A Coordinated Ciphertext Policy Attribute-based PHR Access Control with User Accountability

Guofeng Lin[1], Lirong You[2], Bing Hu[1], Hanshu Hong[1] and Zhixin Sun[1]

[1] Key Lab of Broadband Wireless Communication and Sensor Network Technology, Nanjing University of Posts and Telecommunications
Nanjing, China
[e-mail: 2015070249@njupt.edu.cn]
[2]Jiangsu Zhongtian Software Technology Limited Company
[e-mail: iamyoulirong@163.com]
[1] Key Lab of Broadband Wireless Communication and Sensor Network Technology, Nanjing University of Posts and Telecommunications
[e-mail: hubing_2009@163.com]
[1] Key Lab of Broadband Wireless Communication and Sensor Network Technology, Nanjing University of Posts and Telecommunications
[email: 2014070244@njupt.edu.cn]
[1] Key Lab of Broadband Wireless Communication and Sensor Network Technology, Nanjing University of Posts and Telecommunications
[e-mail: sunzx@njupt.edu.cn]
*Corresponding author: Zhixin Sun

Received March 21, 2017; revised July 14, 2017; revised September 22, 2017; accepted November 15, 2017; published April 30, 2018

Abstract

The personal health record (PHR) system is a promising application that provides precise information and customized services for health care. To flexibly protect sensitive data, attribute-based encryption has been widely applied for PHR access control. However, escrow, exposure and abuse of private keys still hinder its practical application in the PHR system. In this paper, we propose a coordinated ciphertext policy attribute-based access control with user accountability (CCP-ABAC-UA) for the PHR system. Its coordinated mechanism not only effectively prevents the escrow and exposure of private keys but also accurately detects whether key abuse is taking place and identifies the traitor. We claim that CCP-ABAC-UA is a user-side lightweight scheme. Especially for PHR receivers, no bilinear pairing computation is needed to access health records, so the practical mobile PHR system can be realized. By introducing a novel provably secure construction, we prove that it is secure against selectively chosen plaintext attacks. The analysis indicates that CCP-ABAC-UA achieves better performance in terms of security and user-side computational efficiency for a PHR system.

Keywords: Personal health record, access control, accountability, provably secure

http://doi.org/10.3837/tiis.2018.04.024 **ISSN : 1976-7277**

1. Introduction

The personal health record (PHR) system [1,2] is a promising cloud-based application that has powerful capability to analyze users' health conditions, disease histories, medications, and so on. Modern PHR systems provide increasingly exciting functions based on the large numbers of health records [33]. Currently, PHR systems are widely used in disease rehabilitation, disease prevention and medical treatment [22]. With huge data capacity, PHR systems naturally contain sensitive private user information. It is convenient for visitors to access this mass digital resource without any supervision [16,23]. However, untrusted service providers and unauthorized users should not be able to gain access to this sensitive data. Consequently, it is essential to apply security-oriented techniques to limit access.

Attribute-based encryption (ABE) [4,5] is an interesting technique because it describes users' identities by collections of authorized attributes rather than looking up certificates. When a collection of attributes satisfies the customized access policy, the corresponding user can correctly implement decryption. Due to the fuzzy mechanism of identity verification, ABE inherently has a unique one-to-many property. Consequently, it is reasonable to establish an access control method based on ABE. Because ciphertext policy attribute based encryption (CP-ABE) [6] associates a ciphertext with an access policy, and binds a data receiver's private key with attributes, it allows data owners to define the access policy by themselves. Therefore, it constitutes an effective method of cryptology to build secure PHR access control.

However, current PHR systems based on CP-ABE may cause a series of open problems. One of the major threats is key escrow [21]. Key escrow means that a key generation center plays a dominant role in key generation. Consequently, it must be absolutely trusted and be able to decrypt all ciphertexts via private keys without any supervision. In addition, the phenomenal increase in usage and deployment of mobile applications stimulates the merging of mobile PHR systems [31,32], but two currently popular mobile operating systems (Android OS and iOS) use built-in storage that requires few permissions to manipulate or access data [9]. It is not difficult to covertly access sensitive data available through mobile PHR applications. Thus, the vulnerability of terminal storage protection may lead to easy exposure of private keys. Furthermore, because PHR owners with the same set of attributes share the same private key, current schemes cannot immediately identify a traitor if an anonymous PHR receivers exists who deliberately shares his/her key with other users. This threat is called key abuse, which is still an open problem [26]. In addition, it is costly in terms of memory and power for a user-side mobile PHR system to execute large computations especially bilinear pairing computations. Therefore, an approach for building a secure and efficient attribute-based PHR access control is needed.

In this paper, we propose a coordinated ciphertext policy attribute-based access control with user accountability (CCP-ABAC-UA), which aims to provide a practical, secure, and efficient access control method for PHR systems. The main contributions are summarized as follows:

1). A coordinated scheme is introduced for synchronous generation and distributed storage of private keys. Therefore, both key escrow and key exposure are perfectly solved.

2). We present an effective user accountability mechanism that requires a simple computation to identify the traitor so that the key abuse problem can be solved.

3). A user-side lightweight scheme is introduced. Especially for PHR receivers, no bilinear pairing computation is needed to extract health records from a ciphertext, which enhances the efficiency of attribute-based PHR access control.

4). We propose a reduction of our proposed scheme to the original CP-ABE. With the help of this reduction, we prove that CCP-ABAC-UA is secure under chosen-plaintext attacks in the random oracle model.

The rest of the paper is organized as follows: In Section 2, we review the existing works. In Section 3, we present the algorithm and security definition of provably secure CCP-ABAC-UA for PHR access control. In Section 4, we give the construction details and present the correctness of the scheme. In Section 5, we provide proof that our scheme is secure under the selective CPA game. In Section 6, we present a performance analysis of CCP-ABAC-UA compared with representative ABEs and current PHR systems. We conclude the paper and discuss our future work in Section 7.

2. Related Works

During the past decades, many studies focusing on health care management have considered the practical application of PHR. Win [3] indicated that it is necessary for electronic medical management to provide privacy protection because large numbers of health records are personal and sensitive. In her work, a comprehensive analysis demonstrated that existing techniques of access control are inadequate to establish concrete protection. In 2012, a personally controlled electronic health record (PCEHR) [2] system was released in Australia. The PCEHR defined some primitive functions of a PHR system for various users, including management of health summaries, pathology reports, medical history, and organ donor profies. However, the maturity and usage rate of its access control urgently needed to be raised. Laranjo *et al.* [1] described a potential transformation from a physician-centered care system to a patient-centered care system due to patients' emerging requirements for direct control of their own health records. The access control in future PHR systems should be more flexible than in previous ones.

Sahai and Waters [4] first utilized two sets of attributes to represent the fuzzy identity of a user and the access policy of a ciphertext. When these two sets are closer than a given threshold, the user can extract the plaintext. Their work laid a good foundation for attribute-based encryption (ABE), which is potentially suitable for establishing secure access control. In 2006, Goyal *et al.* [14] proposed a formal definition of ABE. By introducing an access tree, they built a fine-grained access policy for ABE. Their research work indicated that their construction, based on FIBE, is a key policy ABE (KP-ABE), which means that each private key is associated with an access policy and each ciphertext is associated with a set of attributes. Another type of ABE is called ciphertext policy ABE (CP-ABE). For CP-ABE, each ciphertext is associated with an access policy, and each private key is associated with a set of attributes. Bethencourt *et al.* [6] proposed a specific realization of CP-ABE. They indicated that CP-ABE is more suitable than KP-ABE for establishing flexible access control because it allows data owners to define access policies by themselves. Chase and Chow [17] proposed a multi-authority ABE that significantly addressed the key escrow problem that inherently existed in ABEs, but resulted in tremendous overhead when updating keys. Zhang *et al.* [18] provided an improvement of [17] that deployed only one sub-authority to generate private keys by its interactions with the key authority, while keeping public keys and private keys short. Hur [10] introduced a two-party computational protocol between the key authority and the data-storing center to issue private keys. His construction was a novel solution to the key escrow problem but failed to provide a concrete security proof of the global system. Green *et al.* [8] proposed a concept of outsourcing decryption that delegates a high-performance server to execute part of the decryption while leaving no information with it. Finally, only a small amount of computations are needed for the data receiver to extract a plaintext. Lai *et al.* [7] proposed a concrete implementation of verifiable outsourcing ABE. However, their verification nearly doubled the decryption overhead in [8]. Chandar *et al.* [19] proposed a hierarchical ABE (HABE) based on lazy re-encryption, in which only the message to be updated is re-encrypted. The analysis demonstrated that HABE markedly reduces computation overhead for attribute revocation and user revocation.

Considering the open environment of a cloud-based PHR system, Chen *et al.* [23] proposed a dynamic access control for a PHR system based on Lagrange. Their scheme can be regarded as a kind of adoption of an ABE primitive into PHR access control. Li *et al.* [16] divided the access requirements of a PHR system into two domains to improve the efficiency of key management. The public domain consists of doctors, nurses, medical researchers, etc., in which PHR sharing is built based on multi-authority CP-ABE. The personal domain consists of those who have personal connections with the PHR owner, in which the PHR owner is qualified to be a trusted authority. Therefore, PHR sharing in the personal domain is based on KP-ABE. Xhafa *et al.* [20] established an ABE-based fine-grained access control that roughly offers the same functions as in [23] but supports fuzzy keyword searching. Wungpornpaiboon *et al.* [15] proposed a two-layer CP-ABE, in which PHR owners are responsible for defining access policies in the inner layer and professionals such as doctors are responsible for redefining the access policy in the outer layer to share the PHR with other colleagues. Thus, their access control is more flexible than previous PHR systems. Qian *et al.* [24] proposed a PHR sharing scheme based on multi-authority ABE to realize flexible access control, on-demand revocation and dynamic policy updating. Qin *et al.* [27] indicated that although KP-ABE does not allow data owners to define access policies, it is possible to update access policies with the help of a third-party delegation. Based on this, they proposed a redefinable KP-ABE for PHR sharing, which provides flexible access control but also markedly reduces ciphertext size. Xhafa *et al.* [25] built a PHR access control based on multi-authority ABE that supports not only hidden access policy but also user accountability. However, their scheme was realized at the cost of computing tremendous numbers of bilinear pairings. Considering that the attribute sets of PHR receivers are highly susceptible to privacy leaks, Zhang *et al.* [28] proposed an anonymous hierarchical ABE for PHR access control that hides attribute sets and reduces the sizes of public parameters and private keys. Hong *et al.* [26] proposed an access control for a PHR system based on a novel ABE without bilinear pairing to achieved lightweight computational overhead as well as user accountability. Noting that their scheme is secure only based on Computational Diffie-Hellman (CDH) Assumption, so we consider it efficient at the cost of security. Miao *et al.* [30] exploited the integration of searchable encryption and CP-ABE, and proposed an attribute-based multi-keyword search algorithm over encrypted PHR. Rao [29] proposed a provably secure ciphertext-policy attribute-based signcryption which concentrates on ciphertext authenticity in PHR system. This approach guarantees not only fine-grained access control but also confidentiality and non-repudiation. However, if trust level of trusted attribute authority and PHR receiver degenerates, his work becomes unreliable.

As mentioned above, current ABEs guarantee data security when the key authority is semi-trusted but fail to give a concrete security proof of the global construction. Meanwhile, current schemes hardly consider the user accountability when key exposure or key abuse occurs. Moreover, security enhancement comes at the cost of user-side computational overhead. Therefore, existing schemes are inadequate for building a concrete and secure access control for PHR systems.

3. Modeling Syntax

3.1 Original CP-ABE for a PHR System

The original realization of CP-ABE proposed by Waters [12] deployed four entities, which are the data owner, key authority, data-storing center, and data receiver. The data owner possesses data to be shared by CP-ABE. The key authority is responsible for generating and issuing keys. The data-storing center is responsible for storing encrypted data, which can be regarded as cloud storage with help of cloud computing. The data receiver wants to obtain data stored in the data-storing center. Except that data-storing center is honest but curious, other entities must be fully trusted. In particular, the original CP-ABE consists of the following 4 algorithms:

- Setup: When a security parameter is input, the key authority runs the setup algorithm to output a group of public parameters and a master secret. Note that public parameters are available to all entities, while the master secret is held secretly by the key authority.
- Key Generation: When a data receiver uploads his/her set of attributes, the key authority runs the key generation algorithm. According to the attribute set and the master secret, it generates a private key for this data receiver.
- Encryption: When a data owner wants to share his/her data with a customized access policy, he/she executes the encryption algorithm to encrypt data by the public parameters and the access policy. Subsequently, a ciphertext is output and uploaded to the data-storing center.
- Decryption: When a data receiver wants to obtain information from a ciphertext, he/she executes the decryption algorithm with this ciphertext and his/her private key. If his/her set of attributes matches the access policy, he/she obtains all information from the ciphertext. If not, he/she cannot extract any useful information.

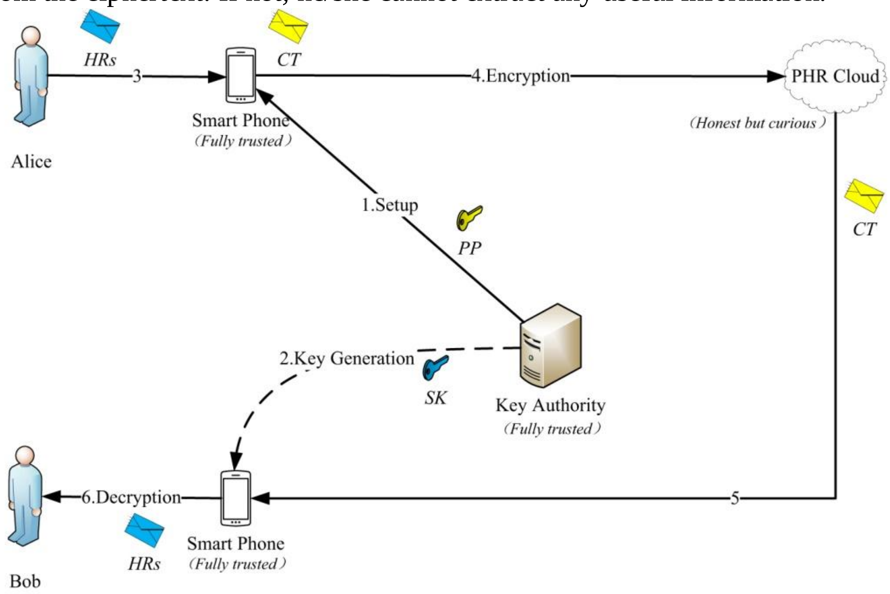


Fig. 1. The model of PHR Sharing based on the Original CP-ABE

Now, consider a PHR sharing scenario based on the original CP-ABE: Alice wants to share her health records in a PHR Cloud for convincing diagnosis. Subsequently, she opens a PHR application on her smart phone. It runs the encryption algorithm to encrypt the health records with her access policy. Then, it uploads the ciphertext to the PHR cloud. Bob is a physician specialist who has been registered in the system. He possesses a private key associated with his attribute set. To diagnose patients anywhere and anytime, he installs a mobile PHR application on his smart phone. If Bob's attribute set satisfies Alice's access policy, he will be able to extract Alice's health records correctly. We give the model of such a scenario in **Fig. 1**, which shows the threat model of the system and the sequence of how the original CP-ABE works in it. In this model, PP represents public parameters generated by the key authority, SK represents a private key associated with Bob's attribute set, HRs represents Alice's health records, and CT represents a ciphertext. There are some open problems that potentially hinder its practical application:

1). The key authority must be fully trusted because it completely takes over the generation of private keys. If the key authority is compromised, it is highly possible that data owners will unknowingly share their sensitive information with illegal data receivers.

2). Bob must be fully trusted. If not, there is a possibility that he will profit financially by selling his private key. For PHR system, such abuse is inevitable because of the tremendous value of health records. When abuse occurs, it is hard to identify the traitor. In addition, the weak privacy protection of a smart device could lead to the unintentional exposure of his private key.

3). Bob's smart phone must be a high-performance device because the original CP-ABE requires many user-side computations, especially bilinear pairing computations during decryption, which consume a large amount of memory, power and computation resources. If Bob has a performance-constrained smart phone, the execution time required to extract health records may be unacceptable.

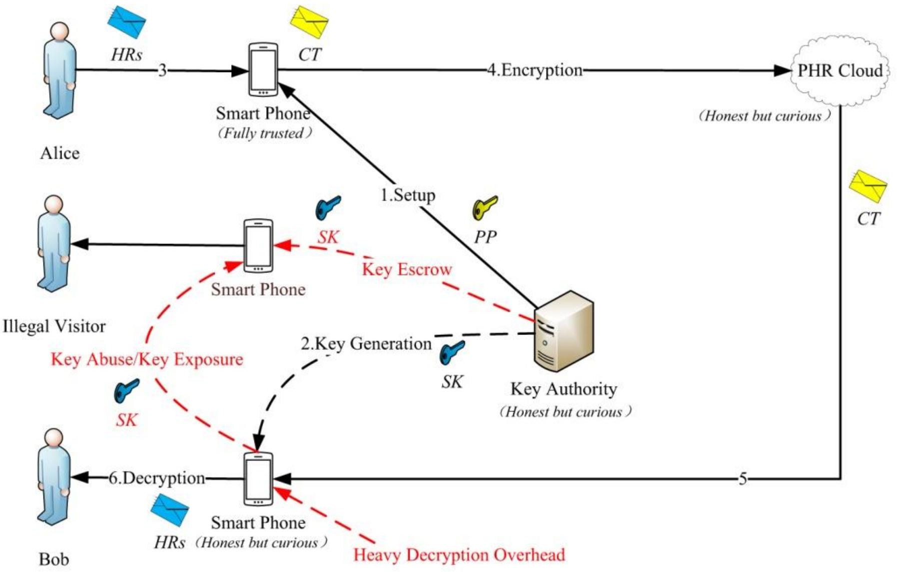


Fig. 2. The Threat Model of PHR Sharing based on the Original CP-ABE

Thus, if the threat model changes, that is, key authority and data receiver (for example, Bob's smart phone) are honest but curious, which are the same as the PHR cloud. The security of the system is compromised. In **Fig. 2**, we show open problems as aforementioned, which are represented by red dotted lines. In conclusion, it is inappropriate to build a PHR access control based on the original CP-ABE with respect to both security and efficiency.

3.2 Provably Secure CCP-ABAC-UA for a PHR System

To address the problems in PHR sharing based on the original CP-ABE, we propose a provably secure CCP-ABAC-UA for PHR systems. There are five entities involved in our scheme: the PHR owner, the key authority, the PHR cloud, the decryption server and the PHR receiver. We assume the key authority, the PHR cloud and the PHR receiver are honest but curious. The key authority tends to leak private keys that it generates for illegal visitors. Additionally, the PHR cloud tends to share data that it stores with unknown visitors. Meanwhile, if the PHR receiver uses a performance-constrained smart phone to access the PHR cloud, it is possible to abuse his/her private key. Moreover, we deploy only an honest but curious decryption server to execute a part of the decryption, which helps to realize secure and efficient access control. We assume that the PHR cloud, the key authority and the decryption server do not collude with each other, otherwise our scheme will be unreliable and meaningless.

The provably secure CCP-ABAC-UA for PHR systems consists of the following seven sub-algorithms:

- $Setup(1^k,\Omega)\rightarrow\{PP,MK\}$: The setup algorithm is run in a coordinated fashion by the key authority and the PHR cloud. It takes as input a security parameter k and authorized attribute space Ω. Then, the key authority generates its public parameter PP_K and master key MK_K. Meanwhile, the PHR cloud generates its public parameter PP_C and master key MK_K. Finally, the setup algorithm asks the key authority and the PHR cloud to output their public parameters to form the universal public parameters $PP=\{PP_K,PP_C\}$. Note that the universal master keys $MK=\{MK_K,MK_C\}$ are respectively kept by the key authority and the PHR cloud secretly.
- $KeyGen(PP,MK,S)\rightarrow\{SK_{init,1},SK_{init,2}\}$: The key generation algorithm is run in a coordinated fashion by the key authority and the PHR cloud. It takes as input the universal public parameters PP, the universal master keys MK, and an attribute set S. It asks the key authority and the PHR cloud to cooperatively generate the first initial key $SK_{init,1}$ and the second initial key $SK_{init,2}$. Note that $SK_{init,1}$ and $SK_{init,2}$ are respectively held by the key authority and the PHR cloud. The key generation algorithm can be executed based on the 2PC protocol [10], the details of which are provided in Section 4.
- $Encrypt(PP,\mathbb{A},HRs)\rightarrow CT$: The encryption algorithm is run by the PHR owner. It takes as input the universal public parameters PP, a customized access policy $\mathbb{A}$, and a portion of health records HRs. Then, it returns a ciphertext CT.
- $Decrypt(PP,CT,SK_{init,1},SK_{init,2})\rightarrow HRs$: The decryption algorithm is adopted to provide reduction to the original CP-ABE. It takes as input the universal public parameters PP, a ciphertext CT, the first initial key $SK_{init,1}$, and the second initial key $SK_{init,2}$. If $S\in\mathbb{A}$, it will output HRs. If not, it will return $\perp$ to indicate a

running error. We adopt this algorithm for the sake of theoretical correctness; this encryption algorithm does not exist in any practical application.

- $AcKeyGen(PP, SK_{init,1}, SK_{init,2}, id) \rightarrow \{SK_{ulti,1}, SK_{ulti,2}, CK_{ulti}\}$: The accountable key generation algorithm is coordinately run by the key authority, the PHR cloud, and a PHR receiver. It takes as input the universal public parameters PP, the first initial key $SK_{init,1}$, the second initial key $SK_{init,2}$, and the identity id of a PHR receiver. Then, it generates the first ultimate server key $SK_{ulti,1}$, the second ultimate server key $SK_{ulti,2}$ and the ultimate client key CK_{ulti} , which are respectively returned to the key authority, the PHR cloud and the PHR receiver. During the execution, id is embedded into all these keys to effectively prevent this PHR receiver from abusing his/her keys.
- $Transform(PP, CT, SK_{ulti,1}, SK_{ulti,2}) \rightarrow CT'$: The transformation algorithm is run by the decryption server when it receives a decryption request from a PHR receiver. It takes as input the universal public parameters PP, a ciphertext CT, the first ultimate server key $SK_{ulti,1}$ and the second ultimate server key $SK_{ulti,2}$. Then, it executes a part of the decryption and outputs a semi-decrypted ciphertext CT' to a PHR receiver. Because the decryption server needs $SK_{ulti,1}$ and $SK_{ulti,2}$ to run the transformation algorithm, it cannot be any server of the PHR receiver's choice. Otherwise, the illegal visitor would have absolute power to utilize the decryption server to coordinate all private keys for decryption, and as a result, our construction would make no sense.
- $UsDecrypt(id^*, CT', CK_{ulti}) \rightarrow HRs$: The user-side decryption algorithm is run by the PHR application on the PHR receiver's smart phone. It takes as input the identity id^* of the PHR receiver, a semi-decrypted ciphertext CT', and the ultimate client key CK_{ulti}. Then, it verifies the identity of the PHR receiver. If there is no key abuse, it outputs the corresponding portion of the health records HRs.

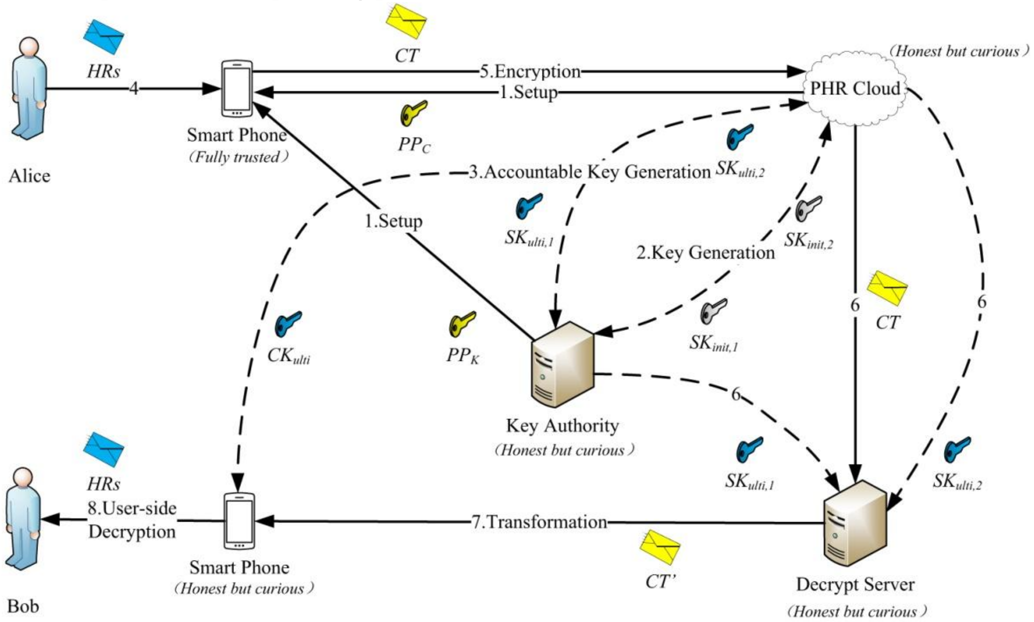


Fig. 3. The Model of CCP-ABAC-UA for the PHR System

Considering a scene similar to that described in Section 3.1, we illustrate the model of CCP-ABAC-UA for a PHR system in **Fig. 3**, in which solid lines represent the process through which Bob gains access to Alice's health records and dotted lines demonstrate how private keys are generated and issued among those entities. In this system, Alice wants to share her health records for convincing diagnosis. She opens a PHR application on her smart phone to contact the PHR cloud. Then, it runs the encryption algorithm that uses the public parameter PP_K of the key authority and the public parameter PP_C of the PHR cloud to encrypt health records with her access policy. Subsequently, it uploads the ciphertext to the PHR cloud. Bob is a doctor with great specialization. To diagnose patients anywhere and anytime, he installs a mobile PHR application on his smart phone and registers in the PHR system. With help of the key generation algorithm and accountable key generation algorithm, he receives an ultimate client key. Simultaneously, the key authority and the PHR cloud respectively receive the first ultimate server key and the second ultimate server key. Note that all keys are associated with Bob's identity and attribute set. When Bob sends a decryption query for Alice's health records, the key authority sends the first ultimate server key of Bob to the decryption server. Meanwhile, the PHR cloud sends the corresponding ciphertext and the second ultimate server key of Bob to the decryption server. If Bob's attribute set satisfies Alice's access policy, he will receive a semi-decrypted ciphertext from the decryption server. Subsequently, the mobile PHR application runs the user-side decryption algorithm to verify his ultimate client key. If there is no abuse, he can extract Alice's health records.

3.3 Security Definition

The conventional security game against chosen-ciphertext attacks (CCA) does not allow any transformation of the ciphertext. Thus, we introduce its relaxation from Green *et al.* [8] and Canetti *et al.* [13]. That is the definition of re-playable chosen-ciphertext attacks (RCCA) security, which allows alternation of the ciphertext but no underlying message changed. We describe an RCCA game of CCP-ABAC-UA as follows:

Setup: The challenger launches the setup algorithm to generate the universal public parameters and the universal master keys. The universal public parameters are sent to the adversary, while the universal master keys are kept in secret.

Query phase 1: The challenger first generates an empty table T and an empty set D. Then, it issues following queries:

1). *Generation query:* After the adversary selects a set of attributes S, the challenger launches the key generation algorithm to give the first initial key $SK_{init,1}$ and the second initial key $SK_{init,2}$ to the adversary. Then, the set D is altered as $D = D \cup \{S\}$.

2). *Corruption query:* After the adversary selects a set of attributes S, the challenger immediately scans the table T to look up the following tuple:

$$\{S, SK_{init,1}, SK_{init,2}, SK_{ulti,1}, SK_{ulti,2}, CK_{ulti}\}.$$

If there is a matched result, it will return the corresponding $\{SK_{ulti,1}, SK_{ulti,2}\}$. Otherwise, it implements the key generation and the accountable key generation algorithms to store the new tuple in T and returns $\{SK_{ulti,1}, SK_{ulti,2}\}$.

3). *Decryption query:* After the adversary selects an attribute set S and a ciphertext CT, the challenger executes the key generation algorithm for $\{SK_{init,1}, SK_{init,2}\}$. Then, it runs the decryption algorithm and returns a portion of health records HRs to the adversary.

4). *User-side Decryption query:* After the adversary selects an attribute set S and a pair of ciphertexts $\{CT, CT'\}$, the challenger immediately scans T to search for the corresponding tuple. If there is a matched result, it will run the user-side decryption algorithm and give a portion of health records to the adversary. Otherwise, it aborts the computation and returns $\perp$ to indicate a running error.

Challenge: The adversary submits two portions of health records, HRs_0 and HRs_1, of equal length and a challenge access policy $\mathbb{A}^*$ that all attribute sets $\{S\}_{S \in D}$ cannot satisfy. Then, the challenger randomly tosses a coin $\beta \in \{0,1\}$, and runs the encryption algorithm to output challenge ciphertext CT^*.

Query phase 2: The adversary adaptively repeats queries in phase 1, while some restrictions must be followed:

1). The attribute set that satisfies the challenge access policy cannot be used as input for any query. Otherwise, it outputs $\perp$ to abort the challenge game.

2). All decryption queries will be answered normally, as in phase 1, other than the one whose response is either HRs_0 or HRs_1. In that case, it outputs $\perp$ to abort the challenge game.

Guess: The adversary selects $\beta' \in \{0,1\}$ as its guess for β. If $\beta' = \beta$, it wins the game.

The advantage of the adversary to win this game is defined as follows:

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

We note that CCP-ABAC-UA is RCCA-secure if all polynomial-time adversaries have at most a negligible advantage to win this game. If the adversary cannot access all the decryption queries, we can say that CCP-ABAC-UA is secure against the selective chosen plaintext attack (selectively CPA-secure) game.

4. Main Construction

4.1 Setup

Define two cyclic groups $\mathbb{G}_1$ and $\mathbb{G}_T$ with prime order p. Let g be a generator of the group $\mathbb{G}_1$. The setup algorithm takes as input a security parameter k and an authorized attribute space $\Omega = \{\theta_1, \theta_2, \ldots, \theta_m\}$. It first sets a bilinear map $e(\cdot,\cdot): \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$. Then, it chooses a set of random numbers $\{h_1, h_2, \cdots, h_m\} \in Z_p^*$, in which each element is associated with an attribute in Ω. We adopt the improved setup computation based on [12], which is coordinately run by the key authority and the PHR cloud. It first outputs the initial public parameter as follows:

$$PP' = \{g, h_1, h_2, \cdots, h_m\}.$$

Then, the key authority randomly chooses an exponent $a \in Z_p^*$ as its master key MK_K, and computes its public parameter $PP_K = g^a$. The PHR cloud simultaneously chooses a random exponent $\alpha \in Z_p^*$; then, it generates the master key $MK_C = g^\alpha$ and outputs its public parameter $PP_C = e(g,g)^{\alpha}$. It finally generates the universal public parameters and the universal master keys as follows:

$$PP = \{PP', PP_K, PP_C\},\ MK = \{MK_K, MK_C\}.$$

4.2 Key Generation

The key generation algorithm takes as input the universal public parameters PP, the universal master keys MK, and an attribute set S. Then, it chooses a random exponent $\tau \in Z_p^*$ to generate the first initial key:

$$SK_{init,1} = \{L' = g^{\tau}, \forall x \in S : K_x' = h_x^{\tau}\}.$$

Let F be a two-party computation function. This function serves to coordinate the key authority and PHR cloud server to generate the second initial key using their master keys:

$$F(\alpha, a, \tau) \to SK_{init,2} = \{K' = g^{\alpha + a\tau}\}.$$

Following [10,21], this can be easily implemented in five steps:

1). The key authority and the PHR cloud cooperatively launch a secure two-party computation in which the key authority takes as input a and τ, and the PHR cloud takes as input α.

2). The secure two-party computation function returns $f = (\alpha/\tau + a)/\tau$ to the PHR cloud.

3). The PHR cloud selects a random element $\varepsilon \in Z_p^*$ and computes $A = g^{f/\varepsilon}$. Then, it sends A to the key authority.

4). The key authority computes $B = A^{\tau^2}$. Then, it sends B to the PHR cloud.

5). The PHR cloud obtains the second initial key by computing $K' = B^{\varepsilon}$.

4.3 Encryption

The encryption algorithm takes as input the universal public parameters PP, a customized access policy $\mathbb{A}$ and a portion of the heath records HRs. We build the access policy via the linear secret sharing scheme (LSSS) [11]. Let Λ be a matrix with m rows and n columns. Define the map $\rho : \{1, 2, \cdots, m\} \to P$ from each row to an attribute for labeling. The LSSS for access policy $\mathbb{A}$ is represented by (Λ, ρ). It first selects a secret $s \in Z_p^*$ to be shared and a group of random exponents $\sigma_2, \sigma_3, \cdots, \sigma_m \in Z_p^*$. Then it returns $\{\lambda_{\rho(i)} = m_i \cdot (s, \sigma_2, \cdots, \sigma_m)^{\mathrm{T}}\}$ for sharing s, where m_i is the ith row of Λ. Finally, it outputs the initial ciphertext as follows:

$$CT = \{(\Lambda, \rho), C = HRs \cdot e(g,g)^{\alpha s}, C' = g^s, \forall \theta_t \in (\Lambda, \rho) : C_t' = g^{a\lambda_t} h_{\rho(t)}^{-s\sigma_t}, D_t' = g^{s\sigma_t}\}.$$

4.4 Decryption

The decryption algorithm is adopted to provide reduction to the original CP-ABE. It takes as input the universal public parameters PP, a ciphertext CT, the first initial key $SK_{init,1}$ and the second initial key $SK_{init,2}$. If user's attribute set S matches the access policy, it runs the following computations to extract the health records:

$$\frac{e(C',K')}{\prod_{\rho(t)\in S}(e(C'_t,L')\cdot e(D'_t,K'_{\rho(t)}))^{\omega_t}}=e(g,g)^{\alpha s},\ \frac{C}{e(g,g)^{\alpha s}}=HRs\ .$$

If the attribute set does not match, it aborts the computation and returns $\perp$ to indicate a running error. We define the construction state above as the basic CCP-ABAC-UA for theoretical correctness. The decryption algorithm does not exist in practical use, but its deployment facilitates an effective reduction to the original CP-ABE. Therefore, we are able to give the theorem as follows:

Theorem 1. Assume that the original CP-ABE scheme [12] is secure under the selective CPA game. Then the basic CCP-ABAC-UA is selectively CPA-secure.

Proof of Theorem 1. We give the proof in Section 5.

4.5 Accountable Key Generation

The accountable key generation algorithm takes as input the universal public parameters PP, the first initial key $SK_{init,1}$, the second initial key $SK_{init,1}$ and the identity id of a PHR receiver.. It coordinates the key authority， the PHR cloud and a PHR receiver to generate the first ultimate server key $SK_{ulti,1}$, the second ultimate server key $SK_{ulti,2}$ and the ultimate client key CK_{ulti}. Let the identity of the PHR receiver be $id\in\{0,1\}^*$, which is a public and unique constant defined by the PHR receiver when he/she registers in the PHR system. Define a hash function $H_{ac}:\{0,1\}^*\to Z_p^*$ with collision resistance, which is kept in secret by the PHR receiver. First, the PHR receiver utilizes the hash function H_{ac} to generate a hash value $H_{ac}(id)$. Then, it coordinates the key authority to compute the first ultimate server key by two-party computation function F:

$$F(H_{ac}(id),SK_{init,1})\to SK_{ulti,1}=\{L=(g^{\tau})^{H_{ac}(id)},\forall x\in S:K_x=(h_x^{\tau})^{H_{ac}(id)}\}\ .$$

Its generation is implemented as follows:

1). The key authority and the PHR receiver cooperatively launch a secure two-party computation, in which the key authority takes as input the first initial key $SK_{init,1}$ and the PHR receiver takes as input a hash value $H_{ac}(id)$ of his/her identity.

2). The key authority selects a random element $\phi\in Z_p^*$, and computes $L''=(L')^{\phi}$ and $\forall x\in S:K''_x=(K'_x)^{\phi}$. Then, it sends L'' and $\forall x\in S:K''_x$ to the PHR receiver.

3). The PHR receiver computes $L'''=(L'')^{H_{ac}(id)}$ and $\forall x\in S:K'''_x=(K''_x)^{H_{ac}(id)}$. Then, it sends L''' and $\forall x\in S:K'''_x$ to the key authority.

4). The key authority obtains the first ultimate server key by computing $L=(L''')^{1/\phi}$ and $\forall x\in S:K_x=(K'''_x)^{1/\phi}$.

Meanwhile, the PHR receiver coordinates the PHR cloud to compute the second ultimate server key using the two-party computation function F:

$$F(H_{ac}(id),SK_{init,2})\to SK_{ulti,2}=\{K=(g^{\alpha+a\tau})^{H_{ac}(id)}\}\ .$$

Its generation is implemented as follows:

1). The PHR cloud and the PHR receiver coordinately launch a secure two-party computation, in which the PHR cloud takes as input the second initial key $SK_{init,2}$ and the PHR receiver takes as input a hash value $H_{ac}(id)$ of his/her identity.

2). The PHR cloud selects a random element $\varphi \in Z_p^*$ and computes $K'' = (K')^{\varphi}$. Then, it sends K'' to the PHR receiver.

3). The PHR receiver computes $K''' = (K'')^{H_{ac}(id)}$. Then, it sends K''' to the PHR cloud.

4). The PHR cloud obtains the second ultimate server key by computing $K = (K''')^{1/\varphi}$.

Finally, the PHR receiver uses the hash value $H_{ac}(id)$ as his/her the ultimate client key:

$$CK_{ulti} = H_{ac}(id).$$

After the execution, CK_{ulti} is sent to the client while $SK_{ulti,1}$ and $SK_{ulti,2}$ are respectively held by the key authority and the PHR cloud. Note that these ultimate server keys are generated without leaking any knowledge about the secret to each other. We synchronously generate and store the ultimate keys among a PHR receiver, the PHR cloud server and the key authority, but none of them have any knowledge about what information is kept by the others. Therefore, both the key escrow and key exposure problems can be solved by our coordinated mechanism. Moreover, we embed the identity into the ultimate keys so that accountability can be realized.

4.6 Transformation

The transformation algorithm is run by the decryption server when it receives a decryption request from a PHR receiver. It takes as input the universal public parameters PP, a ciphertext CT, and the ultimate server keys $\{SK_{ulti,1}, SK_{ulti,2}\}$. When a user launches a decryption query for CT, the decryption server runs the following computation to transform CT:

$$\prod\nolimits_{\rho(t)\in S} (e(C_t, L)e(D_t, K_{\rho(t)}))^{\omega_t} = e(g,g)^{as\tau\cdot H_{ac}(id)}.$$

We define the parameter $T = e(g,g)^{\alpha s\cdot H_{ac}(id)}$. The semi-decrypted ciphertext is as follows:

$$CT' = \{C = HRs\cdot e(g,g)^{\alpha s}, T = e(g,g)^{\alpha s\cdot H_{ac}(id)}\}.$$

If the attribute set is illegal, it aborts the computation and returns symbol $\perp$ to indicate an error. Note that these executions run by the decryption server effectively reduce user-side decryption overhead while leaking no useful information to it.

4.7 User-side Decryption

The user-side decryption algorithm is run by a PHR application on the PHR receiver's smart phone. It takes as input the identity id^* of the PHR receivers, the semi-decrypted ciphertext CT' and the ultimate client key CK_{ulti}. First, it verifies that the CK_{ulti} belongs to this user by computing $H_{ac}(id^*)$. If $H_{ac}(id^*) \neq CK_{ulti} = H_{ac}(id)$, it will abort the computation and match $H_{ac}(id^*)$ with all PHR receivers to identify the traitor. Consequently, this user accountability can prevent key abuse in attribute-based PHR access control. If the verification is successful, all that needs to be done is an easy computation to access the corresponding health records:

$$\frac{C}{T^{\frac{1}{CK_{ulti}}}} = HRs \cdot$$

We note that it is run on local PHR apps with no pairing computation, by which a secure mobile PHR system can be realized with light computational overhead.

Theorem 2. Assume that the basic CCP-ABAC-UA is secure under the selective CPA game. Then the CCP-ABAC-UA is a secure scheme under the selective CPA game.

Proof of Theorem 2. We give the proof in Section 5.

4.8 Correctness Proof

For any authorized PHR receiver who launches a decryption query for CT, if his/her attribute set does not satisfy the access policy, the transformation algorithm will abort the computation and output $\perp$ to indicate a running error. If his/her attribute set satisfies the access policy in CT, the transformation algorithm computes as follows to extract T:

$$\begin{aligned} T &= \prod\nolimits_{\rho(t)\in S} (e(C_t, L)e(D_t, K_{\rho(t)}))^{\omega_t} \\ &= \prod\nolimits_{\rho(t)\in S} (e(g^{a\lambda_t} h_{\rho(t)}^{-s\sigma_t}, (g^{\tau})^{H_{ac}(id)})e(g^{s\sigma_t}, (h_{\rho(t)}^{\tau})^{H_{ac}(id)}))^{\omega_t} \\ &= \prod\nolimits_{\rho(t)\in S} (e(g^{a\lambda_t}, g^{\tau H_{ac}(id)})e(g^{\tau H_{ac}(id)}, h_{\rho(t)}^{-s\sigma_t})e(g^{s\sigma_t}, (h_{\rho(t)}^{\tau})^{H_{ac}(id)}))^{\omega_t} \\ &= \prod\nolimits_{\rho(t)\in S} (e(g,g)^{a\tau\lambda_t H_{ac}(id)} e(g,h)^{-s\tau\sigma_t H_{ac}(id)} e(g,h)^{s\tau\sigma_t H_{ac}(id)})^{\omega_t} \\ &= \prod\nolimits_{\rho(t)\in S} e(g,g)^{a\tau\lambda_t\omega_t H_{ac}(id)} \\ &= e(g,g)^{as\tau \cdot H_{ac}(id)} \end{aligned} .$$

Then, the user-side decryption algorithm computes $H_{ac}(id^*)$ via the identity id^* of the client. If $H_{ac}(id^*) \neq CK_{ulti}$, it will abort the decryption and output $\perp$ to indicate a running error. Then, it will scan the receiver list and find the traitor. If $H_{ac}(id^*) = CK_{ulti}$, the authorized client will receive the semi-decrypted ciphertext $CT' = \{C, T\}$. Finally, he/she extracts the corresponding health records as follows:

$$\begin{aligned} \frac{C}{T^{\frac{1}{CK_{ulti}}}} &= \frac{HRs \cdot e(g,g)^{\alpha s}}{(e(g,g)^{\alpha s H_{ac}(id)})^{\frac{1}{H_{ac}(id)}}} \\ &= \frac{HRs \cdot e(g,g)^{\alpha s}}{e(g,g)^{\alpha s}} \\ &= HRs \end{aligned} .$$

5. Security Proof

5.1 Proof of Theorem 1

Theorem 1. Assume that the original CP-ABE scheme [12] is secure under the selective CPA game. Then the basic CCP-ABAC-UA is selectively CPA-secure.

Proof. Suppose there exists an adversary A that can attack the basic CCP-ABAC-UA. Then we can build an algorithm O that can break the original CP-ABE scheme.

Let N be the challenger of the original CP-ABE scheme in the selective CPA game. The execution is carried out in steps:

- **Initialize:** A submits the challenge access structure $(\Lambda,\rho)^*$. Then, O gives $(\Lambda,\rho)^*$ to N as its challenge access structure and receives public parameters:
$$PP=\{g,g^a,e(g,g)^\alpha,h_1,h_2,\cdots,h_m\}.$$
- **Setup:** O sends public parameters to A as follows:
$$PP=\{PP'=\{g,h_1,h_2,\cdots,h_m\},PP_K=g^a,PP_C=e(g,g)^\alpha\}.$$
- **Query phase 1:** A starts to issue a private key query with a set of attributes S. At exactly the same time, O executes the key generation of N with attribute set S as well. When O receives the private key $PK=\{K,L,\forall x\in S:K_x\}$ associated with attribute set S, it returns $SK_{init,1}=\{L,\forall x\in S:K_x\}$ and $SK_{init,2}=K$ to A.
- **Challenge:** A submits two portions of health records, HRs_0 and HRs_1, of equal length. Then, O sends these two portions of health records to N. The challenger picks up a random bit $\beta\in\{0,1\}$ and encrypts HRs_β as follows:
$$CT^*=\{(\Lambda,\rho)^*,C=HRs_\beta\cdot e(g,g)^{\alpha s},C'=g^s,\forall\theta_t\in(\Lambda,\rho)^*:C_t=g^{a\lambda_t}h_{\rho(t)}^{-r_i},D_t=g^{r_i}\}.$$
Then, O transfers CT^* to A as its challenge ciphertext.
- **Query phase 2:** A continues to launch the private key query adaptively, as in phase 1.
- **Guess:** A outputs its speculation β' for β. Then, O gives β' to N as its guess.

For the encryption algorithm of the basic CCP-ABAC-UA, it selects a group of random exponents $\sigma_1,\sigma_2,\cdots,\sigma_m\in Z_p^*$, which are associated with attributes in access policy to compute $\{\forall\theta_t\in(\Lambda,\rho):C_t',D_t'\}$, so these components are uniformly distributed in $\mathbb{G}_1$. Note that in the original scheme, a group of random elements $r_1,r_2,\cdots,r_m\in Z_p^*$ are selected to compute $\{\forall\theta_t\in(\Lambda,\rho):C_t,D_t\}$ for encryption. Consequently, these two tuples are identically distributed in $\mathbb{G}_1$. A cannot distinguish whether the ciphertext is generated by basic CCP-ABAC-UA or the original CP-ABE. Thus, if A can attack the basic CCP-ABAC-UA in the selective CPA game in polynomial time with non-negligible advantage, we can successfully build an algorithm O that can break the original CP-ABE scheme in the selective CPA game in polynomial time with non-negligible advantage.

5.2 Proof of Theorem 2

Theorem 2. Assume that the basic CCP-ABAC-UA is secure under the selective CPA game. Then the CCP-ABAC-UA is a secure scheme under the selective CPA game.

Proof. Suppose there exists a polynomial-time adversary A that can attack our proposed CCP-ABAC-UA in the selective CPA game with non-negligible advantage. Then, we can build a polynomial-time algorithm O that can break the basic CCP-ABAC-UA in the selective CPA game with non-negligible advantage.

Let N be a challenger of the basic CCP-ABAC-UA in the selective CPA game. Then, the execution is carried out in the following steps:

- **Initialize:** A submits the challenge access policy $(\Lambda,\rho)^*$ to O. Then, O gives $(\Lambda,\rho)^*$ to the challenger N as its challenge access policy and receives the universal public parameters:

$$PP = \{PP' = \{g, h_1, h_2, \cdots, h_m\}, PP_K = g^a, PP_C = e(g,g)^{\alpha}\}.$$

- **Setup:** O transfers the universal public parameters PP to A.
- **Query phase 1:** The adversary adaptively and repeatedly makes the following queries:

1). Generation query: O initializes a table:

$$T_{GenQ} = < S, c, SK_{init,1}, SK_{init,2} >.$$

When A issues a generation query with a set of attributes S, O checks the corresponding entry in T_{GenQ} and returns $(SK_{init,1}, SK_{init,2})$ to A. If there is no matched result, it selects a bit $c = \{0,1\}$ that satisfies:

$$\begin{cases} \Pr\{c = 0\} = \varphi \\ \Pr\{c = 0\} = 1 - \varphi \end{cases}.$$

If $c = 0$, it calls the key generation algorithm of N and stores the entry in T_{GenQ} as follows:

$$< S, c = 0, SK_{init,1} = \{g^{\tau}, \forall x \in S : h_x^{\tau}\}, SK_{init,2} = g^{\alpha + a\tau} >.$$

If $c = 1$, it selects random $U, \{\forall x \in S : V_x\}, W \in \mathbb{G}_1$ and stores the entry in T_{GenQ} as follows:

$$< S, c = 1, SK_{init,1} = \{U, \forall x \in S : V_x\}, SK_{init,2} = W >.$$

Finally, it returns the tuple $\{SK_{ulti,1}, SK_{ulti,2}\}$ to A.

2). Corruption query: O initializes a table:

$$T_{CorQ} = < S, SK_{ulti,1}, SK_{ulti,2}, CK_{ulti} >.$$

When the adversary issues a corruption query with an attribute set S, O searches for S in T_{GenQ}. If there is no matched result, O aborts the game and outputs $\perp_1$ to indicate an error. Otherwise, if $c = 0$, it also aborts the game and outputs $\perp_2$ to indicate an error, and if $c = 1$, it checks the corresponding entry in T_{CorQ} and returns CK_{ulti} to A. If there is no matched result, it selects a random exponent $\psi \in Z_p^*$ and stores the entry in T_{CorQ} as follows:

$$< S, SK_{ulti,1} = SK_{init,1}^{\psi}, SK_{ulti,2}^{\psi}, CK_{ulti} = \psi >.$$

Finally, it returns CK_{ulti} to A.

- **Challenge:** A submits two pieces of health records, HRs_0 and HRs_1, of equal length. Then, O transfers these two portions of health records to N. The challenger N chooses a random bit $\beta \in \{0,1\}$ and encrypt HRs_{β} under the universal public parameters PP and the challenge access policy $(\Lambda, \rho)^*$. Then, N returns the ciphertext CT^* to O. Finally, CT^* is transferred to A as the challenge ciphertext.
- **Query phase 2:** A continues to adaptively and repeatedly issue queries, as in phase 1, and O and N also continue to work as in phase 1.
- **Guess:** A outputs β' as its guess for β. The algorithm O simultaneously gives β' to the challenger as its guess. If $\beta' = \beta$, A wins the game.

If O does not abort the game, the responses of the generation query and the corruption query are valid and indistinguishable. Let q_{GenQ} and q_{CorQ} be the maximum numbers of generation queries and corruption queries. The probability that the game is not aborted is:

$$\Pr[(!\perp_1)\ \&\ \&(!\perp_2)]=((\frac{q_{GenQ}}{2^l})(1-\varphi))^{q_{CorQ}} .$$

Consequently, if A can attack our proposed CCP-ABAC-UA in the selective CPA game with advantage ε, O can break the basic CCP-ABAC-UA in the selective CPA game with advantage ε', which satisfies:

$$\varepsilon' = ((\frac{q_{GenQ}}{2^l})(1-\varphi))^{q_{CorQ}}\,\varepsilon \le (\frac{q_{GenQ}}{2^l})^{q_{CorQ}}\,\varepsilon .$$

6. Performance Analysis

As described above, we introduce an enhanced access control system. We summarized some aspects of the performance comparison among some representative ABEs [6,8,10,17], current attribute-based PHR systems [26,27,29] and CCP-ABAC-UA for PHR system. The performance comparison on security is presented in **Table 1**.

Table 1. Perfomance comparison on security

Schemes	Escrow free	Exposure free	User Accountability	Provable Security
BSW07[6]	×	×	×	CPA-secure[1]
CC09[17]	✓	×	×	×
GSW11[8]	×	×	×	×
Hur13[10]	✓	×	×	×
QDWDNZ15[27]	×	×	×	CPA-secure
HCS16[26]	×	×	✓	CPA-secure
Rao17[29]	×	×	×	CCA-secure[2]
Our scheme	✓	✓	✓	CPA-secure

[1]Secure against chosen plaintext attacks; [2]Secure against chosen ciphertext attacks

CCP-ABAC-UA provides synchronous generation and distributed storage of private keys among the key authority, the PHR cloud, and a PHR receiver and each of these entities does not know what information is kept by the others. This coordinated mechanism helps solve both key the escrow and key exposure problems. In addition, the identity information of a private key's true owner is embedded into the ultimate client key so the system can verify whether the private key belongs to this PHR receiver when implementing decryption. If key abuse occurs, our scheme can identify the traitor by scanning the receiver list. Furthermore, given a novel provably secure construction, we prove that our scheme is a CPA-secure scheme. Therefore, compared to those schemes, CCP-ABAC-UA has a totally better performance on security.

Our scheme is lighter in user-side computation than previous schemes. To give a reasonable analysis, we present some notations with respect to computational efficiency in **Table 2**.

Table 2. Notations with respect to user-side computational efficiency

$\|PP\|$	Size of public parameters
$\|CT'\|$	Size of a ciphertext received by a data receiver
$\|CK\|$	Size of a private key held by a data receiver
C_p^u	Total times of user-side bilinear pairing computation
$\hat{\mathbb{G}}_*$	Bit size in storage of an element in $\mathbb{G}_*$
$\hat{Z}_p^*$	Bit size in storage of an element in Z_p^*
$\|\Omega\|$	Total numbers of authorized attribute space
$\|Y\|$	Total numbers of attributes in an access policy
$\|S\|$	Total number of attributes in an attribute set

We give a detailed comparison on user-side computational efficiency with four aspects, which is the size of public parameters, the size of a ciphertext received by a data receiver, the size of a private key held by a data receiver, and the total times of user-side bilinear pairing computation. The size of public parameters has a positive correlation with the user-side encryption overhead of a data owner, and other aspects have a positive correlation with user-side decryption overhead. Note that system parameters such as universal attribute group, cyclic groups and their generators are not taken into consideration in size assessment of public parameters because they can be set in advance of encryption. The detailed comparison on user-side computational efficiency is present in **Table 3**.

Table 3. Perfomance comparison on user-side computational efficiency

Schemes	$\|PP\|$	$\|CT\|$	$\|CK\|$	C_p^u
BSW07[6]	$2\hat{\mathbb{G}}_1+\hat{\mathbb{G}}_T$	$(2\|Y\|+1)\hat{\mathbb{G}}_1+\hat{\mathbb{G}}_T$	$(2\|S\|+1)\hat{\mathbb{G}}_1$	$2\|S\|+1$
CC09[17]	$N\hat{\mathbb{G}}_1+\|\Omega\|\hat{\mathbb{G}}_2+\hat{\mathbb{G}}_T$	$(\|Y\|+1)\hat{\mathbb{G}}_2+\hat{\mathbb{G}}_T$	$N\hat{\mathbb{G}}_1$	$\|S\|+1$
GHW11[8]	$\hat{\mathbb{G}}_1+\hat{\mathbb{G}}_T$	$2\hat{\mathbb{G}}_T$	$\hat{Z}_p^*$	0
Hur13[10]	$2\hat{\mathbb{G}}_1+\hat{\mathbb{G}}_T$	$(2\|Y\|+1+\sum_{i=1}^{\|Y\|}m_i)\hat{\mathbb{G}}_1+\hat{\mathbb{G}}_T$	$(2\|S\|+2)\hat{\mathbb{G}}_1$	$2\|S\|+2$
QDWDNZ15[27]	$(L+D+1)\hat{\mathbb{G}}_1+\hat{\mathbb{G}}_2+\hat{\mathbb{G}}_T$	$(2\|S\|+1)\hat{\mathbb{G}}_1+\hat{\mathbb{G}}_T$	$(\|Y\|\times L)\hat{\mathbb{G}}$	$3\|S\|$
HCS16[26]	$(\|\Omega\|+2)\hat{\mathbb{G}}_1$	$(2\|Y\|+2)\hat{\mathbb{G}}_1$	$(\|S\|+1)\hat{\mathbb{G}}_1$	0
Rao17[29]	$(\|\Omega\|+l+4)\hat{\mathbb{G}}_1+\hat{\mathbb{G}}_T$	$(\|Y_e\|+\|Y_s\|+3)\hat{\mathbb{G}}_1+\hat{\mathbb{G}}_T$	$(\|S\|+2)\hat{\mathbb{G}}_1$	$\|Y_s\|+5$
Our scheme	$\|\Omega\|\hat{Z}_p^*+\hat{\mathbb{G}}_1+\hat{\mathbb{G}}_T$	$2\hat{\mathbb{G}}_T$	$\hat{Z}_p^*$	0

For Chase and Chow's scheme [17], N represents the total numbers of key authorities. Its multi-authority mechanism causes too much computation overhead for both data owner and data receiver. For Hur's scheme [10], it introduces attribute groups to add a header message in each ciphertext so that fine-grained revocation is realized while the size of a ciphertext received by a data receiver increases. Noting that m_i represents the total number of data receivers that share the ith attribute in an access policy. For Qin *et al.*'s scheme [27], it introduces a attribute matrix with L rows and D columns to represent hierarchy of all authorized attributes. Meanwhile, their construction is built based on composite-order bilinear pairing computation. Thus, their user-side computation overhead is very large. To realize both secure data sharing and public data verifiability, Rao [29] introduces two access policy of which $|Y_e|$ represents the total number of attributes in encryption access policy and $|Y_e|$ represents the total number of signing access policy.

As is summarized in **Table 3**, the size of public parameters in our scheme seems a little bit large because it generates a group of elements corresponding to each attribute in authorized attribute space. But it has trivial impact on encryption overhead of data owner. It is worth noting that the size of ciphertext received by a data receiver and the size of a private key held by a data receiver in our scheme and Green *et al.*'s scheme [8] are the smallest among those schemes. Meanwhile, CCP-ABAC-UA does not need a PHR receiver to execute any bilinear computation, which is identical with [8] and [26]. Therefore, it is reasonable to claim that CCP-ABAC-UA is user-side lightweight scheme. Due to its coordinated mechanism, our scheme only supports on-line decryption, because we consider it insecure and inefficient to add off-line mode.

7. Conclusions

In this paper, we propose a coordinated ciphertext policy attribute-based PHR access control with user accountability (CCP-ABAC-UA). Its coordinated mechanism provides synchronous generation and distributed storage of private keys to effectively prevent escrow and exposure of private keys. During decryption, it can accurately detect whether key abuse is taking place and identify the traitor. The CCP-ABAC-UA is also a user-side lightweight scheme, in which no bilinear pairing computations are needed for PHR receivers to access data, so a secure mobile PHR application can be realized with a small amount of user-side computational overhead. By introducing a novel provably secure construction of CCP-ABAC-UA, we prove that it is secure against selectively chosen-plaintext attacks. Our future work will concentrate on further reducing the decryption overhead and ciphertext size.

References

[1] L. Lanranjo, A. L. Neves, T. Vilanueva, J. Cruz, A. Brito de Sa and C. Sakellarides, "Patient's Access to their Medical Records," *Acta Medica Portuguesa*, vol. 26, no. 3, pp. 265-270, 2013. Article (CrossRef Link)

[2] C. Pearce and M. Bainbridge, "A personally Controlled Electronic Health Record for Australia," *Journal of the American Medical Informatics Association*, vol. 21, no. 4, pp. 707-713, 2014. Article (CrossRef Link)

[3] K. T. Win, "A Review of Security of electronic Health Records," *The HIM Journal*, vol. 34, no. 1, pp. 13-18, 2005. Article (CrossRef Link)

[4] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Proc. of 24th International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 457-473, May 22-26, 2005. Article (CrossRef Link)

[5] M. Pirretti, P. Traynor, P. McDaniel and B. Waters, "Secure attribute-based systems," in *Proc. of 13th ACM Conference on Computer and Communications Security*, pp. 99-112, October 30-November 03, 2006. Article (CrossRef Link)

[6] J. Benthencourt, A. Sahai and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. of 3rd International Conference on Pairing-based Cryptography*, pp.321-334, May 20-23, 2007. Article (CrossRef Link)

[7] J. Lai, R. H. Deng, C. Guan and J. Weng, "Attribute-based encryption with verifiable outsourcing systems," *IEEE Trans. Inf. Forens. Security*, vol. 8, no. 8, pp. 1343-1354, 2013. Article (CrossRef Link)

[8] M. Green, S. Hohenberger and B. Waters, "Outsourcing the decryption of ABE ciphertexts," in *Proc. of 20th USENIX Security Symposium*, pp. 34, August 8-12, 2011. Article (CrossRef Link)

[9] M. S. Ahmad, N. E. Musa, R. Nadarajah, R. Hassan and N. E. Othman, "Comparison between android and iOS Operating System in terms of security," in *Proc. of 8th International Conference on Information Technology in Asia*, pp. 1-4, July 1-4, 2013. Article (CrossRef Link)

[10] J. Hur, "Improving security and efficiency in attribute-based data sharing," *IEEE Trans. Knowledge and Data Engineering*, vol. 25, no. 10, pp. 2271-2282, 2013. Article (CrossRef Link)

[11] N. Attrapadung and H. Imai, "Conjunctive broadcast and attribute-based encryption," in *Proc. of 3rd Int. Conference on Paring-Based Cryptography*, pp. 248-265, August 12-14, 2009. Article (CrossRef Link)

[12] B. Waters, "Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization," in *Proc. of 14th Int. Conference on Practice and Theory in Public Key Cryptography*, pp. 53-70, March 6-9, 2011. Article (CrossRef Link)

[13] R. Canetti, H. Krawczyk and J. B. Nielsen, "Relaxing chosen-ciphertext security," in *Proc. of 23rd Annual International Cryptology Conference*, pp. 565-582, August 17-21, 2003. Article (CrossRef Link)

[14] V. Goyal, O. Pandey, A. Sahai and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. of 13th ACM Conference on Computer and Communications Security*, pp. 89-98, October 30-November 3, 2006. Article (CrossRef Link)

[15] G. Wungpornpaiboon and S. Vasupongayya, "Two-layer Ciphertext-Policy Attribute-Based Proxy Re-Encryption for Supporting PHR Delegation," in *Proc. of 19th International Computer Science and Engineering Conference*, pp. 1-6, November 23-26, 2015. Article (CrossRef Link)

[16] M. Li, S. Yu, Y. Zheng, K. Ren and W. Lou, "Scalable and Secure Sharing of Personal Health Records in Cloud Computing Using Attribute-Based Encryption," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 1, pp. 131-143, 2013. Article (CrossRef Link)

[17] M. Chase and S. S. M. Chow, "Improving privacy and security in multi-authority attribute-based encryption," in *Proc. of 16th ACM Conference on Computer and Communications Security*, pp. 121-130, November 9-13, 2009. Article (CrossRef Link)

[18] G. Zhang, L. Liu and Y. Liu, "An attribute-based encryption scheme secure against malicious KGC," in *Proc. of 11th IEEE Conference on Trust, Security and Privacy in Computing and Communications*, pp. 1376-1380, June 25-27, 2012. Article (CrossRef Link)

[19] P. P. Chandar, D. Mutkurman and M. Rathinrai, "Hierarchical attribute based proxy re-encryption access control in cloud computing," in *Proc. of International Conference in Circuits, Power and Computing Technologies*, pp. 1565-1570, March 20-21, 2014. Article (CrossRef Link)

[20] F. Xhafa, J. F. Wang, X. F. Chen, J. K. Liu, J. Li and P. Krause, "An Efficient PHR Service System Supporting Fuzzy Keyword Search and Fine-Grained Access Control," *Soft Computing*, vol. 18, no. 9, pp. 1795-1802, 2014. Article (CrossRef Link)

[21] S. S. M. Chow, "Removing escrow from identity-based encryption," in *Proc. of 12th International Conference on Practice and Theory in Public Key Cryptography*, pp. 256-276, March 18-20, 2009. Article (CrossRef Link)

[22] A. Roehrs, C. A. da Costa, K. S. F. de Oliveira, "Personal Health Records: A Systematic Literature Review," *Journal of Medical Internet Research*, vol. 19, no. 1, pp. 100-120, 2017. Article (CrossRef Link)

[23] T. S. Chen, C. H. Liu, C. S. Chen, J. G. Bau and T. C. Lin, "Secure Dynamic Access Control Scheme of PHR in Cloud Computing," *Journal of Medical System*, vol. 26, no. 6, pp. 4005-4020, 2012. Article (CrossRef Link)

[24] H. L. Qian, J. G. Li, Y. C. Zhang and J. G. Han, "Privacy Preserving Personal Health Record Using Multi-Authority Attribute-Based Encryption with Revocation," *International Journal of Information Security*, vol. 14, no. 6, pp. 487-497, 2015. Article (CrossRef Link)

[25] F. Xhafa, J. Feng and Y. Zhang, "Privacy-Aware Attribute-Based PHR Sharing with User Accountability in Cloud Computing," *Journal of Supercomputing*, vol. 71, no. 5, pp. 1607-1619, 2015. Article (CrossRef Link)

[26] H. Hong, D. Chen and Z. Sun, "A Practical Application of CP-ABE for Mobile PHR System: A Study on User Accountability," *SpringerPlus*, vol. 5, no. 1, pp. 1320, 2016. Article (CrossRef Link)

[27] B. Qin, H. Deng, Q. H. Wu, J. Domingo-Ferrer, D. Naccache and Y. Y. Zhou, "Flexible Attribute-Based Encryption Applicable to Secure E-Healthcare Records," *International Journal of Information Security*, vol. 14, no. 6, pp. 499-511, 2015. Article (CrossRef Link)

[28] L. Zhang, Q. Wu, Y. Mu and J. Zhang, "Privacy-Preserving and Secure Sharing of PHR in the Cloud," *Journal of Medical Systems*, vol. 40, no. 12, pp. 267, 2016. Article (CrossRef Link)

[29] Y. Sreenivasa. Rao, "A secure and efficient Ciphertext-Policy Attribute-Based Signcryption for Personal Health Records sharing in cloud computing," *Future Generation Computer Systems*, vol. 67, pp. 133-151, 2017. Article (CrossRef Link)

[30] Y. B. Miao, J. F. Ma, X. Liu, F. S. Wei, Z. Q. Liu and X. A. Wang, "m(2)-ABKS: Attribute-Based Multi-Keyword Search over Encrypted Personal Health Records in Multi-Owner Setting," *Journal of Medical Systems*, vol. 40, no. 11, pp. 246, 2016. Article (CrossRef Link)

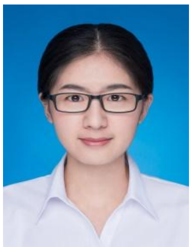
[31] N. Fernandez, D. J. Copenhaver, D. K. Vawdrey, H. Kotchoubey and M. S. Stockwell, "Smartphone Use Among Postpartum Women and Implications for Personal Health Record Utilization," *Clinical Pediatrics*, vol. 56, no. 4, pp. 376-381, 2017. Article (CrossRef Link)

[32] M. Bachiri, A. Idri, J. L. Fernandez-Aleman and A. Toval, "Mobile personal health records for pregnancy monitoring functionalities: Analysis and potential," *Computer Methods and Programs in Biomedicine*, vol. 134, pp. 121-135, 2016. Article (CrossRef Link)

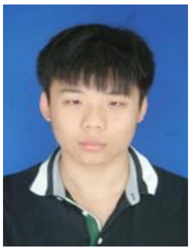
[33] H. Yoo and K. Chung, "PHR Based Diabetes Index Service Model Using Life Behavior Analysis," *Wireless Personal Communications*, vol. 93, no. 1, pp. 161-174, 2017. Article (CrossRef Link)

Guofeng Lin received the B.Eng. degree in electronic information engineering from Nantong University, Nantong, China. He is currently pursuing the M.D.-Ph.D. degree in information network from Nanjing University of Posts and Telecommunications. His primary research interests are cloud computing, network security, and cryptography.

Lirong You received the B. Eng. degree in information system and management from Chanzhou Unversity, Changzhou, China. She is currently undertaking information system development in Jiangsu Zhongtian Technology Limited Company.



Bing Hu received the B.Eng. degree in computer science and technology from The PLA Information Engineering University. She is now a Ph.D. candidate of Nanjing University of Posts and Telecommunications. Her research interests include network security, cloud computing, and wireless sensor network communication.



Hanshu Hong received the B.Eng. degree in network engineering from Nanjing University of Posts and Telecommunications, Nanjing, China. He is currently pursuing the M.D.-Ph.D. degree in information network from Nanjing University of Posts and Telecommunications. His research interests are information security and cryptography.

Zhixin Sun received the Ph.D. degree from Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 1998. From 2001 to 2002, he held a post-doctoral position with School of Engineering, Seoul National University, South Korea. He is currently a Professor and the Dean of School of Modern Posts, Nanjing University of Posts and Telecommunications. His research interests include cloud computing, information security.