SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

June 2021, Vol. 64 162301:1–162301:14
https://doi.org/10.1007/s11432-020-2957-5

Achieving adaptively secure data access control with privacy protection for lightweight IoT devices

Zhitao GUAN[1*], Wenti YANG[1], Liehuang ZHU[2], Longfei WU[3] & Ruimiao WANG[1]

[1]*School of Control and Computer Engineering, North China Electric Power University, Beijing* 102206*, China;*
[2]*School of Computer, Beijing Institute of Technology, Beijing* 100081*, China;*
[3]*Department of Mathematics and Computer Science, Fayetteville State University, Fayetteville* 28301*, USA*

Received 13 February 2020/Revised 14 April 2020/Accepted 5 June 2020/Published online 8 April 2021

Abstract The Internet of things (IoT) technology has been used in a wide range of fields, ranging from industrial manufacturing to daily lives. The IoT system contains numerous resource-constrained lightweight devices such as wireless sensors and radio frequency identification (RFID) tags. A massive amount of sensitive data is generated and transmitted by these devices to a variety of users. The complexity of the IoT system places a high demand on security. Therefore, it is necessary to develop an encryption scheme with access control to provide flexible and secure access to the sensitive data. The ciphertext policy attribute-based encryption (CP-ABE) scheme is a potential solution. However, the long ciphertext as well as the slow encryption and decryption operations in traditional ABE schemes make it inappropriate for most IoT systems, which require low latency and contain many devices with limited memory size and computing capability. In this paper, we propose a modified CP-ABE scheme with constant length of ciphertext and low computation overhead in the encryption and decryption phases. Additionally, our scheme is proven to be adaptively secure under the standard model. Moreover, two enhanced schemes are developed to prevent authorized users from leaking data and protect the privacy of data owners by combining chameleon hash, bloom filters and CP-ABE, respectively. Finally, the experimental evaluation and analysis prove the feasibility of our scheme.

Keywords IoT, CP-ABE, constant-size ciphertexts, adaptively secure, privacy protection

Citation Guan Z T, Yang W T, Zhu L H, et al. Achieving adaptively secure data access control with privacy protection for lightweight IoT devices. Sci China Inf Sci, 2021, 64(6): 162301, https://doi.org/10.1007/s11432-020-2957-5

1 Introduction

The Internet of things (IoT) plays a remarkable role in today's society, more and more fields have adopted the IoT technologies, such as healthcare, manufacturing, smart grid, and smart home [1]. The proliferation of IoT applications and services has brought great convenience to the society and individual lives. As a result, a massive amount of sensitive data is generated and transmitted in the IoT systems. To protect the sensitive data from malicious users, many efficient and secure asymmetric cryptographic algorithms have been proposed [2–4]. However, the one-to-one access mode between data and users in the traditional public key encryption schemes cannot satisfy the requirements of complex IoT systems. For example, a piece of sensitive data may need to be accessed by various smart devices or users from different organizations, which means that the data owners need to save the public keys of all data users to generate different ciphertexts for the respective users. It is inefficient and insecure for the complex IoT systems. CP-ABE (ciphertext policy attribute-based encryption), a fine-grained access control scheme, can support one-to-many access mode between data and multiple users which can solve the problems mentioned above.

Lots of studies have utilized the attribute-based encryption (ABE) in various IoT applications [5, 6]. However, in practice, the ciphertext length and the computation overhead of ABE will restrain its

*Corresponding author (email: guan@ncepu.edu.cn) application in IoT context. For example, in some cases, the ciphertext needs to be stored in IoT devices with limited memory size such as radio frequency identification (RFID) tags and quick response (QR) codes, but the length of the ciphertext in traditional ABE scheme grows with the complexity of the access structure. Additionally, the large computation overhead for encryption and decryption is also a challenge for the IoT devices with limited computing capability. Moreover, the complexity of the IoT system leads to a higher demand in security and privacy protection.

Some researchers have proposed ABE schemes with short ciphertexts [7–9] and some of them can achieve a lower computational overhead for decryption. However, these schemes are proven secure under the selective security model, in which attackers are required to declare the access structure to be attacked before the setup phase. It is weaker than in the adaptive security (full security) which does not require attackers to declare any information. Furthermore, many ABE schemes with full security [10, 11] are constructed based on the composite-order bilinear pairing, where the order is a big composite number which makes the scheme inefficient.

The motivations of our work are summarized as follows:

(1) Ensure the security of massive amounts of sensitive data and flexible and efficient access control in IoT.

(2) Reduce the computation costs of encryption and decryption of the access control scheme, and construct a short ciphertext length, to adapt to the applications in lightweight devices in IoT.

(3) Achieve higher security to meet the needs of complex IoT environment.

Thus, we propose a constant-ciphertext ABE scheme which can achieve the flexible and efficient access control under the type-III bilinear pairing. Our scheme is proven secure in the full security model and has a low encryption and decryption computation overhead. The main contributions are listed below.

(1) The ciphertext length in our scheme is short, constant, and independent of the AND gate access structure on which it is based. So, it can be stored in IoT devices with limited memory size such as RFID tags and QR codes.

(2) The encryption and decryption phases have a low computation overhead, which makes our scheme suitable for IoT systems with limited computing capability and low-latency requirement.

(3) Our scheme is proven to be adaptively secure under the standard model, and the algorithms are implemented under the type-III bilinear pairing which is better than type-I and type-II in terms of the computational performance, the length of the group element, and the flexibility. To better satisfy the security requirements, we introduce chameleon hash to prevent authorized users from leaking data and bloom filter to protect the privacy of data owners. They can be applied in different IoT scenarios according to the specific needs.

2 Related work

2.1 The application of ABE in IoT

Lots of studies have introduced ABE in IoT in order to manage data securely. Xu et al. [5] proposed an efficient, flexible ABE scheme with data verification and lightweight decryption in healthcare IoT system. Odelu et al. [12] designed lightweight CP-ABE scheme for battery-limited mobile devices in IoT based on the cloud computing. The proposed schemes mentioned above and many existing schemes such as [13, 14] focused more on the computation costs of ABE. However, our scheme focuses on improving the security of ABE while reducing the computation overhead. In [6], the authors presented a revocable ABE scheme which can achieve secret key revocation and resist the key exposure in IoT cloud. Guan et al. [15] proposed an efficient and secure data acquisition scheme based on ABE scheme which can achieve parallel data processing and protect the access structure information in smart grid. But this kind of works often ignored the efficiency of ABE in practical applications of IoT.

2.2 The short ciphertext and efficient ABE scheme

After Bethencourt et al. [16] gave the first concrete construction of CP-ABE with the tree access structure, there have been many works enhancing the CP-ABE scheme like [17–19]. It is a significant and challenging problem in ABE that the length of ciphertext and the computation overhead increase linearly with the access structure, making it inappropriate to be adopted in lightweight devices with limited memory size and computing capability. Jiang et al. [7] developed a short-ciphertext ABE scheme which supports both

AND gate and threshold. In [8], a CP-ABE with constant size secret keys is proposed by using elliptic curve cryptography. Susilo et al. [9] implemented an ABE scheme which has constant ciphertext length and has no extra dummy attributes. However, the references mentioned above only provide selective security.

The encryption and decryption efficiency of ABE is also one of the researches focuses. In [20], the ABE scheme requires only a few pairing operations in the decryption phase, and supports fast encryption and key generation. Teng et al. [21] proposed an attribute-based access control scheme in cloud computing which can achieve both constant ciphertext length and constant computation overhead of bilinear pairing. There are some similar studies like [22, 23].

2.3 The security and privacy protection of ABE

In the selective security model, the attacker is allowed to choose the challenge access structure before the setup phase. It is weaker than the adaptive security model. Many researchers have proposed the adaptively secure ABE scheme. The first adaptively secure CP-ABE was proposed by Lewko et al. [10], but it adopts the composite-order bilinear pairing. Doshi et al. [11] extended Ref. [10] and proposed an AND gate scheme with short ciphertext and fast decryption under the adaptive security model. Chen et al. [24] developed a adaptively secure ABE scheme with short ciphertexts and threshold access structures.

To prevent the data users from leaking the data, ABE schemes with support of accountability have been proposed. Xu et al. [25] proposed a revocable ABE scheme which can track the users who maliciously leak their keys. Liu et al. [26] presented an ABE scheme with traceability in which the access permissions given to the tracked malicious users will be revoked. There are also some researchers concerned about the disclosure of user privacy caused by public access policies. In [27], the authors applied ABE to smart healthcare. The privacy of patients contained in the access policy is protected by hiding the part of access policy with sensitive information. Han et al. [28] thought that user's attribute privacy may be leaked during both the key generation and the encryption phases, so they proposed an ABE scheme that hides both the attributes of users and the attributes of access policies.

3 Preliminaries

3.1 Bilinear maps

We set the group $\mathbb{BG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e)$ as a bilinear pairing group where $\mathbb{G}_T$ is a multiplicative group, $\mathbb{G}_1$ and $\mathbb{G}_2$ are elliptic groups. These three groups are of the same prime order p. Let g and h be the generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. We have the bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, for $a, b \in \mathbb{Z}_p$ and $u \in \mathbb{G}_1, v \in \mathbb{G}_2$:

(1) $e(u^a, v^b) = e(u, v)^{ab}$.

(2) The generator of $\mathbb{G}_T$ is $e(g, h)$.

(3) The $e(u, v)$ can be computed by an efficient algorithm.

In order to find an isomorphism from the elliptic curve group to the multiplicative easily, we should build the bilinear pairing with a pairing-friendly elliptic curve. The group $\mathbb{BG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e)$ has three types as follows:

(1) Type-I. Symmetric pairing, $\mathbb{G}_1 = \mathbb{G}_2$.

(2) Type-II. Asymmetric pairing 1, $\mathbb{G}_1 \neq \mathbb{G}_2$, and there exists an efficient homomorphism $\psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$.

(3) Type-III. Asymmetric pairing 2, $\mathbb{G}_1 \neq \mathbb{G}_2$, there is no efficient homomorphism between $\mathbb{G}_2$ and $\mathbb{G}_1$.

Moreover, symmetric bilinear pairing usually can only be built on a super singular elliptic curve, while the asymmetric bilinear pairing can be constructed from ordinary elliptic curve. Due to the MOV attacks on super singular elliptic curves [29], the asymmetric bilinear pairing is theoretically more secure. In addition, the type-III bilinear pairing is better than type-I and type-II in terms of the computational performance, the length of the group element, and the flexibility [30]. There is also a kind of bilinear pairing group constructed by groups of composite order. The order of the bilinear group is a big composite number, which makes the scheme inefficient. Additionally, for some elements in the group, there exists either no or more than one discrete logarithm, which can lead to some security issues.

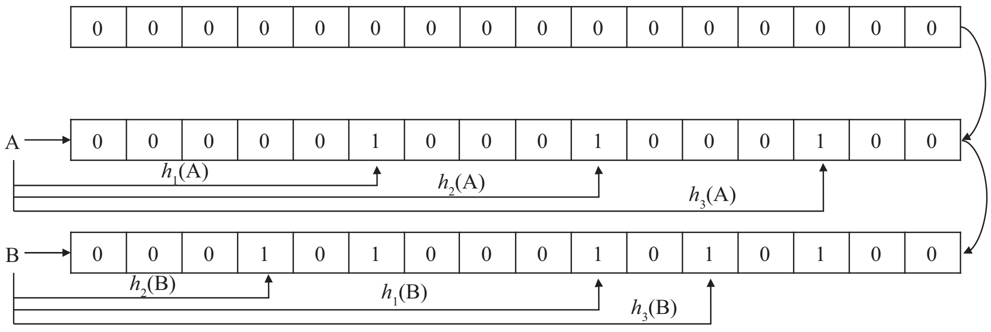

Figure 1 Bloom filter.

3.2 Bloom filter

A Bloom filter is consisted of an array BF with length L and some random mapping functions such as n hash functions $h_1, h_2, \ldots, h_n$, which are used to determine whether an element exists in the array BF. We give a simple example to show the implementation.

As shown in Figure 1, we first set an array in which each position is initialized to 0. Three hash functions h_1, h_2, h_3 are chosen to map elements into the array BF. Here we map A and B into BF. Suppose $h_1(\mathrm{A}) = 6$, $h_2(\mathrm{A}) = 10$, $h_3(\mathrm{A}) = 14$, $h_1(\mathrm{B}) = 10$, $h_2(\mathrm{B}) = 4$, $h_3(\mathrm{B}) = 12$, the values of the corresponding positions in BF are changed to 1 (or other non-zero numbers, based on needs). When we want to determine whether elements A, C, D exist in BF, for example, $h_1(\mathrm{D}) = 1$, $h_2(\mathrm{D}) = 9$, $h_3(\mathrm{D}) = 4$, obviously D is not in BF. However, when $h_1(\mathrm{A}) = 6$, $h_2(\mathrm{A}) = 10$, $h_3(\mathrm{A}) = 14$, it can be determined that A may exist in BF. Because when $h_1(\mathrm{C}) = 12$, $h_2(\mathrm{C}) = 6$, $h_3(\mathrm{C}) = 4$, even if C does not actually exist in BF, the corresponding positions are still not 0. This is the probability of false positives in Bloom filter. But when the length of BF is appropriate and the number of hash functions is enough, the probability of false positives can be ignored.

3.3 Chameleon hash

Chameleon hash function is also known as trapdoor hash function, which is a special collision-resistant hash function with a key pair $(\mathrm{pk_{ch}}, \mathrm{sk_{ch}})$. $\mathrm{pk_{ch}}$ is used to compute the hash value. Users who has $\mathrm{sk_{ch}}$ can efficiently find the collisions of the corresponding hash function, but it is hard for those who do not know $\mathrm{sk_{ch}}$ to find a collision, that is to say, a chameleon hash function is collision-resistant for those who do not have the trapdoor key $\mathrm{sk_{ch}}$. The algorithms are as follows:

(1) $\mathrm{CH.KeyGen}(1^k) \rightarrow (\mathrm{pk_{ch}}, \mathrm{sk_{ch}})$: It takes security parameter k as input, and outputs a key pair $(\mathrm{pk_{ch}}, \mathrm{sk_{ch}})$.

(2) $\mathrm{CH.Hash}(\mathrm{pk_{ch}}, m, z) \rightarrow H_{\mathrm{ch}}$: It takes the public key $\mathrm{pk_{ch}}$, a message m, and a random parameter z as inputs, and outputs the hash value H_{ch}.

(3) $\mathrm{CH.Trap}(\mathrm{sk_{ch}}, m, m', z) \rightarrow z'$: It takes the trapdoor $\mathrm{sk_{ch}}$, m, z, and a collision m as inputs, and computes a z such that $\mathrm{CH.Hash}(\mathrm{pk_{ch}}, m, z) = \mathrm{CH.Hash}(\mathrm{pk_{ch}}, m', z')$.

3.4 Complexity assumptions

Definition 1 (Multi-sequence of exponents Diffie-Hellman assumption (MSEDH)). $\mathbb{BG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e)$ is a bilinear mapping group. Let g_0 and h_0 be the generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. Two random polynomials f and g are of the orders l and m. l, m, t are integers. We have $x_1, \ldots, x_l \in \mathbb{Z}_p^*$, $y_1, \ldots, y_m \in \mathbb{Z}_p^*$, $f(X) = \prod_{i=1}^{l}(X + x_i)$, $g(Y) = \prod_{i=1}^{m}(Y + y_i)$. The other inputs are listed as follows:

$$D = \left\{ \begin{matrix} g_0 \; g_0^{\alpha} \; \cdots \; g_0^{\alpha^{l+t-2}} & g_0^{\omega} \; g_0^{\omega\cdot\alpha} \; \cdots & g_0^{\omega\cdot\alpha^{l+t}} & g_0^{r\cdot\alpha\cdot f(\alpha)} \\ h_0 \; h_0^{\alpha} \; \ldots \; h_0^{\alpha^{m-2}} & h_0^{\omega} \; h_0^{\omega\cdot\alpha} \; \cdots & h_0^{\omega\cdot\alpha^{2m-2}} & h_0^{r\cdot g(\alpha)} \end{matrix} \right\}, \tag{1}$$

where α, ω, r are random elements of $\mathbb{Z}_p^*$. The element $T \in \mathbb{G}_T$. The MSEDH problem is to decide whether $T = e(g_0, h_0)^{rf(\alpha)}$ or $T \in_R \mathbb{G}_T$.

The MSEDH assumption holds in the asymmetric pairing group if there is no probabilistic polynomial-time (PPT) adversaries $\mathcal{A}$ who can solve the MSEDH with non-negligible advantage. The advantage of $\mathcal{A}$

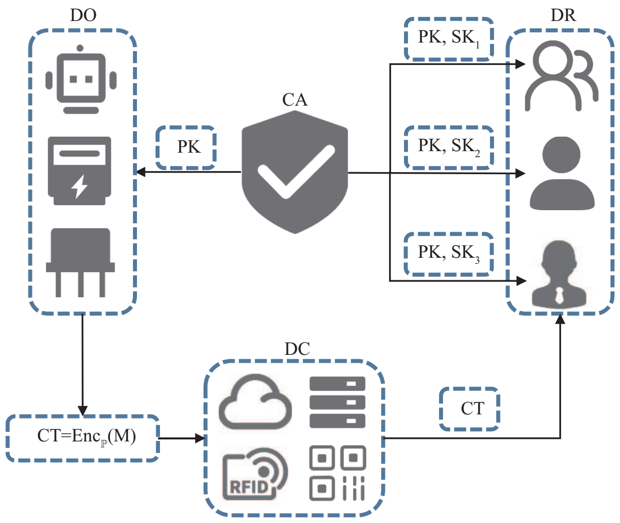

Figure 2 (Color online) System model.

is defined as $\mathrm{Adv}^{\mathcal{A}}_{\mathrm{MSEDH}}(\lambda) = |\Pr[\mathcal{A}(f, g, \mathbb{BG}, D, T_0) = 1] - \Pr[\mathcal{A}(f, g, \mathbb{BG}, D, T_1) = 1]|$, where we denote that $T_0 = e(g_0, h_0)^{rf(\alpha)}$, $T_1 \in \mathbb{G}_T$ is a random element.

Definition 2 (Computational Diffie-Hellman problem (CDH))**.** Let $\mathbb{G}$ be a cyclic group of prime order p with generator g. Given a tuple $\langle g, g^a, g^b \in \mathbb{G}\rangle$, where $a, b \in_R \mathbb{Z}_p$. The CDH problem is to compute g^{ab}. The CDH problem holds in the cyclic group if there is no PPT adversaries $\mathcal{A}$ who can compute g^{ab} with non-negligible advantage. The advantage of $\mathcal{A}$ is defined as $\mathrm{Adv}^{\mathcal{A}}_{\mathrm{CDH}}$.

4 System and security model

4.1 Overview

We give an overview of the system. As shown in Figure 2, four entities are involved: certificate authority (CA), data owner (DO), data requester (DR), and data center (DC). Note that in the following content, the notation $x \in_R X$ means that x is a randomly selected element from the set X. The detailed description is as follows.

(1) CA is a fully trusted party which is responsible for generating the public key (PK) and the master key (MK). It is also in charge of the secret key (SK) generation according to the set of attributes S of DR.

(2) DO generates the sensitive data that needs to be encrypted under an access structure. Firstly, DO develops an access structure $\mathbb{P} = \{P_1, P_2, \ldots, P_j\}$ (for $1 \leqslant i \leqslant j$, $P_j \in \mathbb{U}$, $\mathbb{U}$ is the universe of attributes), that determines the users who can access the data. Then it encrypts the sensitive data with $\mathbb{P}$ and PK, then gains the ciphertext CT (denoted by $\mathrm{Enc}_{\mathbb{P}}(M)$).

(3) DR submits its attributes S to CA in the key generation phase, and gains the corresponding SK. In the decryption phase, DR decrypts CT with SK. Only if the attributes involved in SK satisfy the access structure $\mathbb{P}$ (that is for all $i = 1, 2, \ldots, j$, $P_i \in S$), can CT be successfully decrypted.

(4) DC provides data storage services. CT is stored in DC which can be a cloud, database or an IoT device (such as RFID tags, QR codes). If DC is a third party, it cannot get any sensitive information from CT.

4.2 Design goal

For security goals, we consider CA a fully trusted party, but DR and DC may be honest-but-curious. Hence, the following security requirements need to be implemented.

(1) Data confidentiality. The ciphertext is stored in the DC or in an IoT device. Therefore, DC and the semi-trusted users can easily get the ciphertext. It must be guaranteed that no sensitive information will be obtained by unauthorized users.

(2) Collusion-resistance. DRs who are unauthorized to access the data may collude with each other. They may combine their secret keys to produce a key that can satisfy the access structure of the data. To prevent such collusion attack, the secret keys assigned to different DRs are generated with different random parameters.

(3) Information leakage. Generally, DRs who are authorized to access the data are considered trustful and will not leak the data to other unauthorized users. But in practice, it needs to be taken into consideration that malicious DRs may leak the sensitive data. Some researchers have proposed traceable ABE scheme to catch DRs leaking the data. A new method will be given in our enhanced scheme.

For performance goals, to better work with the large number of lightweight devices and satisfy the low latency requirement in IoT, the following performance requirements need to be implemented.

(1) Efficient encryption and decryption. There are many devices with limited computing capability in IoT systems. The heavy computation costs of encryption and decryption in ABE would become a critical barrier for practical applications. Hence, encryption and decryption methods with low computational overhead is highly desirable to satisfy the low-latency requirement of IoT.

(2) Short ciphertext length. In traditional ABE schemes, the length of ciphertext grows with the complexity of the access structure. However, the ciphertext often needs to be stored in an IoT device which have limited memory size (RFID tags and QR codes usually can only store a few thousands of bits). Hence, a new ABE scheme with short and constant ciphertext length is needed.

4.3 Algorithm description

The algorithms used in the scheme are described as follows.

(1) The basic scheme:

• $\mathrm{Setup}(1^\lambda) \rightarrow (\mathrm{PK}, \mathrm{MK})$. It takes a security parameter λ as input, then outputs the PK and MK.

• $\mathrm{KeyGen}(S, \mathrm{PK}, \mathrm{MK}) \rightarrow \mathrm{SK}$. It takes a set of attributes S, PK and MK as inputs, then outputs the secret key SK.

• $\mathrm{Encryption}(M, \mathrm{PK}, \mathbb{P}) \rightarrow \mathrm{CT}$. It takes the message M, PK and the access structure $\mathbb{P}$ as inputs, then outputs the ciphertext CT.

• $\mathrm{Decryption}(\mathrm{PK}, \mathrm{SK}, \mathrm{CT}) \rightarrow \mathrm{Mor}\perp$. It takes PK, SK and CT as inputs, then outputs the corresponding message M or returns $\perp$.

(2) The signature scheme:

• $\mathrm{SigGen}(1^k) \rightarrow (\mathrm{pk}, \mathrm{sk})$. It takes a security parameter k as input, then outputs the key pair $(\mathrm{pk}, \mathrm{sk})$.

• $\mathrm{Sign}(\mathrm{sk}, C) \rightarrow \mathrm{SIGN}$. It takes sk and a message C as inputs, then outputs the signature SIGN.

• $\mathrm{Verify}(\mathrm{pk}, C, \mathrm{SIGN}) \rightarrow$ True or False. It takes pk, the message C and the signature SIGN as inputs, if the signature is valid, it outputs True, else it outputs False.

4.4 Security model (indistinguishable under chosen plaintext attack (IND-CPA) model)

We denote our scheme by Π. Let $\mathcal{A}$ be the adversary to attack the scheme Π and $\mathcal{C}$ be the challenger. The game is described as follows.

Setup. $\mathcal{C}$ runs this algorithm to generate the PK and MK. It gives PK to $\mathcal{A}$.

Phase1. $\mathcal{A}$ adaptively sends a set of attributes S. $\mathcal{C}$ generates the corresponding SK which is returned to $\mathcal{A}$.

Challenge. $\mathcal{A}$ submits two messages M_0, M_1 and an access structure $\mathbb{P}^*$ to $\mathcal{C}$. Note that for every S queried by $\mathcal{A}$, S cannot satisfy $\mathbb{P}^*$. $\mathcal{C}$ flips a coin $b \in 0, 1$ and encrypts M_b with the access structure $\mathbb{P}^*$ to obtain CT^*. $\mathcal{C}$ sends the ciphertext CT^* to $\mathcal{A}$.

Phase2. Repeat Phase 1. For every S queried by $\mathcal{A}$, S cannot satisfy the access structure $\mathbb{P}^*$.

Guess. $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$ for b and wins the game if $b' = b$.

The advantage of $\mathcal{A}$ in this game is defined as $\mathrm{Adv}(\mathcal{A}) = |\Pr[b' = b] - 1/2|$.

Definition 3. The scheme Π is CPA security in adaptive security model if no PPT adversary has a non-negligible advantage in the above game.

Note that we can add an initialization phase before setup to construct a selective security model, in which $\mathcal{A}$ should declare the challenge access structure.

5 Basic scheme

5.1 The construction

The detailed description of the construction of the basic scheme is as follows.

(1) Setup. CA runs this algorithm. It takes as inputs a security parameter λ and an universe of attributes $\mathbb{U} = \{U_1, U_2, \ldots, U_n\}$, the positive integer n represents the number of attributes. First, it chooses a pairing group $\mathbb{BG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e)$, and the generators g and h. Then it chooses $\alpha \in_R \mathbb{Z}_p$ and for all $i = 1, 2, \ldots, n$, it computes $v = g^{\alpha}$, $h_i = h^{\alpha^i}$.

The PK and MK are set as: $\mathrm{PK} = \{\mathbb{BG}, h, v, h_i, e(g, h)\}$, $\mathrm{MK} = \{\alpha, g\}$.

(2) KeyGen. CA runs this algorithm. It takes as inputs a set of attributes S, PK and MK. It chooses $s \in_R \mathbb{Z}_p^*$ and computes a vector $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)$ as: for $i = 1, 2, \ldots, n$, if $U_i \in S$, $y_i = 1$; if $U_i \notin S$, $y_i = 0$.

Set $t = \sum_{i=1}^n \alpha^{iy_i} + n$ and $t' = \sum_{i=1}^n \alpha^{iy_i+1} + n\alpha$, it then computes $\mathrm{SK}_1 = g^{s/t}$, $\mathrm{SK}_2 = h^{(s-1)/\alpha}$. We set $y_0 = 1$, for $i = 0, 1, 2, \ldots, n$, if $y_i = 1$, it computes $\mathrm{SK}_{3,i} = h^{s\alpha^i/t'}$.

It generates the secret key for the attributes set S as $\mathrm{SK} = \{\boldsymbol{y}, \mathrm{SK}_1, \mathrm{SK}_2, \mathrm{SK}_{3,i}\}$.

(3) Encryption. DO runs this algorithm. It takes as inputs the message M, PK, and an access structure $\mathbb{P} = \{P_1, P_2, \ldots, P_j\}$, $1 \leqslant j \leqslant n$, for $1 \leqslant i \leqslant j$, $P_i \in \mathbb{U}$. It chooses $r \in_R \mathbb{Z}_p^*$ and computes a vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ as: for $i = 1, 2, \ldots, n$, if $U_i \in \mathbb{P}$, $x_i = 1$; if $U_i \notin \mathbb{P}$, $x_i = 0$.

Then it computes $C_1 = \mathrm{M}e(g, h)^r$, $C_2 = v^{-r}$, $C_3 = (\prod_{i=1}^n h^{\alpha^{ix_i}})^r$. Note that when $x_i = 0$, it has $h^{\alpha^{ix_i}} = h$, and when $x_i = 1$, $h^{\alpha^{ix_i}} = h^{\alpha^i} = h_i$.

The ciphertext is set as $\mathrm{CT} = \{\mathbb{P}, \boldsymbol{x}, C_1, C_2, C_3\}$.

(4) Decryption. DR runs this algorithm. It takes as inputs the PK, SK, and CT. If $\boldsymbol{x} \cdot \boldsymbol{y} \neq j$, then return $\perp$. Otherwise, it first sets $\varepsilon = \sum_{i=1}^n \alpha^{ix_i}$, $\varepsilon' = \sum_{i=1}^n \alpha^{i\cdot(y_i - x_i)}$ and then computes

$$\begin{aligned} &A = e(\mathrm{SK}_1, C_3) = e(g,h)^{sr\frac{\varepsilon}{t}}, \\ &B = e\left(C_2^{-1}, \prod_{i=1}^n h^{s\alpha^{i\cdot(y_i-x_i)}/t'}\right) = e(g,h)^{sr\frac{\varepsilon'}{t}}, \\ &C = e(C_2, \mathrm{SK}_2) = e(g,h)^{-sr+r}, \end{aligned} \tag{2}$$

if the corresponding attributes satisfy the access structure $\mathbb{P}$, we have $e(g,h)^r = A \cdot B \cdot C$.

Finally, it will recover M by computing:

$$M = \frac{C_1}{A \cdot B \cdot C} = \frac{Me(g,h)^r}{e(g,h)^r}.$$

5.2 Correctness

When a user's attributes satisfy the access structure $\mathbb{P}$, for $i = 1, 2, \ldots, n$, if $x_i = 1$, there must be $y_i = 1$. We have $\boldsymbol{x} \cdot \boldsymbol{y} = j$, where j is the number of attributes in $\mathbb{P}$, that is, the number of elements in $\boldsymbol{x}$ when $x_i = 1$. The correctness analysis of the decryption algorithm is as follows:

$$\begin{aligned} A = e(\mathrm{SK}_1, C_3) &= e\left(g^{s/(\sum_{i=1}^n \alpha^{i\cdot y_i}+n)}, \left(\prod_{i=1}^n h^{\alpha^{ix_i}}\right)^r\right) \\ &= e\left(g^{s/(\sum_{i=1}^n \alpha^{i\cdot y_i}+n)}, h^{r\sum_{i=1}^n \alpha^{i\cdot x_i}}\right) = e(g,h)^{sr\cdot\sum_{i=1}^n \alpha^{i\cdot x_i}/(\sum_{i=1}^n \alpha^{i\cdot y_i}+n)}, \end{aligned} \tag{3}$$

$$\begin{aligned} B = e\left(C_2^{-1}, \prod_{i=1}^n h^{s\alpha^{i\cdot(y_i-x_i)}/t'}\right) &= e\left(g^{r\alpha}, h^{s\cdot\sum_{i=1}^n \alpha^{i\cdot(y_i-x_i)}/(\sum_{i=1}^n \alpha^{iy_i+1}+n\alpha)}\right) \\ &= e\left(g,h\right)^{rs\alpha\cdot\sum_{i=1}^n \alpha^{i\cdot(y_i-x_i)}/(\sum_{i=1}^n \alpha^{iy_i+1}+n\alpha)} = e\left(g,h\right)^{rs\cdot\sum_{i=1}^n \alpha^{i\cdot(y_i-x_i)}/(\sum_{i=1}^n \alpha^{iy_i}+n)}, \end{aligned} \tag{4}$$

$$C = e(C_2, \mathrm{SK}_2) = e(g^{-r\alpha}, h^{(s-1)/\alpha}) = e(g,h)^{-sr+r}, \tag{5}$$

$$A \cdot B = e(g,h)^{sr}. \tag{6}$$

When the set of attributes S satisfies the access structure $\mathbb{P}$, $y_i - x_i = 0$ or 1, the values $h^{s\alpha^{i\cdot(y_i-x_i)/t'}}$ in (4) is computed from the set $\mathrm{SK}_{3,i} = h^{s\alpha^i/t'}$. It is easy to find that

$$\left(\sum_{i=1}^{n} \alpha^{i\cdot x_i} + \sum_{i=1}^{n} \alpha^{i(y_i-x_i)}\right) \bigg/ \left(\sum_{i=1}^{n} \alpha^{i\cdot y_i} + n\right) = 1.$$

Then we can get $e(g,h)^r = A\cdot B\cdot C$.

If the attributes S do not satisfy the access structure $\mathbb{P}$, then it has $\exists i \in [1,n]$, $y_i - x_i = -1$, $y_i = 0$ the value $h^{s\alpha^i/t'}$ cannot be computed. And there must be $(\sum_{i=1}^{n} \alpha^{i\cdot x_i} + \sum_{i=1}^{n} \alpha^{i(y_i-x_i)})/(\sum_{i=1}^{n} \alpha^{i\cdot y_i} + n) \neq 1$. It cannot calculate any information about the plaintext.

5.3 Security analysis

Theorem 1. Our basic scheme is CPA security under adaptive security model if the MSEDH problem is hard.

Proof. Suppose that there is an adversary $\mathcal{A}$ who can attack the scheme Π with non-negligible advantage. We can build a simulator $\mathcal{B}$ which can solve the MSEDH problem with a non-negligible advantage. Let the input of the MSEDH problem instance be the input of $\mathcal{B}$. $\mathcal{B}$ interacts with $\mathcal{A}$ as follows.

Setup. $\mathcal{B}$ first specifies a universe of attributes $\mathbb{U}^* = \{U_1, U_2, \ldots, U_n\}$. Then it defines $h = h_0$, $g = g_0^{f(a)}$ which is computed from $h_0, h_0^{\alpha}, \ldots, h_0^{\alpha^{m-2}}$ and $g_0, g_0^{\alpha}, \ldots, g_0^{\alpha^{l+t-2}}$, note that f is a polynomial of degree l. The other components of public key are computed as follows:

(1) $v = g^{\alpha} = g_0^{\alpha f(\alpha)}$ is computed from $g_0, g_0^{\alpha}, \ldots, g_0^{\alpha^{l+t-2}}$, note that $\alpha f(\alpha)$ is a polynomial of degree $l+1$, the coefficients of $\alpha f(\alpha)$ are known from f.

(2) $h_i = h^{\alpha^i} = h_0^{\alpha^i}$ $(i = 1, \ldots, n)$, we set $n \leqslant m-2$, so it can be computed from $h_0, h_0^{\alpha}, \ldots, h_0^{\alpha^{m-2}}$.

(3) $e(g,h) = e(g_0^{f(\alpha)}, h_0)$, it can be easily computed by h_0 and $g_0^{f(\alpha)}$.

Eventually, $\mathcal{B}$ gives to $\mathcal{A}$ the resulting

$$\mathrm{PK} = \left\{\mathbb{BG}, h_0, v = g_0^{\alpha f(\alpha)}, h_i = h_0^{\alpha^i}, e(g,h) = e\left(g_0^{f(\alpha)}, h_0\right)\right\}. \tag{7}$$

Phase 1. $\mathcal{A}$ submits a set of attributes S^* to $\mathcal{B}$ to query a decryption key, $\mathcal{B}$ sets a vector $\boldsymbol{y}^* = (y_1, y_2, \ldots, y_n)$ as the KeyGen algorithm.

To simulate the secret key $\mathrm{SK} = \{\boldsymbol{y}, \mathrm{SK}_1, \mathrm{SK}_2, \mathrm{SK}_{3,i}\}$, we first define $s = (\sum_{i=1}^{n} \alpha^{i\cdot y_i} + n)\cdot(\varepsilon\alpha + n^{-1})$, where $\varepsilon \in_R \mathbb{Z}_p^*$. Thus, s is uniformly distributed in $\mathbb{Z}_p^*$ in the game. The components of SK are computed as follows:

(1) $\mathrm{SK}_1^* = g^{s/(\sum_{i=1}^{n} \alpha^{iy_i}+n)} = g_0^{f(\alpha)\cdot(\varepsilon\alpha+n^{-1})} = g_0^{\varepsilon\alpha f(\alpha)+n^{-1}f(\alpha)}$. We have $f(\alpha)$ and $\alpha f(\alpha)$ which are polynomials of degree l and $l+1$, respectively. SK_1^* can be computed from $g_0, g_0^{\alpha}, \ldots, g_0^{\alpha^{l+t-2}}$.

(2) $\mathrm{SK}_2^* = h^{(s-1)/\alpha} = h_0^{(\sum_{i=1}^{n} \alpha^{i\cdot y_i}+n)\cdot(\varepsilon\alpha+n^{-1})-1/\alpha} = h_0^{\varepsilon\cdot\sum_{i=1}^{n} \alpha^{i\cdot y_i}+n^{-1}\cdot\sum_{i=1}^{n} \alpha^{i\cdot y_i-1}+\varepsilon n}$. It can be computed from $h_0, h_0^{\alpha}, \ldots, h_0^{\alpha^{m-2}}$.

(3) $\mathrm{SK}_{3,i}^* = h^{s\alpha^i/(\sum_{i=1}^{n} \alpha^{iy_i+1}+n\alpha)} = h_0^{\alpha^{i-1}\cdot(\varepsilon\alpha+n^{-1})} = h_0^{\varepsilon\alpha^i+n^{-1}\cdot\alpha^{i-1}}$.

$\mathcal{B}$ sends $\mathrm{SK}^* = \{\boldsymbol{y}^*, \mathrm{SK}_1^*, \mathrm{SK}_2^*, SK_{3,i}^*\}$ to $\mathcal{A}$.

Challenge. $\mathcal{A}$ submits two messages M_0, M_1 which have the equal length and submits an access structure $\mathbb{P}^*$ to $\mathcal{B}$. Note that for every S^* queried by $\mathcal{A}$, S^* cannot satisfy the access structure $\mathbb{P}^*$, $\mathcal{B}$ flips a coin $b' \in \{0,1\}$ and encrypts M_b with the access structure $\mathbb{P}^*$ to obtain the corresponding $\mathbb{CT}^*$, $\mathcal{B}$ first sets a vector $\boldsymbol{x}^* = (x_1, x_2, \ldots, x_n)$ as the Encryption algorithm. The components of ciphertext are computed as follows:

(1) $C_1^* = M_b T$; (2) $C_2^* = v^{-r} = g_0^{-r\alpha f(\alpha)}$; (3) $C_3 = (\prod_{i=1}^{n} h^{\alpha^{ix_i}})^r = h_0^{r\cdot\sum_{i=1}^{n} \alpha^{i\cdot x_i}}$.

$\mathcal{B}$ sends the ciphertext $\mathrm{CT}^* = (\mathbb{P}^*, \boldsymbol{x}^*, C_1^*, C_2^*, C_3^*)$ to $\mathcal{A}$.

Phase 2. This phase is the same as Phase 1.

Guess. $\mathcal{A}$ outputs its guess $b' \in \{0,1\}$ for b, if $b' = b$, $\mathcal{B}$ outputs $T = T_0 = e(g_0, h_0)^{rf(\alpha)}$ as its guess to the MSEDH problem. Otherwise, $\mathcal{B}$ outputs $T = T_1$ in which T_1 is a random element in $\mathbb{G}_T$.

If $\mathcal{A}$ can break the scheme Π with non-negligible advantage δ, we have

$$\begin{aligned}\mathrm{Adv}_{\mathrm{MSEDH}}^{\mathcal{B}}(\lambda) &= \left|\Pr\left[\mathcal{A}\left(f, g, \mathbb{BG}, D, T_0\right) = 1\right] - \Pr\left[\mathcal{A}\left(f, g, \mathbb{BG}, D, T_1\right) = 1\right]\right| \\ &= \left|\Pr\left[b = b' | T = T_0\right] - \Pr\left[b = b' | T = T_1\right]\right| = \delta.\end{aligned} \tag{8}$$

Thus, $\mathcal{B}$ can solve the MSEDH problem with the advantage δ.

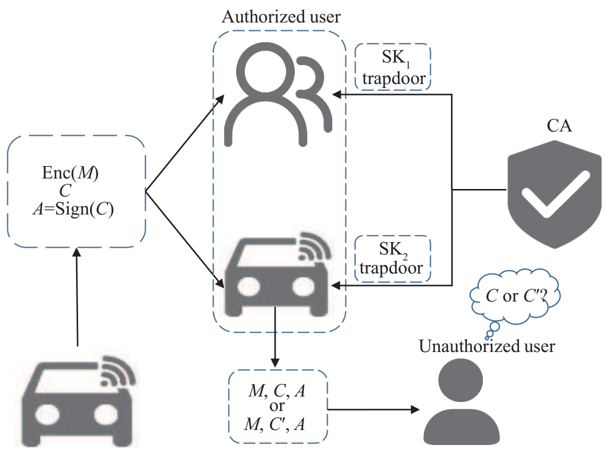

Figure 3 (Color online) An enhanced scheme to prevent information leakage.

6 Enhanced scheme

Considering of practical cases, we further develop two enhanced schemes to make our scheme more suitable for specific scenarios in the IoT system.

6.1 Prevent information leakage

Generally, DRs which are authorized to access the data are considered trustful and will not leak the data to unauthorized users. But in practice, it needs to be taken into consideration that malicious DRs may leak the sensitive data. We take the Internet of vehicles (IoV) as an example, as shown in Figure 3, most sensitive information (denoted by M) is timely generated and associated with iconic information (denoted by C) such as geography and time. Sensitive data without C is considered no longer valuable. But due to the low latency requirement of IoV, C should be published on the network in plaintext with encrypted M (denoted by $\mathrm{Enc}(M)$) instead of encrypted as ciphertext, so that the data requesters can quickly find the corresponding information.

Data requesters whose attributes satisfy the access structure of the $\mathrm{Enc}(M)$ can decrypt $\mathrm{Enc}(M)$ to get the sensitive information M. But the data requesters such as vehicles in the nearby area or 3rd-party organizations may not be fully trusted, they could leak the sensitive information M to unauthorized users without the access permission. We introduce the chameleon signature to address this problem. The information published on the network includes $\mathrm{Enc}(M)$, C, and the chameleon signature of C (denoted by $\mathrm{Sign}(C)$).

The chameleon signature in our scheme is used to:

(1) Verify whether the information C has been tampered by the malicious users during transmission. After receiving the information C, DR can verify whether C has been tampered by using the signature $\mathrm{Sign}(C)$. The details of the verification are shown in the decryption phase.

(2) Prevent the authorized users from leaking the information. The authorized users who have a trapdoor of the chameleon hash function can forge a C' with $\mathrm{Sign}(C') = \mathrm{Sign}(C)$ so that unauthorized users cannot distinguish whether the leaked information C was constructed by the authorized users or the data owner. When an authorized user leaks the information, he/she has no evidence to prove that the leaked information came from the data owner instead of constructed by himself/herself using trapdoor.

The details of the algorithm are as follows. The unexplained parameters have been defined in the basic scheme.

(1) Setup. Input: λ, $\mathbb{U} = \{U_1, U_2, \ldots, U_n\}$. Parameters: $\mathbb{BG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e)$, $g, h, \alpha \in_R \mathbb{Z}_p$, a hash function $H_1 : \{0,1\}^* \to \mathbb{G}_2$. Output: $\mathrm{PK} = \{\mathbb{BG}, h, v = g^{\alpha}, h_i = h^{\alpha^i}, e(g,h), H_1\}$, $\mathrm{MK} = \{\alpha, g\}$.

(2) Chameleon signature KeyGen. Input: λ. Parameters: $\beta_u, \gamma \in_R \mathbb{Z}_p$. Output: for each registered user, a public/secret key pair ($\mathrm{pk}_u = g^{\beta}$, $\mathrm{sk}_u = \beta$). For some specific areas, a trapdoor key pair ($\mathrm{pk}_{\mathrm{ch}} = g^{\gamma}$, $\mathrm{sk}_{\mathrm{ch}} = \gamma$).

(3) Key generation. Same as basic scheme.

(4) Encryption. Input: M, PK, $\mathbb{P} = \{P_1, P_2, \ldots, P_j\}$, C, $\mathrm{pk}_{\mathrm{ch}}$, sk_u. The encryption of M is the same as basic scheme. Signature: it chooses $z \in_R \mathbb{Z}_p$, $H_{\mathrm{ch}}(C) = g^C \mathrm{pk}_{\mathrm{ch}}^z$, $\mathrm{Sign} = H_1(H_{\mathrm{ch}}(C))^{\beta}$,

$\text{SIGN} = (\text{Sign}, C, z)$. Output: $\text{CT} = \{\mathbb{P}, \boldsymbol{x}, C_1, C_2, C_3, \text{SIGN}\}$.

(5) Decryption. Input: PK, SK, CT, pk_u. Verification: compute $H_{\text{ch}}(C) = g^C \text{pk}_{\text{ch}}^z$, if $e(H_1(H_{\text{ch}}(C)), \text{pk}_u) = e(\text{Sign}, g)$, then decrypt as basic scheme do. Otherwise, it returns $\perp$.

(6) Forgery of signature. Data requester owns the trapdoor $\text{sk}_{\text{ch}} = \gamma$, so it can forge a C' which has $H_{\text{ch}}(C) = g^C \text{pk}_{\text{ch}}^z = g^{C'} \text{pk}_{\text{ch}}^{z'}$, where $z' = (C + z\gamma - C')/\gamma$. Thus, when the data receiver attempts to leak this message, other users cannot distinguish whether the signature was constructed by the data requester or the data owner. The information C may have been tempered by the data requester. It prevents the leakage of user privacy to some degree.

6.2 Privacy protection

Sometimes a malicious user can infer private information about the data owner based on the corresponding access structure. For example, the access structure of patient A is {hospital, oncology, doctor}, then others can guess that A may have a tumor-related disease according to this access structure. Therefore, we need to hide the important information in the access structure. Our scheme uses Bloom filter to achieve privacy protection. The details of the algorithm are as follows.

(1) Setup. Input: λ, $\mathbb{U} = \{U_1, U_2, \ldots, U_n\}$. Parameters: $\mathbb{BG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e)$, g, h, $\alpha \in_R \mathbb{Z}_p$, a hash function $H_2 : \{0,1\}^* \to \mathbb{Z}_p$, $h_1, h_2, h_3 : \{0,1\}^* \to \mathbb{Z}_p$. Output: $\text{PK} = \{\mathbb{BG}, h, v = g^\alpha, h_i = h^{\alpha^i}, e(g,h), H_2, h_1, h_2, h_3\}$, $\text{MK} = \{\alpha, g\}$.

(2) Key generation. Same as basic scheme.

(3) Encryption. Same as basic scheme.

(4) Hide access structure. Input: CT.

For each attribute P_i in the access structure $\mathbb{P} = \{P_1, P_2, \ldots, P_j\}$, it first computes $a_i = h_1(P_i) \bmod L$, $b_i = h_2(P_i) \bmod L$, $c_i = h_3(P_i) \bmod L$, L is the length of the Bloom filter array BF. If $\text{BF}[a_i] = 0$, $\text{BF}[b_i] = 0$, $\text{BF}[c_i] = 0$, then it computes $\text{BF}[a_i] = r_{a,i}$, $\text{BF}[b_i] = \text{BF}[a_i] \oplus r_{b,i}$, $\text{BF}[c_i] = \text{BF}[a_i] \oplus \text{BF}[b_i] \oplus H_2(P_i)$, $r_{a,i}, r_{b,i} \in_R \mathbb{Z}_p$. If one of $\text{BF}[a_i]$, $\text{BF}[b_i]$, $\text{BF}[c_i]$ is not equal to 0, then the attribute P_i is the public part, and add it to the set $\mathbb{P}'$ which is empty initially (this rarely happens when the array BF is long enough).

For convenience, we choose three hash functions here to map each attribute to three locations in BF. In fact, the number of hash functions can be dynamically adjusted according to the actual conditions and needs.

Output: $\text{CT}' = \{\text{BF}, \mathbb{P}', C_1, C_2, C_3\}$.

(5) Decryption. Input: PK, SK, CT'.

For each attribute S_i that does not exist in $\mathbb{P}'$, it computes $a_i' = h_1(S_i) \bmod L$, $b_i' = h_2(S_i) \bmod L$, $c_i' = h_3(S_i) \bmod L$. If $\text{BF}[a_i']$, $\text{BF}[b_i']$, $\text{BF}[c_i']$ are not equal to 0, compute $H_2'(S_i) = \text{BF}[a_i'] \oplus \text{BF}[b_i'] \oplus \text{BF}[c_i']$. If $H_2(S_i) = H_2'(S_i)$, then it can be confirmed that S_i must exist in the access structure. Otherwise, even if all the corresponding positions in the array are not 0, S_i does not exist in the access structure. If any $\text{BF}[x_i] = 0, (x \in \{a, b, c\})$, then S_i does not exist in the access structure.

If the number of attributes in S which belong to the access structure is equal to the number of attributes of the access structure, then continue to decrypt in the same way as the basic scheme. Otherwise, it returns $\perp$.

6.3 Security analysis

The security analysis of the enhanced scheme is as follows.

(1) The security of the chameleon signature.

Theorem 2. If the CDH problem is hard, the chameleon signature is unforgeable in random oracle.

Proof. Let $\mathcal{A}$ be the adversary to attack the chameleon signature scheme with non-negligible advantage. We can build a simulator $\mathcal{B}$ which can solve the CDH problem with a non-negligible advantage. Let the input of the CDH problem instance $\langle g, g^a, g^b \rangle$ be the input of $\mathcal{B}$. $\mathcal{B}$ interacts with $\mathcal{A}$ as follows.

Setup. Let $\mathbb{BG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e)$ be the system parameter. $\mathcal{B}$ runs this algorithm to generate the key pair $(\text{pk}_u = g^\beta, \text{sk}_u = \beta)$ where $\beta = \alpha$, and chooses $\gamma, z \in_R \mathbb{Z}_p$ to generate a trapdoor key pair $(\text{pk}_{\text{ch}} = g^\gamma, \text{sk}_{\text{ch}} = \gamma)$. Then, it gives pk_u to $\mathcal{A}$.

H-Query. $\mathcal{A}$ issues hash queries to $\mathcal{B}$. $\mathcal{B}$ creates an empty table T and randomly chooses $i^* \in [1, Q_H]$ where Q_H is the number of H-Queries. Each time $\mathcal{A}$ queries $\mathcal{B}$ for the hash value of C_i, $\mathcal{B}$ first computes

Table 1 The description of symbols

Symbol	Definition
m	Number of attributes that generate the SK
j	Number of attributes included in the access structure
n	Number of attributes in the universe
$\|\mathbb{G}\|$	Length of $\mathbb{G}$ in type-I bilinear pairing
$\|\mathbb{G}_1\|$	Length of $\mathbb{G}_1$ in type-III bilinear pairing
$\|\mathbb{G}_2\|$	Length of $\mathbb{G}_2$ in type-III bilinear pairing
$\|\mathbb{G}_C\|$	Length of $\mathbb{G}_C$ in composite-order bilinear pairing
$\|\mathbb{G}_T\|$	Length of $\mathbb{G}_T$

$H_{\rm ch}(C_i) = g^{C_i}{\rm pk}_{\rm ch}^z$ and searches the table T for $H(H_{\rm ch}(C_i))$. If it exists, then $\mathcal{B}$ returns the corresponding result to $\mathcal{A}$. Otherwise, $\mathcal{B}$ chooses $\omega_i \in_R \mathbb{Z}_p$ and computes

$$H_m\left(H_{\rm ch}\left(C_i\right)\right) = \begin{cases} g^{b+\omega_i}, & i = i^*, \\ g^{\omega_i}, & \text{others}, \end{cases} \tag{9}$$

$\mathcal{B}$ stores it in table T and returns it to $\mathcal{A}$.

S-Query. $\mathcal{A}$ adaptively chooses message C_i for signature queries. If $i = i^*$, then abort. Otherwise, $\mathcal{B}$ computes ${\rm Sign}_C = H_m(H_{\rm ch}(C_i))^\beta = (g^\beta)^{\omega_i}$ and gives it to $\mathcal{A}$.

Forgery. $\mathcal{A}$ returns a forged ${\rm Sign}_{C^*}$ about C^* which has never been queried in S-Query before. If $C^* \neq C_{i^*}$, then abort. Otherwise, it returns ${\rm Sign}_{C^*}$.

If $\mathcal{A}$ wins the game, ${\rm Sign}_{C^*}$ is a valid signature so it must have ${\rm Sign}_{C^*} = (g^{b+\omega_i})^\beta = g^{ab+\alpha\omega_i}$, then $\mathcal{B}$ can computes $g^{ab} = {\rm Sign}_{C^*}/g^{a\omega_i}$.

If $\mathcal{A}$ can attack the chameleon signature scheme with advantage δ, we have ${\rm Adv}_{\rm CDH}^{\mathcal{B}} = \frac{1}{Q_H}\delta$.

So, if the CDH problem is hard, the chameleon signature is unforgeable by $\mathcal{A}$. Theorem 2 is proved.

(2) The collision resistance of chameleon hash.

Theorem 3. The chameleon hash $H_{\rm ch}$ is collision-resistant for users without the trapdoor key ${\rm sk}_{\rm ch}$.

Proof. Suppose that there is a user $\mathcal{U}$ who can compute a collision value of $H_{\rm ch}(C)$ without ${\rm sk}_{\rm ch}$. It means that $\mathcal{U}$ can forge a C' which has $H_{\rm ch}(C) = g^C{\rm pk}_{\rm ch}^z = g^{C'}{\rm pk}_{\rm ch}^{z'}$ and $z' = (C + \gamma z - C')/\gamma$, but Y knows the values of $H_{\rm ch}(C)$, C, z and ${\rm pk}_{\rm ch} = g^\gamma$ except γ. Then $\mathcal{U}$ can get γ by computing $\gamma = (C' - C)/(z - z')$.

The discrete logarithm (DL) problem under cyclic group holds that given a tuple $\langle g, g^x\rangle$ where $g \in_R \mathbb{G}$ and $x \in_R \mathbb{Z}_p$, no PPT adversary $\mathcal{A}$ can recover x with a non-negligible advantage. But the user $\mathcal{U}$ mentioned above can recover γ by the tuple $\langle g, g^\gamma\rangle$. Theorem 3 is proved.

7 Performance evaluation

7.1 Numerical analysis

We give the numerical comparison between our scheme and some previous schemes. The definitions of symbols are in Table 1.

In Table 2, we give the comparison of access structure, secret key size, ciphertext size, security model and the category of bilinear pairing. Except for Ref. [19] in which the secret key size is constant, the secret key size increases linearly with the number of user attributes in all other schemes including our scheme. Our scheme has a constant-size ciphertext. There is an intuitive example here, to construct a security level of 80 bits, we can find a pairing group $\mathbb{G}_1$ with 160 bits, a pairing group $\mathbb{G}_2$ with 512 bits, and a $\mathbb{G}_T$ with 1024 bits to satisfy the security level and the bilinear mapping. Then the length of ciphertext $|\mathbb{G}_1| + |\mathbb{G}_2| + |\mathbb{G}_T|$ is 1696 bits which is applicable for light-weight devices a memory size of only a few kilobits. Meanwhile, some other schemes [9, 11, 18, 32] can also achieve a constant ciphertext length, but only the scheme in [11] can provide full security like our scheme. However, the scheme proposed in [11] is implemented in the composite-order bilinear pairing, in which the size of elements is much larger compared to our scheme which is implemented in type-III bilinear pairing.

In Table 3, we give the average time spent on operations such as multiplication, exponentiation, pairing, and the hash operation. We measured the time taken by those operations on three groups of bilinear pairs implemented using Python 2.7 on a laptop with a 3.2 GHz Intel Core i5 processor and 8 GB RAM.

Table 2 Comparison results of the ABE based schemes

Scheme	Access structure	Secret key size	Ciphertext size	Security model	Category of bilinear pairing
[16]	Tree	$(2m+1)\|\mathbb{G}\|$	$(2j+1)\|\mathbb{G}\|+\|\mathbb{G}_T\|$	Selective security	Type-I
[17]	LSSS	$(m+2)\|\mathbb{G}\|$	$(4j+1)\|\mathbb{G}\|+\|\mathbb{G}_T\|$	Selective security	Type-I
[20]	LSSS	$(3m+3)\|\mathbb{G}_1\|+3\|\mathbb{G}_2\|$	$3j\|\mathbb{G}_1\|+3\|\mathbb{G}_2\|+\|\mathbb{G}_T\|$	Full security	Type-III
[10]	LSSS	$(m+2)\|\mathbb{G}_C\|$	$(2j+1)\|\mathbb{G}_C\|+\|\mathbb{G}_T\|$	Full security	Composite-order bilinear pairing
[31]	LSSS	$(m+2)\|\mathbb{G}\|$	$(j+1)\|\mathbb{G}\|+\|\mathbb{G}_T\|$	Selective security	Type-I
[18]	Threshold	$m\|\mathbb{G}_1\|+(n-1)\|\mathbb{G}_2\|$	$\|\mathbb{G}_1\|+\|\mathbb{G}_2\|+\|\mathbb{G}_T\|$	Selective security	Type-III
[9]	Threshold	$(m+n)\|\mathbb{G}\|$	$2\|\mathbb{G}\|+\|\mathbb{G}_T\|$	Selective security	Type-I
[11]	AND	$(m+2)\|\mathbb{G}\|$	$2\|\mathbb{G}_C\|+\|\mathbb{G}_T\|$	Full security	Composite-order bilinear pairing
[19]	AND	$\|\mathbb{G}_1\|+\|\mathbb{G}_2\|$	$(n-j+1)\|\mathbb{G}_1\|+\|\mathbb{G}_2\|+\|\mathbb{G}_T\|$	Selective security	Type-III
[32]	AND	$(3m+1)\|\mathbb{G}\|$	$2\|\mathbb{G}\|+\|\mathbb{G}_T\|$	Selective security	Type-I
Our scheme	AND	$(m+1)\|\mathbb{G}_1\|+\|\mathbb{G}_2\|$	$\|\mathbb{G}_1\|+\|\mathbb{G}_2\|+\|\mathbb{G}_T\|$	Full security	Type-III

Table 3 The average time spent on various operations (ms)

Scheme	Multiplication	Exponentiation	Hash	Paring
$\mathbb{G}_1$	0.017	1.89	0.25	
$\mathbb{G}_2$	0.040	10.56	–	9.16
$\mathbb{G}_T$	0.043	2.50	–	

Table 4 The number of various operations

Scheme	Key generation				Encryption					Decryption					
	$\mathbb{G}_1$			$\mathbb{G}_2$	$\mathbb{G}_1$			$\mathbb{G}_2$		$\mathbb{G}_1$		$\mathbb{G}_2$		$\mathbb{G}_T$	Paring
	Mul.	Exp.	Hash	Exp.	Mul.	Exp.	Hash	Mul.	Exp.	Mul.	Exp.	Mul.	Exp.	Mul.	
[16]	m	$1+2m$	m	m	–	j	j	–	$j+1$	–	–	–	–	$2j+2$	$2j+1$
[17]	1	$1+m$	–	1	j	$2j$	–	–	$j+1$	–	–	–	–	$j+2$	$2j+1$
[19]	–	1	–	1	–	1	–	j	j	$n-j+1$	$n-j+1$	$n-j$	$n-j$	2	3
Ours	–	1	–	$1+m$	–	1	–	j	1	–	–	j	–	4	3

The elements being tested are randomly chosen from the corresponding groups. We performed 100 tests on each operation and took the average time. As shown in Table 3, multiplication (denoted by Mul) takes the shortest time compared to the other operations. The exponentiation (denoted by Exp) operation in group $\mathbb{G}_2$ takes much more time than in groups $\mathbb{G}_1$ and $\mathbb{G}_T$. We only tested the hash operation time under group $\mathbb{G}_1$, because all the hash operations in the schemes under comparison were performed under group $\mathbb{G}_1$. Pairing is the slowest operation except for exponentiation in group $\mathbb{G}_2$.

In Table 4, we give the number of multiplications, exponentiations, pairings, and hash operations involved in these schemes. We applied type-III bilinear pairing to the schemes in [16, 17] like [20]. In the original schemes, they used type-I bilinear pairing which has some security issues. This will not affect the number of operations used in [16, 17]. Note that we omitted operations that are not used in all schemes in the table. In our scheme, the number of pairing operations in the decryption phase is constant. And the number of pairing operations in the schemes [16, 17] increases linearly with the number of attributes in the access structure. In the encryption phase, the number of exponentiation operations in group $\mathbb{G}_2$ is 1 in our scheme, but it increases linearly with the number of attributes in the access structure in the other three schemes.

7.2 Experimental analysis

We implement a testing environment for ABE using Python 2.7 on top of the Charm framework [33]. We compare the encryption time, the decryption time, and the key generation time of the BSW scheme [16] and the Waters' scheme [17] with our scheme under the same setting (a lap-top with a 3.2 GHz Intel Core i5 processor and 8 GB RAM, ubuntu 18.04.3 with 2 GB RAM).

As shown in Figure 4(a), the encryption time of our scheme only has a slight increase when the number of attributes is growing, while the encryption time of the other two schemes [16, 17] increases sharply along with the number of attributes. When the amount of attributes is 100, the encryption time of our scheme is only 12.60 ms, while the other two schemes require 903.89 and 867.08 ms, respectively.

The decryption time of our scheme is almost constant as shown in Figure 4(b). This is because when the

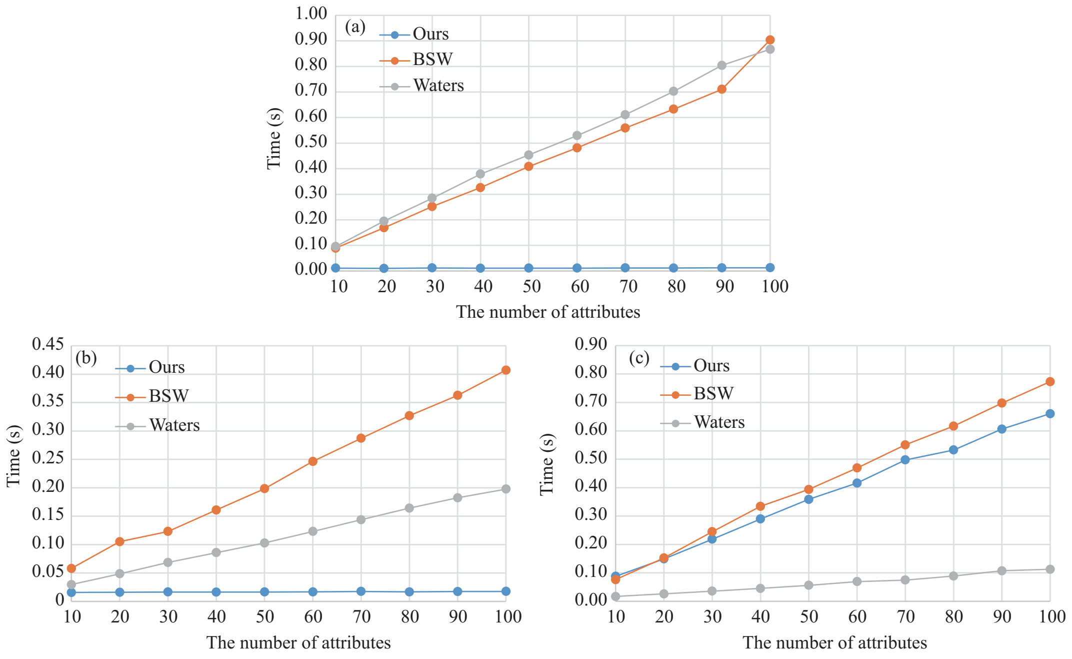

Figure 4 (Color online) (a) Encryption time; (b) decryption time; (c) key generation time.

number of attributes in the access structure is increased by one, the additional deccryption computation in our scheme is just one multiplication operation. When the amount of attributes is 100, the decryption time of our scheme only takes 17.52 ms, while for BSW's scheme that is 407.07 ms, nearly 23 times larger.

As shown in Figure 4(c), our scheme and BSW scheme [16] are similar in terms of the key generation time, while the Waters' scheme [17] shows a big advantage in the key generation phase. However, considering the huge amount of data to be encrypted and decrypted, the time overhead of key generation is trivial and overall our scheme has better performance than the existing schemes. It is clear that all the experimental results mentioned above are consistent with the results of numerical analysis in the numerical analysis section.

8 Conclusion

In this paper, an efficient CP-ABE scheme with short ciphertext and full security is proposed, which is suitable for the lightweight devices in the IoT system. In our scheme, the computation overhead of decryption needs only a constant amount of three pairing operations and some multiplication operations, which is independent of the number of attributes. Additionally, our scheme works with constant-size ciphertext. Moreover, our scheme is fully secure and uses type-III bilinear pairs under the access structure of AND gate, which can achieve an improved security over some previous ABE schemes. To tackle some potential vulnerabilities in practice, we give two enhanced schemes which can prevent information leakage and protect the privacy of data owners, respectively. The performance of our scheme is evaluated against existing schemes. The results prove that our scheme is suitable for the resource-limited IoT system.

In the future work, we will first focus on reducing the computation costs of the key generation phase. Then we will extend our scheme to be applied in other contexts and develop new access structures. Moreover, we will try to construct a distributed authority by combining promising technologies such as blockchain technology.

Acknowledgements This work was jointly supported by Beijing Natural Science Foundation (Grant No. 4182060), National Natural Science Foundation of China (Grant No. 61972148), and Fundamental Research Funds for the Central Universities (Grant No. 2019MS020).

References

1 Wan S H, Zhao Y, Wang T, et al. Multi-dimensional data indexing and range query processing via Voronoi diagram for internet of things. Future Gener Comput Syst, 2019, 91: 382–391

2 Ammar M, Russello G, Crispo B. Internet of things: a survey on the security of IoT frameworks. J Inf Secur Appl, 2018, 38: 8–27

3 Wu Y K, Huang H, Wu Q, et al. A risk defense method based on microscopic state prediction with partial information observations in social networks. J Parallel Distrib Comput, 2019, 131: 189–199

4 Guan Z T, Liu X Y, Wu L F, et al. Cross-lingual multi-keyword rank search with semantic extension over encrypted data. Inf Sci, 2020, 514: 523–540

5 Xu S M, Li Y J, Deng R H, et al. Lightweight and expressive fine-grained access control for healthcare Internet-of-things. IEEE Trans Cloud Comput, 2019. doi: 10.1109/tcc.2019.2936481

6 Xu S M, Yang G M, Mu Y, et al. A secure IoT cloud storage system with fine-grained access control and decryption key exposure resistance. Future Gener Comput Syst, 2019, 97: 284–294

7 Jiang Y H, Susilo W, Mu Y, et al. Flexible ciphertext-policy attribute-based encryption supporting AND-gate and threshold with short ciphertexts. Int J Inf Secur, 2018, 17: 463–475

8 Odelu V, Das A K. Design of a new CP-ABE with constant-size secret keys for lightweight devices using elliptic curve cryptography. Secur Commun Netw, 2016, 9: 4048–4059

9 Susilo W, Yang G M, Guo F C, et al. Constant-size ciphertexts in threshold attribute-based encryption without dummy attributes. Inf Sci, 2018, 429: 349–360

10 Lewko A, Okamoto T, Sahai A, et al. Fully secure functional encryption: attribute-based encryption and (hierarchical) inner product encryption. In: Proceedings of Annual International Conference on the Theory and Applications of Cryptographic Techniques, 2010. 62–91

11 Doshi N, Jinwala D C. Fully secure ciphertext policy attribute-based encryption with constant length ciphertext and faster decryption. Secur Commun Netw, 2014, 7: 1988–2002

12 Odelu V, Das A K, Khan M K, et al. Expressive CP-ABE scheme for mobile devices in IoT satisfying constant-size keys and ciphertexts. IEEE Access, 2017, 5: 3273–3283

13 Banerjee S, Roy S, Odelu V, et al. Multi-authority CP-ABE-based user access control scheme with constant-size key and ciphertext for IoT deployment. J Inf Secur Appl, 2020, 53: 102503

14 Cui H, Deng R H, Liu J K, et al. Server-aided attribute-based signature with revocation for resource-constrained industrial-Internet-of-things devices. IEEE Trans Ind Inf, 2018, 14: 3724–3732

15 Guan Z T, Li J, Wu L F, et al. Achieving efficient and secure data acquisition for cloud-supported Internet of things in smart grid. IEEE Int Things J, 2017, 4: 1934–1944

16 Bethencourt J, Sahai A, Waters B. Ciphertext-policy attribute-based encryption. In: Proceedings of IEEE Symposium on Security and Privacy, 2007. 321–334

17 Waters B. Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization. In: Proceedings of International Workshop on Public Key Cryptography, 2011. 53–70

18 Herranz J, Laguillaumie F, Rafols C. Constant size ciphertexts in threshold attribute-based encryption. In: Proceedings of International Workshop on Public Key Cryptography, 2010. 19–34

19 Guo F C, Mu Y, Susilo W, et al. CP-ABE with constant-size keys for lightweight devices. IEEE Trans Inf Forensic Secur, 2014, 9: 763–771

20 Agrawal S, Chase M. FAME: fast attribute-based message encryption. In: Proceedings of ACM SIGSAC Conference on Computer and Communications Security, 2017. 665–682

21 Teng W, Yang G, Xiang Y, et al. Attribute-based access control with constant-size ciphertext in cloud computing. IEEE Trans Cloud Comput, 2017, 5: 617–627

22 Odelu V, Das A K, Rao Y S, et al. Pairing-based CP-ABE with constant-size ciphertexts and secret keys for cloud environment. Comput Stand Interface, 2017, 54: 3–9

23 Xu S M, Yang G M, Mu Y, et al. Secure fine-grained access control and data sharing for dynamic groups in the cloud. IEEE Trans Inf Forensic Secur, 2018, 13: 2101–2113

24 Chen C, Chen J, Lim H W, et al. Fully secure attribute-based systems with short ciphertexts/signatures and threshold access structures. In: Proceedings of Cryptographers' Track at the RSA Conference, 2013. 50–67

25 Xu S M, Yang G M, Mu Y. Revocable attribute-based encryption with decryption key exposure resistance and ciphertext delegation. Inf Sci, 2019, 479: 116–134

26 Liu Z H, Duan S H, Zhou P L, et al. Traceable-then-revocable ciphertext-policy attribute-based encryption scheme. Future Gener Comput Syst, 2019, 93: 903–913

27 Zhang Y H, Zheng D, Deng R H. Security and privacy in smart health: efficient policy-hiding attribute-based access control. IEEE Int Things J, 2018, 5: 2130–2145

28 Han Q, Zhang Y H, Li H. Efficient and robust attribute-based encryption supporting access policy hiding in Internet of things. Future Gener Comput Syst, 2018, 83: 269–277

29 Menezes A J, Okamoto T, Vanstone S A. Reducing elliptic curve logarithms to logarithms in a finite field. IEEE Trans Inf Theory, 1993, 39: 1639–1646

30 Galbraith S D, Paterson K G, Smart N P. Pairings for cryptographers. Discrete Appl Math, 2008, 156: 3113–3121

31 Malluhi Q M, Shikfa A, Trinh V C. A ciphertext-policy attribute-based encryption scheme with optimized ciphertext size and fast decryption. In: Proceedings of ACM on Asia Conference on Computer and Communications Security, 2017. 230–240

32 Zhou Z B, Huang D J. On efficient ciphertext-policy attribute based encryption and broadcast encryption. In: Proceedings of the 17th ACM Conference on Computer and Communications Security, 2010. 753–755

33 Akinyele J A, Garman C, Miers I, et al. Charm: a framework for rapidly prototyping cryptosystems. J Cryptogr Eng, 2013, 3: 111–128