Sharing and Privacy in PHRs: Efficient Policy Hiding and Update Attribute-based Encryption

Zhenhua Liu[1], Jiaqi Ji[1*], Fangfang Yin[1], and Baocang Wang[2]
[1] School of Mathematics and Statistics, Xidian University
Xi'an, Shaanxi, 710071, China
[e-mail: zhualiu@hotmail.com, jiaqiluck@163.com, yinff21@163.com]
[2] State Key Laboratory of Integrated Services Networks, Xidian University
Xi'an, 710071, China
[e-mail: bcwang79@aliyun.com]
*Corresponding author: Jiaqi Ji

Received September 17, 2020; revised December 8, 2020; accepted January 14, 2021; published January 31, 2021

Abstract

Personal health records (PHRs) is an electronic medical system that enables patients to acquire, manage and share their health data. Nevertheless, data confidentiality and user privacy in PHRs have not been handled completely. As a fine-grained access control over health data, ciphertext-policy attribute-based encryption (CP-ABE) has an ability to guarantee data confidentiality. However, existing CP-ABE solutions for PHRs are facing some new challenges in access control, such as policy privacy disclosure and dynamic policy update. In terms of addressing these problems, we propose a privacy protection and dynamic share system (PPADS) based on CP-ABE for PHRs, which supports full policy hiding and flexible access control. In the system, attribute information of access policy is fully hidden by attribute bloom filter. Moreover, data user produces a transforming key for the PHRs Cloud to change access policy dynamically. Furthermore, relied on security analysis, PPADS is selectively secure under standard model. Finally, the performance comparisons and simulation results demonstrate that PPADS is suitable for PHRs.

Keywords: Attribute-based Encryption, Attribute Bloom Filter, Personal Health Records, Policy Hiding, Policy Update

This work is supported by the National Key R&*D* Program of China under GrantNo.2017YFB0802000, the National Natural Science Foundation of China under Grants No.61807026, the Natural Science Basic Research Plan in Shaanxi Province of China under Grant No.2019JM-198, the Plan for Scientific Innovation Talent of Henan Province under Grant No.184100510012, and in part by the Program for Science and Technology Innovation Talents in the Universities of Henan Province under Grant No.18HASTIT022.

http://doi.org/10.3837/tiis.2021.01.018 **ISSN : 1976-7277**

1. Introduction

Personal health records (PHRs) [1] is a system that allows medical staffs and patients to retrieve PHR information in a timely way via any smart wearable devices (SWDs) [2]. As a major fundamental service, cloud storage [3] possessing powerful computation and data storage capabilities is very suitable for PHRs. Particularly in PHRs based on cloud storage, a patient only needs to upload her/his electronic health records to PHRs rather than submitting paper medical records. Whereas, PHRs still involves too much practical application concerns that have not been addressed.

Attribute-based encryption (ABE) [4] realized fine-grained access control was classified into CP-ABE [5] and key-policy attribute-based encryption (KP-ABE) [6]. Utilizing CP-ABE, which can well meet the security requirements in PHRs, patients can share their encrypted electronic health records embedded with an access policy with others. Medical staffs could decrypt correctly while her/his attribute sets were in accord with the access policy. Generally, a ciphertext of CP-ABE with a plaintext form of access structure involving user attribute can be accessed by anyone. Therefore, it is not suitable for PHRs. For example, a patient needs to consult with medical staff about medical records of a Psychiatry Department of Medical Institution 1 or 2. The patient can encrypt medical information with an access structure {[Department: ("Psychiatry")] AND [Medical Institution: ("1" OR "2")]} and upload it to PHRs Cloud (PHRC). Under the circumstance, anyone can access the ciphertext even if she or he can't decrypt it, but she or he can infer that the patient might suffer from a mental illness. Therefore, the privacy of the patient is violated and policy hiding plays a crucial role in CP-ABE. Policy hiding can be divided into two types: full hiding and partial hiding. Attribute in the access structure can be concealed in full hiding policy CP-ABE. But in a partial hiding access policy, only partial attribute information is hidden. Specifically, attribute includes two portions: attribute name and attribute value, and partial hiding simply conceals attribute value. Note that full hiding CP-ABE has a more adequate ability to ensure attribute privacy. As described in the previous example, located in a partial hiding scheme, an attacker can utilize the captured ciphertext to detect that the patient was in the fixed department of the hospital to seek medical advice, while it is impossible to acquire any information in a full hiding scheme.

To our knowledge, most of ABE schemes can encrypt message with static access policy, but the patient's medical record information needs to be modified at any time in PHRs. Traditionally speaking, the patient has to decrypt an original ciphertext to obtain plaintext, then encrypt plaintext with a new structure, and upload a new ciphertext to PHRC, which undoubtedly increases the computation cost and communication consumption. Therefore, it is meaningful to research on policy update that outsources a ciphertext update to PHRC.

2. Related Work

Nowadays, a growing number of people hope their health care will be protected prudently. The connection of IoT and cloud computing with PHRs is widely used, which will generate massive medical data. Since the data scale of medical IoT is huge, some traditional encryption technologies are difficult to manage and process them effectively. PHRs involving IoT and clouding computing were proposed in Xu et al. [7] and Namani et al. [8].

Nevertheless, above environment cannot refer to data confidentiality or privacy issues in PHRs. ABE, as a primitive, gives a positive solution of data confidentiality in PHRs. Sahai and Waters [4] introduced the ABE concept firstly. Along with the further improvement of ABE, there exists two basic types: KP-ABE [9] and CP-ABE [10]. Furthermore, according to

the form of expression, access policy can be classified into the three types: AND-based [11], tree-based [9] and LSSS-based [12]. Additionally, many other efficient and functional ABE solutions have been put forward [13-15]. But these schemes cannot involve user privacy and dynamic update, and thus cannot been applied directly on PHRs.

To protect user attribute privacy, a sequence of privacy protection schemes have been presented [16, 17]. Generally, policy hiding consists of two types: partial hiding and full hiding. The concept of partial hiding was presented by Nishide et al. [11], where the attribute value is hidden. To improve Nishide et al.'s scheme [11], Lai et al. [18] proposed a concrete construction supporting multi-valued attributes with wildcards. But their schemes only support AND-gate policy. Subsequently, an improved composite-order scheme with expressive LSSS was presented by Lai et al. [19]. Nonetheless, above schemes are limited for composite-order groups, since the size of composite-order group is bigger than prime order that guarantees an equivalent level of security. Later, Cui et al. [20] constructed an efficient scheme with partially hiding access policy based on linear secret sharing scheme (LSSS) in prime-order groups. Furthermore, attribute value is hidden by wildcards [21] and inner product encryption (IPE) [22]. To some extent, hiding attribute value can protect privacy, but attribute name can still reveal user information. Afterwards, Michalevsky et al. [23] utilized IPE and Khan et al. [24] took advantage of hidden vector encryption to guarantee a stronger privacy protection, but some shortcomings in efficiency and expressiveness were still existed. Hao et al. [25] constructed a full hiding attribute CP-ABE, but their scheme only possessed one specific functionality of policy hiding.

To decrease the computation burden and communication overhead, Sahai et al. [26] presented a method utilizing ciphertext authorization to update access structure, but restricted that a new policy was more restrained than previous structure. Later on, Yang et al. [27] presented a variety of policy update mechanisms for various access structures, where these structures could be converted into LSSS matrix. Then Zhang et al. [28] came up with a new policy update method, which was proved secure based on the standard model. However, they utilized the composite order groups. Later, Ying et al. [29] put forward a modified policy update for PHRs, where data owner needed to generate the update component and outsourced to PHRC. Whereas, it is possible to increase the amount of computation for data owner to some extent. Yuan [30] showed a fresh LSS matrix update algorithm, which was a novel way to update the policy, but had low efficiency.

2.1 Our Contributions

In this paper, we recommend PPADS to resolve both data confidentiality and user privacy in PHRs. In PPADS, we present a solution focused on policy hiding and policy update. The policy is fully hidden by hiding the whole attribute and the attribute is hidden by attribute bloom filter, which plays a role for locating row number of the LSSS matrix about the attribute and restoring the corresponding attribute mapping function. As far as policy update is concerned, the patient generates a transforming key and uploads it to PHRC, then PHRC updates the corresponding ciphertext with the transforming key. Our rigorous security proofs and performance comparisons indicate that PPADS is selectively secure. Thus, as shown below are our contributions:

- In PPADS, the whole attribute in access policy can be hidden rather than attribute value, thus the ciphertext does not disclose any user privacy information. Then the attribute can be located and recovered by a fuzzy attribute location mechanism.

- Furthermore, to support a flexible data sharing mechanism, the patient needs to update an old policy to a new one. Considering three scenarios, the patient brings out a transforming key and sends it to PHRC, and then PHRC can update the old corresponding ciphertext to the new ciphertext.
- Through comparing efficiency and functional diversification, the final result shows that PPADS can achieve a stronger privacy protection and smaller computing storage.

2.2 Organization

Our paper is distributed as below. We will describe the preliminary knowledge in Section 3. In Section, the detailed procedure of our system is proposed in Section 4. Finally, we provide a security proof and performance analysis comparisons in Section 5 and Section 6, then make a summary in Section 7.

3. Preliminaries

The definitions of bilinear pairing, decisional q-parallel bilinear Diffie-Hellman exponent (BDHE) problem, linear secret sharing scheme (LSSS), and Bloom filter are given in this part.

3.1 Bilinear Pairing

Note G and G_T as two cyclic multiplicative groups with prime order p, and g can be regarded as a generator of G. A bilinear pairing is a map [12] $e: G \times G \to G_T$, which satisfies the following characters:

1) Bilinearity: $\forall u, v \in G$, and $x, y \in Z_p^*$, $e(u^x, v^y) = e(u,v)^{xy}$ holds.

2) Non-degeneracy: $e(g,g) \neq 1$.

3) Computability: On the basis of $u, v \in G$, there has an ability to calculate $e(u,v)$.

3.2 Decisional q-BDHE Assumption

The decisional q-parallel BDHE problem is described as below. Given a group G with prime order p, where g is a generator of G. Furthermore, if an adversary $\mathcal{A}$ is put $\vec{y} = \left\{g, g^s, g^a, \cdots, g^{a^q}, g^{a^{q+2}}, \cdots, g^{a^{2q}}\right\}$, where $a, s \in Z_p^*$, the value $e(g,g)^{a^{q+1}s} \in G_T$ and a random element $Z \in G_T$ need to be distinguished. An adversary $\mathcal{A}$ has advantage ε in attacking decisional q-BDHE [12] while

$$\left|\Pr\left[\mathcal{A}\left(\vec{y}, e(g,g)^{a^{q+1}s}\right) = 0\right] - \Pr[\mathcal{A}(\vec{y}, Z) = 0]\right| \geq \varepsilon \tag{1}$$

Definition 1. *The decisional q-BDHE hardness assumption holds, while no polynomial time adversary $\mathcal{A}$ has a non-negligible advantage in resolving the decisional q-BDHE problem.*

3.3 Linear Secret Sharing Scheme

A linear secret sharing scheme [12] is applied to a structure by (M,ρ), where M is an access matrix with the size of $l \times n$ and ρ is an injective function which maps each row of M to an attribute. Specifically, there are two algorithms:

- **Share** $((M,\rho),S)$**:** The subalgorithm is applied to distribute a secret value $s \in Z_p$ to attributes. Given a vector $\vec{v}=(s,z_2,\cdots,z_n)^{\mathrm{T}}$, where s is the secret value and $r_2,\cdots,r_n \in Z_p$ are selected at random, set M_i as i-th row of M and calculate $\lambda_i=M_i\cdot\vec{v}$, which is one of l sharing values of the secret s.
- **Reconstruction** $\left((\lambda_1,\cdots,\lambda_l),(M,\rho)\right)$**:** The subalgorithm is utilized to recover the secret value s according to $(\lambda_I,\cdots,\lambda_l)$. For any authorized attribute set S, define $I=\{i \mid \rho(i)\in S\}\subset\{1,2,\cdots,l\}$. A serious of coefficients $\{\omega_i \in Z_p \mid i\in I\}$ will satisfy $\sum_{i\in I}\omega_i \cdot M_i = (1,0,\cdots,0)$. Therefore, $s=\sum_{i\in I}\omega_i\lambda_i$ can be reconstructed.

3.4 Bloom Filter

In 1970, Bloom [31] presented the concept of Bloom filter that is a sort of data structure for permitting membership querying and can be applied to make a judgment about whether a value belongs to a collection. Conveniently, BF_A denotes a Bloom filter encoding for a set A. Subsequently, in 2013, Dong et al. [32] proposed the garbled Bloom filter by introducing the XOR operation. Similarly, to add an element $x \in A$ to the filter, x is divided into k shares utilizing the XOR-based secret sharing scheme, which are set on the locations $\{h_i(x)\}_{i\in[k]}$. To inquire x, the relevant values in these positions are executed by the XOR operation. If the value recovered from the above values is equal to x, then $x \in A$, otherwise $x \notin A$.

Furthermore, garbled Bloom filter is employed as a block to build attribute Bloom filter

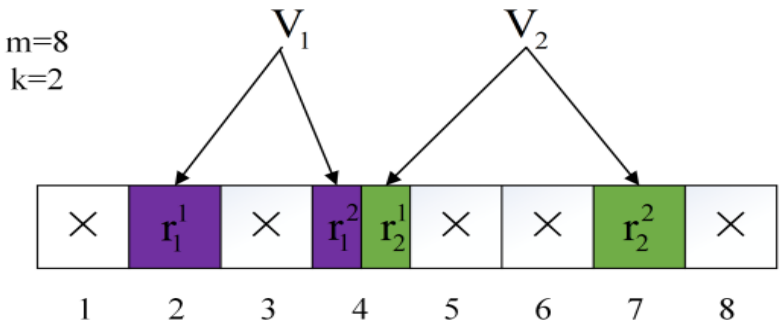


Fig. 1. Example for Inserting Values to ABF

(ABF) parameterized by (m,l,k,H,η). Specifically, l represents the number of inserted attributes, $H=\{h_j\}_{j\in[k]}$ are k independent hash functions, and η denotes the added value's bit

length. To insert an element $\rho(i)$ to the filter, $v_i = \xi_i l + i$ is calculated, where ξ_i is a random value. At the moment, v_i is divided into k η-bit shares $\{v_i^j\}_{j\in[k]}$ utilizing the XOR-based secret sharing scheme, which are presented $pos = h_j(\rho(i))$ located on the corresponding positions. There exists one situation that some values position $pos = h_j(\rho(i))$ is taken up by an existed value. As shown in **Fig. 1**, the existing value will be reused, which sets $v_2^1 = v_1^2$. In addition, the k shares of v_i are calculated. Choose $k-1$ random numbers $v_i^1, v_i^2, \cdots, v_i^{k-1}$ with η bits and calculate $v_i^k = v_i^1 \oplus v_i^2 \oplus \cdots \oplus v_i^{k-1} \oplus v_i$. Specifically, **Algorithm 1** shows the detailed process of *ABFBuild*.

3.5 Formal Definition

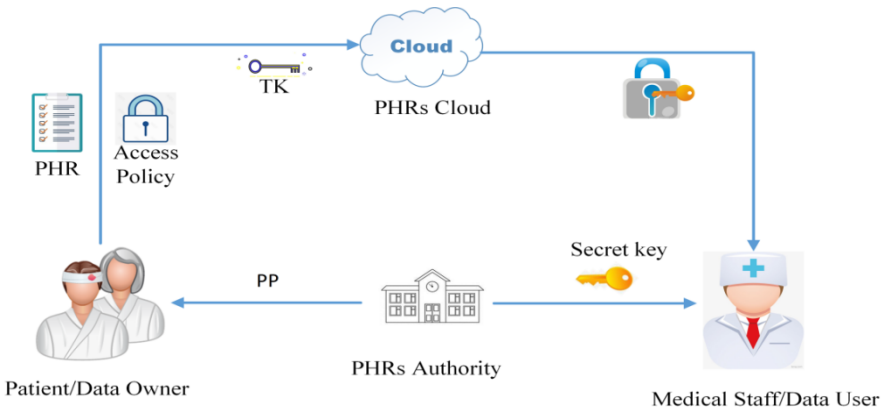


Fig. 2. Architecture of the Privacy-Aware and Data Sharing PHRs

Fig. 2 shows the personal health record system structure, containing following participants: PHRs Authority (PHRA), PHRs Cloud (PHRC), Data Owner (DO), and Data User (DU).

- **Setup** $(\lambda, U) \to (PP, MSK)$**:** This step is executed by PHRA. Put a secure param λ and an attribute universe U into the algorithm. PHRA generates public parameters PP and a master secret key MSK.
- **KeyGen** $(PP, MSK, S) \to SK_S$**:** This step is operated by PHRA. Put PP, MSK, and attribute set S, and PHRA generates relative attribute private key SK_S.
- **Encrypt** $(PP, m, (M, \rho)) \to CT$**:** DO performs the step. Put PP, a message m and a policy (M, ρ), then DO generates a ciphertext CT.
- **Decrypt** $(CT, SK_S) \to m/\perp$**:** DU performs the step. Taking ciphertext CT and corresponding secret key SK_S as input, then DU recovers m while user attribute set satisfy structure located in the *Encrypt* algorithm. Otherwise, the algorithm outputs $\perp$.
- **PolUpdate** $(PP, EnInfo(m), (M, \rho), (M', \rho')) \to TK_m$**:** This step is managed by DO. Taking as input PP, encryption information $EnInfo(m)$ derived from a part of

generated ciphertext, an old policy (M,ρ) and a new policy (M',ρ'), then DO outputs a transforming key TK_m.

- **CTUpdate** $(CT,TK_m)\rightarrow CT'$**:** This step is carried out by PHRC. Put the ciphertext CT and the transforming key TK_m into the algorithm, and PHRC calculates an update ciphertext CT'.

3.6 IND-CPA Security Model

To ensure the security of PPADS, our security model will be built on Sahai and Waters's model [4]. The concrete selective security model is built on an interactive game between a simulator $\mathcal{B}$ and an adversary $\mathcal{A}$. In addition, the ciphertexts before and after updating are indistinguishable, then we merely consider the security before policy update.

1) **Initialization**: $\mathcal{A}$ specifies an access structure $\mathbb{A}^*=(M^*,\rho^*)$, where M^* represents an $l^*\times n^*$ access matrix and ρ^* is a mapping function which maps each row of matrix to an attribute, then transmits it to $\mathcal{B}$.
2) **Setup**: $\mathcal{B}$ executes the algorithm after obtaining $\mathbb{A}^*$, and then returns PP to $\mathcal{A}$.
3) **Phase 1**: $\mathcal{A}$ queries the attribute secret key connected to S.

   Case 1: If attribute set satisfies (M^*,ρ^*), then abort.

   Case 2: $\mathcal{B}$ produces a private key related to S for $\mathcal{A}$.
4) **Challenge**: $\mathcal{A}$ picks two messages m_0,m_1 of equal length and sends them to $\mathcal{B}$. Then $\mathcal{B}$ randomly chooses a bit $\beta\in\{0,1\}$, executes *Encryption* algorithm to produce a challenging ciphertext CT^* and returns it to $\mathcal{A}$.
5) **Phase 2**: *Phase 2* is identical to *Phase 1*.
6) **Guess**: $\mathcal{A}$ returns a guess β' of β. Define the advantage of $\mathcal{A}$ in the security game as:

$$Adv_{\mathrm{A}}=|\Pr[\beta'=\beta]-\frac{1}{2}|.$$

Definition 2. *If a polynomial-time adversary has a negligible advantage in an interactive game, PPADS is IND-CPA secure under the framework of the selective access structure attacks.*

4. Design Details of PPADS

Enlightened by Hao et al.'s scheme [25] and Li et al.'s scheme [33], PPADS is described as shown below.

- **Setup.** PHRA first carries out the *Setup* algorithm by taking as input λ and $U=\{att_1,\cdots,att_{|U|}\}$. The algorithm randomly selects $\alpha,\beta\in Z_p$, $\lambda_1',\cdots,\lambda_l'\in Z_p$ served as attribute masks and group elements $h_{att_1},\cdots,h_{att_{|U|}}\in G$ for all the attributes in U. The

public parameter PP is issued as $PP=\langle e(g,g)^{\alpha},g,g^{\beta},\lambda_1,\cdots,\lambda_{l'},h_{att_1},\cdots,h_{att_{|U|}}\rangle$. The master secret key $MSK=g^{\alpha}$ is held by PHRA.

- **KeyGen.** When DU joins the system, she or he should register and authenticate to PHRA to obtain the related secret key. Along with these attributes S, PHRA generates a corresponding secret key. Select a value $t\in Z_p$ randomly and calculate $D'=g^t, D=g^{\alpha}\cdot(g^{\beta})^t, \forall att_x\in S, D_{att_x}=h_{att_x}^t$. Then, a secret key $SK_S=\langle D,D',\{D_{att_x}\}_{att_x\in S}\rangle$ is distributed to DU through a safe channel.

- **Encrypt.** Put PP, m and (M,ρ) into the algorithm, and DO produces a ciphertext CT and then uploads it to PHRC. In order to hide policy, the generated ciphertext is different from the common ciphertext in basic CP-ABE such as [14]. Specifically, the $Encrypt$ phase of PPADS contains two steps: $CTGen$ and $ABFBuild$. The $CTGen$ step generates common ciphertext and the $ABFBuild$ step assists DU to determine their attribute positions on the access matrix M. It is crucial for the second step that the attribute mapping function ρ can be recovered according to attribute bloom filter.

1) Step 1. $CTGen(PP,m,(M,\rho))\rightarrow CT_0$. The step is regarded as a normal encryption algorithm. According to a LSSS, DO selects a vector $\vec{z}=(s,z_2,\cdots,z_n)$, where s is a secret value and $z_2,\cdots,z_n\in Z_p$ are chosen randomly, calculates $\lambda_i=M_i\cdot\vec{z}$ for each $i\in[l]$, and picks random values $r_1,\cdots,r_l\in Z_p$. Then DO produces the corresponding ciphertext CT_0 as below.

$$CT_0=\langle C=m\cdot e(g,g)^{\alpha s}, C_0=g^s, \{C_{i,1}=g^{\beta\lambda_{i'}}h_{\rho(i)}^{-r_i}, C_{i,2}=g^{r_i}, C_{i,3}=g^{\beta(\lambda_i-\lambda_{i'})}\}_{i\in[l]}\rangle,$$

where $C_{i,3}$ will be used to update ciphertext in the **CTUpdate** algorithm. That is to say, the difference between the generated ciphertext and the common ciphertext in CP-ABE is the component $C_{i,3}$.

2) Step 2. $ABFBuild((M,\rho))\rightarrow T$. The step calls **Algorithm 1** to generate T that hides ρ.

At last, DO uploads ciphertext CT_0 and (M,T), instead of (M,ρ), that is $CT=\langle CT_0,M,T\rangle$, to PHRC.

- **Decrypt.** DU receives the ciphertext $CT=\langle CT_0,M,T\rangle$, she or he can decrypt successfully while her or his attribute sets meet specified access policy contained in the ciphertext. The $Decrypt$ algorithm in PPADS includes three steps: $ABFQuery, MapRecover$ and $DecTest$. The $ABFQuery$ algorithm aims at inquiring for the row value in terms of each user attribute, the $MapRecover$ algorithm is designed to restore the mapping functions corresponding to the row number and then the $DecTest$ step is to test that the decryption can pass or not.

1) Step 1. $ABFQuery(S,T)\rightarrow\Theta$.

The step is executed by calling **Algorithm 2**. Put the attribute set S and the Attribute Bloom filter T into **Algorithm 2**, then outputs a mapping function $\Theta: S \to J$. In terms of each attribute $att_x \in S$, relevant value $r_x = \xi_x l + x$ is inserted. In light of the algorithm, calculate k locations $\{h_j(att_x)\}_{j\in[k]}$, and get the corresponding value $\{r_x^j = T[h_j(x)]\}_{j\in[k]}$. Therefore, there exists $r_x = r_x^1 \oplus r_x^2 \oplus \cdots \oplus r_x^k$. Furthermore, the row number of the inserted attribute att_x can be represented as $rownum_x = r_x \bmod l = \left(r_x^1 \oplus r_x^2 \oplus \cdots \oplus r_x^k\right) modl$. Hence, the row numbers will be generated and then perform the following algorithm.

Remarks. Notice that the returned row numbers are valid while user attributes are lying in the structure, and other row numbers of the rest of attributes are merely random values. Besides, it is worth noting that quite other attributes maybe regain an identical row number. In this case, it is generally impacted by the quantity of attributes belonged to *S* and the size of access structure.

2) Step 2. $MapRecover(\Theta) \to P$.

The step is executed by **Algorithm 3**. Put Θ into the algorithm, and outputs a set P filled with $\overline{\rho_i}$ by choosing all the attributes in S. As shown in **Algorithm 3**, in terms of each row number, the algorithm selects all attributes in J to form an attribute set $\overline{S_i}$ while satisfying the properties of injective functions $i \neq j \Rightarrow \overline{S_i} \neq \overline{S_j}$. Later, the set P is composed by all $\overline{\rho_i}, i \in [l]$. After then, for each $\overline{\rho_i} \in P$ and relative secret key $SK_{\overline{S_i}} \subseteq SK_S$, the following algorithm can be performed.

3) Step 3. $DecTest(CT_0, (M_J, \overline{\rho_i}), SK_{\overline{S_i}}) \to m/\perp$.

This step is a normal decryption algorithm. The algorithm extracts an attribute set I derived from M_J, which $I = \{i \mid \rho(i) \in S\} \subseteq \{1, \cdots, l\}$, M_J represents the specific matrix formed by the row number attached to J and computes the coefficients $\{\omega_i \in Z_p\}_{i\in I}$ such that $\sum_{i\in I} w_i \cdot M_i = (1,0,\cdots,0)$ and $\sum_{i\in I} \omega_i \cdot \lambda_i = s$. Then for each $i \in I$, the algorithm calculates the above formula. If $\overline{S_i}$ cannot be accordant to $(M_J, \overline{\rho_i})$, then output $\perp$.

$$
\begin{aligned}
B &= \frac{e(C_0, D)}{\prod_{i\in I} (e(C_{2,i}, D_{\overline{\rho_i}}) \cdot e(C_{1,i}, D'))^{w_i}} \cdot \frac{1}{e(\prod_{i\in I} C_{3,i}^{\omega_i}, D')} \\
&= \frac{e(g,g)^{\alpha s} \cdot e(g,g)^{\beta t s}}{\prod_{i\in I} e(g,g)^{\beta t w_i \lambda_i}} = e(g,g)^{\alpha s} \\
\frac{C}{B} &= \frac{m \cdot e(g,g)^{\alpha s}}{e(g,g)^{\alpha s}} = m
\end{aligned}
\tag{2}
$$

Algorithm 1: *ABFBuild*
Input: (M,ρ)

1. Select m,k,H,η optimally
2. T =**new** m-element array of η-bit strings
3. **for** $i=1$ to m **do**
4. $T[i]=null$ //initialized
5. **end for**
6. **for** $i=1$ to l **do**
7. Select a random number ξ, such that $\xi_i l+i<2^{\eta}$
8. *EmptyPos=0, FinalShare=* $\xi_i l+i$
9. **for** $j=1$ to k **do**
10. $pos=h_j(\rho(i))$ //get the position index
11. **if** $T[pos]==null$ **then**
12. **if** *EmptyPos==0* **then**
13. *EmptyPos=pos* //store the position
14. **Else**

Select a random number v from $\{0,1\}^{\eta}$

15. $T[pos]=v$
16. $FinalShare=FinalShare\oplus T[pos]$
17. **else**
18. $FinalShare=FinalShare\oplus T[pos]$
19. **end for**
20. $T[EmptyPos]=FinalShare$ //reserve
21. **end for**
22. **for** $i=1$ to m **do**
23. **if** $T[i]==null$ **then**
24. Select a random number v from $\{0,1\}^{\eta}$
25. $T[i]==v$
26. **end for**

Output: $T(m,k,H,\eta)$

Algorithm 2: *ABFQuery*
Input: S,T

1. $J=\varnothing,\Theta=\varnothing$
2. **for** $att_x\in S$ **do**
3. $temp=0$
4. **for** $j=1$ to k **do**
5. $pos=h_j(att_x)$ //get the index
6. $temp=temp\oplus T[pos]$
7. **end for**
8. $rownum_x=temp \bmod l$
9. **if** $rownum_x==0$ **then**
10. $rownum_x==l$
11. **if** $rownum_x\notin J$ **then**
12. add $rownum_x$ into J //randomized
13. **end if**
14. Add $att_x\rightarrow rownum_x$ into Θ
15. **end for**

Output: $\Theta: S\rightarrow J$

Algorithm 3: *MapRecover*
Input: $\Theta: S\rightarrow J$

1. $Num=1$
2. **for** each $rownum\in J$ **do**
3. $attS_{rownum}=\Theta^{-1}(rownum)$
4. $Num=Num*length(attS_{rownum})$
5. **end for**
6. **for** $i=0$ to $(Num-1)$ **do**
7. $step=Num,\overline{\rho_i}=\varnothing,\overline{S_i}=\varnothing$
8. **for** each $rownum\in J$ **do**
9. $len=length(attS_{rownum})$
10. $step=step/len$
11. $attIndex=(i/step) \bmod len$
12. $att=attS_{rownum}[attIndex]$
13. Add att to $\overline{S_i}$
14. Add $rownum\rightarrow att$ into $\overline{\rho_i}: J\rightarrow\overline{S_i}$
15. **end for**
16. Add $\overline{\rho}$ into F
17. **end for** //compose an attribute set

Output: F

- **PolUpdate.** DO performs the policy update and generates a transforming key, which is used to update ciphertext for PHRC. Set the old structure (M,ρ), the updating or new

structure (M', ρ') and encrypted information $EnInfo(m)$, which is defined as $C_{i,3}$. Define $num_{\rho(i),M}$ as the quantity of attribute $\rho(i)$ in M and $num_{\rho(i),M'}$ as the quantity of attribute $\rho(i)$ in M', respectively. Concretely, update algorithm is classified as two steps.

1) Step 1. This step is used to pick the secret value s and $EnInfo(m)$ and attribute mask λ_i'. Policy update would be classified into three cases according to the distribution of attribute location:

Case 1: Let $I_{1,M'}$ be a set of attributes which existed in an original structure if $num_{\rho(i),M'} \geq num_{\rho(i),M}$.

Case 2: Let $I_{2,M'}$ be a set of attributes which existed in an original structure and appear more than once only if $num_{\rho(i),M'} \leq num_{\rho(i),M}$.

Case 3: Let $I_{3,M'}$ be a set of attributes which did not exist in an original structure.

2) Step 2. This step is used to generate a transforming key. Specifically, on the basis of the new access structure (M', ρ'), the patient generates the random vector $\vec{z}'=(s, z_2', \cdots, z_n') \in Z_p$ with the secret value s. Compute $\lambda_j = M_j' \cdot \vec{z}'$, where M_j' is j-th of M'. Attribute parameter λ_i and mask λ_i' are reserved by the original encryption. On account of the above three cases, the transforming key can be regarded as:

Case 1: If $(j,i) \in I_{1,M'}$, select random number $\lambda_j' \in Z_p$, generate the transforming key as:

$$TK_{j,i,m} = (TK_{j,i,m}^{(1)}, TK_{j,i,m}^{(2)}) = (g^{\beta(\lambda_j' - \lambda_i')}, g^{\beta(\lambda_j - \lambda_j')}).$$

Case 2: If $(j,i) \in I_{2,M'}$, select $\lambda_j', a_j \in Z_p$, compute the transforming key as:

$$TK_{j,i,m} = (TK_{j,i,m}^{(1)}, TK_{j,i,m}^{(2)}, TK_{j,i,m}^{(3)}) = (a_j, g^{\beta(\lambda_j' - a_j\lambda_i')}, g^{\beta(\lambda_j - \lambda_j')}).$$

Case 3: If $(j,i) \in I_{3,M'}$, select random number $\lambda_j' \in Z_p$, generate the transforming key as:

$$TK_{j,i,m} = (TK_{j,i,m}^{(1)}, TK_{j,i,m}^{(2)}, TK_{j,i,m}^{(3)}) = (g^{\beta\lambda_j'} h_{\rho(i)}^{-r_j}, g^{r_j}, g^{\beta(\lambda_j - \lambda_j')}).$$

At last, the transforming key TK_m is described as:

$$TK_m = \langle (Case\,1, \{TK_{j,i,m}\}_{(j,i)\in I_{1,M'}}), (Case\,2, \{TK_{j,i,m}\}_{(j,i)\in I_{2,M'}}), (Case\,3, \{TK_{j,i,m}\}_{(j,i)\in I_{3,M'}}) \rangle.$$

- **CTUpdate.** After receiving the transforming key, PHRC will generate a new ciphertext. Then a final ciphertext is composed of the new ciphertext and an updated attribute bloom filter. Therefore, there exist two steps as follows.

1) Step 1. $CTUpdate(CT, TK_m) \rightarrow CT'$. The update ciphertext algorithm inputs the transforming key TK_m and the old ciphertext CT, and then outputs an update ciphertext CT' according to the following three cases.

a. If $I_{1,M'}$ in *Case 1* holds, the updated ciphertext C_j' is described as:

$$C_j' = (C_{j,1}' = C_{i,1} \cdot TK_{j,i,m}^{(1)} = g^{\beta\lambda_j'} \cdot h_{\rho(i)}^{-r_j}, C_{j,2}' = C_{i,2} = g^{r_j}, C_{j,3}' = TK_{j,i,m}^{(2)} = g^{\beta(\lambda_j - \lambda_j')}).$$

In this formula, $r_j = r_i$ is consistent with the original ciphertext.

b. If $(j,i) \in I_{2,M'}$ in *Case 2* holds, the updated ciphertext C_j' is described as:

$$C_j' = (C_{j,1}' = C_{i,1}^{a_j} \cdot TK_{j,i,m}^{(2)} = g^{\beta\lambda_j'} \cdot h_{\rho(i)}^{-r_j}, C_{j,2}' = C_{i,2}^{a_j} = g^{r_j}, C_{j,3}' = TK_{j,i,m}^{(3)} = g^{\beta(\lambda_j - \lambda_j')}),$$

where $r_j = a_j r_i$.

c. If $(j,i) \in I_{3,M'}$ in *Case 3* holds, the C_j' is described as:

$$C_j' = (C_{j,1}' = TK_{j,i,m}^{(1)} = g^{\beta\lambda_j'} \cdot h_{\rho(i)}^{-r_j}, C_{j,2}' = TK_{j,i,m}^{(2)} = g^{r_j}, C_{j,3}' = TK_{j,i,m}^{(3)} = g^{\beta(\lambda_j - \lambda_j')}).$$

2) Step 2. $UpdateABFBulid(M', \rho') \to T'$. The $ABFBulid$ algorithm is run again. The algorithm inputs the new policy, then outputs a new attribute bloom filter. Since DO has a new access policy, we can run the $ABFBulid$ algorithm to get a new T' as a part of the update ciphertext.

In conclusion, the final ciphertext could be regarded as $CT' = (CT_0', M', T')$, where CT_0' is defined as $CT_0' = (C, C_0, \{C_j'\}_{j \in [l']})$.

5. Security Analysis

5.1 Analysis of Ciphertext Indistinguishability

Theorem 1. *Assume q-parallel BDHE assumption holds in groups (G, G_T), then PPADS is IND-CPA secure under the framework of the selective access policy attacks in the standard model.*

Proof: Supposing that an attacker $\mathcal{A}$ could breach our system in a polynomial time with non-negligible advantage of ε in CPA security game, then a challenger $\mathcal{B}$ would have an advantage of $\frac{\varepsilon}{2}$ to resolve the difficult problem.

Pick $s, \beta \in Z_p$ randomly, the decisional q-*parallel* BDHE problem used in PPADS is defined as: $\vec{y} = \left(g, g^s, g^\beta, \cdots, g^{\beta^q}, g^{\beta^{q+2}}, \cdots, g^{\beta^{2q}}\right)$ and Z. Later on, given a coin flip u, if $u = 1$, then $Z = e(g,g)^{\beta^{q+1}s}$; Or else, Z is selected from G_T randomly. Then $\mathcal{B}$ is given a guess value.

Initialization. $\mathcal{A}$ chooses the challenging policy (M^*, ρ^*) and transmits it to $\mathcal{B}$.

Setup. $\mathcal{B}$ simulates PP as below:

- Pick a random number $\alpha' \in Z_p$ and figure up $e(g,g)^\alpha = e(g^\beta, g^{\beta^q}) \cdot e(g,g)^{\alpha'}$, denoted $\alpha = \alpha' + \beta^{q+1}$.
- Select $\beta \in Z_p$ randomly, then compute g^β.
- $\forall x \in U$, put a corresponding number z_x randomly, then calculate h_x:

1) While ρ^* maps an index $i \in \{1,2,\cdots,l^*\}$ to an element x, put

$$h_x = g^{z_x} (g^\beta)^{M_{i,1}^*} (g^{\beta^2})^{M_{i,2}^*} \cdots (g^{\beta^{n^*}})^{M_{i,n}^*}. \quad (3)$$

2) Otherwise, put $h_x = g^{z_x}$.

Phase 1. In the phase, $\mathcal{A}$ can obtain a group of secret keys except that attribute S that satisfy access matrix M^*. $\mathcal{B}$ picks a value $r \in Z_p$ randomly and vector $\vec{w} = (w_1, w_2, \cdots, w_{n^*}) \in Z_p^{n^*}$ with the first element $w_1 = -1$. Put $\vec{w} \cdot M_i^* = 0$, $\forall i, \rho(i)^* \in S$. From the property of a LSSS, there consequentially exists such a vector. Then the simulator $\mathcal{B}$ implicitly defines t as (denoted $N' = \{2, \cdots, n^*\}$):

$$t = r + \omega_1 \beta^q + \omega_2 \beta^{q-1} + \omega_{n^*} \beta^{q-n^*+1} \tag{4}$$

Generate D, D', D_{att_x} as follows:

$$D = g^{\alpha'} g^{\beta r} \prod_{i \in N'} (g^{\beta^{q+2-i}})^{w_i}, D' = g^{r + w_1 \beta^q + w_2 \beta^{q-1} + w_{n^*} \beta^{q-n^*+1}} \tag{5}$$

While there not exists an index $i \in \{1, 2, \cdots, l^*\}$ mapped to x, put $D_{att_x} = (D')^{z_x}$. Otherwise, put (denoted $N = \{1, \cdots, n^*\}$)

$$D_{att_x} = (D')^{z_x} \cdot \prod_{j \in N} \left(g^{(\beta_j) \cdot r} \cdot \prod_{k \in N, k \neq j} (g^{\beta^{q+1+j-k}})^{\omega_k} \right)^{M_{i,j}^*} \tag{6}$$

Challenge. $\mathcal{A}$ returns two plaintexts m_0 and m_1. Then $\mathcal{B}$ opts for a random value $\beta = \{0,1\}$ and calculates $C = m_\beta Z \cdot e(g^s, g^{\alpha'})$ and $C_0 = g^s$. Next, $\mathcal{B}$ picks $y_2', \cdots, y_n'$ randomly and forms the vector $\vec{v}: \vec{v} = (s, s\beta + y_2', s\beta^2 + y_3', \cdots, s\beta^{n-1} + y_{n^*}') \in Z_p^{n^*}$. Furthermore, $\mathcal{B}$ selects some randomized numbers $r_1', \cdots, r_l' \in Z_p$. The challenge ciphertext is produced:

$$C_{i,1} = (g^s)^{-z_{\rho(i)}} \cdot g^{z_{\rho(i)} r_i'} \cdot \prod_{j \in N} (g^{\beta^j})^{M_{i,j}^* \cdot r_i'} \cdot g^{\beta \lambda_i''} \tag{7}$$

$$C_{i,2} = g^s \cdot g^{-r_i'}, \; C_{i,3} = \prod_{j \in N'} (g^{\beta})^{y_j M_{i,j}^*} \cdot g^{-\lambda_i''} \tag{8}$$

Set $r_i = s - r_i', \lambda_i' = sM_{i,1}^* + s\beta M_{i,2}^* + \cdots + s\beta^{n-1} M_{i,n}^* + \lambda_i'', i \in \{1, 2, \cdots, l^*\}$.

Phase 2. *Phase 2* is identical to *Phase 1*.

Guess. $\mathcal{A}$ outputs a guess β'. While $\beta = \beta'$, $\mathcal{B}$ returns 1 to suggest that $Z = e(g,g)^{a^{q+1} s}$. Or else, $\mathcal{B}$ gets back 0 that signifies Z is a random element. From the above interactive game, it is visible that the simulation of key queries and ciphertext performance was identical to the real system.

1) If $u=0$, $Z \in G_T$, $\mathcal{A}$ is winner possessing the probability $\Pr[\beta' = \beta \mid u = 0] = \frac{1}{2}$.

Then $\mathcal{B}$ outputs $u'=0$ while $\beta' \neq \beta$ and $\Pr[u' = u \mid u = 0] = \frac{1}{2}$.

2) If $u=1$, $\mathcal{B}$ successfully simulated the challenge ciphertext. Suppose $\mathcal{A}$ break the system with the advantage of ε, $\mathcal{B}$outputs $u'=1$ while $\beta' = \beta$ and $\Pr[u' = u \mid u = 1] = \frac{1}{2} + \varepsilon$.

In a word, the advantage of $\mathcal{B}$ can be described as:

$$\mathrm{Adv}_{\mathrm{B}} = \left|\Pr[u' = u \mid u = 0] \cdot \Pr[u = 0]\right| + \left|\Pr[u' = u \mid u = 1] \cdot \Pr[u = 1] - \frac{1}{2}\right|$$

$$= \left|\frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2} \cdot (\frac{1}{2} + \varepsilon) - \frac{1}{2}\right| = \frac{1}{2}\varepsilon$$

Hence, we proved that PPADS is IND-CPA secure under the (decisional) q-parallel BDHE assumption.

5.2 Security Comparison

Security level. Table 1 presents intuitional function comparisons. However, PPADS and Hao et al.'s scheme [25] can provide a stronger privacy protection. In particular, the disclosure of access structure can lead to the theft of privacy information, since access structure existed in the form of plaintext. However, the full policy hiding mentioned in PPADS cannot obtain any sensitive information through access policy, which can guarantee higher security. In addition, **Theorem 1** indicates that PPADS is selectively secure. Dynamic update can be realized in PPADS and [29, 33]. Specifically, the difference is that their schemes adopted different update classification ways. However, all of them only implemented single functionality. Based on the above comparisons, PPADS possesses more powerful functionalities.

Table 1. Functionality Comparisons

Schemes	Policy update	Partial policy hiding	Full policy hiding	Security	Expressiveness of policy
[11]	×	√	×	selective	AND-gates
[19]	×	√	×	full	LSSS
[25]	×	×	√	selective	LSSS
[29]	√	×	×	selective	LSSS
[33]	√	×	×	selective	LSSS
Ours	√	×	√	selective	LSSS

Privacy protection and policy update. In PPADS, attribute can be hidden by concealing the attribute mapping function ρ. Data users are allowed to query the corresponding mapping function for their owned attributes. However, users who can pass decryption test are authorized medical staffs. Later on, the *ABFQuery* oracle returns a random value to $\mathcal{A}$. Consequently, *ABFQuery* algorithm cannot reveal any user privacy. Furthermore, transforming key queries

still can not improve the advantage of the adversary. Assume that (M_i^*, ρ_i^*) and (M_j^*, ρ_j^*) are old and new access structure, respectively. Then considering the transforming key queries $TK(m_0, M_i^*, \rho_i^*)$ and $TK(m_1, M_j^*, \rho_j^*)$, the transforming key oracle returns the same transforming key while the adversary $\mathcal{A}$ can not distinguish the encryption between m_0 and m_1. Thus, PPADS is secure to protect the policy privacy.

6. Performance Analysis

In the section, we contrast our system with other corresponding works [11, 19, 25, 29, 33]. **Table 2** displays the specific symbol notations. We further give the storage cost as shown in **Table 3**. Note that an element length in each group $\mathbb{G}, \mathbb{G}_T$ is set to 512 bits. From **Table 3**, due to the characteristic of bloom filter, the size of ciphertext in PPADS is smaller contrasted with Lai et al.'s scheme [19]. Furthermore, since the classification of updating ciphertext is different from Ying et al.'s scheme [29], the size of transforming key and update ciphertext in PPADS is shorter. **Table 4** shows the time complexity comparisons of each algorithm among these schemes. Since the time complexity of updating ciphertext is almost the same, we only compare the computation time of updating transforming key.

Table 2. Notations

Notations	Terms
$\|U\|$	The number of attributes in the system attribute universe
$\|S\|$	The number of attributes in the user attribute set
$\|L_l\|$	The bit-length of the value l
$\|\mathbb{G}\|$	The bit-length of element in $\|\mathbb{G}\|$
$\|\mathbb{G}_T\|$	The bit-length of element in $\|\mathbb{G}_T\|$
$\|Z_p\|$	The bit-length of element in $\|Z_p\|$
$\|h\|$	The bit-length of hash function
t	The number of the hash functions
k	The number of the update attributes
m	The size of the Attribute Bloom Filter
l	The number of attributes in the access policy

Table 3. Comparisons of Storage Overhead

Schemes	[11]	[19]	[25]	[29]	[33]	Ours
PP	$(3+\|U\|)\|\mathbb{G}\| + \|\mathbb{G}_T\|$	$(4+\|U\|)\|\mathbb{G}\| + \|\mathbb{G}_T\|$	$(2+\|U\|)\|\mathbb{G}\| + \|\mathbb{G}_T\|$	$(2+\|U\|)\|\mathbb{G}\| + \|\mathbb{G}_T\|\|h\|$	$(2+\|U\|)\|\mathbb{G}\| + \|\mathbb{G}_T\|$	$(2+\|U\|)\|\mathbb{G}\| + \|\mathbb{G}_T\| + l\|Z_p\|$
MSK	$(2+\|U\|)\|Z_p\|$	$\|U\|\|\mathbb{G}\| + \|Z_p\|$	$\|\mathbb{G}\|$	$\|\mathbb{G}\|$	$\|\mathbb{G}\|$	$\|\mathbb{G}\|$

SK		$(1+2\|S\|)\|G\|$	$(1+2\|S\|)\|G\|$	$(1+2\|S\|)\|G\|$	$(2+\|S\|)\|G\|$	$(2+\|S\|)\|G\|$	$(2+\|S\|)\|G\|$
CT		$(1+2)\|G\|$ $+\|G_T\|$	$(1+4l)\|G\|$ $+2\|G_T\|$ $+\|(M,\rho)\|$	$(1+l)\|G\|$ $+\|G_T\|$ $+$ $\|M\|+m\|L_l\|$ $+k\|h\|$	$(1+2l)\|G\|$ $+\|G_T\|$ $+\|(M,\rho)\|$	$(1+2l)\|G\|$ $+\|G_T\|$ $+\|(M,\rho)\|$ $+l\|Z_P\|$	$(1+3l)\|G\|$ $+\|G_T\|$ $+\|M\|+m\|L_l\|$ $+k\|h\|$
TK_m	Type1	—	—	—	$2t\|G\|$	$\|Z_P\|$	$2\|G\|$
	Type2					$3\|Z_P\|$	$\|Z_P\|+2\|G\|$
	Type3					$\|Z_P\|+2\|G\|$	$(t+2)\|G\|$
CT'	Type1	—	—	—	$2t\|G\|$	$2\|G\|+\|Z_P\|$	$(t+2)\|G\|$)
	Type2						
	Type3						

Table 4. Comparisons of Computation Cost

Schemes		[11]	[19]	[25]	[29]	[33]	Ours
Encryption		$(l^2+l+2)E$ $+M$	$(6l+1)M$ $+(6l+4)E$	$(l+1)M+tH$ $+(2l+1)E$	$(l+1)M+tH$ $+(3l+2)E$	$(2l+1)M+tH$ $+(3l+2)E$	$(l+1)M+tH$ $+(4L+2)E$
Decryption		$\le 2M$ $+(2l+1)P$	$\le 2M+lE$ $+(2l+1)P$	$\le 2M+2lE$ $+(2l+1)P+tH$	$\le 2M+lE$ $+(2l+1)P+tH$	$\le 2M+tH$ $+(l+1)E$ $+(2l+2)P$	$\le 2M+2tE$ $+(2l+2)P+tH$
Up-date	Type1	—	—	—	$M+3E+tH$	$M+E$	$2M+2E$
	Type2	—	—	—	$M+3E+tH$	$3M+2E$	$3M+2E$
	Type3	—	—	—	$M+3E+tH$	$2M+3E+tH$	$2M+4E+tH$

We can draw an intuitive efficiency comparison graph based on **Table 4**, where M, E, P, and H denote a multiplication operation, an exponent operation, pairing operation, and hash operation, respectively. To evaluate the feasibility of PPADS for PHR system, some necessary experiments are conducted to measure time operation. These experiments are carried out by a laptop with an Intel configuration, CPU, TM i5-7500@3.40GHz, and 4GB RAM. We detect the efficiency of PPADS on the basis of Pairing-based Cryptography (PBC) library [34]. Since the encryption and decryption time are concerned factors to assess the efficiency of the system, we make a comparison of the computational time between PPADS and Lai et al.'s scheme [19]. **Fig. 3** and **Fig. 4** illustrate the comparison of encryption time and decryption time for policy hiding, respectively. Though the encryption and decryption time of both our system and Lai et al.'s scheme [19] increase along with the number of attributes, PPADS is more efficient since fewer pair operations are required.

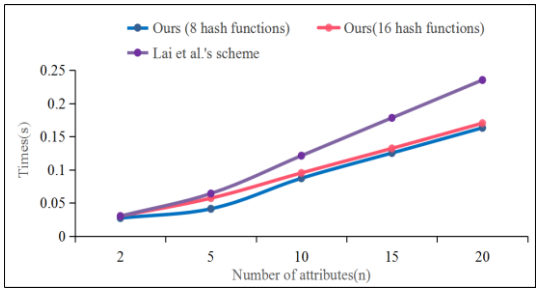


Fig. 3. Encryption Time for Policy Hiding

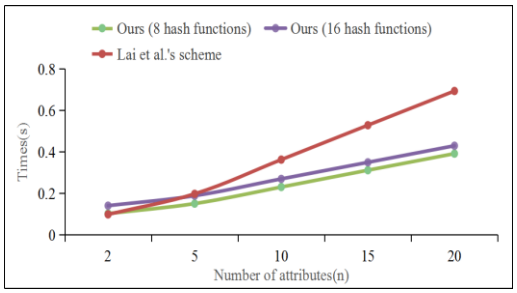


Fig. 4. Decryption Time for Policy Hiding

In addition, focusing on policy update, **Fig. 5** elaborates the concrete computational time between PPADS and Ying et al.'s scheme [29]. In [29], since the time complexity of each type for transforming key is the same, we present only a curve comparison analysis. **Table 4** presents the time complexity of Type 1 and Type 2 for transforming key do not increase along with the number of attributes in PPADS, while only Type 3 in PPADS and all types in Ying et al.'s scheme [29] require them. Generally, PPADS can achieve higher efficiency than Ying et al.'s scheme [29].

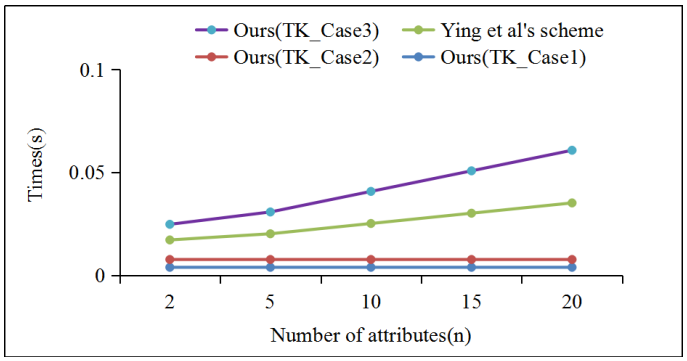


Fig. 5. Computation Time of TK_m

In summary, PPADS has the advantage of supporting expressive access structure, full policy hiding, and flexible policy update over the existing schemes. Therefore, PPADS is more applicable for data confidentiality and user privacy in PHRs. As depicted that the experimental results are coincident with the theory analysis, thus PPADS is feasible.

7. Conclusions

In this paper, we have feasibly addressed data confidentiality and user privacy in PHRs by recommending PPADS, which helps patients to attain medical assistance conveniently. The core building block of PPADS is a basic CP-ABE scheme that realizes full policy hiding and dynamic update simultaneously. In PPADS, the whole attribute can be hidden by an attribute bloom filter and the ciphertext can be updated by PHRC with a transforming key. Moreover, the system provides a specific security proof under decisional q -BDHE assumption. Theoretical analysis and extensive experiment result demonstrate that PPADS has the advantage over other schemes. However, PPADS can only support small universe. Thus, our future work will pay more attention to how to set up a system with large universe effectively.

Furthermore, the proposed system from bilinear pairing cannot resist quantum computation, and thus post-quantum secure PPADS over lattice is on the list of things worth studying.

References

[1] M. L. Braunstein, "Health care in the age of interoperability part 5: the personal health record," *IEEE Pulse*, vol. 10, no. 3, pp. 19-23, May 2019. Article (CrossRef Link)

[2] J. Li, N. Zhang, J. Ni, J. Chen, and R. Du, "Secure and lightweight authentication with key agreement for smart wearable systems," *IEEE Internet Things Jouranl*, vol. 7, no. 8, pp. 7334-7344, 2020. Article (CrossRef Link)

[3] S. Namani and B. Gonen, "Smart agriculture based on IoT and cloud computing," in *Proc. of the 3rd International Conference on Information and Computer Technologies*, pp. 553-556, Mar. 2020. Article (CrossRef Link)

[4] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Proc. of Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 457-473, 2005. Article (CrossRef Link)

[5] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. of IEEE Symposium on Security and Privacy*, pp. 321-334, May 2007. Article (CrossRef Link)

[6] R. Ostrovsky, A. Sahai, and B. Waters, "Attribute-based encryption with non-monotonic access structures," in *Proc. of the 14th ACM Conference on Computer and Communications Security*, pp. 195-203, Oct. 2007. Article (CrossRef Link)

[7] B. Xu, L. D. Xu, H. Cai, C. Xie, J. Hu, and F. Bu, "Ubiquitous data accessing method in IoT-based information system for emergency medical services," *IEEE Transactions Industrial Informatics*, vol. 10, no. 2, pp. 1578-1586, Feb. 2014. Article (CrossRef Link)

[8] S. Namani and B. Gonen, "Smart agriculture based on IoT and cloud computing," in *Proc. of the 3rd International Conference on Information and Computer Technologies*, pp. 553-556, Mar. 2020. Article (CrossRef Link)

[9] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. of the 13th ACM Conference on Computer and Communications Security*, pp. 89-98, Oct. 2006. Article (CrossRef Link)

[10] L. Cheung and C. Newport, "Provably secure ciphertext policy abe," in *Proc. of the 14th ACM Conference on Computer and Communications Security*, pp. 456-465, 2007. Article (CrossRef Link)

[11] N. Takashi, Y. Kazuki, and O. Kazuo, "Attribute-based encryption with partially hidden encryptor-specifified access structures," in *Proc. of the 6th International Conference on Applied Cryptography and Network Security*, pp. 111-129, 2008. Article (CrossRef Link)

[12] B. Waters, "Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization," in *Proc. of the 14th International Workshop on Public Key Cryptography*, pp. 53-70, 2011. Article (CrossRef Link)

[13] Z. Liu, J. Xu, Y. Liu, and B. Wang, "Updatable ciphertext-policy attribute-based encryption scheme with traceability and revocability," *IEEE Access*, vol. 7, pp. 66832-66844, May 2019. Article (CrossRef Link)

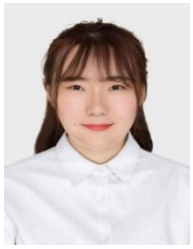
[14] X. Yan, X. He, J. Yu, and Y. Tang, "White-box traceable ciphertext-policy attribute-based encryption in multi-domain environment," *IEEE Access*, vol. 7, pp. 128298-128312, Sep. 2019. Article (CrossRef Link)

[15] Y. Miao, J. Ma, X. Liu, and H. Li, "Practical attribute-based multi-keyword search scheme in mobile crowdsourcing," *IEEE Internet Things Jouranl*, vol. 5, no. 4, pp. 3008-3018, Dec. 2018. Article (CrossRef Link)

[16] F. Ö. Çatak and A. F. Mustacoglu, "CPP-ELM: Cryptographically Privacy-Preserving Extreme Learning Machine for Cloud Systems," *International Journal of Computational Intelligence Systems*, vol. 11, no. 1, pp. 33-44, Jan. 2018. Article (CrossRef Link)

[17] T. T. Thwin and S. Vasupongayya, "Performance Analysis of Blockchain-based Access Control Model for Personal Health Record System with Architectural Modelling and Simulation," *International Journal of Networked and Distributed Computing*, vol. 8, no. 3, pp. 139-151, May 2020. Article (CrossRef Link)
[18] J. Lai, R. H. Deng, and Y. Li, "Fully secure ciphertext-policy hiding cp-abe," in *Proc. of the 7th International Conference on Information Security Practice and Experience*, pp. 24-39, 2011. Article (CrossRef Link)
[19] J. Lai, Y. Li, R. H. Deng, and Y. Li, "Expressive cp-abe with partially hidden access structures," in *Proc. of the 7th ACM Symposium on Information, Computer and Communications Security*, pp. 18-19, May 2012. Article (CrossRef Link)
[20] H. Cui, R. H. Deng, G. Wu, and J. Lai, "An efficient and expressive ciphertext-policy attribute-based encryption scheme with partially hidden access structures," in *Proc. of the 10th International Conference on Provable Security*, pp. 19-38, 2016. Article (CrossRef Link)
[21] Z. Wang, J. Han, M. Wang, Y. Shi, and H. Dong, "Public key encryption with wildcards keyword search," in *Proc. of the 8th International Conference on Instrumentation & Measurement, Computer, Communication and Control*, pp. 538-541, Mar. 2018. Article (CrossRef Link)
[22] H. Yang, Y. Su, J. Qin, and H. Wang, "Privacy-preserving outsourced inner product computation on encrypted database," *IEEE Transactions on Dependable and Secure Computing*, 2020. Article (CrossRef Link)
[23] Y. Michalevsky and M. Joye, "Decentralized policy-hiding attribute-based encryption with receiver privacy," in *Proc. of European Symposium on Research in Computer Security*, pp. 548-567, Sep. 2018. Article (CrossRef Link)
[24] F. Khan, H. Li, L. Zhang, and J. Shen, "An expressive hidden access policy cp-abe," in *Proc. of IEEE 2nd International Conference Data Science in Cyberspace*, pp. 178-186, June 2017. Article (CrossRef Link)
[25] J. Hao, C. Huang, J. Ni, H. Rong, M. Xian, and X. S. Shen, "Fine-grained data access control with attribute-hiding policy for cloud-based iot," *Computer Networks*, vol. 153, pp. 1-10, Apr. 2019. Article (CrossRef Link)
[26] A. Sahai, H. Seyalioglu, and B. Waters, "Dynamic credentials and ciphertext delegation for attribute-based encryption," in *Proc. of Annual Cryptology Conference on Advances in Cryptology*, vol. 7414, pp. 199-217, 2012. Article (CrossRef Link)
[27] K. Yang, X. Jia, and K. Ren, "Secure and verifiable policy update outsourcing for big data access control in the cloud," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 12, pp. 3461-3470, Dec. 2015. Article (CrossRef Link)
[28] Y. Zhang, H. Li, J. Zhang, and J. Cui, "Adaptively secure ciphertext-policy attribute-based encryption with dynamic policy updating," *Science China Information Sciences*, vol. 59, no. 4, pp. 1-16, Apr. 2016. Article (CrossRef Link)
[29] Z. Ying, W. Jang, S. Cao, X. Liu, and J. Cui, "A lightweight cloud sharing phr system with access policy updating," *IEEE Access*, vol. 6, pp. 64 611-64 621, Oct. 2018. Article (CrossRef Link)
[30] W. Yuan, "Dynamic policy update for ciphertext-policy attribute-based encryption," *IACR Cryptol. ePrint Arch*, 2016. Article (CrossRef Link)
[31] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422-426, July 1970. Article (CrossRef Link)
[32] C. Dong, L. Chen, and Z. Wen, "When private set intersection meets big data: an efficient and scalable protocol," in *Proc. of 2013 ACM SIGSAC Conference on Computer & Communications Security*, pp. 789-800, Nov. 2013. Article (CrossRef Link)
[33] J. Li, S. Wang, Y. Li, H. Wang, H. Wang, H. Wang, J. Chen, and Z. You, "An efficient attribute-based encryption scheme with policy update and file update in cloud computing," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 12, pp. 6500-6509, Dec. 2019. Article (CrossRef Link)
[34] A. De Caro and V. Iovino, "jPBC: Java pairing based cryptography," in *Proc. of 2011 IEEE Symposium on Computers and Communications*, pp. 850-855, Aug. 2011. Article (CrossRef Link)

Zhenhua Liu received the B.S. degree from Henan Normal University in 2000, and the M.S. and Ph.D. degrees from Xidian University, China, in 2003 and 2009, respectively. He is currently a Professor of Xidian University, China. His current research interests include cryptography and information security.



Jiaqi Ji received the B.S. degree from Taiyuan Normal University in 2017. She is currently going in for the M.S. degree in mathematics with Xidian University, China. Her research focuses on network and information security.

Fangfang Yin received the B.S. degree from Henan Normal University in 2018. She is currently going in for the M.S. degree in mathematics with Xidian University, China. Her research interests concentrate on cryptography and cloud security.

Baocang Wang received the B.S., the M.S. and Ph.D. degrees from Xidian University, China, in 2001, 2004, and 2006, respectively. He is currently a Professor with the State Key Laboratory of Integrated Services Networks of Xidian University, China. His research focuses on post-quantum cryptography, number theoretic algorithms, and cloud security.