IET Information Security

Research Article

Re-definable access control over outsourced data in cloud storage systems

ISSN 1751-8709
Received on 8th October 2017
Revised 16th July 2018
Accepted on 30th November 2018
E-First on 20th February 2019
doi: 10.1049/iet-ifs.2018.5365
www.ietdl.org

Zhigang Zhang[1,2] ✉, Chaowen Chang[1], Zhimin Guo[2], Peisheng Han[1]
[1]*Zhengzhou Information Science and Technology Institute, Henan, Zhengzhou 450000, People's Republic of China*
[2]*Electric Science Institute of State Grid He'nan Electric Power Company, Zhengzhou 450000, People's Republic of China*
✉ *E-mail: 15838150770@126.com*

Abstract: There is an increasing concern for data privacy when people outsource their data to remote cloud storage servers. To secure outsourced data, cloud users are suggested to employ cryptographic encryption to specify access policies such that only users meeting the policies can access the data. After the application of an encryption, however, users are difficult to modify their access policies since the policies were already formulated by the encryption. To address such problem, the authors propose a new approach referred to as re-definable access control (RDAC). The RDAC utilises identity-based encryption (IBE) and attribute-based encryption (ABE) to secure outsourced data and allows users to choose either to achieve access control according to their capability and requirements. Moreover, the RDAC allows users to change simple access policies into fine-grained access policies by converting IBE encrypted files into ABE encrypted ones, without leaking the underlying data. Surprisingly, the access policy conversion does not require the users to perform any costly computation, nor the storage servers to be disturbed. The authors prove the security of RDAC under a rigorous definition, and empirically show that the introduction of the conversion incurs almost no costs to the outsourcing and access procedures.

1 Introduction

Cloud computing has become a promising paradigm for individuals and organisations to save costs, improve efficiency and achieve flexibility. It provides scalable, ubiquitous and on-demand services, which frees people from the burden of local data management. Cloud storage has many outstanding advantages, such as considerable storage, flexible accessibility and convenient data retrieval. In cloud storage systems [1, 2], users can outsource their database to a remote storage server so that themselves and other authorised users can retrieve the data when necessary.

Outsourcing and retrieving data remotely delivers great conveniences to users and meanwhile, brings security concerns. The fact that users lose physical control of their data rises up at least two serious problems: how to protect the privacy and security of their data, and how cloud storage users can securely share their data with intended receivers. For the first question, a typical countermeasure is cryptographic encryption, such that all outsourced data are encrypted and unreadable to cloud servers. Compared to conventional public key infrastructure (PKI)-based encryption schemes [3], identity-based encryption (IBE) [4] provides more efficient generation of public keys and has been adopted in many conditional storage applications, such as body sensor network [5], enterprise data storage system [6] and internet of things [7]. Despite of its various potential applications, IBE has a major insufficiency that it requires an explicit identity of the recipient for each encryption, while a user in encryption might do not know who will request his data. Attribute-based encryption (ABE) [8] extends the IBE by providing a fine-grained access control mechanism by which users only need to specify access policies (not specific identities) on their data such that only the users meeting the policies can decrypt and read. This advantageous feature renders ABE extensively employed in many cloud-based environments [9–11].

The IBE and ABE, as two powerful cryptographic tools, provide proper encryption approaches to protect the privacy and security of cloud data. However, there is still a problem that how cloud data protected by one encryption system (e.g. IBE) can be shared with users belonging to another encryption system (e.g. ABE). This is inspired by the fact that in complicated cloud storage systems (e.g. hybrid clouds [12]), there may be not only one encryption system adopted to secure outsourced data; contrary, different organisations and individuals could employ appropriate encryption systems according to their demands. To be more clear, we consider the following scenario.

Motivating scenario: Company A and Company B allow its employees to outsource data to a cloud service provider. For data security purpose, Company A chooses a cryptographic system to secure outsourced data. The IBE would be an appropriate choice for Company A if it does not have a large number of employees and would like efficient and simple access control on outsourced data. On contrary, Company B has a certain number of employees and prefers to the ABE encryption system to realise more flexible access control. Company A signs contracts with Company B to share its outsourced data. Unfortunately, Company A's outsourced data are encrypted by IBE system and therefore are unable to be decrypted by users in a different cryptographic system, i.e. ABE. A trivial solution may be asking Company A to first decrypt its IBE encrypted files and then encrypt to Company B via ABE, which, however, would introduce great inconvenience and heavy computation tasks for the company.

A feasible solution may be a re-definable access control (RDAC) mechanism for Company A. In RDAC mechanism, Company A does not need to run decryption and then encryption; alternatively, it performs the RDAC procedure to directly convert the IBE encrypted files into ABE encrypted ones by specifying a re-defined access policy so that Company B's employees meeting the policy can read data. For instance, Company A can formulate a re-defined access policy 'Company B AND (Engineer OR Manager)' and run RDAC procedure such that only the engineers or the managers of Company B can access. In this way, a simple access policy which points out a specific receiver has been transformed ('re-defined') into a fine-grained access policy that describes attributes of a group of receivers. A fundamental security of RDAC is that only the data owner (Company A) and authorised users can access the data in plaintext. We note that there exists some schemes achieving the similar RDAC functionality, while most of them only support ciphertext conversion in the same encryption system [13], or do not provide fine-grained access control [14, 15], which renders them inappropriate for the cloud storages.

2 Our contributions

In this paper, we propose a new approach referred to as RDAC scheme to enable one to convert a file encrypted by an IBE access policy into a file encrypted by an ABE access policy without exposing or altering the underlying plaintext. In a RDAC system, users who seek for low computation burden and fast encryption can use IBE to encrypt their data, while those who want more flexible access control on data can apply ABE. Between these two separate cryptosystems, the RDAC provides the IBE users with a unique and important mechanism that allows them to directly convert their IBE ciphertexts into ABE ciphertext without performing ABE encryption. During the conversion, the underlying plaintext will not be exposed to any unauthorised users. Besides, the cloud storage servers are not required to take any costs in the re-definition. Specifically, our contributions are described as follows.

We present a framework of RDAC and give a rigorous security definition. Intuitively, there are encrypted files with IBE and ABE access control policies, respectively, in RDAC. When a user wants to convert his IBE encrypted file into an ABE encrypted file, the user just invokes an auxiliary computation server (ACS) to help to compute a re-definition key (RK). A proxy server then applies the RK to transform the IBE-encrypted file associated with the IBE access policy into an ABE encrypted file associated with the ABE access policy. In this access policy re-definition, the user takes almost no efforts, and neither the ACS nor the proxy knows the underlying data. To ensure data privacy, we also define a formal security against practical attacks in outsourced database storage systems.

We construct a RDAC scheme and prove its security under the provided definition. The most challenging problem in the construction is the form of the ciphertexts generated by different IBE and ABE access policies. We overcome this issue by finding two compatible IBE and ABE schemes sharing some parameters and providing a ciphertext conversion that barely needs users' participation. We then prove the security of RDAC scheme by assuming the security of the IBE and the ABE without random oracle.

To evaluate the performance of RDAC, we conduct thorough theoretical and experimental analyses. The results show that the normal encryption procedures of IBE and ABE have not been affected by the introduction of re-definition; accordingly, the normal decryption procedures of IBE and ABE are not affected either. Notably, the access policy re-definition takes the invoking user little time and does not disturb the storage servers since most computations are given to the ACS and the proxy. These features make RDAC a practical solution to re-define the access policy of outsourced data in cloud storage.

The rest of the paper is organised as follows. Section 3 reviews some related works. Section 4 presents the framework of RDAC and Section 5 gives some background knowledge about the construction. Section 6 proposes the concrete RDAC scheme. Section 7 proves the security of RDAC under a rigorous definition. Section 8 analyses the SEC scheme both theoretically and experimentally and Section 9 concludes the paper.

3 Related work

Protecting data privacy in outsourced database storage systems is a hot topic and has been extensively researched. Cryptographic encryption (classified into symmetric and asymmetric encryption) is a typical measure to achieve data privacy protection [16, 17]. The symmetric encryption such as stream cipher [18] and block cipher [19, 20] usually has fast processing speed since it requires only simple computations (e.g. XOR). The asymmetric encryption, i.e. public-key encryption, does not require the encryption key to be identical to the decryption key and allows the encryption one to be public, which overcomes the key transmission issue in symmetric encryption. Kao *et al.* [21] employed the traditional RSA public-key encryption to secure sensitive data in clouds. Lee [3] proposed a private key issuing protocol by combining PKI and IBE, and then an improved secure protection of data privacy is fulfilled. Deng *et al.* [6] applied the IBE into cloud computing such that only the users with correct identities can access the outsourced data. In wireless communication environments, e.g. body sensor network [5], internet of things [7], IBE is also leveraged to secure outsourced data.

The concept of ABE was first introduced by Sahai and Waters [8]. ABE allows one to encrypt a data with an access policy rather than a public key (in PKI-based system) or an identity (in IBE), then only the users meeting the policy can decrypt and read the data. Yu *et al.* [9] used ABE in cloud computing system to achieve the fine-grained access control on outsourced data. Alshehri *et al.* [11] also utilised ABE to protect sensitive health records in cloud. Yang and Jia [10] proposed a multi-authority ABE scheme to solve the issues caused by single authority. Secure multi-party computation (MPC) [22] enables multiple participants to cooperatively compute on their private inputs without revealing nothing but the results. In cloud environments, MPC is a useful technique to protect cloud users' privacy and many efforts have been made to improve the security or efficiency of MPC. Zhang *et al.* [23] focused on the verifiability of users inputs and outcomes and designed a series of MPC protocols for dot product, ranging and ranking. Recently, Gueron *et al.* [24] proposed a MPC protocol that can be proved secure under a weaker assumption. Yoshida and Obana [25] presented a non-interactive MPC against honest-but-curious adversaries (e.g. cloud service providers) and improved its efficiency by using offline-online methods.

Searchable encryption allows a cloud user to securely search over encrypted data outsourced to clouds through keywords and retrieve matching files. Considering typical and frequent searching behaviours, Li *et al.* [26] proposed a fuzzy keyword searchable encryption that enables users to search on encrypted data with tolerance of typos and format inconsistencies. Li *et al.* [27] presented an authorised private keyword search solution over encrypted data to solve the issue of search capability authorisation. Kuzu *et al.* [28] utilised the locality sensitive hashing technique to achieve a scalable scheme for similarity search over encrypted data. Hamlin *et al.* [29] revisited the multi-key searchable encryption to achieve a strengthened security notion and proposed a concrete construction that enables a user to perform searches on both his own files and the files shared by other users.

Fully homomorphic encryption (FHE) [30] is a kind of encryption that allows arbitrary computations on ciphertexts, generating an encrypted result which can be decrypted into the result of the operations as if they had been performed on the plaintext. Gentry proposed the first FHE construction by using ideal lattices. FHE is a promising measure to protect data privacy in clouds. Chatterjee and Sengupta [31] provided techniques to translate basic operators (like bitwise, arithmetic and relational operators) to handle complicated FHE computations on encrypted data on clouds. Cao *et al.* [32] proposed a method to accelerate FHE over large integers. Also, to improve the performance of FHE, Wang *et al.* [33] presented a novel precomputation technique and used to optimise Gentry and Halevi FHE scheme [34]. Doröz *et al.* [35] recently constructed a FHE scheme proved to be secure based on a new hard problem they proposed.

Proxy re-encryption (PRE) is a useful cryptographic tool to achieve encryption conversion. In PRE, a proxy can transform a ciphertext for Alice into a ciphertext for Bob, without seeing the underlying plaintext. Ateniese *et al.* [13] presented several efficient PRE schemes which do not require Alice to store additional secrets, nor interact with Bob. Libert and Vergnaud [36] improved Ateniese *et al.*'s PRE schemes to achieve a stronger security, i.e. chosen-ciphertext security. Identity-based PRE (IBPRE) was first formulated by Green and Ateniese [37], where the public keys can be arbitrary strings. Chu and Tzeng [38] then proposed an IBPRE scheme proved to be secure without random oracle. By combining ABE and PRE, Liang *et al.* [39] extended PRE to attribute-based PRE (ABPRE). Subsequently, Liang *et al.* [40] enhanced the security of the ABPRE scheme of [39] while the security is only proved in random oracle model, i.e. a weaker security definition. Derler *et al.* [41] employed the hierarchical IBE to design a forward-secret PRE construction where the proxy periodically evolves the re-encryption keys. Table 1 compares our scheme with related works [37–40] that also achieve the access control re-definition mechanism, in terms of the size of RK, the size of converted file, the number of bilinear pairings required in the converted file access, the granularity of access control (fine-grained or not) and the consumption of the users (clients) in file conversion. More details are described in Section 8.

Mizuno and Doi [15] presented a hybrid ABPRE that can convert an ABE ciphertext into an IBE ciphertext, which is opposite re-encryption direction compared to the proposed RDAC. In Mizuno and Doi's re-encryption scheme, the size of public key, secret key and ABE ciphertext are all linear with the number of total attributes in the system, while in the RDAC the public key has a constant size, and the size of ABE secret key and ciphertext are linear with the related attributes rather than the total ones. This advantageous feature renders the proposed RDAC a more efficient and practical solution.

All the aforementioned works mainly focus on protecting data privacy using different encryption systems, e.g. PKI, IBE, ABE, to cater for specific applications. Although there are some PRE schemes that achieve the encryption conversion similar with our purpose, they either provide conversion in the same cryptosystem, or are proposed as improvements in terms of security and efficiency. However, the proposed RDAC on the one hand provides both IBE and ABE to protect data privacy, and importantly, on the other hand, offers an efficient way for users to convert their IBE encrypted files associated with simple access policies (i.e. identities) into ABE encrypted ones associated with fine-grained access policies describing what attributes authorised visitors should have.

4 System model and security

4.1 System model

Two cryptosystems, i.e. IBE and ABE, are considered in RDAC to protect data privacy and provide two kinds of access control mechanisms. The IBE provides a simple-yet-efficient access control on outsourced data, while the ABE achieves complicated-yet-versatile access control. As depicted in Fig. 1, our RDAC system consists of six entities as follows:

- Trusted authority (TA): a party trusted by others to publish the system parameters and issue access credentials;
- Data owners: users who encrypt their data with an IBE access policy and outsource them to cloud storage provider (CSP);
- Data consumers: users who can download and decrypt the outsourced files if they have the correct access credentials issued by TA;
- CSP: a storage server which stores files encrypted by IBE or ABE in a cloud storage system.
- ACS: a server which responds to the requests from data owners and helps to compute conversion keys;
- Proxy: a server which transforms the files encrypted with IBE access policies into the files encrypted with ABE access policies by applying the RKs given by the data owners.

The IBE can be employed for the users seeking for low computation and simple access control, and the ABE can be used for the users who want to achieve flexible encryption. The TA is responsible for generating access credentials for both IBE and ABE users. When a data owner (i.e. an IBE user) decides to convert his/her IBE encrypted file into an ABE encrypted file associated with a fine-grained access policy, the ACS can provide computation resources to generate a blinded RK, given that the IBE users very likely possess limited computation capability. The user then can compute a real RK from the blinded one at a very low cost. The IBE user gives this key to the proxy so that the proxy can directly transform the encrypted files, which results in an ABE encrypted file associated with the access policy specified by the IBE user. Then, by the access policy realised on the ABE encrypted file, the fine-grained access control on outsourced data is achieved. In the re-definition phase, the ACS does not know the real RK and the proxy does not know any secret of the data owner nor the data in plaintext. We note that the IBE encryption system and the ABE encryption one can also have data consumers and data owners, respectively. This means that there are data consumers decrypting IBE encrypted files and data owners generating ABE encrypted files in our RDAC system. We omit such data consumers and data owners from the figure to concentrate on the access control re-definition mechanism.

4.2 Security model

We assume that the CSP is honest-but-curious in the sense that it is curious about the content of the file but still honestly perform the assigned tasks. Unauthorised users or intruders may try to access the data outsourced to the CSP. A typical way to protect data privacy is to encrypt the data before outsourcing. The proxy and the ACS also may be compromised by unauthorised users or attackers to get the RKs. If the attackers obtain the RK that can transform a file encrypted under an IBE access policy into a file encrypted under an ABE access policy, and additionally have the access credential that can decrypt the ABE encrypted files, then they can recover the data encrypted in the original file, which severely damages the data privacy. Therefore, to ensure data security, we require that anyone without the correct access credentials issued by the TA cannot access the data stored the CSP. In addition, to secure the file conversion, we also require that even one compromises the proxy server and the ACS, it still cannot know the data encrypted in original files if having no correct access credential. More formal security definition for RDAC will be described in Section 7.

5 Preliminaries

We review some basic concepts and technologies underlying the RDAC construction.

5.1 Bilinear map

Let $\mathbb{G}$ and $\mathbb{G}_T$ be two cyclic groups of order p for some large prime p. Let g be a generator of $\mathbb{G}$. A map e is said to be bilinear if (i): $e\left(u^a, h^b\right) = e(u, h)^{ab}$ for all $u, h \in \mathbb{G}$ and all $a, b \in \mathbb{Z}_p$; and (ii): $e(g, g) \neq 1$.

We say that $\mathbb{G}$ is a bilinear group if there exists an efficiently computable bilinear map $e: \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$.

The bilinear map is widely used in identity-based and attribute-based cryptographic systems to achieve some cancellations of random factors in decryption.

5.2 Access structure and linear secret sharing scheme (LSSS) [42]

We now introduce the definition of access structure and the LSSS.

Definition 1: Let $\{P_1, P_2, \ldots, P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\{P_1, P_2, \ldots, P_n\}}$ is monotone if for $\forall B, C$, we have that $C \in \mathbb{A}$ holds if $B \in \mathbb{A}$ and $B \subseteq C$. An access structure (respectively, monotone access structure) is a collection (respectively, monotone collection) $\mathbb{A}$ of non-empty subsets of $\{P_1, P_2, \ldots, P_n\}$, i.e. $\mathbb{A} \subseteq 2^{\{P_1, P_2, \ldots, P_n\}} \setminus \{\varnothing\}$. The sets in $\mathbb{A}$ are called the authorised sets, and the sets not in $\mathbb{A}$ are called the unauthorised sets.

In the ABE systems, the role of parties is played by the attributes and the access structure is a collection of sets of attributes. Then, in the cloud storage system employing ABE, stored files are associated with access structures specifying which attributes that the authorised visitors should possess. For instance, a data owner specifies an access structure $\{\{\text{Male AND } 40\} \text{OR } \{\text{Professor ANDComputer Science}\}\}$ over his/her file stored on cloud. Then only the data consumers associated with the attribute set including the authorised set {Male, 40} or {Professor, Computer Science} can access the file.

In our construction, we concentrate on the monotone access structures, which means that as a data consumer in an ABE system acquiring more attributes, he would not lose the potential access rights. However, one can still realise general access structures by

Table 1 Comparison with related works

Proposal	Re-definition key size	Converted file size	Pairings in converted file access	Fine-grained access control	Clients' low consumption
[38]	3	2	2	×	✓
[37]	5	5	3	×	✓
[39]	$2\|S\| + l + 4$	$l + 3$	$3\|S^*\| + 1$	✓	×
[40]	$\|S\| + 2l + 6$	$2l + 4$	$2\|S^*\| + 3$	✓	×
ours	$3l + 4$	$3l + 4$	$3\|S^*\| + 2$	✓	✓

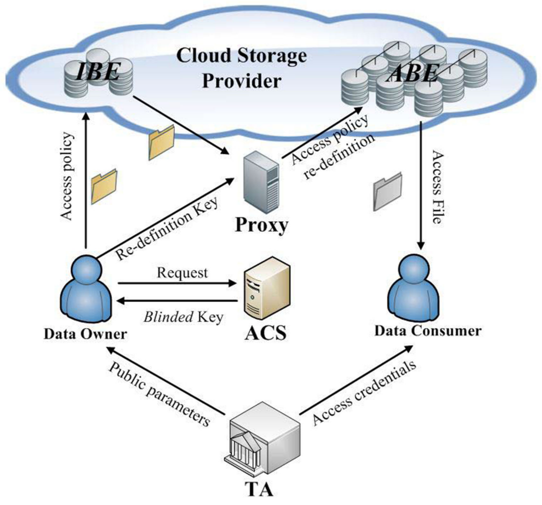

Fig. 1 *System architecture*

taking the negation of an attribute as a single attribute, at the cost of doubling the number of attributes in the system.

As previous work, we employ the LSSS to realise access structure. Informally, an LSSS is an $l \times n$ secret-sharing matrix $\boldsymbol{M}$ whose rows $\{M_i\}$ are mapped to attributes through a function ρ. When we consider a vector $\boldsymbol{v} = (s, y_2, \dots, y_n)$, where $s \in \mathbb{Z}_p$ is the secret to be shared and $y_2, \dots, y_n \in \mathbb{Z}_p$ are randomly picked, $\boldsymbol{Mv}$ is the l shares of the secret s and the value $\lambda_i = M_i \cdot \boldsymbol{v}$ (inner product) is the share belonging to attribute $\rho(i)$. If a user's attribute set S satisfies an access structure realised by an LSSS $(\boldsymbol{M}, \rho)$, then the rows of M labeled by the attributes $\{\rho(i)\}$ in S have the *linear reconstruction* property, which means that there exist constants $\{\omega_i \in \mathbb{Z}_p\}$ such that for the valid shares $\{\lambda_i\}$ of the attributes $\{\rho(i)\}$, we have that $\sum_i \omega_i \lambda_i = s$. Formal definition of LSSS can be found in [42].

5.3 IBE and ABE

IBE [4] is a cryptographic primitive that allows arbitrary string to serve as a user's public key. It removes the public-key certificates which are usually required in traditional public-key cryptosystem and hence allows users not to query public key for each encryption. Generally, an IBE system consists of four algorithms: setup, key generation, encryption and decryption. For each user associated with a recognisable identity ID (e.g. name, email address etc.) serving as the user public key, the key generation algorithm run by a trusted party creates a decryption key for ID. The encryption algorithm generates a ciphertext for an identity and the decryption algorithm succeeds if the identity of the decryption key is equal to that associated with the ciphertext.

ABE [8] is a more flexible cryptosystem than IBE. It is very suitable for the applications where the data owners cannot exactly know who will request to access their data prior to their encrypting. With ABE, the data owners could specify access structures with regard to some attributes such that only the data consumers with the attributes meeting the access policy can access their data. Formally, ABE consists of the similar algorithms of IBE with some difference. Specifically, the encryption algorithm of (ciphertext-policy) ABE (CP-ABE) generates ciphertext with access structures over attributes; the key generation algorithm creates decryption keys using sets of attributes. Then the decryption algorithm judges whether the attribute set of the decryption key satisfies the access structure of the ciphertext and if so, the decryption algorithm successfully works.

6 Proposal

In this section, we propose our RDAC scheme. Before describing the scheme, we first introduce the main idea driving the construction.

6.1 Basic idea

Our RDAC scheme allows data owners to convert their files encrypted by IBE access control policies (i.e. identities) into files encrypted by ABE access control policies (i.e. sets of attributes), without leaking the underlying data nor the secrets of the data owners. The normal files are encrypted by the IBE cryptosystem and accordingly, the data consumers are given by IBE access credentials (IBE decryption keys). In contrast, the converted files are encrypted by the ABE cryptosystem and corresponding data owners have ABE access credentials.

The main challenge in the construction is the non-uniform form of the IBE-encrypted and ABE-encrypted files. Specifically, the normal files are created by the encryption algorithm of the IBE and thus are associated with single identities; however, the converted files are associated with attributes due to the ABE encryption. Hence, it is very difficult to first 'extract' the identities embedded into the IBE encrypted file and then 'replace' them with specified attributes to form ABE encrypted files, provided that the data cannot be revealed.

To overcome the above challenges, we seek for two compatible IBE and ABE schemes which share some common parameters. This similarity in parameters enables us to encrypt data with the two cryptosystems and more importantly, to achieve the encrypted file conversion from one into another. The CP-ABE scheme in [43] allows any string to sever as an attribute, just as the IBE permitting any string to serve as an identity. This property greatly saves the size of the system parameters and makes the encryption more efficient. To retain this enjoyable property in our RDAC system, we need an IBE scheme sharing some system parameters with the CP-ABE scheme. We find that we can directly apply the IBE scheme in [4] with a simple modification. That is, to make the IBE compatible with the CP-ABE scheme, we drop one element from the system parameters of the IBE scheme in [4] and obtain an IBE scheme with little difference. The resulting IBE scheme can be proved to achieve the same security as the original one by simply following its proof techniques.

6.2 Construction

We now describe the RDAC scheme which protects data privacy by two different cryptosystems (i.e. IBE and ABE) and provides an efficient method to securely transform the IBE-encrypted files into ABE-encrypted files. The RDAC system consists of the following five procedures.

6.2.1 Setup: At the beginning, the TA generates system public parameters and master secret key. Instead of generating a pair of public and secret key for IBE and ABE, respectively, the TA runs the following setup algorithm once and creates the public parameters PP and the master secret key MSK, where PP

simultaneously involves the public keys of the two cryptosystems. The TA publishes *PP* to other entities while keeps *MSK* secret. The TA invokes the ACS and proxy by giving them the public parameters *PP* and then the ACS and proxy remain online for responding to users' computational requests.

$(PP, MSK) \leftarrow \text{Setup}(1^{\ell})$: The TA runs the bilinear group generator $\mathscr{G}(1^{\ell})$ to obtain the description of bilinear groups and bilinear map, $(g, \mathbb{G}, \mathbb{G}_T, e)$. It then selects random elements $u, h, w, v \in \mathbb{G}$ and random $\alpha \in \mathbb{Z}_p$. Besides, the TA chooses an encoding function E that maps an element of $\mathbb{G}_T$ to an element of $\mathbb{G}$, i.e. $E: \mathbb{G}_T \to \mathbb{G}$. The TA sets the public parameters and the master secret key as

$$PP = (g, u, h, w, v, e(g, g)^{\alpha}, E),\ MSK = \alpha .$$

In the public parameter *PP*, only the components $(g, u, h, e(g, g)^{\alpha}, E)$ are required in the IBE-based storage system while the whole *PP* is needed in the ABE-based one.

6.2.2 User admission:

In this procedure, the users on client side can request the TA for joining the RDAC system. Since the RDAC system can be applied for cloud storages which probably involve two cryptosystems, this procedure is classified into two cases. One is for the users asking for access credentials to decrypt IBE encrypted files and the other is for the users asking for access credentials to decrypt ABE encrypted files. These two sub-procedures are described as follows.

IBE user admission: For a new user requesting IBE access credential, the TA first verifies the validation of the new user via some identification authorisation protocols, e.g., Kerberos, PAP. If the user is entitled to access data in this system, the TA creates an access credential (which will serve as a decryption key to recover the encrypted data) by calling the following credential generation algorithm $\text{CreGen}_{\text{IBE}}$ and then gives this credential to the requesting user.

$Cre_{ID} \leftarrow \text{CreGen}_{\text{IBE}}(PP, MSK, ID)$: In an IBE-employed storage system, every authorised user is associated with a unique identity *ID*. The TA takes as inputs *ID*, the public parameter *PP* and the master secret key *MSK*. It chooses a random element $r \in \mathbb{Z}_p$ and computes

$$K_0 = g^{\alpha}(u^{ID}h)^r, K_1 = g^r .$$

The TA sets $Cre_{ID} = (K_0, K_1)$ and returns it to user *ID*.

ABE user admission: For a new user requesting ABE access credential, the TA also first verifies the validation of the new user. If the user passes, the TA creates an access credential via the following credential generation algorithm $\text{CreGen}_{\text{ABE}}$. We note that in the ABE, users are associated with sets of attributes instead of single identities, hence the algorithm takes in a set *S* of attribute rather than an identity.

$Cre_S \leftarrow \text{CreGen}_{\text{ABE}}(PP, MSK, S)$: The TA takes as inputs a set of attributes, $S = \{A_1, A_2, \ldots, A_{|S|}\}$, the public parameter *PP* and the master secret key *MSK*. It chooses $|S| + 1$ random element $d, d_1, \ldots, d_{|S|} \in \mathbb{Z}_p$, where $|S|$ is the cardinality of set *S*, and computes

$$K_0 = g^{\alpha}w^d, \quad K_1 = g^d,$$

$$K_{i,2} = g^{d_i}, \quad K_{i,3} = \left(u^{A_i}h\right)^{d_i}v^{-d} .$$

The TA sets

$$Cre_S = \left(K_0, K_1, \{K_{i,2}, K_{i,3}\}_{\forall i \in \{1,2,\ldots,|S|\}}\right)$$

and returns it to the user associated with *S*.

6.2.3 File creation:

In our RDAC system, data owners encrypt data before outsourcing their data to CSP. If they are equipped with IBE cryptosystem, they encrypt their data using the IBE in advance; if equipped with IBE cryptosystem, they first encrypt using the ABE. In practice, *key encapsulation* technique is widely suggested to reduce the encryption overhead. In key encapsulation, data is first encrypted by a symmetric encryption, such as DES, AES, while the relatively costly asymmetric encryption (e.g. IBE, ABE) is applied to encapsulate the symmetric session key used in the symmetric encryption. Hence, the data size will not affect the computation of the asymmetric encryption, which significantly saves the time to create a file to be stored CSP.

Since there are two kinds of cryptosystems that generated two different encrypted files, the file creation procedure is classified into two cases:

IBE file creation: A data owner creates an IBE encrypted file as follows. First, the data owner encrypts the data using a symmetric session key *m*. Second, the owner encapsulates the symmetric key with an identity *ID* so that only the user associated with *ID* can recover *m* and then decrypt the file. The algorithm for the encapsulation of *m* is described by the following.

$CT_{ID} \leftarrow \text{Enc}_{\text{IBE}}(PP, ID, m)$: The algorithm takes as inputs the public parameter *PP*, the identity *ID* and the symmetric key $m \in \mathbb{G}_T$. It chooses a random $s \in \mathbb{Z}_p$ and computes

$$C_0 = me(g, g)^{\alpha s}, C_1 = \left(u^{ID}h\right)^s, C_2 = g^s .$$

The algorithm then outputs the ciphertext for *ID* as

$$CT_{ID} = (C_0, C_1, C_2) .$$

We note that if the data owner hopes that only himself can decrypt the file, he can encrypt by using his own identity *ID*.

ABE file creation: A data owner creates an ABE encrypted file as follows. First, the data owner also encrypts the data by a symmetric session key *m*. Second, the data owner specifies an access policy $\mathbb{A}$ indicating what attributes the potential decryptors should possess, and then encapsulates the symmetric key *m* with $\mathbb{A}$ by calling the following algorithm.

$CT_{\mathbb{A}} \leftarrow \text{Enc}_{\text{ABE}}(PP, \mathbb{A}, m)$: The algorithm takes as inputs the public parameter *PP*, the access policy $\mathbb{A}$ and the symmetric key $m \in \mathbb{G}_T$. It first generates an LSSS $(\boldsymbol{M}, \rho)$ for $\mathbb{A}$, where $\boldsymbol{M}$ is an $l \times n$ matrix and ρ maps each row of $\boldsymbol{M}$ to an attribute. The algorithm chooses a random vector $\boldsymbol{v} = (s, y_2, \ldots, y_n) \in \mathbb{Z}_p^n$, where *s* is the random secret to be shared. According to the terminology of Section 5.2, $\lambda_i = M_i \cdot \boldsymbol{v}$ is the share for the attribute associated with the *i*th row M_i of $\boldsymbol{M}$.

The algorithm chooses *l* random exponents $t_1, t_2, \ldots, t_l \in \mathbb{Z}_p$ and computes

$$C = me(g, g)^{\alpha s}, \quad C_0 = g^s$$

and

$$C_{i,1} = w^{\lambda_i}v^{t_i}, \quad C_{i,2} = \left(u^{\rho(i)}h\right)^{-t_i}, \quad C_{i,3} = g^{t_i}$$

for all $i = 1, 2, \ldots, l$. The algorithm finally outputs the ciphertext for $\mathbb{A}$ as

$$CT_{\mathbb{A}} = \left(C, C_0, \{C_{i,1}, C_{i,2}, C_{i,3}\}_{\forall i \in \{1,2,\ldots,l\}}\right) .$$

We denote by $\{Data\}_m$ the encryption of *Data* under the symmetric key *m*, and denote by $CT_{ID}(m)$ and $CT_{\mathbb{A}}(m)$ the encapsulation of *m* under IBE and ABE, respectively. Then the file stored on CSP is formed by the index identity *ID* (resp. access policy $\mathbb{A}$), the header $CT_{ID}(m)$ (resp. $CT_{\mathbb{A}}(M)$) and the body $\{Data\}_m$, as depicted in Fig. 2.

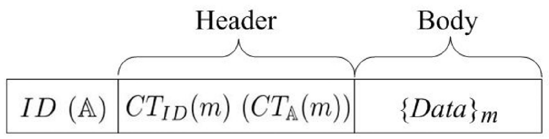

Fig. 2 *Format of files stored on clouds*

6.2.4 File conversion: In some situations, data owners need to re-define the access control policies of their IBE-encrypted files. For instance, a user of mobile devices encrypts his/her data using an identity (e.g. his/her email address) before outsourcing them to CSP; after a while, the user would like to authorise a group of users to access the data without knowing their specific identities. Instead of downloading and then decrypting the original file, the data owner prefers a more efficient mechanism. To protect privacy, the data owner may not want to hand over his/her secret (e.g. access credential) to other parties. In addition, due to computing and battery limits, mobile devices are hard to bear costly computation overhead.

The file conversion procedure deals with these concerns by realising an efficient and secure encryption conversion. In this procedure, a data owner can directly transform his/her file encrypted under IBE access policy into a file encrypted under ABE access policy, without decrypting and encrypting the data again, nor handing over any secret information. Besides, the data owner does not need to perform expensive computation and CSP will not be disturbed. This procedure runs in the following two steps.

Re-definition key generation: To convert an IBE encrypted file into an ABE encrypted file, the data owner first needs to generate a RK. In the RK generation, to reduce the computation overhead, the data owner interacts with the ACS as follows.

- The data owner first specifies an access policy $\mathbb{A}$ and generates an LSSS $(\boldsymbol{M}, \rho)$ for $\mathbb{A}$. The data owner picks random elements $x, z, y \in \mathbb{G}_T$ to compute $k' = xz \in \mathbb{G}_T, k = xy \in \mathbb{G}_T$, giving $(\mathbb{A}, k')$ (also along with $(\boldsymbol{M}, \rho)$) to the ACS;
- The ACS performs

$$R'_{\mathbb{A}} \leftarrow \mathrm{Enc}_{\mathrm{ABE}}(PP, \mathbb{A}, k'),$$

  where $R'_{\mathbb{A}}$ can be viewed as an encryption of k' under $\mathbb{A}$. The ACS then returns $R'_{\mathbb{A}}$ to the data owner;
- Parse $R'_{\mathbb{A}} = (C', C'_0, \{C'_{i,1}, C'_{i,2}, C'_{i,3}\}_{\forall i \in \{1,2,\ldots,l\}})$, where

$$C' = xz \cdot e(g,g)^{\alpha s'}, \quad C'_0 = g^{s'}, \quad C'_{i,1} = w^{\lambda'_i} v^{t_i}$$

  and λ'_i is a share of secret s'.
- The data owner picks a random element $s'' \in \mathbb{Z}_p$ and computes

$$C = C'/z \cdot y \cdot e(g,g)^{\alpha s''}, \quad C_0 = C'_0 \cdot g^{s''}, \quad C_{i,1} = C'_{i,1} \cdot w^{\lambda''_i}$$

  where λ''_i is a share of secret s''. The above computations imply that

$$C = xy \cdot e(g,g)^{\alpha(s'+s'')} = k \cdot e(g,g)^{\alpha s},$$

  and

$$C_0 = g^{s'+s''} = g^s, \quad C_{i,1} = w^{\lambda_i} v^{t_i}$$

  where $\lambda_i = \lambda'_i + \lambda''_i$ is a share of secret $s = s' + s''$.
- The owner last defines $R_{\mathbb{A}}$ the same as $R'_{\mathbb{A}}$ except that the components $C', C'_0, C'_{i,1}$ are replaced by $C, C_0, C_{i,1}$, respectively.
- The data owner uses his/her access credential $Cre_{ID} = (K_0, K_1)$ to compute $K_0 E(k)$, where $E(\cdot)$ is the encoding function published in PP. Then the owner sets

$$RK = (K_0 E(k), K_1, R_{\mathbb{A}})$$

  and gives it to the proxy.

Remark 1: (The ACS alone does not know any secret of the data owner): We note that the data owner gives k' to the ACS and receives $R'_{\mathbb{A}}$ as an encryption of k'. Since $k' = xz$ and $k = xy$ are both uniform random, the ACS cannot obtain any useful information about the secret k.

Remark 2: (The proxy alone does not know any secret of the data owner): We note that the data owner gives $RK = (K_0 E(k), K_1, R_{\mathbb{A}})$ to the proxy, where $R_{\mathbb{A}}$ is an encryption of k. Since the proxy cannot decrypt $R_{\mathbb{A}}$ without the ABE decryption key (K_0, K_1), the user's secret key is well randomised by $E(k)$.

Remark 3: (The collusion of ACS and proxy cannot break the security of the data owner): We note that the ACS generates $R'_{\mathbb{A}}$ which is an encryption of k' with element s'. The proxy has re-definition key $RK = (K_0 E(k), K_1, R_{\mathbb{A}})$, where $R_{\mathbb{A}}$ is an encryption of k with random s=s'+s''. The collusion of the ACS and the proxy can compute $C_{i,1}/v^{t_i} = w^{\lambda_i}$ since the ACS knows t_i. Further, due to the linear reconstruction of LSSS, they can compute w^s. However, they cannot compute s to recover k from C given the difficulty of discrete logarithm problem. Therefore, the secret keys of data owners are secure against the collusion attacks of the ACS and proxy.

File conversion: Receiving the re-definition key RK, the proxy is able to transform the file encrypted in IBE into a file encrypted in ABE associated with access policy $\mathbb{A}$. Given a RK generated from an access credential that is associated with an identity ID, the proxy can convert the file header CT_{ID} into a file header of ABE by calling the following algorithm $\mathrm{Conv}(PP, CT_{ID}, RK)$.

$CCT \leftarrow$ $\mathrm{Conv}(PP, CT_{ID}, RK)$: Parse $CT_{ID} = (C_0, C_1, C_2)$ and $RK = (K'_0, K_1, R_{\mathbb{A}})$. The algorithm computes

$$\begin{aligned} C'_0 &= C_0 \cdot \frac{e(K_1, C_1)}{e(K'_0, C_2)} \\ &= me(g,g)^{\alpha s} \cdot \frac{e\left(g^r, \left(u^{ID}h\right)^s\right)}{e\left(g^\alpha \left(u^{ID}h\right)^r \cdot E(k), g^s\right)} \\ &= me(g,g)^{\alpha s} \cdot \frac{1}{e(g,g)^{\alpha s} \cdot e(E(k), g^s)} \\ &= m/e(E(k), g^s) \end{aligned}$$

Given $C_2 = g^s$, the algorithm outputs the converted file header for the original one as $CCT = (C'_0, C_2, R_{\mathbb{A}})$. In this way, the proxy can ‘automatically’ transform the encryption of IBE into an encryption of ABE, without seeing the underlying plaintext m. The proxy then replaces the file header of the original file with the converted one CCT and outsources CCT and the body of the file, as well as the index $\mathbb{A}$, to the CSP.

6.2.5 File access: Since there are three types of files in the system, i.e. IBE files, ABE files and converted files, this procedure can also be classified into two cases.

IBE file access: A data consumer requests to access an IBE encrypted file stored on CSP. Since the header of the file is an encapsulation of a symmetric key m and the body is an encryption of the data with m, the data consumer first decapsulates the header to obtain m and then decrypts the body using m. The decapsulation of the header CT_{ID} is described by the following algorithm.

$m \leftarrow$ $\mathrm{Dec}_{\mathrm{IBE}}(PP, CT_{ID}, Cre_{ID})$: Parse $CT_{ID} = (C_0, C_1, C_2)$ and $Cre_{ID} = (K_0, K_1)$. The algorithm computes

$$B = \frac{e(K_1, C_1)}{e(K_0, C_2)} = \frac{e\left(g^r, \left(u^{ID}h\right)^s\right)}{e\left(g^\alpha \left(u^{ID}h\right)^r, g^s\right)} = \frac{1}{e(g,g)^{\alpha s}}$$

and outputs $m = C_0 B = me(g,g)^{\alpha s} B$.

ABE file access: A data consumer requests to access an ABE encrypted file stored on CSP. Note that in ABE, ciphertexts are associated with access policies and decryption keys are associated with sets of attributes. Then in this procedure, the condition for the data consumer to decapsulate the header $CT_{\mathbb{A}}$ of the ABE file is that the attribute set S associated with the credential Cre_S must satisfy the access structure $\mathbb{A}$.

$m/\bot \leftarrow \text{Dec}_{\text{ABE}}(PP, CT_{\mathbb{A}}, Cre_S)$: Parse the encapsulation $CT_{\mathbb{A}} = \left(C, C_0, \{C_{i,1}, C_{i,2}, C_{i,3}\}_{i=1}^{l}\right)$ and the access credential $\text{Cre}_S = \left(K_0, K_1, \{K_{j,2}, K_{j,3}\}_{j=1}^{|S|}\right)$. If S does not satisfy $\mathbb{A}$, the algorithm outputs a false symbol $\bot$. Otherwise, it first calculates the set of rows in the matrix $\boldsymbol{M}$ that provides a share to an attribute in S, i.e. $I = \{i: \rho(i) \in S\}$, where $(\boldsymbol{M}, \rho)$ is the LSSS for $\mathbb{A}$ included in $CT_{\mathbb{A}}$. Then, according to the linear reconstruction property of LSSS, the algorithm is able to find the constants $\{\omega_i \in \mathbb{Z}_p\}$ such that $\sum_{i \in I} \omega_i M_i = (1, 0, \ldots, 0)$. Then it computes

$$B = \frac{e(C_0, K_0)}{\prod_{i \in I}\left(e(C_{i,1}, K_1) \cdot e(C_{i,2}, K_{j,2}) \cdot e(C_{i,3}, K_{j,3})\right)^{\omega_i}}$$

and outputs $m = C/B$.

Correctness: If the attribute set of Cre_S satisfies the access policy $\mathbb{A}$, we have that $\sum_{i \in I} \omega_i \lambda_i = s$. Therefore

$$B = \frac{e\left(g^s, g^{\alpha} w^d\right)}{\prod_{i \in I} e\left(w^{\lambda_i} v^{t_i}, g^d\right)^{\omega_i} \cdot e\left(\left(u^{\rho(i)} h\right)^{-t_i}, g^{d_j}\right)^{\omega_i}} \cdot \frac{1}{\prod_{i \in I} e\left(g^{t_i}, \left(u^{A_j} h\right)^{d_j} v^{-d}\right)^{\omega_i}}$$
$$= \frac{e(g^s, g^{\alpha}) \cdot e\left(g^s, w^d\right)}{e(g, w)^{d \sum_{i \in I} \omega_i \lambda_i}} = e(g, g)^{\alpha s}.$$

Converted file access: A data consumer requests to access a converted file stored on CSP. The access credential of the data consumer is associated with a set S of attributes and the converted file is labelled with an access policy $\mathbb{A}$. Like in the ABE file access, the data consumer is able to decapsulate the file header to recover m if and only if S satisfies $\mathbb{A}$.

$m/\bot \leftarrow \text{Dec}_{\text{Con}}(PP, CCT, Cre_S)$: Parse the converted ciphertext $CCT = (C_0', C_2, R_{\mathbb{A}}) = (m/e(E(k), g^s), g^s, R_{\mathbb{A}})$. If S does not satisfy $\mathbb{A}$, the algorithm outputs a false symbol $\bot$. Otherwise, it calls the algorithm $\text{Dec}_{\text{ABE}}(PP, R_{\mathbb{A}}, Cre_S)$ to output the random value k previously encrypted in $R_{\mathbb{A}}$, i.e.

$$k \leftarrow \text{Dec}_{\text{ABE}}(PP, R_{\mathbb{A}}, Cre_S).$$

Then it outputs

$$m = C_0' \cdot e(E(k), C_2) = (m/e(E(k), g^s)) \cdot e(E(k), g^s).$$

7 Security analysis

We now give the rigorous security analysis of the RDAC system. We are only interested in the security of the asymmetric encryption part of the RDAC. At a high level, the security of the RDAC should guarantee that any data consumer cannot access the data they are not authorised to, even they collude with CSP. Since the RDAC involves two cryptosystems, i.e. IBE and ABE, the security also comprises these two systems' security. In the security of IBE or ABE, an adversary is defined to attack the IBE or ABE by accessing system public key and querying users' access credentials; then a secure IBE or ABE can withstand such attacks such that the adversary cannot obtain any useful information about the symmetric key encrypted in the file header.

In this section, we focus on the security of the introduction of the file conversion. In particular, the RDAC is not just a simple combination of an IBE system and an ABE system, but a more versatile system which can transform the ciphertexts of IBE into ciphertexts of ABE. Therefore, its security must take into account the additional ciphertexts generated by the file conversion. To capture the real attacks, we also endow the adversary with the authority to obtain the converted files and the RKs. Then in the security of RDAC, we consider an adversary which can access system public parameter, users' access credentials (decryption keys) and the RKs. To make a challenge, two messages are encrypted with an identity ID^*. Then the security guarantees that the adversary being challenged cannot distinguish the ciphertexts of these two messages, provided that it does not have the decryption key that is able to decrypt the encryption of ID^*. Besides, to avoid trivial solution, the adversary cannot obtain the RK that is able to transform the challenge ciphertext into the ciphertext for which the adversary holds decryption keys. Formally, we define the security of the RDAC system by a game played between a challenger and an adversary $\mathscr{A}$ as follows.

Init: The adversary $\mathscr{A}$ declares an identity ID^* to be attacked.

Setup: The challenger runs the Setup algorithm to obtain the system public parameter PP and gives it to the adversary $\mathscr{A}$.

Phase 1: The adversary $\mathscr{A}$ issues queries $Q_1, \ldots, Q_{q'}$ to the challenger, where Q_i for $1 \le i \le q'$ is one of the following queries:

- $\textbf{Reveal}_{IBE}(ID)$. This is the key query made by $\mathscr{A}$ for the access credential (decryption key) of identity ID. The challenger responds by calling the algorithm $\text{CreGen}_{\text{IBE}}(PP, MSK, ID)$ to obtain access credential Cre_{ID} and returning it to $\mathscr{A}$.
- $\textbf{Reveal}_{ABE}(S)$. This is the key query made by $\mathscr{A}$ for the access credential of attribute set S. The challenger responds by calling the algorithm $\text{CreGen}_{\text{ABE}}(PP, MSK, S)$ to obtain access credential Cre_S and returning it to $\mathscr{A}$.
- $\textbf{RKReveal}(ID \to \mathbb{A})$. This is the key query made by $\mathscr{A}$ for the RK that is able to transform an IBE ciphertext associated with ID into an ABE ciphertext associated with $\mathbb{A}$. In response, the challenger directly creates a re-definition key CK using the decryption key $\text{Cre}_{ID} \leftarrow \text{CreGen}_{\text{IBE}}(PP, MSK, ID)$.

Challenge: In this phase, the adversary $\mathscr{A}$ is challenged by an IBE ciphertext. The adversary $\mathscr{A}$ outputs two equal-length messages M_0 and M_1 with the restrictions that (i) the adversary has not queried $\textbf{Reveal}_{IBE}(ID)$ for $ID = ID^*$; and (ii) for any ID and any $S \in \mathbb{A}$, the adversary $\mathscr{A}$ has queried at most one of two queries $\textbf{RKReveal}(ID \to \mathbb{A})$ and $\textbf{Reveal}_{ABE}(S)$. The challenger flips a coin $\beta \in \{0, 1\}$, encrypts M_β under ID^* and returns the resulting ciphertext CT_{ID^*} to $\mathscr{A}$.

Phase 2: The adversary $\mathscr{A}$ issues queries $Q_{q'+1}, \ldots, Q_q$ to the challenger just as in Phase 1, with the added constraints that (i) it cannot query $\textbf{Reveal}_{IBE}(ID)$ for $ID = ID^*$; and (ii) it can query at most one of two queries $\textbf{CKReveal}(ID \to \mathbb{A})$ and $\textbf{Reveal}_{ABE}(S)$ for any ID and any $S \in \mathbb{A}$.

Guess: The adversary $\mathscr{A}$ outputs a guess $\beta' \in \{0, 1\}$.

The advantage of an attacker $\mathscr{A}$ in this game is defined as $Adv_{\mathscr{A}} = |\Pr[\beta = \beta'] - 1/2|$.

Definition 2: We say that a RDAC system is semantically secure against chosen-plaintext attack if no polynomial-time adversary has a non-negligible advantage in the above game.

In the security model above, the challenge ciphertext is generated with an identity in the IBE system. This is because the converted ciphertexts derive from the IBE ciphertexts and we try to resist the attacks on the converted ciphertexts. The attacks on the standard IBE and ABE ciphertexts are identical to those defined in [4, 43], respectively, hence we omit here for simplification.

The following theorem states that our RDAC system is semantically secure if the underlying IBE and ABE system are both semantically secure..

Theorem 1: Suppose there is an adversary breaking the semantic security of RDAC system with advantage $Adv_{\mathscr{A}}$. Then we

Table 2 Computation

Operation	Computation complexity
setup	$1\tau_p + 1\tau_e$
user admission	IBE: $4\tau_e$
	ABE: $(3\|S\| + 4)\tau_e$
file creation	IBE: $4\tau_e$
	ABE: $(5l + 2)\tau_e$
file conversion	RK Gen: $(5l' + 2)\tau_e$
	Conversion: $2\tau_p$
file access	IBE: $2\tau_p$
	ABE: $(3\|S^*\| + 1)\tau_p + \|S^*\|\tau_e$
	Converted File: $(3\|S'\| + 2)\tau_p + \|S'\|\tau_e$

can construct an algorithm $\mathscr{B}$ that breaks the underlying IBE scheme with advantage ε such that

$$\epsilon + \epsilon' \geq Adv_{\mathscr{A}}/\hat{e}(1 + q_K),$$

where ϵ' is the advantage of breaking the semantic security of the CP-ABE system of [43], *$\hat{e}$ is the base of the natural logarithm and q_K is the number of key queries made by $\mathscr{A}$.*

Proof: We now give formal proof of Theorem 1. We construct an algorithm $\mathscr{B}$ to interact with an adversary $\mathscr{A}$ and leverage the output of $\mathscr{A}$ to break the underlying IBE scheme. As stated in the security definition, an adversary is allowed to query conversion keys and particularly, it can query the RK that is derived from the challenge identity ID^*. Unfortunately, however, algorithm $\mathscr{B}$ cannot generate a decryption key of ID^* and then apply it to create the required RK, because the security definition of an IBE system forbids key queries on the challenge identity. To address such paradox, following the techniques of [38], we let the algorithm $\mathscr{B}$ generate random RK for the challenge identity. We also show the indistinguishability between the correct RK and random one. We construct a table $T = (c \in \{0, 1\}, ID, S, \mathbb{A})$ such that if $c = 1$, algorithm $\mathscr{B}$ generates correct keys and otherwise random keys. We denote by "*" the wildcard.

We prove the semantic security as defined in Definition 2.

Init: The adversary $\mathscr{A}$ outputs an identity ID^* to be attacked.

Setup: Algorithm $\mathscr{B}$ runs the setup algorithm of IBE to obtain public key $(g, u, h, e(g, g)^\alpha)$. Then it chooses random $w, v \in \mathbb{G}$ and function $E: \mathbb{G}_T \to \mathbb{G}$. Algorithm $\mathscr{B}$ returns the public parameter $PP = (g, u, h, w, v, e(g, g)^\alpha, E)$ to $\mathscr{A}$.

Phase 1: The adversary $\mathscr{A}$ makes the following queries:

- **Reveal**$_{IBE}(ID)$. First, algorithm $\mathscr{B}$ chooses a random bit $c \in \{0, 1\}$ so that Pr $[c = 1] = \varepsilon$ which will be determined later. If $c = 0$, or $ID = ID^*$ or $(0, ID, *, *)$ exists on the table T, $\mathscr{B}$ returns a random bit and aborts. Otherwise, $\mathscr{B}$ calls the key generation oracle of the IBE scheme on input ID and returns the resulting access credential Cre_{ID} to $\mathscr{A}$. Finally, $\mathscr{B}$ records $(c, ID, \perp, \perp)$ on the table T.
- **Reveal**$_{ABE}(S)$. Again, $\mathscr{B}$ chooses a random bit $c \in \{0, 1\}$ so that Pr $[c = 1] = \varepsilon$. If $c = 0$, or $(0, *, S, *)$ exists on the table T, $\mathscr{B}$ returns a random bit and aborts. Otherwise, $\mathscr{B}$ calls the key generation oracle of the CP-ABE scheme on input attribute set S and returns the resulting access credential Cre_S to $\mathscr{A}$. Algorithm $\mathscr{B}$ then records $(c, \perp, S, \perp)$ on T.
- **RKReveal**$(ID \to \mathbb{A})$. In response, $\mathscr{B}$ chooses a random bit $c \in \{0, 1\}$ so that Pr $[c = 1] = \varepsilon$. If $c = 1$ and $ID \neq ID^*$, or $(1, ID, *, *)$ exists on the table, $\mathscr{B}$ calls the key generation oracle of the IBE scheme to output an access credential for ID and then uses it to directly create a re-definition key RK associated with $\mathbb{A}$ just as in the RK generation procedure but without the help of ACS. Algorithm $\mathscr{B}$ then returns RK to $\mathscr{A}$. Otherwise, $\mathscr{B}$ returns a random RK $CK = (X_0, X_1, \text{Enc}_{\text{ABE}}(PP, \mathbb{A}, z))$, where X_0 and X_1 are random elements of $\mathbb{G}$ and z is a random of $\mathbb{G}_T$. Finally, $\mathscr{B}$ records $(c, ID, \perp, \mathbb{A})$ on the table T.

Challenge: The adversary $\mathscr{A}$ declares two messages M_0 and M_1. If there exists a record $(1, ID, S, \mathbb{A})$ for any $S \in \mathbb{A}$ on the table, $\mathscr{B}$ returns a random bit and aborts. Otherwise, $\mathscr{B}$ calls the encryption oracle of the IBE scheme on input (ID^*, M_0, M_1) and returns the resulting ciphertext to $\mathscr{A}$ as a challenge.

Phase 2: The same as Phase 1 except the unallowed queries described in Definition 2.

Guess: The adversary $\mathscr{A}$ outputs its guess $\beta' \in \{0, 1\}$ and $\mathscr{B}$ outputs the same.

We now discuss the probability that $\mathscr{B}$ does not abort in the whole game. We denote by q_K the number of key queries made by $\mathscr{A}$. Then $\mathscr{B}$ does not abort in phase 1 and phase 2 with probability ε^{q_K} and in the challenge phase with probability $1 - \varepsilon$. Therefore, the probability that $\mathscr{B}$ does not abort in the whole game is $\varepsilon^{q_K}(1 - \varepsilon)$ with the maximum value $1 - 1/(q_K + 1)$. Then, the probability that $\mathscr{B}$ does not abort is $1/\hat{e}(1 + q_K)$.

We next discuss the probability that $\mathscr{A}$ successfully distinguishes a correct key from a random one. The random RK is $RK = (X_0, X_1, \text{Enc}_{\text{ABE}}(PP, \mathbb{A}, z))$, where (X_0, X_1) must be valid terms for some IBE key (K_0, K_1). Hence, the only way to distinguish a correct key from a random one is to distinguish $\text{Enc}_{\text{ABE}}(PP, \mathbb{A}, z)$ from the encryption $\text{Enc}_{\text{ABE}}(PP, \mathbb{A}, k)$, which is equivalent to breaking the security of the CP-ABE. Therefore, we have that the probability of $\mathscr{A}$ in distinguishing a correct key from a random one is about the advantage ε' in breaking the security of the CP-ABE.

In sum, given the advantage $\text{Adv}_{\mathscr{A}}$ of $\mathscr{A}$ in breaking the semantic security of the RDAC system, algorithm $\mathscr{B}$ can break the semantic security of the underlying IBE scheme with an advantage ϵ such that $\epsilon + \epsilon' \geq \text{Adv}_{\mathscr{A}}/\hat{e}(1 + q_K)$.

8 Experimental evaluation

In this section, we evaluate our RDAC system from both theoretical and experimental aspects.

8.1 Theoretical analysis

We first analyse the computation complexity of each procedure of the RDAC system. In this analysis, we only consider the computation costs of the asymmetric encryption and decryption in RDAC since the costs of the symmetric ones are only related to the data size and independent of the access control re-definition. We focus on the most expensive computations, i.e. bilinear map and exponentiation, performed in groups $\mathbb{G}$ and $\mathbb{G}_T$. We denote by τ_p and τ_e the time consumed by the bilinear map and exponentiation, respectively, without discriminating the exponentiations in $\mathbb{G}$ from those in $\mathbb{G}_T$. Although the multiplication is also a basic operation in the groups, it consumes a negligible time compared to the bilinear map or exponentiation, hence we do not take into account the cost of multiplication.

Table 2 demonstrates the time cost by each procedure of the RDAC system. In this table, $|S|$ denotes the cardinality of the attribute set S associated with an access credential, l the number of attributes associated with an ABE encapsulation, $|S^*|$ the cardinality of the attributes set S^* that satisfies an access policy, l' denoting the number of attributes involved in an access policy used to generate a RK and $|S'|$ denotes the cardinality of the attribute set S' satisfying the access policy associated with a converted file.

Table 2 reveals the flexibility and efficiency of the RDAC. Since the file creation and the file access for IBE consume very less costs ($4\tau_e$ and $2\tau_p$, resp.), the IBE system involved in RDAC is suitable for source-limited access devices, such as mobile phones and tablets, to protect the privacy of outsourced data. The ABE can be employed on source-adequate access devices such as laptops to achieve fine-grained access control on outsourced data. Due to the properties of ABE, the time to create a file and to access a file are both linear with the number of involved attributes. When a user wants to re-define the access policy of his/her IBE encrypted file

Table 3 Performance of the IBE

	$\text{CreGen}_{\text{IBE}}$	Enc_{IBE}	Dec_{IBE}
IBE	38.81 ms	30.23 ms	15.54 ms

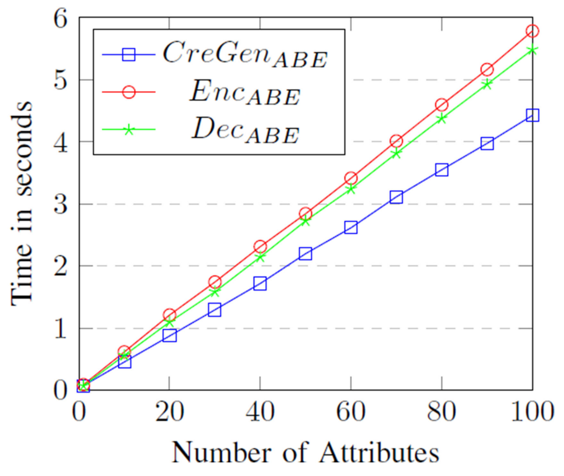

Fig. 3 *Execution time of the CP-ABE*

and allows a group of users to access his data, the RDAC system also provides an efficient way. The time to create a RK is about $(5l' + 2)\tau_e$ and the encrypted file conversion only takes the time $2\tau_p$. Notably, the user spends little time in the files conversion, since the most tasks of the RK generation are given to the ACS and the file conversion is only taken by the proxy. The time to access a converted file is almost the same as that to access an ABE file, hence the file visitors feel no difference between accessing a converted file and a normal file. All these features make our RDAC system very practical to protect the privacy of data outsourced to an untrusted third party.

Table 1 compares our RDAC scheme with other schemes that also achieve the access control re-definition mechanism, in terms of the size of RK, the size of converted file, the number of bilinear pairings required in the converted file access, the granularity of access control (fine-grained or not) and the consumption of the users (clients) in file conversion.

The schemes in [37, 38] achieve the file transformation from an IBE file into another IBE file, which renders the RK and the converted file both short. Although these two schemes fulfil clients' low consumption by requiring less computation of clients to generate RKs, they do not support the fine-grained access control on outsourced data. The works [39, 40] achieve fine-grained access control by implementing file conversion in ABE. Hence, the size of RK is linear with the cardinality of the attribute set S of the original key plus the number l of the attributes involved in the access policy specified by the data owner. Also, the converted file grows linearly with the number of involved attributes and the number of pairings in decrypting a file is linear with the cardinality of the attribute set S^* meeting an access policy.

The main disadvantage of [39, 40] is that the file conversion demands the users on client side too much computation overheads, which seem unbearable for the potential users with limited resources (e.g. mobile phones). However, our RDAC scheme requires the users to take only few computations to generate RKs and no costs to convert files. Compared to [39, 40], the proposed scheme achieves shorter RKs, which also saves the communication overheads of the users.

8.2 Experimental analysis

To evaluate the performance of our RDAC scheme in isolation (i.e. without confounding factors such as network lag, file I/O etc.), we conducted a series experiments on two hardware platforms: a Huawei mobile device equipped with a 2.2 GHz CPU, running at 3 GB RAM and a desktop PC with a 4-core Intel Core i3 CPU, 2.0G RAM. The cryptographic operations were implemented by using the Stanford pairing-based cryptography (PBC) library version 0.5.12 (available at http://crypto.stanford.edu/pbc/). In the experiments, we are mostly interested in the performance of the user admission, file creation, file conversion and file access procedures of the RDAC system.

The RDAC system adopts the key encapsulation technique, that is, a data is first encrypted by a symmetric key and this symmetric key is then encrypted by the IBE scheme or the ABE scheme. We took the symmetric encryption AES-128 as the key encapsulation tool and evaluated the time it consumed to encrypt and decrypt a file. We tested three files with size about 1, 10 and 100 MB, respectively. The time to encrypt these files is about 0.1, 2.5, 26.8 s, respectively; and the time to decrypt their encrypted format is about 0.2, 5.4, 51.7 s, respectively. We can see the symmetric cryptosystem is very efficient for encrypting and decrypting files with common size.

We next evaluated the performance of the encapsulations for random symmetric keys. We first ran the tests on the mobile device. To obtain our baseline results, we encapsulated a random 128-bit symmetric key under an arbitrary string using the IBE scheme. We repeated the credential generation $\text{CreGen}_{\text{IBE}}$, file creation Enc_{IBE} and file access Dec_{IBE} algorithms 100 times on the mobile devices to smooth any experimental variability. Table 3 gives the time benchmarks of our underlying IBE scheme. Since the computations of the IBE are fixed (which has been analysed in Table 2), the algorithms of it consume constant time (indeed, only depending on the bilinear groups used in the system).

We then ran the experimental tests on the desktop PC. Also, we choose to encapsulate a random symmetric key using the CP-ABE scheme. In the CP-ABE, both file creation Enc_{ABE} and file access Dec_{ABE} procedures' performance depend on the complexity of the ciphertext's policy. To reflect this notion in our tests, we first generated the CP-ABE encapsulation associated with the policies in the form of $(A_1 \text{ AND } A_2 \text{ AND } \cdots \text{AND} A_i)$, where A_i is an attribute with index i increasing from 1 to 100. Accordingly, we constructed access credentials via the $\text{CreGen}_{\text{ABE}}$ that involved i attributes necessary to decapsulate the encapsulation. This reasonable setting of the policy makes sure that the file access procedure depends on all i components of the CP-ABE encapsulation.

Fig. 3 shows the time consumed by the credential generation $\text{CreGen}_{\text{ABE}}$, the file creation Eec_{ABE} and the file access Dec_{ABE} algorithms of the CP-ABE. It can be seen that the cost time of these algorithms all grows linearly with the number of attributes. Compared to IBE, the CP-ABE requires more consuming time, which is mainly introduced by the realisation of access policies over attributes. However, for a common case that 20 attributes are usually enough to describe a file, the time cost by the file creation and the file access is also acceptable (just about 1 s) by a PC's user, who wants to leverage the CP-ABE to fulfil the fine-grained access control over his outsourced data.

We finally ran experiments for the file conversion and converted file access. The file conversion consists of the RK generation procedure performed by an IBE user and the ACS, and the file conversion algorithm executed by the proxy server. The converted file access is executed by the ABE user. Hence, we still ran the tests on the mobile devices and the desktop PC, respectively.

Fig. 4 shows the time consumed by an (IBE) data owner and the ACS in RK generation, the time consumed by the proxy to convert a file and the time for an (ABE) data consumer to access a converted file. In the RK generation, the data owner (in IBE) only needs to take one-third computations compared to other works [39, 40] where a data owner has to take the full computation of RK generation. For a practical number 20 of attributes, the data owner only needs to take about 3 s to compute a RK, which is rather acceptable consumption for a mobile phone user. The most computations for RK generation are taken by the ACS which encapsulates the blinded value via the CP-ABE, thus the time cost by ACS is very close to that of Enc_{ABE}. The time to access a converted file is almost the same as that to decapsulate an ABE ciphertext, thus the ABE data consumer will feel little difference in accessing a normal ABE file and a converted file. In the whole file

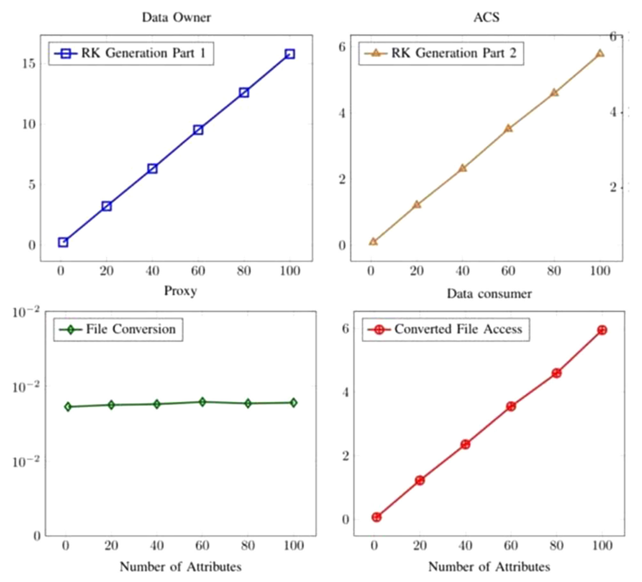

Fig. 4 *Execution time of file conversion and access*

conversion, all required tasks are taken by the data owner, the ACS and the proxy, which leaves the CSP not being disturbed.

From the analyses above, our RDAC scheme provides two cryptosystems for users with different requirements to protect data privacy. Users can choose which cryptosystem to encrypt their data according to their computing ability and requirements. To enable fine-grained access control on their outsourced data, users can pick the CP-ABE cryptosystem to encrypt at an acceptable cost. As an advantageous feature, our RDAC system allows the users to change their simple access control policies achieved by the IBE to fine-grained access control policies provided by the ABE. The costs of the file creation and file access procedures of IBE and ABE are both independent of the file conversion.

9 Conclusion

In this paper, we proposed a new approach referred to as RDAC to enable access policy re-definition in two different cryptosystems, i.e. IBE and ABE. The RDAC adopts IBE and ABE cryptographic tools to protect data privacy and allows users to choose either to encrypt data according to their local conditions. The most appealing advantage is that in RDAC, users can transfer the files encrypted with simple IBE access policies into files encrypted with ABE fine-grained access policies without leaking the sensitive data nor performing laborious computations. These features make the RDAC very suitable for the applications where a preliminary encryption is used to secure sensitive data, but later a fine-grained access control on the data is required. We presented a concrete RDAC scheme and formally proved its security without random oracle. The theoretical and experimental analyses showed that the RDAC scheme achieves a reasonable and acceptable performance.

10 References

[1] Hu, H., Xu, J., Ren, C., *et al.*: ‘Processing private queries over untrusted data cloud through privacy homomorphism’. IEEE Int. Conf. on Data Engineering, Hannover, Germany, 2011, pp. 601–612

[2] Wang, C., Chow, S. S. M., Wang, Q., *et al.*: ‘Privacy-preserving public auditing for secure cloud storage’, *IEEE Trans. Comput.*, 2013, **62**, (2), pp. 362–375

[3] Lee, B.: ‘Unified public key infrastructure supporting both certificate-based and id-based cryptography’. 2010 Int. Conf. on Availability, Reliability and Security, Vienna, Austria, 2010, pp. 54–61

[4] Boneh, D., Boyen, X.: ‘Efficient selective-ID secure identity-based encryption without random oracles’. EUROCRYPT, Interlaken, Switzerland, 2004, pp. 223–238

[5] Tan, C. C., Wang, H., Zhong, S., *et al.*: ‘IBE-lite: a lightweight identity-based cryptography for body sensor networks’, *IEEE Trans. Inf. Technol. Biomed.*, 2009, **13**, (6), pp. 926–932

[6] Deng, H., Wu, Q., Qin, B., *et al.*: ‘Tracing and revoking leaked credentials: accountability in leaking sensitive outsourced data’. ASIACCS, ACM, Kyoto, Japan, 2014, pp. 425–434

[7] Chen, W.: ‘An IBE-based security scheme on internet of things’. 2nd Int. Conf. on Cloud Computing and Intelligence Systems, Hangzhou, China, 2012, pp. 1046–1049

[8] Sahai, A., Waters, B.: ‘Fuzzy identity-based encryption’. EUROCRYPT, north of Aarhus, 2005, pp. 457–473

[9] Yu, S., Wang, C., Ren, K., *et al.*: ‘Achieving secure, scalable, and fine-grained data access control in cloud computing’. 2010 Proc. INFOCOM, San Diego, California, 2010, pp. 1–9

[10] Yang, Y., Jia, X.: ‘Attributed-based access control for multi-authority systems in cloud storage’. 32nd Int. Conf. on Distributed Computing Systems, Macau, China, 2012, pp. 536–545

[11] Alshehri, S., Radziszowski, S. P., Raj, R. K.: ‘Secure access for healthcare data in the cloud using ciphertext-policy attribute-based encryption’. IEEE 28th Int. Conf. on Data Engineering Workshops, Arlington, Virginia, USA, 2012, pp. 143–146

[12] Hudic, A., Smith, P., Weippl, E. R.: ‘Security assurance assessment methodology for hybrid clouds’. Computers and Security, 2017

[13] Ateniese, G., Fu, K., Green, M., *et al.*: ‘Improved proxy re-encryption schemes with applications to secure distributed storage’, *ACM Trans. Inf. Syst. Sec.*, 2006, **9**, (1), pp. 1–30

[14] Matsuo, T.: ‘Proxy re-encryption systems for identity-based encryption’. Pairing, Tokyo, Japan, 2007, pp. 247–267

[15] Mizuno, T., Doi, H.: ‘Hybrid proxy re-encryption scheme for attribute-based encryption’. Information Security and Cryptology, 2011, vol. 6151, pp. 457–473

[16] Liang, K., Huang, X., Guo, F., *et al.*: ‘Privacy-preserving and regular language search over encrypted cloud data’, *IEEE Trans. Inf. Forensics Sec.*, 2016, **11**, (10), pp. 2365–2376

[17] Sandikkaya, M. T., Ovatman, T., Harmanci, A. E.: ‘Design and formal verification of a cloud compliant secure logging mechanism’, *IET Inf. Sec.*, 2016, **10**, (4), pp. 203–214

[18] Hamann, M., Krause, M., Meier, W., *et al.*: ‘Design and analysis of small-state grain-like stream ciphers’, *Cryptography Commun.*, 2018, **10**, pp. 803–834

[19] Albrecht, M.R., Rechberger, C., Schneider, T., *et al.*: ‘Ciphers for MPC and FHE’. EUROCRYPT 2015, Sofia, Bulgaria, 2015, pp. 430–454

[20] Grassi, L., Rechberger, C., Rønjom, S.: ‘A new structural-differential property of 5-round AES’. EUROCRYPT 2017, Paris, France, 2017, pp. 289–317

[21] Kao, Y.-W., Huang, K.-Y., Gu, H.-Z., *et al.*: ‘Ucloud: a user-centric key management scheme for cloud data protection’, *IET Inf. Sec.*, 2013, **7**, (2), pp. 144–154

[22] Yao, A.: ‘How to generate and exchange secrets’. 27th Annual Symp. on Foundations of Computer Science, Toronto, 1986, pp. 162–167

[23] Zhang, L., Li, X.-Y., Liu, Y., *et al.*: ‘Verifiable private multi-party computation: ranging and ranking’. 2013 Proc. IEEE INFOCOM, Turin, Italy, 2013, pp. 605–609

[24] Gueron, S., Lindell, Y., Nof, A., *et al.*: ‘Fast garbling of circuits under standard sssumptions’, *J. Cryptol.*, 2018, **31**, (3), pp. 798–844

[25] Yoshida, M., Obana, S.: ‘On the (in)efficiency of non-interactive secure multiparty computation’, *Des. Codes Cryptogr.*, 2018, **86**, (8), pp. 1793–1805

[26] Li, J., Wang, Q., Wang, C., *et al.*: ‘Fuzzy keyword search over encrypted data in cloud computing’. 2010 Proc. IEEE INFOCOM, San Diego, California, 2010, pp. 441–445

[27] Li, M., Yu, S., Cao, N., *et al.*: ‘Authorized private keyword search over encrypted data in cloud computing’. 2011 31st Int. Conf. on Distributed Computing Systems, Minneapolis, USA, 2011, pp. 383–392

[28] Kuzu, M., Islam, M. S., Kantarcioglu, M.: ‘Efficient similarity search over encrypted data’. 2012 IEEE 28th Int. Conf. on Data Engineering, Washington, DC, USA, 2012, pp. 1156–1167

[29] Hamlin, A., Shelat, A., Weiss, M., *et al.*: ‘Multi-key searchable encryption, revisited’. Public-Key Cryptography-PKC 2018, Rio de Janeiro, Brazil, 2018, pp. 95–124

[30] Gentry, C.: ‘Fully homomorphic encryption using ideal lattices’. Symp. on Theory of computing 2009, ACM, Bethesda, MD, USA, 2009, pp. 169–178

[31] Chatterjee, A, Sengupta, I.: ‘Translating algorithms to handle fully homomorphic encrypted data on the cloud’, *IEEE Trans. Cloud Comput.*, 2018, **6**, (1), pp. 287–300

[32] Cao, X., Moore, C., Neill, M. O., *et al.*: ‘Optimised multiplication architectures for accelerating fully homomorphic encryption’, *IEEE Trans. Comput.*, 2016, **65**, (9), pp. 2794–2806

[33] Wang, W., Hu, Y., Chen, L., *et al.*: ‘Exploring the feasibility of fully homomorphic encryption’, *IEEE Trans. Comput.*, 2015, **64**, (3), pp. 698–706

[34] Gentry, C., Halevi, S.: ‘Implementing gentry¡¯s fully-homomorphic encryption scheme’. EUROCRYPT 2011, Tallinn (Estonia), 2011, pp. 129–148

[35] Doröz, Y., Hoffstein, J., Pipher, J., *et al.*: ‘Fully homomorphic encryption from the finite field isomorphism problem’. Public-Key Cryptography-PKC 2018, Rio de Janeiro, Brazil, 2018, pp. 125–155

[36] Libert, B., Vergnaud, D.: ‘Unidirectional chosen-ciphertext secure proxy re-encryption’, *IEEE Trans. Inf. Theory*, 2011, **57**, (3), pp. 1786–1802

[37] Green, M., Ateniese, G.: ‘Identity-based proxy re-encryption’. Applied Cryptography and Network Security, Zhuhai, China, 2007, pp. 288–306

[38] Chu, C. K., Tzeng, W. G.: ‘Identity-based proxy re-encryption without random oracles’. Information Security, Sandton, South Africa, 2007, pp. 189–202

[39] Liang, X., Cao, Z., Lin, H., *et al.*: ‘Attribute based proxy re-encryption with delegating capabilities’. Int. Symp. on Information, Computer, and Communications Security, ACM, Sydney, Australia, 2009, pp. 276–286

[40] Liang, K., Fang, L., Wang, D.S., *et al.*: ‘A ciphertext-policy attribute-based proxy re-encryption with chosen-ciphertext security’, IACR Cryptology ePrint Archive, Report 2013/236 (2013), http://eprint.iacr.org/

[41] Derler, D., Krenn, S., Lorünser, T., *et al.*: 'Revisiting proxy re-encryption: forward secrecy, improved security, and applications'. Public-Key Cryptography-PKC 2018, Rio de Janeiro, Brazil, 2018, pp. 219–250
[42] Beimel, A.: 'Secure schemes for secret sharing and key distribution', PhD thesis, Israel Institute of Technology, Technion, Haifa, Israel, 1996
[43] Rouselakis, Y., Waters, B.: 'Practical constructions and new proof methods for large universe attribute-based encryption'. SIGSAC Conf. on Computer & communications security, ACM, Berlin, 2013, pp. 463–474