

Flexible Attribute-Based Encryption Applicable to Secure E-Healthcare Records

Bo Qin · Hua Deng · Qianhong Wu* · Josep Domingo-Ferrer · David Naccache · Yunya Zhou

Received: date / Accepted: date

Abstract In e-healthcare record systems (EHRS), attribute-based encryption (ABE) appears as a natural way to achieve fine-grained access control on health records. Some proposals exploit key-policy ABE (KP-ABE) to protect privacy in such a way that all users are associated with specific access policies and only the ciphertexts matching the users' access policies can be decrypted. An issue with KP-ABE is that it requires an *a priori* formulation of access policies during key generation, which is not always practicable in EHRS because the policies to access health records are sometimes determined after key generation. In this paper, we revisit KP-ABE and propose a *dynamic* ABE paradigm, referred to as access policy redefinable ABE (APR-ABE). To address the above issue, APR-ABE allows users to redefine their access policies and delegate keys for the redefined ones; hence *a priori* precise policies are no longer mandatory. We construct an APR-ABE scheme with short ciphertexts and prove

its full security in the standard model under several static assumptions.

Keywords E-Healthcare records · Privacy · Access control · Attribute-based encryption

1 Introduction

Attribute-based encryption (ABE) provides fine-grained access control over encrypted data by using access policies and attributes embedded in secret keys and ciphertexts. ABE cryptosystems [19] fall into two categories: key-policy ABE (KP-ABE) [8] systems and ciphertext-policy ABE (CP-ABE) [3] systems. In a CP-ABE system, the users' secret keys are associated with sets of attributes, and a sender generates a ciphertext with an access policy specifying the attributes that the decryptors must have. Alternatively, in a KP-ABE system, the users' secret keys are labeled with access policies and the sender specifies a set of attributes; only the users whose access policies match the attribute set can decrypt.

ABE requires *a priori* access policies, which are not always available. This may limit its applications in practice. The following scenario illustrates our point.

In an e-healthcare record system (EHRS), Alice's health records are encrypted by the doctors whom she consulted before. When Alice authorizes some doctors to access her encrypted medical records, she may have no sufficient expertise to precisely determine which doctors should access the records. Instead, according to her experience and common sense, she may specify a policy saying that the doctor ought to be medicine professor with five-year working experience from the hospitals she knows. After a matching doctor Bob sees Alice's medical materials, Bob finds that Alice has something wrong with her heart. Hence, a cardiologist's advice must be sought; thus, a cardiologist (who can be a professor or not) must be allowed to see Alice's documents.

Bo Qin
Renmin University of China
No. 59, Zhongguangun Street, Haidian District, Beijing, China

Hua Deng
School of Computer, Wuhan University, Wuhan, China

Qianhong Wu, Corresponding author
School of Electronic and Information Engineering, Beihang University
XueYuan Road No.37, Haidian District, Beijing, China
Tel.: 0086 10 8233 9469
E-mail: qianhong.wu@buaa.edu.cn

Josep Domingo-Ferrer
Universitat Rovira i Virgili, Department of Computer Engineering and Mathematics
UNESCO Chair in Data Privacy, E-43007 Tarragona, Catalonia

David Naccache
École normale supérieure, Département d'informatique
45 rue d'Ulm, F-75230, Paris Cedex 05, France

Yunya Zhou
School of Electronic and Information Engineering, Beihang University
XueYuan Road No.37, Haidian District, Beijing, China

In this application, the main obstacle to apply ABE is that Alice, serving as the key generation authority, cannot generate secret keys for access policies that are *a priori* “carved in stone”, because she does not clearly know which experts are necessary for her diagnosis.

In fact, the access policy must be dynamically modified. That is, authorized users must be able to redefine their access policies and then delegate secret keys for the redefined access policies to other users. For instance, in the above motivating scenario, Alice first authorizes doctors with some general attributes to access her encrypted medical records. After the matching doctor makes a preliminary diagnosis and finds something wrong with Alice’s heart, the doctor redefines his access policy to involve some special attributes (e.g. specialty: cardiologist) and delegates to the doctor with the redefined access policy. In this way, *a priori* precise access policies are not mandatory during key generation because they can be later redefined in delegation.

There are already some ABE schemes supporting delegation. The CP-ABE schemes in [3,7,21] allow users to delegate *more restricted secret keys, that is, keys for attribute sets that are subsets of the original ones*. In KP-ABE, the schemes proposed in [8,13,6,18] provide a delegation mechanism, but all of them require that the access policy to be delegated be more restrictive. This *limited* delegation functionality is often insufficient: for example, in the motivating application above Bob should be able to delegate to a cardiologist even if Bob is not a cardiologist himself. Limiting the user to delegating keys for other users associated with more restrictive access policies is too rigid.

The challenge of providing appropriate delegation for the applications above has to do with the underlying secret sharing scheme. In most KP-ABE schemes ([8,13,18]), secret sharing schemes are employed to share a secret in key generation and reconstruct the secret during decryption. In the key generation, each attribute involved in the access policy needs to be associated with a secret share. If there are new attributes in the target access policy to be delegated to, users cannot delegate a secret key for the access policy since they are unable to generate shares for the new attributes without knowing the secret. This is why the above mentioned KP-ABE schemes require the delegated access policy to be more restrictive than the original one. This hinders applying them for the motivating application, where the doctor with general attributes would like to delegate his access rights to a doctor associated with new special attributes.

1.1 Our Work

We propose a dynamic primitive referred to as access policy redefinable ABE (APR-ABE). The functional goal of APR-ABE is to provide a more dynamic delegation mechanism. In an APR-ABE system, users can play the role of the key

generation authority by delegating secret keys to their subordinates. The delegation does not require the redefined access policy to be more restrictive than the one of the delegating key.

Noting that attributes are very often hierarchically related in the real world, we arrange the attribute universe of APR-ABE in a matrix. For example, we can place the attribute “Internal medicine” at a higher level of the matrix than the attribute “Cardiologist”. Due to this arrangement, the notion of *attribute vector* naturally comes up: an attribute vector can be generated by picking single attributes from upper levels to lower levels. By using attribute vectors, we can realize a delegation that allows new attributes to be added into the original access policy and a secret key to be delegated for the resulting policy. This delegation is similar to the one of hierarchical identity-based encryption (HIBE,[4]), but with the difference that only delegation to the attributes consistent with the attribute matrix is allowed.

We present an APR-ABE framework based on KP-ABE and define its full security. In APR-ABE, the users’ secret keys are associated with an access structure formalized by attribute vectors. Users at higher levels can redefine their access structures and then delegate secret keys to others in lower levels without the constraint that the redefined access structures of the delegated keys be more restrictive. Ciphertexts are generated with sets of attribute vectors, and decryption succeeds if and only if the attribute set of a ciphertext satisfies the access structure associated with a secret key, just as in the ordinary KP-ABE. In full security, a strong security notion in ABE systems, an adversary is allowed to access public keys, create attribute vectors and query secret keys for specified access structures. Full security states that not even such an adversary can get any useful information about the plaintext encrypted in a ciphertext, provided that he does not have the correct decryption key.

We construct an APR-ABE scheme by employing a linear secret sharing scheme (LSSS). An LSSS satisfies linearity, that is, new shares generated by multiplying existing shares by random factors can still reconstruct the secret. Hence, when delegating to new attributes, we create new attribute vectors by combining new attributes with existing attribute vectors and we generate shares for new attribute vectors by randomizing the shares of the existing vectors. In this way, all attribute vectors in the redefined access structure will obtain functional shares and the access structure need not to be more restrictive than the one of the delegating key. One may attempt to trivially construct APR-ABE from HIBE by directly setting each attribute vector as the identity vector in HIBE. However, this trivial construction would suffer from collusion attacks because a coalition of users may collude to decrypt ciphertexts sent to none of them, even though the access structure of none of the colluders matches the attribute sets of the concerned ciphertexts. The

proposed APR-ABE scheme withstands this kind of collusion attack by associating random values to the secret keys of users. The proposed APR-ABE scheme has short ciphertexts and is proven to be fully secure in the standard model under several static assumptions.

APR-ABE can provide an efficient solution to the motivating application. General attributes can be placed in the first level and more specific, professional attributes in the next level. Alice authorizes doctors to access her medical records by specifying access policies in terms of general attributes. These authorized doctors can redefine their access policies in terms of professional attributes and they can delegate keys to other doctors. The matching doctors then can read Alice's records if their general and specific professional attributes match those specified by the doctors who encrypted Alice's health records.

1.2 Applying APR-ABE to EHR Systems

Our APR-ABE can be applied to EHR systems to circumvent the issue of *a priori* formulation of access policies. The APR-ABE solution relies on cleverly designed attribute hierarchies. We can arrange the attribute universe in a matrix such that general attributes like hospital name (for example "Hospital A", "Hospital B"), title (for example "Professor") or working years are placed in the first level, while specific professional attributes of doctors (typically their medical specialty, with values like "Cardiologist", "Gastroenterologist", etc.) are placed in the next level. When delegating, doctors matching general attributes can redefine their access policies in terms of professional attributes. We now describe how does APR-ABE work for such setting in an EHR system.

As depicted in Fig. 1, an EHR system employs a health record repository to store patients' health records. To protect privacy, all health records are encrypted by doctors who make diagnoses. Suppose that Alice's health records are encrypted with an attribute set $S = \{\text{Hospital A, Cardiologist, Professor, Working years} \geq 3\}$. When Alice feels sick, she wants to authorize some doctors to read her health records. However, she may not know what exact experts are necessary for her diagnosis. Instead of generating secret keys for all doctors of Hospital A, Alice specifies an access policy $\mathbb{A} = \{\text{Hospital A AND Professor AND Working years} \geq 5\}$ and generates a secret key $SK_{\mathbb{A}}$ for a doctor matching this access policy. The matching doctor then makes a preliminary diagnosis on Alice's health records. Upon finding that Alice has a heart condition, the doctor redefines the access policy \mathbb{A} to seek greater specialization, $\mathbb{A}' = \{\{\text{Hospital A, Cardiologist}\} \text{ AND Professor AND Working years} \geq 7\}$ and delegates a secret key for \mathbb{A}' . Since the set S associated with Alice's health records satisfies access structure \mathbb{A}' , the doctor with \mathbb{A}' can decrypt and read Alice's health records. We

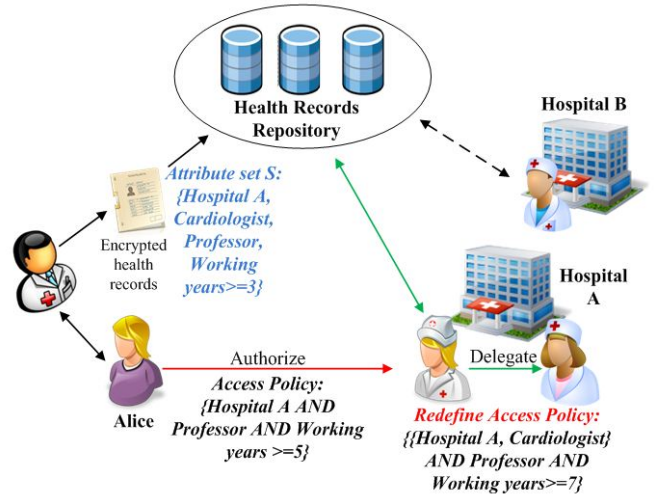


Fig. 1 Application to EHR systems

note that the pair of attributes $\{\text{Hospital A, Cardiologist}\}$ that appears in \mathbb{A}' is treated as an attribute vector in our APR-ABE. Thus in the redefinition of \mathbb{A} as \mathbb{A}' , the new attribute "Cardiologist" can be added, that is, the delegation is *not* more restrictive.

1.3 Paper Organization

The rest of this paper is organized as follows. We recall the related work in Section 2. Section 3 reviews the necessary background for our APR-ABE construction. We formalize the APR-ABE and define its security in Section 4. Section 5 proposes an APR-ABE and proves its security in the standard model. Finally, we conclude the paper in Section 6.

2 Related Work

ABE is a versatile cryptographic primitive allowing fine-grained access control over encrypted files. ABE was introduced by Sahai and Waters [19]. Goyal *et al.* [8] formulated two complementary forms of ABE, i.e., Key-Policy ABE and Ciphertext-Policy ABE, and presented the first KP-ABE scheme. The first CP-ABE scheme was proposed by Bethencourt *et al.* in [3], although its security proof relies on generic bilinear group model. Ostrovsky *et al.* [17] developed a KP-ABE scheme to handle any non-monotone structure; hence, negated clauses can be included in the policies. Waters [21] presented a CP-ABE construction that allows any attribute access structure to be expressed by a Linear Secret Sharing Scheme (LSSS). Attrapadung *et al.* [1] gave a KP-ABE scheme permitting non-monotone access structures and constant-size ciphertexts. To reduce decryption time, Hohenberger and Waters [9] presented a KP-ABE with fast decryption.

The flexible encryption property of ABE made it widely adopted in e-healthcare record systems. Li *et al.* [15] leveraged ABE to encrypt personal health records in cloud computing and exploited multi-authority ABE to achieve a high degree of privacy of records. Yu *et al.* [24] adopted and tailored ABE for wireless sensors of e-healthcare systems. Liang *et al.* [16] also applied ABE to secure private health records in health social networks. In their solution, users can verify each other's identifiers without seeing sensitive attributes, which yields a high level of privacy. Noting that the application of KP-ABE to distributed sensors in e-healthcare systems introduces several challenges regarding attribute and user revocation, Hur [10] proposed an access control scheme using KP-ABE that has efficient attribute and user revocation capabilities.

In order to allow delegation of access rights to encrypted data, some ABE schemes support certain key delegation. CP-ABE [3,7,21] allow users to delegate to attribute sets that are subsets of the original ones. Since a secret sharing scheme is used in key generation, the delegation of KP-ABE is more complicated. Goyal *et al.* [8] adopted Lagrange interpolation to realize secret sharing and achieved a KP-ABE with selective security. This scheme supports key delegation while requiring the tree structures of delegated keys to be more restrictive than the one of the delegating key when new attributes are introduced. Lewko and Waters [13] presented a fully secure KP-ABE which employs a more general LSSS matrix to realize secret sharing. This KP-ABE allows key delegation while requiring the redefined access policy to be either equivalent to the original access policy or more restrictive when new attributes need to be added. The KP-ABE in [18] is an improvement of Lewko and Waters' KP-ABE and inherits its delegation, which is hence limited as well. Recently, Boneh *et al.* [6] proposed an ABE where access policies are expressed as polynomial-size arithmetic circuits. Their system supports key delegation but the size of the secret keys increases quadratically with the number of delegations.

There are some works resolving delegation in different applications. To achieve both fine-grained access control and high performance for enterprise users, Wang *et al.* [23] proposed a solution that combines hierarchical identity-based encryption with CP-ABE to allow a performance-expressivity tradeoff. In that scheme, various authorities rather than attributes are hierarchically organized in order to generate keys for users in their domains. Wan *et al.* [22] extended ciphertext-policy attribute-set-based encryption with a hierarchical structure of users to achieve scalability and flexibility for access control in cloud computing systems. Li *et al.* [14] enhanced ABE by organizing attributes in a tree-like structure to achieve delegation, which is similar to our arrangement of attributes; however, their delegation is still limited to increasingly restrictive access policies. Besides, the security

of the proposed scheme is only selective. Indeed, all these schemes are proposed to adapt ABE for specific applications, while our APR-ABE aims at permitting users to redefine their access policies and delegate secret keys in a way that does not need to be increasingly restrictive.

3 Preliminaries

In this section, we overview access structures, linear secret sharing schemes (LSSS), the composite-order bilinear group equipped with a bilinear map, and several complexity assumptions.

3.1 Access Structures [2]

Definition 1 Let $\{P_1, P_2, \dots, P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\{P_1, P_2, \dots, P_n\}}$ 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 $\{P_1, P_2, \dots, P_n\}$, i.e., $\mathbb{A} \subseteq 2^{\{P_1, P_2, \dots, P_n\}} \setminus \{\emptyset\}$. The sets in \mathbb{A} are called the authorized sets, and the sets not in \mathbb{A} are called the unauthorized sets.

In traditional KP-ABE, the role of the parties is played by the attributes. In our APR-ABE, the role of the parties is taken by attribute vectors. Then an access structure is a collection of sets of attribute vectors. We restrict our attention to monotone access structures in our APR-ABE. However we can realize general access structures by having the negation of an attribute as a separate attribute, at the cost of doubling the number of attributes in the system.

3.2 Linear Secret Sharing Schemes [2]

Definition 2 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 matrix \mathbf{A} called the share-generating matrix for Π , where \mathbf{A} has l rows and n columns. For all $i = 1, \dots, l$, the i -th row of \mathbf{A} is labeled by a party $\rho(i)$, where ρ is a function from $\{1, \dots, l\}$ to \mathcal{P} . When we consider the column vector $\mathbf{s} = (s, s_2, \dots, s_n)$, where $s \in \mathbb{Z}_p$ is the secret to be shared, and $s_2, \dots, s_n \in \mathbb{Z}_p$ are randomly chosen, then $\mathbf{A}\mathbf{s}$ is the vector of l shares of the secret s according to Π . Let A_i denote the i -th row of \mathbf{A} , then $\lambda_i = A_i\mathbf{s}$ is the share belonging to party $\rho(i)$.

Linear Reconstruction. [2] shows that every LSSS Π enjoys the linear reconstruction property. Suppose Π is the LSSS for access structure \mathbb{A} and S is an authorized set in \mathbb{A} ,

i.e., \mathbb{A} contains S . There exist constants $\{\omega_i \in \mathbb{Z}_p\}$ which can be found in time polynomial in the size of the share-generating matrix \mathbf{A} such that if $\{\lambda_i\}$ are valid shares of s , then $\sum_{i \in I} \omega_i \lambda_i = s$, where $I = \{i : \rho(i) \in S\} \subseteq \{1, \dots, \ell\}$.

3.3 Composite-order Bilinear Groups

Suppose that \mathcal{G} is a group generator and ℓ is a security parameter. Composite-order bilinear groups [5] can be defined as $(N = p_1 p_2 p_3, \mathbb{G}, \mathbb{G}_T, e) \leftarrow \mathcal{G}(1^\ell)$, where p_1, p_2 and p_3 are three distinct primes, both \mathbb{G} and \mathbb{G}_T are cyclic groups of order N and the group operations in both \mathbb{G} and \mathbb{G}_T are computable in time polynomial in ℓ . A map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is an efficiently computable map with the following properties.

1. **Bilinearity:** for all $a, b \in \mathbb{Z}_N$ and $g, h \in \mathbb{G}$, $e(g^a, h^b) = e(g, h)^{ab}$.
2. **Non-degeneracy:** $\exists g \in \mathbb{G}$ such that $e(g, g)$ has order N in \mathbb{G}_T .

Let \mathbb{G}_{ij} denote the subgroup of order $p_i p_j$ for $i \neq j$, and $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3$ the subgroups of order p_1, p_2, p_3 in \mathbb{G} , respectively. The *orthogonality* property of $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3$ is defined as:

Definition 3 For all $u \in \mathbb{G}_i, v \in \mathbb{G}_j$, it holds that $e(u, v) = 1$, where $i \neq j \in \{1, 2, 3\}$.

The orthogonality property is essential in our constructions and security proofs.

3.4 Complexity Assumptions

We now list the complexity assumptions which will be used to prove the security of our scheme. These assumptions were introduced by [12] to prove fully secure HIBE and they were also employed by some ABE schemes (e.g., [11, 13]) to attain full security.

Assumption 1 Let $(N = p_1 p_2 p_3, \mathbb{G}, \mathbb{G}_T, e) \leftarrow \mathcal{G}(1^\ell)$. Define a distribution

$$g \xleftarrow{R} \mathbb{G}_1; X_3 \xleftarrow{R} \mathbb{G}_3; \mathcal{D} = (\mathbb{G}, g, X_3); T_1 \xleftarrow{R} \mathbb{G}_1; T_2 \xleftarrow{R} \mathbb{G}_{12}.$$

The advantage of an algorithm \mathcal{A} in breaking Assumption 1 is defined as

$$\text{Adv}_{1, \mathcal{A}}(\ell) = |\Pr[\mathcal{A}(\mathcal{D}, T_1) = 1] - \Pr[\mathcal{A}(\mathcal{D}, T_2) = 1]|.$$

Assumption 1 holds if $\text{Adv}_{1, \mathcal{A}}(\ell)$ is negligible in ℓ for any polynomial-time algorithm \mathcal{A} .

Assumption 2 Let $(N = p_1 p_2 p_3, \mathbb{G}, \mathbb{G}_T, e) \leftarrow \mathcal{G}(1^\ell)$. Define a distribution

$$g, X_1 \xleftarrow{R} \mathbb{G}_1, X_2, Y_2 \xleftarrow{R} \mathbb{G}_2, X_3, Y_3 \xleftarrow{R} \mathbb{G}_3,$$

$$\mathcal{D} = (\mathbb{G}, g, X_1 X_2, X_3, Y_2 Y_3), T_1 \xleftarrow{R} \mathbb{G}, T_2 \xleftarrow{R} \mathbb{G}_{13}.$$

The advantage of an algorithm \mathcal{A} in breaking Assumption 2 is defined as

$$\text{Adv}_{2, \mathcal{A}}(\ell) = |\Pr[\mathcal{A}(\mathcal{D}, T_1) = 1] - \Pr[\mathcal{A}(\mathcal{D}, T_2) = 1]|.$$

Assumption 2 holds if any polynomial-time algorithm \mathcal{A} has $\text{Adv}_{2, \mathcal{A}}(\ell)$ negligible in ℓ .

Assumption 3 Let $(N = p_1 p_2 p_3, \mathbb{G}, \mathbb{G}_T, e) \leftarrow \mathcal{G}(1^\ell)$. Define a distribution

$$\alpha, s \xleftarrow{R} \mathbb{Z}_N, g \xleftarrow{R} \mathbb{G}_1, X_2, Y_2, Z_2 \xleftarrow{R} \mathbb{G}_2, X_3 \xleftarrow{R} \mathbb{G}_3,$$

$$\mathcal{D} = (\mathbb{G}, g, g^\alpha X_2, X_3, g^s Y_2, Z_2), T_1 = e(g, g)^{\alpha s}, T_2 \xleftarrow{R} \mathbb{G}_T.$$

The advantage of an algorithm \mathcal{A} in breaking Assumption 3 is defined as

$$\text{Adv}_{3, \mathcal{A}}(\ell) = |\Pr[\mathcal{A}(\mathcal{D}, T_1) = 1] - \Pr[\mathcal{A}(\mathcal{D}, T_2) = 1]|.$$

Assumption 3 holds if any polynomial-time algorithm \mathcal{A} has $\text{Adv}_{3, \mathcal{A}}(\ell)$ negligible in ℓ .

4 Modeling Access Policy Redefinable Attribute-based Encryption

4.1 Notations

We now model the APR-ABE system. First, we introduce some notations used in the description. Observing that the hierarchical property exists among attributes in the real world, we arrange the APR-ABE attribute universe \mathbf{U} in a matrix with L rows and D columns, that is,

$$\mathbf{U} = (u_{i,j})_{L \times D} = (U_1, \dots, U_i, \dots, U_L)^T,$$

where U_i is the i -th row of \mathbf{U} and contains D attributes and \mathbf{M}^T denotes the transposition of a matrix \mathbf{M} . We note that there may be some *empty* attributes in the matrix. In that case, we use a special character “ \emptyset ” to denote the empty attributes.

The attribute matrix naturally leads to the notion of attribute vector. We define an attribute vector of depth k ($1 \leq k \leq L$) as

$$\mathbf{u} = (u_1, u_2, \dots, u_k),$$

where $u_i \in U_i$ for each i from 1 to k . This means that an attribute vector of depth k is formed by sampling single attributes from the first level to the k -th level. We note that each attribute u_i actually corresponds to two subscripts (i, j)

denoting its position in the attribute matrix, but we drop the second subscript j here to simplify notations.

We next define a set of attribute vectors. Let $S = \{\mathbf{u}\}$ denote a set of attribute vectors of depth k and $|S|$ denote the set's cardinality.

For an attribute vector \mathbf{u}' of depth i and another attribute vector \mathbf{u} of depth k , we say that \mathbf{u}' is a *prefix* of \mathbf{u} if $\mathbf{u} = (\mathbf{u}', u_{i+1}, u_{i+2}, \dots, u_k)$, where $1 \leq i < k \leq L$.

As in Definition 1, we can define \mathbb{A} as an access structure over attribute vectors of depth k such that \mathbb{A} is a collection of non-empty subsets of the set of all attribute vectors of depth k . If for a set S the condition $S \in \mathbb{A}$ holds, then we say that S is an authorized set in \mathbb{A} and S satisfies \mathbb{A} .

In an APR-ABE system, a secret key associated with an access structure \mathbb{A} can decrypt a ciphertext generated with a set S of attribute vectors if and only if $S \in \mathbb{A}$. A secret key associated with an access structure \mathbb{A}' is allowed to delegate a secret key for an access structure \mathbb{A} if these two access structures satisfy a natural condition. That is, each attribute vector of a set $S' \in \mathbb{A}'$ must be a prefix of an attribute vector in some set $S \in \mathbb{A}$ and all attribute vectors involved in \mathbb{A} have prefixes in \mathbb{A}' . This guarantees that the user with access structure \mathbb{A}' can use his existing shares to generate shares for attribute vectors of authorized sets in \mathbb{A} . We note that in the delegation there is no requirement that the redefined access structure \mathbb{A} must be more restrictive than the original access structure \mathbb{A}' when new attributes are added. This is because those new attributes can be concatenated to the end of existing attribute vectors of \mathbb{A}' instead of being treated as new separate attributes that need to be assigned to new secret shares.

4.2 System Model

An APR-ABE system for message space \mathbb{M} and access structure space Γ consists of the following five polynomial-time algorithms:

- $(PK, MSK) \leftarrow \mathbf{Setup}(1^\ell)$: The setup algorithm takes no input other than the security parameter ℓ and outputs the public key PK and a master secret key MSK .
- $CT \leftarrow \mathbf{Encrypt}(M, PK, S)$: The encryption algorithm takes as inputs a message M , the public key PK and a set S of attribute vectors. It outputs a ciphertext CT .
- $SK \leftarrow \mathbf{KeyGen}(PK, MSK, \mathbb{A})$: The key generation algorithm takes as inputs an access structure \mathbb{A} , the master secret key MSK and public key PK . It outputs a secret key SK for the access structure \mathbb{A} .
- $SK \leftarrow \mathbf{Delegate}(PK, SK', \mathbb{A})$: The delegation algorithm takes as inputs a public key PK , a secret key SK' for an access structure \mathbb{A}' and another access structure \mathbb{A} . It outputs the secret key SK for \mathbb{A} if and only if \mathbb{A} and \mathbb{A}' satisfy the delegation condition.
- $M/\perp \leftarrow \mathbf{Decrypt}(CT, SK, PK)$: The decryption algorithm takes as inputs a ciphertext CT associated with a set S of attribute vectors, a secret key for an access structure \mathbb{A} , and the public key PK . If $S \in \mathbb{A}$, it outputs M ; otherwise, it outputs a false symbol \perp .

The correctness property requires that for all sufficiently large $\ell \in \mathbb{N}$, all universe descriptions \mathbb{U} , all $(PK, MSK) \leftarrow \mathbf{Setup}(1^\ell)$, all $\mathbb{A} \in \Gamma$, all $SK \leftarrow \mathbf{KeyGen}(PK, MSK, \mathbb{A})$ or $SK \leftarrow \mathbf{Delegate}(PK, SK', \mathbb{A})$, all $M \in \mathbb{M}$, all $CT \leftarrow \mathbf{Encrypt}(M, PK, S)$, if S satisfies \mathbb{A} , then $\mathbf{Decrypt}(CT, SK, PK)$ outputs M .

4.3 Security

We now define the full security against chosen access structure and chosen-plaintext attacks in APR-ABE. In practice, malicious users are able to obtain the system public key and, additionally, they may collude with other users by querying their secret keys. To capture these realistic attacks, we define an adversary allowed to access the system public key, create attribute vectors and query secret keys for access structures he specifies. The adversary outputs two equal-length messages and a set of attribute vectors to be challenged. Then the full security states that not even such an adversary can distinguish with non-negligible advantage the ciphertexts of the two messages under the challenge set of attribute vectors, provided that he has not queried the secret keys that can be used to decrypt the challenge ciphertext. Formally, the full security of APR-ABE is defined by a game played between a challenger \mathcal{C} and an adversary \mathcal{A} as follows.

- **Setup**: The challenger \mathcal{C} runs the setup algorithm and gives the public key PK to \mathcal{A} .
- **Phase 1**: \mathcal{A} sequentially makes queries Q_1, \dots, Q_{q_1} to \mathcal{C} , where Q_i for $1 \leq i \leq q_1$ is one of the following three types:
 - **Create**(\mathbb{A}). \mathcal{A} specifies an access structure \mathbb{A} . In response, \mathcal{C} creates a key for this access structure by calling the key generation algorithm, and places this key in the set \mathcal{K} which is initialized to empty. He only gives \mathcal{A} a reference to this key, not the key itself.
 - **Delegate**(\mathbb{A}, \mathbb{A}'). \mathcal{A} specifies a key SK' associated with \mathbb{A}' in the set \mathcal{K} and an access structure \mathbb{A} . If allowed by the delegation algorithm, \mathcal{C} produces a key SK for \mathbb{A} . He adds SK to the set \mathcal{K} and again gives \mathcal{A} only a reference to it, not the actual key.
 - **Reveal**(\mathbb{A}). \mathcal{A} specifies a key in the set \mathcal{K} . \mathcal{C} gives this key to the attacker and removes it from the set \mathcal{K} .
- **Challenge**: \mathcal{A} declares two equal-length messages M_0 and M_1 and a set S^* of attribute vectors with an added

restriction that for any revealed key SK for access structure \mathbb{A} , $S^* \notin \mathbb{A}$ and for any new key SK' for access structure \mathbb{A}' that can be delegated from a revealed one, $S^* \notin \mathbb{A}'$. \mathcal{C} then flips a random coin $b \in \{0, 1\}$, and encrypts M_b under S^* , producing CT^* . He gives CT^* to \mathcal{A} .

- **Phase 2:** \mathcal{A} sequentially makes queries Q_{q_1+1}, \dots, Q_q to \mathcal{C} just as in Phase 1, with the restriction that neither the access structure of any revealed key nor the access structure of any key that can be delegated from a revealed one contain S^* .
- **Guess:** \mathcal{A} outputs a guess $b' \in \{0, 1\}$.

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

$$Adv_{\mathcal{A}}^{\text{APR-ABE}} = |\Pr[b = b'] - 1/2|.$$

We note that the model above is for *chosen-plaintext* attacks and one can easily extend this model to handle *chosen-ciphertext* attacks by allowing decryption queries in Phase 1 and Phase 2.

Definition 4 We say that an APR-ABE system is fully secure if all Probabilistic Polynomial-Time (PPT) attackers \mathcal{A} have at most a negligible advantage in the above game.

5 The Access Policy Redefinable Attribute-based Encryption Scheme

In this section, we construct an APR-ABE with short ciphertexts. The proposed scheme is proven to be fully secure in the standard model.

5.1 Basic Idea

We first introduce the basic idea driving the construction of the APR-ABE scheme. We base the scheme on the KP-ABE scheme in [11] and we exploit the delegation mechanism used in several HIBE schemes (e.g., [4, 12]). The key point of this delegation mechanism is to hash an identity vector to a group element, which internally associates the identity vector with a ciphertext or a secret key. When introducing this mechanism into our APR-ABE, which involves multiple attribute vectors in a ciphertext or a secret key, we assign a key component to each attribute vector and randomize every key component to resist collusion attacks.

On the other hand, LSSS have been widely used in many ABE schemes [1, 11, 13, 21]. In our APR-ABE scheme, an LSSS is used to generate a share for each attribute vector of authorized sets in an access structure. The linear reconstruction property of LSSS guarantees that the shares of all attribute vectors in an authorized set can recover the secret.

To realize a delegation not limited to more restrictive access policies, we must additionally manage to generate shares for new incoming attributes. However, without knowing the secret, delegators cannot directly generate new shares. To overcome this problem, we concatenate the new incoming attributes to the end of existing attribute vectors to form new attribute vectors and use the existing shares to generate shares for the new attribute vectors. Specifically, to achieve the access structure control, each share of an attribute vector is blinded in the exponent of a key component. Then, to generate new shares, we lift a key component of an existing attribute vector to the power of a random exponent and define the resulting exponent as the new blinded share for the new attribute vector. Since LSSS satisfies linearity, the randomization of shares can still reconstruct the secret.

To realize the above idea, we slightly extend LSSS to handle attribute vectors. For an access structure \mathbb{A} , we generate an $l \times n$ share-generating matrix \mathbf{A} (l is the number of attribute vectors involved in \mathbb{A}). The inner product of the i -th row vector of \mathbf{A} and a vector taking the secret as the first coordinate is the share for the i -th row. We define an *injection* function ρ which maps the i -th row of the matrix \mathbf{A} to an attribute vector. Then (\mathbf{A}, ρ) is the LSSS for \mathbb{A} . The *injection* function means that an attribute vector is associated with at most one row of \mathbf{A} .

5.2 The Proposal

We are now ready to describe our APR-ABE scheme, which is built from bilinear groups of a composite order $N = p_1 p_2 p_3$, as defined in Section 2.3. The ciphertexts are generated in the subgroup \mathbb{G}_1 . The keys are first generated in \mathbb{G}_1 and then randomized in \mathbb{G}_3 . The subgroup \mathbb{G}_2 is only used to implement semi-functionality in the security proofs.

- $(PK, MSK) \leftarrow \text{Setup}(1^\ell)$: Run $(N = p_1 p_2 p_3, \mathbb{G}, \mathbb{G}_T, e) \xleftarrow{R} \mathcal{G}(1^\ell)$. Let \mathbb{G}_i denote the subgroup of order p_i for $i = 1, 2, 3$. Choose random generators $g \in \mathbb{G}_1, X_3 \in \mathbb{G}_3$. Choose random elements $\alpha \in \mathbb{Z}_N, v_i, h_j \in \mathbb{G}_1$ for all $i = 1, \dots, D$ and $j = 1, \dots, L$. The public key and the master secret key are

$$PK = (\mathbf{U}, N, g, X_3, v_1, \dots, v_D, h_1, \dots, h_L, e(g, g)^\alpha),$$

$$MSK = \alpha.$$

- $CT \leftarrow \text{Encrypt}(M, PK, S)$: Encrypt a message M under a set S of attribute vectors of depth k . Choose a random $s \in \mathbb{Z}_N$ and compute

$$C = M e(g, g)^{\alpha s}, E = g^s.$$

For each j from 1 to $|S|$, choose a random element $t_j \in \mathbb{Z}_N$. Recall that for each attribute vector $\mathbf{u} = (u_1, u_2, \dots, u_k)$ of S , the first coordinate u_1 actually has two subscripts,

denoted by $(1, x)$, representing that u_1 is the x -th entry of the first row in the attribute matrix. Then, choose v_x corresponding to x from the public key and compute

$$C_{j,0} = v_x^s (h_1^{u_1} \cdots h_k^{u_k})^{st_j}, C_{j,1} = g^{st_j}.$$

Define the ciphertext (including S) as

$$CT = (C, E, \langle C_{j,0}, C_{j,1} \rangle_{j=1, \dots, |S|}).$$

- $SK \leftarrow \mathbf{KeyGen}(PK, MSK, \mathbb{A})$: The algorithm generates an LSSS (\mathbf{A}, ρ) for \mathbb{A} , where \mathbf{A} is the share-generating matrix with l rows and n columns, and ρ maps each row of \mathbf{A} to an attribute vector of depth k . Choose $n - 1$ random elements $s_2, \dots, s_n \in \mathbb{Z}_N$ to form a vector

$$\alpha = (\alpha, s_2, \dots, s_n).$$

For each i from 1 to l , compute $\lambda_i = A_i \alpha$, where A_i is the i -th row vector of \mathbf{A} . Let $\mathbf{u} = (u_1, \dots, u_k)$ be the attribute vector mapped by ρ from the i -th row. Assume that the first coordinate u_1 of \mathbf{u} is the x -th entry of the first row in the attribute matrix and choose v_x correspondingly. Then, select random elements $r_i \in \mathbb{Z}_N$ and $R_{i,0}, R_{i,1}, R_{i,2}, R_{i,k+1}, \dots, R_{i,L} \in \mathbb{G}_3$ to compute

$$K_{i,0} = g^{\lambda_i} v_x^{r_i} R_{i,0}, \quad K_{i,1} = g^{r_i} R_{i,1},$$

$$K_{i,2} = (h_1^{u_1} \cdots h_k^{u_k})^{r_i} R_{i,2},$$

$$K_{i,k+1} = h_{k+1}^{r_i} R_{i,k+1}, \quad \dots, \quad K_{i,L} = h_L^{r_i} R_{i,L}.$$

Set the secret key (including (\mathbf{A}, ρ)) to be

$$SK = \langle K_{i,0}, K_{i,1}, K_{i,2}, K_{i,k+1}, \dots, K_{i,L} \rangle_{i=1, \dots, l}.$$

- $SK \leftarrow \mathbf{Delegate}(PK, SK', \mathbb{A})$: The algorithm generates a secret key SK for \mathbb{A} by using the secret key

$$SK' = \langle K'_{i',0}, K'_{i',1}, K'_{i',2}, K'_{i',k+1}, \dots, K'_{i',L} \rangle_{i'=1, \dots, l'}$$

for \mathbb{A}' , where \mathbb{A}' is an access structure over l' attribute vectors of depth k and \mathbb{A} is an access structure over l attribute vectors of depth $k + 1$. If \mathbb{A} and \mathbb{A}' satisfy the delegation condition, the algorithm works as follows.

For each \mathbf{u} involved in \mathbb{A} , find its prefix \mathbf{u}' in \mathbb{A}' such that $\mathbf{u} = (\mathbf{u}', u_{k+1})$. Suppose that \mathbf{u}' is associated with the i' -th row of the share-generating matrix of \mathbb{A}' . Choose random elements $\gamma_i, \delta_i \in \mathbb{Z}_N$ and random group elements $R_{i,0}, R_{i,1}, R_{i,2}, R_{i,k+2}, \dots, R_{i,L} \in \mathbb{G}_3$ for each i from 1 to l . Then pick the key component $(K'_{i',0}, K'_{i',1}, K'_{i',2}, K'_{i',k+1}, \dots, K'_{i',L})$ of \mathbf{u}' from SK' to compute the key component for \mathbf{u} :

$$K_{i,0} = (K'_{i',0})^{\gamma_i} v_x^{\delta_i} R_{i,0},$$

$$K_{i,1} = (K'_{i',1})^{\gamma_i} g^{\delta_i} R_{i,1},$$

$$K_{i,2} = (K'_{i',2})^{\gamma_i} (K'_{i',k+1})^{\gamma_i u_{k+1}} (h_1^{u_1} \cdots h_{k+1}^{u_{k+1}})^{\delta_i} R_{i,2},$$

$$K_{i,k+2} = (K'_{i',k+2})^{\gamma_i} h_{k+2}^{\delta_i} R_{i,k+2}, \quad \dots,$$

$$K_{i,L} = (K'_{i',L})^{\gamma_i} h_L^{\delta_i} R_{i,L}.$$

This implicitly sets $r_i = \gamma_i r'_{i'} + \delta_i$, where $r'_{i'}$ is the random exponent used in creating the key component for \mathbf{u}' . The value r_i is random since δ_i is picked randomly. Finally, output

$$SK = \langle K_{i,0}, K_{i,1}, K_{i,2}, K_{i,k+2}, \dots, K_{i,L} \rangle_{i=1, \dots, l}.$$

Note that this key is identically distributed as the one directly generated by **KeyGen**.

- $M \leftarrow \mathbf{Decrypt}(CT, SK, PK)$: Given a ciphertext $CT = (C, E, \langle C_{j,0}, C_{j,1} \rangle_{j=1, \dots, |S|})$ for S of attribute vectors of depth k and a secret key

$$SK = \langle K_{i,0}, K_{i,1}, K_{i,2}, K_{i,k+1}, \dots, K_{i,L} \rangle_{i=1, \dots, l}$$

for access structure \mathbb{A} over attribute vectors of depth k , if $S \in \mathbb{A}$, compute the constants $\{\omega_i \in \mathbb{Z}_N\}_{\rho(i) \in S}$ such that

$$\sum_{\rho(i) \in S} \omega_i A_i = (1, 0, \dots, 0).$$

Let $\rho(i)$ be the j -th attribute vector in S . Compute:

$$M' = \prod_{\rho(i) \in S} \left(\frac{e(E, K_{i,0}) \cdot e(C_{j,1}, K_{i,2})}{e(C_{j,0}, K_{i,1})} \right)^{\omega_i}.$$

Output $M = C/M'$.

Remark 1 In the key delegation, when delegating a secret key for \mathbb{A} from a secret key for \mathbb{A}' , an LSSS (\mathbf{A}, ρ) for \mathbb{A} is simultaneously generated: the share-generating matrix \mathbf{A} is formed by setting the i -th row as $A_i = A'_{i'} \gamma_i$, where $A'_{i'}$ is the i' -th row of the share-generating matrix of \mathbf{A}' ; the function ρ maps the i -th row to the attribute vector \mathbf{u} . The value $\lambda_i = \gamma_i \lambda'_{i'} = \gamma_i A'_{i'} \alpha = A_i \alpha$ is the share for \mathbf{u} , where $\lambda'_{i'}$ is the share for \mathbf{u}' .

Correctness. Observe that

$$M' =$$

$$\prod_{\rho(i) \in S} \left(\frac{e(g^s, g^{\lambda_i}) \cdot e(g^s, v_x^{r_i}) \cdot e(g^{st_j}, (h_1^{u_1} \cdots h_k^{u_k})^{r_i})}{e(v_x^s, g^{r_i}) \cdot e((h_1^{u_1} \cdots h_k^{u_k})^{st_j}, g^{r_i})} \right)^{\omega_i}$$

$$= e(g, g)^{s \sum_{\rho(i) \in S} \omega_i A_i \alpha} = e(g, g)^{s \alpha}.$$

It follows that $M = C/M'$. The \mathbb{G}_3 parts are canceled out because of the orthogonality property.

Table 1 Computation

Algorithm	Computational Complexity
Key Generation	$(L + 3) \cdot l \cdot t_e$
Key Delegation	$(2L - k + 5) \cdot l' \cdot t_e$
Encryption	$((k + 2) S + 2) \cdot t_e$
Decryption	$3l^* \cdot t_p$

5.2.1 Computational Complexity

We analyze the computational complexity of the main algorithms of the APR-ABE scheme, i.e., key generation, key delegation, encryption and decryption. The proposed scheme is built in bilinear groups \mathbb{G} and \mathbb{G}_T , and most computations take place in the subgroup \mathbb{G}_1 . Therefore we evaluate the times t_p and t_e consumed by the basic group operations, bilinear map and exponentiation in \mathbb{G}_1 , respectively. We do not take into account the multiplication operation since it consumes negligible time compared to t_p and t_e .

Table 1 summarizes the time consumed by the main algorithms of the APR-ABE scheme. In this table, L denotes the maximum depth of the system, l the number of attribute vectors associated with a secret key, l' the number of attribute vectors associated with a delegated key, k the depth of the user delegating a key or the attribute vectors associated with a ciphertext, and l^* is the number of attribute vectors of a set satisfying an access policy in the decryption. We can see that the time cost by the key generation algorithm grows linearly with the product of L and l , but is independent of the user's depth. The time consumed by the delegation is related to the depth of the delegator and decreases as the depth grows. Encryption takes time linear in the product of the cardinality of the set S and the depth of the attribute vectors in S . The ciphertexts of APR-ABE are short in that they are only linear in the cardinality of S . This makes the time consumed by decryption linear in the number of matching attribute vectors and independent of depth. This feature is comparable to the up-to-date KP-ABEs [8, 11, 9], which nonetheless do not allow the flexible key delegation achieved in our scheme.

5.2.2 Security

The new APR-ABE scheme has full security, which means that any polynomial-time attacker cannot get useful information about the messages encrypted in ciphertexts if he does not have correct secret keys. Formally, the full security is guaranteed by Theorem 1.

Theorem 1 *The Access Policy Redefinable Attribute-based Encryption scheme is fully secure in the standard model if Assumptions 1, 2 and 3 hold.*

Our proof exploits the dual system encryption methodology [20]. This approach has been shown to be a powerful tool in proving the full security of properly designed HIBE and ABE schemes (e.g., [12, 13, 11, 20]). Following this proof framework, we construct semi-functional keys and ciphertexts for APR-ABE. A semi-functional APR-ABE key (semi-functional key for short) can be used to decrypt normal ciphertexts; and a semi-functional APR-ABE ciphertext (semi-functional ciphertext for short) can be decrypted by using normal keys. However, a semi-functional key cannot be used to decrypt a semi-functional ciphertext. As in most proofs adopting dual system encryption, there is a subtlety that the simulator can test the nature of the challenge key by using it to try to decrypt the challenge ciphertext. To avoid this paradox, we make sure that the decryption on input the challenge key is always successful by cleverly setting the random values involved in the challenge key and challenge ciphertext. We also need to prove that these values are uniformly distributed from the view of the adversary who cannot query the key able to decrypt the ciphertext.

In the following proof, we define a sequence of games arguing that an attacker cannot distinguish one game from the next. The first game is $Game_{real}$, which denotes the real security game as defined in Definition 4. The second game is $Game_{real'}$, which is the same as $Game_{real}$ except that the attacker \mathcal{A} does not ask the challenger \mathcal{C} to delegate keys. The third game is $Game_0$, in which all keys are normal, but the challenge ciphertext is semi-functional. Let q denote the number of key queries made by \mathcal{A} . For all $\nu = 1, \dots, q$, we define $Game_\nu$, in which the first ν keys are semi-functional and the remaining keys are normal, while the challenge ciphertext is semi-functional. Note that when $\nu = q$, in $Game_q$, all keys are semi-functional. The last game is defined as $Game_{final}$ where all keys are semi-functional and the ciphertext is a semi-functional encryption of a random message. We will prove that these games are indistinguishable under Assumptions 1, 2 and 3.

The semi-functional ciphertexts and keys are constructed as follows.

Semi-functional ciphertext. Let g_2 denote the generator of \mathbb{G}_2 . We first invoke **Encrypt** to form a normal ciphertext $(\bar{C}, \bar{E}, \langle \bar{C}_{i^*,0}, \bar{C}_{i^*,1} \rangle_{i^*=1, \dots, |S^*|})$. We choose a random element $c \in \mathbb{Z}_N$ and for all $i^* = 1, \dots, |S^*|$, select random exponents $\varphi_{i^*}, v_{i^*} \in \mathbb{Z}_N$. Set the semi-functional ciphertext to be

$$C = \bar{C}, E = \bar{E}g_2^c, C_{i^*,0} = \bar{C}_{i^*,0}g_2^{\varphi_{i^*}}, C_{i^*,1} = \bar{C}_{i^*,1}g_2^{v_{i^*}}.$$

Semi-functional key. We first call algorithm **KeyGen** to form normal key $\langle \bar{K}_{i,0}, \bar{K}_{i,1}, \bar{K}_{i,2}, \bar{K}_{i,k+1}, \dots, \bar{K}_{i,L} \rangle_{i=1, \dots, l}$. Then we choose random elements $f_i \in \mathbb{Z}_N$ for the i -th row of the share-generating matrix \mathbf{A} . We choose random ele-

ments $\zeta_1, \zeta_2, \dots, \zeta_D, \eta_1, \eta_2, \dots, \eta_L \in \mathbb{Z}_N$ and a random vector $\vartheta \in \mathbb{Z}_N^n$. The semi-functional key is set as:

$$K_{i,0} = \bar{K}_{i,0} g_2^{A_i \vartheta + f_i \zeta_x}, \quad K_{i,1} = \bar{K}_{i,1} g_2^{f_i},$$

$$K_{i,2} = \bar{K}_{i,2} g_2^{f_i \sum_{j=1}^k u_j \eta_j},$$

$$K_{i,k+1} = \bar{K}_{(i,k+1)} g_2^{f_i \eta_{k+1}}, \dots, K_{i,L} = \bar{K}_{(i,L)} g_2^{f_i \eta_L}.$$

Remark 2 When we use a semi-functional key to decrypt a semi-functional ciphertext, we will have an extra term

$$\prod_{\rho(i) \in S} \left(e(g_2, g_2)^{c A_i \vartheta} e(g_2, g_2)^{f_i (c \zeta_x + v_{i^*} \sum_{j=1}^k u_j \eta_j - \varphi_{i^*})} \right)^{\omega_i}.$$

If $\vartheta \cdot (1, 0, \dots, 0) = 0 \pmod{p_2}$ and $c \zeta_x + v_{i^*} \sum_{j=1}^k u_j \eta_j - \varphi_{i^*} = 0 \pmod{p_2}$, then the extra term happens to be one, which means that the decryption still works. We say that the keys satisfying this condition are *nominally* semi-functional keys. We will show that a nominally semi-functional key is identically distributed as a regular semi-functional key in the attacker's view.

Lemma 1 For any attacker \mathcal{A} ,

$$Game_{real} Adv_{\mathcal{A}} = Game_{real'} Adv_{\mathcal{A}}.$$

Proof From the construction of our APR-ABE, the keys from the key generation algorithm are identically distributed as the keys from the delegation algorithm. Therefore, in \mathcal{A} 's view, there is no difference between these two kinds of games. \square

Lemma 2 If \mathcal{A} can distinguish $Game_{real'}$ from $Game_0$ with advantage ϵ , then we can establish an algorithm \mathcal{B} to break Assumption 1 with advantage ϵ .

Proof We construct an algorithm \mathcal{B} to simulate $Game_{real'}$ or $Game_0$ to interact with \mathcal{A} by using the tuple (g, X_3, T) of Assumption 1.

Setup: Algorithm \mathcal{B} selects a random $\alpha \in \mathbb{Z}_N$. For all $i = 1, \dots, D$ and $j = 1, \dots, L$, it chooses random elements $\bar{\zeta}_i, \bar{\eta}_j \in \mathbb{Z}_N$ and computes $v_i = g^{\bar{\zeta}_i}, h_j = g^{\bar{\eta}_j}$. It provides \mathcal{A} with public key:

$$PK = (\mathbf{U}, N, g, v_1, \dots, v_D, h_1, \dots, h_L, e(g, g)^\alpha).$$

Key generation Phase 1, Phase 2: Note that \mathcal{B} knows the master key $MSK = \alpha$. Therefore, \mathcal{B} can run **KeyGen** to generate normal keys in Phase 1 and Phase 2.

Challenge: \mathcal{A} gives two equal-length messages M_0 and M_1 , and a set $S^* = \{\mathbf{u}\}$ of attribute vectors to \mathcal{B} . \mathcal{B} then uses the T in the given tuple to form a semi-functional or normal ciphertext as follows.

\mathcal{B} flips a random coin $b \in \{0, 1\}$. For all $i^* = 1, \dots, |S^*|$, it chooses random elements $t_{i^*} \in \mathbb{Z}_N$. Finally, it sets the semi-functional ciphertext CT to be:

$$C = M_b e(g, T)^\alpha, \quad E = T,$$

$$C_{i^*,0} = T^{\bar{\zeta}_x} T^{(\bar{\eta}_1 u_1 + \dots + \bar{\eta}_k u_k) t_{i^*}}, \quad C_{i^*,1} = T^{t_{i^*}}.$$

If assuming $T = g^s g_2^c$, this implicitly sets

$$\varphi_{i^*} = c(\bar{\zeta}_x + t_{i^*} \sum_{j=1}^k u_j \bar{\eta}_j), \quad v_{i^*} = c t_{i^*},$$

but there is neither unwanted correlation between values $(\varphi_{i^*} \pmod{p_2})$ and values $(\bar{\zeta}_x, \bar{\eta}_j \pmod{p_2})$, nor correlation between $(t_{i^*} \pmod{p_2})$ and $(v_{i^*} \pmod{p_2})$ by the Chinese Remainder Theorem. Thus, the \mathbb{G}_1 part of the ciphertext is unrelated to the \mathbb{G}_2 part.

Guess: If $T \in \mathbb{G}_{12}$, CT is a properly distributed semi-functional ciphertext. Hence we are in $Game_0$. If $T \in \mathbb{G}_1$, by implicitly setting $T = g^s$, CT is a properly distributed normal ciphertext. Hence we are in $Game_{real'}$. If \mathcal{A} outputs b' such that $b' = b$, then \mathcal{B} outputs 0. Therefore, with the tuple (g, X_3, T) , we have that the advantage of \mathcal{B} in breaking Assumption 1 is

$$\begin{aligned} & |\Pr[\mathcal{B}(g, X_3, T \in \mathbb{G}_{12}) = 0] - \Pr[\mathcal{B}(g, X_3, T \in \mathbb{G}_1) = 0]| \\ &= |Game_0 Adv_{\mathcal{A}} - Game_{real'} Adv_{\mathcal{A}}| = \epsilon, \end{aligned}$$

where $Game_0 Adv_{\mathcal{A}}$ is the advantage of \mathcal{A} in $Game_0$ and $Game_{real'} Adv_{\mathcal{A}}$ is the advantage of \mathcal{A} in $Game_{real'}$. \square

Lemma 3 If \mathcal{A} can distinguish $Game_{\nu-1}$ from $Game_\nu$ with advantage ϵ , then we can construct an algorithm \mathcal{B} to break Assumption 2 with advantage ϵ .

Proof We construct an algorithm \mathcal{B} to simulate $Game_{\nu-1}$ or $Game_\nu$ to interact with \mathcal{A} by using the tuple $(g, X_1 X_2, X_3, Y_2 Y_3, T)$ of Assumption 2.

Setup: The public key PK generated by \mathcal{B} is the same as that in Lemma 2. Algorithm \mathcal{B} gives PK to \mathcal{A} .

Challenge: For convenience, we bring the Challenge phase before Phase 1. This will not affect the security proof. When \mathcal{A} queries the challenge ciphertext with two equal-size messages M_0, M_1 and a set S^* of attribute vectors, \mathcal{B} flips a random coin $b \in \{0, 1\}$ and randomly chooses $t_{i^*} \in \mathbb{Z}_N$ for all $i^* = 1, \dots, |S^*|$. It sets the ciphertext to be

$$C = M_b e(g, X_1 X_2)^\alpha, \quad E = X_1 X_2,$$

$$C_{i^*,0} = (X_1 X_2)^{\bar{\zeta}_x} (X_1 X_2)^{(\bar{\eta}_1 u_1 + \dots + \bar{\eta}_k u_k) t_{i^*}},$$

$$C_{i^*,1} = (X_1 X_2)^{t_{i^*}}.$$

By assuming $X_1 X_2 = g^s g_2^c$, this implicitly sets

$$\varphi_{i^*} = c(\bar{\zeta}_x + t_{i^*} \sum_{j=1}^k u_j \bar{\eta}_j), v_{i^*} = ct_{i^*}.$$

Again there is no correlation between values $(\varphi_{i^*} \bmod p_2)$ and values $(\bar{\zeta}_x, \bar{\eta}_j \bmod p_2)$, nor is there any correlation between $(t_{i^*} \bmod p_2)$ and $(v_{i^*} \bmod p_2)$ by the Chinese Remainder Theorem. Thus the \mathbb{G}_1 part is unrelated to the \mathbb{G}_2 part of this ciphertext. Therefore, this ciphertext is a well distributed semi-functional ciphertext.

Key generation Phase 1, Phase 2: For the first $\nu - 1$ key queries, \mathcal{B} simulates the semi-functional keys. For a queried \mathbb{A} , it first calls the key generation algorithm to generate an LSSS (\mathbf{A}, ρ) and a normal key $(\bar{K}_{i,0}, \bar{K}_{i,1}, \bar{K}_{i,2}, \bar{K}_{i,k+1}, \dots, \bar{K}_{i,L})_{\forall i \in [l]}$ for this LSSS. Then, for each i from 1 to l , \mathcal{B} picks a random element $\bar{f}_i \in \mathbb{Z}_N$. \mathcal{B} also chooses random elements $\zeta_1, \dots, \zeta_D, \eta_1, \dots, \eta_L \in \mathbb{Z}_N$. Finally \mathcal{B} chooses a random vector $\bar{\vartheta} \in \mathbb{Z}_N^n$ and computes the secret key:

$$\begin{aligned} K_{i,0} &= \bar{K}_{i,0} (Y_2 Y_3)^{A_i \bar{\vartheta} + \bar{f}_i \zeta_x}, K_{i,1} = \bar{K}_{i,1} (Y_2 Y_3)^{\bar{f}_i}, \\ K_{i,2} &= \bar{K}_{i,2} (Y_2 Y_3)^{\bar{f}_i \sum_{j=1}^k u_j \eta_j}, \\ K_{i,k+1} &= \bar{K}_{i,k+1} (Y_2 Y_3)^{\bar{f}_i \eta_{k+1}}, \dots, \\ K_{i,L} &= \bar{K}_{i,L} (Y_2 Y_3)^{\bar{f}_i \eta_L}. \end{aligned}$$

If we assume $Y_2 = g_2^a$ for some a , this implicitly sets $\vartheta = a\bar{\vartheta}$, $f_i = a\bar{f}_i$. Thus this key is identically distributed as the semi-functional key.

For the rest of key queries but the ν -th one, \mathcal{B} simulates normal keys. Because \mathcal{B} knows the master key $MSK = \alpha$, it can easily create the normal keys by running the key generation algorithm.

To respond to the ν -th key query on an access structure \mathbb{A} , algorithm \mathcal{B} will either simulate a normal key or a semi-functional key depending on T . Algorithm \mathcal{B} generates an LSSS for \mathbb{A} to prepare for key generation. It creates a vector $\bar{\alpha}$ with the first coordinate equal to α and the remaining $n - 1$ coordinates picked randomly in \mathbb{Z}_N . \mathcal{B} also creates a vector $\bar{\vartheta}$ with the first coordinate equal to 0 and the remaining $n - 1$ coordinates picked randomly in \mathbb{Z}_N . For each row A_i of \mathbf{A} , \mathcal{B} chooses random elements $\bar{r}_i \in \mathbb{Z}_N$ and $R_{i,0}, R_{i,1}, R_{i,2}, R_{i,k+1}, \dots, R_{i,L} \in \mathbb{G}_3$. Then \mathcal{B} computes:

$$\begin{aligned} K_{i,0} &= g^{A_i \bar{\alpha}} T^{A_i \bar{\vartheta}} T^{\bar{r}_i \zeta_x} R_{i,0}, K_{i,1} = T^{\bar{r}_i} R_{i,1}, \\ K_{i,2} &= T^{(\bar{\eta}_1 u_1 + \dots + \bar{\eta}_k u_k) \bar{r}_i} R_{i,2}, \\ K_{i,k+1} &= T^{\bar{r}_i \bar{\eta}_{k+1}} R_{i,k+1}, \dots, K_{i,L} = T^{\bar{r}_i \bar{\eta}_L} R_{i,L}. \end{aligned}$$

If $T \in \mathbb{G}_{13}$, by assuming $T = g^{c_1} g_3^{c_3}$, this implicitly sets $r_i = c_1 \bar{r}_i$ and $\alpha = \bar{\alpha} + c_1 \bar{\vartheta}$. Thus this key is identically distributed as the normal key. If $T \in \mathbb{G}$, by assuming $T = g^{c_1} g_2^{c_2} g_3^{c_3}$, this implicitly sets:

$$r_i = c_1 \bar{r}_i, f_i = c_2 \bar{r}_i, \vartheta = c_2 \bar{\vartheta}, \alpha = \bar{\alpha} + c_1 \bar{\vartheta},$$

and $\zeta_1 = \bar{\zeta}_1, \dots, \zeta_D = \bar{\zeta}_D, \eta_1 = \bar{\eta}_1, \dots, \eta_L = \bar{\eta}_L$. Since r_i are created by \bar{r}_i in \mathbb{G}_1 and f_i are created by \bar{r}_i in \mathbb{G}_2 , there is no unwanted correlation between the \mathbb{G}_1 part and the \mathbb{G}_2 part by the Chinese Remainder Theorem. Similarly, the fact $\zeta_1 = \bar{\zeta}_1, \dots, \zeta_D = \bar{\zeta}_D, \eta_1 = \bar{\eta}_1, \dots, \eta_L = \bar{\eta}_L$ will not result in unwanted correlation between the \mathbb{G}_1 and the \mathbb{G}_2 of this key.

When the simulator \mathcal{B} uses the ν -th key to decrypt the semi-functional ciphertext to test whether the key is normal or semi-functional, it will obtain

$$\prod_{\rho(i) \in S^*} \left(e(g_2, g_2)^{c A_i \vartheta} e(g_2, g_2)^{f_i (c \zeta_x + v_{i^*} \sum_{j=1}^k u_j \eta_j - \varphi_{i^*})} \right)^{\omega_i} = 1.$$

This is because from the simulation of semi-functional ciphertext we have that

$$\varphi_{i^*} = c(\bar{\zeta}_x + t_{i^*} \sum_{j=1}^k u_j \bar{\eta}_j), v_{i^*} = ct_{i^*}$$

and from the simulation of the ν -th key, we have that

$$\zeta_1 = \bar{\zeta}_1, \dots, \zeta_D = \bar{\zeta}_D, \eta_1 = \bar{\eta}_1, \dots, \eta_L = \bar{\eta}_L.$$

Moreover, since the inner product

$$\vartheta \cdot (1, 0, \dots, 0) = c\bar{\vartheta} \cdot (1, 0, \dots, 0) = 0,$$

$\sum_{\rho(i) \in S^*} \omega_i A_i \vartheta = 0$. Thus, when \mathcal{B} uses the ν -th key to decrypt the semi-functional ciphertext, the decryption will still work and the ν -th key is nominally semi-functional. Now, we argue that the nominally semi-functional key is identically distributed as a semi-functional key in \mathcal{A} 's view. That is, if \mathcal{A} is prevented from asking the ν -th key that can decrypt the challenge ciphertext, the fact that $\vartheta_1 = 0$ (ϑ_1 is set as the first coordinate of ϑ) should be information-theoretically hidden in \mathcal{A} 's view.

Because the ν -th key cannot decrypt the challenge ciphertext, the vector $(1, 0, \dots, 0)$ is linearly independent of \mathbf{R}_{S^*} , which is a submatrix of \mathbf{A} and contains only those rows that satisfy $\rho(i) \in S^*$. From the basics of linear algebra and similarly to Proposition 11 in [11], we have the following proposition:

Proposition 1 *A vector \mathbf{v} is linearly independent of a set of vectors represented by a matrix \mathbf{M} if and only if there exists a vector \mathbf{w} such that $\mathbf{M}\mathbf{w} = \mathbf{0}$ while $\mathbf{v} \cdot \mathbf{w} = 1$.*

Since $(1, 0, \dots, 0)$ is linearly independent of \mathbf{R}_{S^*} , a vector \mathbf{w} must exist such that for each row $A_i \in \mathbf{R}_{S^*}$, it holds that $A_i \cdot \mathbf{w} = 0$, $\mathbf{w} \cdot (1, 0, \dots, 0) = 1$. Then for the fixed vector \mathbf{w} , we can write

$$A_i \cdot \vartheta = A_i \cdot (z\mathbf{w} + \mathbf{r}),$$

where \mathbf{r} is uniformly distributed and reveals no information about z . We note that $\vartheta \cdot (1, 0, \dots, 0)$ can not be determined

from \mathbf{r} alone, some information about z is also needed. If $\rho(i) \in S^*$, then $A_i \cdot \vartheta = A_i \cdot \mathbf{r}$. Thus, no information about z is revealed and $A_i \cdot \vartheta$ is hidden. If $\rho(i) \notin S^*$, then $A_i \cdot \vartheta + f_i \zeta_x = A_i \cdot (z\mathbf{w} + \mathbf{r}) + f_i \zeta_x$. This equation introduces a random element $f_i \zeta_x$, where f_i is random and appears only once because ρ is injective. Hence if not all of the f_i values are congruent to 0 modulo p_2 , no information about z is revealed. The probability that all f_i 's are 0 modulo p_2 is negligible. Therefore, the value being shared in \mathbb{G}_2 is information-theoretically hidden in \mathcal{A} 's view with probability close to 1. Hence, \mathcal{B} simulates the semi-functional keys with a probability close to 1.

Guess: If $T \in \mathbb{G}_{13}$, we are in $Game_{\nu-1}$. If $T \in \mathbb{G}$, we are in $Game_{\nu}$. If \mathcal{A} outputs $b' = b$, \mathcal{B} outputs 0. Then, with the input tuple $(g, X_1 X_2, X_3, Y_2 Y_3, T)$, the advantage of \mathcal{B} in breaking Assumption 2 is:

$$\begin{aligned} & |\Pr[\mathcal{B}(g, X_1 X_2, X_3, Y_2 Y_3, T \in \mathbb{G}_{13}) = 0] - \\ & \Pr[\mathcal{B}(g, X_1 X_2, X_3, Y_2 Y_3, T \in \mathbb{G}) = 0]| \\ & = |Game_{\nu-1} \text{Adv}_{\mathcal{A}} - Game_{\nu} \text{Adv}_{\mathcal{A}}| = \epsilon, \end{aligned}$$

where $Game_{\nu-1} \text{Adv}_{\mathcal{A}}$ is the advantage of \mathcal{A} in $Game_{\nu-1}$ and $Game_{\nu} \text{Adv}_{\mathcal{A}}$ is the advantage of \mathcal{A} in $Game_{\nu}$. \square

Lemma 4 *If \mathcal{A} can distinguish $Game_q$ from $Game_{final}$ with advantage ϵ , then we can construct an algorithm \mathcal{B} that contradicts Assumption 3 with advantage ϵ .*

Proof We construct \mathcal{B} to simulate $Game_q$ or $Game_{final}$ to interact with \mathcal{A} by using the tuple $(g, g^{\alpha} X_2, X_3, g^s Y_2, Z_2, T)$ of Assumption 3.

Setup: For all $i = 1, \dots, D$ and all $j = 1, \dots, L$, \mathcal{B} chooses random exponents $\tilde{\zeta}_i, \bar{\eta}_j \in \mathbb{Z}_N$ and computes $v_i = g^{\tilde{\zeta}_i}, h_j = g^{\bar{\eta}_j}$. Then it sets

$$PK = (\mathbf{U}, N, g, v_1, \dots, v_D, h_1, \dots, h_L, e(g, g^{\alpha} X_2))$$

and gives PK to \mathcal{A} . We note that \mathcal{B} does not know the secret α .

Key generation Phase 1, Phase 2: To simulate the semi-functional keys for \mathbb{A} , \mathcal{B} first generates an LSSS (\mathbf{A}, ρ) for \mathbb{A} . It then selects two vectors: ϕ , which has the first coordinate set to 1 and the remaining $n - 1$ coordinates randomly chosen in \mathbb{Z}_N , and ψ , which has the first coordinate set to 0 and the remaining $n - 1$ coordinates randomly chosen in \mathbb{Z}_N . We note that this implicitly sets $\alpha = \alpha\phi + \psi$.

For the i -th row A_i of \mathbf{A} , algorithm \mathcal{B} chooses random elements $r_i, \bar{f}_i \in \mathbb{Z}_N; R_{i,0}, R_{i,1}, R_{i,2}, R_{i,k+1}, \dots, R_{i,L} \in$

\mathbb{G}_3 . \mathcal{B} randomly chooses $\zeta_1, \dots, \zeta_D, \eta_1, \dots, \eta_L \in \mathbb{Z}_N$ and computes the key as follows:

$$\begin{aligned} K_{i,0} &= g^{A_i \psi} (g^{\alpha} X_2)^{A_i \phi} v_x^{r_i} Z_2^{\bar{f}_i \zeta_x} R_{i,0}, \\ K_{i,1} &= g^{r_i} Z_2^{\bar{f}_i} R_{i,1}, \\ K_{i,2} &= (h_1^{u_1} \dots h_k^{u_k})^{r_i} Z_2^{\bar{f}_i \sum_{j=1}^k u_j \eta_j} R_{i,2}, \\ K_{i,k+1} &= h_{k+1}^{r_i} Z_2^{\bar{f}_i \eta_{k+1}} R_{i,k+1}, \\ &\vdots \\ K_{i,L} &= h_L^{r_i} Z_2^{\bar{f}_i \eta_L} R_{i,L}. \end{aligned}$$

By assuming $X_2 = g_2^{c_2}, Z_2 = g_2^{d_2}$, this implicitly sets $\vartheta = c_2 \phi, f_i = d_2 \bar{f}_i$. We also note that the values being shared in \mathbb{G}_2 are properly randomized by f_i . Therefore, this key is identically distributed as the semi-functional key in \mathcal{A} 's view.

Challenge: When \mathcal{B} is given two equal-length messages M_0 and M_1 and a set S^* of attribute vectors, \mathcal{B} flips a random coin $b \in \{0, 1\}$ and chooses $t_{i^*} \in \mathbb{Z}_N$ for all $i^* = 1, \dots, |S^*|$. Then it sets the ciphertext to be:

$$C = M_b T, E = g^s Y_2,$$

$$\begin{aligned} C_{i^*,0} &= (g^s Y_2)^{\tilde{\zeta}_x} (g^s Y_2)^{t_{i^*} (\bar{\eta}_1 u_1 + \dots + \bar{\eta}_k u_k)}, \\ C_{i^*,1} &= (g^s Y_2)^{t_{i^*}}. \end{aligned}$$

Assuming $Y_2 = g_2^c$, this implicitly sets

$$\varphi_{i^*} = c(\tilde{\zeta}_x + t_{i^*} \sum_{j=1}^k u_j \bar{\eta}_j)$$

and $v_{i^*} = c t_{i^*}$, but again there is neither correlation between $(\varphi_{i^*} \bmod p_2)$ and $(\tilde{\zeta}_x, \bar{\eta}_j \bmod p_2)$, nor correlation between $(t_{i^*} \bmod p_2)$ and $(v_{i^*} \bmod p_2)$ by the Chinese Remainder Theorem.

If $T = e(g, g)^{\alpha}$, then this ciphertext is the semi-functional ciphertext of message M_b . If T is a random element in \mathbb{G}_T , this ciphertext is a semi-functional encryption of a random message.

Guess: If $T = e(g, g)^{\alpha}$, we are in $Game_q$. If T is a random element in \mathbb{G}_T , we are in $Game_{final}$. \mathcal{B} outputs 0 when \mathcal{A} outputs $b' = b$. Given the tuple $(g, g^{\alpha} X_2, X_3, g^s Y_2, Z_2, T)$, the advantage of \mathcal{B} in breaking Assumption 3 is:

$$\begin{aligned} & |\Pr[\mathcal{B}(g, g^{\alpha} X_2, X_3, g^s Y_2, Z_2, T = e(g, g)^{\alpha}) = 0] - \\ & \Pr[\mathcal{B}(g, g^{\alpha} X_2, X_3, g^s Y_2, Z_2, T \xleftarrow{R} \mathbb{G}_T) = 0]| \\ & = |Game_q \text{Adv}_{\mathcal{A}} - Game_{final} \text{Adv}_{\mathcal{A}}| = \epsilon, \end{aligned}$$

where $Game_q \text{Adv}_{\mathcal{A}}$ is the advantage of \mathcal{A} in $Game_q$ and $Game_{final} \text{Adv}_{\mathcal{A}}$ is the advantage of \mathcal{A} in $Game_{final}$. \square

From all the lemmas proven above, the proof of Theorem 1 follows:

Proof In $Game_{final}$, the ciphertext completely hides the bit b , so the advantage of \mathcal{A} in this game is negligible. Through Lemmas 1, 2, 3 and 4, we have shown that the real security game $Game_{real}$ is indistinguishable from $Game_{final}$. Therefore, the advantage of \mathcal{A} in $Game_{real}$ is negligible. Hence, there is no polynomial-time adversary with a non-negligible advantage in breaking our APR-ABE system. This completes the proof of Theorem 1. \square

6 Conclusion

We revisited KP-ABE and proposed a dynamic ABE referred to as APR-ABE. APR-ABE distinguishes itself from other KP-ABE schemes by providing a delegation mechanism that allows a user to redefine the access policy and delegate a secret key without making the redefined access policy more restrictive. This feature renders APR-ABE especially suitable to e-healthcare record systems where *a priori* specification of access policies for secret keys is too rigid or simply not practical. We constructed an APR-ABE scheme with short ciphertexts and proved its full security in the standard model under several non-interactive assumptions.

Acknowledgements and disclaimer

We thank the anonymous reviewers for their valuable suggestions. The following funding sources are gratefully acknowledged: Natural Science Foundation of China (projects 61370190, 61173154, 61272501, 61402029, 61202465 and 61472429), China National Key Basic Research Program (973 program, project 2012CB315905), Beijing Natural Science Foundation (project 4132056), Fundamental Research Funds for the Central Universities of China, Research Funds of Renmin University (No. 14XNLF02), European Commission (projects FP7 “DwB”, FP7 “Inter-Trust” and H2020 “CLARUS”), Spanish Govt. (project TIN2011-27076-C03-01), Govt. of Catalonia (ICREA Acadèmia Prize to the fourth author). The fourth author leads the UNESCO Chair in Data Privacy, but the views in this paper do not commit UNESCO.

References

- Attrapadung, N., Libert, B., De Panafieu, E. Expressive Key-Policy Attribute-Based Encryption with Constant-Size Ciphertexts. PKC 2011. LNCS 6571, pp. 90-108. Springer (2011)
- Beimel, A. Secure Schemes for Secret Sharing and Key Distribution. PhD thesis, Israel Institute of Technology, Technion, Haifa, Israel (1996)
- Bethencourt, J., Sahai, A., Waters, B. Ciphertext-Policy Attribute-Based Encryption. IEEE Symp. Sec. & Priv. 2007, pp. 321-334. IEEE Press (2007)
- Boneh, D., Boyen, X., Goh, E. Hierarchical Identity Based Encryption with Constant Size Ciphertext. EUROCRYPT 2005. LNCS 3493, pp. 440-456. Springer (2005)
- Boneh, D., Goh E., Nissim, K.: Evaluating 2-DNF Formulas on Ciphertexts. TCC 2005. LNCS 3378, pp. 325-341. Springer (2005)
- Boneh, D., Nikolaenko, V., Segev G. Attribute-Based Encryption for Arithmetic Circuits. Cryptology ePrint Archive, Report 2013/669, 2013, <https://eprint.iacr.org/2013/669.pdf>
- Goyal, V., Jain, A., Pandey, O., Sahai, A. Bounded Ciphertext Policy Attribute Based Encryption. ICALP 2008. LNCS 5126, pp. 579-591. Springer (2008)
- Goyal, V., Pandey, O., Sahai, A., Waters, B. Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data. ACM CCS 2006, pp. 89-98. ACM Press (2006)
- Hohenberger, S., Waters, B. Attribute-Based Encryption with Fast Decryption. PKC 2013. LNCS 7778, pp. 162-179. Springer (2013)
- Hur, J. Fine-grained Data Access Control for Distributed Sensor Networks. Wireless Networks, 17(5), 1235-1249 (2011).
- Lewko, A., Okamoto, T., Sahai, A., Takashima, K., Waters, B. Fully Secure Functional Encryption: Attribute-Based Encryption and (Hierarchical) Inner Product Encryption. EUROCRYPT 2010. LNCS 6110, pp. 62-91. Springer (2010)
- Lewko, A., Waters, B. New Techniques for Dual System Encryption and Fully Secure HIBE with Short Ciphertexts. TCC 2010. LNCS 5978, pp. 455-479. Springer (2010)
- Lewko, A., Waters, B. Unbounded HIBE and Attribute-Based Encryption. EUROCRYPT 2011. LNCS 6632, pp. 547-567. Springer (2011)
- Li, J., Wang, Q., Wang, C., Ren, K. Enhancing Attribute-Based Encryption with Attribute Hierarchy. ChinaCom 2009, pp. 1-5. IEEE Press (2009)
- Li, M., Yu, S., Zheng, Y., Ren, K., Lou, W. Scalable and Secure Sharing of Personal Health Records in Cloud Computing Using Attribute-based Encryption. Parallel and Distributed Systems, IEEE Transactions on, 24(1), 131-143 (2013).
- Liang, X., Barua, M., Lu, R., Lin, X., Shen, X. S. HealthShare: Achieving Secure and Privacy-preserving Health Information Sharing Through Health Social Networks. Computer Communications, 35(15), 1910-1920 (2012).
- Ostrovsky, R., Sahai, A., and Waters, B. Attribute-Based Encryption with Non-monotonic Access Structures. ACM CCS 2007, pp. 195-203. ACM Press (2007)
- Rouselakis, Y., Waters, B. Practical Constructions and New Proof Methods for Large Universe Attribute-based Encryption. ACM CCS 2013, pp. 463-474. ACM Press (2013)
- Sahai, A., Waters, B. Fuzzy Identity-Based Encryption. EUROCRYPT 2005. LNCS 3494, pp. 457-473. Springer (2005)
- Waters, B. Dual System Encryption: Realizing Fully Secure IBE and HIBE under Simple Assumptions. CRYPTO 2009. LNCS 5677, pp. 619-636. Springer (2009)
- Waters, B. Ciphertext-Policy Attribute-Based Encryption: An Expressive, Efficient, and Provably Secure Realization. PKC 2011. LNCS 6571, pp. 53-70. Springer (2011)
- Wan, Z., Liu, J., Deng, R.H. HASBE: A Hierarchical Attribute-Based Solution for Flexible and Scalable Access Control in Cloud Computing. Information Forensics and Security, IEEE Transactions on, 7(2), pp. 743-754, 2012.
- Wang, G., Liu, Q., Wu, J. Hierarchical Attribute-Based Encryption for Fine-Grained Access Control in Cloud Storage Services. ACM CCS 2010, pp. 735-737. ACM Press (2010)
- Yu, S., Ren, K., Lou, W. FDAC: Toward Fine-grained Distributed Data Access Control in Wireless Sensor Networks. Parallel and Distributed Systems, IEEE Transactions on, 22(4), 673-686 (2011).