Compute
C∗ 0 = MβD · e(gs, fℓ)α′ ,C∗ 1 = (gs)−1,C∗ 2 = (gs)K(ID∗);
2) For the node vt′∗ ∈ Tt′∗ , compute
C∗ vt′∗ = (C∗ vt′∗ ,0) = ((gs)γ0+Pℓ j=1 t′∗[j]γj);
3) For each node v ∈ Tt′∗ \ {vt′∗}, choose sv R ←− Zp, and compute
Cv = (C∗ v,0,C∗ v,|bv|+1,C∗ v,|bv|+2, ...,C∗ j=1 |bv|+1, hsv |bv|+2, ..., hsv ℓ _;

4) Return the challenged ciphertext
CTID∗,t′∗ = C∗0 ,C∗ 1 ,C∗ 2 , {C∗

Now, we show that if D = e(gs, fℓ+1) then CTID∗ ,t′∗ is correctly formed. To this end, recall that Fu(ID∗) = gK(ID∗)and Fh(t′∗) = gγ0+Pℓ j=1 t′∗ [j]γj , as well as observe that
D · e(gs, fℓ)α′ = e(gs, fℓ+1) · e(gα′, fℓ)s
= e(ga, fℓ)s · e(gα′ , fℓ)s
= e(ga+α′ , fℓ)s
= e(g1, g2)s.

In addition, if D is a random element in G2, then Mβ isperfectly hidden from the adversary A's view.

Phase 2. C proceeds the same as in Phase 1.

Guess. Eventually, the adversary A outputs a bit β′ as a guess of β. If β = β′, the algorithm C outputs 0, and 1 otherwise. • The case ctype = 1. Recall that the adversary A does not query OSK(ID∗) in this case at any time. However, A may make several decryption key queries on the challenged identity ID∗ . Denote the time of such queries by Count∗ . The algorithm C flips an unbiased coin to guess if Count∗ = 0 or not. If not, C randomly guesses that A's k∗ th decryption key query happens to be the first one on ID∗ .

⋆ The case Count∗ = 0.

Phase 1. The algorithm C answers A's queries as follows:

• OKU(t): For each node θ ∈ KUNodes(BT ,RL, t), the algorithm C does the following:

1) Recall Yθ from BT if it was defined. Otherwise, choose Yθ R ←− G1 and store it in the node θ.

2) Select rθ,1 R ←− Zp, and compute
KUt,θ = (KUθ,0,KUθ,1) = (Y −1 θ · Fh(t)rθ,1 , grθ,1).

3) Return KUt = {(θ,KUt,θ)}θ∈KUNodes(BT ,RL,t). • OSK (IDk): If J(IDk) 6= 0, the algorithm C aborts.
Otherwise, to generate the secret key for the identity IDk, the algorithm C proceeds as follows:

1) Randomly choose a leaf node η from BT and store ID in η;

2) For each node θ ∈ Path(η), retrieve Yθ from BT if it was defined. Otherwise, choose Yθ R ←− G1 and store it in the node θ.

3) Choose r′θ,0 R ←− Zp and let rθ,0 = − a J(IDk) +r′θ ,0 for each node θ ∈ Path(η), and set
SKIDk,θ =(SKθ,0, SKθ,1) = (Yθ · gα 2 · Fu(IDk)rθ,0 , grθ,0).
As shown in the case ctype = 0, the value SKID,θ is correctly formed, and is computable for the algorithm C.

5) Return SKIDk = {(θ, SKIDk,θ)}θ∈ Path(η).

• ODK(ID, t): The algorithm C queries OSK(ID) and subsequently runs **DKGen**(PP, SKID,KUt), and returns the corresponding output. Similar to the case ctype = 0, we can verify that the output of the decryption key query is well defined.

Challenge. Given (M0,M1) and (ID∗ , t′∗), the algorithm C checks if J(ID∗) = 0 and t∗ = t′∗ . If not, C aborts.Otherwise, C generates the challenged ciphertext according to the same way as in the case ctype = 0.

Phase 2. C acts the same as in Phase 1.
Guess. C acts the same as in the case ctype = 0.
⋆ The case Count∗ 6= 0.

Phase 1. The algorithm C uses the following strategy to answer A's queries.
• OKU(t) and OSK(ID): The algorithm proceeds the same as in the case Count∗ = 0.
• ODK(IDk, t): Let (IDk, t) be the input of A's decryption key query.
– If k < k∗ , the algorithm C proceeds the sameas in the case Count∗ = 0;
– If k = k∗ (i.e., IDk = ID∗), it must be t 6= t∗ . Thus, there exists the smallest index 1 ≤ l ≤ ℓ such that t[l] 6= t∗ [l]. The algorithm C does the following:

1) Choose r0, r′1 R ←− Zp and implicitly assign r1 = al t[l]−t∗ [j] + r′1;
2) Compute and return m
DKID∗ ,t = (DKt,1,DKt,2,DKt,3) = (gα 2 Fu(ID∗)r0Fh(t)r1 , gr0, gr1).
We can see that DKID∗ ,t is well-formed. Furthermore, as shown in the case ctype = 0, the value DKID∗ ,t is also computable for C.
– If k > k∗ , the way to response the query depends on IDk. More precisely, if IDk = ID∗ , the algorithm proceeds the same as in the case k = k∗ . Otherwise, C proceeds the same as in the case k < k∗ .

Challenge. C acts the same as in the case ctype = 0.
Phase 2. C proceeds the same as in Phase 1.
Guess. C performs the same as in the case ctype = 0.
Analysis of C. To enable the algorithm C to complete the experiment without aborting, the following conditions are required to be fulfilled:
1) E1: t∗ = t′∗ ;
2) In the case ctype = 0:m
• E2,1: For OSK(IDk∗), it must be IDk∗ = ID∗ ;
• E2,2: For 1 ≤ k 6= k∗ ≤ q, it must be J(IDk) 6= 0;
• E2,3: J(ID∗) = 0;
3) In the case (ctype = 1 ∧ Count∗ = 0):
• E3,1: For 1 ≤ k ≤ q, it must be J(IDk) 6= 0; • E3,2: J(ID∗) = 0;
4) In the case (ctype = 1 ∧ Count∗ 6= 0):
• E4,1: For 1 ≤ k ≤ q, it must be J(IDk) 6= 0;
• E4,2: For ODK(IDk∗ , t), it must be IDk∗ = ID∗ ;
• E4,3: J(ID∗) = 0.
Denote by E the event that C does not abort the experiment, we can that
E = □(E1 ∧ E2,1 ∧ E2,2 ∧ E2,3) ∨ (E1 ∧ E3,1 ∧ E3,2) ∨ (E1 ∧ E3,1 ∧ E3,2 ∧ E3,3)_.

Furthermore, by using Waters's [38] "artificial abort" technique, we obtain that
Pr[E1 ∧ (E2,1 ∧ E2,2 ∧ E2,3)] ≥ 1 2·1T·1 q·1 8q(n + 1)
Pr[E1 ∧ (E3,1 ∧ E3,2)] ≥ 1 4 · 1 T · 1 8q(n + 1) Pr[E1 ∧(E3,1 ∧ E3,2 ∧ E3,3)] ≥ 1 4 · T q· 1 8q(n + 1)

TABLE 1
Comparisons of communication and storage cost with previous works

Schemes	Private key size	Update key size	Ciphertext size
Libert and Vergnaud [22]	$O(\log N)\tau_{G_1}$	$O(r \log N/r)\tau_{G_1}$	$O(1)\tau_{G_1} + O(1)\tau_{G_2}$
Seo and Emura [24]	$O(\log N)\tau_{G_1}$	$O(r \log N/r)\tau_{G_1}$	$O(1)\tau_{G_1} + O(1)\tau_{G_2}$
Liang et al. [26]	$O(1)\tau_{G_1}$	$O(1)\tau_{G_1}$	$O(1)\tau_{G_1} + O(1)\tau_{G_2}$
Our scheme	$O(\log N)\tau_{G_1}$	$O(r \log N/r)\tau_{G_1}$ ‡	$O(\log(T)^2)\tau_{G_1} + O(1)\tau_{G_2}$

* τ_{G_1} and τ_{G_2} are the sizes of group elements in G_1 and G_2, respectively. N is the maximum number of system users. r is the number of revoked users. T is the total number of time periods.
‡ In fact, it is $\mathcal{O}(r \log(N/r))$ when $1 \le r \le N/2$, and $\mathcal{O}(N-r)$ when $N/2 < r \le N$.

TABLE 2
Comparisons of time complexity with previous works

Schemes	Encryption	Decryption	CTUpdate
Libert and Vergnaud [22]	$O(1)e + O(1)p$	$O(1)p$	0
Seo and Emura [24]	$O(1)e + O(1)p$	$O(1)p$	0
Liang et al. [26]	$O(1)e + O(1)p$	$O(1)p$	$(O(N))e + O(1)p$
Our scheme	$O(\log T)e + O(1)p$	$O(1)p$	$O(\log(T)^2)e + O(1)p$

* p and e indicate the cost of performing a bilinear pairing and exponentiation.

TABLE 3
Comparisons of security and functionality with previous works

Schemes	Model	Assumption	PKU	PCU	CA	DKE	FS	BS
Libert and Vergnaud [22]	Adaptive	DBDH	✓	×	✓	×	✓	×
Seo and Emura [24]	Adaptive	DBDH	✓	×	✓	✓	✓	×
Liang et al. [26]	Adaptive	DBDH	×	×	×	✓	✓	✓
Our scheme	Adaptive	ℓ-dBDHE	✓	✓	✓	✓	✓	✓

* Adaptive means an adaptive-secure model. DBDH is Decisional Bilinear Diffie-Hellman assumption, and ℓ-dBDHE is decisional ℓ-Bilinear Diffie-Hellman Exponent assumption. CA is collusion attack. DKE is decryption key exposure. FS and BS indicate forward and backward secrecy, respectively.

Pr[E] ≥ min nPr_E1 ∧ (E2,1 ∧ E2,2 ∧ E2,3)_, Pr_E1 ∧ (E3,1 ∧ E3,2)_, Pr_E1 ∧ (E3,1 ∧ E3,2 ∧ E3,3)_o ≥ 1 32T q2(

Now, recall that under the condition that C does not abort, if D = e(gs, gaℓ+1) then C perfectly simulates the experiment, and thus Pr_C(***f***,D = e(gs, gaℓ+1)) = 0_ = □**Adv**IND-RID-CPA RS-IBE,A (λ, T,N) + 1 2 _ · Pr[E].

However, if D is a random element from G2 then encrypted message M∗ β is perfectly hidden from the A's view, and thus
Pr[C(***f***,D R ←− G2) = 0] = 1 2 · Pr[E].
By combining the above equalities, we get that

Advℓ−dBDHE C (λ) =___ Pr_C(***f***,D = e(gs, gaℓ+1)) = 0_ − Pr_C(***f***,D R ←− G2) = 0____ m≥ 1 32T q2(n + 1) · **Adv**IND-RID-CPA RS-IBE,A (λ, T,N).

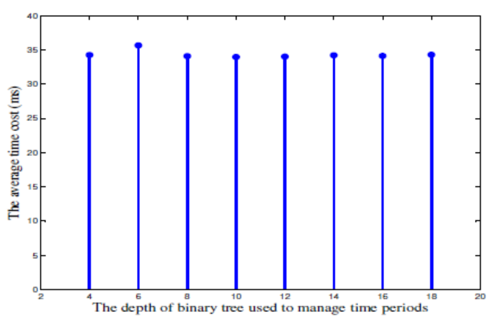
This completes the proof.

4.3 Performance discussions

In this section, we discuss the performance of the proposed RS-IBE scheme by comparing it with previous works in terms of communication and storage cost, time complexity and functionalities, which are summarized in Table 1, Table 2 and Table 3.

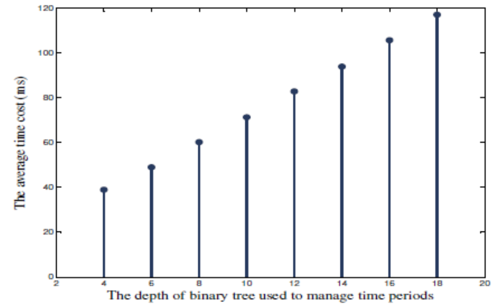
From Table 1 we can see that the sizes of private key and update key in schemes [22], [24] and our scheme are mall upper bounded by O(r logN/r), since these schemes all utilize binary data structure to achieve revocation. On the other hand, Liang et al.'s [26] scheme involves a broadcast encryption scheme to distribute update key such that their scheme has constant sizes of private key and update key. Furthermore, by delegating the generation of re-encryption key to the key authority, the ciphertext size of their scheme also achieves constant. However, to this end, the key authority has to maintain a data table for each user to store the user's secret key for all time periods, which brings O(T)τG1

JOURNAL OF LaTeX CLASS FILES, VOL. 14, NO. 8, AUGUST 2015



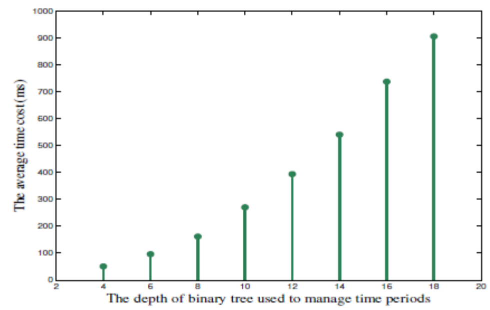
(a) PKGen

Fig. 5. The time costs of the algorithms **PKGen** and **KeyUpdate**.



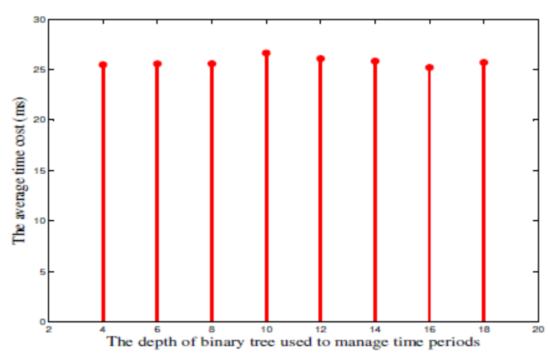
(a) Encrypt

Fig. 6. The time costs of the algorithms **Encrypt** and **DKGen**.

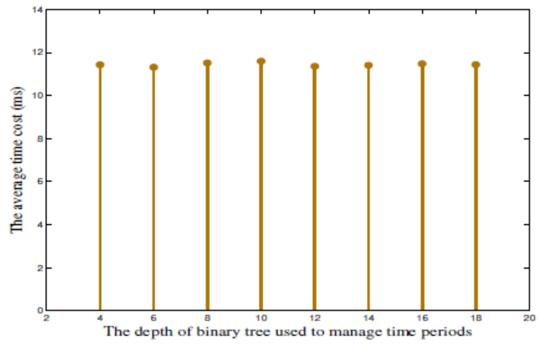


(a) CTUpdate

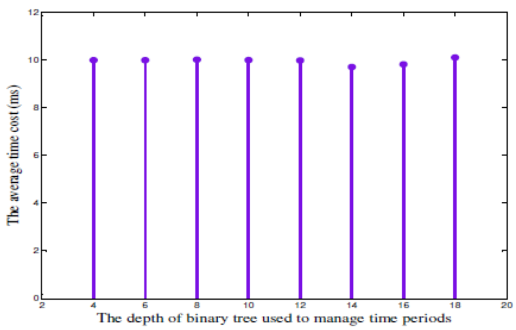
storage cost for the key authority. Conversely, the cipher text size of our scheme is just linear in log(T)2. In addition, we note that in all listed schemes, the private key generator needs to periodically produce an update key, it must be online if each time period is rather short, e.g., an hour. However, from the perspective of practical applications, the frequency of updating users' decryption keys should not be too small. A time period like a week, half a month or a moth is more desirable. As a consequence, the private key generator just needs to produce an update key for the next



(b) KeyUpdate



(b) DKGen



(b) Decrypt

period when the current time period is over. Thus the PKG does not need to be always online. Another limitation of these listed schemes is that the generated ciphertext has the size linear with the number of receivers. To overcome this issue, a natural manner is to construct a similar scheme in the setting of broadcast encryption. On the aspect of time complexity, as illustrated in

Table 2, the enumerated schemes all have constant time of /. Citation information: DOI decryption6. For two schemes supporting ciphertext update, the time complexity of ciphertext update in Liang et al.'scheme is linear in N since the key authority needs to produce a re-encryption key for each user to re-encrypt the ciphertext. However, the time complexity of ciphertext update in our scheme is linear in log(T)2.

As shown inTable 3, the four schemes are all proved secure in an adaptive-secure model, and can also provide backward secrecy since they all supports identity revocation. But the security of our scheme is built upon a relatively strong security assumption, decisional ℓ-DBHE assumption. The schemes [22], [24] and ours update user's secret keys in a public way, namely, the update key is available for all users. However, Liang et al.'s [26] scheme involves the method of broad encryption to update user's secret key such that only non-revoked users can obtain the update key. Consequently, their scheme cannot resist collusion attack of revoked users and non-revoked users. Compared with the schemes [22] and [24], Liang et al.'s [26] scheme and ours can both provide forward secrecy by additionally introducing the functionality of ciphertext update. But the procedure of ciphertext update in Liang et al.'s [26] scheme is performed in a private and interactive way, since it requires the key authority to periodically produce and provide reencryption keys for the cloud server to update ciphertext. However, in our schemes, the cloud server itself can update ,ciphertext by just using public parameter.

4.4 Implementation

To show the practical applicability of the proposed RSIBE scheme, we further implement it using codes from the Pairing-Based Cryptography library version 0.5.14 [39]. Specifically, we use the symmetric super singular curve y2 = x3 + x, where the base field size is 512-bit and the embedding degree is 2. The implementation is taken on a Linux-like system (Win7 + MinGW) with an Intel(R) Core(TM) i5 CPU (650@3.20GHz) and 4.00 GB RAM. In the implementation, we set the number of users to be N = 8 and the revoked uses to be R = 4 (the nodes η2, η3, η4, η7 in Figure 2 are revoked). In Figure 5, Figure 6 and Figure 7, we present the running time of the basic algorithms, i.e., **PKGen**, **KeyUpdate**, **DKGen**, **Encrypt**, **CTUpdate** and **Decrypt**, for different choice of the total number of time periods T ∈ {24, 26, 28, 210, 212, 214, 216, 218}. To generate the experimental results, we perform as the following procedure: generate the private key and encrypt a messageat the initial time period, then, periodically update the private key and the ciphertext, and decrypt the ciphertext. For a small number of time periods: T ∈ {24, 26, 28}, the running time of each algorithm is obtained by computing the average of running the above procedure 100 times. While, for a large number of time periods: T ∈ {210, 212, 214, 216, 218}, the running time for each algorithm is obtained by running the above procedure only once, and the running time for update algorithm is the mean of the first 512 time periods. We observe that, the time costs of the algorithms **PKGen**, 6. In our scheme, given the decryption DKID,t and ciphertext CTID,t′ , if t _ t′ then the cloud server would update CTID,t′ to CTID,t. Here, we just consider the decryption complexity for an individual **KeyUpdate**, **DKGen** and **Decrypt** are independent of the total number of time periods, and no more than 40 milliseconds.

On the other hand, it takes less than 1 second for the user to initially encrypting the message, which would beshare on the cloud. Although the time cost of the algorithm **CTUpdate** is apparently greater than other algorithms, it is run by a cloud server with powerful capability of computation. Thus, our RS-IBE scheme is feasible for practical applications.

V. CONCLUSIONS

Cloud computing brings great convenience for people. Particularly, it perfectly matches the increased need of sharing data over the Internet. In this paper, to build a cost-effective and secure data sharing system in cloud computing, we proposed a notion called RS-IBE, which supports identity revocation and ciphertext update simultaneously such that a revoked user is prevented from accessing previously shared data, as well as subsequently shared data. Furthermore, a concrete construction of RS-IBE is presented. The proposed RS-IBE scheme is proved adaptive-secure in the standard model, under the decisional ℓ-DBHE assumption. The comparison results demonstrate that our scheme has advantages in terms of efficiency and functionality, and thus is more feasible for practical applications.

ACKNOWLEDGEMENTS

We thank the anonymous reviewers for their helpful comments and suggestions.

REFERENCES

[1] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A break in the clouds: towards a cloud definition," ACM SIGCOMM Computer Communication Review, vol. 39, no. 1, pp. 50–55, 2008.
[2] iCloud. (2014) Apple storage service. [Online]. Available: https://www.icloud.com/
[3] Azure. (2014) Azure storage service. [Online]. Available: http://www.windowsazure.com/
[4] Amazon. (2014) Amazon simple storage service (amazon s3). [Online]. Available: http://aws.amazon.com/s3/
[5] K. Chard, K. Bubendorfer, S. Caton, and O. F. Rana, "Social cloud computing: A vision for socially motivated resource sharing," Services Computing, IEEE Transactions on, vol. 5, no. 4, pp. 551–563, 2012.
[6] C. Wang, S. S. Chow, Q. Wang, K. Ren, and W. Lou, "Privacypreserving public auditing for secure cloud storage," Computers, IEEE Transactions on, vol. 62, no. 2, pp. 362–375, 2013.
[7] G. Anthes, "Security in the cloud," Communications of the ACM, vol. 53, no. 11, pp. 16–18, 2010.
[8] K. Yang and X. Jia, "An efficient and secure dynamic auditing protocol for data storage in cloud computing," Parallel and Distributed Systems, IEEE Transactions on, vol. 24, no. 9, pp. 1717–1726, 2013.
[9] B. Wang, B. Li, and H. Li, "Public auditing for shared data with efficient user revocation in the cloud," in INFOCOM, 2013 Proceedings IEEE. IEEE, 2013, pp. 2904–2912.
[10] S. Ruj, M. Stojmenovic, and A. Nayak, "Decentralized access control with anonymous authentication of data stored in clouds," Parallel and Distributed Systems, IEEE Transactions on, vol. 25, no. 2, pp. 384–394, 2014.
[11] X. Huang, J. Liu, S. Tang, Y. Xiang, K. Liang, L. Xu, and J. Zhou, "Cost-effective authentic and anonymous data sharing with forward security," Computers, IEEE Transactionson,2014,doi: m10.1109/TC.2014.2315619.
[12] C.-K. Chu, S. S. Chow, W.-G. Tzeng, J. Zhou, and R. H. Deng, "Key-aggregate cryptosystem for scalable data sharing in cloud storage,"
[13] A. Shamir, "Identity-based cryptosystems and signature schemes," in Advances in cryptology. Springer, 1985, pp. 47–53.
[14] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," SIAM Journal on Computing, vol. 32, no. 3, pp. 586– 615, 2003.
[15] S. Micali, "Efficient certificate revocation," Tech. Rep., 1996.
[16] W. Aiello, S. Lodha, and R. Ostrovsky, "Fast digital identity revocation," in Advances in Cryptology–CRYPTO 1998. Springer, 1998, pp. 137–152.
[17] D. Naor, M. Naor, and J. Lotspiech, "Revocation and tracing schemes for stateless receivers," in Advances in Cryptology–CRYPTO 2001. Springer, 2001, pp. 41–62.
[18] C. Gentry, "Certificate-based encryption and the certificate revocation problem," in Advances in Cryptology–EUROCRYPT 2003. Springer, 2003, pp. 272–293.
[19] V. Goyal, "Certificate revocation using fine grained certificate space partitioning," in Financial Cryptography and Data Security. Springer, 2007, pp. 247–259.
[20] A. Boldyreva, V. Goyal, and V. Kumar, "Identity-based encryption with efficient revocation," in Proceedings of the 15th ACM conference on Computer and communications security. ACM, 2008, pp. 417–426.
[21] B. Libert and D. Vergnaud, "Adaptive-id secure revocable identitybased encryption," in Topics in Cryptology–CT-RSA 2009. Springer, 2009, pp. 1–15.
[22] ——, "Towards black-box accountable authority ibe with short ciphertexts and private keys," in Public Key Cryptography–PKC 2009. Springer, 2009, pp. 235–255.
[23] J. Chen, H. W. Lim, S. Ling, H. Wang, and K. Nguyen, "Revocable identity-based encryption from lattices," in Information Security and Privacy. Springer, 2012, pp. 390–403.
[24] J. H. Seo and K. Emura, "Revocable identity-based encryption revisited: Security model and construction," in Public-Key Cryptography– PKC 2013. Springer, 2013, pp. 216–234.
[25] "Efficient delegation of key generation and revocation functionalities in identity-based encryption," in Topics in Cryptology CT- RSA 2013. Springer, 2013, pp. 343–358.
[26] K. Liang, J. K. Liu, D. S. Wong, and W. Susilo, "An efficient cloudbased revocable identity-based proxy re-encryption scheme for public clouds data sharing," in Computer Security-ESORICS 2014. Springer, 2014, pp. 257–272.
[27] D.-H. Phan, D. Pointcheval, S. F. Shahandashti, and M. Strefler, "Adaptive cca broadcast encryption with constant-size secret keys and ciphertexts," International journal of information security, vol. 12, no. 4, pp. 251–265, 2013.
[28] R. Anderson, "Two remarks on public-key cryptology (invitedlecture)," 1997.
[29] M. Bellare and S. K. Miner, "A forward-secure digital signature scheme," in Advances in Cryptology–CRYPTO 1999. Springer, 1999, pp. 431–448.
[30] M. Abdalla and L. Reyzin, "A new forward-secure digital signature scheme," in Advances in Cryptology–ASIACRYPT 2000. Springer, 2000, pp. 116–129.
[31] A. Kozlov and L.Reyzin, "Forward-secure signatures with fast key update," in Security in communication Networks. Springer, 2003, pp. 241–256.
[32] X. Boyen, H.Shacham, E. Shen, and B. Waters, "Forward-secure signatures with untrusted update," in Proceedings of the 13th ACM conference on Computer and communications security. ACM, 2006, pp. 191–200.
[33] J. Yu, R. Hao, F. Kong, X. Cheng, J. Fan, and Y. Chen, "Forwardsecure identity-based signature: security notions and

construction," Information Sciences, vol. 181, no. 3, pp. 648–660, 2011.
[34] R. Canetti, S. Halevi, and J. Katz, "A forward-secure public-key encryption scheme," in Advances in Cryptology–Eurocrypt 2003. Springer, 2003, pp. 255–271.
[35] D. Yao, N. Fazio, Y. Dodis, and A. Lysyanskaya, "Id-based encryption for complex hierarchies with applications to forward security and broadcast encryption," in Proceedings of the 11th ACM conference on Computer and communications security. ACM, 2004, pp. 354–363.
[36] J. M. G. Nieto, M. Manulis, and D. Sun, "Forward-secure hierarchical predicate encryption," in Pairing-Based Cryptography–Pairing 2012. Sprnger, 2013, pp. 83–101.
[37] A. Sahai, H. Seyalioglu, and B. Waters, "Dynamiccredentials a ciphertext delegation for attribute-based encryption," in Advances in Cryptology–CRYPTO 2012. Springer, 2012, pp. 199–217.
[38] B. Waters, "Efficient identity-based encryption without random oracles," in Advances in Cryptology–EUROC Springer, 2005, pp. 114–127.
[39] B. Lynn. (2014) Pbc library: The pairing-based cryptography library.

Authors profile

Maadaala Chandra Sekhar completed B.Tech in Computer Science & Engineering from JNTUK and is pursuing Mtech in Qis college and Engineering and Technology in Depatrment of Computer Science and Engineering, Ongole.

Mrs. Keerthi Kethineni is currently working as an Assistant Professor in Department of Computer and Science and Engineering with the Qualification M.Tech.