8-2019

A secure IoT cloud storage system with fine-grained access control and decryption key exposure resistance

Shengmin XU
Singapore Management University, smxu@smu.edu.sg

Guomin YANG

Yi MU

Ximeng LIU
Singapore Management University, xmliu@smu.edu.sg

Published in Future Generation Computer Systems, 2019 August, 97, 284-294.
https://doi.org/10.1016/j.future.2019.02.051

A secure IoT cloud storage system with fine-grained access control and decryption key exposure resistance

Shengmin Xu [a], Guomin Yang [b,*], Yi Mu [c], Ximeng Liu [a,d]

[a] School of Information Systems, Singapore Management University, Singapore
[b] Institute of Cybersecurity and Cryptology, School of Computing and Information Technology, University of Wollongong, Australia
[c] Fujian Provincial Key Laboratory of Network Security and Cryptology, Fujian Normal University, Fuzhou, Fujian, China
[d] College of Mathematics and Computer Science, Fuzhou University, Fuzhou, Fujian, China

HIGHLIGHTS

- Our system provides fine-grained access control for data in the IoT cloud.
- Our system achieves revocable storage against revoked users.
- Our system can provide decryption key exposure resistance for non-revoked users.
- We correct a time encoding mechanism proposed by Xu et al. in IEEE TIFS'18.

Keywords:
IoT cloud
Attribute-based encryption
Revocation
Decryption key exposure

ABSTRACT

Internet of Things (IoT) cloud provides a practical and scalable solution to accommodate the data management in large-scale IoT systems by migrating the data storage and management tasks to cloud service providers (CSPs). However, there also exist many data security and privacy issues that must be well addressed in order to allow the wide adoption of the approach. To protect data confidentiality, attribute-based cryptosystems have been proposed to provide fine-grained access control over encrypted data in IoT cloud. Unfortunately, the existing attributed-based solutions are still insufficient in addressing some challenging security problems, especially when dealing with compromised or leaked user secret keys due to different reasons. In this paper, we present a practical attribute-based access control system for IoT cloud by introducing an efficient revocable attribute-based encryption scheme that permits the data owner to efficiently manage the credentials of data users. Our proposed system can efficiently deal with both secret key revocation for corrupted users and accidental decryption key exposure for honest users. We analyze the security of our scheme with formal proofs, and demonstrate the high performance of the proposed system via experiments.

1. Introduction

Internet of Things (IoT) cloud provides a scalable solution for storing and managing IoT data. It consists of a set of things (i.e., sensors) connected to the cloud through the Internet to provide (Internet of) Things as a Service (TaaS). By centralizing the large amount of IoT data in the cloud, the data owner can reduce the cost of data management and enjoy productivity enhancements and flexible data sharing with other data users. Along with the development of IoT cloud, an increasing number of IoT ecosystems have been integrated with the cloud, e.g., Google Cloud IoT, iDigi, Nimbits, ThingSpeak, etc.

With the development of IoT cloud, data sharing becomes one of the important and useful services. The data owners of IoT networks can provide the data sharing services to various data users (or service subscribers) via the cloud. The data users may acquire the data sharing service by purchasing and subscripting the services from the owners of IoT networks. However, data privacy and access control must be addressed for protecting the benefit of the data owners. Specifically, the could service provider (CSP) is an untrusted entity who may have the intention to obtain the data for financial gains or other incentives. Hence, some security mechanisms are needed to protect data privacy and enforce access control in IoT cloud. Attribute-based encryption is a useful cryptographic tool for such a need. It can provide fine-grained access control over encrypted data that will be shared to multiple users. However, to make ABE really practical in IoT cloud, some additional security issues must be well addressed.

* Corresponding author.
E-mail address: gyang@uow.edu.au (G. Yang).

Specifically, the following issues must be addressed securely and efficiently for a large-scale IoT cloud.

(1) User Revocation: Supporting user revocation is a fundamental requirement since the data service subscribers may change or misuse their subscribed services. To protect the interests of the IoT network owners, an efficient and secure revocation mechanism must be designed. The existing revocation mechanisms can be categorized into two methodologies: direct revocation and indirect revocation. In direct revocation setting, the IoT network owner publishes a revocation list and the data service subscribers are required to synchronize this revocation list all the time to keep it up-to-date, which makes it not suitable for large-scale systems. In the case of indirect revocation, the IoT network owner periodically broadcasts the key-updating material to all non-revoked users for updating their credentials via a public channel. The indirect revocation is efficient and practical, nevertheless, it still faces a problem that revoked data service subscribers still can access the old data they were authorized to access before being revoked.

(2) Revocable Storage: To address the above problem, Sahai et al. [1] introduced the concept of revocable storage, which allows a CSP to update the encrypted data without accessing any secret information and the updated ciphertexts are no longer decryptable by revoked users. However, designing an efficient ABE scheme with revocable storage is a non-trivial task. In the original scheme introduced in [1], two ABE instances are used in the construction, which makes the scheme rather inefficient.

(3) Decryption Key Exposure Resistance: The above security issues are related to handling corrupted/revoked users. When applying the indirect revocation approach for ABE, another security issue that should be considered is the exposure of ephemeral decryption keys belonging to honest users during a revocation epoch. In practice, decryption key exposure could happen due to key leakage attacks [2] or side-channel attacks [3]. We found that in many existing revocable ABE (RABE) systems [1,4–7], exposure of an ephemeral decryption key leads to the compromise of the user's long-term secret key (and hence the decryption keys corresponding to all the revocation epochs).

1.1. Our contributions

In this work, we introduce a novel data storage and sharing system for the IoT cloud as shown in Fig. 1 by presenting an efficient and scalable ciphertext-policy attribute-based encryption scheme with revocable storage and decryption key exposure resistance to support fine-grained access control for dynamic user groups. We also correct an error in a time encoding mechanism introduced in a previous revocable attribute-based encryption scheme [7].

Sahai et al. [1] first introduced a generic construction of RABE with revocable storage. However, their work has some shortcomings. First, the computational and communication costs of their scheme are high since two ABE instances are required to achieve the indirect revocation with revocable storage. Moreover, decryption key exposure attacks are not considered in Sahai et al.'s scheme. Based on the above observations, our idea is to replace the underlying scheme by a more efficient primitive and also allow the decryption key to be re-randomizable to remove the relationship among the secret key, the decryption key and the key-updating material to resist decryption key exposure attacks.

To demonstrate the feasibility of our idea, we present a concrete RABE scheme based on Rouselakis–Waters ABE [8] and Waters IBE [9] to provide key re-randomization to realize decryption key exposure resistance. Our concrete RABE also allows the untrusted CSP to update ciphertexts for realizing ciphertext revocation. We also present performance analysis to demonstrate that our proposed scheme is competitive in computational cost, functionality and security.

1.2. Related work

Boneh and Franklin [10] proposed the first practical IBE in the random oracle model and pointed out the importance of efficient revocation in IBE. They provided a revocation mechanism that requests each user to update the corresponding secret key periodically by representing the identity id as the identity appending current date $id \parallel t$. Unfortunately, this approach is impractical since a secure channel is required for the KGC to distribute the secret key to every non-revoked user in each revocation epoch. To improve the efficiency of user revocation in IBE, Boldyreva et al. [4] introduced the concept of indirect revocation, which is to divide the decryption key into a secret key and a public key-updating material. In each revocation epoch, the KGC broadcasts the key-updating material via the public channel and the non-revoked users use the key-updating material to update the corresponding secret key to be the decryption key in the new revocation epoch. Libert and Vergnaud [11] then improved the security by proposing a secure RIBE scheme in the adaptive model. Seo and Emura [12] and Watanabe et al. [13] further improved the security by considering the decryption key exposure attack and pointed out that the previous IBE schemes with indirect revocation are insecure in this strong model.

Given that traditional PKE and IBE only provide a coarse-level of access control, Sahai and Waters [14] introduced attribute-based encryption (ABE) to provide access control over encrypted data at a fine-grained level. To enrich expressiveness of access control policies, Goyal et al. [15] introduced key-policy ABE (KP-ABE) and Bethencourt et al. proposed ciphertext-policy ABE (CP-ABE) [16], respectively. To improve the performance, Attrapadung et al. [17] proposed the first constant-size ABE scheme and Lewko et al. [18] presented the first fully secure ABE. Rouselakis and Waters [8] then further improved the efficiency in large attribute universe setting by reducing the size of parameters. After that, a variety of ABE systems, including traceable ABE [19,20], outsourced ABE [21], anonymous ABE [22,23], have been proposed. However, user revocation was not considered in the above systems.

Similar to PKE and IBE, efficient user revocation is also essential in the ABE setting. Pirretti et al. [24] extended the revocation method in the IBE setting [10] to ABE by extending the attribute with a timestamp, i.e., by appending a timestamp to an attribute $att \parallel t$. Bethencourt et al. [16] then introduced a bit representation to provide integer comparisons in the timestamp. However, it still needs a secure channel between the KGC and all non-revoked data users in each revocation epoch. Since the concept of indirect revocation [4] was proposed, many following works for RABE [1,5–7] have been introduced by applying this method. Unfortunately, there does not exist a solution to prevent key exposure attack in ABE setting while it has been well studied in IBE setting [13,25]. Some other works related to revocation and revocable storage, such as revocable predicate encryption [26,27] and self-update encryption [28,29], also did not consider decryption key exposure attack.

1.3. Paper organization

In Section 2, we introduce some preliminaries related to our proposed system. In Section 3, we provide the system model including the system architecture and the threat model. In Section 4, we present our concrete construction of RABE and its application in IoT cloud. The security analysis is presented in Section 5, followed by the performance analysis in Section 6. Finally, we summarize this work in Section 7.

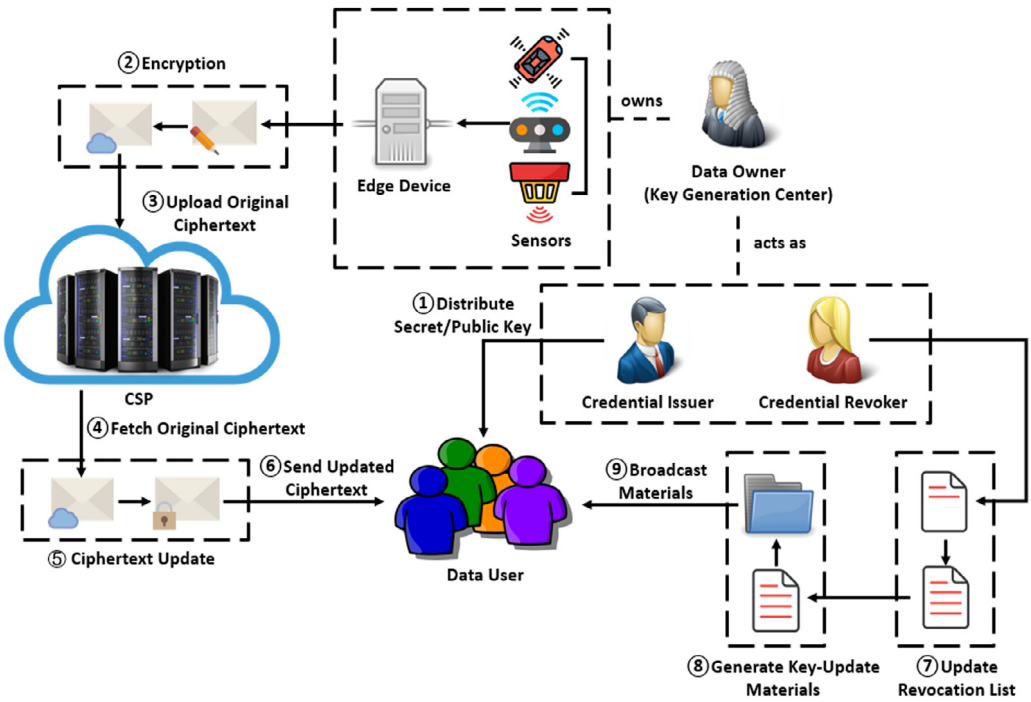


Fig. 1. System model in IoT cloud.

2. Preliminaries

2.1. Notations

Let $\mathbb{N}$ denote the set of all natural numbers, and for $n \in \mathbb{N}$, we define $[n] := \{1, \ldots, n\}$. If s is a string, then $s[i]$ is the ith bit of s. Let $\vec{u} := (u_1, u_2, \ldots, u_\ell)$ be a vector of dimension ℓ in $\mathbb{Z}_p^\ell$.

2.2. Bilinear map

Let $\mathbb{G}$ and $\mathbb{G}_T$ be two cyclic multiplicative groups of prime order p and g be a generator of $\mathbb{G}$. The map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is said to be an admissible bilinear pairing if following properties hold.

1. Bilinearity: for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$, $e(u^a, v^b) = e(u, v)^{ab}$.
2. Non-degeneration: $e(g, g) \neq 1$.
3. Computability: it is efficient to compute $e(u, v)$ for any $u, v \in \mathbb{G}$.

We say that $(\mathbb{G}, \mathbb{G}_T)$ are bilinear map groups if there exists a bilinear pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ as above.

2.3. Assumptions

We recall the definition of decisional Bilinear Diffie–Hellman (BDH) assumption [9]. It is defined via the following game: Initially, the challenger takes the security parameter λ as input to run the group generation algorithm. Next, it picks a random group generator $g \in \mathbb{G}$, and three random exponents $a, b, c \in \mathbb{Z}_p^3$. Then the challenger sends $g, g^a, g^b, g^c, e(g, g)^z$ to distinguish the value z is abc or a random value.

Definition 1 (*Decisional BDH Assumption*)**.** We say that the decisional BDH assumption holds if all polynomial probabilistic time attackers have at most a negligible advantage in λ in the above game.

Rouselakis and Waters [8] introduced a q-type assumption (refers to q-1 assumption) based on decisional parallel bilinear Diffie–Hellman exponent assumption. It is defined via the following game: Initially, the challenger takes the security parameter λ as input to run the group generation algorithm. Next, it picks a random group generator $g \in \mathbb{G}$, and $q + 2$ random exponents $a, s, b_1, b_2, \ldots, b_q \in \mathbb{Z}_p^{q+2}$. Then the challenger sends the attacker the group description and all of the following terms:

$$
\begin{array}{ll}
g, g^s & \\
g^{a^i}, g^{b_j}, g^{sb_j}, g^{a^i b_j}, g^{a^i/b_j^2} & \forall (i, j) \in [q, q] \\
g^{a^i b/b_{j'}^2} & \forall (i, j, j') \in [2q, q, q] \text{ with } j \neq j' \\
g^{a^i/b_j} & \forall (i, j) \in [2q, q] \text{ with } i \neq q + 1 \\
g^{sa^i b_j/b_{j'}}, g^{sa^i b_j/b_{j'}^2} & \forall (i, j, j') \in [q, q, q] \text{ with } j \neq j'
\end{array}
$$

The challenger also tosses a random coin $b \leftarrow \{0, 1\}$ and if $b = 0$, it sends the attacker the term $e(g, g)^{sa^{q+1}}$, otherwise, it gives a random term $R \in \mathbb{G}_T$. Finally, the attacker returns a guess $b' \in \{0, 1\}$.

Definition 2 (q-1 *Assumption*)**.** We say that the q-1 assumption holds if all polynomial probabilistic time attackers have at most a negligible advantage in λ in the above game.

2.4. RABE

We present the definition and the security model for the RABE schemes. Our security definition refines definitions in RIBE with decryption key exposure resistance [13] and RABE with revocable storage [1].

Definition 3 (*RABE*)**.** An RABE scheme with an attribute set Ω that supports policies $\mathcal{P}$ and the bounded system lifetime $\mathcal{T}$, an identifier space $\mathcal{I}$ and the message space $\mathcal{M}$ consists of nine algorithms given below.

$\mathsf{Init}(\lambda) \rightarrow pp$: The probabilistic initialization algorithm takes a security parameter $\lambda \in \mathbb{N}$ as input, and outputs a public parameter pp.

$\mathsf{Setup}(pp, \mathcal{N}, \mathcal{T}) \rightarrow (pk, msk, rl, st)$: The probabilistic setup algorithm takes the public parameter pp, the number of system users $\mathcal{N}$ and the bounded system lifetime $\mathcal{T}$ as input, and outputs a public key pk, a master secret key msk, a revocation list rl and a state st.

$\mathsf{KeyGen}(st, \mathcal{S}, id) \rightarrow (sk_{id}, st)$: The probabilistic key generation algorithm takes the state st, an attribute set $\mathcal{S} \subseteq \Omega$ and an identifier $id \in \mathcal{I}$ as input, and outputs the secret key sk_{id} and the state st.

$\mathsf{KeyUpdate}(msk, st, t, rl) \rightarrow ku_t$: The probabilistic key update algorithm takes the master secret key msk, the state st, the time $t \in \mathcal{T}$ and the revocation list rl as input, and outputs the key-updating material ku_t.

$\mathsf{DKGen}(pk, sk_{id}, ku_t) \rightarrow dk_{id,t}/\perp$: The probabilistic decryption key generation algorithm takes the public key pk, the secret key sk_{id} and key-updating material ku_t as input, and outputs the decryption key $dk_{id,t}$ or a failure symbol $\perp$.

$\mathsf{Enc}(pk, \mathbb{A}, t, m) \rightarrow c$: The probabilistic encryption algorithm takes the public key pk, an access structure $\mathbb{A} \in \mathcal{P}$, the time $t \in \mathcal{T}$ and a message $m \in \mathcal{M}$ as input, and outputs a ciphertext c.

$\mathsf{CTUpdate}(pk, c, t') \rightarrow c'/\perp$: The probabilistic ciphertext update algorithm takes the public key pk, a ciphertext c and a timestamp $t' \in \mathcal{T}$ as input, and outputs a ciphertext c' or a failure symbol $\perp$.

$\mathsf{Dec}(pk, dk_{id,t}, c) \rightarrow m/\perp$: The deterministic decryption algorithm takes the public key pk, the decryption key $dk_{id,t}$ and a ciphertext c as input, and outputs a message $m \in \mathcal{M}$ or a failure symbol $\perp$.

$\mathsf{Rev}(rl, id, t) \rightarrow rl$: The deterministic revocation algorithm takes the revocation list rl, an identifier $id \in \mathcal{I}$ and the timestamp $t \in \mathcal{T}$ as input, and outputs the revocation list rl.

We describe the security model called indistinguishable against chosen plaintext attack (IND-CPA) for RABE. Particularly, the difference between the following model and the traditional RABE model is that our model provides an additional oracle called decryption key generation oracle, which allows the adversary to query the short-term decryption key. Many RABE schemes [1,4–7] are insecure under this model.

Definition 4 (IND-CPA *in RABE*)**.** An RABE scheme with an attribute set Ω that supports policies $\mathcal{P}$ and the bounded system lifetime $\mathcal{T}$, an identifier space $\mathcal{I}$ and the message space $\mathcal{M}$ consists of nine algorithms given above. For an adversary $\mathcal{A}$, we define the following experiment:

$\mathbf{Exp}_{\mathcal{A}}^{\mathsf{IND-CPA}}(\lambda, \mathcal{N}, \mathcal{T})$

$pp \leftarrow \mathsf{Init}(\lambda)$;

$(pk, msk, rl, st) \leftarrow \mathsf{Setup}(pp, \mathcal{N}, \mathcal{T})$;

$(m_0, m_1, \mathbb{A}^*, t^*) \leftarrow \mathcal{A}^{\mathcal{O}}(pp, pk)$;

$b \leftarrow \{0, 1\}$;

$c^* \leftarrow \mathsf{Enc}(pk, \mathbb{A}^*, t^*, m_b)$;

$b' \leftarrow \mathcal{A}^{\mathcal{O}}(c^*)$;

If $b = b'$ return 1 else return 0.

$\mathcal{O}$ is a set of oracles, $\{\mathcal{O}_{\mathsf{KeyGen}}(\cdot,\cdot), \mathcal{O}_{\mathsf{KeyUpdate}}(\cdot), \mathcal{O}_{\mathsf{Rev}}(\cdot,\cdot), \mathcal{O}_{\mathsf{DKGen}}(\cdot,\cdot,\cdot)\}$ and the details are given below:

- $\mathcal{O}_{\mathsf{KeyGen}}(\cdot,\cdot)$ is the key generation oracle that allows $\mathcal{A}$ to query an attribute set $\mathcal{S} \subseteq \Omega$ and an identifier $id \in \mathcal{I}$, and it runs $\mathsf{KeyGen}(st, \mathcal{S}, id)$ to return the secret key sk_{id}.
- $\mathcal{O}_{\mathsf{KeyUpdate}}(\cdot)$ is the key update oracle that allows $\mathcal{A}$ to query the time $t \in \mathcal{T}$, and it runs $\mathsf{KeyUpdate}(msk, st, t, rl)$ to return the key update ku_t.
- $\mathcal{O}_{\mathsf{Rev}}(\cdot,\cdot)$ is the revocation oracle that allows $\mathcal{A}$ to query an identifier $id \in \mathcal{I}$ and the time $t \in \mathcal{T}$, and it runs $\Phi.\mathsf{Rev}(rl, id, t)$ to update the revocation list rl.
- $\mathcal{O}_{\mathsf{DKGen}}(\cdot,\cdot,\cdot)$ is the decryption key generation oracle that allows $\mathcal{A}$ to query the attribute set $\mathcal{S} \in \Omega$, the timestamp $t \in \mathcal{T}$ and an identifier $id \in \mathcal{I}$, and it runs $\mathsf{DKGen}(pk, sk_{id}, ku_t)$ to return the decryption key $dk_{id,t}$ if the secret key sk_{id} and the key update ku_t are available. Otherwise, it first runs $\mathsf{KeyGen}(st, \mathcal{S}, id)$ and $\Phi.KeyUpdate(msk, st, t, rl)$ to obtain the secret key sk_{id} and the key update ku_t.

$\mathcal{A}$ is allowed to issue above oracles with the following restrictions:

1. $\mathcal{O}_{\mathsf{KeyUpdate}}(\cdot)$ and $\mathcal{O}_{\mathsf{Rev}}(\cdot,\cdot)$ can be queried at the time t which is greater than or equal to that of all previous queries.
2. $\mathcal{O}_{\mathsf{Rev}}(\cdot,\cdot)$ cannot be queried at the time t if $\mathcal{O}_{\mathsf{KeyUpdate}}(\cdot)$ was queried at the time t.
3. If $\mathcal{O}_{\mathsf{KeyGen}}(\cdot,\cdot)$ was queried on an identifier $id \in \mathcal{I}$ with an attribute set $\mathcal{S} \subseteq \Omega$ s.t. $\mathbb{A}^*(\mathcal{S}) = 1$, then $\mathcal{O}_{\mathsf{Rev}}(\cdot,\cdot)$ must be queried on this identifier id at the time $t \leq t^*$.
4. $\mathcal{O}_{\mathsf{DKGen}}(\cdot,\cdot,\cdot)$ cannot be queried on any identifier $id \in \mathcal{I}$ with the attribute set $\mathcal{S}$ s.t. $\mathbb{A}^*(\mathcal{S}) = 1$ at the challenge time t^* or any identifier $id \in \mathcal{I}$ has been revoked.

An RABE scheme is said to be IND-CPA secure if for any probabilistic polynomial time adversary $\mathcal{A}$, the following advantage is negligible:

$$\mathbf{Adv}_{\mathcal{A}}^{\mathsf{IND-CPA}}(\lambda, \mathcal{N}, \mathcal{T}) = \left| \Pr[\mathbf{Exp}_{\mathcal{A}}^{IND-CPA}(\lambda, \mathcal{N}, \mathcal{T}) = 1] - 1/2 \right|.$$

An RABE scheme is said to be selective IND-CPA secure if $\mathcal{A}$ is requested to send the challenge access structure and time $(\mathbb{A}^*, t^*)$ at the beginning of adversarial game.

2.5. Linear secret sharing scheme

We recall the definition of linear secret sharing scheme (LSSS), as defined in [1]. A LSSS policy is of the type $(\mathbb{M}, \rho)$ where $\mathbb{M}$ is an $n \times l$ matrix over the base field $\mathbb{F}$ and ρ is a map from the set $[n]$ to the attribute universe Ω. A policy $(\mathbb{M}, \rho)$ satisfies an attribute set $S \subseteq \Omega$ if $1 = (1, 0, \ldots, 0) \in \mathbb{F}^l$ is contained in $\mathsf{Span}_{\mathbb{F}}(\mathbb{M}_i : \rho(i) \in S)$, where $\mathbb{M}_i$ is the ith row of $\mathbb{M}$.

2.6. Tree-based revocation approach

The tree-based data structure is widely used to reduce the computational time of generating and transmitting key updates from linear to logarithmic. To revoke a user, the subset-cover algorithm $\mathsf{KUNode}(st, rl, t)$ [30] can be used, where st is the state representing the tree-based data structure, rl is the revocation list recording identities of revoked users and t is the time representing the current revocation epoch. When a user joins the system, who will be assigned a random identifier $id \in \mathcal{I}$ and an undefined leaf node in st will be labeled this identifier id. The revocation method only requires the user id to store the keys in $\mathsf{Path}(id)$, where $\mathsf{Path}(id)$ denotes nodes in the path from the root node to the leaf node id.

2.7. Managing the time structure efficiently

Xu et al. [7] pointed out Waters IBE [9] can be used to shorten the ciphertext update. In their scheme, the encryption algorithm takes the set $\mathcal{V}$ recording the all 0-bit of the timestamp instead

of the user identity, and the ciphertext update algorithm is to update the timestamp t to the timestamp $t' \geqslant t$. Specifically, for updating the timestamp t to t', the ciphertext update algorithm allows the 0-bit in timestamp t to be 1-bit in timestamp t', while 1-bit in t cannot change to be 0-bit in t'. For example, assume the bounded system lifetime has 4-bit length, the original ciphertext is encrypted under the time $8 = 1000$ and the CSP will update this original ciphertext associated with the time $8 = 1000$ until the bounded system lifetime $15 = 1111$ without any secret information.

However, their proposed encoding method sometimes cannot work since it records the position of the first 1-bit, and the ciphertext update algorithm cannot work when the most significant bit is zero. For example, the original method encodes the timestamp $5 = 0101$ to $4 = 0100$ which cannot support the subsequent period from $8 = 1000$ to $11 = 1011$. We modify the ciphertext encoding algorithm in [7] as follows.

```
CTEncode(t, 𝒯)
    check ← false
    len ← log_2 𝒯
    for i = log_2 𝒯 to i = 1 do
        if t[i] = 1 and i = len and check = false then
            t̃[i] = 1, len = len − 1
        else
            check ← true, t̃ = 0
    return t̃
```

Our modified ciphertext encoding algorithm takes the timestamp t and the bounded system lifetime $\mathcal{T}$ as input. It outputs the timestamp $\tilde{t}$ in bit representation, which records as many 1-bit as possible from the most significant bit, e.g., it encodes the timestamp $5 = 0101$ to $0 = 0000$ and $13 = 1101$ to $12 = 1100$, which allows the ciphertext update algorithm to update the timestamp in the original ciphertext by changing 0-bit to 1-bit.

3. System model

In this section, we describe the system architecture and the threat model of the IoT cloud storage system.

3.1. System architecture

As shown in Fig. 1, the architecture of the IoT cloud mainly consists of three types of parties: the data owner, the data user and the CSP. The characteristic and function of each entity are described as follows.

Data owner. The data owner manages user credentials and provides data sharing services. Specifically, the data owner is the KGC who owns the IoT networks to collect the data in the surrounding area and the related edge devices to pre-process the data collected by wireless sensors. The KGC has two responsibilities and acts as the credential issuer and the credential revoker. The credential issuer defines the system parameters for the entire system. The credential revoker periodically generates the key-updating materials to manage the user revocation.

Data user. The data users are a group of users who obtain the data from the CSP. Each data user maintains a set of attributes and is authorized to decrypt the encrypted data if their attributes satisfy the encryption policy. In our system, data users will be categorized into two types: revoked users and non-revoked users. The revoked users cannot decrypt any ciphertext from the CSP. The non-revoked users decrypt the ciphertext by updating the corresponding long-term secret key to the valid decryption key.

CSP. The CSP has a vast amount of storage to accommodate the data from the data owners and computing power to update the original ciphertext to the updated ciphertext under current timestamp.

3.2. Threat model

We assume the data owner is a fully trusted entity. The data owner generates the system parameter, distributes the valid secret key for the data users and revokes compromised or expired users. The devices of the data owner (e.g., sensors and edge device) honestly collect and encrypt data, and outsource the data to the CSP.[1] In our system, the CSP is deemed as semi-trusted, which may try to learn sensitive information. The revoked data users have the intention to leak the corresponding long-term secret key. The non-revoked data users may accidentally exposure their short-term decryption key. To ensure the security of our IoT system against the semi-trusted CSP and untrusted data user, the system should be indistinguishable against chosen plaintext attack (IND-CPA). The concrete security model can be found in Section 2.4. In this model, the attackers play the roles of semi-trusted CSP and untrusted data user, and is also allowed to obtain the short-term decryption key to capture decryption key exposure attacks[2].

4. Proposed system

4.1. System overview

The workflow of the system architecture as shown in Fig. 1 is described as follows.

1. When the system is initialized, the data owner acts as the credential issuer to define the entire system parameters including the bounded system lifetime, the maximum number of system users and the period of each revocation epoch (e.g., 1 min/hour/day), and then distributes secret keys to the data users[3](See ①).
2. When the data is aggregated from the corresponding wireless sensors to the edge device, the edge device extracts the confidential and sensitive data and encrypts the data under a specific access structure and the current revocation epoch to derive the original ciphertext (See ②). After that, the original ciphertext is outsourced to the CSP (See ③).
3. When the data user intends to request the ciphertext from the CSP, the CSP first fetches the corresponding original ciphertext (See ④), then updates the original ciphertext to the ciphertext in current revocation epoch (See ⑤). After that, the CSP responses the updated ciphertext to the data user (See ⑥).
4. When the data user revocation (e.g., the key is expired/stolen/misused) is required, the data owner acts as the credential revoker to update the revocation list by appending the revoked data users to the revocation list (See ⑦). Based on the latest revocation list, the credential revoker derives the corresponding key-updating materials and broadcast these materials at the beginning of each revocation epoch (See ⑧ and ⑨).

[1] We assume the data transformation between the sensors and edge device is protected by a secure wireless protocol.

[2] All attackers are assumed to have polynomial time bounded computation ability such that they cannot solve the hardness problems in Section 2.3.

[3] The data owner has the sensors and the edge device. We assume these devices have the system parameter.

5. At the beginning of each revocation epoch, all data users receive the latest key-updating material from the data owner (See ⑨). The data user first discards the old short-term decryption key and retrieves the long-term secret key (from ①) combining the key-updating material (from ⑨) to derive the short-term decryption key in the latest revocation epoch. Note that the revoked user cannot derive the valid short-term decryption based on the key-updating material.

4.2. Concrete construction

In this subsection, we present a concrete RABE construction to realize the IoT cloud storage system described above.

The proposed RABE scheme follows the definition of RABE in Section 2.4. The construction is based on Rouselakis–Waters ABE [8] and Waters IBE [9], and they are provable security under the q-1 assumption and decisional BDH assumption (refer to Section 2.3). The high-level idea of our proposed RABE is to use the secret key of Rouselakis–Waters ABE to provide the fine-grained access control and the secret key of Waters IBE as the key-updating material to manage user revocation. To allow the distribution of key-updating material publicly, the secret key derived from Rouselakis–Waters ABE scheme embeds some noises that cannot be eliminated before knowing the key-updating material, and the key-updating material excludes the secret information that to be broadcasted publicly. To achieve key exposure resistance, we derive the decryption key that can be re-randomizable to remove the relationship among the secret key, the public key-updating material, and the decryption key. To realize the revocable storage, we modify the Waters IBE with the encoding method in Section 2.7. Our revocation mechanism is based on the tree structure given in Section 2.6 in which the cost is logarithmic to the number of users in the system. To prove our concrete scheme is secure, we will present the security reduction under the RABE security model without random oracle. The details are given in Section 5.

We should note that the following construction is an asymmetric-key cryptosystem. To improve the efficiency, we can use hybrid encryption method in which an extra symmetric cryptosystem (e.g., AES) is needed. When the data owner encrypts a message, a random symmetric key as the session key will be used to encrypt this message. The ciphertext includes two components: one is the message encrypted under the symmetric key, and the other one is the symmetric key encrypted by the RABE encryption algorithm.

4.2.1. System setup

The data owner as the credential issuer takes the security parameter λ, the number of system users $\mathcal{N}$ and the bounded system lifetime $\mathcal{T}$ as input, and outputs the public parameter pp, the public key pk, the master secret key msk, the revocation list rl and the state st by running the initialization algorithm and the setup algorithm. The data owner keeps secret the master secret key msk and the state st. The details of system setup are described as follows.

$\mathsf{Init}(\lambda) \rightarrow pp$: The initialization algorithm takes a security parameter $\lambda \in \mathbb{N}$ as input. It generates a bilinear group of order p according to the bilinear group parameter generator $(g, p, \mathbb{G}, \mathbb{G}_T) \leftarrow \mathcal{G}(\lambda)$, and outputs the public parameter $pp = (g, p, \mathbb{G}, \mathbb{G}_T)$.

$\mathsf{Setup}(pp, \mathcal{N}, \mathcal{T}) \rightarrow (pk, msk, rl, st)$: The setup algorithm takes the public parameter pp, the number of system users $\mathcal{N}$ and the bounded system lifetime $\mathcal{T}$ as input. It chooses a binary tree BT with at least $\mathcal{N}$ leaves, picks a random term $\alpha \in \mathbb{Z}_p$ and generates the system parameter as follows.

- For the attribute-related system parameter, it picks random terms $u, h, w, v \in \mathbb{G}^4$.
- For the time-related system parameter, let ℓ denote the size of $\mathcal{T}$, it chooses $u_0, \ldots, u_\ell \in \mathbb{G}^{\ell+1}$.

It returns the public key pk, the master secret key msk, the revocation list $rl = \emptyset$ and the state $st = \mathsf{BT}$.

$$pk = (e(g, g)^\alpha, u, h, w, v, u_0, \ldots, u_\ell), \quad msk = \alpha.$$

4.2.2. Key generation

The data owner as the credential issuer takes the state st, an attribute set $\mathcal{S}$ and the identifier of the data user id as input, and outputs the secret key sk_{id} and the updated state st by running the key generation algorithm. The data owner then sends the secret key sk_{id} to the data user and keeps secret the updated state st. Note that the identifier id is not the user's identity, it is an identifier initialized when the data user joins the system. The details of key generation are presented below.

$\mathsf{KeyGen}(st, \mathcal{S}, id) \rightarrow (sk_{id}, st)$: The key generation algorithm takes the state st, an attribute set $\mathcal{S} = (A_1, A_2, \ldots, A_k) \subseteq \Omega$ and an identifier $id \in \mathcal{I}$ as input. It randomly pick an unassigned leaf node from BT and stores id in this node. For each node $\theta \in \mathsf{Path}(id)$, it runs as follows:

- It retrieves α_θ from the node θ. If α_θ is not available, it randomly picks and stores $\alpha_\theta \in \mathbb{Z}_p$ in the node θ.
- It then picks $k+1$ random exponents $r, r_1, r_2, \ldots, r_k \in \mathbb{Z}_p^{k+1}$, and outputs the secret key $sk_{id,\theta}$:

$$sk_{id,\theta} = (g^{\alpha_\theta} w^r, g^r, \{g^{r_i}, (u^{A_i} h)^{r_i} v^{-r}\}_{i \in [k]}).$$

It returns the secret key $sk_{id} = \{sk_{id,\theta}\}_{\theta \in \mathsf{Path}(id)}$ and the updated state st.

4.2.3. Key update

The data owner as the credential revoker takes the revocation list rl, a serial of identifiers and the related revocation epochs (id_i, t_i), the master secret key msk, the state st and the current revocation epoch t as input, and outputs the updated revocation list rl and the key-updating material ku_t in current revocation epoch t by running the revocation algorithm and the key update algorithm as follows.

For each identifier and revocation epoch pair in the list (id_i, t_i), the data owner runs the following revocation algorithm.

$\mathsf{Rev}(rl, id_i, t_i) \rightarrow rl$: The revocation algorithm takes the revocation list rl, an identifier $id_i \in \mathcal{I}$ and the time $t_i \in \mathcal{T}$ as input. It returns the revocation list rl as:

$$rl \leftarrow rl \cup (id, t).$$

After the data owner obtains the latest revocation list rl, the data owner runs the key update algorithm.

$\mathsf{KeyUpdate}(msk, st, t, rl) \rightarrow ku_t$: The key update algorithm takes the master secret key msk, the state st, the time $t \in \mathcal{T}$ and the revocation list rl as input. It encodes the time t to the bit representation $\tilde{t}$. Let $\mathcal{V} \in [\ell]$ be the set of all indices i for $\tilde{t}[i] = 0$. For each node $\theta \in \mathsf{KUNodes}(st, rl, t)$, the key-updating material ku_t is constructed as:

- Retries α_θ.
- Randomly pick $s \in \mathbb{Z}_p$, and outputs the key update $ku_{t,\theta}$ as:

$$ku_{t,\theta} = \left(g^{\alpha - \alpha_\theta} \left(u_0 \prod_{i \in \mathcal{V}} u_i\right)^s, g^s\right).$$

It returns $ku_t = \{ku_{t,\theta}\}_{\theta \in \mathsf{KUNodes}(st,rl,t)}$.

4.2.4. Decryption key generation

The data user takes the public key pk, the related secret key sk_{id} and the key-updating material ku_t as input, and outputs the decryption key $dk_{id,t}$ if this data user is a non-revoked user in the revocation epoch t or the failure symbol $\perp$ if this data user is a revoked user before or in the revocation epoch t.

$\mathsf{DKGen}(pk, sk_{id}, ku_t) \rightarrow dk_{id,t}/\perp$: The decryption key generation algorithm takes the public key pk, the secret key sk_{id} and key update ku_t as input. Let I and J denote sets $\mathsf{Path}(id)$ and $\mathsf{KUNodes}(st, rl, t)$, respectively. It returns a failure symbol $\perp$ if $\mathsf{I} \cap \mathsf{J} = \emptyset$, otherwise, we have node $\theta \in \mathsf{I} \cap \mathsf{J}$. Parse the secret key $sk_{id,\theta}$ and the key update $ku_{t,\theta}$ are:

$$sk_{id,\theta} = (sk_1, sk_2, \{sk_{3,i}, sk_{4,i}\}_{i\in[k]}), \quad ku_{t,\theta} = (ku_1, ku_2).$$

It randomly picks $r', s' \in \mathbb{Z}_p^2$, and for $i \in [k]$, computes the decryption key $dk_{id,t}$:

$$d_1 = sk_1 \cdot ku_1 \cdot w^{r'} \left(u_0 \prod_{i\in\mathcal{V}} u_i\right)^{s'} = g^{\alpha} w^{r+r'} \left(u_0 \prod_{i\in\mathcal{V}} u_i\right)^{s+s'},$$
$$d_2 = sk_2 \cdot g^{r'} = g^{r+r'}, \quad d_{3,i} = sk_{3,i} = g^{r_i},$$
$$d_{4,i} = sk_{4,i} \cdot v^{-r} = (u^{A_i} h)^{r_i} v^{-(r+r')}, \quad d_5 = ku_2 \cdot g^{s'} = g^{s+s'}.$$

It returns $dk_{id,t} = (d_1, d_2, \{d_{3,i}, d_{4,i}\}_{i\in[k]}, d_5)$. Note that the decryption key $dk_{id,t}$ has been re-randomized and the relationships among the long-term secret key, public key-updating material, and short-term decryption key has been removed. Hence, the long-term secret key is secure even if the short-term decryption key is leaked.

4.2.5. Data encryption

The data owner (i.e., edge devices of the IoT network) takes the public key pk, a specific access structure $\mathbb{A}$, the current time t and a message m (collected by the sensors) as input, and outputs the ciphertext c by running the encryption algorithm as follows.

$\mathsf{Enc}(pk, \mathbb{A}, t, m) \rightarrow c$: The encryption algorithm takes the public key pk, an access structure $\mathbb{A} \in \mathcal{P}$, the time $t \in \mathcal{T}$ and a message $m \in \mathcal{M}$ as input, where the access structure $\mathbb{A}$ encodes an LSSS policy with the matrix $\mathbb{M} \in \mathbb{Z}_p^{n\times l}$ and a mapping function $\rho : [n] \rightarrow \Omega$. It then generates the ciphertext for attributes and time component, respectively.

- For the attribute-related ciphertext, it picks a vector $\vec{y} = (\xi, y_2, \ldots, y_n)^{\top} \in \mathbb{Z}_p^n$ and computes the vector $\vec{v} = (v_1, v_2, \ldots, v_n)^{\top} = \mathbb{M}\vec{y}$. It then chooses n random exponent $\phi_1, \phi_2, \ldots, \phi_n \in \mathbb{Z}_p^n$ and for $i \in [n]$, computes the attribute-related ciphertext:

  $$c_0 = m \cdot e(g, g)^{\alpha\xi}, \quad c_1 = g^{\xi}, \quad c_{2,i} = w^{v_i} v^{\phi_i},$$
  $$c_{3,i} = (u^{\rho(i)} h)^{-\phi_i}, \quad c_{4,i} = g^{\phi_i}.$$

- For the time-related ciphertext, it encodes the timestamp t to the bit representation $\tilde{t}$ and derives the time $\tilde{t} \leftarrow \mathsf{CTEncode}(\tilde{t}, \mathcal{T})$. Then, it randomly picks $s_2 \in \mathbb{Z}_p$ and let $\mathcal{V} \in [\ell]$ be the set of all j for which $\tilde{t}[j] = 0$, for $j \in \mathcal{V}$, computes the time-related ciphertext:

  $$c_5 = u_0^{\xi}, \quad c_{6,j} = u_j^{\xi}.$$

It returns $c = (c_0, c_1, \{c_{2,i}, c_{3,i}, c_{4,i}\}_{i\in[n]}, c_5, \{c_{6,j}\}_{j\in\mathcal{V}})$.

4.2.6. Ciphertext update

Upon receiving data request from a data user, the CSP takes the public key pk, the original ciphertext c and the current revocation epoch $t' \in \mathcal{T}$ as input, and outputs the updated ciphertext c' or a failure symbol $\perp$ (indicating the invalid ciphertext).

$\mathsf{CTUpdate}(pk, c, t') \rightarrow c'/\perp$: The ciphertext update algorithm takes the public key pk, the ciphertext c and the time $t' \in \mathcal{T}$ as input. It returns a failure symbol $\perp$ if the timestamp in ciphertext c is greater than the time t'. Otherwise, it encodes t' to the bit representation $\tilde{t'}$, let $\mathcal{V} \in [\ell]$ be the set of all indices i for $\tilde{t'}[i] = 0$, it computes the time-related ciphertext as:

$$c_t = c_5 \prod_{i\in\mathcal{V}} c_{6,i} = \left(u_0 \prod_{i\in\mathcal{V}} u_i\right)^{\xi}.$$

After that, it randomly chooses a vector $\vec{y'} = (\xi', y'_2, \ldots, y'_n)^{\top} \in \mathbb{Z}_p^n$ and computes the vector $\vec{v'} = (v'_1, v'_2, \ldots, v'_n) = \mathbb{M}\vec{y'}$. It then chooses n random exponent $\phi'_1, \phi'_2, \ldots, \phi'_n \in \mathbb{Z}_p^n$ and for $i \in [n]$, computes the ciphertext:

$$c'_0 = c_0 \cdot e(g, g)^{\alpha\xi'} = m \cdot e(g, g)^{\alpha(\xi+\xi')}, \quad c'_1 = c_1 \cdot g^{\xi'} = g^{\xi+\xi'},$$
$$c'_{2,i} = c_{2,i} \cdot w^{v'_i} v^{\phi'_i} = w^{v_i+v'_i} v^{\phi_i+\phi'_i},$$
$$c'_{3,i} = c_{3,i} \cdot (u^{\rho(i)} h)^{-\phi'_i} = (u^{\rho(i)} h)^{-(\phi_i+\phi'_i)},$$
$$c'_{4,i} = c_{4,i} \cdot g^{\phi'_i} = g^{\phi_i+\phi'_i},$$
$$c'_t = c_t \cdot \left(u_0 \prod_{i\in\mathcal{V}} u_i\right)^{\xi'} = \left(u_0 \prod_{i\in\mathcal{V}} u_i\right)^{\xi+\xi'}.$$

It returns $c' = (c'_0, c'_1, \{c'_{2,i}, c'_{3,i}, c'_{4,i}\}_{i\in[n]}, c'_t)$.

4.2.7. Ciphertext decryption

Upon receiving the updated ciphertext from the CSP, the data user takes the public key pk, the decryption key $dk_{id,t}$ and the received ciphertext c as input, and output the message m or a failure symbol $\perp$ by running the decryption algorithm as follows.

$\mathsf{Dec}(pk, dk_{id,t}, c) \rightarrow m/\perp$: The decryption algorithm takes the public key pk, the decryption key $dk_{id,t}$ and a ciphertext c as input. It returns a failure symbol $\perp$ if the attribute set $\mathcal{S}$ in the decryption key $dk_{id,t}$ does not match the access policy $\mathcal{P}$ in the ciphertext, e.g., $\mathbb{A}(\mathcal{S}) = 0$ or the time t in the ciphertext c does not match the time t in the decryption $dk_{id,t}$, otherwise, it computes the rows in $\mathbb{M}$ that provides a share to attributes in $\mathcal{S}$, i.e. $\mathcal{S} = \{i : \rho(i) \in \mathcal{S}\}$ and the constant $\{w_i \in \mathbb{Z}_p\}_{i\in\mathcal{S}}$ s.t. $\sum_{i\in\mathcal{S}} w_i \mathbb{M}_i = (1, 0, \ldots, 0)$. Thus, we have $\sum_{i\in\mathcal{S}} w_i u_i = \xi$ and computes the attribute-related component P_a as:

$$\begin{aligned} P_a &= \prod_{i\in\mathcal{S}} \left(e(c_{2,i}, d_2) \cdot e(c_{3,i}, d_{3,i}) \cdot e(c_{4,i}, d_{4,i})\right)^{w_i} \\ &= \prod_{i\in\mathcal{S}} (e(w^{v_i} v^{\phi_i}, g^{r+r'}) \\ &\quad \cdot e((u^{\rho(i)} h)^{-\phi_i}, g^{r_i}) \cdot e(g^{\phi_i}, (u^{A_i} h)^{r_i} v^{-(r+r')}))^{w_i} \\ &= \textstyle\prod_{i\in\mathcal{S}} e(w, g)^{w_i v_i (r+r')} \\ &= e(w, g)^{\xi(r+r')}. \end{aligned}$$

Then, the message hiding component P_m can be recovered as:

$$\begin{aligned} P_m &= \frac{e(c_1, d_1)}{P_a \cdot e(c_t, d_5)} \\ &= \frac{e(g^{\xi}, g^{\alpha} w^{r+r'} (u_0 \prod_{i\in\mathcal{V}} u_i)^{s+s'})}{e(w, g)^{\xi(r+r')} e((u_0 \prod_{i\in\mathcal{V}} u_i)^{\xi}, g^{s+s'})} \\ &= e(g, g)^{\alpha\xi}. \end{aligned}$$

Finally, it returns the message m as:

$$c_0/P_m = m \cdot e(g, g)^{\alpha\xi}/e(g, g)^{\alpha\xi} = m.$$

5. Security analysis

Theorem 1. *If the Rouselakis–Waters ABE [8] and Waters IBE [9] are secure, the proposed RABE scheme is secure.*

Rouselakis–Waters ABE [8] and Waters IBE [9] are provable security based on the q-1 assumption and decisional BDH assumption (refer to Section 2.3).

Our security proof is similar to the proof in Xu et al. [7] except the decryption key generation oracle generates the decryption key by combining the secret key from the key generation oracle and the key update from the key update oracle with some randomnesses picked by the RABE game simulator $\mathcal{B}$. The sketch of the proof is that we can construct an algorithm $\mathcal{B}$ which breaks Rouselakis–Waters ABE simulated by $\mathcal{C}_{\mathsf{ABE}}$ or Waters IBE simulated by $\mathcal{C}_{\mathsf{IBE}}$ by interacting with $\mathcal{A}$ which can break our proposed RABE scheme. $\mathcal{B}$ first tosses a random coin $rev \in \{0, 1\}$ at the beginning of the adversarial game to guess $\mathcal{A}$ represents a non-revoked user or not.

If $\mathcal{A}$ acts as a non-revoked user (i.e., $rev = 0$), $\mathcal{B}$ simulates the ABE component embedding the information $(\alpha - \alpha_\theta)$ by querying to $\mathcal{C}_{\mathsf{ABE}}$ and simulates the time-based component embedding the information α_θ by itself, where α_θ is the secret information in the state st in the form of full binary tree BT. Next, $\mathcal{B}$ forwards the challenge message from $\mathcal{A}$ to $\mathcal{C}_{\mathsf{ABE}}$ and computes the missing time-based component based on the returning message from $\mathcal{C}_{\mathsf{ABE}}$. After that, $\mathcal{B}$ returns the message from $\mathcal{C}_{\mathsf{ABE}}$ and the time-based component to $\mathcal{A}$. $\mathcal{A}$ then submits a bit b' as the guessing of challenge message. Finally, $\mathcal{B}$ forwards the bit b' to $\mathcal{C}_{\mathsf{ABE}}$ to break Rouselakis–Waters ABE.

If $\mathcal{A}$ is a revoked user (i.e., $rev = 1$), $\mathcal{B}$ constructs the binary tree BT for user revocation and picks a leaf node for the challenge user id^* with the attribute set $\mathcal{S}$ $s.t.$ $\mathbb{A}^*(\mathcal{S}) = 1$. For the nodes $\theta \in (\mathsf{Path}(id^*) \cap \mathsf{KUNodes}(st, rl, t))$, $\mathcal{B}$ returns the secret key embedding α by itself and the key update embedding $\alpha - \alpha_\theta$ by interacting with $\mathcal{C}_{\mathsf{IBE}}$. For the nodes $\theta \in (\mathsf{Path}(id^*) \setminus \mathsf{KUNodes}(st, rl, t))$, $\mathcal{B}$ returns the secret key embedding $\alpha - \alpha_\theta$ and the key update embedding α_θ by itself. Next, $\mathcal{B}$ forwards the challenge message from $\mathcal{A}$ to $\mathcal{C}_{\mathsf{IBE}}$ and computes the missing attribute-based component based on the returning message from $\mathcal{C}_{\mathsf{IBE}}$. After that, $\mathcal{B}$ returns the message from $\mathcal{C}_{\mathsf{IBE}}$ and the attribute-based component to $\mathcal{A}$. $\mathcal{A}$ then submits a bit b' as the guessing of challenge message. Finally, $\mathcal{B}$ forwards the bit b' to $\mathcal{C}_{\mathsf{IBE}}$ to break Waters IBE.

6. Performance analysis

As shown in Table 1, we give performance comparison between some RABE schemes [1,4,5,24] and ours. Let $\mathcal{S}$ be the attribute set, $\mathcal{T}$ be the system bounded lifetime, and $\mathcal{N}$ be the number of system users. It is straightforward to see from Table 1 that our proposed RABE is competitive in both computation and functionality. Although the scheme [1] is fully secure, it requires the composite order group and two instantiations of ABE schemes, which is inefficient and hence impractical.

In [24], the initial work of RABE was introduced with direct revocation, the expression of attribute is changed to the attribute att appending the current date t, e.g., $att \parallel t$, and hence the KGC has to keep a secure channel to each non-revoke user for transmitting the newly update secret key. The schemes [1,4,5] utilize indirect revocable method to improve the performance in space and time complexity but they suffer the decryption key exposure attack. Our RABE scheme has the advantage over previous solutions in that it is secure against key exposure attack and has revocable storage simultaneously. Also, our RABE has high performance in computational cost and storage requirement.

Next, we present the experimental analysis comparing the RABE schemes [1] with ciphertext-policy setting by applying CP-ABE [8] and our proposed RABE scheme. We have implemented the above schemes in Java using the jPBC library with the Type A elliptic curve and the symmetric pairing setting from "a.properties" in the jPBC library (an elliptic curve bilinear group with the 160-bit group order, 512-bit base field and embedded degree 2). Hence, p is a 160-bit prime number, and elements in $\mathbb{G}$ and $\mathbb{G}_T$ are 512-bit and 1024-bit, respectively. The software implementation was performed on a PC running 64-bit Windows 10 with Dual 2.8 GHz Intel(R) Core(TM) i7-7700HQ CPU and 16GB memory. We denote Sahai et al.'s RABE by SSW, and also limit the size of the attribute universe $|\Omega| = 30$ in the following experiments.

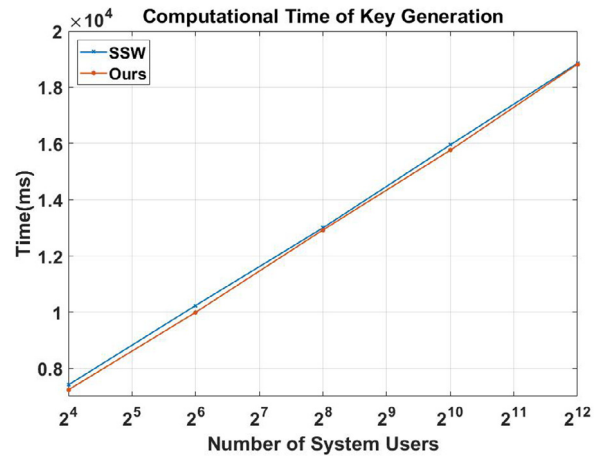


Fig. 2. Computational cost.

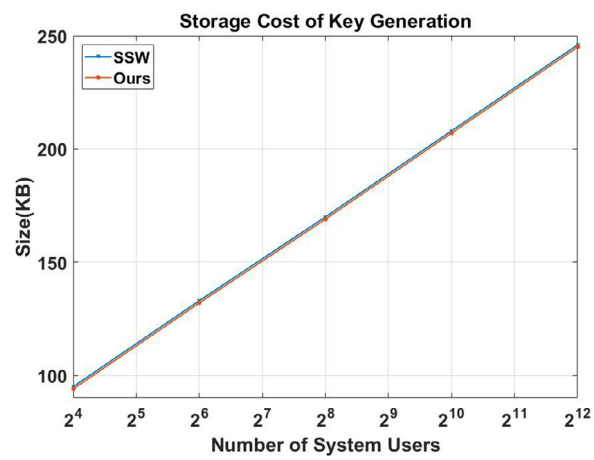


Fig. 3. Storage cost.

The average performance of the key generation algorithm is in Figs. 2 and 3. To evaluate the scalability of the scheme, we present the experimental performance of different user groups with up to $2^{12} = 4096$ users. The key generation algorithm of ours and [1] are based on the CP-ABE [8] scheme. Hence, the performance of the key generation algorithm in both the computational cost and the storage cost is very similar.

For the performance of the key update algorithm, we present the experimental results in a lightweight system and a large system with $2^{12} = 4096$ users including $2^7 = 128$ revoked users. In Figs. 4 and 5, we simulate the key update algorithm in lightweight system. In Figs. 6 and 7, we give the performance of the key update algorithm in large system. The x-axis is based on the exponent of 2 since the constructions of ours and SSW are based on the tree structure and bit representation. The details of the experimental performances are described as follows.

Our key update algorithm is based on the IBE scheme with constant computational and storage costs rather than the CP-ABE scheme with linear computational and storage costs. Hence, our

Table 1
Comparison between our proposed RABE scheme and some existing RABE schemes.

	PTMW [24]	*BGK* [4]	*AI* [5]	*SSW* [1]	*XYMD* [7]	*Our RABE*
Revocation mode	Direct	Indirect	Direct & Indirect	Indirect	Indirect	Indirect
ABE policy	KP-ABE	KP-ABE	KP-ABE	KP-ABE & CP-ABE	KP-ABE	CP-ABE
Revocation storage	×	×	×	√	√	√
Resist key exposure	√	×	×	×	×	√
Order of group	Prime	Prime	Prime	Composite	Prime	Prime
Security	Selective	Selective	Selective	Adaptive	Selective	Selective
Size of key update	$O(\mathcal{N}\cdot\mathcal{S})$	$O(\log\mathcal{N})$	$O(\log\mathcal{N})$	$O(\log\mathcal{N}\log\mathcal{T})$	$O(\log\mathcal{N})$	$O(\log\mathcal{N})$
Size of ciphertext	$O(\mathcal{S})$	$O(\mathcal{S})$	$O(\mathcal{S})$	$O(\log\mathcal{T}(\mathcal{S}+\log\mathcal{T}))$	$O(\mathcal{S}+\log\mathcal{T})$	$O(\mathcal{S}+\log\mathcal{T})$
Computation cost in key update	$O(\mathcal{N}\cdot\mathcal{S})$	$O(\log\mathcal{N})$	$O(\log\mathcal{N})$	$O(\log\mathcal{N}\log\mathcal{T})$	$O(\log\mathcal{N})$	$O(\log\mathcal{N})$
Computation cost in encryption	$O(\mathcal{S})$	$O(\mathcal{S})$	$O(\mathcal{S})$	$O(\mathcal{S}+\log\mathcal{T})$	$O(\mathcal{S})$	$O(\mathcal{S})$
Computation cost in decryption	$O(\mathcal{S})$	$O(\mathcal{S})$	$O(\mathcal{S})$	$O(\mathcal{S}+\log\mathcal{T})$	$O(\mathcal{S})$	$O(\mathcal{S})$

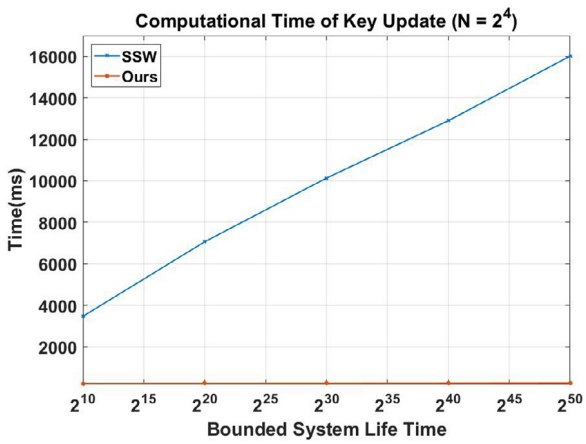


Fig. 4. Computational time.

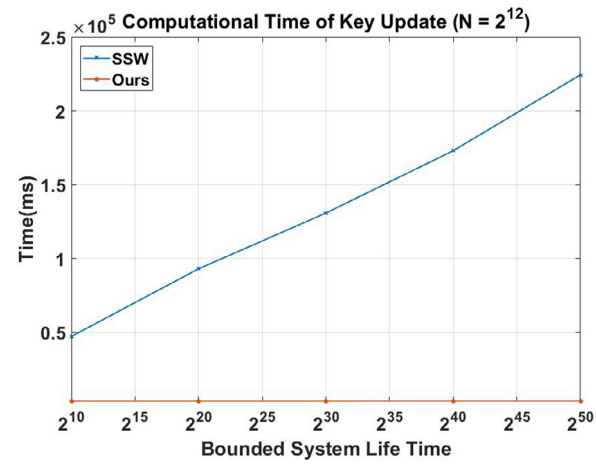


Fig. 6. Computational time.

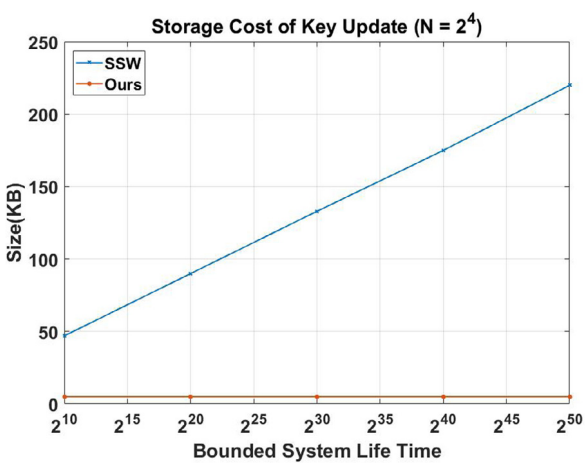


Fig. 5. Storage cost.

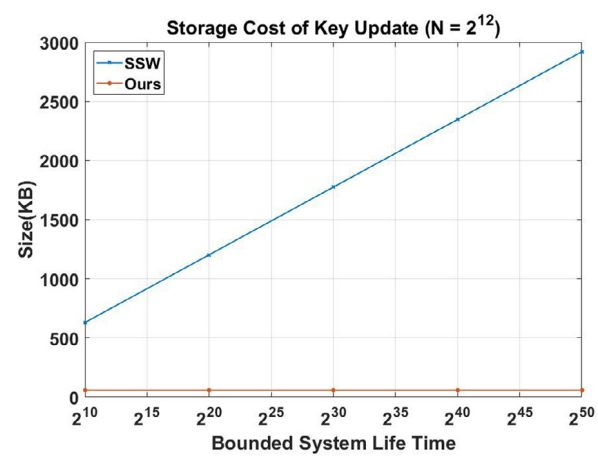


Fig. 7. Storage cost.

key update algorithm has better performance than Sahai et al.'s RABE [1].

The trends in Figs. 6 and 7 are similar to Figs. 4 and 5. In the large-scale 555 system, the costs are much higher than the lightweight system since the size of the key-updating material depends on the number of users. As we mentioned in Table 1, the size of the key-updating material is logarithmic to the number of users in the system. Hence, the trends of computational time and storage cost in the lightweight system and the large-scale system are similar, but the large-scale takes more time.

We also give the average performance of the encryption algorithm in Figs. 8 and 9. The performance of our scheme is better than Sahai et al.'s RABE [1] since our time component is based on the IBE scheme rather than the CP-ABE scheme. For the storage cost, the performance is very similar at 565 the beginning but along with the increasing of the bounded system lifetime, our scheme has better performance since Sahai et al.'s RABE needs to record the time component in the form of ABE ciphertext rather than IBE ciphertext in ours.

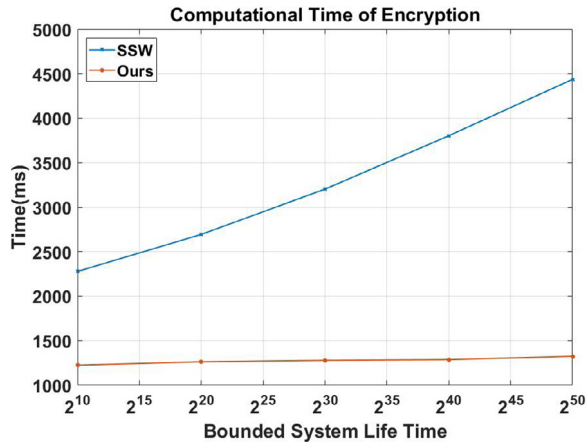


Fig. 8. Computational time.

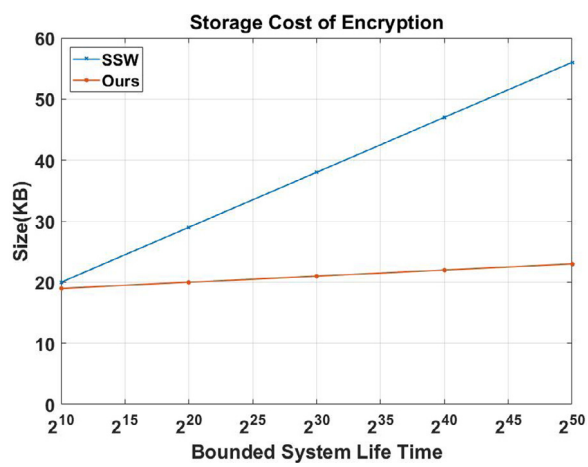


Fig. 9. Storage cost.

The above figures show that the experimental outcomes are similar to what we expected in Table 1. Therefore, the proposed construction has better performance than the RABE scheme [1], and also resists decryption key exposure attacks.

7. Conclusion

In this paper, we proposed a solution for securing the IoT cloud storage system by introducing a revocable attribute-based encryption with revocable storage and decryption key exposure resistance. We also presented a formal security model and the security proof of our proposed scheme. Both theoretical and experimental performance analysis demonstrated the high performance of the proposed scheme in comparison with the previous solutions.

Acknowledgments

This paper is supported by National Natural Science Foundation of China (61472308, 61702105, 61822202, 61872087, 61872089, 61872091).

References

[1] A. Sahai, H. Seyalioglu, B. Waters, Dynamic credentials and ciphertext delegation for attribute-based encryption, in: CRYPTO, 2012, pp. 199–217.

[2] X. Liu, R.H. Deng, K.R. Choo, J. Weng, An efficient privacy-preserving outsourced calculation toolkit with multiple keys, IEEE Trans. Inf. Forensics Secur. 11 (11) (2016) 2401–2414.

[3] J. Lee, M. Tehranipoor, C. Patel, J. Plusquellic, Securing designs against scan-based side-channel attacks, IEEE Trans. Dependable Sec. Comput. 4 (4) (2007) 325–336.

[4] A. Boldyreva, V. Goyal, V. Kumar, Identity-based encryption with efficient revocation, in: ACM CCS, 2008, pp. 417–426.

[5] N. Attrapadung, H. Imai, Attribute-based encryption supporting direct/indirect revocation modes, in: IMA, 2009, pp. 278–300.

[6] K. Nomura, M. Mohri, Y. Shiraishi, M. Morii, Attribute revocable attribute-based encryption for decentralized disruption-tolerant military networks, in: CANDAR, 2015, pp. 491–494.

[7] S. Xu, G. Yang, Y. Mu, R.H. Deng, Secure fine-grained access control and data sharing for dynamic groups in the cloud, IEEE Trans. Inf. Forensics Secur. 13 (8) (2018) 2101–2113.

[8] Y. Rouselakis, B. Waters, Practical constructions and new proof methods for large universe attribute-based encryption, in: ACM CCS, 2013, pp. 463–474.

[9] B. Waters, Efficient identity-based encryption without random oracles, in: EUROCRYPT, 2005, pp. 114–127.

[10] D. Boneh, M.K. Franklin, Identity-based encryption from the weil pairing, in: CRYPTO, 2001, pp. 213–229.

[11] B. Libert, D. Vergnaud, Adaptive-id secure revocable identity-based encryption, in: CT-RSA, 2009, pp. 1–15.

[12] J.H. Seo, K. Emura, Revocable identity-based cryptosystem revisited: security models and constructions, IEEE Trans. Inf. Forensics Secur. 9 (7) (2014) 1193–1205.

[13] Y. Watanabe, K. Emura, J.H. Seo, New revocable IBE in prime-order groups: adaptively secure, decryption key exposure resistant, and with short public parameters, in: CT-RSA, 2017, pp. 432–449.

[14] A. Sahai, B. Waters, Fuzzy identity-based encryption, in: EUROCRYPT, 2005, pp. 457–473.

[15] V. Goyal, O. Pandey, A. Sahai, B. Waters, Attribute-based encryption for fine-grained access control of encrypted data, in: ACM CCS, 2006, pp. 89–98.

[16] J. Bethencourt, A. Sahai, B. Waters, Ciphertext-policy attribute-based encryption, in: IEEE S & P, 2007, pp. 321–334.

[17] N. Attrapadung, J. Herranz, F. Laguillaumie, B. Libert, E. de Panafieu, C. Ràfols, Attribute-based encryption schemes with constant-size ciphertexts, Theoret. Comput. Sci. 422 (2012) 15–38.

[18] A.B. Lewko, T. Okamoto, A. Sahai, K. Takashima, B. Waters, Fully secure functional encryption: attribute-based encryption and (hierarchical) inner product encryption, in: EUROCRYPT, 2010, pp. 62–91.

[19] J. Ning, X. Dong, Z. Cao, L. Wei, X. Lin, White-box traceable ciphertext-policy attribute-based encryption supporting flexible attributes, IEEE Trans. Inf. Forensics SecuR. 10 (6) (2015) 1274–1288.

[20] J. Ning, Z. Cao, X. Dong, L. Wei, White-box traceable CP-ABE for cloud storage service: how to catch people leaking their access credentials effectively, IEEE Trans. Dependable Sec. Comput. 15 (5) (2018) 883–897.

[21] J. Ning, Z. Cao, X. Dong, K. Liang, H. Ma, L. Wei, Auditable σ-time outsourced attribute-based encryption for access control in cloud computing, IEEE Trans. Inf. Forensics Secur. 13 (1) (2018) 94–105.

[22] Y. Zhang, D. Zheng, R.H. Deng, Security and privacy in smart health: efficient policy-hiding attribute-based access control, IEEE Internet Things J. 5 (3) (2018) 2130–2145.

[23] Y. Zhang, X. Chen, J. Li, D.S. Wong, H. Li, I. You, Ensuring attribute privacy protection and fast decryption for outsourced data security in mobile cloud computing, Inform. Sci. 379 (2017) 42–61.

[24] M. Pirretti, P. Traynor, P.D. McDaniel, B. Waters, Secure attribute-based systems, in: ACM CCS, 2006, pp. 99–112.

[25] J.H. Seo, K. Emura, Revocable identity-based encryption revisited: security model and construction, in: PKC, 2013, pp. 216–234.

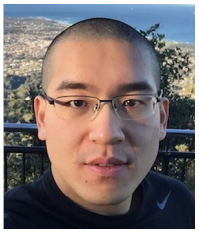
[26] J. Li, W. Yao, Y. Zhang, H. Qian, J. Han, Flexible and fine-grained attribute-based data storage in cloud computing, IEEE Trans. Serv.Comput. 10 (5) (2017) 785–796.

[27] J.M.G. Nieto, M. Manulis, D. Sun, Fully private revocable predicate encryption, in: ACISP, 2012, pp. 350–363.

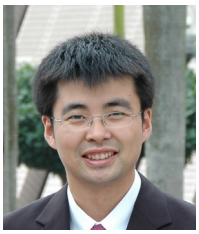
[28] K. Lee, S.G. Choi, D.H. Lee, J.H. Park, M. Yung, Self-updatable encryption: time constrained access control with hidden attributes and better efficiency, in: ASIACRYPT, 2013, pp. 235–254.

[29] K. Lee, Self-updatable encryption with short public parameters and its extensions, Des. Codes Cryptogr. 79 (1) (2016) 121–161.

[30] D. Naor, M. Naor, J. Lotspiech, Revocation and tracing schemes for stateless receivers, in: CRYPTO, 2001, pp. 41–62.



Shengmin Xu received the B.Sc. degree in the School of Computing and Information Technology, University of Wollongong, Australia, in 2014 and Ph.D. degree in Cryptography from University of Wollongong under the supervision of Dr. Guomin Yang and Prof. Yi Mu. His research interests include cryptography and information security.



Guomin Yang obtained his PhD in Computer Science from the City University of Hong Kong in 2009. He worked as a Research Scientist at the Temasek Laboratories of the National University of Singapore (NUS) from Sep 2009 to May 2012. He is currently a Senior Lecturer at the School of Computing and Information Technology, University of Wollongong, Australia. His research mainly focuses on Applied Cryptography and Network Security. He received the Australian Research Council Discovery Early Career Researcher Award in 2015.

Yi Mu received his PhD from Australian National University in 1994. He is currently a full professor at Fujian Normal University, China. Prior to his position with Fujian Normal University, he was a full professor with University of Wollongong, Australia. His research interest includes cryptography and information security. He is a senior member of IEEE.

Ximeng Liu received the B.Sc. degree in electronic engineering from Xidian University, Xi'an, China, in 2010 and Ph.D. degree in Cryptography from Xidian University, China, in 2015. Now, he is a full professor at College of Mathematics and Computer Science, Fuzhou University, China. Also, he is a research fellow at School of Information System, Singapore Management University, Singapore. He has published over 100 research articles include IEEE TIFS, IEEE TDSC, IEEE TC, IEEE TII, IEEE TSC and IEEE TCC. His research interests include cloud security, applied cryptography and big data security.