CABE: A New Comparable Attribute-Based Encryption Construction with 0-Encoding and 1-Encoding

Kaiping Xue, *Senior Member, IEEE*, Jianan Hong, Yingjie Xue, David S. L. Wei, *Senior Member, IEEE*, Nenghai Yu, and Peilin Hong

Abstract—Attribute-based encryption (ABE) has opened up a popular research topic in cryptography over the past few years. It can be used in various circumstances, as it provides a flexible way to conduct fine-grained data access control. Despite its great advantages in data access control, current ABE based access control system cannot satisfy the requirement well when the system judges the access behavior according to attribute comparison, such as "greater than x" or "less than x", which are called *comparable attributes* in this paper. In this paper, based on a set of well-designed sub-attributes representing each comparable attribute, we construct a comparable attribute-based encryption scheme (CABE for short) to address the aforementioned problem. The novelty lies in that we provide a more efficient construction based on the generation and management of the sub-attributes with the notion of 0-encoding and 1-encoding. Extensive analysis shows that: Compared with the existing schemes, our scheme drastically decreases the storage, communication and computation overheads, and thus is more efficient in dealing with the applications with comparable attributes.

Index Terms—Comparable attribute, access control, attribute-based encryption, 0-encoding and 1-encoding

1 Introduction

Attribute-based encryption (ABE), first proposed by Sahai and Waters in [1], has become a popular core cryptographic technique for access control. It has great advantages over identity-based encryption (IBE) [2] in many kinds of complex communication environments: Instead of encrypting data under the intended users' public keys in IBE, it implements flexible management of association between ciphertexts and users' security keys efficiently, thus it is suitable for many scenarios, such as cloud computing [3], [4], [5], [6], [7], e-Health [8], [9], [10], wireless sensor networks [11] and social networks [12], [13], [14]. In ABE, both the users' security keys and the ciphertexts are labelled with sets of attributes. If there are adequate attributes that are matched between a key and a ciphertext, the key can be used to decrypt the data correctly.

There are two different implementation methods for ABE: key-policy attribute-based encryption (KP-ABE) [15], and ciphertext-policy attribute-based encryption (CP-ABE) [16]. The main difference between these two categories is the method how to embed the access policy: In KP-ABE, access policy is embedded within a user's security key, and the ciphertexts are associated with several attributes; On the contrary, in CP-ABE, the access policy is embedded within the corresponding ciphertext, and the user's keys are linked to the attributes. Both of these schemes utilize a monotonous rule: The decryption succeeds, if and only if one entity's attributes are adequate to satisfy a certain element's access policy.

However, in current ABE systems, attributes comparison between security keys and ciphertexts is not flexible to face practical applications: The comparison operation between a user's and a file's attributes only includes "=". In fact, many traditional access control systems, such as an operation system and a database, need to judge the users' access behaviors based on diverse relationships between a user's and a file's attributes, such as "$=, <, >$". Meanwhile, in many novel access control systems, such as wireless sensor networks, e-Health, cloud storage platforms, etc, most access policies probably involve the attributes in such form of comparison (we call them *comparable attributes* in this paper). For instance, in wireless sensor network environment like military domains, a sensor node may collect information and encrypt it under attributes such as geographic location, time, etc.. These attributes are numerical information, and will probably be used in comparison. For instance, one ciphertext contains attributes as $\{Distance = 750\ miles, Date = July\ 7th, Owner = (experts, officers)\}$. However, a user may be assigned a key embedded with access policy as: "$(Distance < 1000\ miles)$ AND $(Date > May\ 1st)$". Such numerical attributes are also usually involved in other

- *J. Hong, Y. Xue, N. Yu and P. Hong are with the Department of Electronic Engineering and Information Science, University of Science and Technology of China, Hefei 230027, China. E-mail: {hongjn, xyj1108}@mail.ustc.edu.cn, {ynh, plhong}@ustc.edu.cn.*
- *D.S.L. Wei is with the Computer and Information Science Department, Fordham University, New York, NY 10458. E-mail: wei@dsm.fordham.edu.*
- *K. Xue is with the Department of Electronic Engineering and Information Science, University of Science and Technology of China, Hefei 230027, China and also with the State Key Laboratory of Information Security (Institute of Information Engineering), Chinese Academy of Sciences, Beijing 100093, China. E-mail: kpxue@ustc.edu.cn.*

Manuscript received 19 Sept. 2016; revised 6 Apr. 2017; accepted 7 Apr. 2017. Date of publication 11 Apr. 2017; date of current version 15 Aug. 2017. Recommended for acceptance by W. Liu. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TC.2017.2693265

communication environments such as e-Health, social networks, cloud storages, etc..

In current ABE, if the boolean function doesn't work with value comparison, the match will fail because *"Distance* = $750\ miles$" doesn't equal to *"Distance* $<\ 1000\ miles$". A trivial way to deal with comparable attributes in current ABE is to involve all allowed attributes to express the range. For example, *"Distance* $<\ 1000\ miles$" can be represented as a formula like (*"Distance* = $999\ miles$" $\vee$ *"Distance* = $998\ miles$" $\vee \ldots \vee$ *"Distance* = $0\ miles$"), but the overhead increases linearly with the growth of attribute's value space, which will become a performance bottleneck of the system.

In [16], the authors first did preliminary attempts to address the above problem. This scheme divided such numerical attribute into pieces in units of bits as several sub-attributes to solve this problem. However, the mechanism to design a numeric-comparison policy is too complex, and the most essential problem is that the additional overhead is still relatively high.

In this paper, we propose a new scheme to enable ABE to implement comparable attributes, called *Comparable Attribute-based Encryption* (CABE). In CABE, we use a unique way to generate and manage sub-attributes for comparable attributes, so as to make a solution for addressing the above issue in an efficient way, in terms of both storage overhead and computation overhead. Our solution in dealing with sub-attributes is based on a special notion called 0-encoding and 1-encoding [17].

The main contributions of this paper can be summarized as follows:

1) We propose an efficient method based on a special concept 0-encoding and 1-encoding, so that the attributes can be used in arbitrary comparison, which is suitable for ABE system;
2) A lightweight and efficient CABE construction is proposed. This construction halves the expanded storage overhead in average compared with related schemes, and significantly decreases the computation overhead in encryption and decryption from $\Theta(\log N)$ to $\Theta(1)$ (N denotes the value space of the attribute dimension).

The rest of this paper is organized as follows: Section 2 presents related work of *attribute-based encryption* and some numeric comparison mechanism in encryption field. Section 3 depicts the core concept and some definitions in CABE. Section 4 describes CABE construction in detail. In Section 5, we analyze our proposed scheme in terms of its security and performance. Finally, in Section 6, we conclude this paper.

2 Related Work

Since attribute-based encryption [1], [18] provides cryptographic techniques a way to realize fine-grained data access control, there have been various ABE based schemes proposed. These schemes can be categorized into two types: key-policy attribute-based encryption (KP-ABE), such as [15], [19], and ciphertext-policy attribute-based encryption (CPABE), such as [16], [20], [21], [22], [23], [24]. Furthermore, from the perspective of access policy construction, these schemes can be classified into two kinds: one is that supports arbitrary threshold gates, such as [15], [16], [20], the other is that only supports AND gate, such as [23]. Schemes only supporting AND gate show their advantages in some scenarios, such as constant-size storage [24], [25], [26], [27], and policy hiding [28]. However, AND gate alone cannot designate intended users well in many cases. Thus the schemes supporting threshold gates have been attracting much more attentions.

ABE provides fine-grained access control in many areas, such as cloud computing, e-health, wireless sensor networks. These scenarios potentially comprise energy or storage constraint terminals. Hence, many schemes have been proposed to improve the performance of ABE from different aspects. Encryption and decryption outsourcing [4], [29] migrates most of the computational burden from the users to cloud servers. The support of online/offline encryption mechanism [10], [30] saves most of the computation overhead in encryption, because it has been pre-operated when the energy was plenty. Lightweight policy update mechanism [31] allows data owners to dynamically update access structures of his/her stored data with as small overhead as possible. Such lightweight policy update can also be realized when time-domain is introduced into ABE algorithm, such as [32], [33].

Although the performance improvement of ABE has attracted a considerable mount of researchers' attention, the existing ABE schemes' operation in matching is still inefficient. The research of [11] points out that on a low-end device, an encryption and decryption algorithm with fewer participated attributes and less complex policies is preferred. The cost increases linearly with the attribute number. However, the value comparison in access policy should expand the attribute number of the structure. Thus, an inefficient mechanism may affect the feasibility of the scheme. Bethencourt et al. [16] have proposed a relatively efficient solution for numeric inequalities in ABE. Their scheme is as follows: 1) A certain numeric attribute is extended to a set of sub-attributes; Each sub-attribute represents each bit. 2) A comparison operator ("$>$" or "$<$") is represented as a sub-tree, whose leaf nodes represent the sub-attributes of diverse bits. Such sub-tree can be attached to an access structure, replacing a leaf node. Numerous studies in dealing with numeric inequalities have used Bethencourt et al.'s mechanism, such as [21], [34], [35]. However, there still exists plenty of room to improve the performance efficiency in dealing with numerical comparison. Attrapadung et al. [36] have proposed a scheme that supports range attributes, which can realize a great improvement for numeric comparison. Compared with our scheme presented in this paper, Attrapadung et al.'s scheme needs to initialize a binary tree to organize its sub-attributes for numeric attributes, which is relatively complex, and is thus not suitable for storage/computation constrained devices.

In reality, in the area of predicate encryption [37], many schemes have provided numeric comparison in encryption and decryption [38], [39], [40]. In these schemes, the numeric, which is in the designate range may help to decrypt the content. However, the predicate encryption, which uses the result of vector inner product to judge the access privilege, is much less expressive than the access structures of ABE [41].

Furthermore, in Table 1, we give a comparison about some related schemes and our proposed CABE in terms of critical aspects on a high level.

TABLE 1
Comparison of Related Schemes

Scheme	Basic Algorithm	Support for Comparison	Highlight	Auxiliary Structure
Goyal et al.'s [15]	KP-ABE	No	The First to Realize CP-ABE/KP-ABE	N/A
Waters's [20]	CP-ABE	No	LSSS for Arbitrary Thresholds	N/A
Bethencourt et al.'s [16]	CP-ABE	Not Efficient	The First to Provide Attribute Comparison	Not Required
Boneh el al.'s [38] Shi et al.'s [39] Lin et al.'s [40]	Predicate Encryption	Yes	Arbitrary Range Border	Binary Tree
Attrapadung et al.'s [36]	CP-ABE and KP-ABE	Yes	Arbitrary Range Border	Binary Tree
Ours	CP-ABE and KP-ABE	Yes	Efficient Attribute Comparison	Not Required

3 Preliminaries and Definitions

The inspiration for this work comes from Lin et al.'s scheme in [17]. The concept of 0-encoding and 1-encoding was proposed initially to solve the Millionaires' Problem (MP) [42]. The common issue of MP and the comparable attribute construction is that: Both of them have numerical comparison process of two entities. However, their security assumptions are totally different. In MP, the involved parties are assumed to be semi-trusted, which means each user would follow the protocol in general, while the wealth of the other party is private, and should be protected in the procedure. On the contrary, in access control system, users may go against the protocol for some unauthorized data, while the privacy is less concerned compared with data confidentiality. Because of the different security assumptions, the mechanisms of using this concept are diverse.

In this section, we will first present the definition of Lin et al.'s 0-encoding and 1-encoding, and then depict the model of our system, using 0-encoding and 1-encoding in a different way with the different environment from MP. In the end, we will propose the security definition.

3.1 Definition of 0-Encoding and 1-Encoding

Our system uses the concept of two special encodings, 0-*encoding* and 1-*encoding*.

Let $s = s_n s_{n-1} \dots s_1 \in \{0,1\}^n$ be an n-length binary string of a value for a certain attribute dimension. The *0-encoding* of s is defined as a set S_s^0 such that

$$S_s^0 = \{s_n s_{n-1} \dots s_{i+1} 1 | s_i = 0, 1 \le i \le n\}.$$

The *1-encoding* of s is the set S_s^1 such that

$$S_s^1 = \{s_n s_{n-1} \dots s_i | s_i = 1, 1 \le i \le n\}.$$

Intuitively, 1-encoding of s is the set of all its odd prefix substrings, and the 0-encoding is the set of all of its modified even prefix substrings, where the least significant bit is flipped from "0" to "1". The size of set S_s^0 equals to the number of characters "0" in string s, and meanwhile the size of S_s^1 equals to the number of "1". Compared with the value space of n-length binary string: $N = 2^n$, both S_s^1 and S_s^0 have at most $\log_2 N$ elements.

To compare two integers x and y in form of n-length binary string, we encode x into 1-encoding S_x^1, and y into 0-encoding S_y^0. We make the judgment that $x > y$ if and only if there's an element in both S_x^1 and S_y^0. A formula to express this theorem is as

$$x > y \Longleftrightarrow S_x^1 \bigcap S_y^0 \neq \varnothing. \tag{1}$$

Lin et al. [17] have proved Eq. (1) clearly. Here we take two 4-bit numbers $x = 11$ (1011_2) and $y = 6$ (0110_2) as an example to explain it. They are encoded as Table 2.

From the result presented in Table 2, one common element "1" exists in both S_x^1 and S_y^0, so we can make a conclusion that $x > y$. On the contrary, there's no common element in S_y^1 and S_x^0, which represents almost the same result that $y \le x$.

3.2 Attribute Management

The object we deal with is such attribute that is not an exact value, but stands as a range of continuous values, and may be matched in comparison in ABE system, such as "*Score* > 75", "*Age* < 25". For one attribute field $\mathcal{F}$, whose value space is N and minimum value is val_{min}, we can reduce the storage overhead to $1/2\lceil \log_2 N \rceil$ in average to make this kind of attribute suit CP-ABE construction with utilization of *0-encoding* and *1-encoding*. Our procedure is implemented as follows:

For a value $x \in \mathcal{F}$, if its minimum value is not zero, compute $x_m = x - val_{min}$; otherwise, such operation can be skipped. For clarity of description, let $x_m = x$ in this case. If the length of the binary string x_m is less than $\lceil \log_2 N \rceil$, use 0 to fill at high bits. Then what the program deals with is a new $\lceil \log_2 N \rceil$-long binary string x_m. We encode x_m into 0-encoding $S_{x_m}^0$ and 1-encoding $S_{x_m}^1$ with the rule described in Section 3.1. For a certain access structure, if the access policy needs the corresponding attribute $\mathcal{F}$ to satisfy that $\mathcal{F} > x$, an attribute set $Set_{c0}(\mathcal{F}, x)$ can be designed as

$$Set_{c0}(\mathcal{F}, x) = \left\{ (\mathcal{F} || \text{“} > x \text{”} || c) | c \in S_{x_m}^0 \right\}. \tag{2}$$

TABLE 2
0-Encoding and 1-Encoding of 11 and 6

	1-encoding	0-encoding
x=1011_2	1 101 1011	11
y=0110_2	01 011	1 0111

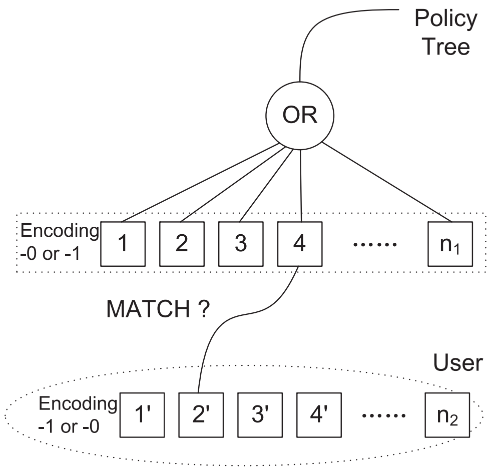


Fig. 1. Model of comparison and matching for comparable attribute.

Where, each element of this set includes three parts, and the symbol "||" means a concatenation of two different parts. Each element of the set represents a sub-attribute of the comparable attribute $\mathcal{F} > x$. If the policy needs "$\mathcal{F} < x$", we use a set $Set_{c1}(\mathcal{F}, x)$ to represent this attribute

$$Set_{c1}(\mathcal{F}, x) = \left\{(\mathcal{F}||\text{``}<x\text{''}||c)|c \in S^1_{x_m}\right\}. \tag{3}$$

For a numerical attribute $\mathcal{F} = x$, two sets $Set_{u0}(\mathcal{F}, x)$ and $Set_{u1}(\mathcal{F}, x)$ are generated on behalf of this attribute like

$$\begin{aligned} Set_{u0}(\mathcal{F}, x) &= \left\{(\mathcal{F}||\text{``}<x\text{''}||c)|c \in S^0_{x_m}\right\}, \\ Set_{u1}(\mathcal{F}, x) &= \left\{(\mathcal{F}||\text{``}>x\text{''}||c)|c \in S^1_{x_m}\right\}. \end{aligned} \tag{4}$$

To compare and match the attribute in field $\mathcal{F}$, if an access structure raises $Set_{c1}(\mathcal{F}, x)$ for comparison, $Set_{u0}(\mathcal{F}, x)$ should be utilized to implement the procedure of matching on behalf of a certain attribute set to be compared. Conversely, if $Set_{c0}(\mathcal{F}, x)$ is raised, $Set_{u1}(\mathcal{F}, x)$ should be utilized. Comparison will succeed if two matching sets have a common element. A dishonest user who uses attribute from other field different from $\mathcal{F}$, or confusedly misuses $Set_{u0}(\mathcal{F}, x)$ and $Set_{u1}(\mathcal{F}, x)$ will fail in the procedure of comparison because of the existence of the first two parts of the element.

The procedure of comparison and matching is shown in Fig. 1. For an access structure, the elements of encoded attribute set are managed as a policy tree with an OR gate. While for a user with a specific attribute set, there exists no such organization for the elements. As we can see, if and only if there exists a common element between the two sets, the OR gate will get a *true* result. This architecture can suit the basic *attribute-based encryption* well, and the detailed explanation of this model is given in Section 4.

3.3 Security Definition

3.3.1 Cryptographic Assumption

Let $\mathbb{G}_0$, $\mathbb{G}_1$ be two multiplicative cyclic groups of a prime order p, and g be a generator of $\mathbb{G}_0$. Let $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$ be a bilinear pairing, satisfying the following properties:

1) $e(g^x, g^y) = e(g, g)^{xy}, \forall x, y \in \mathbb{Z}_p$;
2) $e(g, g) \neq 1$;
3) To compute the pairing e is efficient.

Definition 1. *Decisional Bilinear Diffie-Hellman (DBDH) assumption. Given a tuple (g^a, g^b, g^c, Z), where $a, b, c \in \mathbb{Z}_p$ and $Z \in \mathbb{G}_1$, for any polynomial algorithm adversary $\mathcal{A}$, it is negligible to determine whether $Z = e(g, g)^{abc}$, or Z is a random element of $\mathbb{G}_1$. The advantage is defined as follows:*

$$\begin{aligned} \frac{1}{2}|&Pr[\mathcal{A}(g, g^a, g^b, g^c, e(g, g)^{abc}) = 1] \\ &-Pr[\mathcal{A}(g, g^a, g^b, g^c, e(g, g)^z) = 1]|. \end{aligned} \tag{5}$$

3.3.2 Security Model

The security model is formalized by the following security game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

- *Setup.* $\mathcal{C}$ runs *Setup*(1^λ) to generate system parameters.
- *Phase 1.* $\mathcal{A}$ makes security key request according to an arbitrary attribute set:

$$\{A_1, A_2, \ldots, A_{l_1}, \{F_1\}, \{F_2\}, \ldots, \{F_{l_2}\}\},$$

  where, the A_i is the normal attribute, and $\{F_i\}$ is a set of elements for a comparable attribute, including $Set_{u0}(\mathcal{F}, x)$ and $Set_{u1}(\mathcal{F}, x)$.
- *Challenge.* $\mathcal{A}$ submits two messages M_0 and M_1 with equal length to $\mathcal{C}$, along with an access policy $\mathcal{T}$. The access policy $\mathcal{T}$ cannot be satisfied by the attribute set in *Phase 1*. $\mathcal{C}$ randomly chooses $M_\nu, \nu \in (0, 1)$ and encrypts it under the access policy $\mathcal{T}$. The generated ciphertext is sent to $\mathcal{A}$.
- *Phase 2.* $\mathcal{A}$ repeat Phase 2 to ask for more attribute keys, as long as the entire attribute set does not satisfy $\mathcal{T}$.
- *Guess.* $\mathcal{A}$ outputs a guess ν' of ν. The advantage of $\mathcal{A}$ is as:

$$\left|Pr[\nu' = \nu] - \frac{1}{2}\right|. \tag{6}$$

Definition 2. *Our proposed CABE is secure if all polynomial-time adversaries have at most a negligible advantage in the above game.*

4 Construction of Our Proposed CABE

In this section, we adopt CP-ABE [16] as the basic construction to depict the architecture of CABE, but the mechanism also works well in KP-ABE. We will propose several specific models of CABE, and describe four fundamental algorithms of *Setup*, *Encrypt*, *KeyGen* and *Decrypt* in details.

4.1 Some Definitions in CABE

Before presenting the CABE, there are several definitions given first. *Extended Attribute Set* represents the set of attributes we deal with in CABE, which has some differences from original CP-ABE. *Access structure $\mathcal{T}$* is presented to show the basic idea on how it works to achieve fine-grained access control using CABE.

4.1.1 Extended Attribute Set

Let $\mathcal{S}_0$ be the set of attributes involved in the access structure of data. Original $\mathcal{S}_0$ has both boolean elements and

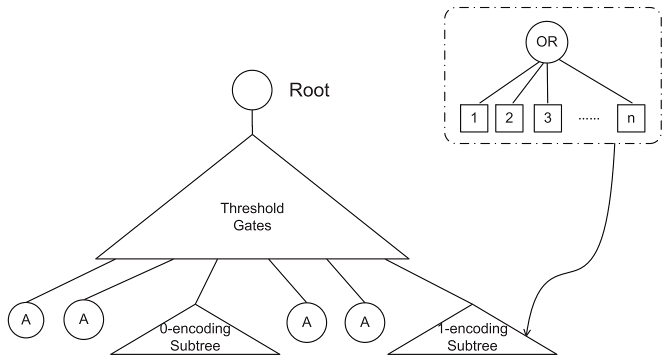


Fig. 2. Model of access structure with comparable attributes.

comparable ones. To deal with the latter, for example, an involved attribute is "$Score > 75$", we can replace it by elements in $Set_{c0}(Score, 75)$. We will get a new attribute set after all comparable attributes are replaced with elements of their 0-encodings (if the policy needs "$>$") or 1-encodings (if the policy needs "$<$"). We use a new symbol $\mathcal{S}_1$ to represent modified attribute set of the structure.

Accordingly, $\mathcal{S}_0$ represents the original attribute set without encoding for the comparable attributes. These comparable attributes can be divided into 3 categories: 1) For the attribute that only needs inequality "$<$", we replace it with elements of $Set_{u0}(\mathcal{F}, x)$; 2) For that only needs "$>$", we replace it with elements of $Set_{u1}(\mathcal{F}, x)$; 3) For the one that needs both inequalities "$<$" and "$>$", we replace it with elements of both $Set_{u0}(\mathcal{F}, x)$ and $Set_{u1}(\mathcal{F}, x)$. The new attribute set is denoted as $\mathcal{S}_2$.

4.1.2 Access Structure $\mathcal{T}$

We use the popular model of access strategy usually utilized in ABE, whose structure is an access policy tree, denoted as $\mathcal{T}$. Fig. 2 shows the model of our access structure, which is a little different from the access policy tree in typical CP-ABE. Each circle represents a single node in the tree, especially, each circle with "*A*" represents an attribute, and the circle with "*OR*" represents an *OR* gate. Each triangle represents a subtree composed of some nodes: A "*Threshold Gates*" is composed of a number of non-leaf nodes, while "*0-encoding subtree*" and "*1-encoding subtree*" are respectively composed of an only-one-layer sub-tree with one *OR* gate and several leaf nodes, where each leaf node represents an element of $Set_{c0}(\mathcal{F}, x)$(if the operation is "$\mathcal{F} > x$") or $Set_{c1}(\mathcal{F}, x)$ (if the operation is "$\mathcal{F} < x$").

To be noted, each non-leaf node of $\mathcal{T}$ actually represents a threshold gate according to the number of its child nodes and the threshold value of sharing strategy. For a non-leaf node x, if its sharing is (t, n), the number of its child nodes is n, while t represents its threshold gate. If $t = 1$, the threshold is an *OR* gate, while if $t = n$, it is an *AND* gate.

Compared with the traditional access policy tree without consideration of attribute comparison, our designed tree replaces each comparable attribute node with a one-layer subtree depicted in Fig. 2.

4.2 Construction with Symmetric Bilinear Pairing

In our construction, we will present four basic phases: *Setup*, *Encrypt*, *Key Generation (KeyGen) and Decrypt*.

4.2.1 Setup$(1^{\lambda}) \rightarrow (MK, PK)$

In this process, with a security requirement λ, an authority randomly chooses two multiplicative cyclic groups $\mathbb{G}_0$ and $\mathbb{G}_1$ with the same prime order p, satisfying the bilinear map as $e : \mathbb{G}_0 \times \mathbb{G}_0 \rightarrow \mathbb{G}_1$. The process selects a generator g from $\mathbb{G}_0$, and selects two secrets $\alpha, \beta \in \mathbb{Z}_p^*$. The secret master key MK of the authority is (β, g^{α}), and the public parameter PK is published as

$$PK = \{\mathbb{G}_0,\ g,\ h = g^{\beta},\ e(g, g)^{\alpha}\}.$$

This phase also generates a hash algorithm $H : (0, 1)^* \rightarrow \mathbb{G}_0$, mapping any binary string to a random element of $\mathbb{G}_0$.

4.2.2 Encrypt$(M, PK, \mathcal{T}) \rightarrow CT$

The subject of this phase is the data owner who wants to share a document M under the access policy $\mathcal{T}$. To achieve this goal, we can divide this process into two steps, the object we deal with in the first step is the plaintext M, and the object of the other is the policy structure $\mathcal{T}$.

In the first step, K is a randomly selected key of a symmetric cryptography, and s is a random secret from $\mathbb{Z}_p^*$. M is encrypted with K as: $\tilde{C} = E_K(M)$, $C = h^s$, $\hat{C} = K \cdot e(g, g)^{\alpha s}$.

In the second step, all nodes of $\mathcal{T}$ will be assigned a secret number from root $\mathcal{R}$ to leaf nodes with the following rules:

The root $\mathcal{R}$ is assigned with the secret s corresponding to C produced in the former step. For a non-leaf node x (including $\mathcal{R}$) that has been assigned a secret s_x and with the threshold value k_x, the algorithm randomly generates a polynomial q_x, which holds three characters listed below:

1) The degree of polynomial q_x must satisfy: $d_x = k_x - 1$.
2) The value of this polynomial holds that: $q_x(0) = s_x$. This property makes the polynomial relative to the secret of corresponding node x.
3) Each value $q_x(i)$ with different index i is assigned to each of x's child nodes.

For instance, faced with a threshold gate (3, 5) and its secret s_x, the polynomial can be generated as the format $q_x = a_2x^2 + a_1x + s_x$, where $a_2 \in_R \mathbb{Z}_p^*$, $a_1 \in_R \mathbb{Z}_p$. Such format satisfies the above mentioned restrictions: 1) its degree is $d = k - 1 = 2$; 2) $q_x(0) = s_x$. Then, each of its child nodes x_i (assume $i = 1, 2, \ldots, 5$) is assigned a value $q_x(i)$ according to its unique index i.

For a leaf node x that has been assigned a secret s_x, and representing the attribute y, calculate C_y, C'_y as: $C_y = g^{s_x}$, $C'_y = H(y)^{s_x}$.

The ciphertext is

$$CT = \left\{\mathcal{T},\ \tilde{C},\ \hat{C},\ C;\ \forall y \in \mathcal{S}_1 :\ C_y,\ C'_y\right\}.$$

Here, $\mathcal{S}_1$ is the extended attribute set, defined in Section 4.1.1. In Section 4.1.1, we also define $\mathcal{S}_2$, which will be used in the next paragraph.

4.2.3 KeyGen$(MK, \mathcal{S}_2) \rightarrow SK$

In this phase, the authority will determine the attribute set $\mathcal{S}_0$ for each user, and then extends the set to $\mathcal{S}_2$, modifying the comparable attributes. Then, the authority will select two kinds of security parameters u and r, that should be masked to the user. The first one is related to the identity of user and it stands the same among all attributes in $\mathcal{S}_2$, the second is related to the attribute, and the authority can choose different "r" for different attributes. For every element $y \in \mathcal{S}_2$ (Boolean attribute or sub-attribute of a comparable attribute), we use r_y to represent the different security parameter r. Then the security key SK is generated as

$$D = g^{\frac{\alpha+u}{\beta}};\ \forall y \in \mathcal{S}_2:\ D_y = g^u \cdot H(y)^{r_y},\ D'_y = g^{r_y}.$$

4.2.4 *Decrypt*$(CT,\ SK) \rightarrow M$

We divide this operation into two steps like Encrypt phase. The first step aims at getting the corresponding secret s buried in the root of $\mathcal{T}$, and in the second step, the reconstructed s is used to decrypt the content of the data.

In the first mission, the process carries on dealing with the policy tree $\mathcal{T}$, and the algorithm is as follows:

- For a leaf node that is matched with one attribute from $\mathcal{S}_2$, let the corresponding attribute be y, and the secret value be s_x. The algorithm is as follows:

$$\begin{aligned} F_x &= \frac{e(C_y, D_y)}{e(D'_y, C'_y)} \\ &= \frac{e(g^u \cdot H(y)^{r_y}, g^{s_x})}{e(g^{r_y}, H(y)^{s_x})} \\ &= e(g,g)^{u \cdot s_x}. \end{aligned} \tag{7}$$

- For a non-leaf node x, if no less than k_x of its child nodes have passed decryption algorithm, we denote the set of the decrypted child node as $\mathcal{S}_x$ then the algorithm proceeds as follows:

$$\begin{aligned} F_x &= \prod_{z \in \mathcal{S}_x} F_z^{\Delta_{i,\mathcal{S}_{x,z}}(0)} \\ &= \prod_{z \in \mathcal{S}_x} e(g,g)^{u \cdot s_z \Delta_{i,\mathcal{S}_{x,z}}(0)} \\ &= e(g,g)^{u \cdot s_x}. \end{aligned} \tag{8}$$

  In Eq. (8), $\Delta_{i,\mathcal{S}}(x) = \prod_{j \in \mathcal{S}, j \neq i} \frac{j-x}{j-i}$, which is defined in [16]; $\mathcal{S}_{x,z}$ represents a set $\mathcal{S}_x$ without the element z. The equation will return true, because these nodes are in the same polynomial and s_x is the secret value in this polynomial.

We can re-use the instance of threshold (3, 5) with polynomial $q_x = a_2 x^2 + a_1 x + s_x$ for description clarification. Consider three child nodes are satisfied with shares $q_i = a_2 i^2 + a_1 i + s_x$, $i = 1, 2, 3$. In the previous procedures, the user has already gotten $e(g,g)^{u \cdot (a_2 i^2 + a_1 i + s_x)}$, $i = 1, 2, 3$. Following Eq. 8, we can get

$$\begin{aligned} &\left(e(g,g)^{u \cdot (a_2 \cdot 1^2 + a_1 \cdot 1 + s_x)}\right)^{\frac{2}{1}\frac{3}{2}} \cdot \left(e(g,g)^{u \cdot (a_2 \cdot 2^2 + a_1 \cdot 2 + s_x)}\right)^{\frac{1}{-1}\frac{3}{1}} \\ &\cdot \left(e(g,g)^{u \cdot (a_2 \cdot 3^2 + a_1 \cdot 3 + s_x)}\right)^{\frac{1}{-2} \cdot \frac{2}{-1}} = e(g,g)^{u s_x}, \end{aligned}$$

which is a correct output of the threshold gate.

When the algorithm in root node $\mathcal{R}$ returns true, it means we get $S = e(g,g)^{u \cdot s}$, which is the input parameter of the second step. In the second step, the algorithm is as follows:

$$\begin{aligned} \hat{K} &= \frac{\hat{C} \cdot S}{e(C, D)} \\ &= \frac{K \cdot e(g,g)^{\alpha s} e(g,g)^{us}}{e(h^s, g^{\frac{\alpha+u}{\beta}})} = K, \\ M &= D_{\hat{K}}(\tilde{C}). \end{aligned} \tag{9}$$

If $\mathcal{S}_2$ matches the policy $\mathcal{T}$, then the security key can be used to decrypt the data and get the information. Otherwise, the user cannot decrypt it even he downloads the file from the storage platform.

4.3 Scheme Variant Based on Asymmetric Bilinear Pairing

Our scheme can be easily extended to a scheme based on asymmetric bilinear pairings, which is more efficient than the one based on symmetric pairings. In what follows, we briefly present the extension, mainly the differences compared with the scheme presented above.

In the phase of Setup, the authority generates $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ with a prime order p, satisfying bilinear map $e: \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. Differently, g is a generator in $\mathbb{G}_1$, and the hash algorithm is defined as $H: (0,1)^* \rightarrow \mathbb{G}_2$. We also choose $\mathbb{G}_2$'s generator, denoted as g'. Accordingly, we substitute $(g')^\alpha$ for g^α as a component of MK, and substitute $e(g, g')^\alpha$ for $e(g,g)^\alpha$ as a component of PK. Other parameters are outputted as the same as before.

In the phase of Encrypt, nothing is changed. Be noted that C_y and C'_y are elements in $\mathbb{G}_1$ and $\mathbb{G}_2$.

In the phase of Key Generation, D and every D_y in security key is different: $D = (g')^{(\alpha+u)/\beta}$ and $D_y = (g')^u \cdot H(y)^{r_y}$, where u and r_y are randomly chosen in the same way as before.

In the phase of Decrypt, the user follows the same procedure as the scheme based on symmetric pairings. This extended scheme not only preserves the correctness of the basic scheme, but is also more efficient. Also, be noted that the security features are preserved as well, although our later confidentiality proof is based on the DBDH assumption on symmetric pairings.

4.4 The WSN Application Revisited

We can refer to a specific instance to show how the mechanism of 0-encoding and 1-encoding is embedded into *attribute-based encryption* in our scheme. We re-use the instance of wireless sensor network given in Section 1, where the access policy is "$(Distance < 1000)\ \ AND\,(Date > May\ 1st)$", and the attribute set is $\{Distance = 750, Date = July\ 7th, Owner = (experts, officers)\}$.

In our mechanism, the boolean attributes are not modified, such as "Owners" in this example. For comparable attributes, we first define the range of each attribute's value. For "*distance*", we assume that the value range is from 0 to 1,023, and we use 10-bit string to express every value. For "*date*", 9-bit string is sufficient to express 366 days, we use one integer to represent each day. In the example proposed in Section 1, one entity's attributes {Distance = *750 miles*, Date = *July 7th*, Owner = *(experts, officers)*} can be expressed as an attribute set $\mathcal{S}_0$: $\{Distance = 1011101110, Date = 010111101, experts, officers\}$.

After that, we modify the comparable attributes as Eq. (4) with the encoding scheme, and expand the set as shown in

TABLE 3
Expanded Set of Attributes

Attributes		Expanded Attribute Set
July 7th	0-encoding	$date \| \| < x \| \| 1$
		$date \| \| < x \| \| 011$
		$date \| \| < x \| \| 01011111$
	1-encoding	$date \| \| > x \| \| 01$
		$date \| \| > x \| \| 0101$
		$date \| \| > x \| \| 01011$
		$date \| \| > x \| \| 010111$
		$date \| \| > x \| \| 0101111$
		$date \| \| > x \| \| 010111101$
750	0-encoding	$dist \| \| < x \| \| 11$
		$dist \| \| < x \| \| 101111$
		$dist \| \| < x \| \| 1011101111$
	1-encoding	$dist \| \| > x \| \| 1$
		$dist \| \| > x \| \| 101$
		$dist \| \| > x \| \| 1011$
		$dist \| \| > x \| \| 10111$
		$dist \| \| > x \| \| 1011101$
		$dist \| \| > x \| \| 10111011$
		$dist \| \| > x \| \| 101110111$
Owner		experts
		officers

Table 3. The size of the attribute size is enlarged to 21 items from 4 items.

For an access policy "(Distance $<$ 1000) AND (Date $>$ May 1st)", we first translate the two values into two binary strings. "May 1st" is expressed as an integer "121"(Which means the 121st day of a year.). Then we generate 1000's 1-encoding and 121's 0-encoding. Finally, a CABE access policy tree is depicted as Fig. 3. In this figure, the first *OR* gate and its 6 child nodes jointly represent an attribute (*Distance* < 1000); And the second *OR* gate and its 4 child nodes jointly represent the other attribute (*Date* $>$ *May 1st*).

Because of the property of 0-encoding and 1-encoding, we can find "*date* $||$ $>$ x $||$ *01*" in the attribute set of the ciphertext; This item makes the first *OR* gate return a *true* result. And the item "*dist* $||$ $<$ x $||$ *11*" makes the second *OR* gate return *true*. At last, the root node returns true, meaning the access behaviour is allowed. It means the architecture works in a correct way.

We can also take other examples, in which the access policy will allow or ban access behaviors correctly. Therefore, with the correct structure based on 0-encoding and 1-encoding, and the secure algorithm of *attribute-based encryption*, CABE scheme provides an efficient access control mechanism, being able to deal with any numeric inequality.

5 Security and Performance Analysis

5.1 Security Analysis

We analyze the security properties of CABE according to the security model defined in Section 3.3.

5.1.1 Confidentiality

Theorem 1. *If an adversary can break CABE in polynomial time, then a simulator can be constructed to play the DBDH game with a non-negligible advantage.*

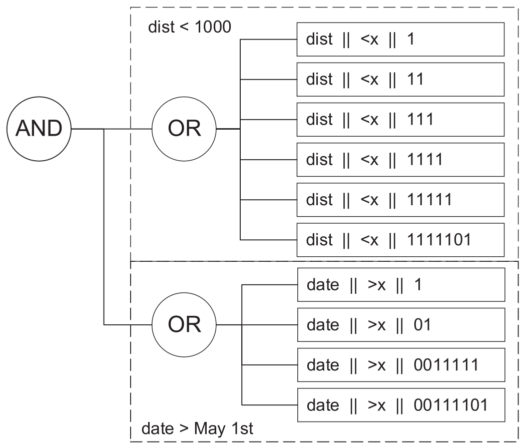


Fig. 3. Policy tree of CABE (The top 6 leaf nodes represent all elements, whose binary string is "1111101000", in 1-encoding of "1000". The 4 bottom leaf nodes represent all elements, whose binary string is "001111001", in 0-encoding of "121" (represents the date of May 1st)).

Proof. Suppose there exists a polynomial-time adversary $\mathcal{A}$, who can compromise our scheme with advantage ϵ. We build a simulator $\mathcal{B}$ who can play the DBDH game with advantage $\frac{\epsilon}{2}$. The simulation procedures are as follows:

At first, the challenger $\mathcal{C}$ of DBDH game sets the groups $\mathbb{G}_0$ and $\mathbb{G}_1$ with the bilinear map e and generator $g \in \mathbb{G}_0$. $\mathcal{C}$ securely flips a random coin $\mu \in (0, 1)$. If $\mu = 0$, $\mathcal{C}$ sets a tuple $(A, B, C, Z) = (g^a, g^b, g^c, e(g, g)^{abc})$; otherwise it sets $(A, B, C, Z) = (g^a, g^b, g^c, e(g, g)^z)$ for random a, b, c, z. Then, $\mathcal{C}$ sends (A, B, C, Z) to the simulator $\mathcal{B}$.

Setup. The simulator $\mathcal{B}$ sets $\mathbb{G}_0$, $\mathbb{G}_1$, e, and g from $\mathcal{C}$, and randomly chooses β, $\alpha \in \mathbb{Z}_p^*$. For each attribute A_i (including normal attribute and the sub-attribute of the comparable attribute), $\mathcal{B}$ uses g^{t_i} to formulate $H(A_i)$ in this simulation. It then gives $\mathcal{A}$ the public parameters as

$$\left(\mathbb{G}_0, \mathbb{G}_1, h = g^{\beta}, e(g, g)^{\alpha};\ \forall A_i :\ g^{t_i}\right).$$

Phase 1. $\mathcal{A}$ adaptively makes requests for security keys corresponding to attribute set $\mathcal{S}_U = (A_1, A_2, \ldots, A_{l_1}, \{F_1\}, \{F_2\}, \ldots, \{F_{l_2}\})$, and arbitrarily designs a challenge access policy $\mathcal{T}$ such that none subset of $\mathcal{S}_U$ satisfies $\mathcal{T}$. Especially for comparable attribute, each pair F_i and F_j $(\forall i \neq j)$ can be either in the same attribute dimension or in different ones. Let $\mathcal{S}_T$ denote the attribute set in $\mathcal{T}$, and Γ denote the subset of $\mathcal{S}_T - \mathcal{S}_U$, such that none of $\mathcal{S}_T - \Gamma$ satisfy the policy of $\mathcal{T}$. $\mathcal{B}$ accepts the requests, it randomly selects r, and r_i for each element in $\mathcal{S}_U$. It computes the global security key $D = (C \cdot g^{\alpha})^{1/\beta}$, and then computes each attribute key as $(D_i = C \cdot (g^{t_i})^{r_i}, D'_i = g^{r_i})$ for each $A_i \in \mathcal{S}$. Then $\mathcal{B}$ returns the above created security key to $\mathcal{A}$.

Particularly for a comparable attribute dimension, because of the property of *0-encoding* and *1-encoding*, if the attribute value is not in the range of the policy, then there is no common element (or the same sub-attribute) between $\mathcal{T}$ and S_U for this dimension. Detailed Security analysis of the comparable attributes are proposed in Section 5.1.3.

Before the *Challenge* procedure, we first define the following two procedures: *PolySat* and *PolyUnsat*.

PolySat$(\mathcal{T}_x, s_x)$. This procedure sets up the polynomials for the nodes of a sub-tree with satisfied root node

x, that is $\mathcal{S}_U$ satisfies the access policy of $\mathcal{T}_x$. It first sets a polynomial q_x of degree d_x for x, and sets $q_x(0) = s_x$. Each child node y will obtain $s_y = q_x(index_y)$ in this procedure. Now it sets polynomials for each child node y by calling *PolySat*($\mathcal{T}_y, s_y$).

PolyUnsat($\mathcal{T}_x, g^{s_x}$). This procedure sets up the polynomials for nodes of an access tree with unsatisfied root node, that is $\mathcal{S}_U$ does not satisfy $\mathcal{T}_x$. It first defines a polynomial q_x of degree d_x for x, such $g^{q_x(0)} = g^{s_x}$. Because x is an unsatisfied node, no more than d_x child nodes of are satisfied. For each satisfied node y, the procedure chooses a random $s_y \in \mathbb{Z}_p$. It then fixes the remaining unsatisfied points of q_x to completely define q_x. The procedure recursively defines the polynomials for the child nodes of x by calling:

- PolySat($\mathcal{T}_y, q_x(index_y)$), if y is a satisfied node. $\mathcal{B}$ knows the value $s_y = q_x(index_y)$ in this case.
- PolyUnsat($\mathcal{T}_y, g^{q_x(index_y)}$), if y is not a satisfied node. In this case, only $g^{s_y} = g^{q_x(index_y)}$ is known.

Notice that in both of the two procedures, $q_y(0) = q_x(index_y)$ for each child node y of x. The procedure with satisfied and unsatisfied comparable attributes belongs to PolySat and PolyUnsat, respectively.

When receiving the access structure $\mathcal{T}$, $\mathcal{B}$ first runs *PolyUnsat* ($\mathcal{T}, A$) to define polynomials for each node $x \in \mathcal{T}$. Notice that for each attribute $i \in \mathcal{S}_T$, we know q_x completely of the corresponding node x if $i \notin \Gamma$; if x is not satisfied, then we only have knowledge of $g^{q_x(0)}$. In addition, the base secret of $\mathcal{T}$ is a from DBDH game.

Challenge. $\mathcal{A}$ submits two challenge messages M_0 and M_1 to $\mathcal{B}$, and $\mathcal{B}$ flips a secure coin $\nu \in (0, 1)$. For each attribute i in $\mathcal{S}_T$, if $i \in \Gamma$, $C_i = B^{q_i(0)}$, $C'_i = (B^{t_i})^{q_i(0)}$; otherwise, $C_i = g^{q_i(0)}$, $C'_i = (g^{t_i})^{q_i(0)}$. $\mathcal{B}$ returns the following CT^* to $\mathcal{A}$

$$CT = \Big(\mathcal{T}, M_\nu \cdot \frac{e(C \cdot g^\alpha, A)}{Z}, h^s = A^\beta, \{C_i, C'_i\}_{i \in \mathcal{S}_T}\Big).$$

Therefore, $\mathcal{B}$ is able to simulate the scheme. Furthermore, from the perspective of $\mathcal{A}$, the distribution of each component is identical to that in the original scheme.

If $\mu = 0$, then $Z = e(g,g)^{abc}$. We let the attribute key of unsatisfied attribute be $D_i = g^{bc} \cdot (g^{t_i})^{r_i}$, $D'_i = g^{r_j}$. We assume the lagrange interpolation for a secret s is: $s = \Sigma_i \lambda_i \cdot x_i$ for some share set $\{x_i\}$. Then for any sub-set $S'_T \subseteq \mathcal{S}_T$, that satisfies $\mathcal{T}$, we can find λ_{x_i} for each attribute $i \in S_T$ such that $\sum_{i \in S'_T} \lambda_{x_i} \cdot q_{x_i}(0) = a$. We then have reconstruction of F_R as

$$\begin{aligned} F_R &= \prod_{i \in S'_T} F_{x_i}^{\lambda_{x_i}} = \prod_{i \in S'_T} \Big(\frac{D_i, C_{x_i}}{D'_i, C'_{x_i}}\Big)^{\lambda_{x_i}} \\ &= (e(g,g)^{bc})^{\sum_{i \in S'_T} \lambda_{x_i} q_{x_i}(0)} \\ &= e(g,g)^{abc}. \end{aligned} \tag{10}$$

Therefore, CT^* is a valid random encryption of M_ν.

Otherwise, if $\mu = 1$, the $Z = e(g,g)^z$ is only a random element from $\mathbb{G}_2$ from the view of $\mathcal{A}$, and such CT^* contains no information about M_ν.

Phase 2. Repeat Phase 1 adaptively. $\mathcal{A}$ cannot obtain security key associated to any normal or comparable attribute in Γ.

Note that this phase is not exactly the same as Phase 2 in the proposed security model in Section 3.3. If $\mathcal{A}$ applies for some attributes from Γ, the ultimate attribute set may still do not satisfy the challenge access policy; however, in this simulation, $\mathcal{B}$ cannot generate the corresponding security key, whose form is as $D_i = g^{bc} \cdot (g^{t_i})^{r_i}$, $D'_i = g^{r_i}$. Due to the above reason, the simulation has to abort. Whereas, such abortion is not due to the security problem of CABE, and our analysis does not take such issue into consideration.

Guess. $\mathcal{A}$ will submits a guess ν' of ν. If $\nu' = \nu$, $\mathcal{B}$ will output its own guess $\mu' = 0$ to indicate that the tuple of DBDH game is a valid BDH-tuple. Otherwise it outputs $\mu' = 1$ to indicate it was given a random 4-tuple.

We assume the distribution of μ and ν is independent. In this case, where $\mu = 1$ $\mathcal{A}$ obtains no information about ν. We have $Pr[\nu \neq \nu' | \mu = 1] = \frac{1}{2}$. Since $\mu' = 1$ when $\nu \neq \nu'$, we have $Pr[\mu' = \mu | \mu = 1] = \frac{1}{2}$.

Otherwise if $\mu = 0$, CT^* is a valid encryption of M_ν. The adversary $\mathcal{A}$ has an advantage ϵ by definition. We have $Pr[\nu = \nu' | \mu = 0] = \frac{1}{2} + \epsilon$. Since $\mathcal{B}$ will guess $\mu' = 0$ when $\nu = \nu'$, we have $Pr[\mu' = \mu | \mu = 0] = \frac{1}{2} + \epsilon$.

The overall advantage of $\mathcal{B}$ in DBDH game is

$$\begin{aligned} Adv_{\mathcal{B}} &= Pr[\mu' = \mu | \mu = 1]\frac{1}{2} + Pr[\mu' = \mu | \mu = 0]\frac{1}{2} - \frac{1}{2} \\ &= \frac{1}{2}\frac{1}{2} + \frac{1}{2}(\frac{1}{2} + \epsilon) - \frac{1}{2} = \frac{1}{2}\epsilon. \end{aligned} \tag{11}$$

As proved above, there exists a non-negligible polynomial-time adversary $\frac{\epsilon}{2}$ in DBDH game if the advantage for the polynomial-time adversary in our scheme is ϵ. We can conclude that our scheme is semantically secure against *chosen plaintext attack*. □

5.1.2 Security Against Collusion Attack

Inherited from CP-ABE, the system is secure against the collusion attack. Because when a user deals with the policy tree $\mathcal{T}$, the secret of each node is blinded by user's unique random number u.

Let's consider a file M embedded with an access policy $\mathcal{T}$, and assume that there is an unauthorized user U_i who cannot satisfy the policy because of the lack of an attribute y_A. Meanwhile, User U_j, who has attribute y_A, is assumed to try to collude with U_i. We can revisit the user's security key for U_i as

$$SK = \Big\{D = g^{\frac{\alpha+u}{\beta}};\ \forall y \in \mathcal{S}_2 :\ D_y = g^u \cdot H(y)^{r_y},\ D'_y = g^{r_y}\Big\}.$$

Here $\mathcal{S}_2$ can be divided into two sets: $\mathcal{S}_a$ and $\mathcal{S}_b$, where $\mathcal{S}_a$ is the attribute set that is for the decryption of M, and $\mathcal{S}_b$ is the set of remaining attributes. Furthermore, assume $\mathcal{S}_A \bigcup \{y_A\}$ can satisfy $\mathcal{T}$, while $\mathcal{S}_a$ cannot. In U_j's security key, there exists $D_{y_A} = g^{u_j} \cdot H(y)^{r_{y_A}}$, $D'_{y_A} = g^{r_{y_A}}$. As mentioned in Section 5.1.1, D_y, D'_y for $y \in \mathcal{S}_b$ can be easily simulated in the security game proved in Proof 1, which means that they cannot provide any advantages for any adversaries. On the other hand, in U_j's security key, only the presented components can be useful for U_i in this collusion.

For attribute set $\mathcal{S}_A \bigcup \{y_A\}$ and access policy $\mathcal{T}$, there exists a set of constants (denoted as l_k) that satisfy

TABLE 4
Example: Common Element of S

Attribute	Legal Encoding	Illegal Encoding
$x > 11$	$S_x^0 = \{11\}$	
$y = 9$	$S_y^1 = \{1, 1001\}$	$S_y^0 = \{11, 101\}$

$$\sum_{k \in \mathcal{S}_A \bigcup \{y_A\}} l_k \cdot q_k = s.$$

where $\{q_k\}$ $(k \in \mathcal{S}_A \bigcup \{y_A\})$ are the secret values of the attributes in $\mathcal{S}_A \bigcup \{y_A\}$ and s is the secret value of root node in $\mathcal{T}$. Due to the lack of y_A, U_i can get only $e(g,g)^{u_i(s - l_{y_A} q_{y_A})}$ with his/her own security key. With U_j's D_{y_A}, D'_{y_A}, $e(g,g)^{u_j \cdot q_{y_A}}$ can be generated. However, as $u_j \neq u_i$, and they are unknown to users, $e(g,g)^{u_j \cdot q_{y_A}}$ may traverse all elements in $\mathbb{G}_1$ with different u_j. Thus, it cannot get any information from this attack. In conclusion, U_i cannot get any help from U_j. The result is still the same when we reverse the roles of U_i and U_j.

Thus, it is impossible to decrypt the ciphertext with attributes owned by different users.

5.1.3 Security for Comparable Attributes

The correctness of 0-encoding and 1-encoding is proved in Lin et al.'s work [17]. Due to the theorem of Eq. (1), if a user's attribute value x meets the access policy($> y$ or $< y$), there must be a common element between $S_{x_m}^1$ and $S_{y_m}^0$, or between $S_{x_m}^0$ and $S_{y_m}^1$ respectively. With the access structure proposed in Fig. 1, the match algorithm will succeed. If user's attribute does not meet access policy, two encoded sets satisfy the relationship: $S_{x_m}^{1/0} \bigcap S_{y_m}^{0/1} = \varnothing$, meaning the match will fail according to the model in Fig. 1.

While we append two additional parts to elements of 0-encoding and 1-encoding as Eqs. (2), (3), and (4), we can resist dishonest actions misusing encoded elements or using other attributes. The first part of the element determines the attribute field to prevent dishonest users from combining attributes across different fields. The second part of the element determines the type of encoding, resisting dishonest users getting illegal information when they misuse encodings.

Table 4 shows an example that a user with the attribute "9", which should be denied in the policy $x > 11$, but will pass the match phrase with its 0-encoding. While in our CABE scheme, although the original 0-encodings have a common element, the second parts of the elements are different, resisting dishonest users getting benefits from such attacks. Eqs. (2) and (4) shows it clearly that: $Set_{c0}(\cdot)$ contains the second part "$> x$", but $Set_{u0}(\cdot)$ has "$< x$", which makes the two sets different and cannot be matched to each other.

5.2 Performance Analysis

We conduct an experiment to execute the functions of data encryption, key generation and data decryption. The running environment is a standard 64-bit Fedora release 21 Operating System with Intel (R) Core (TM) i3-4130 3.40 GHz. The program environment is C based, which uses PBC library 0.5.14 with type-A1 curve, and partly refers to the software implementation in [43]. Furthermore, the symmetric-key encryption/decryption is realized with OpenSSL 1.0.1k with AES256 and CFB mode, where the symmetric key is through a hash function from $\mathbb{G}_1$ to 256-bit standard key. The plaintext is a 15.0-MB compressed file for each encryption and decryption.

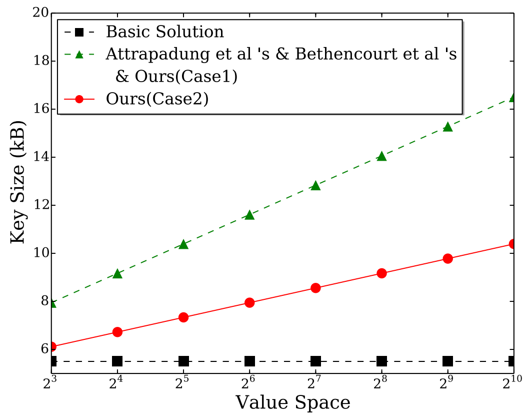


Fig. 4. Comparison of security key size.

All evaluations are executed for CP-ABE prototype. Except for the value space of the attribute number, all parameters for the access policy and user's security key are the same in our evaluation: 1) An access policy includes 6 boolean attributes and 1 comparable attribute, where the user on average needs 4 boolean attributes and the comparable one to decrypt the relevant ciphertext; 2) In user's security key, the attribute number is a constant, with one comparable attribute and some other boolean attributes.

We evaluate the performance in terms of storage and computation compared with three most related works: 1) A trivial but common method as presented in Section 1 (denoted as "Basic Solution"), which simply expands the attribute on the ciphertext side; 2) Bethencourt et al.'s mechanism [16] (denoted as "Bethencourt et al.'s"); 3) The range attribute scheme proposed in [36], (denoted as "Attrapadung et al.'s"). Also, as our scheme supports diverse cases, it is classified into two in our evaluation:

(1) Case 1: The attribute that needs to prepare both 0-encoding and 1-encoding for the number point.
(2) Case 2: The attribute that needs only one kind of the encodings (0-encoding or 1-encoding).

5.2.1 Storage Overhead

Fig. 4 illustrates the size of user's security key versus the value space of the comparable attribute, which represents the key storage overhead comparison of the different schemes. On this aspect, the basic solution shows the best performance, and other schemes perform exactly the same, except our scheme in Case 2 (only one kind of attribute inequality is required). This evaluation shows that, if the comparable attribute only needs one number inequality ("$>$" or "$<$") in policy match, our scheme can significantly reduce the overhead of security key storage.

The result meets the theoretical analysis, as shown in Table 5 (the item "Security Key Extension"). In the basic solution, as it only sacrifices the ciphertext performance, no sub-attribute should be generated for the attribute number point, the key size is not affected by the attribute number

TABLE 5
Performance Analysis Result with Theoretic Derivation

Overhead(AVG)	Basic Solution	Bethencourt et al.'s	Attrapadung et al.'s	Ours (Case1)	Ours (Case2)
Extended Attributes in Access Policy	$N/2$	$\log_2 N + \frac{\log_2 N - N + 1}{N}$	$N/2$	$\log_2 N/2$	$\log_2 N/2$
Extended Thresholds in Access Policy	1	$(\log_2 N - 1)/2$	$1 - \log_2 N/N$	$1 - \log_2 N/N$	$1 - \log_2 N/N$
Security Key Extension	1	$\log_2 N$	$\log_2 N$	$\log_2 N$	$\log_2 N/2$

The object in this table is the average expanded overhead for an attribute field with value space N (We assume that $N = 2^n$, $n \in N^*$).

value size. While in other schemes, sub-attributes are generated according to the length of its binary string. Whereas, in our scheme, 0-encoding and 1-encoding respectively aims at one kind of number inequality, the number of the sub-attributes can be halved in Case 2.

To compare the ciphertext size, we first visit the access policy of Bethencourt et al.'s scheme [16] for attribute comparison with a brief instance in Fig. 6, which represents a number comparison "< 11". The left structure depicts Bethencourt et al.'s scheme, and the right one is ours. Attrapadung et al.'s scheme outputs a similar structure with ours. From this figure, we can intuitively find that the sub-policy for attribute comparison is far less complex in our scheme than in Bethencourt et al.'s.

To compare these schemes generally and objectively, we give the efficiency analysis of our scheme in detail, and present the comparison result in Table 5.

Fig. 5 shows the comparison of ciphertext size versus the attribute value space, which doesn't include the symmetric encryption result of the file. The experimental results show the superiority of Attrapadung et al.'s scheme and our scheme compared with other schemes. Note that the huge size of the ciphertext produced by the basic solution put unbearable burden on the system. Moreover, the increased ciphertext size will introduce much higher communication overhead, since ciphertext needs to be transferred between end entities and storage platforms. Basically, our scheme trades using a bit more security key storage for reducing the ciphertext size significantly, which can be verified by the experimental results shown in Figs. 4 and 5.

From the theoretic analysis, the difference of ciphertext size results from the different complexities of the access policies between the compared schemes, especially the number of leaf nodes in a policy is the most important critical factor. Table 5 presents the comparison of extended sub-attributes and threshold gates for the attribute comparison. Especially, the items of our scheme is analyzed as follows.

Theorem 2. *In CABE construction, for a comparable attribute $\mathcal{F}$ with a value space of N, the average number of expanded leaf nodes is $1/2\lceil\log_2 N\rceil$, and the average number of additional OR gates is $1 - \frac{\lceil\log_2 N\rceil}{N}$.*

Proof. A $\lceil\log_2 N\rceil$-length binary string is able to express all values in $\mathcal{F}$ with the range of N. We assume N equals to 2^n (n is an arbitrary integer) to make the analysis described briefly. When one policy is $\mathcal{F} < x$, construct 1-encoding for x. While in an arbitrary n-length binary string, the value "0" and "1" appear in any bit independently with the probability of 1/2. The expectation of the occurrence of "1" in x is

$$E(Num_1) = \sum_{i=1}^{n} P_i(1) \cdot 1 = \frac{n}{2}. \tag{12}$$

In Eq. (12), $P_i(1)$ presents the probability of the case as "1" appears in the bit i. As $n = \log_2 N$, we can also express the result in another form as: $E(Num_1) = \frac{\log_2 N}{2}$.

According to the concept of 1-encoding, the number of value "1" in x equals to the size of x's 1-encoding S_x^1, and also equals to the size of $Setc1(\mathcal{F}, x)$. In a word, the number of "1" in x determines the expanded leaf nodes of the attribute $\mathcal{F} < x$. All conclusion suits the case when $\mathcal{F} > x$. To integrate the above results, the average number of leaf nodes for a comparable attribute is $1/2\log_2 N$.

To prove the second part of the theorem, two cases are considered respectively. When there's only one element in the constructed 0-encoding or 1-encoding, there's no additional threshold gate, and the probability of this case is

$$\begin{aligned} P(Num_G = 0) &= \binom{\log_2 N}{1} \cdot P(1) \cdot P(0)^{\log_2 N - 1} \\ &= \log_2 N \cdot \tfrac{1}{2} \cdot (\tfrac{1}{2})^{\log_2 N - 1} \\ &= \frac{\log_2 N}{N}. \end{aligned} \tag{13}$$

In Eq. (13), $P(1)$ represents the probability of the occurrence of "1" in one bit, so as to $P(0)$. When number

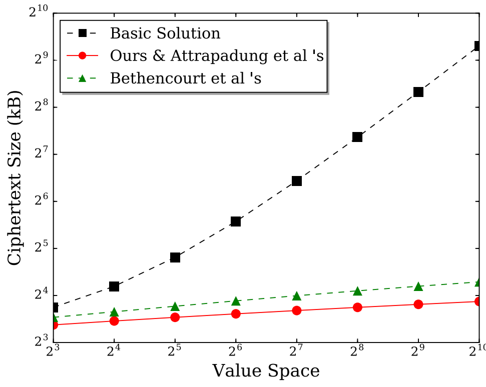


Fig. 5. Comparison of ciphertext size.

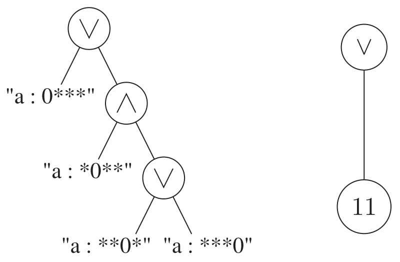


Fig. 6. Comparison of policy structure "< 11".

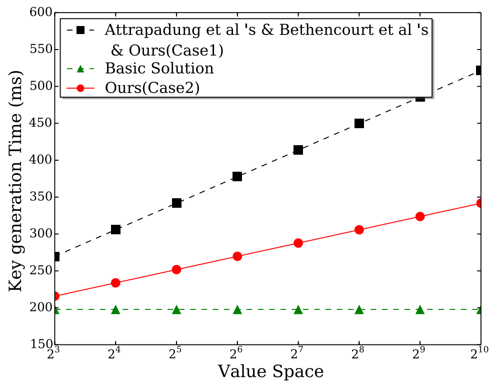


Fig. 7. Computation overhead of SecurityKey generation.

of elements is more than one, an OR gate is necessary. Then the expectation of involved gates is calculated as

$$E(Num_G) = P(Num_G = 0) \cdot 0 + P(Num_G = 1) \cdot 1 = 1 - \frac{\log_2 N}{N}. \quad (14)$$

Thus, the expected number of extended leaf nodes and that of threshold gates are derived. □

5.2.2 Computation Overhead

In this section, we evaluate the computation overhead in terms of the following procedures: key generation, encryption and decryption.

Fig. 7 illustrates the computation overhead of security key generation. In this evaluation, apart from the value space of the comparable attribute, all the other factors (e.g., security parameter, attribute number) are controlled as the same. As Fig. 7 shows, our scheme in Case 2 is more efficient than the other schemes except the basic solution, while our scheme in Case 1 has the same computation overhead of security key generation with Attrapadung et al.'s scheme and Bethencourt et al.'s scheme.

The processing time of encryption and decryption are shown in Figs. 8 and 9, respectively. It should be noted that the *basic solution* takes far longer time than our experiment can tolerate to encrypt a file, e.g., when value space reaches 2^7, the average encryption time has exceeded 2700 ms, which is unbearable for many scenarios. Thus, Fig. 8 only shows the comparison of the rest schemes.

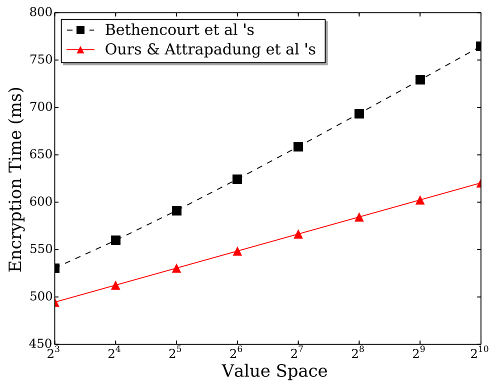


Fig. 8. Computation overhead of encryption.

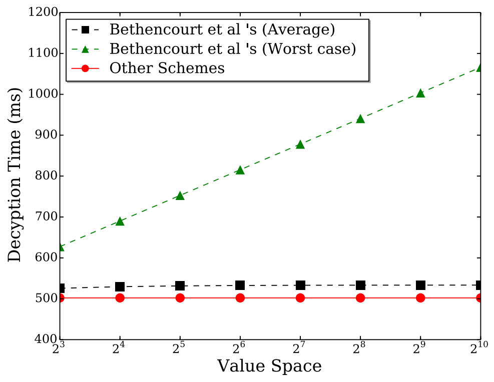


Fig. 9. Computation overhead of decryption (except Bethencourt et al.'s scheme, all schemes in comparison show the same performance in this term, thus denoted as *other schemes*).

Fig. 8 shows that our scheme and Attrapadung et al's far outperform Bethencourt et al.'s scheme, as our scheme almost halves the number of sub-attributes for the comparable attributes, which is analyzed in Table 5.

In terms of the decryption computation, for each scheme, we set the comparable attribute to be with the same attribute value space and diverse number ranges/points, and conduct experiments to decrypt the encrypted file. The evaluation results are shown in Fig. 9, and one can see that the decryption time in Bethencourt et al.'s scheme varies greatly and it spends much longer time than other schemes, while the decryption time in other schemes is not affected by the value space.

Hence, Fig. 9 also depicts the average and worst-case decryption time of Bethencourt et al.'s scheme. From the figure, we can see that its average decryption time increases very slowly by increasing the attribute value space, and the gap of average performance with our scheme is not too large. However, its worst-case decryption time increases drastically when the attribute value space increases. This shows that the system has to take non-ignorable computation overhead when the value space is large enough.

The evaluation of computation shows that our scheme takes a bit more overhead for the key generation procedure, compared with the basic scheme. In our experiment, when the value space of the comparable attribute becomes 2^{10}, the basic solution takes about 198 ms to generate security key for a user, while in our scheme, the computation overhead is 520 and 342 ms in Case 1 and Case 2, respectively. This sacrifice is acceptable since the gap is not too large. Meanwhile, key generation is conducted by the authority, rather than users' device, thus, the performance will not be affected by ABE's application scenarios or the energy-constraint devices. Note that with the acceptable sacrifice, the performance of encryption and decryption in our scheme improves greatly: As mentioned above, the basic solution takes unbearable computation overhead for encryption, and it thus cannot be applied to attribute comparison well; Bethencourt et al.'s scheme needs higher computation overhead in encryption and decryption than our scheme (e.g., when value space is 2^{10}, it consumes 144 ms more time to encrypt a file on average, and 31 ms more time to decrypt the file).

Based on the performance evaluation, our scheme can well tolerate the increasing value space of the comparable attributes. Additionally, compared with Bethencourt et al.'s and Attrapadung et al.'s scheme, our scheme does not require extra data structure to generate the sub-attributes for number point and number inequality, which makes it more suitable to be applied in power-constrained devices.

6 Conclusion

In this paper, we constructed an efficient solution to apply comparable attributes to attribute-based encryption, namely CABE, based on the notion of 0-encoding and 1-encoding. Our attempt is to make attribute-based encryption available to involve attribute comparison in access policy. Both analysis and experimental results show that compared to the best of existing works as far as we know, our scheme achieves attribute comparison with better performance: 1) halving the storage overhead of the involved sub-attributes on average; 2) reducing the additional computation overhead from $\Theta(\log N)$ to $\Theta(1)$. The decreased sub-attributes economize not only the data storage platform, but also the communication overhead, because the involved sub-attributes need to be transported along with ciphertexts in the network. In a nut shell, we trade using a bit more security key storage for reducing the ciphertext size significantly.

Therefore, we believe that our research can help make attribute-based encryption satisfy more diversified requirements in access control. Also, in CABE and other analogous schemes, a sub-attribute update may influence quite a number of users and ciphertexts with different attributes. Therefore, our future work may consider to develop the mechanisms, such as an implementation of a revocation method in CABE, with higher efficiency and stronger security.

Acknowledgments

The authors sincerely thank the anonymous referees for their invaluable suggestions that have led to the present improved version of the original manuscript. This work is supported in part by the National Natural Science Foundation of China under Grant No. 61379129 and No. 61671420, Youth Innovation Promotion Association CAS, and the Fundamental Research Funds for the Central Universities.

References

[1] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Proc. 24th Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 2005, pp. 457–473.

[2] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," in *Proc. Annu. Int. Cryptology Conf.*, 2001, pp. 213–229.

[3] T. Jung, X.-Y. Li, Z. Wan, and M. Wan, "Privacy preserving cloud data access with multi-authorities," in *Proc. 32nd IEEE Int. Conf. Comput. Commun.*, 2013, pp. 2625–2633.

[4] K. Yang, X. Jia, K. Ren, and B. Zhang, "DAC-MACS: Effective data access control for multi-authority cloud storage systems," in *Proc. 32nd IEEE Int. Conf. Comput. Commun.*, 2013, pp. 2895–2903.

[5] W. Li, K. Xue, Y. Xue, and J. Hong, "TMACS: A robust and verifiable threshold multi-authority access control system in public cloud storage," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 5, pp. 1484–1496, May 2016.

[6] J. Shao, R. Lu, and X. Lin, "Fine-grained data sharing in cloud computing for mobile devices," in *Proc. 34th IEEE Int. Conf. Comput. Commun.*, 2015, pp. 2677–2685.

[7] K. Xue, et al., "RAAC: Robust and auditable access control with multiple attribute authorities for public cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 4, pp. 953–967, Apr. 2017.

[8] L. Guo, C. Zhang, J. Sun, and Y. Fang, "PAAS: A privacy-preserving attribute-based authentication system for ehealth networks," in *Proc. 32nd IEEE Int. Conf. Distrib. Comput. Syst.*, 2012, pp. 224–233.

[9] S. Narayan, M. Gagné, and R. Safavi-Naini, "Privacy preserving EHR system using attribute-based infrastructure," in *Proc. ACM Workshop Cloud Comput. Security Workshop*, 2010, pp. 47–52.

[10] M. Li, S. Yu, Y. Zheng, K. Ren, and W. Lou, "Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 1, pp. 131–143, Jan. 2013.

[11] S. Yu, K. Ren, and W. Lou, "FDAC: Toward fine-grained distributed data access control in wireless sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 4, pp. 673–686, Apr. 2011.

[12] R. Baden, A. Bender, N. Spring, B. Bhattacharjee, and D. Starin, "Persona: An online social network with user-defined privacy," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 4, pp. 135–146, 2009.

[13] W. Dong, V. Dave, L. Qiu, and Y. Zhang, "Secure friend discovery in mobile social networks," in *Proc. 30th IEEE Int. Conf. Comput. Commun.*, 2011, pp. 1647–1655.

[14] S. Jahid, P. Mittal, and N. Borisov, "Easier: Encryption-based access control in social networks with efficient revocation," in *Proc. 6th ACM Symp. Inf. Comput. Commun. Security*, 2011, pp. 411–415.

[15] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. 13th ACM Conf. Comput. Commun. Security*, 2006, pp. 89–98.

[16] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. 28th IEEE Symp. Security Privacy*, 2007, pp. 321–334.

[17] H.-Y. Lin and W.-G. Tzeng, "An efficient solution to the millionaires' problem based on homomorphic encryption," in *Proc. Int. Conf. Appl. Cryptography Netw. Security*, 2005, pp. 456–466.

[18] R. Ostrovsky, A. Sahai, and B. Waters, "Attribute-based encryption with non-monotonic access structures," in *Proc. 14th ACM Conf. Comput. Commun. Security*, 2007, pp. 195–203.

[19] N. Attrapadung, B. Libert, and E. Panafieu, "Expressive key-policy attribute-based encryption with constant-size ciphertexts," in *Proc. 14th Int. Conf. Practice Theory Public Key Cryptography*, 2011, pp. 90–108.

[20] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *Proc. Int. Workshop Public Key Cryptography*, 2011, pp. 53–70.

[21] R. Bobba, H. Khurana, and M. Prabhakaran, "Attribute-sets: A practically motivated enhancement to attribute-based encryption," in *Proc. 28th Eur. Symp. Res. Comput. Security*, 2009, pp. 587–604.

[22] M. Chase, "Multi-authority attribute based encryption," in *Proc. Theory Cryptography Conf.*, 2007, pp. 515–534.

[23] L. Cheung and C. Newport, "Provably secure ciphertext policy ABE," in *Proc. 14th ACM Conf. Comput. Commun. Security*, 2007, pp. 456–465.

[24] V. Odelu, A. K. Das, Y. S. Rao, S. Kumari, M. K. Khan, and K.-K. R. Choo, "Pairing-based CP-ABE with constant-size ciphertexts and secret keys for cloud environment," *Comput. Standards Interfaces*, (2016). [Online]. Avaliable: https://doi.org/10.1016/j.csi.2016.05.002

[25] F. Guo, Y. Mu, W. Susilo, D. S. Wong, and V. Varadharajan, "CP-ABE with constant-size keys for lightweight devices," *IEEE Trans. Inf. Forensics Security*, vol. 9, no. 5, pp. 763–771, May 2014.

[26] J. Kim, W. Susilo, M. H. Au, and J. Seberry, "Adaptively secure identity-based broadcast encryption with a constant-sized ciphertext," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 3, pp. 679–693, Mar. 2015.

[27] C. Chen, Z. Zhang, and D. Feng, "Efficient ciphertext policy attribute-based encryption with constant-size ciphertext and constant computation-cost," in *Proc. Int. Conf. Provable Security*, 2011, pp. 84–101.

[28] N. Doshi and D. Jinwala, "Hidden access structure ciphertext policy attribute based encryption with constant length ciphertext," in *Proc. Int. Conf. Advanced Comput. Netw. Security*, 2011, pp. 515–523.

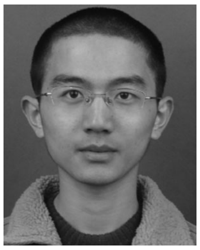
[29] J. Li, X. Huang, J. Li, X. Chen, and Y. Xiang, "Securely outsourcing attribute-based encryption with checkability," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 8, pp. 2201–2210, Aug. 2014.

[30] S. Hohenberger and B. Waters, "Online/offline attribute-based encryption," in *Proc. Int. Workshop Public Key Cryptography*, 2014, pp. 293–310.

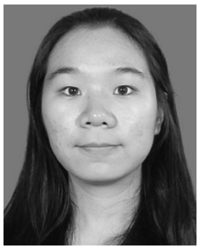
[31] K. Yang, X. Jia, and K. Ren, "Secure and verifiable policy update outsourcing for big data access control in the cloud," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 12, pp. 3461–3470, Dec. 2015.

[32] K. Yang, Z. Liu, X. Jia, and X. Shen, "Time-domain attribute-based access control for cloud-based video content sharing: A cryptographic approach," *IEEE Trans. Multimedia*, vol. 18, no. 5, pp. 940–950, May 2016.
[33] J. Hong, et al., "TAFC: Time and attribute factors combined access control for time-sensitive data in public cloud," *IEEE Trans. Serv. Comput.*, (2017). [Online] Avaliable: https://doi.org/10.1109/TSC.2017.2682090
[34] M. Ion, G. Russello, and B. Crispo, "Design and implementation of a confidentiality and access control solution for publish/subscribe systems," *Comput. Netw.*, vol. 56, no. 7, pp. 2014–2037, 2012.
[35] Z. Wan, J. Liu, and R. H. Deng, "HASBE: A hierarchical attribute-based solution for flexible and scalable access control in cloud computing," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 2, pp. 743–754, Apr. 2012.
[36] N. Attrapadung, G. Hanaoka, K. Ogawa, G. Ohtake, H. Watanabe, and S. Yamada, "Attribute-based encryption for range attributes," in *Proc. Int. Conf. Security Cryptography Netw.*, 2016, pp. 42–61.
[37] J. Katz, A. Sahai, and B. Waters, "Predicate encryption supporting disjunctions, polynomial equations, and inner products," in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 2008, pp. 146–162.
[38] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," in *Proc. Theory Cryptography Conf.*, 2007, pp. 535–554.
[39] E. Shi, J. Bethencourt, T. H. Chan, D. Song, and A. Perrig, "Multi-dimensional range query over encrypted data," in *Proc. IEEE Symp. Security Privacy*, 2007, pp. 350–364.
[40] H. Lin, Z. Cao, X. Liang, M. Zhou, H. Zhu, and D. Xing, "How to construct interval encryption from binary tree encryption," in *Proc. Int. Conf. Appl. Cryptography Netw. Security*, 2010, pp. 19–34.
[41] A. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters, "Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption," in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 2010, pp. 62–91.
[42] A. C.-C. Yao, "Protocols for secure computations," in *Proc. 23rd IEEE Symp. Found. Comput. Sci.*, 1982, pp. 160–164.
[43] E. Zavattoni, L. J. D. Perez, S. Mitsunari, A. H. Sánchez-Ramírez, T. Teruya, and F. Rodríguez-Henríquez, "Software implementation of an attribute-based encryption scheme," *IEEE Trans. Comput.*, vol. 64, no. 5, pp. 1429–1441, May 2015.

Kaiping Xue (M'09-SM'15) received the BS degree from the Department of Information Security, University of Science and Technology of China (USTC), in 2003 and the PhD degree from the Department of Electronic Engineering and Information Science (EEIS), USTC, in 2007. Currently, he is an associate professor in the Department of Information Security and Department of EEIS, USTC. His research interests include future Internet, distributed networks, and network security. He is a senior member of the IEEE.



Jianan Hong received the BS degree from the Department of Information Security, University of Science and Technology of China (USTC), in 2012. He is currently working toward the PhD degree in information security in the Department of Electronic Engineering and Information Science(EEIS), USTC. His research interests include secure cloud computing and mobile network security.



Yingjie Xue received the BS degree from the Department of Information Security, University of Science and Technology of China (USTC), in July, 2015. Currently, she is working toward the graduate degree in communication and information system in the Department of Electronic Engineering and Information Science(EEIS), USTC. Her research interests include network security and cryptography.

David S. L. Wei (SM'07) received the PhD degree in computer and information science from the University of Pennsylvania, in 1991. He is currently a professor of Computer and Information Science Department, Fordham University. From May 1993 to August 1997, he was on the Faculty of Computer Science and Engineering, University of Aizu, Japan (as an associate professor and then a professor). He has authored and co-authored more than 100 technical papers in various archival journals and conference proceedings. He served on the program committee and was a session chair for several reputed international conferences. He was a lead guest editor of the *IEEE Journal on Selected Areas in Communications* for the special issue on Mobile Computing and Networking, a lead guest editor of the *IEEE Journal on Selected Areas in Communications* for the special issue on Networking Challenges in Cloud Computing Systems and Applications, a guest editor of the *IEEE Journal on Selected Areas in Communications* for the special issue on Peer-to-Peer Communications and Applications, and a lead guest editor of the *IEEE Transactions on Cloud Computing* for the special issue on Cloud Security. He is currently an associate editor of the *IEEE Transactions on Cloud Computing*, an associate editor of the *Journal of Circuits, Systems and Computers*, and a guest editor of the *IEEE Transactions on Big Data* for the special issue on Trustworthiness in Big Data and Cloud Computing Systems. Currently, his research interests include cloud computing, big data, IoT, and cognitive radio networks. He is a senior member of the IEEE.

Nenghai Yu received the BS degree from the Nanjing University of Posts and Telecommunications, Nanjing, China, in 1987, the ME degree from Tsinghua University, Beijing, China, in 1992, and the PhD degree from the University of Science and Technology of China, Hefei, China, in 2004. Since 1992, he has been a faculty in the Department of Electronic Engineering and Information Science, USTC, where he is currently a professor. He is the executive director of the Department of Electronic Engineering and Information Science, USTC, and the director of the Information Processing Center, USTC. He has authored or co-authored more than 130 papers in journals and international conferences. His research interests include multimedia security, multimedia information retrieval, video processing, and information hiding.

Peilin Hong received the BS and MS degrees from the Department of Electronic Engineering and Information Science (EEIS), University of Science and Technology of China (USTC), in 1983 and 1986, respectively. Currently, she is a professor and advisor for PhD candidates in the Department of EEIS, USTC. Her research interests include next-generation Internet, policy control, IP QoS, and information security. She has published two books and more than 150 academic papers in several journals and conference proceedings.