IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 10, 2018, accepted June 3, 2018, date of publication June 11, 2018, date of current version June 26, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2846037

Efficiently Revocable and Searchable Attribute-Based Encryption Scheme for Mobile Cloud Storage

SHANGPING WANG[1], DUO ZHANG[2], YALING ZHANG[3], AND LIHUA LIU[2]
[1]School of Science, Xi'an University of Technology, Xi'an 710055, China
[2]School of Automation and Information Engineering, Xi'an University of Technology, Xi'an 710055, China
[3]School of Computer Science and Engineering, Xi'an University of Technology, Xi'an 710055, China

Corresponding author: Duo Zhang (zhangduo029@163.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61572019, in part by the Key Project of the Natural Science Foundation of Shaanxi Province of China under Grant 2016JZ001, and in part by the Key Laboratory Research Project of the Education Bureau of Shaanxi Province of China under Grant 16JS078.

ABSTRACT Attribute-based encryption (ABE) is suitable for mobile cloud storage to protect data confidentiality and realize fine-grained data access control. It is essential for ABE schemes to achieve attribute revocation as users' attributes may be changed frequently. Keyword search over encrypted data also needs to be solved in the mobile cloud storage. In addition, computational efficiency is a consideration for the resource-constrained mobile device. Focusing on the above-mentioned problems, an efficiently revocable and searchable ABE (RSABE) scheme for the mobile cloud storage is proposed. In our scheme, the function of attribute revocation is efficiently achieved by delegating the update of secret key and ciphertext to the powerful cloud server. Keyword search is also supported, in which data owners and users can generate the keywords index and search trapdoor, respectively, without relying on always online trusted authority. Furthermore, an outsourced decryption technology is used to reduce the computational load of decryption on user side. Our RSABE scheme is proven to be semantically secure against selective ciphertext policy and chosen plaintext attack, and to be secure against chosen keyword attack in the random oracle model. Finally, performance evaluation demonstrates that our scheme is highly efficient.

INDEX TERMS Attribute-based encryption, attributes revocation, fine-grained access control, keywords search, mobile cloud storage.

I. INTRODUCTION

With the rapid development of mobile networks [1], portable mobile devices are widely favored by users, followed by the rapid growth of per capita holdings of mobile devices. This makes mobile terminals gradually become an important application platform. The contradiction between the growing user demand for data storage and data processing and relatively limited hardware resources of mobile terminals makes the mobile cloud storage services become a hot research area.

In such a mobile cloud environment, mobile users can outsource their data to the cloud sever through the mobile networks so as to store or share it with others. However, one of the main concerns is to preserve the confidentiality of the outsourced data. This is owing to the fact that the data stored in cloud is far from owners' physical control and it would be very vulnerable to unauthorized access by malicious users and even the semi-trusted cloud server itself. An effective method to address the issue is to encrypt the sensitive data before outsourcing. The integrity of data shared is also a security consideration, and it usually can be solved by remote data possession checking protocols [2]. We mainly focus on the confidentiality of the data.

To meet the secure requirements on outsourced data, Sahai and Waters [3] firstly presented a new method for data encryption: attribute-based encryption (ABE), which is envisioned to be a promising cryptographic primitive to protect the data security and realize fine-grained access control in one-to-many communications. ABE features such a mechanism that enables the operations like key issue, data encryption and decryption to be performed on the basis of attributes. In an ABE scheme, a user's keys is associated with a set of attributes, and every data owner can specify an access

control formula described by attributes when encrypting data. A user has access right to the encrypted data only when the attributes set embedded in his private key satisfy the access policy of the encrypted data. However, the computational cost of ABE schemes will dramatically increases with the number of attributes described in the access policy, which have been criticized a lot. The shortcoming becomes more serious in the resource-limited mobile environment.

With the deepening research on ABE, more and more attention has been paid to the revocation mechanism, for user's attributes can be changed dynamically in practice. However, attribute revocation is a challenging issue in ABE schemes as every attribute is conceivably shared by multi-users. It means that revocation of any attribute of a user would affect the other users who possess the revoked attribute. To overcome the difficulty, the traditional solution inevitably require data re-encryption and user secret key updates [4]–[7], which would bring heavy computation overhead when there are a large quantity of data stored on cloud and lots of users in the system.

Moreover, how to retrieve the encrypted data is another issue when applying ABE to practice. Searchable encryption (SE), put forward by Song *et al.* [8], is such a technology that allows a semi-trusted server to provide retrieval service on encrypted data with search trapdoor of keyword provided by a user, while the server knows nothing about the search keyword.

These above issues can be summarized as two points: firstly, most of the existing ABE schemes are not able to simultaneously achieve efficient attribute revocation and keyword search.While it is not easy to combine the exiting techniques in a straightforward way, since system parameter between a revocable ABE scheme and a searchable ABE scheme usually are different, and their focus are also different. Moreover, even if the direct combination is feasible, it will possibly result in redundancy of some parameters; secondly, the heavy computational cost on user side will affect the quality improvement of mobile cloud storage service. Thus, we argue that it is of great significance to design a revocable and searchable attribute-based encryption scheme, to deal with data access control in mobile cloud environment.

Our Contribution. In this paper, an efficiently revocable and searchable attribute-based encryption (RSABE) scheme for mobile cloud storage is proposed. We stretch Waters' CP-ABE scheme [9] from the aspects of functionality, efficiency and security. The contributions of our scheme can be concluded as follows.

1) We construct an RSABE scheme, which simultaneously supports efficient attribute revocation, attribute grant and keyword search in mobile cloud environment.
2) We provide an immediate revocation method with high efficiency. In our RSABE scheme, the attribute authority securely delegates the most update tasks to cloud server. During the whole revocation, the secret key component that user holds keeps unchanged, which brings great convenience for mobile users.
3) We present an efficient solution to search keywords on the encrypted data. The cloud server will return the search results only when the keywords and indexes are matched and the attributes set of user satisfies the access policy in ciphertext. Moreover, data owner and user can generate the keywords index and search trapdoor respectively without relying on trusted third party.
4) We greatly improve the computation efficiency of users. Specifically, each user has a delegated secret key for cloud server, so partial decryption operations can be outsourced to the cloud server. By this way, user decryption cost can be greatly reduced, which makes our scheme more suitable for mobile cloud environment.
5) Our RSABE scheme is proven to be selectively secure in the security model we defined in the subsequent section. The analysis shows that our solution has better performance than schemes [4], [6].

II. RELATED WORK

A. ATTRIBUTE-BASED ENCRYPTION

ABE, introduced by Sahai and Waters [3], is regarded as a creative extension of identity based encryption (IBE) [10] by viewing an identity as set of descriptive attributes. In [3], the data are encrypted under a set of attributes ω, and the private key was labeled with another set of attributes ω'. The ciphertext can only be decrypted only if there are at least d attributes overlap between the set of ω and ω', where d is the threshold parameter. However, there are a limitation in this scheme: lack of expressibility. Goyal *et al.* [11] further introduced the concept of ABE and proposed two different forms of ABE: ciphertext-policy ABE (CP-ABE) and key-policy ABE (KP-ABE). In the first type, the ciphertext is associated with an access policy and the secret key is integrated with an attributes set, a secret key can decrypt a ciphertext if and only if the attributes set satisfies the access policy, while the situation is inversed in KP-ABE schemes. Due to its implement about fine-grained access control and expressive access policy, ABE can be widely applied to many practical scenarios [11]–[13], such as electronic medical system, online social network and so on. Consider an honest-but-curious server model, CP-ABE fits well with the application in mobile cloud storage system.

B. ATTRIBUTE-BASED ENCRYPTION WITH ATTRIBUTE REVOCATION

The attribute revocation is an essential mechanism in ABE applications as the fact that user's attribute may change frequently in reality. The attribute revocation was first introduced by Pirretti *et al.* [14], which was realized by means of the timed rekeying mechanism. In their scheme, each attribute was designated to an expiration date, thus authority center are required to periodically reissue updated key. To revoke an attribute in the system, the authority center only need to stop issuing and updating the new version of the attribute.

Enhancing Pirretti's solution, Bethencourt *et al.* [12] associated the private key of user with a single expiration time. This scheme produces a lower load on authority than that in [14] since a user needn't update his secret key frequently. Boldyreva *et al.* [15] presented an efficient revocable KP-ABE scheme extended from their previous revocable IBE. Wang *et al.* [16], [17] gave two directly revocable construction of CP-ABE based on the bilinear maps and multi-linear maps respectively.

In recent years, several ABE schemes with immediate attribute revocation have been proposed. Yu *et al.* [20] and Ibraimi *et al.* [21] put forward CP-ABE schemes which enforced immediate attribute revocation by introducing a semi-trusted proxy server. This method transferred most of the workload of the authority to the proxy server, which greatly reduced the pressure of the authority. However, they have failed to achieve fine-grained access control. Moreover, the update workload of proxy server is very large when the number of users has a sharp increase. Hur and Noh [4] presented a fine-grained attribute revocation scheme using a binary key encrypting key (KEK) tree to implement rekeying. However the maintenance of the KEK tree cost highly. Xie *et al.* [22] made some improvements over Hur's scheme, reduced the size of the ciphertexts and secret keys and the computation amount in the key update phase. Li *et al.* [23] provided an efficient CP-ABE scheme with user revocation, and the lower computation cost make it can be applied to resource constrained devices. They also presented a CP-ABE scheme [24] with attribute revocation, which is secure against user collusion attack by existing users and revoked users. Several recent schemes on attribute revocation can be seen in [5], [7], [19], and [25]–[27].

C. KEYWORD SEARCH OVER ENCRYPTED DATA

SE is such a cryptographic primitive that enables users to search keywords over the encrypted data without leaking keywords information. Song *et al.* [8] firstly designed a practical SE scheme based on symmetric cryptography, which created an important precedent for implementing keyword search on the encrypted data. Later, Boneh *et al.* [28] originated the study of SE on the public-key cryptography. In the scheme, bilinear map is adopted and single keyword search is supported. After that, various SE schemes have been devised to promote the search efficiency, enhance the system security or enrich the searching functionalities. The schemes in [29]–[32] supported conjunctive keyword search, SE with fuzzy keyword search can be seen in [33]–[36]. Also, multi-user SE [37]–[39] has been developed since it is more practical than single-user setting in cloud environment. However, none of the above schemes support the fine-grained search authorization to distinguish users' search right. In recent years, attribute-based keyword search integrating the properties of ABE with SE have attracted much attention. Li *et al.* [40] proposed a searchable CP-ABE scheme with attributes revocation. In their scheme, the keyword search is supported and the access structure is partially hidden to protect privacy information in ciphertexts. More elated achievements can be seen in [18] and [41]–[46].

D. OUTSOURCED DECRYPTION

Computational efficiency is another point need to be taken into account in the current ABE schemes. Outsourced decryption technology can largely reduce the computational load of user side. Green *et al.* [47] firstly presented an efficient ABE scheme supporting outsourced decryption, where majority of decryption tasks are performed by the could server with a transformation key of users. Zhou and Huang [48] proposed an efficient data storage scheme focusing on mobile devices, where parts of the encryption and decryption operations are securely outsourced to the service providers without critical secrets leakage. Lai *et al.* [49] put forward an ABE scheme considering verifiability of outsourced decryption. The verifiability assures that users could examine whether the outsourced decryption is completed correctly. Lately, Li *et al.* [50] firstly designed a fine-grained access control scheme with outsourced key generation and decryption, where two secure cloud service providers are adopted to execute key-issuing and decryption respectively. On the basis of Li *et al.*, Lv *et al.* [19] applied outsourced key generation and decryption to the mobile environment, and efficiently achieved user revocation. Li *et al.* [51] presented an outsourced ABE scheme with checkability of the outsourced computation results, where two service providers are also used to delegate key generation and decryption operations. They also introduced the outsourced technology into IBE and achieved outsourced user revocation in scheme [52]. Later, they constructed an ABE scheme [53] overcoming several drawbacks of outsourcing computation when applying it to resource-limited mobile users. Li *et al.* [54] presented an ABE scheme with full verifiability for outsourced decryption, where the issue of how to guarantee the outsourced decryption correctness for unauthorized users is considered. There are several applications of outsourced decryption in [45] and [55]–[57].

Here, we compare the functions of some existing schemes with our scheme as shown in Table 1. We can see that the functionalities of our scheme are more perfect, which makes our scheme more suitable for mobile cloud storage system.

III. PRELIMINARIES

A. BILINEAR MAP

Definition 1 (Bilinear Map [9]): Choose two multiplicative cyclic group $\mathbb{G}$ and $\mathbb{G}_T$ of prime order p. Let g be a generator of $\mathbb{G}$. A function $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is called a bilinear map, if it satisfies:

1) *Bilinearity: For all $u, v \in \mathbb{G}$, and all $a, b \in \mathbb{Z}_p$, $e(u^a, v^b) = (u, v)^{ab}$.*
2) *Non-degeneracy: $e(g, g) \neq 1$.*
3) *Computability: For all $u, v \in \mathbb{G}$, the pairing $e(u, v)$ can be efficiently computed.*

TABLE 1. Comparisons of functionality.

Schemes	Revocation	Attribute grant	Keyword search	Delegate decryption	Access control
Hur's [4]	attribute level user revocation	×	×	×	access tree
Yu's [13]	system level user revocation	×	×	×	access tree
Sun's [18]	system level user revocation	×	√	×	AND gate
Lv's [19]	attribute level user revocation	×	×	√	access tree
Ours	attribute level user revocation	√	√	√	LSSS

B. DECISIONAL PARALLEL BILINEAR DIFFIE-HELLMAN EXPONENT ASSUMPTION

The decisional q-parallel bilinear Diffie-Hellman exponent (BDHE) problem [9] can be described as follows. Choose a group $\mathbb{G}$ with prime order p. Given

$$\mathbf{y} = (g, g^s, g^a, \cdots, g^{a^q}, , g^{a^{q+2}}, \cdots, g^{a^{2q}}, \\ \forall_{1\le j\le q}\ g^{sb_j}, g^{a/b_j}, \cdots, g^{a^q/b_j}, , g^{a^{q+2}/b_j}, \cdots, g^{a^{2q}/b_j}, \\ \forall_{1\le k,j\le q,k\ne j}\ g^{asb_k/b_j}, \cdots, g^{a^q sb_k/b_j})$$

where $a, s, b_1, \cdots, b_q \in \mathbb{Z}_p$ are chosen at random and g is a generator of $\mathbb{G}$. It's hard to distinguish a valid tuple $e(g,g)^{a^{q+1}s} \in \mathbb{G}_T$ from a random element $R \in \mathbb{G}_T$.

An algorithm $\mathcal{B}$ that outputs $z \in \{0,1\}$ has advantage ϵ in solving q-parallel BDHE problem in $\mathbb{G}$ if

$$\left|\Pr\left[\mathcal{B}(\mathbf{y}, e(g,g)^{a^{q+1}s}) = 0\right] - \Pr\left[\mathcal{B}(\mathbf{y}, R) = 0\right]\right| \ge \epsilon$$

Definition 2 (Decisional q-Parallel BDHE Assumption): We say the decisional q-parallel BDHE assumption holds if no probabilistic polynomial time algorithm can solve the decisional q-parallel BDHE problem with a non-negligible advantage.

C. BILINEAR DIFFIE-HELLMAN ASSUMPTION

The bilinear Diffie-Hellman (BDH) problem [28] can be described as follows. Choose a group $\mathbb{G}$ of prime order p. Given a tuple (g, g^a, g^b, g^c), where $a, b, c \in \mathbb{Z}_p$ are chosen at random and g is a generator of $\mathbb{G}$. It's hard to compute $e(g,g)^{abc} \in \mathbb{G}_T$.

Definition 3: (BDH Assumption): We say the BDH assumption holds if no probabilistic polynomial time algorithm can solve the BDH problem with a non-negligible probability.

D. LINEAR SECRET SHARING SCHEME

Definition 4: (Linear Secret Sharing Scheme, LSSS [9]): A secret-sharing scheme Π over a set of parties $\mathcal{P}$ is called linear (over $\mathbb{Z}_p$) if

1) *The shares for each party form a vector over $\mathbb{Z}_p$.*
2) *There exists a share-generating matrix M for Π, where M has l rows and n columns. For all $i = 1, 2, \cdots, l$, we let the function ρ label the i-th row of M as $\rho(i)$. Consider the vector $\mathbf{v} = (s, r_2, \cdots, r_n)$, where $s \in \mathbb{Z}_q$ is the secret to be shared, and $r_2, \cdots, r_n \in \mathbb{Z}_q$ are chosen at random. $M\mathbf{v}$ is the vector of l shares of the secret s according to Π. The share $(M\mathbf{v})_i$ belongs to party $\rho(i)$.*

Every LSSS has the property of linear reconstruction: Assume that Π is an LSSS for the access structure $\mathbb{A}$. Let $S \in \mathbb{A}$ be any authorized set, and $I \subset \{1, 2, \cdots, l\}$ be defined as $I = \{i : \rho(i) \in S\}$. Then, there exist constants $\{\omega_i \in \mathbb{Z}_p\}_{i\in I}$ such that, if $\{\lambda_i\}$ are valid shares of any secret s according to Π, we have $\Sigma_{i\in I}\omega_i\lambda_i = s$. These constants $\{\omega_i\}$ can be found in polynomial time in the size of the share-generating matrix M.

Note that the vector $(1, 0, \cdots, 0)$ is the "target" vector for any linear secret sharing scheme. For any authorized set of rows I in M, the target vector is in the span of I. For any unauthorized set of rows I, the target vector is not in the span of the rows of set I. Moreover, there exists a vector $\boldsymbol{\eta}$ such that $\boldsymbol{\eta} \cdot (1, 0, \cdots, 0) = -1$ and $\boldsymbol{\eta} \cdot M_i = 0$ for all $i \in I$.

IV. FRAMEWORK AND SECURITY MODEL

A. FRAMEWORK

Our RSABE scheme includes four entities: attribute authority (AA), cloud servers (CS), data owners and users.

- **AA**. The AA is fully trusted by the other three entities. It takes charge of setup of the system parameters and the registration of users. Moreover, the AA is in charge of the revoking and entitling user's attributes on the basis of his dynamical role.
- **Data Owners**. The data owner is responsible for uploading data to the cloud. The data owner firstly encrypts his data file with a symmetric key. Then, the owner extracts keywords set from the data file and establishes keywords index. Moreover, the owner defines an access policy of the data file and encrypts the symmetric key under the access policy to obtain a corresponding ciphertext.
- **CS**. The CS is in charge of data storage and provides data access service for its significant storage space and computation resource. Once it receives an access request from a user, the CS performs the retrieval operation with the submitted search trapdoor. If the search trapdoor matches the keyword index of some data file, the CS will partially decrypt the ciphertext of that file, and it will be successful only if the attributes of user satisfy the access policy in ciphertext. If succeed, then partially decrypted ciphertext and corresponding encrypted data file will be sent to the user. In addition, the CS is also responsible for large computing tasks during revocation phase,

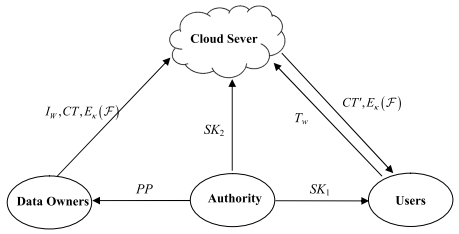


FIGURE 1. Framework of Our RSABE Scheme.

such as ciphertexts update and delegated secret keys update et al.

- **Users**. Every user has a unique identity, an attributes set, and a secret key associated with his attributes set. A user can access data files on the CS by providing a search trapdoor generated with an interested keyword and his secret key. Once receiving the search results, the user decrypts the partially decrypted ciphertext by using of his secret key, and finally gets the symmetric key.

The framework of our RSABE scheme is shown in Fig. 1.

Definition 5: Our RSABE scheme includes thirteen probabilistic polynomial time algorithms as follows:

- **Setup**$(\lambda) \rightarrow (MK, PP, \{VK_x\}_{x\in\mathcal{U}}, \{PK_x\}_{x\in\mathcal{U}})$. The setup algorithm is run by the AA. It takes as input a security parameter λ, and then outputs the master key MK, public parameters PP, a set of attribute version keys $\{VK_x\}_{x\in\mathcal{U}}$ and public attribute keys $\{PK_x\}_{x\in\mathcal{U}}$ for the attribute universe $\mathcal{U}$ in the system.
- **KeyGen**$(MK, S_{uid}, \{VK_x\}_{x\in S_{uid}}) \rightarrow SK$. The secret key generation algorithm is run by the AA. It takes as inputs the master key MK, an attributes set S_{uid} that describes the user uid and the corresponding attribute version keys $\{VK_x\}_{x\in S_{uid}}$, then outputs a secret key $SK = \{SK_1, SK_2\}$, where SK_1 will be sent to the user, and the SK_2 will be sent to the CS as the delegation key of user uid.
- **KeyIndex**$(PP, W) \rightarrow I_W$. The keywords index generation algorithm is run by the data owner. It takes as inputs the public parameters PP and a keywords set W of the shared data file, then outputs the keywords index I_W.
- **Encrpt**$(PP, \{PK_x\}_{x\in\mathcal{U}}, \kappa, \mathbb{A}) \rightarrow CT$. The encryption algorithm is run by the data owner. It takes as inputs the public parameters PP, the public attribute keys $\{PK_x\}_{x\in\mathcal{U}}$, the symmetric key κ and an access policy $\mathbb{A}$, then outputs a ciphertext CT.
- **Trapdoor**$(w, SK_1) \rightarrow T_w$. The trapdoor generation algorithm is run by user. Taking a search keyword w and the secret key component SK_1 of user uid, this algorithm outputs the search trapdoor T_w.
- **Test**$(T_w, SK_2, I_W) \rightarrow \{0, 1\}$. The test algorithm is run by the CS. Taking a search trapdoor T_w submitted by user uid, the secret key component SK_2 of user uid and a keywords index I_W, this algorithm outputs 1 if it is matched and 0 otherwise.
- **PreDecrpt**$(CT, SK_2) \rightarrow CT'$. The preprocessing decryption algorithm is run by the CS. Taking the ciphertext CT associated with an access policy $\mathbb{A}$ and the secret key component SK_2 integrated with attributes set S_{uid}, this algorithm will output a partially decrypted ciphertext CT' if S_{uid} satisfies $\mathbb{A}$, and the error symbol $\perp$ otherwise.
- **PostDecrpt**$(CT', SK_1) \rightarrow \kappa$. The post-processing decryption algorithm is run by user. Taking the ciphertext CT' which is partially decrypted by SK_2 and the secret key component SK_1 of user uid, this algorithm outputs the symmetric key κ only if SK_1 and SK_2 are both generated for the user uid by the AA.
- **UKeyGen**$(x', VK_{x'}) \rightarrow \widetilde{VK}_{x'}, UK_{x'}$. The update key generation algorithm is run by the AA. Taking the revoked attribute x' and its current attribute version key $VK_{x'}$, this algorithm outputs a new attribute version key $\widetilde{VK}_{x'}$ and an attribute update key $UK_{x'}$.
- **PKUpdate**$(PK_{x'}, UK_{x'}) \rightarrow \widetilde{PK}_{x'}$. The public attribute key update algorithm is run by the AA. Given the public attribute key $PK_{x'}$ and the attribute update key $UK_{x'}$, this algorithm outputs an updated public attribute key $\widetilde{PK}_{x'}$.
- **CTUpdate**$(CT, UK_{x'}) \rightarrow \widetilde{CT}$. The ciphertext update algorithm is run by the CS. Given the ciphertext CT associated with the revoked attribute x' and the attribute update key $UK_{x'}$, the algorithm outputs an update ciphertext $\widetilde{CT}$.
- **SKUpdate**$(SK_2, UK_{x'}) \rightarrow \widetilde{SK}_2$. The secret key update algorithm is run by the CS. Given the secret key component SK_2 of non-revoked user uid and the attribute update key $UK_{x'}$, the algorithm outputs an updated secret key component $\widetilde{SK}_2$.
- **GrantAtt**$(SK_2, VK_{x''}) \rightarrow \widetilde{SK}_2$. The attribute grant algorithm is run by the AA. Given the secret key component SK_2 of granted user uid and the version number $VK_{x''}$ of the granted attribute x'', the algorithm outputs an updated secret key component $\widetilde{SK}_2$.

B. SECURITY MODEL

The security of RSABE relies on decisional q-parallel BDHE assumption and BDH assumption. As most of the update works are implemented by the CS, it is assumed that the CS will not collude with the revoked users. To demonstrate the security of our scheme, we design two security games: indistinguishability against selective ciphertext-policy and chosen plaintext attack (IND-sCP-CPA) game and indistinguishability against chosen keyword attack (IND-CKA) game.

IND-sCP-CPA game:

Init. The adversary $\mathcal{A}$ declares an access structure $\mathbb{A}^*$ which he wishes to challenge upon.

Setup. The challenger $\mathcal{B}$ runs the Setup algorithm. Then he sends the public parameters PP to $\mathcal{A}$, and keeps the master key MK for himself.

Phase 1. $\mathcal{A}$ can issue the following oracles in polynomial many times:

- $\mathcal{O}_{SK}(uid, S_{uid})$: $\mathcal{A}$ can issue queries for secret key SK by submitting pairs (uid, S_{uid}), where uid is an identity, S_{uid} is an attributes set of user uid, with the restriction that S_{uid} does not satisfy $\mathbb{A}^*$.

- $\mathcal{O}_{UK}(x')$: $\mathcal{A}$ also can make update key queries by submitting any attribute x'.
- $\mathcal{O}_{GA}(uid, S_{uid}, x'')$: $\mathcal{A}$ is allowed to issue queries for update secret key SK by submitting tuple (uid, S_{uid}, x'') with the following constrains: (uid, S_{uid}) has been queried for oracle $\mathcal{O}_{SK}$, and $S_{uid} \cup \{x''\}$ does not satisfy $\mathbb{A}^*$, where x'' is the attribute granted to user uid.

Challenge. $\mathcal{A}$ submits two messages κ_0, κ_1 of equal length. $\mathcal{B}$ randomly tosses a bit $b \in \{0, 1\}$ and encrypts κ_b under $\mathbb{A}^*$, then passes the challenge ciphertext CT^* to $\mathcal{A}$.

Phase 2. Same as Phase 1.

Guess. $\mathcal{A}$ outputs a guess b' of b and wins the game if $b' = b$.

The advantage of $\mathcal{A}$ in this game is defined as

$$Adv_{\mathcal{A}}^{\text{IND}-\text{sCP}-\text{CPA}} = \left| \Pr\left[b' = b\right] - \frac{1}{2} \right|.$$

Definition 6: An RSABE scheme is IND-sCP-CPA secure if all polynomial time adversaries have at most a negligible advantage in the above game.

IND-CKA game:

Setup. The challenger $\mathcal{B}$ runs the Setup algorithm. Then he sends the public parameters PP to $\mathcal{A}$, and keeps the master key MK for himself.

Phase 1. $\mathcal{A}$ can query the trapdoor of a keyword w in polynomial many times.

Challenge. $\mathcal{A}$ submits two equal length keywords w_0, w_1 with the restriction that w_0 and w_1 have not been queried for the trapdoor. $\mathcal{B}$ flips a random bit $b \in \{0, 1\}$ and generates index I_{w_b} for keyword w_b, then passes the challenge index I_{w_b} to $\mathcal{A}$.

Phase 2. $\mathcal{A}$ can issue more trapdoor queries for keyword w with the restriction $w \neq w_0, w_1$.

Guess. $\mathcal{A}$ outputs a guess b' of b and wins the game if $b' = b$.

The advantage of $\mathcal{A}$ in this game is defined as

$$Adv_{\mathcal{A}}^{\text{IND}-\text{CKA}} = \left| \Pr\left[b' = b\right] - \frac{1}{2} \right|.$$

Definition 7: An RSABE scheme is IND-CKA secure if all polynomial time adversaries have at most a negligible advantage in the above game.

V. OUR CONSTRUCTION

This paper aims to address the issue of how to support efficient computation, attribute revocation, attribute grant and keyword search. Specially, our solution greatly reduces the computational and storage overload of the user side by means of outsourced decryption technology. The users only store secret key component SK_1 independent with user's attributes, and outsource secret key component SK_2 associated with user's attributes to the CS, which reduces storage load on user side to a large extent. Thus, partial decryption operations in which computational cost grows with the number of attributes are outsourced to the cloud server, so the user only needs to decrypt with constant computation. During the revocation, user's secret key SK_2 update and data re-encryption can be outsourced to the cloud server, and the other secret key component SK_1 that user holds does not need to update. Our RSABE scheme can satisfy the special application need of low computation and storage overhead in mobile cloud storage systems with resource-limited mobile users, so it can be applied to many mobile cloud storage systems, such as electronic health record system.

Inspired by Waters' CP-ABE [9] and Boneh's PKES [28], we give the detailed construction of our RSABE scheme in this section. There are seven phases in the whole construction: system setup, key generation, data outsourcing, keyword search, data decryption, attribute revocation and attribute grant.

A. SYSTEM SETUP

At this phase, AA initializes the system by calling the algorithm Setup.

Setup$(\lambda) \rightarrow (MK, PP, \{VK_x\}_{x\in\mathcal{U}}, \{PK_x\}_{x\in\mathcal{U}})$: It takes as input a security parameter λ. This algorithm chooses a bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$, where $\mathbb{G}$ and $\mathbb{G}_\mathbb{T}$ are two multiplicative cyclic groups of prime order p. Let g be a generator of $\mathbb{G}$. It also chooses three hash functions $H : \{0, 1\}^* \rightarrow \mathbb{G}$, $H_1 : \{0, 1\}^* \rightarrow \mathbb{G}$, and $H_2 : \mathbb{G}_T \rightarrow \{0, 1\}^{\log p}$ that will be modeled as random oracles. Then, define $\mathcal{U}$ as attribute universe. For each attribute $x \in \mathcal{U}$, denote the set of users whose attribute x is revoked as attribute revocation list RL_x, and denote the set of users who are granted the attribute x as attribute grant list GL_x. It randomly chooses numbers $a, \alpha, \bar{\alpha} \in \mathbb{Z}_p$ and sets the master keys as

$$MK = \{\alpha, \bar{\alpha}\},$$

where α is used for data encryption, and $\bar{\alpha}$ is used for keyword search. It sets the public parameters as

$$PP = \{p, g, H, H_1, H_2, g^a, e(g, g)^\alpha, g^{\bar{\alpha}}\}.$$

For each attribute $x \in \mathcal{U}$, it randomly chooses $v_x \in \mathbb{Z}_p$ and computes the public attribute key $PK_x = g^{v_x}$ by implicitly setting the corresponding attribute version key $VK_x = v_x$. Then AA publishes the public parameters PP and public attribute keys $\{PK_x\}_{x\in\mathcal{U}}$, and keep the master keys MK and attribute version keys $\{VK_x\}_{x\in\mathcal{U}}$ secret.

B. KEY GENERATION

When a new user joins the system, AA assigns a unique identity uid and an attributes set S_{uid} for him. Then, AA runs the algorithm KeyGen to issues secret key SK to this user.

KeyGen$(MK, S_{uid}, \{VK_x\}_{x\in S_{uid}}) \rightarrow SK$: It takes as inputs the master keys MK, an attributes set S_{uid} belonging to user uid and the corresponding attribute version keys $\{VK_x\}_{x\in S_{uid}}$. The algorithm chooses random numbers $\alpha_1, \alpha_2 \in \mathbb{Z}_p^*$ such that $\alpha = \alpha_1 + \alpha_2 \bmod p$. It also randomly chooses $\delta_{uid}, t \in \mathbb{Z}_p^*$, then computes the secret key as $SK = \{SK_1, SK_2\}$, where

$$SK_1 = \{\tilde{\alpha} = \bar{\alpha}/\delta_{uid}, K = g^{\alpha_1} g^{at}\},$$
$$SK_2 = \{\delta_{uid}, E = g^{\alpha_2}, L = g^t, \{K_x = H(x)^{t/v_x}\}_{x\in S_{uid}}\}.$$

The AA sends SK_1 to user uid and SK_2 to the CS. The CS adds the tuple (uid, S_{uid}, SK_2) into the user secret key list L_{SK_2}.

C. DATA OUTSOURCING

When a data owner outsource a file $\mathcal{F}$ to the mobile cloud sever, he proceeds as follows:

Step 1: keywords index building

To share the data file with other users, the owner should first build an encrypted keywords index for the file by calling the algorithm KeyIndex.

KeyIndex$(PP, W) \rightarrow I_W$: It takes as inputs the public parameters PP, and the keywords set $W = \{w_1, w_2, \cdots, w_m\}$ extracted from the data file $\mathcal{F}$. For each keyword $w_j \in W$, the algorithm randomly selects a number $\lambda_j \in \mathbb{Z}_p^*$ and computes

$$I_{w_j} = \left\{A_j = g^{\gamma_j}, B_j = H_2\left(e\left((g^{\tilde{\alpha}})^{\gamma_j}, H_1(w_j)\right)\right)\right\}$$

Finally, the algorithm outputs the index for the keywords set W as $I_W = \{I_{w_j}\}_{j=1,\cdots,m}$.

Step 2: data encryption

The data owner encrypts the data file $\mathcal{F}$ with a random symmetric key κ by means of symmetric encryption algorithm such as AES, and denotes the encrypted result as $E_\kappa(\mathcal{F})$. Then it defines an LSSS access policy (M, ρ) about the symmetric key κ and encrypts κ under (M, ρ) by calling algorithm Encrypt.

Encrpt$(PP, \{PK_x\}_{x\in\mathcal{U}}, \kappa, (M, \rho)) \rightarrow CT$: It inputs the public parameters PP, the public attribute keys $\{PK_x\}_{x\in\mathcal{U}}$, the symmetric key κ and an access policy (M, ρ), where M is an $l \times n$ matrix, the function ρ maps each row of M to an attribute. In our construction, the function ρ is no more limited to an injective function. It first chooses a random encryption exponent $s \in \mathbb{Z}_p$ and vector $\mathbf{v} = (s, y_2, \cdots, y_n) \in \mathbb{Z}_p^n$, where $y_2, \cdots, y_n$ will be used to share the encryption exponent s. For $i = 1$ to l, it computes $\lambda_i = M_i \cdot \mathbf{v}$, where M_i is the vector corresponding to the i-th row of M. Then, it randomly chooses $r_1, \cdots, r_l \in \mathbb{Z}_p$ and computes the ciphertext as

$$\begin{aligned} CT = \{ &(M, \rho), C = \kappa \cdot e(g, g)^{\alpha s}, C' = g^s, \\ &\forall i = 1, \cdots, l : C_i = g^{a\lambda_i} H(\rho(i))^{r_i}, \\ &D_i = (PK_{\rho(i)})^{r_i} \} \end{aligned}$$

After all that work, the owner sends the keyword index I_W, the ciphertext CT and the encrypted data file to the CS with the format in Fig. 2.

D. KEYWORD SEARCH

When a data user want to download some interested encrypted file in CS, he proceeds as follows:

Step 1: trapdoor generation by user

The user uid first generates a trapdoor of the interested keyword w by calling the algorithm Trapdoor.

Trapdoor$(w, SK_1) \rightarrow T_w$: It takes as inputs a search keyword w and the secret key component SK_1 of user uid.

I_W	CT	$E_\kappa(\mathcal{F})$

FIGURE 2. Format of encrypted data stored in cloud sever.

The algorithm computes the trapdoor as

$$T_w = H_1(w)^{\tilde{\alpha}}.$$

On obtaining the trapdoor T_w, the user sends the tuple (uid, T_w) to the CS.

Step 2: search for data by the CS

The CS performs the search operation by calling the algorithm Test.

Test$(T_w, SK_2, I_W) \rightarrow \{0, 1\}$: It takes as inputs the trapdoor T_w submitted by user uid, the secret key component SK_2 of user uid and an index $I_W = \{I_{w_j}\}_{j=1,\cdots,m}$ of the data file $\mathcal{F}$. It checks if the following equation holds $H_2(e(A_j, (T_w)^{\delta_{uid}})) = B_j$ for some keyword index $I_{w_j} \in I_W$. If the equation holds, it outputs 1 and go to the phase of data decryption. Otherwise, it outputs 0.

E. DATA DECRYPTION

The outsourced decryption technology is used here to reduce the decryption overhead on user side. The decryption phrase consists of two steps.

Step1: decryption by the CS

If the output of the previous Test algorithm is 1, which means that the keyword index is matched with the keyword trapdoor, the CS partially decrypts the corresponding ciphertext CT by calling the algorithm PreDecrpt.

PreDecrpt$(CT, SK_2) \rightarrow CT'$: It takes as inputs the ciphertext CT and the secret key component SK_2 of user uid. If the attributes set S_{uid} used in SK_2 satisfies the access structure (M, ρ) embedded in CT, the algorithm proceeds as follows:

Let $I \subset \{1, 2, \cdots, l\}$ be defined as $I = \{i : \rho(i) \in S_{uid}\}$. Then, it chooses a set of constants $\{\omega_i \in \mathbb{Z}_p\}_{i\in I}$ such that $\Sigma_{i\in I}\omega_i\lambda_i = s$ if $\{\lambda_i\}$ are valid shares of any secret s according to M. It computes

$$A = \frac{\Pi_{i\in I} e(C_i, L)^{\omega_i}}{e(C', E)\Pi_{i\in I} e(D_i, K_{\rho(i)})^{\omega_i}} = \frac{e(g, g)^{ats}}{e(g, g)^{\alpha_2 s}}.$$

The algorithm outputs a partially decrypted ciphertext $CT' = \{C, C', A\}$. Then, the CS transmits the partially decrypted ciphertext CT' along with the encrypted data file $E_\kappa(\mathcal{F})$ to user uid.

Step2: decryption by user

When the user uid receives the search results from the CS, it further decrypts the partially decrypted ciphertext CT' by calling the algorithm PostDecrpt.

PostDecrpt$(CT', SK_1) \rightarrow \kappa$: It takes as inputs the partially decrypted ciphertext CT' and the secret key component SK_1 of user uid. The symmetric key is computed as

$$\kappa = \frac{C \cdot A}{e(C', K)}.$$

Eventually, the user could decrypt the encrypted data file $E_\kappa(\mathcal{F})$ utilizing the symmetric key κ, and finally gets the data file $\mathcal{F}$.

F. ATTRIBUTE REVOCATION

When an attribute x' is revoked from some users, the AA adds these users' identity into attribute revocation list $RL_{x'}$. Our revocation method includes the following two steps:

Step 1: attribute update key generation and public attribute key update by the AA

The AA computes the attribute update key by running the algorithm UKeyGen.

UKeyGen$(x', VK_{x'}) \rightarrow \widetilde{VK}_{x'}, UK_{x'}$: It takes as inputs the revoked attribute x' and its current attribute version key $VK_{x'} = v_{x'}$. The algorithm first chooses a new attribute version key $\widetilde{VK}_{x'} = \tilde{v}_{x'} \in \mathbb{Z}_p^* \ (\tilde{v}_{x'} \neq v_{x'})$ for the revoked attribute x', and then the attribute update key is computed as

$$UK_{x'} = \tilde{v}_{x'}/v_{x'}.$$

Then AA updates the public attribute key of the revoked attribute x' by calling the algorithm PKUpdate.

PKUpdate$(PK_{x'}, UK_{x'}) \rightarrow \widetilde{PK}_{x'}$: It takes as inputs the public attribute key $PK_{x'}$ and attribute update key $UK_{x'}$, the new public attribute key is computed as

$$\widetilde{PK}_{x'} = (PK_{x'})^{UK_{x'}}.$$

The AA publishes the updated public attribute key on public board, and sends the update key $UK_{x'}$ together with the attribute revocation list $RL_{x'}$ to the CS.

Step 2: ciphertexts and delegated secret keys update by the CS

When the CS receives the attribute update key $UK_{x'}$, it updates the ciphertexts related to the revoked attribute x' by running the algorithm CTUpdate.

CTUpdate$(CT, UK_{x'}) \rightarrow \widetilde{CT}$: It takes as inputs the current ciphertext CT and the attribute update key $UK_{x'}$. The new ciphertext is computed as:

$$\begin{aligned}\widetilde{CT} = \{ & \widetilde{C} = C, \widetilde{C}' = C', \\ & \forall i = 1, \cdots, l : \tilde{C}_i = C_i, \\ & for\ \rho(i) \neq x' : \tilde{D}_i = D_i, \\ & for\ \rho(i) = x' : \tilde{D}_i = (D_i)^{UK_{\rho(i)}} \}.\end{aligned}$$

The CS updates the delegated secret key component SK_2 for these non-revoked users by running the algorithm SKUpdate.

SKUpdate$(SK_2, UK_{x'}) \rightarrow \widetilde{SK}_2$: It inputs the current secret key component SK_2 of user $uid \notin RL_{x'}$ and the attribute update key $UK_{x'}$. The algorithm computes the updated secret key component $\widetilde{SK}_2$ as:

$$\begin{aligned}\widetilde{SK}_2 = \{ & \tilde{\delta}_{uid} = \delta_{uid}, \tilde{E} = E, \tilde{L} = L, \\ & for\ x \neq x' : \tilde{K}_x = K_x, \\ & for\ x = x' : \tilde{K}_x = (K_x)^{UK_{x'}^{-1}} \}.\end{aligned}$$

Then CS maintains the user secret key list L_{SK_2}.

Note.

1) Only the secret key components SK_2 need to be updated, while components SK_1 held by users keep unchanged. By this means, our solution can largely reduce the communication overload of users during the revocation.
2) When an attribute x' is revoked from some users, the AA adds these users' identity into attribute revocation list RL'_x. The components associated with attributes will be affected, like cipertexts and user's secret key. Note that attribute revocation would not influence keyword searching operations, as the keyword index and trapdoor that keyword searching operations involve are unrelated with attributes.
3) To handle the revocation more efficiently in mobile cloud storage, we could adopt the lazy-update technique [58]. The CS stores the attribute update keys $UK_{x'}$ and once the ciphertext is accessed by some user, the CS will first check whether any attributes embedded in the ciphertext have been revoked. If any, the CS will update the ciphertext. Moreover, the CS is able to batch multiple $UK_{x'}$ by calling the CTUpdate algorithm once with $\prod UK_{x'}$. For example, in a certain accessed ciphertext, there are three attributes x', x'', x''' which have been revoked from some users, the CS only needs to call the CTUpdate algorithm once with $UK_{x'} \times UK_{x''} \times UK_{x'''}$. Similarly, secret key component SK_2 will not be updated until the corresponding user makes search requests.

G. ATTRIBUTE GRANT

When some users come to hold a new attribute x'', the AA adds these users' identity into attribute grant list $GL_{x''}$, then issues the secret key component related to the granted attribute x'' for these users by running the algorithm GrantAtt.

GrantAtt$(SK_2, VK_{x''})$: It takes as inputs the current secret key component SK_2 of user $uid \in GL_{x''}$ and the attribute version key $VK_{x''}$. Then the algorithm computes the update secret key component $\widetilde{SK}_2$ as

$$\begin{aligned}\widetilde{SK}_2 = \{ & \tilde{\delta}_{uid} = \delta_{uid}, \tilde{E} = E, \tilde{L} = L, \\ & for\ x \neq x'' : \tilde{K}_x = K_x, \\ & for\ x = x'' : \tilde{K}_x = H(x)^{t/VK_{x''}} \}.\end{aligned}$$

Then, the AA sends the $\widetilde{SK}_2$ together with the attribute grant list $GL_{x''}$ to the CS. The CS maintains the user secret key list L_{SK_2}.

VI. SECURITY ANALYSIS

A. CORRECTNESS

The correctness of decryption:

$$\begin{aligned}A &= \frac{\prod_{i\in I} e(C_i, L)^{\omega_i}}{e(C', E)\prod_{i\in I} e\left(D_i, K_{\rho(i)}\right)^{\omega_i}} \\ &= \frac{\prod_{i\in I} e\left(g^{a\lambda_i} H\left(\rho\left(i\right)\right)^{r_i}, g^t\right)^{\omega_i}}{e\left(g^s, g^{\alpha_2}\right)\prod_{i\in I} e\left(\left(g^{v_{\rho(i)}}\right)^{r_i}, H\left(\rho\left(i\right)\right)^{t/v_{\rho(i)}}\right)^{\omega_i}} \\ &= \frac{e(g, g)^{ats}}{e(g, g)^{\alpha_2 s}}.\end{aligned}$$

then

$$\frac{C \cdot A}{e(C', K)} = \frac{\kappa \cdot e(g, g)^{\alpha s} \cdot e(g, g)^{ats}}{e(g^s, g^{\alpha_1} g^{at}) \cdot e(g, g)^{\alpha_2 s}} = \kappa.$$

The correctness of keyword searchčž

$$\begin{aligned} H_2\left(e(A_j, (T_w)^{\delta_{uid}})\right) &= H_2\left(e(g^{\gamma_j}, \left(H_1(w)^{\bar{\alpha}}\right)^{\delta_{uid}})\right) \\ &= H_2\left(e(g^{\gamma_j}, (H_1(w)^{\bar{\alpha}/\delta_{uid}})^{\delta_{uid}})\right) \\ &= H_2\left(e(g^{\gamma_j}, H_1(w)^{\bar{\alpha}})\right) \\ &= H_2\left(e((g^{\bar{\alpha}})^{\gamma_j}, H_1(w))\right) \\ &= B_j\left(\text{iff } w = w_j\right). \end{aligned}$$

B. SECURITY PROOF

We exploit Waters' [9] proof method to prove the security of IND-sCP-CPA game in our RSABE.

Theorem 1: Suppose the decisional q-parallel BDHE assumption holds in $\mathbb{G}$ and $\mathbb{G}_T$. Then no polynomial time adversary can win the IND-sCP-CPA game with a non-negligible advantage.

Proof: Suppose there exists a polynomial time adversary $\mathcal{A}$ who is able to distinguish a valid ciphertext from a random element with non-negligible advantage $\epsilon_1 = Adv_{\mathcal{A}}^{\text{IND-sCP-CPA}}$. Then we can construct a simulator $\mathcal{B}$ to solve the decisional q-parallel BDHE problem with non-negligible advantage $\epsilon_1/2$. The simulation proceeds as follows.

The q-parallel BDHE challenger $\mathcal{C}$ first chooses $a, s, b_1, \cdots, b_q$ in $\mathbb{Z}_p$ at random, and sets:

$$\begin{aligned} \mathbf{y} = (&g, g^s, g^a, \cdots, g^{a^q}, , g^{a^{q+2}}, \cdots, g^{a^{2q}}, \\ &\forall_{1\le j\le q}\ g^{sb_j}, g^{a/b_j}, \cdots, g^{a^q/b_j}, , g^{a^{q+2}/b_j}, \cdots, g^{a^{2q}/b_j}, \\ &\forall_{1\le k,j\le q, k\ne j}\ g^{asb_k/b_j}, \cdots, g^{a^q sb_k/b_j}), \end{aligned}$$

Then $\mathcal{C}$ picks a fair binary coin $\mu \in \{0, 1\}$: if $\mu = 0$, $\mathcal{C}$ sets $T = e(g, g)^{a^{q+1}s}$; if $\mu = 1$, then $\mathcal{C}$ chooses a random element T from $\mathbb{G}_T$.

Init. $\mathcal{B}$ is given a decisional q-parallel BDHE challenge instance $(\mathbf{y}, T)$. $\mathcal{A}$ declares the challenge access structure (M^*, ρ^*), where M^* has $n^* < q$ columns.

Setup. $\mathcal{B}$ randomly selects a number $\alpha' \in \mathbb{Z}_p$ and implicitly lets $\alpha = \alpha' + a^{q+1}$ by setting $e(g, g)^{\alpha} = e(g^a, g^{a^q}) \cdot e(g, g)^{\alpha'}$.

$\mathcal{B}$ programs the random oracle H using a hash list L_H. Consider a call to $H(x)$, if $H(x)$ was already appeared in L_H, then $\mathcal{B}$ simply returns the same answer as before. Otherwise, let X denote the set of indices i, such that $\rho^*(i) = x$. $\mathcal{B}$ randomly selects $z_x \in \mathbb{Z}_p$, and processes as:

$$H(x) = g^{z_x} \prod_{i\in X} g^{aM^*_{i,1}/b_i} \cdot g^{a^2 M^*_{i,2}/b_i} \cdots g^{a^{n^*} M^*_{i,n^*}/b_i}.$$

Note that if $X = \varnothing$ then $H(x) = g^{z_x}$, and the responses of $H(x)$ are randomly distributed due to the value g^{z_x}. Moreover, for each attribute x, $\mathcal{B}$ initializes an attribute version key v_x by randomly selecting $v_x \in \mathbb{Z}_p$, and sets public attribute key $PK_x = g^{v_x}$.

Phase I. $\mathcal{B}$ maintains a list of tuples (uid, S_{uid}, SK) denoted as L_{SK} which is initially empty. $\mathcal{A}$ can polynomially query the following oracles:

- $\mathcal{O}_{SK}(uid, S_{uid})$: Assume that $\mathcal{B}$ is given secret key query for (uid, S_{uid}), where S_{uid} does not satisfy (M^*, ρ^*), $\mathcal{B}$ proceeds as follows:
  If the pair (uid, S_{uid}) has been queried before, retrieve SK from the L_{SK} with respect to (uid, S_{uid}), and then returns SK.
  Otherwise, $\mathcal{B}$ finds a vector $\boldsymbol{\eta} = (\eta_1, \cdots, \eta_{n^*}) \in \mathbb{Z}_p^*$ such that $\eta_1 = -1$ and $M_i^* \cdot \boldsymbol{\eta} = 0$ for all i where $\rho^*(i) \in S_{uid}$. Such a vector must exist by the definition of LSSS. The simulator randomly chooses $\alpha'_1, \alpha'_2 \in \mathbb{Z}_p$, such that $\alpha'_1 + \alpha'_2 = \alpha' \mod p$, and set $\alpha_1 = \alpha'_1 + a^{q+1}$, $\alpha_2 = \alpha'_2$. Then $\mathcal{B}$ computes E as:

  $$E = g^{\alpha'_2} = g^{\alpha_2}.$$

  $\mathcal{B}$ randomly selects a number $r \in \mathbb{Z}_p$, and implicitly defines t as:

  $$t = r + \eta_1\, a^q + \eta_2\, a^{q-1} + \cdots + \eta_{n^*} a^{q+1-n^*}.$$

  It performs this by setting L as:

  $$L = g^r \cdot \prod_{i=1,\cdots,n^*} (g^{a^{q+1-i}})^{\eta_i} = g^t.$$

  By the definition of t, we know that g^{at} includes a item of the form $g^{-a^{q+1}}$, which will cancel out with the unknown item in g^{α_1} when computing K. $\mathcal{B}$ calculates K as:

  $$K = g^{\alpha'_1} g^{ar} \cdot \prod_{i=2,\cdots,n^*} (g^{a^{q+2-i}})^{\eta_i} = g^{\alpha_1} g^{at}.$$

  Now $\mathcal{B}$ calculates K_x for all $x \in S_{uid}$. For each attributes $x \in S_{uid}$, when there is no i such that $\rho^*(i) = x$, let $K_x = L^{z_x/v_x}$. While for those attributes $x \in S_{uid}$ used in the access structure, K_x contains items with form g^{a^{q+1}/b_i} that $\mathcal{B}$ cannot simulate. However, we have $M_i^* \cdot \boldsymbol{\eta} = 0$, all of these items of form g^{a^{q+1}/b_i} will be canceled. $\mathcal{B}$ computes K_x as:

  $$\begin{aligned} K_x &= L^{z_x/v_x} \prod_{i\in X} \prod_{j=1,\cdots,n^*} ((g^{a^j/b_i})^r)^{M^*_{i,j}/v_x} \\ &\quad \cdot \prod_{i\in X} \prod_{j=1,\cdots,n^*} \prod_{\substack{k=1,\cdots,n^* \\ k\ne j}} ((g^{a^{q+1+j-k}/b_i})^{\eta_k})^{M^*_{i,j}/v_x} \\ &= H(x)^{t/v_x} \end{aligned}$$

  $\mathcal{B}$ adds the secret key $SK = \{E, L, K, \{K_x\}_{x\in S_{uid}}\}$ into list L_{SK} and gives it to $\mathcal{A}$.
- $\mathcal{O}_{UK}(x')$: $\mathcal{A}$ submits an attribute x' for the attribute update key query. $\mathcal{B}$ chooses a random value $\tilde{v}_{x'} \in \mathbb{Z}_p$ as a new attribute version number of x', and returns the attribute update key $UK_{x'} = \tilde{v}_{x'}/v_{x'}$ to $\mathcal{A}$.
- $\mathcal{O}_{GA}(uid, S_{uid}, x'')$: $\mathcal{A}$ submits a pair (uid, S_{uid}) that has been queried for SK and the attribute x'' granted to uid,

where $S_{uid} \cup \{x''\}$ does not satisfy (M^*, ρ^*). When there is no i such that $\rho^*(i) = x''$, $\mathcal{B}$ returns $K_{x''} = L^{z_{x''}/v_{x''}}$. Otherwise, $\mathcal{B}$ queries $\mathcal{O}_{SK}$ with $(uid, S_{uid} \cup \{x''\})$ to get a new secret key SK for uid, then updates the list L_{SK} and returns the new secret key SK to $\mathcal{A}$.

Challenge. $\mathcal{A}$ gives two challenge messages κ_0, κ_1 of equal length to $\mathcal{B}$. Then $\mathcal{B}$ flips a coin $b \in \{0, 1\}$, and calculates C^* as follows:

$$C^* = \kappa_b \cdot T \cdot e(g^s, g^{\alpha'}), \quad C'^* = g^s$$

It is difficult for $\mathcal{B}$ to simulate C_i^* because it contains items $g^{a^j s}$ that $\mathcal{B}$ cannot simulate. However, $\mathcal{B}$ could cancel out these items by choosing the secret splitting. Intuitively, $\mathcal{B}$ randomly chooses $y_2', \cdots, y_{n^*}' \in \mathbb{Z}_p$, then shares the secret s utilizing the vector

$$\mathbf{v} = (s, sa + y_2', sa^2 + y_3', \cdots, sa^{n^*-1} + y_{n^*}') \in \mathbb{Z}_p^{n^*}$$

For $i = 1, \cdots, l$, we define R_i as the set of all $k \neq i$ such that $\rho^*(i) = \rho^*(k)$. $\mathcal{B}$ also randomly chooses $r_1', \cdots, r_l' \in \mathbb{Z}_p$. By implicitly setting $r_i = -r_i' - sb_i$, $\mathcal{B}$ computes D_i^* and C_i^* as:

$$\begin{aligned} D_i^* &= g^{(-r_i' - sb_i) \cdot v_{\rho^*(i)}} \\ C_i^* &= H(\rho^*(i))^{-r_i'} \cdot (g^{sb_i})^{-z_{\rho^*(i)}} \cdot \prod_{j=2,\cdots,n^*} (g^a)^{M_{i,j}^* \cdot y_j'} \\ &\quad \cdot \prod_{k \in R_i} \prod_{j=1,\cdots,n^*} (g^{a^j sb_i/b_k})^{-M_{k,j}^*} \end{aligned}$$

$\mathcal{B}$ gives the challenge ciphertext $CT^* = \{C^*, C'^*, \{C_i^*, D_i^*\}_{i=1,2,\cdots,l}\}$ to $\mathcal{A}$.

Phase II. Same as Phase I.

Guess. $\mathcal{A}$ will eventually output a guess b' of b. Then $\mathcal{B}$ outputs $\mu' = 0$ to guess that $T = e(g, g)^{a^{q+1}s}$ if $b' = b$. Otherwise, $\mathcal{B}$ outputs $\mu' = 1$ to indicate that T is a random group element in $\mathbb{G}_T$. If $\mu = 0$, $\mathcal{A}$ obtains a valid ciphertext of κ_b. $\mathcal{A}$'s advantage in this situation is ϵ_1 by definition, therefore $\Pr[b' = b|\mu = 0] = 1/2 + \epsilon_1$. Since $\mathcal{B}$ guesses $\mu' = 0$ when $b' = b$, we have $\Pr[\mu' = \mu|\mu = 0] = 1/2 + \epsilon_1$. If $\mu = 1$, $\mathcal{A}$ gets no information about b, we have $\Pr[b' \neq b|\mu = 1] = 1/2$. As $\mathcal{B}$ guesses $\mu' = 1$ when $b' \neq b$, we have $\Pr[\mu' = \mu|\mu = 1] = 1/2$. The advantage of $\mathcal{B}$ to solve the decisional q-parallel BDHE problem is

$$\begin{aligned} \Pr[\mu' = \mu] - \frac{1}{2} &= \frac{1}{2}\Pr[\mu' = \mu|\mu = 0] \\ &\quad + \frac{1}{2}\Pr[\mu' = \mu|\mu = 1] - \frac{1}{2} \\ &= \frac{1}{2}(\frac{1}{2} + \epsilon_1) + \frac{1}{2} \cdot \frac{1}{2} - \frac{1}{2} \\ &= \frac{\epsilon_1}{2}. \end{aligned}$$

□

We expand Boneh's [28] proof method to prove the security of IND-CKA game in our RSABE.

Theorem 2: Suppose the BDH assumption holds in $\mathbb{G}$ and $\mathbb{G}_T$. Then no polynomial time adversary can win the IND-CKA game with a non-negligible advantage.

Proof: Suppose there exists a polynomial time adversary $\mathcal{A}$ who is able to distinguish a valid index from a random element with non-negligible advantage $\epsilon_2 = Adv_{\mathcal{A}}^{\text{IND-CKA}}$. Suppose $\mathcal{A}$ makes at most $q_{H_2} > 0$ hash queries to H_2 and at most $q_T > 0$ trapdoor queries. Then we can construct an algorithm $\mathcal{B}$ that solves the BDH problem with probability at least $\epsilon_2/(eq_T q_{H_2})$, where e is the base of the natural logarithm.

Init. Let g be a generator of $\mathbb{G}$. The simulator $\mathcal{B}$ is given a BDH instance $(g, u_1 = g^{\tilde{\alpha}}, u_2 = g^{\beta}, u_3 = g^{\gamma})$. $\mathcal{B}$ aims at computing $e(g, g)^{\tilde{\alpha}\beta\gamma} \in \mathbb{G}_T$.

Setup. $\mathcal{B}$ gives the public parameter component (g, u_1) that only related to keyword search to $\mathcal{A}$.

Phase 1. $\mathcal{A}$ can polynomially query the following oracles:

- $\mathcal{O}_{H_1}(w_i)$: To respond to H_1 queries, $\mathcal{B}$ maintains a hash list of tuples (w_j, h_j, a_j, c_j) denoted as L_{H_1}. When $\mathcal{A}$ issues H_1 query with a keyword $w_i \in \{0, 1\}^*$, $\mathcal{B}$ proceeds as follows:
  - If w_i already appears in a tuple (w_i, h_i, a_i, c_i) of L_{H_1}, then $\mathcal{B}$ responds with $H_1(w_i) = h_i \in \mathbb{G}$.
  - Otherwise, $\mathcal{B}$ picks a random coin $c_i \in \{0, 1\}$ such that $\Pr[c_i = 0] = 1/(q_T + 1)$, and then randomly selects a number $a_i \in \mathbb{Z}_p$. If $c_i = 0$, $\mathcal{B}$ calculates $h_i \leftarrow u_2 \cdot g^{a_i} \in \mathbb{G}$. Otherwise, $\mathcal{B}$ calculates $h_i \leftarrow g^{a_i} \in \mathbb{G}$. Then simulator $\mathcal{B}$ adds the tuple (w_i, h_i, a_i, c_i) to L_{H_1} and returns $H_1(w_i) = h_i$ to $\mathcal{A}$. Note that either way h_i is uniform in $\mathbb{G}$ and is independent of $\mathcal{A}$'s current view as required.
- $\mathcal{O}_{H_2}(t_i)$: To respond to H_2 queries, $\mathcal{B}$ maintains a hash list of pair (t_i, v_i) denoted as L_{H_2}. When $\mathcal{A}$ issues H_2 query with an element $t_i \in \mathbb{G}_T$, $\mathcal{B}$ proceeds as follows:
  - If t_i has been queried before, $\mathcal{B}$ retrieves $H_2(t_i)$ from the L_{H_2} with respect to t_i. Eventually, it returns $H_2(t_i)$.
  - Otherwise, $\mathcal{B}$ picks a new random value $v_i \in \{0, 1\}^{\log p}$ for a new t_i. Then it responds to $\mathcal{A}$ by setting $H_2(t_i) = v_i$, and adds the pair (t_i, v_i) to L_{H_2}.
- $\mathcal{O}_{SK}(uid)$: $\mathcal{B}$ initializes an empty list L_u. Given a user identity uid, the simulator $\mathcal{B}$ proceeds as follows:
  - If the user uid has been queried before, $\mathcal{B}$ retrieves δ_{uid} with respect to uid from list L_u.
  - Otherwise, $\mathcal{B}$ selects a random number $\delta_{uid} \in \mathbb{Z}_p$, and then adds (uid, δ_{uid}) to list L_u and returns δ_{uid}.
- $\mathcal{O}_{T_w}(uid, w_i)$: When $\mathcal{A}$ issues a trapdoor query with keyword w_i and user uid, $\mathcal{B}$ responds as follows:
  - $\mathcal{B}$ queries $\mathcal{O}_{H_1}$ to obtain an $h_i \in \mathbb{G}$ such that $H_1(w_i) = h_i$. Let (w_i, h_i, a_i, c_i) be the corresponding tuple in L_{H_1}. If $c_i = 0$ then $\mathcal{B}$ reports failure and terminates.
  - Otherwise $c_i = 1$, so $h_i = g^{a_i} \in \mathbb{G}$. $\mathcal{B}$ queries $\mathcal{O}_{SK}(uid)$ to obtain δ_{uid} with respect to the user uid, then defines $T_i = u_1^{a_i/\delta_{uid}} = H_1(w_i)^{\tilde{\alpha}/\delta_{uid}}$. we can see that T_i is the correct simulation of the trapdoor

TABLE 2. Comparisons of computation complexity.

Operation	Hur's [4]	Yang's [6]	Our scheme
System setup	$P + 2E$	$P + (2n_a + 3)E + E_T$	$P + (n_a + 2)E + E_T$
Key generation	$(3n_{a,u} + 2)E$	$(2n_{a,u} + 3)E$	$(n_{a,u} + 4)E$
Encryption	$E_T + (2l + 1)E$	$E_T + (2l + 4)E$	$E_T + (3l + 1)E$
Decryption by CS	–	–	$(2n_{a,u} + 1)P + 2n_{a,u}E_T$
Decryption by user	$(2n_{a,u} + 1)P + n_{a,u}E + \log l \cdot E_T$	$(2n_{a,u} + 1)P + n_{a,u}E_T$	P
Ciphertext update	$E_T + (l + 2)E$	$2n_xE$	n_xE
Secret key update	$n_{a,u}E$	$2n_xE$	n_xE

– : no such operation in the literature.

for the keyword w_i with the public parameters component (g, u_1). $\mathcal{B}$ returns T_i to $\mathcal{A}$.

Challenge. $\mathcal{A}$ submits a pair of challenge keywords w_0, w_1 of equal length. $\mathcal{B}$ produces the challenge index as follows:

$\mathcal{B}$ queries $\mathcal{O}_{H_1}$ twice to obtain $h_0, h_1 \in \mathbb{G}$ such that $H_1(w_0) = h_0$ and $H_1(w_1) = h_1$. For $i = 0, 1$, let (w_i, h_i, a_i, c_i) be the corresponding tuple in L_{H_1}. If both $c_0 = 1$ and $c_1 = 1$, then $\mathcal{B}$ reports failure and terminates.

We know that at least one of c_0, c_1 is equal to 0. $\mathcal{B}$ randomly picks a bit $b \in \{0, 1\}$ such that $c_b = 0$.

$\mathcal{B}$ responds with the challenge index $I_b = (u_3, J)$ for a random $J \in \{0, 1\}^{\log p}$. Observe that this challenge index implicitly defines $H_2(e(u_1^{\gamma}, H_1(w_b))) = J$, that is,

$$J = H_2(e(u_1^{\gamma}, H_1(w_b))) = H_2(e(u_1^{\gamma}, u_2\, g^{a_b})) = H_2(e(g, g)^{\bar{\alpha}\gamma(\beta + a_b)}).$$

Therefore, I_b is a valid index of w_b as required.

Phase 2. $\mathcal{A}$ can query the trapdoor oracle for keywords w_i same as Phase 1, where the only restriction is $w_i \neq w_0, w_1$.

Guess. $\mathcal{A}$ outputs his guess $b' \in \{0, 1\}$ of b. Then, $\mathcal{B}$ picks a random pair (t_i, v_i) from L_{H_2} and outputs $t_i / e(u_1, u_3)^{a_b}$ as his guess for $e(g, g)^{\bar{\alpha}\beta\gamma}$. This is because that $\mathcal{A}$ must have issued a query for either $H_2(e(u_1^{\gamma}, H_1(w_0)))$ or $H_2(e(u_1^{\gamma}, H_1(w_1)))$. Therefore, with probability 1/2 the list L_{H_2} contains a pair whose left hand side is $t_i = e(u_1^{\gamma}, H_1(w_b)) = e(g, g)^{\bar{\alpha}\gamma(\beta + a_b)}$. If $\mathcal{B}$ picks this pair (t_i, v_i) from the list L_{H_2} then $t_i / e(u_1, u_3)^{a_b} = e(g, g)^{\bar{\alpha}\beta\gamma}$ as required.

This finishes the description of the simulation process. Now we will analyze the probability that $\mathcal{B}$ correctly outputs $e(g, g)^{\bar{\alpha}\beta\gamma}$. During the simulation phase, $\mathcal{B}$ does not abort with probability at least $1/(eq_T)$. And in a real attack game, $\mathcal{A}$ issues a query for either $H_2(e(u_1^{\gamma}, H_1(w_0)))$ or $H_2(e(u_1^{\gamma}, H_1(w_1)))$ with probability at least $2\epsilon_2$. The detailed analyses of above two results are shown in [28]. That is, the value $e(u_1^{\gamma}, H_1(w_b)) = e(g, g^{\beta + a_b})^{\bar{\alpha}\gamma}$ will appear on the left hand side of some pair in the list L_{H_2} with probability at least ϵ_2. And $\mathcal{B}$ will choose the correct pair with probability at least $1/q_{H_2}$. Therefore, $\mathcal{B}$ will output $e(g, g)^{\bar{\alpha}\beta\gamma}$ with probability at least ϵ_2 / q_{H_2} in the case of that $\mathcal{B}$ completes the simulation without aborting. In general, the success probability of $\mathcal{B}$ is at least $\epsilon_2 / (eq_T q_{H_2})$ as required. □

VII. PERFORMANCE ANALYSIS

In this section, we give the performance analysis from the perspective of computation cost, storage overhead and communication cost, and make some comparison with schemes [4], [6]. The notations used in this section is introduced as follows.

P	the pairing operation
E	the group exponentiation in $\mathbb{G}$
M	the group multiplication in $\mathbb{G}$
E_T	the group exponentiation in $\mathbb{G}_T$
M_T	the group multiplication in $\mathbb{G}_T$
$\|p\|$	bit size of an element in $\mathbb{Z}_p$
$\|g\|$	bit size of an element in $\mathbb{G}$
$\|g_T\|$	bit size of an element in $\mathbb{G}_T$
n_a	the number of attributes in the system
n_u	the number of users in the system
n_x	the number of revoked attributes
$n_{a,u}$	the number of attributes a user possesses
l	the number of attributes embedded in a ciphertext

A. COMPUTATION COST

The numerical evaluation of the computation amount of the key operations in our RSABE scheme is shown in Table 2.

The simulation of our proposed RSABE scheme is conducted on a Windows system with an Intel Core i7 CPU at 3.60GHz and 8.00 GB RAM. The implementation is based on the Pairing-Based Cryptography (PBC) library [59]. The type A elliptic curve of 160-bit group order is adopted to provide groups in which a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is defined. The base field size is 512-bit and the size of the group elements is set to be 1024-bit. The curve provides 1024-bit discrete log security. The computation cost is evaluated in terms of system setup, key generation, encryption, decryption, ciphertext update and secret key update as shown in Fig. 3.

Fig. 3(a), Fig. 3(b) and Fig. 3(c) describe the time cost of setup, key generation and encryption respectively. It is clear that their time cost linearly increase with the number of attributes in universe, secret keys and ciphertexts. As shown in Fig. 3(d), the time cost of decryption at user side is bounded by a small constant, since most decryption task is outsourced to CS. In the revocation phase, AA delegate update tasks

TABLE 3. Comparisons of storage overhead.

Entity	Hur's [4]	Yang's [6]	Our scheme
Authority	$2\|p\|$	$(n_a+4)\|p\|$	$n_a+(2+n_a)\|p\|$
Owner	$2\|g\|+\|g_T\|$	$(2+n_a)\|g\|+\|g_T\|$	$(3+n_a)\|g\|+\|g_T\|$
Sever	$(3l+3)\|g\|+2\|g_T\|+(l\cdot n_u\cdot\|p\|)/2$	$(3l+1)\|g\|+\|g_T\|$	$(2l+n_{a,u}+4)\|g\|+\|g_T\|+2\|p\|$
user	$(2n_{a,u}+1)\|g\|+\log(n_u+1)\|p\|$	$(n_{a,u}+2)\|g\|$	$\|g\|+\|p\|$

TABLE 4. Comparisons of communication cost.

Entity	Hur's [4]	Yang's [6]	Our scheme
Owner & Authority	$2\|g\|+\|g_T\|$	$(2n_a+4)\|g\|+\|g_T\|$	$(n_a+3)\|g\|+\|g_T\|$
Owner & Sever	$2l\cdot\|g\|+(l+1)\|g_T\|$	$(3l+1)\|g\|+\|g_T\|$	$(2l+2)\|g\|+\|g_T\|+\|p\|$
User & Authority	$(2n_{a,u}+1)\|g\|$	$(n_{a,u}+4)\|g\|$	$\|g\|+\|p\|$
User & Sever	$(2l+1)\|g\|+\|g_T\|+(l\cdot n_u/2+\log(n_u+1))\|p\|$	$(3l+1)\|g\|+\|g_T\|$	$2\|g\|+2\|g_T\|$

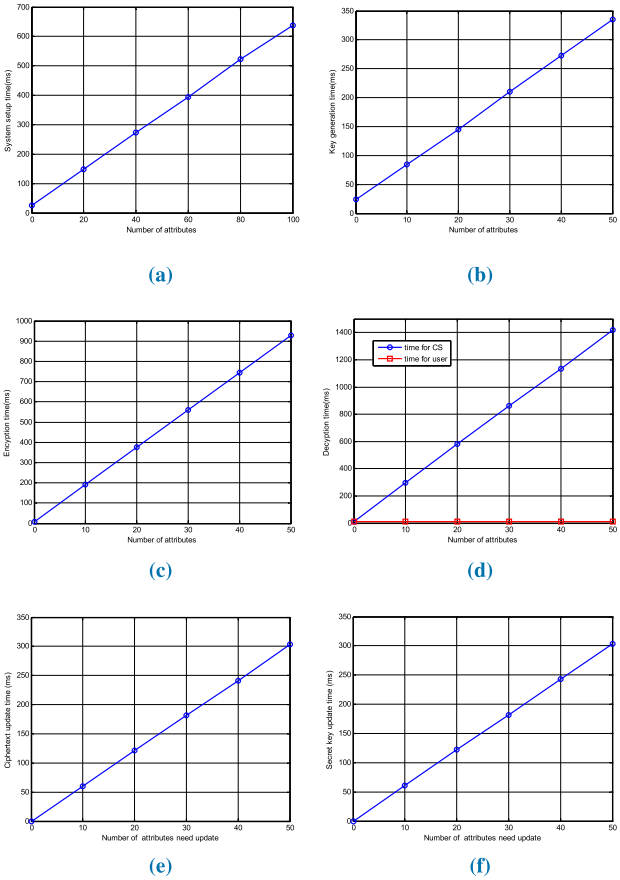


FIGURE 3. Performance evaluation on RSABE (a) System setup (b) Key generation (c) Encryption (d) Decryption (e) Ciphertext update (f) Secret key update.

of ciphertexts and secret keys to CS, Fig. 3(e) and Fig. 3(f) show that the time cost is linear with the number of revoked attributes.

B. STORAGE OVERHEAD

The storage overhead is one of the important factors that needs to be considered in mobile cloud storage. comparisons of the storage overhead on each entity between our RSABE scheme and schemes [4], [6] are shown in Table 3.

Note that, comparing with the schemes [4], [6], the storage overheads of our solution for the AA and owner are relatively increased. This is due to that we use the public attribute keys. Although the storage overhead on the CS is remarkably increased, this will not produce much effect since we know that the storage ability of CS is so strong. In comparison, we add the storage of the secret key component of users and the index over keywords. However, in this paper, it should be noted that the storage load on users is limited in a small constant, for the reason that the users only store secret key component SK_1. This feature is really suitable for the mobile environment.

C. COMMUNICATION COST

In Table 4, we compare the communication cost on each entity of our RSABE scheme with that of the schemes in [4] and [6].

Observe that, communication cost of owner is relatively increased as we add keywords index. However, we greatly reduce the user's communication costs. Especially, the communication cost between user and the authority, cloud server respectively is independent to the number of attributes $n_{a,uid}$ that the user *uid* owns.

VIII. CONCLUSION

In this article, we have presented an attribute-based encryption scheme for mobile storage system supporting efficient attribute revocation, attribute grant and keyword search. Specially, our solution greatly enhances the computational efficiency of the client by means of outsourced decryption technology. In revocation phase, the CS is delegated by the AA to execute a series of update operation. And the secret key component user holds does not need to update during the revocation. Furthermore, our scheme is proven IND-sCP-CPA secure and IND-CKA secure. Our RSABE

scheme can be applied to many mobile cloud storage systems, such as electronic health record system. A further direction is to consider the situation of multi-authority which is more accordant with practical circumstances.

REFERENCES

[1] N. Fernando, S. W. Loke, and W. Rahayu, "Mobile cloud computing: A survey," *Future Generat. Comput. Syst.*, vol. 29, no. 1, pp. 84–106, 2013.
[2] J. Li, H. Yan, and Y. Zhang, "Certificateless public integrity checking of group shared data on cloud storage," *IEEE Trans. Services Comput.*, to be published, doi: 10.1109/TSC.2018.2789893.
[3] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Advances in Cryptology—EUROCRYPT*, R. Cramer, Ed. Berlin, Germany: Springer, 2005, pp. 457–473.
[4] J. Hur and D. K. Noh, "Attribute-based access control with efficient revocation in data outsourcing systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 7, pp. 1214–1221, Jul. 2011.
[5] L. Zu, Z. Liu, and J. Li, "New ciphertext-policy attribute-based encryption with efficient revocation," in *Proc. IEEE Int. Conf. Comput. Inf. Technol. (CIT)*, Sep. 2014, pp. 281–287.
[6] K. Yang and X. Jia, "ABAC: Attribute-based access control," in *Security for Cloud Storage Systems*. New York, NY, USA: Springer, 2014, pp. 39–58.
[7] T. Naruse, M. Mohri, and Y. Shiraishi, "Provably secure attribute-based encryption with attribute revocation and grant function using proxy re-encryption and attribute key for updating," *Human-Centric Comput. Inf. Sci.*, vol. 5, no. 1, p. 8, 2015.
[8] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symp. Secur. Privacy (S&P)*, May 2000, pp. 44–55.
[9] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *Public Key Cryptography—PKC*, D. Catalano, N. Fazio, R. Gennaro, and A. Nicolosi, Eds. Berlin, Germany: Springer, 2011, pp. 53–70.
[10] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Advances in Cryptology—CRYPTO*, G. R. Blakley and D. Chaum, Eds. Berlin, Germany: Springer, 1985, pp. 47–53.
[11] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. 13th ACM Conf. Comput. Commun. Secur. (CCS)*. New York, NY, USA: ACM, 2006, pp. 89–98.
[12] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE Symp. Secur. Privacy (S&P)*. Washington, DC, USA: IEEE Computer Society, May 2007, pp. 321–334.
[13] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in *Proc. 29th IEEE INFOCOM*. Piscataway, NJ, USA: IEEE Press, Mar. 2010, pp. 534–542.
[14] M. Pirretti, P. Traynor, P. McDaniel, and B. Waters, "Secure attribute-based systems," in *Proc. 13th ACM Conf. Comput. Commun. Secur. (CCS)*. New York, NY, USA: ACM, 2006, pp. 99–112.
[15] A. Boldyreva, V. Goyal, and V. Kumar, "Identity-based encryption with efficient revocation," in *Proc. 15th ACM Conf. Comput. Commun. Secur. (CCS)*. New York, NY, USA: ACM, 2008, pp. 417–426.
[16] H. Wang, Z. Zheng, L. Wu, and P. Li, "New directly revocable attribute-based encryption scheme and its application in cloud storage environment," *Cluster Comput.*, vol. 20, no. 3, pp. 2385–2392, Sep. 2017, doi: 10.1007/s10586-016-0701-7.
[17] H. Wang, D. He, J. Shen, Z. Zheng, X. Yang, and M. H. Au, "Fuzzy matching and direct revocation: A new CP-ABE scheme from multilinear maps," *Soft Comput.*, vol. 22, no. 7, pp. 2267–2274, Apr. 2018, doi: 10.1007/s00500-017-2488-8.
[18] W. Sun, S. Yu, W. Lou, Y. T. Hou, and H. Li, "Protecting your right: Verifiable attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 4, pp. 1187–1198, Apr. 2016.
[19] Z. Lv, J. Chi, M. Zhang, and D. Feng, "Efficiently attribute-based access control for mobile cloud storage system," in *Proc. 13th IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun. (TrustCom)*, Sep. 2014, pp. 292–299.
[20] S. Yu, C. Wang, K. Ren, and W. Lou, "Attribute based data sharing with attribute revocation," in *Proc. 5th ACM Symp. Inf., Comput. Commun. Secur. (ASIACCS)*. New York, NY, USA: ACM, 2010, pp. 261–270.
[21] L. Ibraimi, M. Petkovic, S. Nikova, P. Hartel, and W. Jonker, "Mediated ciphertext-policy attribute-based encryption and its application," in *Information Security Applications*, H. Y. Youm and M. Yung, Eds. Berlin, Germany: Springer, 2009, pp. 309–323.
[22] X. Xie, H. Ma, J. Li, and X. Chen, "New ciphertext-policy attribute-based access control with efficient revocation," in *Information and Communication Technology*, K. Mustofa, E. J. Neuhold, A. M. Tjoa, E. Weippl, and I. You, Eds. Berlin, Germany: Springer, 2013, pp. 373–382.
[23] J. Li, W. Yao, Y. Zhang, H. Qian, and J. Han, "Flexible and fine-grained attribute-based data storage in cloud computing," *IEEE Trans. Services Comput.*, vol. 10, no. 5, pp. 785–796, Sep./Oct. 2017.
[24] J. Li, W. Yao, J. Han, Y. Zhang, and J. Shen, "User collusion avoidance CP-ABE with efficient attribute revocation for cloud storage," *IEEE Syst. J.*, vol. 12, no. 2, pp. 1767–1777, Jun. 2018.
[25] H. Qian, J. Li, Y. Zhang, and J. Han, "Privacy-preserving personal health record using multi-authority attribute-based encryption with revocation," *Int. J. Inf. Secur.*, vol. 14, no. 6, pp. 487–497, Nov. 2015, doi: 10.1007/s10207-014-0270-9.
[26] P. Zhang, Z. Chen, K. Liang, S. Wang, and T. Wang, "A cloud-based access control scheme with user revocation and attribute update," in *Information Security and Privacy*, J. K. Liu and R. Steinfeld, Eds. Cham, Switzerland: Springer, 2016, pp. 525–540, doi: 10.1007/978-3-319-40253-6_32.
[27] Z. Liu, S. Duan, P. Zhou, and B. Wang, "Traceable-then-revocable ciphertext-policy attribute-based encryption scheme," *Future Gener. Comput. Syst.*, to be published. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X17320964, doi: 10.1016/j.future.2017.09.045.
[28] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Advances in Cryptology—EUROCRYPT*, C. Cachin and J. L. Camenisch, Eds. Berlin, Germany: Springer, 2004, pp. 506–522.
[29] P. Golle, J. Staddon, and B. Waters, "Secure conjunctive keyword search over encrypted data," in *Applied Cryptography and Network Security*, M. Jakobsson, M. Yung, and J. Zhou, Eds. Berlin, Germany: Springer, 2004, pp. 31–45.
[30] Y. H. Hwang and P. J. Lee, "Public key encryption with conjunctive keyword search and its extension to a multi-user system," in *Pairing-Based Cryptography—PAIRING*, T. Takagi, T. Okamoto, E. Okamoto, and T. Okamoto, Eds. Berlin, Germany: Springer, 2007, pp. 2–22.
[31] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," in *Theory of Cryptography*, S. P. Vadhan, Ed. Berlin, Germany: Springer, 2007, pp. 535–554.
[32] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 1, pp. 222–233, Jan. 2014.
[33] S. Ji, G. Li, C. Li, and J. Feng, "Efficient interactive fuzzy keyword search," in *Proc. 18th Int. Conf. World Wide Web (WWW)*. New York, NY, USA: ACM, 2009, pp. 371–380
[34] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. Lou, "Fuzzy keyword search over encrypted data in cloud computing," in *Proc. 29th IEEE INFOCOM*, Mar. 2010, pp. 1–5.
[35] S. Sedghi, P. van Liesdonk, S. Nikova, P. Hartel, and W. Jonker, "Searching keywords with wildcards on encrypted data," in *Security and Cryptography for Networks*, J. A. Garay and R. De Prisco, Eds. Berlin, Germany: Springer, 2010, pp. 138–153.
[36] C. Wang, K. Ren, S. Yu, and K. M. R. Urs, "Achieving usable and privacy-assured similarity search over outsourced cloud data," in *Proc. 31st IEEE INFOCOM*, Mar. 2012, pp. 451–459.
[37] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," in *Proc. 13th ACM Conf. Comput. Commun. Secur. (CCS)*. New York, NY, USA: ACM, 2006, pp. 79–88.
[38] F. Bao, R. H. Deng, X. Ding, and Y. Yang, "Private query on encrypted data in multi-user settings," in *Information Security Practice and Experience*, L. Chen, Y. Mu, and W. Susilo, Eds. Berlin, Germany: Springer, 2008, pp. 71–85.
[39] C. Dong, G. Russello, and N. Dulay, "Shared and searchable encrypted data for untrusted servers," in *Data and Applications Security XXII*, V. Atluri, Ed. Berlin, Germany: Springer, 2008, pp. 127–143.
[40] J. Li, Y. Shi, and Y. Zhang, "Searchable ciphertext-policy attribute-based encryption with revocation in cloud storage," *Int. J. Commun. Syst.*, vol. 30, no. 1, p. e2942, 2015.

[41] J. Li, J. Li, X. Chen, C. Jia, and Z. Liu, "Efficient keyword search over encrypted data with fine-grained access control in hybrid cloud," in *Network and System Security*, L. Xu, E. Bertino, and Y. Mu, Eds. Berlin, Germany: Springer, 2012, pp. 490–502.

[42] Q. Wang, Y. Zhu, and X. Luo, "Multi-user searchable encryption with fine-grained access control without key sharing," in *Proc. Int. Conf. Cloud Comput. Big Data (CCBD)*, Dec. 2014, pp. 119–125.

[43] F. Zhou, Y. Li, A. X. Liu, M. Lin, and Z. Xu, "Integrity preserving multi-keyword searchable encryption for cloud computing," in *Provable Security*, L. Chen and J. Han, Eds. Cham, Switzerland: Springer, 2016, pp. 153–172, doi: 10.1007/978-3-319-47422-9_9.

[44] Y. Miao, J. Ma, X. Liu, F. Wei, Z. Liu, and X. A. Wang, "m^2-ABKS: Attribute-based multi-keyword search over encrypted personal health records in multi-owner setting," *J. Med. Syst.*, vol. 40, no. 11, p. 246, Nov. 2016, doi: 10.1007/s10916-016-0617-z.

[45] J. Li, X. Lin, Y. Zhang, and J. Han, "KSF-OABE: Outsourced attribute-based encryption with keyword search function for cloud storage," *IEEE Trans. Services Comput.*, vol. 10, no. 5, pp. 715–725, Sep./Oct. 2017.

[46] Y. Li, F. Zhou, Y. Qin, M. Lin, and Z. Xu, "Integrity-verifiable conjunctive keyword searchable encryption in cloud storage," *Int. J. Inf. Secur.*, vol. 17, pp. 1–20, Nov. 2017, doi: 10.1007/s10207-017-0394-9.

[47] M. Green, S. Hohenberger, and B. Waters, "Outsourcing the decryption of abe ciphertexts," in *Proc. 20th USENIX Conf. Secur. (SEC)*. Berkeley, CA, USA: USENIX Association, 2011, p. 34.

[48] Z. Zhou and D. Huang, "Efficient and secure data storage operations for mobile cloud computing," in *Proc. 8th Int. Conf. Netw. Service Manage. (CNSM)*. Laxenburg, Austria: International Federation for Information Processing, Oct. 2012, pp. 37–45.

[49] J. Lai, R. H. Deng, C. Guan, and J. Weng, "Attribute-based encryption with verifiable outsourced decryption," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 8, pp. 1343–1354, Aug. 2013.

[50] J. Li, X. Chen, J. Li, C. Jia, J. Ma, and W. Lou, "Fine-grained access control system based on outsourced attribute-based encryption," in *Computer Security—ESORICS*, J. Crampton, S. Jajodia, and K. Mayes, Eds. Berlin, Germany: Springer, 2013, pp. 592–609.

[51] J. Li, X. Huang, J. Li, X. Chen, and Y. Xiang, "Securely outsourcing attribute-based encryption with checkability," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 8, pp. 2201–2210, Aug. 2014.

[52] J. Li, J. Li, X. Chen, C. Jia, and W. Lou, "Identity-based encryption with outsourced revocation in cloud computing," *IEEE Trans. Comput.*, vol. 64, no. 2, pp. 425–437, Feb. 2015.

[53] J. Li, Y. Zhang, X. Chen, and Y. Xiang, "Secure attribute-based data sharing for resource-limited users in cloud computing," *Comput. Secur.*, vol. 72, pp. 1–12, Jan. 2018.

[54] J. Li, Y. Wang, Y. Zhang, and J. Han, "Full verifiability for outsourced decryption in attribute based encryption," *IEEE Trans. Services Comput.*, to be published, doi: 10.1109/TSC.2017.2710190.

[55] K. Zhang, J. Ma, J. Liu, and H. Li, "Adaptively secure multi-authority attribute-based encryption with verifiable outsourced decryption," *Sci. China Inf. Sci.*, vol. 59, no. 9, p. 99105, Aug. 2016, doi: 10.1007/s11432-016-0012-9.

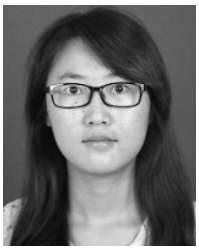
[56] H. Wang, D. He, J. Shen, Z. Zheng, C. Zhao, and M. Zhao, "Verifiable outsourced ciphertext-policy attribute-based encryption in cloud computing," *Soft Comput.*, vol. 21, no. 24, pp. 7325–7335, Dec. 2017, doi: 10.1007/s00500-016-2271-2.

[57] K. Zhang *et al.*, "Practical and efficient attribute-based encryption with constant-size ciphertexts in outsourced verifiable computation," in *Proc. 11th ACM Asia Conf. Comput. Commun. Secur. (ASIA CCS)*. New York, NY, USA: ACM, 2016, pp. 269–279, doi: 10.1145/2897845.2897858.

[58] M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, and K. Fu, "Plutus: Scalable secure file sharing on untrusted storage," in *Proc. 2nd USENIX Conf. File Storage Technol. (FAST)*. Berkeley, CA, USA: USENIX Association, 2003, pp. 29–42.

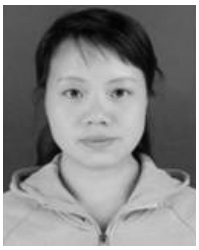
[59] B. Lynn. (2017). *The Pairing-Based Cryptography Library*. [Online]. Available: https://crypto.stanford.edu/pbc/

SHANGPING WANG received the B.S. degree in mathematics from the Xi'an University of Technology, Xi'an, China, in 1982, the M.S. degree in applied mathematics from Xi'an Jiaotong University, Xi'an, in 1989, and the Ph.D. degree in cryptology from Xidian University, Xi'an, in 2003. He is currently a Professor with the Xi'an University of Technology. His current research interests are cryptography and information security.



DUO ZHANG received the B.S. and M.S. degrees from the School of Science, Xi'an University of Technology, Xi'an, China, in 2014 and 2017, respectively, where she is currently pursuing the Ph.D. degree with the School of Automation and Information Engineering. Her research interests include information security and modern cryptography.

YALING ZHANG received the B.S. degree in computer science from Northwest University, Xi'an, China, in 1988, and the M.S. degree in computer science and the Ph.D. degree in mechanism electron engineering from the Xi'an University of Technology, Xi'an, in 2001 and 2008, respectively. She is currently a Professor with the Xi'an University of Technology. Her current research interests include cryptography and network security.



LIHUA LIU received the M.S. degree in computer science from Shaanxi Normal University in 2006. She is currently pursuing the Ph.D. degree with the School of Automation and Information Engineering, Xi'an University of Technology, Xi'an, China. She is also an Associate Professor with the Shaanxi University of Technology, Hanzhong, China. Her research interests include information security and modern cryptography.