

Article

Privacy-Preserving Broker-ABE Scheme for Multiple Cloud-Assisted Cyber Physical Systems

Po-Wen Chi ^{1,*}  and Ming-Hung Wang ² 

¹ Department of Computer Science and Information Engineering, National Taiwan Normal University, Taipei 11677, Taiwan

² Department of Information Engineering and Computer Science, Feng Chia University, Taichung 40724, Taiwan; mhwang@fcu.edu.tw

* Correspondence: neokent@gapps.ntnu.edu.tw

Received: 11 November 2019; Accepted: 9 December 2019; Published: 11 December 2019



Abstract: Cloud-assisted cyber-physical systems (CCPSs) integrate the physical space with cloud computing. To do so, sensors on the field collect real-life data and forward it to clouds for further data analysis and decision-making. Since multiple services may be accessed at the same time, sensor data should be forwarded to different cloud service providers (CSPs). In this scenario, attribute-based encryption (ABE) is an appropriate technique for securing data communication between sensors and clouds. Each cloud has its own attributes and a broker can determine which cloud is authorized to access data by the requirements set at the time of encryption. In this paper, we propose a privacy-preserving broker-ABE scheme for multiple CCPSs (MCCPS). The ABE separates the policy embedding job from the ABE task. To ease the computational burden of the sensors, this scheme leaves the policy embedding task to the broker, which is generally more powerful than the sensors. Moreover, the proposed scheme provides a way for CSPs to protect data privacy from outside coercion.

Keywords: attribute-based encryption; multiple cloud-assisted cyber-physical system; deniable encryption

1. Introduction

Cyber-physical systems (CPSs), first proposed in Rajkumar et al. [1], is a concept in which field data are monitored and collected by sensor systems and are then relayed to computer systems. Computer systems then use computer-based algorithms to make decisions regarding how field agents should proceed to obtain better results and pass these commands on field agents. This architecture is also the main concept underpinning Industry 4.0 [2,3] and the internet of things (IoT) [4].

Today, cloud computing provides convenient, reliable, and on-demand services and is becoming increasingly popular. Accordingly, the cyber component in CPS is being extended to cloud computing to conform to the trend of the time. In 2013, National Institute of Standards and Technology (NIST) integrated cloud computing and CPS and proposed an architecture known as the cloud-assisted cyber-physical system (CCPS) [5] (actually, NIST uses the term cyber-physical cloud computing (CPCC). In this paper, we prefer to use CCPS since this term is consistent with the term CPS). The associated white paper [5] lists five benefits of CCPS: (1) efficient use of resources, (2) modular composition, (3) rapid development, (4) smart adaption to the environment at every scale, (5) reliability and resiliency.

The modular composition aspect is very important. Since it is almost impossible for one cloud service vendor to provide every type of service, many cloud services are composed of multiple sub-cloud systems. For example, when a user wants to choose a restaurant, he may draw upon a number of cloud systems simultaneously to do so, including a map service, a customer comment service, and a

booking service. This is also the case in CCPS. Field sensors collect different types of data. Each CSP may require a specific subset of data for analysis, decision-making, and the provision of its own unique service. For example, in a hospital, there may be lots of sensors in a patient, and they periodically report to different health-care systems, like the disease tracking system, patient caring system, or even air conditioning system. This facility is called a multiple CCPS (MCCPS). For this scenario, a new entity called a broker is introduced between the field sensors and CSPs [6,7]. The broker is responsible for dispatching sensor data to appropriate CSPs according to their requirements or CSP properties. In general, the IoT gateway can play the broker role. Compared to the sensor, the IoT gateway is usually considered to be much more powerful. So it is reasonable to move most computational works from the sensor to the broker. Figure 1 shows a diagram of the broker architecture for the MCCPS.

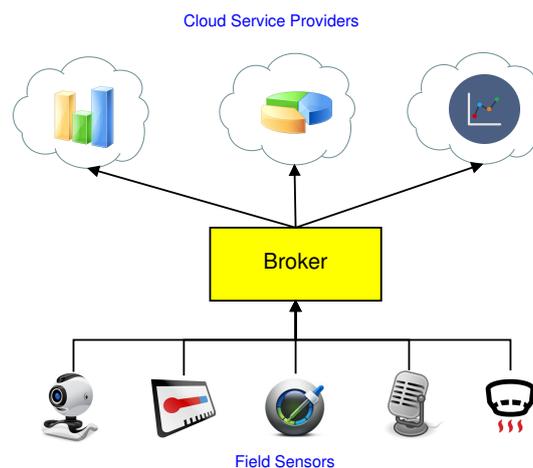


Figure 1. Broker architecture for the multiple cyber-physical cloud computing (MCCPS).

To secure data communication in the MCCPS, attribute-based encryption (ABE) is an appropriate technique since sensor data are required by multiple CSPs at the same time. By embedding the appropriate policy-checking mechanism, only CSPs that satisfy pre-defined policies are authorized to access field data. As such, data can be encrypted once and then be forwarded to multiple possible cloud service vendors. However, most ABE schemes require intensive computational resources. Considering the limited sensor capabilities, applying existing ABE schemes to sensors may not be a good idea.

Data privacy is another important issue in the MCCPS. Since data are processed in clouds, some entities may force clouds to release field data through some type of power enforcement. For example, a patient may wear many biosensors to monitor his/her health status. These biosensors forward collected data to the cloud for further analysis and health determination. If some third party forces the cloud service vendor to release the patient's health information, traditional encryption schemes would not protect user privacy since CSPs handle unencrypted data. Other than expecting the CSP to reject outside requests directly, CSPs must be provided with a tool to prevent coercion and ensure data privacy.

To solve these two problems, we propose a privacy-preserving broker-based ABE scheme in which the policy generation role is moved from the sender, which is the field sensor in this scenario, to the broker. The reason for this is that the broker is typically more powerful than the sensor. As for the data privacy issue, we use a deniable encryption property to protect CSPs from being coerced. When being coerced by outside forces, CSPs can release fake data that include persuasive evidence, and the outside forces cannot reject claims from the CSPs since the claims are indistinguishable from those of real data. Thus, the real data are kept safe from coercion.

A deniable encryption scheme is a kind of non-commitment encryption scheme since it is possible to interpret an existing ciphertext into a fake message other than the user data. Some deniable encryption schemes will be described in Section 2. In general, these schemes cost a lot of resources and it is not affordable for a sensor to implement them. For example, Canetti et al. proposed a bitwise deniable

encryption scheme that requires each transmission bit has an independent encryption environment [8]. To overcome this problem, our design is pre-determined and multi-distributional. We show that the cost of our scheme is reasonable for a sensor in Section 4.

1.1. Contributions

In this paper, we present a deniable broker-based CP-ABE scheme for the MCCPS scenario. The advantages of this scheme are as follows:

- **Broker-based ABE.** Most ABE schemes consider only the sender and receiver. In a ciphertext-policy ABE (CP-ABE) scheme, the sender must embed a policy-checking mechanism in the ciphertext generation process. Generally speaking, the computational cost of this process depends on the number of attributes included. When more attributes are taken into account, more computation power is required in the encryption process. That is, when applying CP-ABE in the MCCPS scenario, many attributes are required and the field sensor may not have sufficient resources to encrypt the collected data.

To solve this problem, in this scheme, the broker assumes the policy embedding task. The sensor encrypts the collected data without considering the required policy. The semi-finished ciphertext, which as yet remains confidential, is forwarded to the broker. The broker then processes the received ciphertext and re-encrypts the ciphertext into a new version which requires policy checking upon decryption. As such, the encryption step, which requires the most computational power, is moved from the sensor to the broker. Therefore, broker-based ABE is more suitable for the MCCPS scenario.

- **Arbitrary policy support without dedicated keys.** The broker-based ABE is a variant of the proxy-ABE, where the broker role is similar to that of a proxy. This intermediary step transforms the given ciphertext into a new ciphertext. The proxy-ABE requires a re-encryption key, which is often bundled with the new policy. However, this is not practical in the MCCPS scenario since there are many types of sensors, many CSPs, and having a re-encrypt key for each combination would be almost impossible. Besides, as described above, the sensor cannot complete the policy embedding task due to its limited computational resources. So, the broker does not transform the ciphertext from one policy into another, but rather adds a policy-checking step to an existing ciphertext. Moreover, without transformation by the broker, CSPs cannot decrypt these messages from the sensors. This ensures that a message from a field sensor must pass through the broker and have had the policy-checking step appended before being received by the CSP. Thirdly, although the broker is an intermediate entity that participates in the encryption process, it is not allowed to see the data from field sensors. This arrangement is known as a semi-trusted broker.

In this work, to achieve this feature, we use a composite order group, which can be divided into two subgroups. The sender, which is a sensor in this scenario, encrypts a message in the composite order group. The broker and the CSP have keys for different respective subgroups. When receiving a ciphertext, the broker first removes the secret key for the particular subgroup. Then, the broker applies the attribute encryption mechanism to the other subgroup and forward the new ciphertext to the CSP. Note that no re-encryption key is required when embedding policies. The broker only takes public information to embed the access structure into the ciphertext. Since the CSP has the key of the subgroup, the CSP can correctly decipher the message.

- **Lightweight and blockwise deniable ABE.** Deniable encryption is an encryption scheme in which the sender and receiver can persuade others that a given ciphertext is from a fake rather than a real message. Most deniable encryption schemes are bitwise encryption schemes that encrypt only one bit at a time. Undoubtedly, a bitwise encryption can support any kind of message by repeating the encryption process many times. However, in practice, this is not efficient especially, when considering the MCCPS scenario, since the sensor cannot support this approach due to its resource limitations.

Following the concept proposed by Chi et al. [9], in this work, we constructed a blockwise deniable encryption scheme for the MCCPS scenario. The sensor can easily generate two messages, one using real data and the other fake data, and prepare two convincing pieces of evidence respectively. The evidence is taken from a chameleon hash function, for which the collision derivation is very lightweight, so the overall computation required by the sensor is much less than that required in other deniable encryption schemes.

Note that in this MCCPS scenario, we focus only on the sensor and CSP in providing the deniability feature, without considering the broker. The reason for this is that the broker cannot see the content of the ciphertext, and can, therefore, make no claims about the content.

- Multiple brokers support. In the MCCPS scenario, there are many cloud services that may be shared by sensors from different fields. That is, one cloud service must simultaneously support multiple brokers and the sensors that are under these brokers. It is a trivial matter that each broker and CSP share one unique encryption environment, which is usually the public key. However, this will increase the management complexity since many brokers and multiple public keys are required.

In this work, we separate public information into two parts. The first part comprises system-wise public information and the second broker-wise public information. Broker-wise public information is only shared by one broker and the sensors that are under this particular broker. The CSP needs only system-wise public information. This is an advantage that will decrease the public key management burden on the CSP.

1.2. Organization

The rest of the paper is organized as follows. In Section 2, we briefly introduce some existing ABE, deniable encryption (DE) techniques and the preliminaries used in this scheme. We present the proposed scheme in Section 3. In Section 4, we evaluate the scheme's security and performance. In Section 5, we further discuss the proposed scheme. We then conclude the paper and address future work in Section 6.

2. Related Works

2.1. Previous Works on Attribute-Based Encryption

ABE is an encryption system in which only those who satisfy some specific conditions can successfully decrypt the ciphered data. This ABE concept was first introduced by Sahai and Waters [10] (although identity-based encryption (IBE) is also a type of ABE and was first proposed in Boneh et al. [11], here we only focus on ABE review and do not consider IBE schemes). There are two types of ABE; key-policy ABE (KP-ABE) and cipher-policy ABE (CP-ABE). Their difference lies in where the policy-checking step occurs. KP-ABE embeds the decryption policy in the key generation process, whereas CP-ABE embeds the policy in the encryption process. Goyal et al. proposed the first KP-ABE construction [12] and Bethencourt et al. proposed the first CP-ABE [13]. Both these schemes support monotonic formula policy checking. In this paper, we focus only on CP-ABE schemes because CP-ABE enables the sender to decide who is allowed to see the content. The MCCPS scenario implies that the broker can decide which cloud services should be launched and that the policy can be modified by the broker. As such, we only review CP-ABE related techniques. Waters proposed the first fully expressive CP-ABE based on linear secret sharing schemes (LSSS) [14]. In this paper, we use Water's CP-ABE as the base scheme and enhance it to a privacy-preserving broker-ABE. Lewko et al. [15] improved the Waters scheme [14] to a fully secure CP-ABE, albeit with some efficiency loss. Attrapadung et al. [16] constructed a CP-ABE that supports a constant-size ciphertext. Tysowski et al. [17] designed their CP-ABE scheme to support dynamic key management for mobile users.

There is a variant ABE called proxy-ABE, which is a concept for importing a new entity called a proxy that can re-encrypt ciphertexts. When the sender encrypts data for some group, the proxy can

transform the ciphertext from one receiver group to another. In the re-encryption process, the proxy is not allowed to decrypt the ciphertext. In this paper, since the broker is at the same logical location as the proxy and also performs the re-encryption process, we review some proxy-ABE studies. Luo et al. [18] first proposed ciphertext-policy attribute-based proxy re-encryption (CP-AB-PRE), which allows a proxy to change the receiver group from one to another. However, their scheme only supports the AND-gate operation. Li [19] enhanced the Luo et al. scheme [18] by using a matrix access structure. Liang et al. built a secure proxy-ABE in the chosen-ciphertext model [20]. Chandar et al. [21] used proxy to re-encrypt ciphertexts for removing the access right of revoked users. Touati et al. [22] studied the issue of private key updating in a CP-AB-PRE scheme.

Some researchers also focus on how to apply ABE on the IoT environment. Yao et al. claimed that the bilinear operation is expensive, so they built their scheme the elliptic curve cryptography without pairing [23]. Oualha et al. made the sensor compute lots of elements in advance to accelerate the encryption procedure [24]. However, this approach required the sensor to have a large space or to interact with a trusted party. Some used fog/edge nodes as the intermediate layer and implement mechanisms on them. Jiang et al. focused on the key-delegation abuse problem and solved this problem with a traitor [25]. Zuo et al. developed ABE with outsourced decryption for the fog computing [26]. Li et al. outsourced the exponential operation to speed up both the encryption and decryption process [27]. Fischer et al. used a proxy to convert a ciphertext into an ABE ciphertext [28]. The problem of Fischer approach is that it is an interactive model. The receiver needs to communicate with some authority before decryption.

In this paper, we propose an ABE scheme called broker-based ABE. We outsource partial encryption works to the entity called broker. Although the broker ABE is a kind of proxy-ABE, it is actually more than that. The broker-ABE is an idea that integrates secret-sharing and proxy re-encryption, whereby the sender encrypts a message using the secret shared by the broker and CSP. So, neither the broker nor the CSP can decrypt the ciphertext individually. For re-encryption, the broker can add a policy-checking mechanism to ciphertexts to verify the CSPs' attributes.

2.2. Previous Works on Deniable Encryption

The idea of DE was first proposed by Canetti et al. [8]. Like other encryption schemes, DE provides semantic security to protect encrypted data. In addition, DE allows the sender and/or receiver to persuade outside coercers that the given ciphertexts have been encrypted from fake messages. A number of DE techniques have been proposed. Since ABE is a public key encryption system, here we only review some deniable public-key encryption works. In the scheme proposed by Canetti et al. [8], the authors used a translucent set to provide fake messages with convincing evidence. A translucent set is one that contains a trapdoor subset. It is easy to choose a random element from the universal set or the subset, but it is difficult to determine if a given element belongs to the subset without the use of the trapdoor. If a sender wants to encrypt one bit 0, the sender sends an element not contained in the subset. To encrypt one bit 1, the sender sends an element that is contained in the subset. When being coerced, the sender simply claims a bit from 1 to 0 by claiming the random element from the universal set that lies in the subset coincidentally. Although this is a sender-deniable scheme, Canetti et al. proved that this scheme can be extended to a bi-deniable scheme through an interactive model. Following this idea, Durmuth et al. [29] used samplable encryption to build a translucent set. O'Neill et al. [30] constructed a bi-translucent set based on a lattice. So, without using an interactive model, this scheme can provide both sender and receiver deniability, which is called bi-deniability.

Apart from translucent-based schemes, other techniques are used to build DE schemes. O'Neill et al. [30] made use of a simulatable public key system and used the voting approach to provide deniability. O'Neill et al. also proposed a new concept called multi-distributional deniability. They proposed two sets of algorithms and their outputs are computationally indistinguishable. So a user can use one set but claims using the other one. This approach can support non-interactivity and fully deniability at the same time. Gasti et al. [31] proposed a DE scheme that creates a secret key

pair between the sender and receiver. Chi et al. [9] proposed a DE scheme based on composite order groups and used chameleon hash functions to make fake data convincing.

In our proposed scheme, we use a DE scheme to protect user data privacy in CSPs. When being forced to release field data, CSPs can provide false data to outside coercers, with evidence in the existing ciphertexts. Note that there are three types of DE schemes: sender-deniable encryption, receiver-deniable encryption, and bi-deniable encryption. The names specify the entity that can generate fake evidence. In the MCCPS scenario, since senders are usually sensors with limited resources and cannot store data for very long, this paper focuses on a receiver-DE scheme to protect the data privacy of data stored in CSPs. The broker is not allowed to see the content of ciphertexts and therefore cannot participate in the coercion process.

2.3. Preliminaries

2.3.1. Prime Order Bilinear Groups

Let \mathbb{G} and \mathbb{G}_T be two multiplicative cyclic groups of prime order p , with map function $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$. Let g be a generator of \mathbb{G} . \mathbb{G} is a bilinear map group if \mathbb{G} and e have the following properties:

- Bilinearity: $\forall u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}$, $e(u^a, v^b) = e(u, v)^{ab}$.
- Non-degeneracy: $e(g, g) \neq 1$.
- Computability: the group action in \mathbb{G} and map function e can be computed efficiently.

2.3.2. Waters CP-ABE Scheme

In this paper, we extend the Waters CP-ABE scheme [14] to a privacy-preserving broker CP-ABE scheme. Waters used the LSSS as the basis of his construction. To form the secret, he treated attributes as secret shares and policies. So if an entity can rebuild the secret, this implies that the entity has enough attributes to satisfy the given policies. The LSSS used by Waters is as follows:

Definition 1 (LSSS: Linear Secret Sharing Schemes [32]). *A secret sharing scheme Π over a set of parties \mathcal{P} is called linear (over \mathbb{Z}_p) if*

1. *The shares for each party form a vector over \mathbb{Z}_p .*
2. *There exists an $l \times n$ matrix M called the share-generating matrix for Π . For all $i = 1, \dots, l$, the i 'th row of M is labeled by party $\rho(i)$, where ρ is a mapping function from $\{1, \dots, l\}$ to party field \mathcal{P} . When considering column vector $v = (s, r_2, \dots, r_n)$, where $s \in \mathbb{Z}_p$ is the secret to be shared and $r_2, \dots, r_n \in \mathbb{Z}_p$ are randomly chosen, Mv is the vector of l shares of secret s according to Π . The share $(Mv)_i$ belongs to party $\rho(i)$.*

Based on the above definition, an LSSS has a linear reconstruction property. That is, with an LSSS Π , an access structure \mathbb{A} , and valid shares of a secret s , s can be easily recovered. Beimel [32] showed that the recovery procedure is time polynomial to the message size. For an ABE scheme, a party in \mathcal{P} can be treated as an attribute. So if the receiver has attributes that satisfy the decryption policy, this implies that the receiver can recover the secret s and correctly decrypt the message. The Waters CP-ABE scheme includes the following algorithms:

- $\text{Setup}() \rightarrow (MSK, PK)$: This algorithm chooses a bilinear group of prime order p with generator g , random elements $\alpha, a \in \mathbb{Z}_p$, and hash function $H : \{0, 1\}^* \rightarrow \mathbb{G}$. The public key PK is $\{g, e(g, g)^\alpha, g^a\}$ and the system secret key MSK is g^α .
- $\text{Encrypt}(PK, (\mathcal{M}, \rho), M) \rightarrow CT$: Given message M and LSSS access structure (\mathcal{M}, ρ) , this algorithm first chooses a random vector $\vec{v} = (s, y_2, \dots, y_n) \in \mathbb{Z}_p^n$. Let \mathcal{M} be an $l \times n$ matrix and

\mathcal{M}_i denote the i th row of \mathcal{M} . This algorithm calculates $\lambda_i = \vec{v} \cdot \mathcal{M}_i, \forall i \in \{1, \dots, l\}$. Next, this algorithm chooses $r_1, \dots, r_l \in \mathbb{Z}_p$. The output ciphertext will be as follows:

$$\begin{aligned} CT &= \{M \cdot e(g, g)^{as}, g^s, (g^{a\lambda_1} H(\rho(1))^{-r_1}, g^{r_1}), \\ &\quad \dots, (g^{a\lambda_l} H(\rho(l))^{-r_l}, g^{r_l})\} \\ &= \{C, C', (C_1, D_1), \dots, (C_l, D_l)\}, \end{aligned}$$

with a description of (\mathcal{M}, ρ) .

- **KeyGen**(MSK, S) \rightarrow SK: Given set S of attributes, this algorithm chooses $t \in \mathbb{Z}_p$ randomly and outputs the private key as:

$$K = g^{\alpha+at}, L = g^t, \forall x \in SK_x = H(x)^t.$$

- **Decrypt**(CT, SK) \rightarrow M: Suppose that S satisfies the access structure and let $I \subset \{1, \dots, l\}$ be defined as $I = \{i : \rho(i) \in S\}$. This algorithm then finds a set of constants $\{w_i \in \mathbb{Z}_p\}$ such that $\sum_{i \in I} w_i \lambda_i = s$. The decryption algorithm computes

$$e(C', K) / \left(\prod_{i \in I} (e(C_i, L) e(D_i, K_{\rho(i)}))^{w_i} \right) = e(g, g)^{as}$$

and derives M from the ciphertext.

The security of Waters CP-ABE scheme is based on the decisional q -parallel bilinear Diffie–Hellman exponent (BDHE) assumption, which is defined as follows:

Definition 2 (Decisional q -parallel BDHE Assumption). Let $a, s, b_1, \dots, b_q \xleftarrow{R} \mathbb{Z}_p$ and g be a generator of \mathbb{G} . Given

$$D := \left\{ \begin{array}{l} g, g^s, g^a, \dots, g^{(a^q)}, g^{(a^{q+2})}, \dots, g^{(a^{2q})} \\ \forall_{1 \leq j \leq q} \quad g^{s \cdot b_j}, g^{a/b_j}, \dots, g^{(a^q/b_j)}, \\ \quad g^{(a^{q+2}/b_j)}, \dots, g^{(a^{2q}/b_j)} \\ \forall_{1 \leq j, k \leq q, k \neq j} \quad g^{a \cdot s \cdot b_k / b_j}, \dots, g^{a^q \cdot s \cdot b_k / b_j} \end{array} \right\}$$

and element $T \in \mathbb{G}_T$, we assume that for any probabilistic polynomial time (PPT) algorithm A that outputs in $\{0, 1\}$,

$$Adv_A := |P[A(D, e(g, g)^{a^{q+1}s}) = 1] - P[A(D, T) = 1]|$$

is negligible.

Theorem 1. Suppose the decisional q -parallel BDHE assumption holds, then no polynomial-time adversary can selectively break the Waters CP-ABE system in the CPA-model.

In this paper, we also use this access structure to build our ABE system. Waters CP-ABE scheme is semantically secure when the decisional q -parallel BDHE assumption holds. More construction details and the security proof can be found in Waters' work [14].

2.3.3. Composite Order Bilinear Groups

The composite order bilinear group was first introduced by Boneh et al. [33]. Let \mathbb{G} and \mathbb{G}_T be two multiplicative cyclic groups of composite order $N = p_1 p_2 \dots p_m$, where p_1, p_2, \dots, p_m are distinct primes, with a bilinear map function $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$. \mathbb{G} has a subgroup \mathbb{G}_{p_i} of order p_i for each prime p_i . Let g_1, g_2, \dots, g_m be the generators of these subgroups respectively. Each element in \mathbb{G} can be expressed in the form $g_1^{a_1} g_2^{a_2} \dots g_m^{a_m}$, where $a_1, a_2, \dots, a_m \in \mathbb{Z}_N$.

Orthogonality is an important property of the composite bilinear groups. If $u \in \mathbb{G}_{p_i}, v \in \mathbb{G}_{p_j}$ and $i \neq j$, then $e(u, v) = 1$, where 1 is the identity element in \mathbb{G}_T .

There is a complexity assumption in the composite group called the subgroup decision assumption. In ref. [34], the definition is given as follows:

Definition 3 (General Subgroup Decision Assumption). *Let $S_0, S_1, S_2, \dots, S_k$ be non-empty subsets of $1, \dots, m$ such that for each $2 \leq j \leq k$, either $S_j \cap S_0 = \emptyset = S_j \cap S_1$ or $S_j \cap S_0 \neq \emptyset \neq S_j \cap S_1$. Given group generator \mathcal{G} , we define the following distribution:*

$$\begin{aligned} PP &:= \{N = p_1 p_2 \dots p_m, \mathbb{G}, \mathbb{G}_T, e\} \xleftarrow{R} \mathcal{G} \\ Z_i &\xleftarrow{R} \mathbb{G}_{S_i} \forall i \in \{1, \dots, k\}, \\ D &:= \{PP, Z_2, \dots, Z_k\}. \end{aligned}$$

We assume that for any PPT algorithm A with output in $\{0, 1\}$,

$$\text{Adv}_{\mathcal{G}, A} := |P[A(D, Z_0) = 1] - P[A(D, Z_1) = 1]|$$

is negligible.

This assumption implies that it is hard to determine whether two given elements are composed of the same subgroups when they contain at least one common subgroup. In this work, we utilize this property to construct a deniable ciphertext that is indistinguishable to a normal ciphertext.

2.3.4. Chameleon Hash

The chameleon hash scheme, which was first introduced by Krawczyk et al. [35], uses trapdoor pseudo-random permutation functions as one-way functions. Without the trapdoor, a pseudo-random permutation function can output random results that satisfy the collision resistance and semantic security of a one-way hash function. With the trapdoor, it is easy to generate collisions. Most public-key encryption systems can be used as trapdoor pseudo-random permutation functions. That is, it is easy to construct a chameleon hash function from a public key system. The input of a chameleon hash has two parts, the input message m and the random string r . The random string r is a parameter used to provide an opportunity to generate a collision. There are three algorithms in a chameleon hash scheme, as defined below.

1. $\text{Setup}(1^\lambda) \rightarrow \{PK, SK\}$: Given a security parameter, the scheme outputs public parameter PK and secret trapdoor SK .
2. $\text{Hash}(PK, m, r) \rightarrow h$: An efficient and probabilistic algorithm, with inputs PK , a message m , and a random string r , outputs a hash value h .
3. $\text{Forgery}(PK, SK, m_0, r_0, m_1) \rightarrow r_1$: An efficient and probabilistic algorithm, with a given message m_0 , a random r_0 , the trapdoor SK and another message m_1 , outputs a random string r_1 that satisfies the following equation:

$$\text{CH}(m_0, r_0) = \text{CH}(m_1, r_1).$$

There are three associated requirements, collision resistance, semantic security and collision forgery, are listed below.

Definition 4 (Collision Resistance). *Given a chameleon hash scheme $\{PK, SK, \text{CH}(\cdot, \cdot)\}$, where PK is the public information, SK is the trapdoor and $\text{CH}(\cdot, \cdot)$ is the hash function. Let m, m' be two different messages and r a random string. We call the scheme collision resistant if for any PPT algorithm A , it is hard to output r' such that $\text{CH}(m, r) = \text{CH}(m', r')$ without SK .*

Definition 5 (Semantic Security). *Given a chameleon hash scheme $\{PK, SK, \text{CH}(\cdot, \cdot)\}$, where PK is the public information, SK is the trapdoor and $\text{CH}(\cdot, \cdot)$ is the hash function. We call the scheme semantically secure*

if for all pairs of message m, m' and random string r , the probability distribution of $CH(m, r)$ and $CH(m', r)$ are computationally indistinguishable.

Definition 6 (Collision Forgery). Given a chameleon hash scheme $\{PK, SK, CH(\cdot, \cdot)\}$, where PK is the public information, SK is the trapdoor and $CH(\cdot, \cdot)$ is the hash function. Let m, m' be two different messages and r is a random string. We call the scheme a collision forgery scheme if there exists one PPT algorithm A that on input SK , outputs a string r' that satisfies $CH(m, r) = CH(m', r')$.

In the remainder of this paper, for simplicity, we use CH to denote the chameleon hash public information and $CH(\cdot, \cdot)$ to denote the chameleon hash operation.

3. Privacy-Preserving Broker-ABE Scheme

3.1. Overview and Attack Model

The privacy-preserving broker-ABE scheme includes four roles, the trusted key server, the sender, the broker, and the receiver. The key server generates the encryption environment and related keys. The sender, which is usually a sensor in the MCCPS scenario, is in charge of data encryption and forwarding ciphertexts to a broker. The broker re-encrypts the received ciphertexts by embedding policies according to the sender's identity. Then the broker sends the ciphertexts to all possible receivers, which are CSPs in the MCCPS. If a CSP's attributes satisfy the policy setup by the broker, the CSP can successfully decrypt the ciphertexts and process the data collected by senders. This procedure is illustrated in Figure 2.

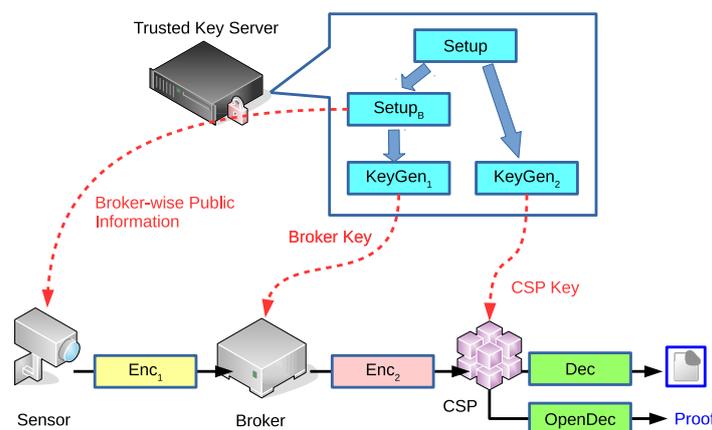


Figure 2. Key server, sensor, broker and cloud service provider (CSP) in the MCCPS.

Now we describe how the attacker can attack this system. In our scheme, we focus on two attack ways. First, we assume that an attacker can intercept all encrypted data, including data between the sensor and the broker and between the broker and the CSP. So the transmission data must be encrypted confidentially. Those who cannot satisfy the ciphertext policy should not be able to decrypt the message. Second, we assume that the attacker can force the CSP to release the user data with a proof to show that the data is decrypted from the ciphertext. Undoubtedly, the most trivial proof is the CSP key so the attacker can decrypt the ciphertext itself. However, the user data will be released to the attacker. We want to protect the CSP under this attack scenario.

The first attack is common for most encryption schemes. The security model used in this paper is described in Section 3.3. As for the second attack way, we apply the concept of deniable encryption. We make the CSP convince the attacker with fake data so the real data is kept secret. The key point is that the fake data must be verified with the ciphertext captured by the attacker.

In this paper, we use the multi-distributional deniable encryption approach. The idea of the multi-distributional deniable encryption scheme is to use two different sets of algorithms. One is

the normal set and the other is the deniable set. The outputs generated from these two sets are computationally indistinguishable. So the whole system claims to use the normal set while the system actually uses the deniable set and the attacker cannot challenge the claim. In the deniable set, the sender can embed a fake message in the encryption step. Since the ciphertext is indistinguishable to the ciphertext generated from the normal encryption algorithm, we can claim the ciphertext is normally encrypted and there is no fake data. The process can be shown in Figure 3.

We simply use the secret key in the CSP as the proof, so we use the decryption algorithm as the verification function.

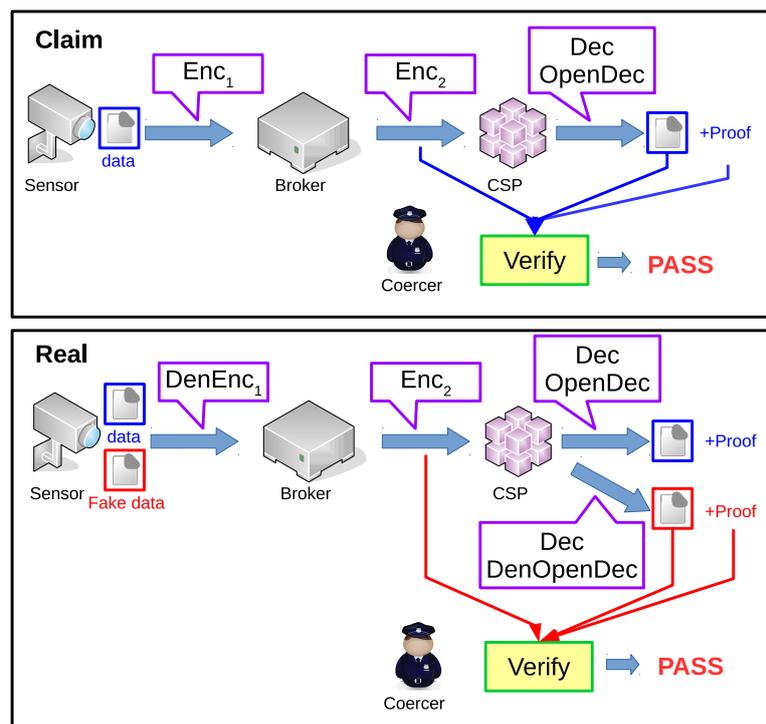


Figure 3. Data privacy is protected from outside coercion. The CSP can determine which algorithm should be used when being coerced. With different algorithms, the coercer will get convincing proofs for different messages.

Note that in our design, the broker will not be allowed to participate in the deniable process.

3.2. Definition

The notations listed in Table 1 are applied throughout this paper.

Table 1. List of notations.

Symbol	Meaning
PP, MSK	System-wise public information and master key.
PP_B, MSK_B	Broker-wise public information and master key.
SK_1, SK_2	The broker's secret key and the CSP's secret key.
C, C^*, C'	The ciphertext encrypted by the sensor, the ciphertext re-encrypted by the broker and the ciphertext deniably encrypted by the sensor.
A, S	The access structure for a ciphertext and an attribute set.
M, M'	The message and the predefined fake message.
PK	System-wise public information but not known to outsiders. That is, PK is claimed to be not existent.

The algorithms of the privacy-preserving broker-ABE scheme are described below:

1. $\text{Setup}(1^\lambda) \rightarrow (PP, MSK)$: This algorithm takes security parameter λ as input and returns public parameter PP and system master key MSK .
2. $\text{Setup}_B(PP, MSK, ID_B) \rightarrow (PP_B, MSK_B)$: This algorithm takes the broker ID and system-wise information as input and outputs broker-wise public parameter PP_B and broker-wise master key MSK_B . Note that this information is only for the broker and the sensors under that broker. This algorithm enables the whole system to simultaneously support multiple brokers.
3. $\text{KeyGen}_1(MSK_B) \rightarrow SK_1$: This algorithm takes MSK_B as input and returns SK_1 for the broker.
4. $\text{KeyGen}_2(MSK, S) \rightarrow SK_2$: Given an attribute set S of a CSP and MSK , this algorithm outputs private key SK for the CSP.
5. $\text{Enc}_1(PP_B, M) \rightarrow C$: The phase 1 encryption is performed in the sender. Given the message M and PP_B , the sender can output a ciphertext C that will be delivered to the broker.
6. $\text{Enc}_2(PP, C, SK_1, A) \rightarrow C^*$: The phase 2 encryption is performed in the broker. Given the ciphertext C , PP and LSSS access structure A , the broker can re-encrypt C to a new ciphertext C^* .
7. $\text{Dec}(PP, SK_2, C^*) \rightarrow \{M, \perp\}$: The decryption is performed in the CSP. If S satisfies A , the decryption algorithm returns M ; otherwise, \perp .
8. $\text{OpenDec}(PP, SK_2, C^*, M) \rightarrow P_D$: The algorithm is used to release evidence for proving that C^* is the encryption result from M .
9. $\text{DenSetup}(1^\lambda) \rightarrow (PP, MSK, PK)$: This is the deniable version of Setup algorithm. Except for PP and MSK , the algorithm also generates the system public information PK . PK is known by all sensors and CSPs, and is kept secret from outsiders. Note that the broker is not in charge of the deniable part so PK is also kept secret from the broker.
10. $\text{DenKeyGen}_2(MSK, PK, S) \rightarrow (SK_2, SK'_2)$: This is the deniable version of KeyGen algorithm. Except for SK_2 , which is derived from the normal function, the algorithm also returns a fake key SK'_2 , which is used later to generate fake proof.
11. $\text{DenEnc}_1(PP_B, PK, M, M', A) \rightarrow C'$: Except for the inputs of the normal encryption algorithm, this deniable encryption algorithm requires public key PK and a pre-defined fake message M' . The output ciphertext C' must be indistinguishable from the output of Enc .
12. $\text{DenOpenDec}(PP, SK, SK'_2, C^*, M') \rightarrow P'_D$: Compared to the normal version algorithm, the algorithm takes additional inputs FK and M' . The output is evidence that C^* is from M' instead of M .

Note that the broker cannot see the message in plaintext form so will not participate in the open process. Therefore, there is no DenEnc_2 and DenKeyGen_1 . Because of the storage constraint issue, the sensor stores no historical data and the deniability is only designed for the CSP. For this reason, there are no OpenEnc and DenOpenEnc algorithms.

For each algorithm, we use Table 2 to point out which entity should run it.

Table 2. List all entities with their own algorithms.

Role	Algorithms
Trusted Key Server	Setup , Setup_B , KeyGen_1 , KeyGen_2 , DenSetup , DenKeyGen_2
CSP	Dec , OpenDec , DenOpenDec
Broker	Enc_2
Sensor	Enc_1 , DenEnc_1

The following properties are required in the proposed scheme:

1. **Security:** the outputs of Enc_1 and Enc_2 must be proved to be confidential under the security model, which is described in Section 3.3. The first protects the segment between the sensor and the broker, whereas the latter protects the segment between the broker and the CSP. The security of Enc_1 also maintains secrecy from the broker. Note that here we do not mention the security of DenEnc_1 because the outputs of Enc_1 and DenEnc_1 must be indistinguishable. If one is secure and the other is not, this will make these two algorithms distinguishable. So, here, the indistinguishability proof also implies the security proof.
2. **Deniability:** the proposed scheme is a receiver-DE scheme. That is, given public parameter PP , the two distribution tuples (M, C, P_D) and (M', C', P'_D) are computationally indistinguishable, where M, M' are claimed messages, C, C' are normally and deniably encrypted ciphertexts, respectively, and P_D, P'_D are proofs generated from the normal and deniable open algorithms, respectively. That is, there is no PPT algorithm A for which

$$Adv_A := \left| \begin{array}{l} P[A(PP, (M, C, P_D)) = 1] \\ - P[A(PP, (M', C', P'_D)) = 1] \end{array} \right|$$

is non-negligible. Note that the deniably encrypted ciphertexts are the outputs of Enc_2 since no entity should be able to decrypt the outputs of Enc_1 or DenEnc_1 . Deniability in the sensor is not considered since it makes no sense for a sensor to provide lots of storage for evidence.

3. **Deniable proof consistency:** in the MCCPS scenario, a cloud service may be used by many sensors. Some sensors may use the privacy-preserving encryption scheme and some may not. When releasing a deniable proof of a CSP, the proof should look convincing not only to the sensor that uses the DE algorithms, but also to the sensor that uses the normal encryption algorithms. That is, given a set of ciphertexts \mathbb{C} , including normally encrypted ciphertexts and deniably encrypted ciphertexts, normal proof P_D and deniable proof P'_D , there is no PPT algorithm A for which

$$Adv_A := |P[A(\mathbb{C}, P_D) = 1] - P[A(\mathbb{C}, P'_D) = 1]|$$

is non-negligible.

3.3. Security Model

In this subsection, we define the security model used in this scheme. First, an adversary is given a challenge question and is allowed to query an oracle for some information. The adversary wins the game if it can correctly answer the question. The formal security game is as follows:

- **Setup:** the challenger first runs Setup and outputs PP to the adversary.
- **Phase 1:** the adversary generates queries q_1, \dots, q_m for the challenger. Query q_i can be one of the following two types of queries:
 - **Key query:** the adversary asks the challenger a key for some entity and obtains its private key from the challenger.
 - **Decryption query:** the adversary asks the challenger to decrypt ciphertext C_i and obtains its plaintext.
- **Challenge:** the adversary chooses two plaintexts M_0, M_1 for the challenger. The adversary also provides a challenge condition A^* , which cannot be authorized by the entities used in q_1, \dots, q_m . The challenger randomly chooses one bit $b \in \{0, 1\}$ and encrypts the message via $\text{Enc}(PP, A^*, M_b) \rightarrow C^*$. The challenger sends C^* to the adversary as the challenge ciphertext.
- **Phase 2:** As in Phase 1, the adversary generates queries q_{m+1}, \dots, q_n for the challenger. Query q_i can be one of the following two types of queries:

- Key query: the adversary asks the challenger a key for some entity and obtains its private key from the challenger. Note that the entity cannot be the one who is authorized to decrypt the challenge ciphertext.
- Decryption query: the adversary asks the challenger to decrypt ciphertext C_i and obtains its plaintext. C_i cannot be C^* .
- Guess: The adversary returns guess result $b' \in \{0, 1\}$. The adversary wins if $b' = b$.

The advantage is defined as $|P(b' = b) - \frac{1}{2}|$.

Note that the proposed scheme include two encryption algorithms Enc_1 and Enc_2 . The above definition can be applied to both encryption algorithms (though that there is actually no key can be used to decrypt the output of Enc_1 . So we will prove Enc_1 to be semantic secure in Section 4. Note that semantic security is equal to CPA-security). The entity in Enc_2 can be treated as a set of attributes and the condition can be treated as the policy.

Definition 7. *An encryption scheme is CPA secure if all polynomial-time adversaries have at most a negligible advantage in the above game without any decryption queries.*

Definition 8. *An encryption scheme is CCA secure if all polynomial-time adversaries have at most a negligible advantage in the above game.*

Note that in the privacy-preserving broker-ABE scheme, there are two encryption algorithms, Enc_1 and Enc_2 . Both output ciphertexts must be verified in the security model.

3.4. Assumptions

In this subsection, we describe some non-computational assumptions in this scheme. First, there is no traitor on either the sensor side or the CSP side. DE is a technique in which both the sender and receiver agree to cheat outsiders to protect privacy. If anyone refuses to apply this technique and releases real data to any third party, the whole system will collapse. That is, in this paper, we do not consider the issues of a compromising sensor or the CSP leakage.

Second, according to the definition above, the CSP and the sensor share one semi-public secret PK . This information is shared by every entity that uses the privacy preservation service but is confidential to outsiders. Here, we assume the existence of an authentication mechanism to determine whether a new applicant is a spy. If PK is leaked to an outsider, the outsider is then enabled to answer whether a released proof is normal or deniable by generating a deniable ciphertext. So, we must require PK to be unknown to adversaries.

Third, we assume that the broker will honestly re-encrypt ciphertexts. In this work, we assume the broker is semi-trusted which means the broker cannot see the data content. However, the broker is responsible for embedding the policy checking mechanism into the ciphertexts according to the sensor types. Although the broker cannot decrypt the ciphertexts, it may embed the wrong policy and launch other attacks, e.g., denial-of-service. In this work, we assume the broker is honest-but-curious.

3.5. Construction

The proposed scheme, which is based on Waters's scheme [14], is as follows.

1. $\text{Setup}(1^\lambda) \rightarrow (PP, MSK)$: this algorithm generates a bilinear group \mathbb{G} of order $N = p_1 p_2 p_3$, where p_1, p_2, p_3 are distinct primes with the bilinear map function $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$. Note that \mathbb{G}_T is also order N . \mathbb{G} can be separated into three orthogonal subgroups $\mathbb{G}_{p_1}, \mathbb{G}_{p_2}, \mathbb{G}_{p_3}$. This algorithm

picks three generators g_1, g_2, g_3 for $\mathbb{G}_{p_1}, \mathbb{G}_{p_2}, \mathbb{G}_{p_3}$ respectively. The algorithm also randomly picks $a, \beta \in \mathbb{Z}_N$ and $\alpha_1 \in \mathbb{Z}_{p_1}$ and chooses a hash function $H_1 : \{0, 1\}^* \rightarrow \mathbb{G}_{p_1}$. The output will be:

$$PP = \{\mathbb{G}, e, H_1, g_1, g_2, g_1^a, e(g_1, g_1)^{\alpha_1}, e(g_1, g_1)^\beta\},$$

$$MSK = \{\alpha_1, \beta\}.$$

2. $\text{Setup}_B(PP, MSK, ID_B) \rightarrow (PP_B, MSK_B)$: taking PP and MSK as inputs, the algorithm generates a new public information and a new master key in a broker domain. ID_B is the broker identity and acts an index for recording the secret of a broker. The algorithm first randomly picks $\alpha_2 \in \mathbb{Z}_{p_2}$. The output will be:

$$PP_B = \{\mathbb{G}, e, H_1, g_1, g_2, g_1^a, e(g_1, g_1)^\beta, e(g_1, g_1)^{\alpha_1}, e(g_2, g_2)^{\alpha_2}\},$$

$$MSK_B = \{\alpha_1, \alpha_2, \beta\}.$$

3. $\text{KeyGen}_1(MSK_B) \rightarrow SK_1$: the algorithm generates SK_1 for the broker as follows:

$$SK_1 = \{g_2^{\alpha_2}\}.$$

4. $\text{KeyGen}_2(MSK, S) \rightarrow SK_2$: the algorithm generates SK_2 for the CSP based on its attributes S . It chooses $t \in \mathbb{Z}_N$ randomly and outputs private key SK_2 as follows:

$$SK_2 = \{g_1^{\alpha_1}, g_1^{\beta+at}, g_1^t, \{H_1(x)^t\}_{\forall x \in S}\}$$

$$= \{K, K^*, L, \{K_x\}_{\forall x \in S}\}.$$

Note that, in this system, each CSP will share the same K .

5. $\text{Enc}_1(PP_B, M) \rightarrow C$: given message M , the algorithm randomly picks $s \in \mathbb{Z}_N$. Then, the algorithm sets up a one-way hash function H . Note that hash function H can be any kind of one-way function, including pseudo random permutation functions. Next, the algorithm flips two coins b_0, b_1 and selects two random strings t_0, t_1 . The output ciphertext C will be:

$$C = \{A_0, A_1, B, H, t_0, t_1, V\},$$

where,

$$A_{b_0} = M \cdot e(g_1, g_1)^{\alpha_1 s} e(g_2, g_2)^{\alpha_2 s},$$

$$A_{1-b_0} \stackrel{R}{\leftarrow} \mathbb{G}_T,$$

$$B = (g_1 g_2)^s,$$

$$V = H(M, t_{b_1})$$

$$\neq H(A_{1-b_0} \cdot e(g_1, g_1)^{-\alpha_1 s} e(g_2, g_2)^{-\alpha_2 s}, t_{1-b_1}).$$

6. $\text{Enc}_2(PP, C, SK_1, \mathcal{A}) \rightarrow C^*$: given a ciphertext C and an LSSS access structure \mathcal{A} , the algorithm first removes the \mathbb{G}_{p_2} part in A_0 and A_1 as follows:

$$A'_b = A_b \cdot e(B, g_2^{\alpha_2})^{-1} = M \cdot e(g_1, g_1)^{\alpha_1 s}, \forall b = \{0, 1\}.$$

Let $\mathcal{A} = (\mathcal{M}, \rho)$ where \mathcal{M} is a $l \times n$ matrix and ρ is a mapping function from $\{1, \dots, l\}$ to the attribute field. The algorithm randomly generates two vectors $\vec{v} = (s^*, y_2, \dots, y_n) \in \mathbb{Z}_N^n$ and $\vec{r} = (r_1, \dots, r_l) \in \mathbb{Z}_N^l$. Then it calculates $\lambda_i = \vec{v} \cdot \mathcal{M}_i, \forall i \in \{1, \dots, l\}$. The output ciphertext C' will be as follows:

$$C^* = \{ A_0^*, A_1^*, B, B^*, (C_1, D_1), \dots, (C_l, D_l), H, t_0, t_1, V \},$$

where

$$\begin{aligned} A_b^* &= A_b' \cdot e(g_1, g_1)^{\beta s^*}, \forall b \in \{0, 1\}, \\ B^* &= g_1^{s^*}, \\ C_i &= g_1^{a\lambda_i} H_1(\rho(i))^{-r_i}, D_i = g_1^{r_i}, i = 1 \dots l, \end{aligned}$$

B, H, t_0, t_1, V are directly derived from C and are not changed. \mathcal{A} is also appended to C^* .

7. $\text{Dec}(PP, SK_2, C^*) \rightarrow \{M, \perp\}$: to decrypt ciphertext C^* for access structure \mathcal{A} , the algorithm first checks if attribute set S of SK_2 satisfies \mathcal{A} . Suppose S satisfies \mathcal{A} and let $I \subset \{1, 2, \dots, l\}$ be defined as $I = \{i : \rho(i) \in S\}$. The algorithm is able to find a set of constants $\{w \in \mathbb{Z}_N\}$ such that $\sum_{i \in I} w_i \lambda_i = s^*$. This algorithm computes M_0, M_1 as follows:

$$M_b = A_b^* \cdot \frac{\prod_{i \in I} (e(C_i, L) e(D_i, K_{\rho(i)}))^{w_i}}{e(B, K) e(B^*, K^*)}, \forall b \in \{0, 1\}.$$

The algorithm then checks tag V as follows:

$$v_{i,j} = H(M_i, t_j), \forall i, j \in \{0, 1\}.$$

If $v_{i,j}$ is equal to V , then M_i is a true message. Otherwise, this algorithm returns \perp .

8. $\text{OpenDec}(PP, SK_2, C^*, M) \rightarrow SK_2$: to show that C^* is encrypted from M , the algorithm simply returns SK_2 .
9. $\text{DenSetup}(1^\lambda) \rightarrow (PP, MSK, PK)$: (PP, MSK) tuple is generated as Setup . The algorithm also randomly picks $\alpha_3 \in \mathbb{Z}_{p_3}$. PK is generated as follows:

$$PK = \{g_3, e(g_3, g_3)^{\alpha_3}\}.$$

Note that PK is kept secret from outsiders, including the broker, since it does not participate in the deniable encryption process. For the same reason, there is no DenSetup_B algorithm.

10. $\text{DenKeyGen}_2(MSK, PK, S) \rightarrow (SK_2, SK_2')$: SK_2 is derived from KeyGen and SK_2' is generated as follows:

$$\begin{aligned} SK_2' &= \{g_1^{\alpha_1} g_3^{\alpha_3}, g_1^{\beta + \alpha t}, g_1^t, \{H_1(x)^t\}_{\forall x \in S}\} \\ &= \{K', K^*, L, \{K_x\}_{\forall x \in S}\}. \end{aligned}$$

11. $\text{DenEnc}_1(PP, PK, M, M') \rightarrow C$: the algorithm needs one more input M' which is the pre-determined fake message (that is, a fake message is given by users. If the fake message is obviously far from the normal message, it is the user's responsibility rather than that of the proposed scheme). This algorithm first runs Enc and gets $\lambda_i, \forall i \in \{1, \dots, l\}$ and b_0, b_1 . Next, this algorithm sets up a chameleon hash function $CH(\cdot, \cdot)$. The output deniable ciphertext C will be:

$$C' = \{A_0, A_1, B, CH, t_0, t_1, V\},$$

where,

$$\begin{aligned} A_{b_0} &= M \cdot e(g_1, g_1)^{\alpha_1 s} e(g_2, g_2)^{\alpha_2 s}, \\ A_{1-b_0} &= M' \cdot e(g_1, g_1)^{\alpha_1 s} e(g_2, g_2)^{\alpha_2 s} e(g_3, g_3)^{\alpha_3 s}, \\ B &= (g_1 g_2 g_3)^s, \\ V &= CH(M, t_{b_1}) = CH(M', t_{1-b_1}). \end{aligned}$$

Since a chameleon hash function is a trapdoor pseudo-random permutation function, it is easy to find t_{b_1}, t_{1-b_1} to generate a collision, which implies that M and M' can both be valid in decryption.

12. $\text{DenOpenDec}(PP, SK_2', FK, C^*, M') \rightarrow SK_2'$: to show that C^* is encrypted from the fake message M' , the algorithm simply returns SK_2' .

In this construction, the sensor is in charge of Enc_1 and the broker is in charge of Enc_2 for embedding policy re-encryption. There is no deniable encryption feature for Enc_2 since the broker knows nothing about the message, which makes it impossible for the broker to create fake messages. There is only one decryption algorithm because outsiders may run Dec with keys derived from CSPs to verify message contents. Note that with the correct key, no matter which encryption algorithm is used, the CSP can always get the correct message. With respect to the fake key, the outsider will get fake messages if they are deniably encrypted. However, if a message is normally encrypted, the real message can also be derived via the fake key.

3.6. Correctness

In this subsection, we check the correctness of this construction. Since only the receiver can correctly decrypt the ciphertext, the correction check is only invoked for ciphertext C and SK_2 . Below, we present four scenarios and their corresponding checks.

1. When using the normal key SK_2 to decrypt a normally encrypted ciphertext C , the decryption process will be as follows:

$$\begin{aligned} & \frac{\prod_{i \in I} (e(C_i, L) e(D_i, K_{\rho(i)}))^{w_i}}{e(B, K) e(B^*, K^*)} \\ = & \frac{\prod_{i \in I} (e(g_1^{a\lambda_i}, g_1^t))^{w_i}}{e((g_1 g_2)^s, g_1^{\alpha_1}) e(g_1^{s^*}, g_1^{\beta+at})} \\ = & \frac{e(g_1, g_1)^{at \sum_{i \in I} \lambda_i w_i}}{e(g_1, g_1)^{s\alpha_1 + s^*(\beta+at)}} \\ = & e(g_1, g_1)^{-\alpha_1 s - \beta s^*}. \end{aligned}$$

With the hash function H and V , the receiver can correctly get the message M .

2. When using the normal key SK_2 to decrypt a deniably encrypted ciphertext C , the decryption process will be as follows:

$$\begin{aligned} & \frac{\prod_{i \in I} (e(C_i, L) e(D_i, K_{\rho(i)}))^{w_i}}{e(B, K) e(B^*, K^*)} \\ = & \frac{\prod_{i \in I} (e(g_1^{a\lambda_i}, g_1^t))^{w_i}}{e((g_1 g_2 g_3)^s, g_1^{\alpha_1}) e(g_1^{s^*}, g_1^{\beta+at})} \\ = & \frac{e(g_1, g_1)^{at \sum_{i \in I} \lambda_i w_i}}{e(g_1, g_1)^{s\alpha_1 + s^*(\beta+at)}} \\ = & e(g_1, g_1)^{-\alpha_1 s - \beta s^*}. \end{aligned}$$

With the chameleon hash function CH and V , the receiver can correctly get the message M .

3. When using the deniable key SK'_2 to decrypt a deniably encrypted ciphertext C , the decryption process will be as follows:

$$\begin{aligned} & \frac{\prod_{i \in I} (e(C_i, L) e(D_i, K_{\rho(i)}))^{w_i}}{e(B, K') e(B^*, K^*)} \\ = & \frac{\prod_{i \in I} (e(g_1^{a\lambda_i}, g_1^t))^{w_i}}{e((g_1 g_2 g_3)^s, g_1^{\alpha_1} g_3^{\alpha_3}) e(g_1^{s^*}, g_1^{\beta+at})} \\ = & \frac{e(g_1, g_1)^{at \sum_{i \in I} \lambda_i w_i}}{e(g_1, g_1)^{s\alpha_1 + s^*(\beta+at)} e(g_3, g_3)^{\alpha_3 s}} \\ = & e(g_1, g_1)^{-\alpha_1 s - \beta s^*} e(g_3, g_3)^{-\alpha_3 s}. \end{aligned}$$

With the chameleon hash function CH and V , the receiver can correctly get message M' rather than the real message M .

4. When using the deniable key SK'_2 to decrypt a normally encrypted ciphertext C , the decryption process will be as follows:

$$\begin{aligned}
 & \frac{\prod_{i \in I} (e(C_i, L) e(D_i, K_{\rho(i)}))^{w_i}}{e(B, K') e(B^*, K^*)} \\
 = & \frac{\prod_{i \in I} (e(g_1^{a\lambda_i}, g_1^t))^{w_i}}{e((g_1 g_2)^s, g_1^{\alpha_1} g_3^{\alpha_3}) e(g_1^{s^*}, g_1^{\beta+at})} \\
 = & \frac{e(g_1, g_1)^{at \prod_{i \in I} \lambda_i w_i}}{e(g_1, g_1)^{s\alpha_1 + s^*(\beta+at)}} \\
 = & e(g_1, g_1)^{-\alpha_1 s - \beta s^*}.
 \end{aligned}$$

With the hash function H and V , the receiver can correctly get the message M .

Based on the above checking process, this scheme has two important properties. First, if a CSP uses the normal key, whether the sensor normally or deniably encrypts the data, the CSP can always derive the correct message. Second, with the deniable key, the CSP can get the correct message even if the sensor normally encrypts the data. That is, if the CSP supports both privacy-preserving and normal sensors, an outsider cannot challenge the key released from the CSP since the key looks normal, even when the sensor does not participate in the deniability process.

Theorem 2. *The privacy-preserving broker ABE scheme is deniable proof consistent.*

Proof of Theorem 2. In this scheme, the CSP key is treated as proof for the CSP claim. Note that the key is the most immediate proof available. As shown above, both SK_2 and SK'_2 can be applied to correctly decrypt ciphertexts. That is, a ciphertext can be decrypted to a meaningful message, which may be real data or a pre-determined fake message. Given a set of ciphertexts \mathbb{C} where its element may be normally or deniably encrypted, anyone who can differentiate between (\mathbb{C}, SK_2) and (\mathbb{C}, SK'_2) can also differentiate between real and fake data. In other words, these two tuples are indistinguishable. \square

3.7. Implementation Issues

In this subsection, we will describe some implementation issues. First, we focus on the performance issue about composite order bilinear groups. Then, we discuss how to run Enc_1 and Enc_2 on the sensor and the broker efficiently. Note that we do not care about the CSP performance issue because the CSP resource is generally much greater than the sensor and the broker.

In our construction, we use composite order bilinear groups because of their canceling property. However, the composite order bilinear group operations are prolonged. Table 3 shows the comparison result of the bilinear operation on a raspberry PI3, where each prime is 512 bits. It shows that the required operation time grows exponentially. Therefore, some researches suggest that the composite order group is unlikely applicable [36,37].

Table 3. Pairing operation time.

Order	Time (ms)	Simulation Time (ms)
p_1	69.128	-
$p_1 p_2$	418.222	148.912
$p_1 p_2 p_3$	1321.734	222.367
$p_1 p_2 p_3 p_4$	2999.380	296.672
$p_1 p_2 p_3 p_4 p_5$	5676.130	373.077

To solve this problem, in our implementation we used a prime order group to simulate composite order group behavior, as described in Lewko's work [34]. The idea is based on dual orthonormal bases and the subspace assumption. Each prime subgroup is simulated as a distinct orthonormal base. By the orthogonal property, the bilinear operation will be canceled between different prime subgroups. In our implementation, each base contains three prime order group elements. So the bilinear operation is only around three times than the prime order bilinear operation. Table 3 lists the required times of simulation-based pairing operations. We can find that the performance is greatly enhanced.

Some may claim that even with this simulation technique, the bilinear operation is also expensive to a lightweight node like sensors. We emphasize that in our implementation, there is no bilinear operation in the sensor node. According to Table 2, the sensor node only runs Enc and DenEnc. In these two algorithms, $e(g_1, g_1)^{\alpha_1}$, $e(g_2, g_2)^{\alpha_2}$ and $e(g_3, g_3)^{\alpha_3}$ are all public information and can be pre-computed. So the sensor does not need to run any bilinear operations.

Now we focus on the broker part. The broker needs to run one bilinear operation and this operation can be accelerated through the simulation technique. Since the broker is the entity which embeds policies on the ciphertext, some may doubt it takes lots of computational resources and memory, especially with the composite order group. We emphasize that this is not true in our implementation. In this work, we use the composite order group for its canceling property, and we can simulate this property through the prime order group. That is, all elements are in the prime order group. So in Enc, except for one pairing operation, the only operations are the power operation and the multiplication operation in the prime order group. Moreover, the canceling property is used only between B, SK_1, K in SK_2 and K' in SK'_2 . So we only need to use orthonormal bases to simulate these elements and the memory cost will not increase too much.

4. Evaluation

In this section, we evaluate the proposed scheme in three aspects, security, deniability, and performance.

4.1. Security Proof

In this subsection, we will prove the security of our scheme. For simplicity, though we simulate the composite order group with a prime order group in our implementation, here we prove the security with the composite order group. Since this scheme is a re-encryption scheme, it is necessary to show that both Enc₁ and Enc₂ are secure. Although there is also another encryption algorithm DenEnc₁, we need not prove its security because the outputs of Enc₁ and DenEnc₁ are indistinguishable. The indistinguishability proof is presented in the next subsection. If the output of Enc₁ is secure, but the output of DenEnc₁ is not, their indistinguishability will be broken. That is, the indistinguishability proof ensures the security of DenEnc₁ while Enc₁ is secure. First, we show that Enc₁ is secure under the discrete logarithm assumption. Then, we prove that Enc₂ is secure in the CPA model. We discuss the Enc₂ CCA-security issue in Section 5.

Lemma 1. *Enc₁ is semantic secure if the discrete logarithm problem is hard.*

Proof of Lemma 1. To prove the security of Enc₁, the focus should be on $e(g_1, g_1)^{\alpha_1 s} \cdot e(g_2, g_2)^{\alpha_2 s}$ and $(g_1 g_2)^s$. The rest of the parts are verification tags through a one-way function, so they have no information about the message. Therefore, the security proof is the answer to the following question: given $g_1, g_2, (g_1 g_2)^s, e(g_1, g_1)^{\alpha_1}, e(g_2, g_2)^{\alpha_2}$, is finding $e(g_1, g_1)^{\alpha_1 s} \cdot e(g_2, g_2)^{\alpha_2 s}$ easy or not? Since s can only be derived from $(g_1 g_2)^s$, this question is equivalent to a discrete logarithm problem. Although $g_1 g_2$ is a group element that supports the bilinear map operation, there is no α_1, α_2 information in $\mathbb{G}_{p_1 p_2}$ unless α_1, α_2 can be derived from $e(g_1, g_1)^{\alpha_1}, e(g_2, g_2)^{\alpha_2}$, which is also a discrete logarithm problem. So if the discrete logarithm problem is hard, getting $e(g_1, g_1)^{\alpha_1 s} \cdot e(g_2, g_2)^{\alpha_2 s}$ is also a hard

problem. Therefore, $e(g_1, g_1)^{\alpha_1 s} \cdot e(g_2, g_2)^{\alpha_2 s}$ can be treated as a random element in \mathbb{G}_T . So, Enc_1 is semantic secure. \square

The above proof demonstrates that the ciphertext encrypted by the sensor is semantic secure. Because semantic security is equal to CPA-security, Enc_1 is CPA-secure. That is, the message transferred from the sensor to the broker is secure, even secure from the broker. The next step is to prove Enc_2 is CPA-secure, which implies that the communication between the broker and the CSP is secure. When considering Enc_2 , we skip the Setup_B step and directly integrate Enc_1 into Enc_2 , since these two steps belong to their own broker domain. The attacker now is outside the broker domain and has the system's public information from Setup . The attacker does not participate in the Enc_1 process.

Lemma 2. *Enc₂ is CPA-secure if Waters CP-ABE scheme is CPA-secure.*

Proof of Lemma 2. Let \mathcal{A} be an adversary that breaks the above deniable CP-ABE scheme. An algorithm \mathcal{B} that can break Waters CP-ABE scheme can be constructed as follows. \mathcal{B} is given public parameters through the Waters CP-ABE scheme's Setup algorithm from challenger \mathcal{X} as follows:

$$PP_W = \{g_1, g_1^a, e(g_1, g_1)^\beta\},$$

with prime number p_1 , \mathbb{G}_{p_1} , $e(\cdot, \cdot)$ and $H_1(\cdot)$. For convenience, we use a suffix to represent different subgroups in our proof. Algorithm \mathcal{B} proceeds as follows.

- **Setup:** \mathcal{B} first picks two different prime numbers p_2 and p_3 . Next, \mathcal{B} generates group \mathbb{G} with order $N = p_1 p_2 p_3$. Note that the subgroup with p_1 order in \mathbb{G} should be the same as \mathbb{G}_{p_1} . \mathcal{B} then randomly picks $\alpha_1 \in \mathbb{Z}_{p_1}$ and picks a generator g_2 for \mathbb{G}_{p_2} . \mathcal{B} then shows the following to \mathcal{A} :

$$PP = \{\mathbb{G}, e, H_1, g_1, g_2, g_1^a, e(g_1, g_1)^{\alpha_1}, e(g_1, g_1)^\beta\}.$$

Note that $e(\cdot, \cdot)$ and $H_1(\cdot)$ are the same as the given function from \mathcal{X} .

- **Phase 1:** when \mathcal{B} receives a key generation query for attribute set S from \mathcal{A} , \mathcal{B} simply relays the query to \mathcal{X} and obtains SK_W as follows:

$$\begin{aligned} SK_W &= \{g_1^{\beta+at}, g_1^t, \{H_1(x)^t\}_{\forall x \in S}\} \\ &= \{K^*, L, \{K_x\}_{\forall x \in S}\}. \end{aligned}$$

\mathcal{B} then generates $g_1^{\alpha_1}$ and outputs SK_2 to \mathcal{X} as follows:

$$\begin{aligned} SK_2 &= \{g_1^{\alpha_1}, g_1^{\beta+at}, g_1^t, \{H_1(x)^t\}_{\forall x \in S}\} \\ &= \{K, K^*, L, \{K_x\}_{\forall x \in S}\}. \end{aligned}$$

- **Challenge:** \mathcal{A} outputs two messages M_0, M_1 with access structure (\mathcal{M}, ρ) to \mathcal{B} , and \mathcal{B} directly relays M_0, M_1 and (\mathcal{M}, ρ) to \mathcal{X} as the challenge and obtains the following from \mathcal{X} .

$$C_W = \{A^*, B^*, (C_1, D_1), \dots, (C_l, D_l)\},$$

where

$$\begin{aligned} A^* &= M_b \cdot e(g_1, g_1)^{\beta s^*}, b \in \{0, 1\} \\ B^* &= g_1^{s^*}, \\ C_i &= g_1^{a \lambda_i} H_1(\rho(i))^{-r_i}, D_i = g_1^{r_i}, i = 1 \dots l. \end{aligned}$$

M_b is chosen by \mathcal{X} . \mathcal{B} setups a chameleon hash function CH and randomly picks b_1, b_2 from $\{0, 1\}$. \mathcal{B} also randomly picks $s \in \mathbb{Z}_N$. Finally, \mathcal{B} outputs C to \mathcal{A} as follows:

$$C = \{ A_0^*, A_1^*, B, B^*, (C_1, D_1), \dots, (C_l, D_l), CH, t_0, t_1, V \},$$

where,

$$\begin{aligned} A_{b_1}^* &= A^* \cdot e(g_1, g_1)^{\alpha s}, \\ A_{1-b_1}^* &\stackrel{R}{\leftarrow} \mathbb{G}_T, \\ B &= g_1^s, \\ V &= CH(M_0, t_{b_2}) = CH(M_1, t_{1-b_2}). \end{aligned}$$

Note that a chameleon hash function is used instead of a common hash function. To \mathcal{A} , without the trapdoor, a chameleon hash function is simply a one-way function. Here we ensure that the verification tag V is valid for both M_0 and M_1 , so the verification process will lease no information to \mathcal{A} .

- Phase 2: the query and response process is the same as that in Phase 1.
- Guess: finally, \mathcal{A} outputs guess b' to \mathcal{B} and \mathcal{B} forwards guess b' to \mathcal{X} .

If \mathcal{A} achieves a non-negligible advantage against the proposed encryption scheme, \mathcal{B} can use the output of \mathcal{A} to also achieve a non-negligible advantage against the Waters CP-ABE scheme in the CPA model. \square

According to Lemma 1, Lemma 2 and Theorem 1, we can derive the following theorem.

Theorem 3. *Suppose the discrete logarithm assumption and the decisional q -parallel BDHE assumption hold, then no polynomial time adversary can selectively break the proposed encryption system in the CPA-model.*

4.2. Deniability Proof

To prove the deniability of this scheme, we must show that the Enc_1 output C and the DenEnc_1 output C' , the KeyGen_2 output SK_2 and the DenKeyGen_2 SK_2' are indistinguishable respectively. Here we do not consider Enc_2 and KeyGen_1 because the broker does not participate in the deniability process.

Lemma 3. *Under the general subgroup decision assumption, normal ciphertext C and deniable ciphertext C' are indistinguishable.*

Proof of Lemma 3. Suppose there exists PPT attacker \mathcal{A} who achieves a non-negligible advantage in distinguishing the deniable ciphertext C' from the normal ciphertext C of the proposed scheme. A PPT algorithm \mathcal{B} can be constructed that also has a non-negligible advantage against the general subgroup decision assumption.

The difference between C and C' is the existence of the g_3 element in B . Note that there is no difference in A_{1-b_0} since in the normal encryption, A_{1-b_0} is randomly selected from \mathbb{G}_T and undoubtedly it is possible to include the $e(g_3, g_3)$ element. There is also no difference about the verification parts because CH can be treated as a common hash function if the secret key is not released. Now, given an element T to be determined if T belongs to $\mathbb{G}_{p_1 p_2}$ or $\mathbb{G}_{p_1 p_2 p_3}$. \mathcal{B} can construct a ciphertext C^* for the two messages M and M' as follows:

$$C^* = \{ A_0, A_1, B, CH, t_0, t_1, V \},$$

where,

$$\begin{aligned} A_{b_0} &= M \cdot e(g_1, T)^{\alpha_1} e(g_2, T)^{\alpha_2}, \\ A_{1-b_0} &= M' \cdot e(g_1, T)^{\alpha_1} e(g_2, T)^{\alpha_2} e(g_3, T)^{\alpha_3}, \\ B &= T, \\ V &= CH(M, t_{b_1}) = CH(M', t_{1-b_1}). \end{aligned}$$

Then \mathcal{B} forwards C^* to \mathcal{A} . If \mathcal{A} says it is a normal ciphertext, then T is in $\mathbb{G}_{p_1 p_2}$. If \mathcal{A} says it is a deniable ciphertext, then T is in $\mathbb{G}_{p_1 p_2 p_3}$.

In the public information PP and PP_B , there is no g_3 element. Given the general subgroup decision assumption, it is hard to determine if a given element belongs to $\mathbb{G}_{p_1 p_2}$ or $\mathbb{G}_{p_1 p_2 p_3}$. So C and C' should be indistinguishable. \square

Next, we check the deniability of the released proof.

Lemma 4. *Under the general subgroup decision assumption, normal decryption proof SK_2 and deniable decryption proof SK'_2 are indistinguishable.*

Proof of Lemma 4. Suppose there exists PPT attacker \mathcal{A} who achieves a non-negligible advantage in distinguishing the deniable proof SK'_2 from the normal proof SK_2 of the proposed scheme. A PPT algorithm \mathcal{B} can be constructed that also has a non-negligible advantage against the general subgroup decision assumption.

In this scheme, SK_2 or SK'_2 used as the receiver proof. The only difference between SK_2 and SK'_2 is the existence of element g_3 in K . Now given an element T to be determined if T belongs to \mathbb{G}_{p_1} or $\mathbb{G}_{p_1 p_3}$. \mathcal{B} can construct a secret key SK_2^* as follows:

$$\begin{aligned} SK_2^* &= \{T^{\alpha_1}, g_1^{\beta+at}, g_1^t, \{H_1(x)^t\}_{\forall x \in S}\} \\ &= \{K, K^*, L, \{K_x\}_{\forall x \in S}\}. \end{aligned}$$

Then \mathcal{B} forwards SK_2^* to \mathcal{A} . If \mathcal{A} says it is a normal key, then T is in \mathbb{G}_{p_1} . If \mathcal{A} says it is a deniable key, then T is in $\mathbb{G}_{p_1 p_3}$.

In the public information PP and PP_B , there is no g_3 element. Given the general subgroup decision assumption, it is hard to determine if a given element belongs to \mathbb{G}_{p_1} or $\mathbb{G}_{p_1 p_3}$. So SK_2 and SK'_2 should be indistinguishable. \square

From Lemmas 3 and 4, the proposed scheme is a receiver-deniable encryption scheme, which means that outsiders cannot challenge fake keys provided by CSPs.

4.3. Performance Evaluation

To evaluate the performance of the proposed scheme, we used a Raspberry Pi 3 computer as the sensor platform and a desktop PC as the broker platform. The PC is equipped with i7-7700 CPU and 16 GB of memory (some may doubt that this PC is too good to be a broker. When considering the fog/edge trend [38], the edge entity is likely to become more and more powerful). In the implementation, we set each prime size to 512 bits, which is equal to 256 bits of security [39]. So, while the composite group order size is 1536 bits, the security level is still 512 bits. We focus on the sensor performance and the broker performance here. The CSP performance is not our concern since the CSP is generally much more powerful.

First, we check the required encryption time in the sensor. We compared our work with Waters scheme [14], which is our base scheme, Yao scheme [23], which is not a pairing scheme, Li scheme [27] (the outsourcing server is a computer with an INTEL i7 CPU. Li et al. separated the encryption work into the data owner and the outsourcing server. Here we combine them together to evaluate the computational cost), which is an outsourcing scheme, and Fischer scheme [28] (Fischer scheme is a KP-ABE scheme), which is a proxy-based scheme like us. We implement our approach in both the composite order group and the prime order group. Figure 4 shows the Enc_1 process times. We can

find that the attribute number affects the encryption a lot. Only the Fischer scheme and our scheme can keep the same encryption time when the attribute number grows. The main reason is that the attribute related part of the encryption process is outsourced to a powerful entity, and therefore, it will not be counted to the sensor side. So we believe that the broker ABE is suitable for the IoT scenario. Besides, we can find that our prime order simulation implementation improves the encryption process. In our experiment, the exponential operation of \mathbb{G} is around 33.66 ms when $N = p_1$ and is around 463.95 ms when $N = p_1 p_2 p_3$. Therefore, we believe that our scheme is affordable for a sensor.

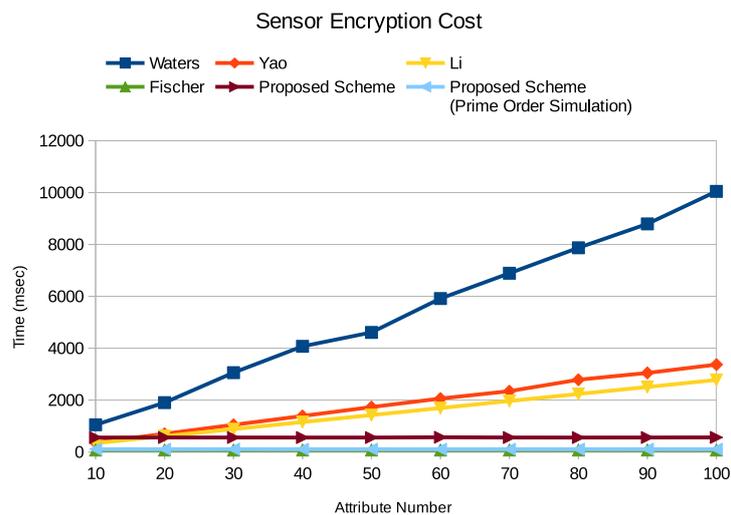


Figure 4. Enc_1 execution time on Raspberry Pi3 which plays the sensor role.

Next, we focused on the transmission cost. We used the ciphertext size to evaluate the transmission cost. Figure 5 is the comparison result. We can find that the ciphertext size has a similar behavior as the encryption time. In most schemes, the ciphertext size grows linear with the attribute number. The ciphertext size of the broker ABE, including our scheme and Fischer scheme, stayed the same since the required attributes are embedded in the broker. Note that the ciphertext size of our scheme was slightly greater than in the Fischer scheme. The reason is that we added a fake element in the ciphertext for data privacy against outside coercion. We believe this is not a problem.

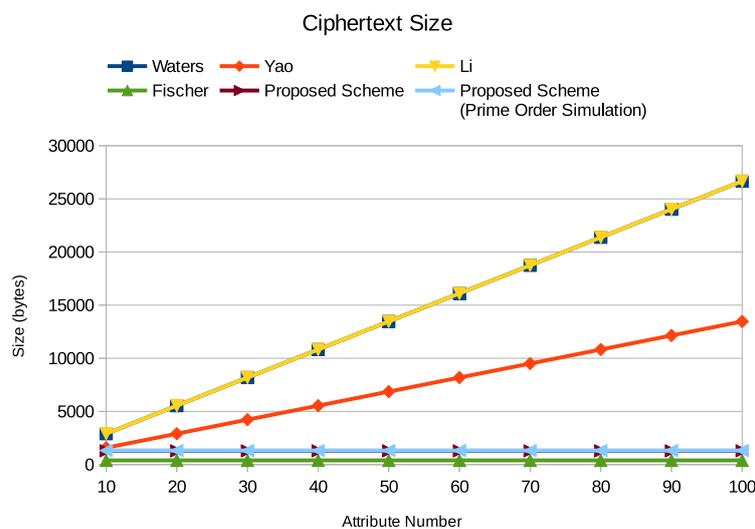


Figure 5. The ciphertext size of Enc_1 .

Then we see the Enc_2 performance on the broker. Generally speaking, the broker is much more powerful than the sensor, so we only compare our scheme with the Waters scheme. Since we want to evaluate the performance, in our experiments, we make all attributes mandatory so the encryption will be the max encryption time. Again, we implement our scheme in both the composite order and the prime order. Moreover, we implement another version with a trade-off between memory and computation. In our scheme, \vec{r} can be precomputed and therefore $H_1(x)^{-r_i}$ and g^{r_i} can also be precomputed, too. For each $x \in S$, we make the broker generates γ random numbers and then prepare a pool of $(H_1(x)^{-r_i}, g^{r_i}), i \in \{1, \dots, \gamma\}$. So there will be $|S|$ pools. Note that one pair in a pool can be reused because there are total $\gamma^{|S|+1}$ combinations among all pools. We make this trade-off because the broker usually has more memory than the sensor. Figure 6 shows the Enc_2 process times. We can see that we greatly decrease the computational loading of the broker.

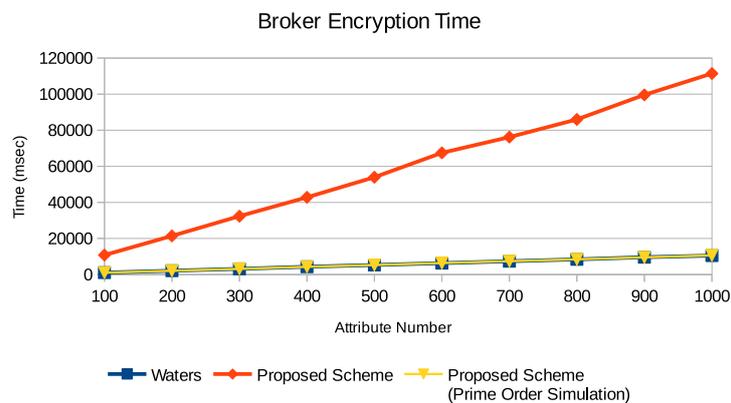


Figure 6. Enc_2 execution time on a PC which plays the broker role.

5. Discussion

In this section, we discuss two topics related to this scheme. The first topic is CCA-security, and the second is regarding Chameleon Hash issues.

5.1. CCA-Security

In Section 4, we proved that the scheme is CPA-secure. Boneh et al. proved that an IND-sID-CPA secure IBE scheme can be transformed into an IND-sID-CCA secure scheme with the help of a one-time signature scheme [40]. Liang et al. [20] applied this concept and used a random oracle to build a re-encryption scheme that is also based on the Waters scheme. It is trivial to apply this same approach to the scheme proposed in this paper. However, since this is a two-step encryption scheme, there will be two signatures, one by the sensor and the other by the broker. The scheme construction is modified as follows. The algorithms that are not mentioned remain the same as the original construction.

1. $\text{Setup}_{\text{cca}}(1^\lambda) \rightarrow (PP, MSK)$: Aside from the original Setup , the algorithm additionally setup a hash function $H_2 : \{0, 1\}^* \rightarrow \mathbb{G}_{p_1}$ and randomly picks $b \in \mathbb{Z}_N$. The algorithm then appends H_2, g_1^b to PP .
2. $\text{Enc}_{1,\text{cca}}(PP_B, M) \rightarrow C_{\text{cca}}$: The sensor first runs the original Enc_1 algorithm and gets $C = \{A_0, A_1, B, H, t_0, t_1, V\}$. Then the algorithm calculates $B_1 = g_1^{bs}$ and the output ciphertext will be as follows:

$$C_{\text{cca}} = \{A_0, A_1, B, B_1, H, t_0, t_1, V, V_1\},$$

where

$$V_1 = H_2(A_0, A_1, B, B_1, H, t_0, t_1, V)^s.$$

3. $\text{Enc}_{2,\text{cca}}(PP, C_{\text{cca}}, SK_1, \mathcal{A}) \rightarrow C_{\text{cca}}^*$: The broker first verifies the following equations:

$$e(B, g_1^b) \stackrel{?}{=} e(B_1, g_1),$$

$$e(V', B_1) \stackrel{?}{=} e(V_1, g_1^b),$$

where

$$V' = H_2(A_0, A_1, B, B_1, H, t_0, t_1, V).$$

If the above two equations do not hold, the algorithm simply drops the C_{cca} and returns. Otherwise, the algorithm runs the original Enc_1 algorithm and gets C^* . Then the algorithm calculates $B_2 = g_1^{bs^*}$ and the output ciphertext will be as follows:

$$C^* = \{ A_0^*, A_1^*, B, B^*, B_2, (C_1, D_1), \dots, (C_l, D_l), \\ H, t_0, t_1, V, V_2 \},$$

where

$$V_1 = H_2 \left(\begin{array}{c} A_0^*, A_1^*, B, B^*, B_2, \\ (C_1, D_1), \dots, (C_l, D_l), \\ H, t_0, t_1, V \end{array} \right)^s.$$

4. $\text{Dec}_{\text{cca}}(PP, SK_2, C_{\text{cca}}^*) \rightarrow \{M, \perp\}$: The CSP first verifies the following two equations:

$$e(B^*, g_1^b) \stackrel{?}{=} e(B_2, g_1),$$

$$e(V', B_2) \stackrel{?}{=} e(V_2, g_1^b),$$

where

$$V' = H_2 \left(\begin{array}{c} A_0^*, A_1^*, B, B^*, B_2, \\ (C_1, D_1), \dots, (C_l, D_l), \\ H, t_0, t_1, V \end{array} \right).$$

If the above two equations do not hold, the algorithm simply returns \perp . Otherwise, it simply runs Dec and outputs the result.

Theorem 4. *The modified scheme is CCA-secure if the original scheme is CPA-secure.*

Proof of Theorem 4. The difference between the original and modified schemes is the existence of two signatures in the broker and CSP. The additional signatures have nothing to do with the confidentiality of the input messages. Since we proved the original scheme to be CPA-secure in Theorem 3, here the proof must focus only on how to answer the decryption query. Since there are two encryption processes in this scheme, we start with the Enc_1 part. Although there is actually no entity that can decrypt the output of Enc_1 in this scheme, we still check the CCA-security of this part. When receiving a decryption query, the oracle proceeds as follows:

1. If $e(B, g_1^b) = e(B_1, g_1)$ and $e(V', B_1) = e(V_1, g_1^b)$ do not hold, the oracle responds \perp .
2. In phase 2, if the queried ciphertext is the same as the challenged ciphertext, the oracle responds \perp .
3. The oracle uses the master secret s to calculate $e(g_1, g_1)^{as}$ and $e(g_2, g_2)^{bs}$ to decrypt the queried ciphertext. The oracle then returns the decryption result to the adversary.

The decryption oracle in the second encryption part is similar. So in both encryption parts, it is easy to build decryption oracles for adversaries. According to the definition in Section 3.3, the modified scheme is thus CCA-secure. \square

5.2. Chameleon Hash Issues

In this scheme, the sensor uses a chameleon hash to build convincing evidence for both the real and fake data. Here, we discuss three issues regarding chameleon hash functions. The first issue relates to the performance of the chameleon hash. The chameleon hash function can be treated as a trapdoor pseudo-random permutation function. That is, the cost of the chameleon hash operation is similar to that of the public key encryption operation. Some may doubt if this expensive operation is suitable for a sensor with constrained computational power. However, in the proposed scheme, the sensor does not need to run the chameleon hash function. Instead, the sensor only needs to make a random string that generates a collision, which is a lightweight operation. In our implementation, we use Krawczyk et al.'s construction [35] as the chameleon hash function. Given m, m', r , the forgery string r' can be derived as follows:

$$r' = \frac{m - m'}{x} + r.$$

So this operation is simple and affordable for a sensor. The chameleon hash function is used only in the broker and the CSP which are more powerful than the sensor.

The second issue is that one collision in a chameleon hash function implies the release of the trapdoor. In the MCCPS scenario, this would become a big problem. In many scenarios, the sensors collect data only from a limited range. For example, a thermometer sensor may only collect data from 35 °C to 42 °C. The attacker may try every possible combination to find the collision pair and get the trapdoor of the chameleon hash. When the trapdoor is leaked, the attacker can get real data from fake data by calculating the collision. To solve this problem, in our implementation, we have added a timestamp on each piece of data to enlarge the data range and avoid the brute force attack.

The last issue to address is that, in the proposed scheme, the chameleon hash can differ on each transmission. However, this is impractical since the chameleon hash setup operation is computationally expensive. So a chameleon hash function will be used many times and it is necessary to consider the need to update the chameleon hash function. In our implementation, to ease the sensor burden, the system owner must prepare a chameleon hash function pool. When the lifetime of a chameleon hash is almost at an end, the system owner dispatches a new chameleon hash function with its corresponding trapdoor to the sensor. Note that the system owner can generate chameleon hash functions to the pool in advance. Besides, since a chameleon hash function can be independent in each transmission, there is no synchronization issue.

6. Conclusions and Future Works

In this paper, we proposed a privacy-preserving broker-ABE for MCCPS scenarios, in which the burden on the edge sensors is light. In this scheme, the broker is responsible for most of the encryption work. The evaluation results show the scheme's performance is acceptable. Moreover, the data stored in clouds can be kept private even against outside coercion.

Future work will address broker security. Although in this paper, the broker is semi-trusted and cannot see the content of ciphertexts, the proposed scheme cannot ensure protection in cases where the broker and CSP are collaborating. Moreover, the broker may embed false policies and try to release data to those who are authorized to access data. So the next step is to require the broker to embed policies based on some unforgeable secrets generated by the sensors.

Author Contributions: Conceptualization, P.-W.C.; methodology, P.-W.C. and M.-H.W.; software, P.-W.C.; validation, P.-W.C. and M.-H.W.; writing—original draft, P.-W.C.; writing—review and editing, P.-W.C. and M.-H.W.

Funding: This research was funded by Ministry of Science and Technology, Taiwan grant number MOST 107-2218-E-003-002-MY3 and MOST 107-2218-E-035-009-MY3, and in part by the Taiwan Information Security Center, National Sun Yat-Sen University (TWISC@NSYSU).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Rajkumar, R.R.; Lee, I.; Sha, L.; Stankovic, J. Cyber-physical systems: the next computing revolution. In Proceedings of the 47th Design Automation Conference, Anaheim, CA, USA, 13–18 June 2010; pp. 731–736.
2. Lee, J.; Bagheri, B.; Kao, H.A. A cyber-physical systems architecture for industry 4.0-based manufacturing systems. *Manuf. Lett.* **2015**, *3*, 18–23. [[CrossRef](#)]
3. Wan, J.; Xia, M. Cloud-Assisted Cyber-Physical Systems for the Implementation of Industry 4.0. *Mob. Netw. Appl.* **2017**, *22*, 1157–1158. [[CrossRef](#)]
4. Mattern, F.; Floerkemeier, C. From the Internet of Computers to the Internet of Things. In *From Active Data Management to Event-Based Systems and More*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 242–259.
5. Simmon, E.; Kim, K.S.; Subrahmanian, E.; Lee, R.; De Vault, F.; Murakami, Y.; Zettsu, K.; Sriram, R.D. *A Vision of Cyber-Physical Cloud Computing for Smart Networked Systems*; US Department of Commerce, National Institute of Standards and Technology: Gaithersburg, MD, USA, 2013.
6. Lucas-Simarro, J.L.; Aniceto, I.S.; Moreno-Vozmediano, R.; Montero, R.S.; Llorente, I.M. A cloud broker architecture for multicloud environments. In *Large Scale Network-Centric Distributed Systems*; Wiley-IEEE Computer Society Press: Hoboken, NJ, USA, 2013; pp. 359–376.
7. Li, X.; Ma, H.; Zhou, F.; Yao, W. T-broker: A trust-aware service brokering scheme for multiple cloud collaborative services. *IEEE Trans. Inf. Forensics Secur.* **2015**, *10*, 1402–1415. [[CrossRef](#)]
8. Canetti, R.; Dwork, C.; Naor, M.; Ostrovsky, R. Deniable Encryption. In *Advances in Cryptology*; Kaliski, B.S., Jr., Ed.; Springer: Berlin/Heidelberg, Germany, 1997; pp. 90–104.
9. Chi, P.W.; Lei, C.L. Audit-Free Cloud Storage via Deniable Attribute-based Encryption. *IEEE Trans. Cloud Comput.* **2018**, *6*, 414–427. [[CrossRef](#)]
10. Sahai, A.; Waters, B. Fuzzy Identity-Based Encryption. In *Eurocrypt*; Cramer, R., Ed.; Springer: Berlin/Heidelberg, Germany, 2005; pp. 457–473.
11. Shamir, A. Identity-Based Cryptosystems and Signature Schemes. In *Advances in Cryptology*; Blakley, G.R., Chaum, D., Eds.; Springer: Berlin/Heidelberg, Germany, 1985; pp. 47–53.
12. Goyal, V.; Pandey, O.; Sahai, A.; Waters, B. Attribute-based encryption for fine-grained access control of encrypted data. In Proceedings of the ACM Conference on Computer and Communications Security, Alexandria, VA, USA, 30 October–3 November 2006; pp. 89–98.
13. Bethencourt, J.; Sahai, A.; Waters, B. Ciphertext-Policy Attribute-Based Encryption. In Proceedings of the IEEE Symposium on Security and Privacy, Berkeley, CA, USA, 20–23 May 2007; pp. 321–334.
14. Waters, B. Ciphertext-Policy Attribute-Based Encryption: An Expressive, Efficient, and Provably Secure Realization. In *Public Key Cryptography*; Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A., Eds.; Springer: Berlin/Heidelberg, Germany, 2011; pp. 53–70.
15. Lewko, A.B.; Okamoto, T.; Sahai, A.; Takashima, K.; Waters, B. Fully Secure Functional Encryption: Attribute-Based Encryption and (Hierarchical) Inner Product Encryption. In *Eurocrypt*; Gilbert, H., Ed.; Springer: Berlin/Heidelberg, Germany, 2010; pp. 62–91.
16. Attrapadung, N.; Herranz, J.; Laguillaumie, F.; Libert, B.; de Panafieu, E.; Ràfols, C. Attribute-based encryption schemes with constant-size ciphertexts. *Theor. Comput. Sci.* **2012**, *422*, 15–38. [[CrossRef](#)]
17. Tysowski, P.K.; Hasan, M.A. Hybrid Attribute- and Re-Encryption-Based Key Management for Secure and Scalable Mobile Applications in Clouds. *IEEE Trans. Cloud Comput.* **2013**, *1*, 172–186. [[CrossRef](#)]
18. Luo, S.; Hu, J.; Chen, Z. Ciphertext Policy Attribute-Based Proxy Re-encryption. In *Information and Communications Security*; Soriano, M., Qing, S., López, J., Eds.; Springer: Berlin/Heidelberg, Germany, 2010; pp. 401–415.
19. Li, K. Matrix access structure policy used in attribute-based proxy re-encryption. *arXiv* **2013**, arXiv:1302.6428.
20. Liang, K.; Fang, L.; Susilo, W.; Wong, D.S. A ciphertext-policy attribute-based proxy re-encryption with chosen-ciphertext security. In Proceedings of the 2013 5th International Conference on Intelligent Networking and Collaborative Systems (INCoS), Xi'an, China, 9–11 September 2013; pp. 552–559.
21. Chandar, P.P.; Mutkuraman, D.; Rathinrai, M. Hierarchical attribute based proxy re-encryption access control in cloud computing. In Proceedings of the 2014 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2014], Nagercoil, India, 20–21 March 2014; pp. 1565–1570. [[CrossRef](#)]

22. Touati, L.; Challal, Y. Instantaneous Proxy-Based Key Update for CP-ABE. In Proceedings of the 2016 IEEE 41st Conference on Local Computer Networks (LCN), Dubai, UAE, 7–10 November 2016; pp. 591–594. [CrossRef]
23. Yao, X.; Chen, Z.; Tian, Y. A lightweight attribute-based encryption scheme for the Internet of Things. *Future Gener. Comput. Syst.* **2015**, *49*, 104–112. [CrossRef]
24. Oualha, N.; Nguyen, K.T. Lightweight Attribute-Based Encryption for the Internet of Things. In Proceedings of the 2016 25th International Conference on Computer Communication and Networks (ICCCN), Waikoloa, HI, USA, 1–4 August 2016; pp. 1–6. [CrossRef]
25. Jiang, Y.; Susilo, W.; Mu, Y.; Guo, F. Ciphertext-policy attribute-based encryption against key-delegation abuse in fog computing. *Future Gener. Comput. Syst.* **2018**, *78*, 720–729. [CrossRef]
26. Zuo, C.; Shao, J.; Wei, G.; Xie, M.; Ji, M. CCA-secure ABE with outsourced decryption for fog computing. *Future Gener. Comput. Syst.* **2018**, *78*, 730–738. [CrossRef]
27. Li, Z.; Li, W.; Jin, Z.; Zhang, H.; Wen, Q. An Efficient ABE Scheme With Verifiable Outsourced Encryption and Decryption. *IEEE Access* **2019**, *7*, 29023–29037. [CrossRef]
28. Fischer, M.; Scheerhorn, A.; Tönjes, R. Using Attribute-Based Encryption on IoT Devices with instant Key Revocation. In Proceedings of the 2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), Kyoto, Japan, 11–15 March 2019; pp. 126–131. [CrossRef]
29. Dürmuth, M.; Freeman, D.M. Deniable Encryption with Negligible Detection Probability: An Interactive Construction. In *Eurocrypt*; Paterson, K.G., Ed.; Springer: Berlin/Heidelberg, Germany, 2011; pp. 610–626.
30. O’Neill, A.; Peikert, C.; Waters, B. Bi-Deniable Public-Key Encryption. In *Crypto*; Rogaway, P., Ed.; Springer: Berlin/Heidelberg, Germany, 2011; pp. 525–542.
31. Gasti, P.; Ateniese, G.; Blanton, M. *Deniable Cloud Storage: Sharing Files via Public-Key Deniability*; WPES: New York, NY, USA, 2010; pp. 31–42.
32. Beimel, A. Secure Schemes for Secret Sharing and Key Distribution. Ph.D. Thesis, Israel Institute of Technology, Haifa, Israel, 1996.
33. Boneh, D.; Goh, E.J.; Nissim, K. Evaluating 2-DNF Formulas on Ciphertexts. In *TCC*; Springer: Berlin/Heidelberg, Germany, 2005; pp. 325–341.
34. Lewko, A.B. Tools for Simulating Features of Composite Order Bilinear Groups in the Prime Order Setting. In *Eurocrypt*; Pointcheval, D., Johansson, T., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; pp. 318–335.
35. Krawczyk, H.; Rabin, T. Chameleon Signatures. In *NDSS*; The Internet Society: San Diego, CA, USA, 2000.
36. Boneh, D. Bilinear Groups of Composite Order. In *Pairing-Based Cryptography—Pairing 2007*; Takagi, T., Okamoto, T., Okamoto, E., Okamoto, T., Eds.; Springer: Berlin/Heidelberg, Germany, 2007.
37. Cao, Z.; Liu, L. A Note on Bilinear Groups of a Large Composite Order. Cryptology ePrint Archive, Report 2013/812. 2013. Available online: <https://eprint.iacr.org/2013/812> (accessed on 1 October 2019).
38. Iorga, M.; Feldman, L.; Barton, R.; Martin, M.; Goren, N.; Mahmoudi, C. *The NIST Definition of Fog Computing*; NIST: Gaithersburg, MD, USA, 2017.
39. Barker, E.; Barker, W.; Burr, W.; Polk, W.; Smid, M. *Recommendation for Key Management: Part 1: General (Revision 3)*; Technical Report; NIST: Gaithersburg, MD, USA, 2012.
40. Boneh, D.; Canetti, R.; Halevi, S.; Katz, J. Chosen-Ciphertext Security from Identity-Based Encryption. *SIAM J. Comput.* **2007**, *36*, 1301–1328. [CrossRef]

