



Article

# An Efficient and Expressive Fully Policy-Hidden Ciphertext-Policy Attribute-Based Encryption Scheme for Satellite Service Systems

Jiaoli Shi <sup>1</sup> , Chao Hu <sup>2</sup>, Shunli Zhang <sup>1,3,\*</sup> , Qing Zhou <sup>1</sup>, Zhuolin Mei <sup>1,4</sup>, Shimao Yao <sup>1</sup> and Anyuan Deng <sup>1</sup>

<sup>1</sup> School of Computer and Big Data Science, Jiujiang University, No. 551, Qianjin East Road, Jiujiang 332000, China; shijiaoli@jju.edu.cn (J.S.); zhouqing@jju.edu.cn (Q.Z.); 6120153@jju.edu.cn (Z.M.); yaosmall@jju.edu.cn (S.Y.); dengay@jju.edu.cn (A.D.)

<sup>2</sup> Institute of Ecological Civilization, Jiangxi University of Finance and Economics, Nanchang 330013, China; 2202320606@stu.jxufe.edu.cn

<sup>3</sup> Qinghai Institute of Technology, Qinghai University, Xining 810016, China

<sup>4</sup> Jiujiang Key Laboratory of Cyberspace and Information Security, Jiujiang University, Jiujiang 332000, China

\* Correspondence: shunlizh@jju.edu.cn; Tel.: +86-13870214634

**Abstract:** Satellite service systems transfer data from satellite providers to the big data industry, which includes data traders and data analytics companies. This system needs to provide access to numerous users whose specific identities are unknown. Ciphertext-Policy Attribute-Based Encryption (CP-ABE) allows unidentified users with the proper attributes to decrypt data, providing fine-grained access control of data. However, traditional CP-ABE does not protect access policies. Access policies are uploaded to the cloud, stored, and downloaded in plain text, making them vulnerable to privacy breaches. When the access policy is completely hidden, users need to use their own attributes to try matching one by one, which is an inefficient process. In order to efficiently hide the access policy fully, this paper introduces a new efficient and expressive Fully Policy-Hidden Ciphertext-Policy Attribute-Based Encryption scheme (CP-ABE-FPH), which integrates the 2-way handshake O-PSI method with the ROBDD method. The integration offers advantages: (1) High efficiency and high expressiveness. The access policy using ROBDD is highly expressive but computationally intensive due to its recursive nature. This shortcoming is overcome in CP-ABE-FPH using the proposed O-PSI method, and the access policy is matched quickly and secretly. (2) High flexibility. The decryption process does not require the owner or the Key Generation Center (KGC) to be online, and system attributes can be added at any time. Security analysis shows that the access policy is fully hidden. Efficiency analysis and simulation results show that the proposed scheme is highly efficient in decryption compared with existing schemes.

**Keywords:** outsourcing privacy set intersection; ciphertext-policy attribute-based encryption; fully policy-hidden; privacy policy matching



**Citation:** Shi, J.; Hu, C.; Zhang, S.; Zhou, Q.; Mei, Z.; Yao, S.; Deng, A. An Efficient and Expressive Fully Policy-Hidden Ciphertext-Policy Attribute-Based Encryption Scheme for Satellite Service Systems. *ISPRS Int. J. Geo-Inf.* **2024**, *13*, 321. <https://doi.org/10.3390/ijgi13090321>

Academic Editors: Wolfgang Kainz and Eliseo Clementini

Received: 20 June 2024

Revised: 29 August 2024

Accepted: 3 September 2024

Published: 5 September 2024



**Copyright:** © 2024 by the authors. Published by MDPI on behalf of the International Society for Photogrammetry and Remote Sensing. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Satellite data have various applications, including vehicle monitoring and navigation, maritime transportation and fisheries, geodesy, land and resource monitoring, meteorological monitoring, and disaster prevention and reduction. Satellite service systems transfer data from satellite providers to the big data industry, which includes data traders and data analytics companies. The big data industry seeks to store its data in a semitrusted cloud hosted by cloud servers to avoid the burden of data maintenance. Data security is crucial for supporting these applications. Without protection, data are vulnerable to privacy breaches (Georgiadou et al. [1] mentioned). Typically, the big data industry needs to provide access to numerous users whose specific identities are unknown. Fortunately, Ciphertext-Policy Attribute-Based Encryption (CP-ABE) enables unidentified users to access data. This process involves the data owner defining an access policy, associating

the data with it into ciphertext, and uploading the ciphertext onto the cloud server. An authorized user receives the authorized attribute set and attribute private key from the trusted Key Generation Center (KGC). When a user downloads data from the cloud server, if their attribute private key matches the access policy, they can successfully decrypt and access the plaintext data.

In traditional CP-ABE, access policies defined by owners are not protected, meaning they are uploaded to the cloud, stored, and downloaded in plain text, making them vulnerable to privacy breaches. To prevent information leakage from access policies, two types of policy hiding methods are proposed: *partially policy-hidden* and *fully policy-hidden* (Zhang et al. [2] mentioned). The former only hides the attribute values in the policy, while the latter conceals both attribute names and values. This paper focuses on the latter. Firstly, during encryption in a fully policy-hidden CP-ABE scheme, all attributes are tied to the ciphertext, leading to increased encryption and communication overhead. Secondly, this method requires users to attempt to decrypt all attributes, resulting in high decryption costs. Thirdly, binding all system attributes on each ciphertext means that a new system attribute cannot be added later to the ciphertext. Nevertheless, fully policy-hidden would require users to try using their own attributes one by one when decrypting, which is an inefficient privacy policy matching problem. This problem has attracted a lot of research, such as [3–6]. In order to fully hide the access policy, Müller et al. [4] used a method similar to *k-anonymity*, and Lai et al. [3] used an inner product PE method. Both Lai et al. [3] and Phuong et al. [6] need to define all system attributes at the beginning and rerun the initialization algorithm if attributes are added. To speed up the privacy policy matching, some existing schemes have adopted the bloom filter method to reduce the cost of privacy policy matching, as seen in Yang et al. [7] and Luo et al. [8]. However, both schemes are unable to withstand the selective plaintext attack.

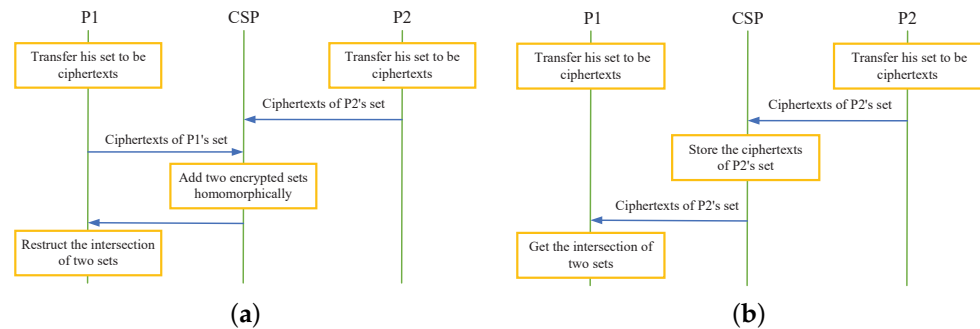
In this paper, the issue of *privacy-preserving policy matching* is reframed as an *Outsourcing Privacy Set Intersection* (O-PSI) problem utilizing *Reduced Ordered Binary Decision Diagrams* (ROBDD).

The ROBDD method was first introduced by Affum et al. [9]. This method can convert a tree-type access policy into multiple attribute sets, which are referred to as feasible paths. If a user's attribute set contains one of the feasible paths, it indicates that the user's attribute set satisfies the access policy. The ROBDD method is capable of providing a combination of rich-expressive access policy and rapid policy matching. However, the ROBDD is openly stored on cloud servers, which may lead to information leaks.

Privacy Set Intersection (PSI) aims to protect the personal privacy of data subjects while ensuring the exact intersection of two data sets. Outsourcing PSI (abbreviated as O-PSI) involves a third-party server, typically referred to as a Cloud Service Provider (CSP), to handle the computational burden that terminals should bear and avoid direct face-to-face interaction between entities. The introduction of CSP makes O-PSI easier to deploy in modern networks (Morales et al. [10] mentioned). Existing research on O-PSI methods often relies on the CSP for set intersection operations, which may compromise the cardinality of the sets. For example, while Zhao et al. [11] and Thapa et al. [12] concealed elements and the intersection of two sets, the cardinalities of the sets remained exposed. Sun et al. [13] made advancements in this area; however, they still disclosed the exact number of elements in one of the sets in the intersection. Both Thapa et al. [12] and Li et al. [14] require parties to be online and parties feel inconvenient. Furthermore, reducing the number of communication rounds is desirable. Most current O-PSI methods usually involve three rounds, which is shown in Figure 1a. The 3-way handshake O-PSI requires CSP to collect and add encrypted sets of two parties homomorphically and send the result to one party to restrict the intersection of two sets. As can be seen, this involves three rounds of communication. This paper proposes a new 2-way handshake O-PSI method shown in Figure 1b. In the new O-PSI method, P2 encrypts their set and uploads the encrypted set onto CSP. When P1 tries to obtain the intersection of their set and P2's set, they download the P2's encrypted set from CSP and obtain the intersection locally. The process only involves two rounds of

communication. The fewer the rounds, the smaller the communication cost, and the smaller the energy consumption, and at the same time, the instability caused by uncertainty can be minimized. Thus, the proposed O-PSI method achieves more efficiency and robustness.

This paper proposes a new method called O-PSI, which requires only a 2-way handshake and conceals both the elements and cardinality of sets. Figure 1 illustrates the comparison between the 3-way handshake process of existing methods with the 2-way handshake process of the proposed method.



**Figure 1.** Comparison between the proposed O-PSI method and others: (a) Others (3-way handshake process). (b) The proposed O-PSI (2-way handshake process).

The proposed O-PSI method only requires two handshake communications and hides the elements and cardinalities of two sets, making it ideal for concealing policies in CP-ABE schemes. This paper introduces an efficient and expressive CP-ABE scheme supporting fully policy-hidden (CP-ABE-FPH) by combining the innovative 2-way handshake O-PSI method with ROBDD (Reduced Ordered Binary Decision Diagrams). Specifically, our main contributions are as follows:

(1) We develop a novel 2-way handshake O-PSI method based on RSA-OPRF (RSA-based oblivious pseudorandom function), as used in Xu et al. [15]. In this new O-PSI method, party P2 converts their set into ciphertext and uploads the encrypted set onto a cloud server. The server stores the P2's encrypted set and offers on-demand services to other parties. When another party, P1, downloads P2's encrypted set, they compute the intersection of P2 and their own set. The proposed O-PSI method reduces the number of communications from three to two without revealing the elements or sizes of the sets.

(2) We propose a fully policy-hidden CP-ABE scheme (CP-ABE-FPH) by integrating the proposed 2-way handshake O-PSI method and ROBDD. This integration innovatively solves the low efficiency of decryption in a fully policy-hidden CP-ABE scheme. The proposed scheme provides data confidentiality protection, fine-grained access control, a fully hidden and expressive access policy, and lightweight terminal computing. Moreover, the flexibility of the proposed scheme is notable. The decryption process eliminates the need for online interactions with owners or the Key Generation Center (KGC), and system attributes can be added at any point in time.

## 2. Related Works

### 2.1. CP-ABE with Policy Hidden

There are two policy-hidden methods (Zhang et al. [2] mentioned): The (1) partially policy-hidden method, which hides only attribute values while exposing attribute names. (2) The fully policy-hidden method, which hides both attribute names and attribute values. When hiding access policies for CP-ABE, all system attributes (whether the ciphertext is bound or not) are tied to the ciphertext to prevent access policies from leaking sensitive information. However, the fully policy-hidden method is neither as efficient as desired nor as flexible as needed.

There are many fully policy-hidden CP-ABE schemes in the literature. Lai et al. [3] originally constructed a scheme that hides access policies in CP-ABE from attribute-hiding

inner-product PE in a formal manner. However, the values for attributes must be specified during the initialization stage. Müller et al. [4] utilized graph theory to improve the anonymity of access policy in CP-ABE, akin to  $k$ -anonymity. Hur [5] enhanced the expressiveness of the access control policy, which is obfuscated. Phuong et al. [6] presented an efficient fully policy-hidden CP-ABE scheme based on Viète’s formulas. Similar to Lai et al. [3], the attribute fields must be determined during the initialization phase. Yang et al. [7] developed an attribute bloom filter to conceal the mapping between attributes and line numbers of the access matrix and subsequently presented a fully policy-hidden CP-ABE scheme. Their method facilitates the addition of new attributes, but the bloom filter may allow users to reveal the hidden map by querying attributes. Like Yang et al. [7], Zhang et al. [16] also used an attribute bloom filter to build CP-ABE schemes. Luo et al. [8] also presented a fully policy-hidden CP-ABE scheme. Despite offering a rich access structure, the scheme is vulnerable to collusion attacks. Table 1 shows the comparison of existing schemes and the proposed CP-ABE-FPH scheme.

In conclusion, these fully policy-hidden CP-ABE schemes effectively conceal all attributes associated with the ciphertext. However, some schemes require the specification of all attributes during initialization, some are susceptible to collusion attacks, and some struggle with inefficient policy matching. Therefore, achieving a balance between efficiency, security, and expressiveness remains a challenging problem.

**Table 1.** Characteristics comparison.

Characteristics	[3]	[4]	[5]	[6]	[7]	[8]	CP-ABE-FPH
Fully policy-hidden	✓	✓	✗	✓	✓	✓	✓
High expressiveness	✓	✗	✓	✗	✓	✓	✓
Add attributes anytime	✗	✓	✗	✗	✗	✗	✓
Encryption cost	H	H	M	M	M	H	M
Decryption cost	M	H	H	H	M	M	L
Collusion-resistance	✓	✓	✓	✓	✗	✗	✓

## 2.2. O-PSI

O-PSI is a type of distributed multiparty computation problem. Thapa et al. [12] secretly computed the intersection of attribute sets of two users. However, their method required both parties to negotiate the key in advance. Meanwhile, both parties negotiate the key in advance and both participants must be online. Zhang et al. [2] added a converting step into hidden vector encryption to achieve private matching decrypting attribute vector with encrypting attribute vector. Nevertheless, their method involves the universe of system attributes, making the scheme less flexible and challenging to add new attributes in the future. Li et al. [14] proposed a similarity function to help a user find potential friends without compromising the user’s privacy, using the concept of friends of friends. However, this matching is a fuzzy match rather than an exact one. Sotiraki et al. [17] computed set-maximal matches between a database of aligned genetic sequences and an individual’s DNA preserving the privacy of both the database owner and the individual using secret sharing techniques. However, their private matching protocol still requires both participants to be online, similar to the method proposed by Thapa et al. [12]’s method. Meanwhile, Zhao et al. [11] can conceal the elements of sets and the cardinality of sets’ intersection; they revealed the cardinalities of each set.

There are numerous other similar privacy-preserving distributed multiparty computation problems. For example, Cheng et al. [18] proposed a novel problem known as Privacy-Preserving Epidemiological Data Collection. In their scenario, privacy protection is achieved among users, servers, and data collectors who do not trust each other. Meanwhile, Li et al. [19] focused on privacy-preserving computational geometry, addressing the privacy concerns related to determining the inclusion of a point within a closed graph. A recent

hot area of research is data privacy protection in machine learning. Feng et al. [20] presented a privacy-preserving tensor decomposition approach over encrypted big data using homomorphic encryption. Subsequently, Feng et al. [21] proposed a differentially private tensor-based recurrent neural network (DPTRNN) to process heterogeneous sequential data in the Internet-of-Things scene.

### 3. Cryptographic Primitive

#### 3.1. RSA-OPRF

RSA-OPRF (RSA-based Oblivious Pseudo-Random Function) was introduced by Keelveedhi et al. [22].

*Initialization phase.* A server generates the RSA parameters to issue a public key  $Params^{RSA-OPRF} = (N)$  and a private key  $SK^{RSA-OPRF} = (N, d)$  with an input  $\bar{e}$ , where  $\bar{e}d \equiv 1 \pmod{\phi(N)}$ .

*Client execution phase 1.* A client selects a random number  $r \in Z_N^*$ , two hash functions:  $H_1 : \{0, 1\}^* \rightarrow Z_N^*$ ,  $H_2 : Z_N^* \rightarrow \{0, 1\}^k$  and calculates:  $h_1 = H_1(m)$ ,  $\tilde{x} = h_1 r^{\bar{e}} \pmod{N}$ . Then, the client sends  $\tilde{x}$  to the server.

*Server execution phase.* The server calculates  $\hat{x} = \tilde{x}^d$  and sends it back to the client.

*Client execution phase 2.* The client computes  $\tilde{x} = \hat{x}(r)^{-1}$ . Then, the client verifies  $(\tilde{x})^{\bar{e}} = h_1$ . If the verification succeeds, the client runs a hash function and obtains  $\tilde{x} = H_2(\tilde{x})$ ;

In this RSA-OPRF protocol, the server cannot determine the client's inputs  $m$ , even if they know the function,  $H_1$ , because  $r$  is randomly selected. Additionally, the client cannot obtain the server's private key  $d$  unless this RSA hard assumption is violated.

#### 3.2. CP-ABE

A CP-ABE scheme consists of four probabilistic algorithms with the following syntax:  $Initial(1^\lambda) \rightarrow (MSK, Params)$ . It takes the inputs of a security parameter  $\lambda$  and outputs the public key  $Params$  and the master key  $MSK$ .

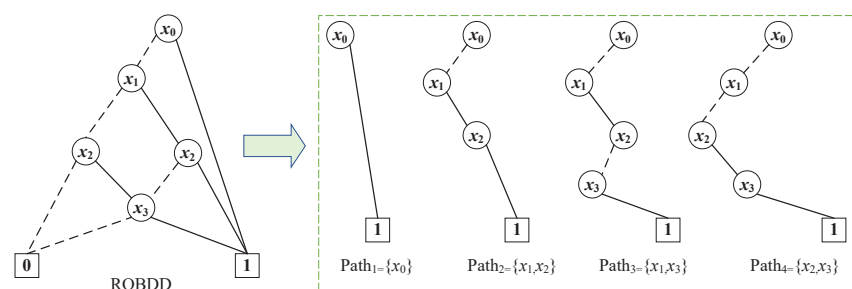
$KeyGen(MSK, Params, Y) \rightarrow (SK)$ . It takes the inputs of  $Params$ ,  $MSK$ , and a user's attribute set  $Y$  and outputs the user's private key  $SK$ .

$Encrypt(Params, m, \Lambda) \rightarrow (Cm)$ . It takes the inputs of  $Params$ , a message  $m$ , and an access structure  $\Lambda$  and outputs a ciphertext  $Cm$ .

$Decrypt(Params, Cm, sk) \rightarrow (m)$ . It inputs  $Params$ , a secret key  $SK$ , and the ciphertext  $Cm$ , then outputs the message  $m$  or a symbol  $\perp$ , indicating a failure decryption.

#### 3.3. ROBDD and Feasible Paths (or Decryption Paths)

Affum et al. [9] proposed efficient attribute expressions by constructing a ROBDD method. Li et al. [23] proposed a CP-ABE scheme based on ROBDD, which enables rapid decryption and utilizes a fixed-length private key. In this paper, the ROBDD is employed to convert an access control policy to a set of feasible paths. Assuming the access policy is  $x_0$ ,  $(x_1$  and  $x_2)$ ,  $(x_1$  and  $x_3)$ , or  $(x_2$  and  $x_3)$  after sorting the attributes, the ROBDD and feasible paths can be depicted as shown in Figure 2. More details can be found in the work of Li et al. [23].



**Figure 2.** The ROBDD structure and feasible paths for access policy.

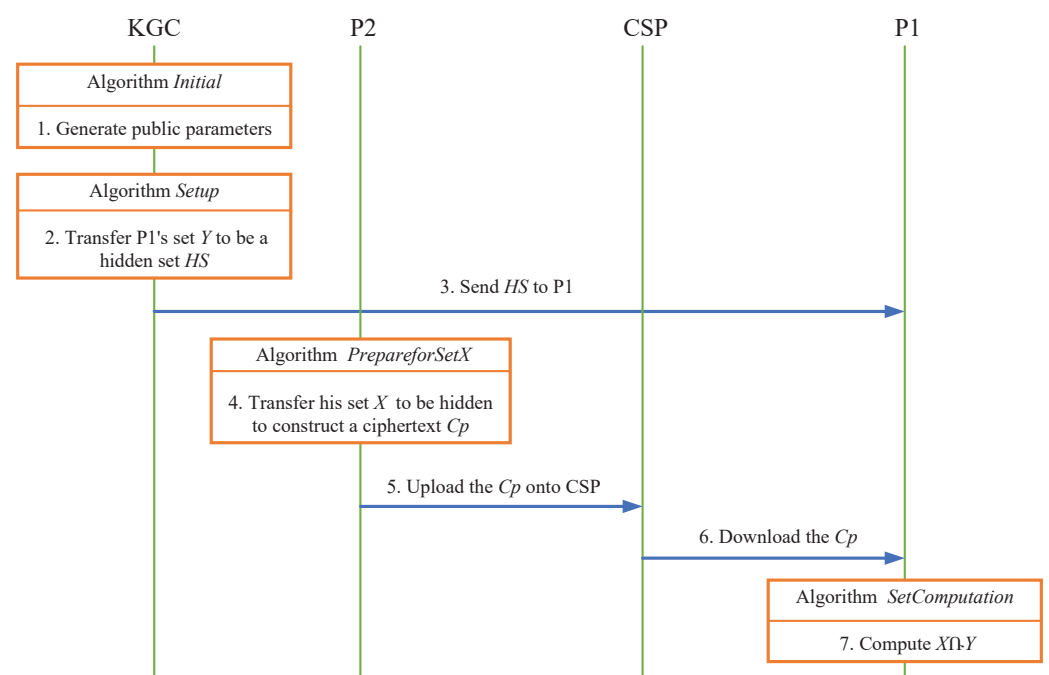


#### 4. Construction of the Proposed 2-Way Handshake O-PSI

The formal definition of the new 2-way handshake process, the O-PSI method, is as follows.

**Formal Definition of the proposed 2-way handshake O-PSI.** Party P2 is assumed to have a set  $X = \{x_i\}_{i \in n_X}$ , and another party P1 is assumed to have another set  $Y = \{y_i\}_{i \in n_Y}$ . With the assistance of a cloud server, P2 does not need to be online continuously. P1 wants to obtain  $Y \cap X$  without revealing to the P2 or the server more than what can be inferred from the result. Specifically, (1) the server cannot obtain  $|X|$ ,  $|Y|$ ,  $|X \cap Y|$  or any  $x_i$  or  $y_j$ , and (2) P1 cannot obtain  $|X|$  or any  $x_i$  outside of the set  $X \cap Y$ .

The proposed 2-way handshake O-PSI method consists of 4 algorithms, illustrated in Figure 3. KGC acts as a trusted authority for authorization. The server is a semitrusted party responsible for storing and providing online access services. P1 seeks to determine the intersection of their set and P2's set.



**Figure 3.** The overview of the proposed 2-way handshake O-PSI.

##### 4.1. Initialization

Algorithm 1 *Initial* runs on KGC to generate public parameters  $params$ .

---

##### Algorithm 1 *Initial*

---

Input: a security parameter  $\lambda$ ;

Output:  $Params, MSK$ ;

**1:** KGC selects a random number  $N$ . For an input  $\bar{e}$ , there is an integer  $d$  that satisfies  $\bar{e}d \equiv 1 \pmod{\phi(N)}$ ;

**2:** KGC selects a group  $G_1$  with the order  $p$ . Let  $p$  be  $N$ . The  $G_1$  has a generator  $g$ ;

**3:** KGC selects three hash functions:

$$H_1 : \{0, 1\}^* \rightarrow G_1;$$

$$H_2 : G_1 \rightarrow Z_N^{\bar{k}};$$

Where  $*$  denotes any length and  $Z_N^{\bar{k}}$  denotes the number formed by extracting  $\bar{k}$  bit after modulo  $N$ ;

**4:** KGC publishes public parameters:  $Params = (N, H_1, H_2, g, G_1, \bar{e})$ ;

**5:** KGC transmits secretly the  $d$  to a semitrusted server.

---

#### 4.2. Prepare for the Set Y for a Party P1

Algorithm 2 *PrepareforSetY* runs on KGC to issue a hidden set (called *HS*) for a party P1.

---

#### Algorithm 2 *PrepareforSetY*

---

Input: *Params*, P1's set  $Y = \{y_j\}_{j \in [1, n_Y]}$ ;

Output: a hidden set *HS*;

1: KGC determines P1's attribute set  $\{y_j\}_{j \in [1, n_Y]}$ ;

2: KGC picks a random number  $r_y \in Z_N^*$  and computes:

$$\{h_j = H_1(y_j)\}_{j \in [1, n_Y]}$$

$$\{\tilde{y}_j = h_j \cdot (g^{r_y})^{\tilde{e}}\}_{j \in [1, n_Y]}$$

wherein  $n_Y$  denotes the cardinalities of the set Y;

3: KGC sends  $\{\tilde{y}_j\}_{j \in [1, n_Y]}$  to the server along with random numbers  $\{r'_j\} \in Z_N^*$ . Then

the server computes:  $\{\hat{y}_j = (\tilde{y}_j)^d\}_{j \in [1, n_Y]}$  and  $\{r'_j{}^d\}$ , and then sends them back;

4: KGC filters out the set  $\{\hat{y}_j\}_{j \in [1, n_Y]}$  and computes  $\{\tilde{y}_j = \hat{y}_j (g^{r_y})^{-1}\}_{j \in [1, n_Y]}$ ;

5: KGC verifies  $(\tilde{y}_j)^{\tilde{e}} = h_j$ . If the verify succeeds, KGC does a hash function on  $\tilde{y}_j$ , and gets:  $\{\tilde{y}_j = H_2(\tilde{y}_j)\}_{j \in [1, n_Y]}$ ;

6: KGC shuffles the order of the sets of  $\{\tilde{y}_j\}_{j \in [1, n_Y]}$  and sends  $HS = \{\tilde{y}_j\}_{j \in [1, n_Y]}$  to P1 secretly.

---

#### 4.3. P2 Prepares the Set X

Algorithm 3 *PrepareforSetX* runs on P2 to prepare the matching set to be uploaded. More narrowly, P2 transforms  $X = \{x_i\}_{i \in n_X}$  to be hidden to construct a ciphertext  $Cp$  and upload the  $Cp$  onto a server.

---

#### Algorithm 3 *PrepareforSetX*

---

Input: *Params*,  $X = \{x_i\}_{i \in n_X}$ ;

Output:  $Cp$ ;

1: P2 selects a random number  $r \in Z_N^*$  and computes:

$$\{h_i = H_1(x_i)\}_{i \in [1, n_X]}$$

$$\{\tilde{x}_i = h_i (g^r)^{\tilde{e}}\}_{i \in [1, n_X]}$$

2: P2 selects sends  $\{\tilde{x}_i\}_{i \in [1, n_X]}$  to the server along with random numbers  $\{r'_i\} \in Z_N^*$ ;

3: The server computes both  $\{\hat{x}_i = (\tilde{x}_i)^d\}_{i \in [1, n_X]}$  and  $\{r'_i{}^d\}$ , and then sends them back;

4: P2 filters out the set  $\{\hat{x}_i\}_{i \in [1, n_X]}$  and computes  $\{\tilde{x}_i = \hat{x}_i (g^r)^{-1}\}_{i \in [1, n_X]}$ ;

5: P2 verifies  $(\tilde{x}_i)^{\tilde{e}} = h_i$ . If the verify succeeds, P2 runs a hash function and gets:

$$\{\tilde{x}_i = H_2(\tilde{x}_i)\}_{i \in [1, n_X]}$$

6: P2 computes  $Cp = \prod_{i \in [1, n_X]} \tilde{x}_i$ , and sends it to a server.

---

#### 4.4. P1 Computes the Set Intersection

Algorithm 4 *SetIntersection* runs on P1 to obtain the intersection of two sets secretly. It is reasonable to assume that P1 knows how many elements are in their own set.

**Algorithm 4** *Setintersection*


---

Input:  $HS, Cp$ ;  
Output:  $(X \cap Y)$  or  $\perp$ ;  
1:  $Y_{inset} = \perp$ ;  
2: FOR  $\tilde{y}_j \in HS$  DO  
3: IF  $Cp \bmod \tilde{y}_j = 0$  THEN  
4: Insert  $\tilde{y}_j$  to the set  $Y_{inset}$ ;  
5: END IF  
6: END FOR  
7: Return  $Y_{inset}$ .

---

**4.5. More Discussion about the 2-Way Handshake O-PSI**

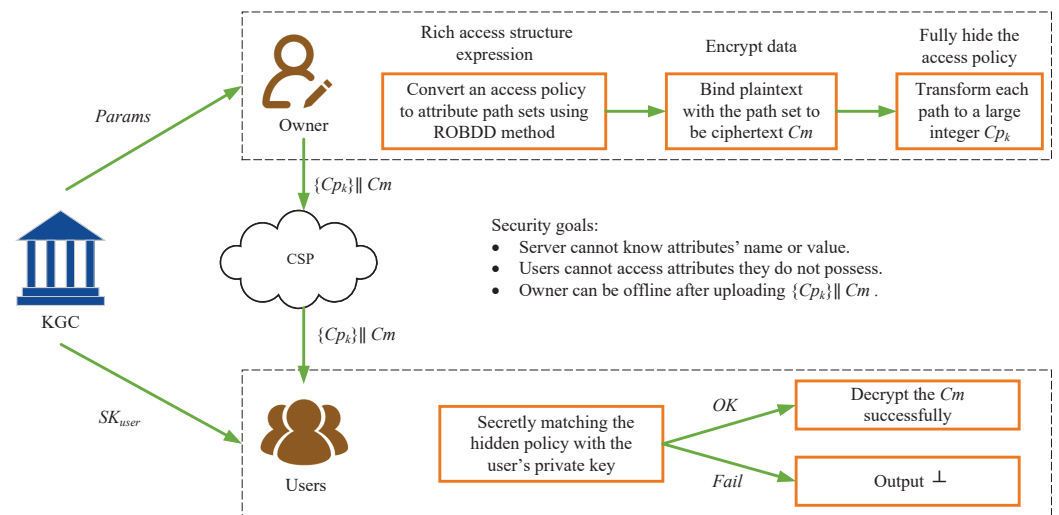
Let us discuss the security of this algorithm. On the one hand,  $\tilde{y}_j$  and  $\hat{y}_j$  do not need to be transmitted secretly. This is because both are dependent on the random number  $r_y$ . Nevertheless, the computation of  $\tilde{y}_j$  is independent with  $r_y$ , because  $\tilde{y}_j = \hat{y}_j(g^{r_y})^{-1} = (\tilde{y}_j)^d(g^{r_y})^{-1} = (h_j)^d(g^{r_y})^{\bar{e}d}(g^{r_y})^{-1} = (h_j)^d$ , which ensures that the matching result is correct if  $x_i = y_j$ . On the other hand,  $r_y$  keeps confidential for the server, and then the server cannot forge the  $\tilde{y}_j$ . Finally, the number of  $\{\tilde{r}_j\}$  prevents the server from knowing the cardinality of set  $Y$ . As same as in the algorithm *PrepareforsetY*,  $\{\tilde{x}_i\}_{i \in [1, n_X]}$  and  $\{\hat{x}_i\}_{i \in [1, n_X]}$  can be transmitted publicly. This is because both are dependent on the random number  $r$ . Nevertheless, the computation of  $\{\tilde{x}_i\}_{i \in [1, n_X]}$  is independent with  $r$ , because  $\tilde{x}_i = \hat{x}_i(g^r)^{-1} = (\tilde{x}_i)^d(g^r)^{-1} = (h_i)^d(g^r)^{\bar{e}d}(g^r)^{-1} = (h_i)^d$ , which ensures that the matching result is correct if  $x_i = y_j$ . Finally,  $r_y$  keeps the server confidential, and then the server cannot forge  $\{\tilde{x}_i\}_{i \in [1, n_X]}$ . Meanwhile,  $d$  keeps P2 secret; thus, P2 cannot forge  $\{\tilde{x}_i\}_{i \in [1, n_X]}$ . Finally, the number of  $\{r_i\}$  prevents the server from knowing the cardinality of the set  $X$ .

Let us discuss the potential applications of the proposed 2-way handshake O-PSI method. It is a kind of private-preserving set match method and can be adopted in many scenes, such as private-preserving keyword searches on semitrusted cloud scenes. In detail, one can define and encrypt some keywords to fetch all relevant ciphertext data sorted by relevance from cloud servers. In the next section of this paper, we apply the 2-way handshake O-PSI method to quickly match access policies in CP-ABE with full policy hiding.

**5. Fully Policy-Hidden CP-ABE Based on 2-Way Handshake O-PSI****5.1. Scenario Description**

The proposed CP-ABE-FPH scheme is shown in Figure 4. Data owners are looking to store their data in a semitrusted cloud, hosted by cloud servers, to avoid the burden of data maintenance. They aim to provide services to thousands of users anytime, anywhere. To ensure that their data are protected from unauthorized access during data outsourcing and to allow multiple authorized users to download and access data on demand, a common approach is to use the CP-ABE method. This involves the data owner defining an access policy, binding the access policy to the data, turning it into ciphertext, and uploading it to the cloud server. Authorized users obtain the authorized attribute sets and attribute private keys from the Key Generation Center (KGC). When users download data from the cloud server, if their attribute private key matches the access policy, they can successfully decrypt and obtain the plaintext data.





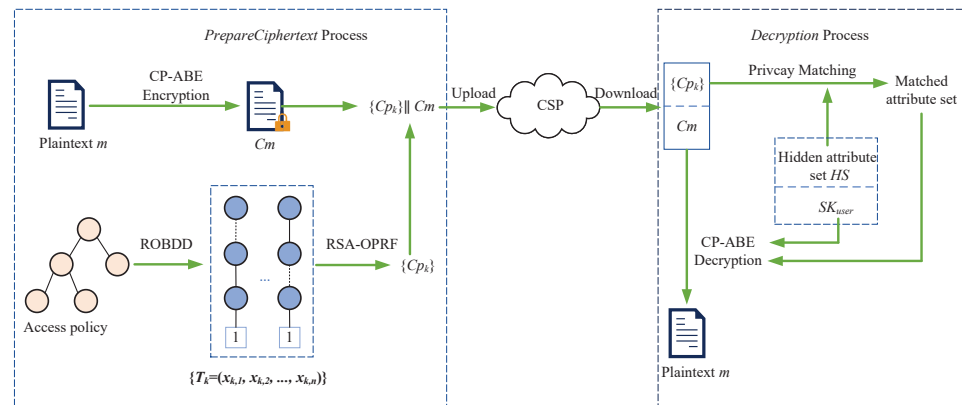
**Figure 4.** The fully policy-hidden CP-ABE scenario.

Like other previous cloud scenes, it is assumed that there is a Key Generation Center (KGC), which is trusted and issues a private key for each user. The data owner is trusted but wants to go offline after uploading the ciphertext. The server is semitrusted. It is honest but curious. The server always wants to obtain information from ciphertexts, and so do the users.

In traditional CP-ABE, a data owner binds data with an access policy, and a user can decrypt the ciphertext if their private key matches the access policy. This cryptographic technology has a one-to-many characteristic, making it suitable for integrating to achieve data confidentiality protection and fine-grained access control in distributed computing environments, such as cloud storage scenes (Zhong et al. [24] mentioned). However, the original CP-ABE did not consider privacy protection issues, and its access policy would leak privacy. In order to balance efficiency, security, and flexibility while also maintaining confidentiality and preventing information leaks, creating fully policy-hidden CP-ABE schemes is challenging. In an effort to reduce the computing cost for users, Li et al. [23] proposed a fast decryption CP-ABE scheme that uses ROBDD (Reduced Ordered Binary Decision Diagrams) to improve the expressiveness of access policy. However, this scheme reveals the access policy because the ROBDD is openly stored on cloud servers. To address the high costs associated with increased privacy protection, Yang et al. [7] designed an attribute bloom filter to conceal the mapping between attributes and access matrix line numbers, resulting in a fully policy-hidden CP-ABE scheme. While this method facilitates the addition of new attributes, it also allows users to uncover the hidden map by querying with attributes. To sum up, the challenge of full policy-hidden CP-ABE scheme design lies in satisfying high expressiveness of access policy, high user-side performance, and high security at the same time.

### 5.2. Construction of CP-ABE-FPH

Based on the proposed O-PSI method, the proposed Ciphertext-Policy Attribute-Based Encryption scheme supporting Full Policy-Hidden (CP-ABE-FPH) is composed of four algorithms shown in Figure 5. In the *PrepareCiphertext* process, two transform steps are performed for an access policy: Firstly, an access policy designated by an owner is transformed to be feasible attribute paths using the ROBDD method proposed by Li et al. [23]. Secondly, the feasible paths are encrypted to be  $\{C_{p_k}\}$  using *Message driver key generation* based on RSA-OPRF [22].



**Figure 5.** The framework of the CP-ABE-FPH scheme.

Let us look at the overall framework of the CP-ABE-FPH scheme. An owner generates a plaintext  $m$ , defines a tree-type access policy for the  $m$ , and encrypts it to be  $C_m$  using a CP-ABE method. Meanwhile, the owner transforms the tree-type access policy to be many attribute sets, called feasible paths  $T_k$ . Next, the feasible paths are encrypted to be  $\{Cp_k\}$  based on RSA-OPRF. In the final,  $\{Cp_k\}$  and  $C_m$  are concatenated and uploaded to the cloud. On the user side, a user downloads  $\{Cp_k\}$  and  $C_m$ . The user first tries to match their attribute set with  $\{Cp_k\}$ . If their attribute set satisfies any  $Cp_k$  in  $\{Cp_k\}$ , they do not try using their attributes one by one and decrypt  $C_m$  to be  $m$  directly.

Essentially, the private matching between an owner's attribute paths and a user's attribute private key is converted to a server-aided Privacy Set Intersection problem. If the new Privacy Set Intersection problem can be solved, a fully policy-hidden CP-ABE scheme can be constructed efficiently and securely with high expressiveness of access policy. Fortunately, the server-aided Privacy Set Intersection problem would be solved perfectly with the proposed 2-way handshake O-PSI method. The main notations are listed in Table 2.

**Table 2.** Notations.

Notations	Description
$MSK = (d, MSK^{CP-ABE})$	The system master key
$MSK^{CP-ABE} = (a, b)$	The master key for CP-ABE
$Params$	The system public parameters
$Params^{CP-ABE}$	The public parameters for CP-ABE
$SK_{user}$	The private key of a user
$H_1, H_2$	The two hash functions
$n_{path}$	The number of feasible paths
$n_{k,attr}$	The number of attributes in path $T_k$
$n_{user,attr}$	The number of attributes in $SK_{user}$
$\{T_k = \{x_{k,i}\}\}$	The set of feasible paths $k \in [1, n_{path}], i \in [1, n_{k,attr}]$
$\{Cp_k\}_{k \in [1, n_{path}]}$	The hidden policy in the form of paths
$m$	The plaintext data
$C_m = (C_1, C_2, \{C_k\})$	The ciphertext data
$a, b, r_y, r_u, r, s$	The random numbers

### 5.2.1. Initialization

Algorithm 5 *Initial* runs on KGC to generate public parameters  $params$  and a master key  $MSK$ .

---

#### Algorithm 5 *Initial*

---

Input: security parameter  $\lambda$ ;

Output:  $Params, MSK$ ;

- 1: KGC selects a random number  $N$ . For an input  $\bar{e}$ , there is an integer  $d$  that satisfies  $\bar{e}d \equiv 1 \pmod{\phi(N)}$ ;
  - 2: KGC selects  $Params^{CP-ABE} = (e, g, G_1, e(g, g)^a, g^b)$  and  $MSK^{CP-ABE} = (a, b)$ ;
  - 3: KGC also chooses two hash functions:  $H_1 : \{0, 1\}^* \rightarrow G_1, H_2 : G_1 \rightarrow Z_N^{\bar{k}}$ ;
  - 4: KGC publishes public parameters:  $Params = (N, \bar{e}, H_1, H_2, Params^{CP-ABE})$ ;
  - 5: KGC keeps the master key  $MSK = (d, MSK^{CP-ABE})$  secretly;
  - 6: KGC transmits the  $d$  secretly to a server.
- 

To generate the public parameters, KGC needs to select a group  $G_1$  with the order  $p$ , a bilinear mapping  $e : G_1 \times G_1 \rightarrow G_T, a, b \in Z_N^*$ . The bilinear mapping has properties:  $e(g, g) \neq 1$  and  $e(g^a, g^b) = e(g, g)^{ab}$ . The  $G_1$  has a generator  $g$ , wherein  $*$  denotes any length, and  $\bar{k}$  denotes the number of digits in an integer.

### 5.2.2. User Registration

Algorithm 6 *KeyGen* runs on KGC to issue a private key  $SK_{user}$  for a user according to the user's attribute set  $Y = \{y_j\}_{j \in [1, n_{user, attr}]}$ , wherein  $n_{user, attr}$  represents the number of attributes given for this user.

---

#### Algorithm 6 *KeyGen*

---

Input:  $MSK$ , a user's attribute set  $Y = \{y_j\}_{j \in [1, n_{user, attr}]}$ ;

Output:  $SK_{user}$ ;

- 1: KGC determines a user's attributes  $Y = \{y_j\}_{j \in [1, n_{user, attr}]}$ ;
  - 2: KGC selects a random number  $r_y \in Z_N^*$  and computes:
 
$$\{h_j = H_1(y_j)\}_{j \in [1, n_{user, attr}]}; \quad \{\tilde{y}_j = h_j(g^{r_y})^{\bar{e}}\}_{j \in [1, n_{user, attr}]};$$
  - 3: KGC sends  $\{\tilde{y}_j\}_{j \in [1, n_{user, attr}]}$  to the server along with some random numbers  $\{r_j\}$ . Then the server computes:  $\{\hat{y}_j = (\tilde{y}_j)^d\}_{j \in [1, n_{user, attr}]}$  and  $\{r_j^d\}$ , then sends them back;
  - 4: KGC filters out  $\{\hat{y}_j\}_{j \in [1, n_{user, attr}]}$  and calculates  $\{\tilde{y}_j = \hat{y}_j(g^{r_y})^{-1}\}_{j \in [1, n_{user, attr}]}$ ;
  - 5: KGC verifies  $(\tilde{y}_j)^{\bar{e}} = h_j$ . If the verification succeeds, KGC does a hash function on  $\tilde{y}_j$ , and obtains  $\{\tilde{y}_j = H_2(\tilde{y}_j)\}_{j \in [1, n_{user, attr}]}$ ;
  - 6: KGC selects  $r_u \in Z_N^*$ , and calculates  $D_0 = g^{-a+b/r_u}, \{D_j = g^{(\tilde{y}_j/r_u)}\}_{j \in [1, n_{user, attr}]}$ ;
  - 7: KGC sends  $SK_{user} = (D_0, \{D_j, \tilde{y}_j\}_{j \in [1, n_{user, attr}]})$  to the user secretly.
- 

### 5.2.3. Owner Prepares Ciphertext

Algorithm 7 *PrepareCiphertext* runs on an owner to prepare the ciphertext and a hidden policy to be uploaded. More narrowly, the owner must follow a two-step process to process the access policy. In the first step, they should use the ROBDD method to convert the tree-shaped access policy into multiple sets of attributes, known as feasible paths  $\{T_k = \{x_{k,i}\}\}_{k \in [1, n_{path}], i \in [1, x_{k, attr}]}$ , wherein  $x_{k,i}$  represents a value corresponding to the  $i$ -th attribute of the  $k$  path. The second step involves using the RSA-OPRF method to convert these attribute sets  $\{T_k\}$  into numbers  $\{Cp_k\}$ . Finally, the owner uploads the  $\{Cp_k\}_{k \in [1, n_{path}]}$  onto a server along with the encrypted data  $Cm$ .

**Algorithm 7** *PrepareCiphertext*

Input: plaintext  $m$ , attribute paths  $\{T_k = \{x_{k,i}\}\}_{k \in [1, n_{path}], i \in [1, x_{k,attr}]}$ ;

Output:  $\{Cp_k\}_{k \in [1, n_{path}]}$  ||  $Cm$ ;

1: The owner selects a random number  $r \in Z_N^*$ ;

2: FOR  $k \in [1, n_{path}]$  DO

3: The owner computes:

$$\{h_{k,i} = H_1(x_{k,i})\}_{i \in [1, n_{k,attr}]}, \{\tilde{x}_{k,i} = h_{k,i}(g^r)^{\tilde{e}}\}_{i \in [1, n_{k,attr}]}$$

4: The owner sends  $\{\tilde{x}_{k,i}\}_{i \in [1, n_{k,attr}]}$  to a server along with random numbers  $\{r_i'\}$ ;

Then the server calculates  $\{\tilde{x}_{k,i} = (\tilde{x}_{k,i})^d\}_{i \in [1, n_{k,attr}]}$  and  $\{r_i'^d\}$ , then sends them back to the owner;

5: The owner filters out  $\{\tilde{x}_{k,i}\}_{i \in [1, n_{k,attr}]}$  and calculates  $\{\tilde{x}_{k,i} = \tilde{x}_{k,i}(g^r)^{-1}\}_{i \in [1, n_{k,attr}]}$ ;

6: The owner verifies  $(\tilde{x}_{k,i})^{\tilde{e}} = h_{k,i}$ . If the verification succeeds, the owner runs a hash function and obtains  $\{\tilde{x}_{k,i} = H_2(\tilde{x}_{k,i})\}_{i \in [1, n_{k,attr}]}$ ;

7: The owner calculates  $Cp_k = \prod_{i \in [1, n_{k,attr}]} \tilde{x}_{k,i}$ ;

8: END FOR

9: The owner selects  $s \in Z_p$ , and calculates:

$$Cm = (C_1, C_2, \{C_k\}_{k \in [1, n_{path}]})$$

$$C_1 = m \cdot (e(g, g)^a)^s,$$

$$C_2 = g^s,$$

$$C_k = g^{b \cdot s / (\sum_{i \in [1, n_{k,attr}]} \tilde{x}_{k,i})}$$

10: The owner uploads  $\{Cp_k\}_{k \in [1, n_{path}]}$  ||  $Cm$  to a server.

## 5.2.4. A User Downloads the Encrypted Data and Decrypts It

Algorithm 8 *Decryption* runs on a user and is used to match a user's attribute paths with an owner's attribute set in secret. If a feasible path is matched, the user can then decrypt and read the encrypted data. However, if none of the paths match, the algorithm will return  $\perp$ , indicating that the user should not attempt to decrypt the data because their attribute set does not meet the access policy.

**Algorithm 8** *Decryption*

Input:  $\{Cp_k\}$  ||  $Cm$ ,  $SK_{user} = (D_0, \{D_j, \tilde{y}_j\})$ ;

Output: The plaintext data  $m$  or  $\perp$ ;

1: Computes  $D_{set} = \prod_{j \in [1, n_{user,attr}]} \tilde{y}_j$ ;

2: FOR  $\{Cp_k\}$  DO

3: FOR  $k \in [1, n_{path}]$  DO

4: IF  $D_{set} \bmod Cp_k = 0$  THEN Jump to Step 6;

5: END FOR

6: IF  $k > n_{path}$  THEN return  $\perp$ ;

7: FOR  $\{\tilde{y}_j\}_{j \in [1, n_{user,attr}]}$  DO

8: IF  $(D_{set} / Cp_k) \bmod \tilde{y}_j \neq 0$  THEN

9: Insert  $\tilde{y}_j$  to the set  $Y_{inset}$ ;

10: ELSE

11: Insert  $\tilde{y}_j$  to the set  $Y_{outset}$ ;

12: END IF

13: END FOR

14: Jump to Step 16;

15: END FOR

16: The plaintext data can be calculated as  $m = C_1 \cdot e(C_2, D_0) / e(C_k, \prod_{\tilde{y}_j \in Y_{inset}} D_j)$ .

### 5.3. More Discussion of CP-ABE-FPH

We integrate the O-PSI method with ROBDD within the CP-ABE-FPH scheme. The integration offers CP-ABE-FPH with the following advantages: (1) Lightweight privacy policy matching of access policy. When decrypting data, privacy policy matching only involves integer operations:  $n_Y E_{mul} + (1 + n_Y) E_{mod} + E_{div}$ . This operation is highly efficient because it does not require pairs. (2) High expressiveness of access policy. The CP-ABE-FPH scheme offers rich expressiveness of access policies due to its construction based on the ROBDD method. (3) High flexibility of users. The owner does not have to be online all the time, and multiple users can decrypt the same ciphertext simultaneously. The owner can be offline after uploading the  $\{Cp_k\} || Cm$ . Users may include anyone, such as Cindy, Denis, and so on. The proposed scheme has excellent one-to-many capabilities.

The scalability of CP-ABE schemes is also one of the key considerations in large-scale deployment. On the one hand, system attributes can be added at any time in the proposed CP-ABE-FPH. When the system attribute is added, the *KeyGen* algorithm and *PrepareCiphertext* algorithm can generate the paired  $x_{k,i}$  and  $y_j$ , and there is no need to rerun the algorithm *Initial*. On the other hand, a user attribute can be revoked efficiently. When an attribute of a user is revoked, to prevent the user from accessing the ciphertext bound to the attribute, all the owners who generate the ciphertext need to reselect the random number  $r$ , recalculate  $C_k$ , and upload it to the cloud server. Then, the cloud server can update  $C_k$ .

Next, the proposed CP-ABE-FPH scheme is suitable for the data security sharing scenario of *Thin – User and Fat – Owner*, such as the big data industry. The CP-ABE-FPH scheme supports *Thin – User* because users only need two pairing operations when decrypting. This is an advantage gained at the expense of the owner spending a lot of time preparing the ciphertext. So, it is called the *Fat – Owner*.

Compared with other fully policy-hidden methods, the proposed 2-way handshake O-PSI method makes this CP-ABE-FPH scheme not require participants to be online all the time and also conceals the number of attributes in access policies. This advantage allows the CP-ABE-FPH scheme to fully hide the access policy.

## 6. Security Analysis

### 6.1. Leakage Analysis of 2-Way Handshake O-PSI

Three leakage functions are defined to formally analyze the security of 2-way handshake O-PSI.

$$\begin{aligned} L^{PrepareforSetX} &= (\{\tilde{x}_i\}_{i \in [1, n_X]} \cup \{r_{i'}\}, \{(\tilde{x}_i)^d\}_{i \in [1, n_X]} \cup \{r_{i'}^d\}, Cp), \\ L^{PrepareforSetY} &= (\{\tilde{y}_j\}_{j \in [1, n_Y]} \cup \{r_{j'}\}, \{(\tilde{y}_j)^d\}_{j \in [1, n_Y]} \cup \{r_{j'}^d\}), \\ L^{Setintersection} &= Cp. \end{aligned}$$

Here,  $n_X$  is the cardinality of the set  $X$ ,  $n_Y$  is the cardinality of the set  $Y$ , and  $Cp = \prod_{i \in [1, n_X]} \tilde{x}_i$ . It can be observed that (1) the information leaked by function  $L^{PrepareforSetY}$  resembles the first two parts of the information leaked by function  $L^{PrepareforSetX}$ ; (2) the information leaked by function  $L^{PrepareforSetX}$  includes the information leaked by function  $L^{Setintersection}$ .

**Definition 1.** Let  $\Omega = (Initial, PrepareforSetX, PrepareforSetY, Setintersection)$  be the proposed 2-way handshake O-PSI. Given leakage functions  $L^{PrepareforSetX}$ ,  $L^{PrepareforSetY}$  and  $L^{Setintersection}$ , the following probabilistic experiments are defined:  $Real_{\Lambda}^{\Omega}(\lambda)$  and  $Ideal_{\Lambda}^{\Omega}(\lambda)$  with a probabilistic polynomial time adversary  $\Lambda$  and a PPT simulator  $\mathcal{S}$ . The information leaked by the function  $L^{PrepareforSetY}$  is similar to the first two parts leaked by the function  $L^{PrepareforSetX}$ . For simplicity, the description of the leaked information for  $L^{PrepareforSetY}$  is omitted below.

$Real_{\Lambda}^{\Omega}(\lambda)$ : The challenger calls the algorithm *Initial* to generate and generates a hidden set (called HS) for a party  $P2$  according to their attributes set.  $\Lambda$  selects a set  $X = \{x_i\}_{i \in n_X}$ , generates  $\{\tilde{x}_i\}_{i \in [1, n_X]} \cup \{r_{i'}\}$ , and obtains  $\{(\tilde{x}_i)^d\} \cup \{r_{i'}^d\}$  from a cloud server. Next,  $\Lambda$  computes  $Cp$  and

sends it to the server. Then,  $\Lambda$  adaptively constructs a polynomial number of queries with different sets of  $X$  via the algorithm  $PrepareforSetX$ . Finally,  $\Lambda$  returns a bit as the output.

$Ideal_{\Lambda}^{\Omega}(\lambda)$ :  $\Lambda$  selects a set  $X = \{x_i\}_{i \in n_X}$ , and  $\mathcal{S}$  simulates  $\{\bar{x}_i\}_{i \in [1, n_X]} \cup \{r_{i'}\}, \{(\bar{x}_i)^d\} \cup \{r_{i'}^d\}$  for  $\Lambda$  based on  $L^{PrepareforSetX}$ . Then,  $\Lambda$  adaptively constructs a polynomial number of queries.  $\mathcal{S}$  simulates  $Cp$  from  $L^{Setintersection}$  or  $L^{PrepareforSetX}$ . Finally,  $\Lambda$  returns a bit as the output.

$\Omega$  is a  $(L^{PrepareforSetX}, L^{PrepareforSetY}, L^{Setintersection})$ -secure scheme, if for all PPT adversaries  $\Lambda$ , there exists a simulator  $\mathcal{S}$  such that  $\Pr[\text{Real}_{\Lambda}^{\Omega}(\lambda) = 1] - \Pr[\text{Ideal}_{\Lambda}^{\Omega}(\lambda) = 1] \leq \text{negl}(\lambda)$ , wherein  $\text{negl}(\lambda)$  is a negligible function in  $\lambda$ .

**Theorem 1.**  $\Omega$  is an adaptively secure scheme with  $(L^{PrepareforSetX}, L^{PrepareforSetY}, L^{Setintersection})$  leakages under the random-oracle mode if the RSA-OPRF (RSA-Based Oblivious Pseudo-Random Function) is secure.

**Proof.** Random oracles are defined:  $H_{H_1}$  and  $H_{H_2}$ . From  $L^{PrepareforSetX}$ , the simulator  $\mathcal{S}$  selects a random number that has the same size as the real one  $\{\bar{x}_i\}_{i \in [1, n_X]} \cup \{r_{i'}\}$ . When the first query  $\{\bar{x}_i\}_{i \in [1, n_X]} \cup \{r_{i'}\}$  is sent,  $\mathcal{S}$  generates simulated  $\{\hat{x}_i = (\bar{x}_i)^d\} \cup \{r_{i'}^d\}$  and  $Cp$  from  $L^{PrepareforSetX}$ . If the  $\Lambda$  sends a query appearing before,  $\mathcal{S}$  returns the same result back. Due to RSA-OPRF,  $\Lambda$  cannot distinguish simulated random numbers from  $\{\bar{x}_i\} \cup \{r_{i'}\}$  and  $\{(\bar{x}_i)^d\} \cup \{r_{i'}^d\}$  generated by  $\text{Real}_{\Lambda}^{\Omega}(\lambda)$ .  $\square$

## 6.2. Security Analysis of the CP-ABE-FPH Scheme

The CP-ABE-FPH scheme is constructed based on the proposed 2-way handshake O-PSI and CP-ABE. The security of the 2-way handshake O-PSI is extensively discussed in the above section. The encryption and decryption process of the CP-ABE scheme follows traditional methods, with guaranteed security. Consequently, this article focuses solely on the information leakage of the scheme.

The cloud server cannot obtain any  $x_{k,i}$  or  $y_j$ . Since  $r_y$  and  $r_x$  are randomly chosen by KGC and the owner, respectively, a server cannot obtain the value of  $x_{k,i}$  or  $y_j$  from  $\{\bar{x}_{k,i}\}$  or  $\{\bar{y}_j\}$ . Additionally, although the  $Cp_k$  is stored in the server, the server still does not obtain the value or number of  $x_{k,i}$  from the  $Cp_k$ .

Users cannot obtain any  $x_{k,i}$  of  $Y_{outset}$ . Firstly, a user cannot obtain any  $x_{k,i}$  from the  $Cp_k$ . Secondly, the computations of  $D_{set} \bmod Cp_k$  do not expose  $x_{k,i}$ , and at the same time, the computation of  $D_{set} / Cp_k$  does not expose  $x_{k,i}$ . The user does not even know which  $\bar{y}_j$  corresponds to which  $y_j$  unless they try one by one.

On the whole, the feasible paths, standing for an access policy by the owner on the ciphertext, are encrypted using the RSA-OPRF. They thus conceal any information. Subsequently, the feasible paths can be matched by adopting the proposed 2-way handshake O-PSI method. The matching process does not give away information. Additionally, a user cannot access attributes they do not own because the 2-way handshake O-PSI prevents one from knowing the cardinality of the feasible paths.

## 7. Efficiency Analysis

This section analyzes the efficiency of the proposed 2-way handshake O-PSI method. The computational costs of each algorithm are listed in Table 3.  $n_X$  and  $n_Y$  denote the number of elements of the set  $X$  and the number of elements of the set  $Y$ , respectively.  $E_H, E_{ex}, E_{mul}, E_{mod}, E_{div}, E_p$  denote the computational cost on Hash, Exponent, Multiplication, Modulo, Division, and Bilinear Pair Map. For simplicity, we ignore the cost of computation and communication caused by extra elements  $\{\bar{r}_j^d\}$  or  $\{r_{i'}^d\}$  doped to hide the two sets' cardinality.

To present effectiveness analysis more vividly, these various operations are simulated on Ubuntu with two processors, each with three cores and 8GB memory. GMP (the GNU Multiple Precision Arithmetic Library) and PBC (the pairing-based cryptosystems) are used. The operations are simulated on a computer running Ubuntu with two processors,



each with three cores, and 8GB of memory. The simulation uses GMP (the GNU Multiple Precision Arithmetic Library) and PBC (the pairing-based cryptosystems). The simulation shows that the costs taken for various operations are as follows:  $E_H$ : 3.12 ms,  $E_{ex}$ : 0.35 ms,  $E_{mul}$ : 0.003 ms,  $E_{mod}$ : 0.0002 ms,  $E_{div}$ : 0.0001 ms,  $Ep$ : 1.02 ms.

**Table 3.** The computational costs on each algorithm in 2-way handshake O-PSI method.

Algorithms	Entities	2-Way Handshake O-PSI
Setup	KGC	$2n_Y E_H + 3n_Y E_{ex} + 2n_Y E_{mul} + n_Y E_{mod}$
	Server	$n_Y E_{ex}$
PrepareforSetX	P2	$2n_X E_H + 3n_X E_{ex} + 3n_X E_{mul} + n_X E_{mod}$
	Server	$n_X E_{ex}$
Setintersection	KGC	$n_Y E_{mod}$

Table 4 compares the computational costs in CP-ABE-FPH with others, wherein  $t$ ,  $n$ , and  $l$  denote the number of hash values designed in a Path Bloom Filter and the number of attributes in the whole system, the number of attributes managed by AA.

Table 5 presents a comparison of communication costs between Xu et al. [15] and the proposed 2-way handshake O-PSI method. Let  $|G|$  and  $|Z_N|$  be the element's size in  $G_1$  and  $Z_N$ . Let  $n_a$ ,  $n_i$ ,  $n_X$ , and  $n_Y$  be the number of attributes of user  $i$ , the universal attributes number, the number of elements of the set  $X$ , and the number of elements of the set  $Y$ , respectively. For the sake of simplicity, let the  $|G_T|$  be the same as  $|G|$  in Xu et al. [15]. In the proposed 2-way handshake O-PSI, the main communication costs between KGC and Server are  $\{\tilde{y}_j\}_{j \in [1, n_Y]}$  and  $\{\hat{y}_j\}_{j \in [1, n_Y]}$ . The communication cost between KGC and P2 is the set  $X$ . The communication cost between KGC and P1 is  $SK = \{\tilde{y}_j\}_{j \in [1, n_Y]}$ . The communication costs between P2 and Server are  $\{\tilde{x}_i\}_{i \in [1, n_X]}$ ,  $\{\hat{x}_i\}_{i \in [1, n_X]}$ , and  $Cp = \prod_{i \in [1, n_X]} \tilde{x}_i$ . The communication cost between P1 and Server is  $Cp$ .

**Table 4.** Comparison of computing costs between CP-ABE-FPH and others.

Algorithms	Entities	CP-ABE-FPH	Luo et al. [8]
KeyGen	KGC	$2n_{user,attr} E_H + 2n_{user,attr} E_{mul} + n_{user,attr} E_{mod} + (1 + 4n_{user,attr}) E_{ex}$	$(2n_{user,attr} + 2n + 3)Ep$
	Server	$n_{user,attr} E_{ex}$	-
Prepare	P2	$2\sum n_{k,attr} E_H + 3(1 + \sum n_{k,attr}) E_{ex} + 3\sum n_{k,attr} E_{mul} + \sum n_{k,attr} E_{mod}$	$(2n + 6)Ee + l t E_H$
Ciphertext	Server	$\sum n_{k,attr} E_{ex}$	-
Decryption	User	$(1 + n_{user,attr}) E_{mul} + k E_{div} + 2Ep + k(1 + n_{user,attr}) E_{mod}$	$(1 + 2n_{user,attr}) Ep + n_{user,attr} t E_H$

**Table 5.** Communication costs in the 2-way handshake O-PSI method.

Entities	Xu et al. [15]	2-Way Handshake O-PSI Method
KGC ↔ Server	0	$2n_Y  G $
KGC ↔ P2	$(3 + n_a)  G $	$ Z_N $
KGC ↔ P1	$(2 + n_i)  G $	$n_Y  G $
P2 ↔ Server	$(4m + 4)  G $	$(2n_X + 1)  G $
Server ↔ P1	$(4m + 5)  G $	$ G $

The storage costs on each entity are listed in Table 6.  $|G|$  denotes the storage cost of an element in group  $G$ .  $|Params|$  denotes the storage cost of public parameters. P2 stores  $\{\tilde{x}_i = H_2(\tilde{x}_i)\}_{i \in [1, n_X]}$ , Server stores  $Cp = \prod_{i \in [1, n_X]} \tilde{x}_i$  and public parameters, and P1 stores  $HS = \{\tilde{y}_j\}_{j \in [1, n_Y]}$ . P2 can also not store public parameters locally, in which case it needs to be temporarily fetched from KGC. The storage cost analysis just focuses on the storage cost caused by each set computation.

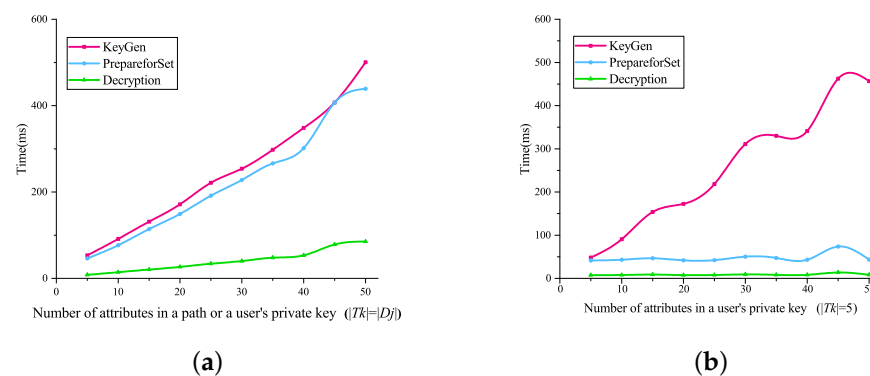
**Table 6.** The storage costs on each entity in 2-way handshake O-PSI.

Entities	Storage Cost
P2	$n_X  G  +  Params $
P1	$n_Y  G $
Server	$ G $

## 8. Simulation of CP-ABE-FPH

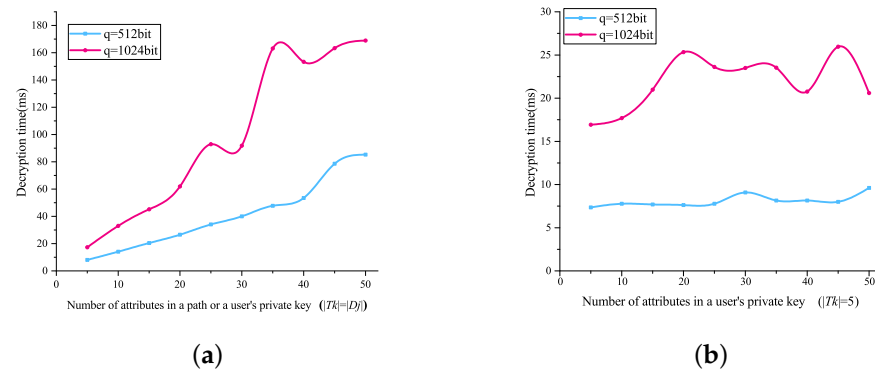
The PBC library is used to simulate the proposed CP-ABE-FPH scheme on Ubuntu 20.04 (4 Cores, 8 GB RAM). Pairing parameters are generated according to Type A. The group order is 160 bits long, and  $q$  (the order of the base field) is 512 bits or 1024 bits long. Elements take 512 bits to represent. The value of  $\bar{k}$  is set to 50. The related code can be downloaded from Github (the source code for the implementation can be found at <https://github.com/shijiaoli/CP-ABE-FPH>, accessed on 18 June 2024).  $|Tk|$  denotes the number of attributes in a path generated by an access policy.  $|Dj|$  denotes the number of attributes in a user's private key.

Figure 6a,b illustrates the computing costs of the core algorithms of CP-ABE-FPH when  $q = 512$  bit. The difference is that  $|Tk|$  and  $|Dj|$  grow synchronously in Figure 6a, while Figure 6b fixes  $|Tk|$  to be 5 and  $|Dj|$  is growing. As can be seen, the decryption costs grow slowly, and it is almost a constant quantity in Figure 6b. Figure 6a shows that all core algorithms naturally increase with the number of attributes. Algorithm *Decryption* is low-cost because it just needs to perform 2 pairing operations. By comparing Figure 6a,b, it can be found that the cost of the *PrepareCiphertext* algorithm is strongly related to  $|Tk|$ .



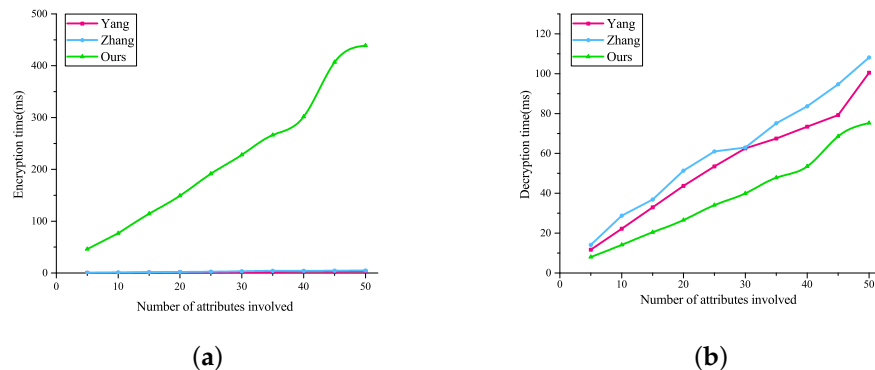
**Figure 6.** The computational costs of the core algorithms of CP-ABE-FPH when  $q = 512$  bits.

Figure 7a,b shows that the decrypt time is independent of the number of attributes in a user's private key and naturally increases with the number of attributes in  $Tk$  when  $q = 512$  bit. The difference between Figure 7a,b is that  $|Tk|$  and  $|Dj|$  grow synchronously in Figure 7a, while Figure 7b fixes  $|Tk|$  to be 5 and  $|Dj|$  is growing. But even with a large attribute set (e.g.,  $|Tk| = |Dj| = 50$ ), Figure 7a shows that the decryption operations can still be completed within tens of milliseconds when  $q = 512$  bit.



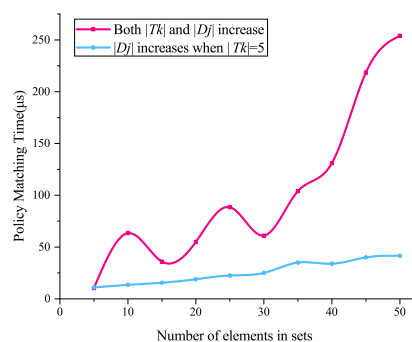
**Figure 7.** The decryption costs for  $q = 512$  bit or  $1024$  bit in CP-ABE-FPH.

Figure 8a,b compares the proposed CP-ABE-FPH scheme with Zhang et al. [2] and Yang et al. [7] when  $q = 512$  bit. To simplify,  $k$  is assumed to be 1 in Zhang et al. [2], and the number of attributes in the access structure is the same as the number of attributes in the user's private key in all schemes. The results show that the proposed CP-ABE-FPH scheme has low efficiency in encryption. This is because the CP-ABE-FPH scheme involves two successive steps (ROBDD and RSA-ORPF) in encryption compared with other schemes. The encryption time can be reduced by calculating and storing the  $Cp_k$  in advance. In this way, the encryption time is small and becomes constant. The results also show that the proposed CP-ABE-FPH scheme is highly efficient in decryption. This is because the CP-ABE-FPH scheme requires only two pairing operations for decryption, and pairing operations are notoriously time-consuming.



**Figure 8.** The comparison of En/Decryption costs between CP-ABE-FPH and others when  $q = 512$  bit.

Figure 9 simulates the computation spent on policy-matching judgments in the privacy set computation phase when  $q = 512$  bit. Like the other simulations, we also simulated two situations by controlling  $|T_k|$  and  $|D_j|$ . One is that  $|T_k|$  and  $|D_j|$  grow at the same time, and the other is that  $|T_k|$  is fixed and  $|D_j|$  grows. In Figure 9, it is observed that the time spent on privacy set computation is just a few tens or hundreds of  $\mu\text{s}$ . Although the efficiency of privacy set computation is linear with the number of elements in the set, it is still insignificant compared with the other two algorithms. The experimental results show that the proposed scheme is well suited for environments with limited terminal computing capacity, such as IoT environments.



**Figure 9.** The comparison of policy matching costs between two cases in CP-ABE-FPH when  $q = 512$  bit.

## 9. Conclusions

In this paper, a 2-way outsourcing PSI method is introduced utilizing RSA-based Oblivious Pseudo-Random Function (RSA-OPRF). In the O-PSI, party P2 encrypts their set and uploads it to a cloud server. The server then stores P2's encrypted set and provides services to other parties as needed. Party P1 can compute the intersection of P2 and their own set after retrieving the encrypted set from the cloud server. The proposed O-PSI approach reduces the number of communications from three to two while keeping the elements and sizes of the sets confidential.

The CP-ABE-FPH scheme combines the 2-way handshake O-PSI method with the ROBDD method. This scheme integrates data confidentiality protection, expressiveness of access policy, fully policy-hidden, fine-grained access control, and lightweight terminal computing. The proposed CP-ABE-FPH scheme offers three advantages: (1) High efficiency and expressiveness, enabling rapid privacy policy matching and rich expressiveness of access policies. (2) Fully policy-hidden, ensuring that the access policy associated with the ciphertext does not reveal any information and users cannot access attributes they do not possess. (3) High flexibility, allowing owners or KGC to not always be online. Additionally, system attributes can be added at any time, further enhancing the flexibility of the system.

**Author Contributions:** For this work, conceptualization and discussion took place between Jiaoli Shi, Shimao Yao, and Chao Hu. Investigation was conducted as a wide scan of the formal literature and discussed online by all authors. Original draft preparation was conducted by Jiaoli Shi. Qing Zhou and Zhuolin Mei led the validation. Shunli Zhang and Anyuan Deng supervised the paper. Review and editing were carried out by all authors equally. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was supported by the National Science Foundation of China (No. 62062045, No. 62341206, No. 61962029 and No. 62462054).

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** This paper proposes a data security scheme that does not involve specific data content. Therefore, only the computational, communication, and storage costs of the security scheme need to be evaluated. The source code for the implementation can be downloaded from Github at <https://github.com/shijiaoli/CP-ABE-FPH>, accessed on 18 June 2024.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

- Georgiadou, Y.; de By, R.A.; Kounadi, O. Location Privacy in the Wake of the GDPR. *ISPRS Int. J. Geoinf.* **2019**, *8*, 157. [CrossRef]
- Zhang, Z.; Zhang, J.; Yuan, Y.; Li, Z. An Expressive Fully Policy-Hidden Ciphertext Policy Attribute-Based Encryption Scheme With Credible Verification Based on Blockchain. *IEEE Internet Things J.* **2021**, *9*, 8681–8692. [CrossRef]

3. Lai, J.; Deng, R.H.; Li, Y. Fully Secure Ciphertext-Policy Hiding CP-ABE. In Proceedings of the 7th International Conference on Information Security Practice and Experience, Guangzhou, China, 30 May–1 June 2011; pp. 24–39.
4. Müller, S.; Katzenbeisser, S. Hiding the Policy in Cryptographic Access Control. In Proceedings of the 7th International Workshop on Security and Trust Management, Copenhagen, Denmark, 27–28 June 2011; pp. 90–105.
5. Hur, J. Attribute-Based Secure Data Sharing with Hidden Policies in Smart Grid. *IEEE Trans. Parallel Distrib. Syst.* **2013**, *24*, 2171–2180. [[CrossRef](#)]
6. Phuong, T.V.X.; Guomin, Y.; Susilo, W. Hidden Ciphertext Policy Attribute-Based Encryption Under Standard Assumptions. *IEEE Trans. Inf. Forensics Secur.* **2016**, *11*, 35–45. [[CrossRef](#)]
7. Yang, K.; Han, Q.; Li, H.; Zheng, K.; Su, Z.; Shen, X. An Efficient and Fine-Grained Big Data Access Control Scheme with Privacy-Preserving Policy. *IEEE Internet Things J.* **2017**, *4*, 563–571. [[CrossRef](#)]
8. Luo, C.; Shi, J.; Xie, M.; Hu, C.; Wang, L.; Mei, Z.; Yao, S.; Li, H. A Lightweight Access Control Scheme Supporting Policy Hidden Based on Path Bloom Filter. In Proceedings of the 19th International Conference on Information Security and Cryptology (Inscrypt), Hangzhou, China, 9–10 December 2023; pp. 433–451.
9. Affum, E.; Zhang, X.; Wang, X. Lattice CP-ABE Scheme Supporting Reduced-OBDD Structure. In Proceedings of the International Conference on Computer, Communication and Computational Sciences, Bangkok, Thailand, 11–12 October 2019; pp. 131–142.
10. Morales, D.; Agudo, I.; Lopez, J. Private Set Intersection: A Systematic Literature Review. *Comput. Sci. Rev.* **2023**, *49*, 1–24. [[CrossRef](#)]
11. Zhao, Y.; Chow, S.S.M. Can You Find The One For Me? In Proceedings of the Workshop on Privacy in the Electronic Society (WPES@CCS), Toronto, ON, Canada, 15–18 October 2018; pp. 54–65.
12. Thapa, A.; Ming, L.; Salinas, S.; Pan, L. Asymmetric Social Proximity Based Private Matching Protocols for Online Social Networks. *Trans. Parallel Distrib. Syst.* **2015**, *26*, 1547–1559. [[CrossRef](#)]
13. Sun, J.; Zhou, F.; Wang, Q.; Jiao, Z.; Zhang, Y. Flexible Revocation and Verifiability for Outsourced Private Set Intersection Computation. *J. Inf. Secur. Appl.* **2023**, *73*, 1–16. [[CrossRef](#)]
14. Li, F.; He, Y.; Niu, B.; Li, H.; Wang, H. Match-MORE: An Efficient Private Matching Scheme Using Friends-of-Friends’ Recommendation. In Proceedings of the International Conference on Computing, Networking and Communications (ICNC), Kauai, HI, USA, 15–18 February 2016; pp. 1–6.
15. Xu, R.; Joshi, J.; Krishnamurthy, P. An Integrated Privacy Preserving Attribute-Based Access Control Framework Supporting Secure Deduplication. *IEEE Trans. Dependable Secur. Comput.* **2019**, *18*, 706–721. [[CrossRef](#)]
16. Zhang, M.; Shao, F.; Zheng, R.; Liu, M.; Ji, Z. An Efficient Encryption Scheme with Fully Hidden Access Policy for Medical Data. *Electronics* **2023**, *12*, 2930. [[CrossRef](#)]
17. Sotiraki, K.; Ghosh, E.; Chen, H. Privately Computing Set-maximal Matches in Genomic Data. *BMC Med Genom.* **2020**, *13*, 72. [[CrossRef](#)] [[PubMed](#)]
18. Cheng, J.; Liu, N.; Kang, W. On the Asymptotic Capacity of Information-Theoretic Privacy-Preserving Epidemiological Data Collection. *Entropy* **2023**, *25*, 625. [[CrossRef](#)] [[PubMed](#)]
19. Li, C.; Lin, B. Privacy-Preserving Point-Inclusion Two-Party Computation Protocol. In Proceedings of the International Conference on Computational and Information Sciences, Shiyuan, China, 21–23 June 2013; pp. 1–4.
20. Feng, J.; Yang, L.T.; Zhu, Q.; Choo, K.-K.R. Privacy-Preserving Tensor Decomposition Over Encrypted Data in a Federated Cloud Environment. *IEEE Trans. Dependable Secur. Comput.* **2020**, *17*, 857–868. [[CrossRef](#)]
21. Feng, J.; Yang, L.T.; Ren, B.; Zou, D.; Dong, M.; Zhang, S. Tensor Recurrent Neural Network With Differential Privacy. *IEEE Trans. Comput.* **2024**, *73*, 683–693. [[CrossRef](#)]
22. Keelveedhi, S.; Bellare, M.; Ristenpart, T. DupLESS: Server-Aided Encryption for Deduplicated Storage. In Proceedings of the 22nd USENIX Security Symposium, Washington, DC, USA, 14–16 August 2013; pp. 179–194.
23. Long, L.; Tianlong, G.; Liang, C.; Zhoubo, X.; Junyan, Q. Expressive Ciphertext-policy Attribute-based Encryption Scheme with Fast Decryption and Constant-size Secret Keys. *J. Electron. Inf. Technol.* **2018**, *40*, 1661–1668.
24. Zhong, H.; Zhou, Y.; Zhang, Q.; Xu, Y.; Cui, J. An Efficient and Outsourcing-Supported Attribute-Based Access Control Scheme for Edge-Enabled Smart Healthcare. *Future Gener. Comput. Syst.* **2021**, *115*, 486–496. [[CrossRef](#)]

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.