Searchable Encryption with Autonomous Path Delegation Function and Its Application in Healthcare Cloud

Qian Wang, Chengzhe Lai, *Member, IEEE,* Rongxing Lu, *Fellow, IEEE,* and Dong Zheng

Abstract—Outsourcing medical data to healthcare cloud has become a popular trend. Since medical data of patients contain sensitive personal information, they should be encrypted before outsourcing. However, information retrieval methods based on plaintext cannot be directly applied to encrypted data. In this paper, we present a new cryptographic primitive named conjunctive keyword search with secure channel free and autonomous path delegation function (AP-SCF-PECKS), which can be applied in scenarios where patients want to search for and autonomous delegate their private medical information without revealing their private key. Particularly, the proposed solution allows patients to set up multi-hop delegation path with their preferences, and the delegated doctors in the path can search for and access the patient's private medical information with priority from high to low. Patients can ensure that authorized doctors are always trustworthy, and unauthorized users cannot obtain the private medical information of patients. Moreover, the scheme supports the conjunctive keyword search, secure channel free, and is secure against chosen keyword attack, chosen ciphertext attack, and keyword guessing attack. The security of proposed scheme has been formally proved in the standard model. Finally, the performance evaluations demonstrate that the overhead of proposed scheme are modest for healthcare cloud scenarios.

Index Terms—searchable encryption, healthcare cloud, autonomous path delegation, proxy re-encryption, conjunctive keyword, secure channel free.

1 Introduction

With the popularization and development of healthcare cloud, personal electronic health record (EHR) has been widely used, which can effectively reduce medical errors [1]. At the same time, hospitals need to keep the medical records of outpatient and inpatient for a long time. Local storage will take up a lot of storage space, which can not realize the sharing of medical data between hospitals and bring inconvenience to patients. Therefore, hospitals can outsource EHRs to the healthcare cloud. However, EHRs usually include sensitive information of patients, and outsourcing to healthcare cloud in plaintext will face the risk of privacy leakage. In order to ensure the privacy of patients, EHR should be encrypted before outsourcing, while information retrieval method based on plaintext can not be directly used to encrypted data, which will cause great inconvenience to obtain some medical records over the healthcare cloud. Public key encryption with keyword search (PEKS) [2] can retrieve ciphertext directly. With the use of PEKS, hospitals can encrypt EHRs and outsource them to healthcare cloud, so that patients can search their EHRs without decryption, which can protect the privacy of patients while providing convenience for patients, doctors, and practitioners.

In actual medical situations, a patient usually makes appointment with different doctors, and needs to share his EHRs with multiple doctors. In traditional PEKS method, hospitals outsource encrypted EHRs to healthcare cloud, only the patient can search for his EHRs, while doctors have no search rights. To solve this issue, proxy re-encryption (PRE) [3] method was introduced, it allows patient (delegator) to delegate his encrypted EHRs' search rights to doctors (delegatees), so that the doctors can search and access them. The delegation need to convert the encrypted EHRs through semi-trusted proxy, and in the course of conversion, proxy does not know the delegator and delegatee's private keys, nor can it learn any information about the EHRs.

However, if the scheduled doctor cannot provide timely service to the patient for some reason, the patient needs to make an appointment with another doctor that he expects or trusts. In fact, making appointment is very difficult in some areas, patient usually needs to make multiple appointments. In some existing research on searchable proxy re-encryption [4], [5], [6], [7], the patient needs to perform delegation multiple times, which will bring inconvenience to the patient. Otherwise, the scheduled doctor needs to delegate the patient to another doctor of his choice, while the second doctor may not be expected and trusted by the patient, and with more times of delegations, the lower the trust between patient and doctor. Even if the patient sets a delegation list in advance and lets the delegation execute according to the list, the delegation is still easy to transfer to other users who are not in the list. None of them can be applied perfectly to this kind of scenario.

- *Qian Wang is with College of Computer, Qinghai Normal University, and also with School of Computer Science and Technology, Xi'an University of Posts and Telecommunications, and Shaanxi Key Laboratory of Network Data Analysis and Intelligent Processing.*
- *Chengzhe Lai is with School of Cyberspace Security, Xi'an University of Posts and Telecommunications.*
- *Dong Zheng is with College of Computer, Qinghai Normal University, and also with School of Cyberspace Security, Xi'an University of Posts and Telecommunications.*
- *Rongxing Lu is with Faculty of Computer Science, University of New Brunswick.*

In this paper, we endeavor to study a novel mechanism and apply it to this kind of scenario. User can search and access his encrypted data, then he also can act as a delegator to sets a multi-hop delegation path in advance according to his preference, so that delegatees in the path have high-to-low search and access rights of the encrypted data. If the first delegatee cannot search and access, the delegation will be automatically transferred to the next delegatee in the path. The delegation will be transferred one by one, until there is a search query by a delegatee in the path, or there is no search query after traversing the path. In this way, each authorized doctor is expected or trusted by the patient, even if the patient is not always online.

As far as we know, this is the first work to implement autonomous path delegation in searchable proxy re-encryption scheme. A conjunctive keyword searchable encryption scheme with secure channel free and autonomous path delegation function (AP-SCF-PECKS) is proposed, which has the following merits.

1) We present a novel searchable encryption scheme that supports autonomous path delegation and conjunctive keyword search functions. Users can search and access his encrypted data through single or multiple keywords, then the user can act as a delegator to delegate the search and access right of user's encrypted data to other users. Meanwhile, the proposed scheme can support secure channel free, which indicates that only the designated tester, usually the server, has right to carry on the test algorithm to check the relationship between trapdoor and ciphertext.
2) The delegation in the proposed scheme strictly follows the delegation path. The delegator sets up a multi-hop delegation path in advance, and the delegatees in the path have search and access rights with priority from high to low. At the same time, original ciphertext and re-encrypted ciphertext cannot be converted and inserted into other paths through meaningful decryption operations. In this way, the delegation strictly follows the delegation path and cannot be transferred to other paths.
3) The security of the proposed scheme has been formally proved against chosen keyword attack and chosen ciphertext attack in standard model. It is also able to resist keyword guessing (KG) attack. Furthermore, we compare the functions and theoretical performances with other relevant schemes, and demonstrate the experimental simulation results. Our scheme can support multiple useful functions, and the overhead are acceptable in healthcare cloud application.

The rest of the paper is organized as follows. In Section 2, we introduce the related works, then we formulate the problem in Section 3. In Section 4, we introduce the preliminary knowledge of bilinear mapping and the assumptions which our AP-SCF-PECKS scheme is based on. In Section 5, we introduce the concept and security model of AP-SCF-PECKS scheme. In Section 6, we describe details of the construction and give the revocation mechanism, then apply it to healthcare cloud. In Section 7, we give the security analysis, then we evaluate the scheme's performance in Section 8. Finally, we conclude this paper in Section 9.

2 Related Works

2.1 Public Key Encryption with Conjunctive Keyword Search

Public key encryption with keyword search (PEKS) is very important to protect the privacy of outsourced data. Boneh et al. [2] proposed a generic conversion from anonymous identity-based encryption to PEKS in 2004, and applied it to encrypted e-mail system. This is the first time that the concept of PEKS was proposed. After that, many kinds of work in PEKS were proposed and applied to the cloud. [8] proposed a lightweight PEKS scheme with semantic security for cloud-assisted wireless sensor networks, which greatly reduces the computation intensive operations, and the search performance is close to some practical symmetric searchable encryption schemes. [9] proposed a designated cloud server PEKS scheme, only the designated cloud server can complete the test. The scheme is based on lattice assumption which can resist the attack of quantum computer. [10] proposed an expressive PEKS scheme that allows keyword search policy to be expressed by conjunctive, disjunctive or any monotonic Boolean formulas, and the performance is greatly improved compared with the existing schemes. These PEKS schemes have different functions and can be applied in different scenarios.

Conjunctive keyword search is a very useful function of searchable encryption. It can provide multiple keywords search concurrently, instead of querying single keyword multiple times and obtaining the need through intersection calculation. Golle et al. [11] and Park et al. [12] first studied this problem on encrypted data. Among them, [12] proposed two constructions of public key encryption with conjunctive keyword search (PECKS) in 2004. Subsequently, various constructions of PECKS were proposed. [7] proposed the scheme with timing enabled function, which also has the function of designed tester and proxy re-encryption. However, the searchable encrypted index is easy to transfer the search right to the user who has not been delegated. [13] proposed a lightweight fine grained PECKS system, which also has the function of attribute update, and it can avoid illegal accesses and returning irrelevant search results. [14] showed two PEKS schemes with extended functionalities, one supports conjunctive queries, another supports subset queries and some more general predicates.

Recently, Sultan et al. proposed two keyword search schemes [15] and [16], which can realize conjunctive keyword search with minimal cost. Among them, [15] adopts role-based encryption (RBE) technique, the authorized users with proper roles that satisfy the defined role-based access control policy can search and access the plaintext through the efficient decryption. The roles are organized in a hierarchy, and the ancestor roles can inherit the access privileges of their descendant roles. The scheme is suitable for multi organization cloud environment and supports user revocation, it also can resist chosen plaintext and chosen keyword attacks. [16] relying on ciphertext-policy attribute-based encryption (CP-ABE), the authorized users with qualified set of attributes can search and access the ciphertext.

2.2 Secure Channel Free Public Key Encryption with Keyword Search

Byun et al. [17] and Yau et al. [18] pointed out that many classic PEKS schemes are vulnerable to KG attack. At the same time, Baek et al. [19] proposed that the PEKS scheme in [2] requires a secure channel between users and server, which means that users cannot use an actual untrusted channel, such as public WiFi or 5G etc, or at least require an expensive secure socket layer connection. Obviously, it is quite impractical. Then they proposed the concept of PEKS scheme without secure channel, called secure channel free public key encryption with keyword search (SCF-PEKS), sometimes also called PEKS with a designated tester (d-PEKS) [20], which can resist KG attack.

Following this work, various constructions of SCF-PEKS were proposed. Such as [21] and [7] proposed SCF-PEKS schemes without random oracle. The construction of them are based on the different pairing, but both of them can resist chosen keyword, chosen ciphertext, and KG attack. [7] also supports the function of proxy search. Recently, Suzuki et al. [22] presented a construction of integrated SCF-PEKS and public key encryption (PKE), the purpose is to solve the problem that PEKS does not support decryption, then apply the scheme to electronic medical record in cloud storage. Lu et al. [23] proposed a certificate-based SCF-PEKS scheme under the random oracle model, and the scheme can support functions of implicit authentication and no key escrow. [16] proposed an expressive authorized keyword search scheme which also supports multiple functions, including secure channel free, conjunctive keyword search, effective attribute revocation and semantically secure against chosen keyword and KG attacks.

2.3 Searchable Proxy Re-encryption

Proxy re-encryption (PRE) enables the proxy transforms the ciphertext encrypted by the delegator’s public key into a ciphertext that can be decrypted by the delegatee’s private key. Blaze et al. [3] first presented the concept of PRE at Eurocrypt’98. They pointed out two ways to classify PRE scheme, one is divided into multi-hop and single-hop according to the number of allowed transformation, another is divided into bidirectional and unidirectional according to the direction of allowed transformation. Subsequently, combined with different application requirements, various PRE schemes with different properties were proposed. [24] proposed a non-transferable PRE scheme. [25] designed a unidirectional PRE scheme, and applied the scheme in secure social cloud storage system. [26] proposed a PRE scheme with autonomous path delegation function in random oracle model, and it cannot convert the ciphertext to other paths.

In 2010, Shao et al. [5] first proposed the concept of proxy re-encryption with keyword search (PRES), which integrated PEKS and PRE. However, the scheme encrypts document and keyword by using same encryption algorithm. Yau et al. extended PEKS in [6], and presented a new concept for searchable proxy re-encryption scheme (Re-PEKS) by separating document encryption and keyword encryption. The restriction of [5] and [6] is that only one keyword can be searched, and all of them have been proved to be secure in random oracle model. Recently, Yang et al.

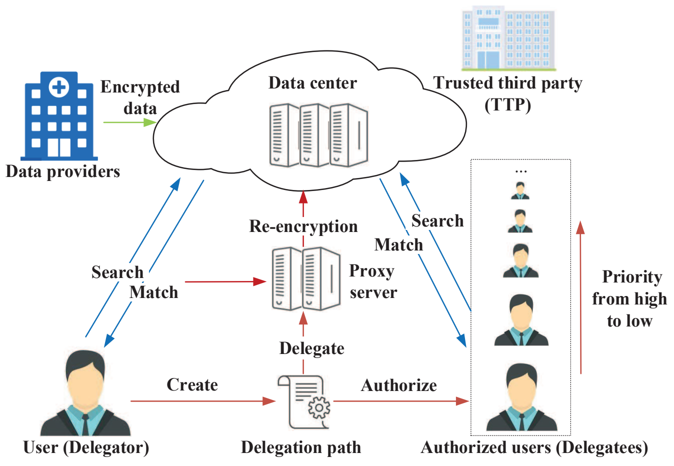

Fig. 1: System Model

[7] presented a conjunctive keyword search and proxy re-encryption scheme in standard model. However, if the user creates a multi-hop delegation path in advance, [5] and [6] will easily transfer the search right to other users who are not in the path. [7] is a Re-PEKS scheme with one-time timing delegation function. None of them can well support autonomous path delegation function.

3 PROBLEM FORMULATION

3.1 System Model

Fig. 1 shows the system model of the AP-SCF-PECKS scheme, which consists of entities: trusted third party (TTP), data provider, user, third-party data center and proxy server. The TTP is responsible for generating key for users and data center, and generating re-encryption key for proxy server. Data providers outsource encrypted data to third-party data center. Then users can search their encrypted data over the data center, and they also can act as delegators to create delegation paths to delegate their encrypted data. When a user create a delegation path, the delegatees (i.e., authorized users) in the path have high-to-low rights to search the encrypted data. The first delegatee in the path has right to search firstly, then the second. Delegation will be transferred one by one, until there is a search query by a delegatee in the path, or there is no search query after traversing the path. The proxy re-encrypts the encrypted data for each delegatee. Only the user and the authorized users can obtain the matching ciphertexts by sending search query. After receiving the search request, the data center performs search operation, then returns the matching ciphertexts to the user or authorized user.

3.2 Threat Model

The third-party data center is deemed as semi-trusted, who is honest to search information for the users, but it also spy out the private information of the user curiously. On the other hand, malicious outside attackers can eavesdrop data from public transmission channel, such as searchable encrypted indexes and trapdoors, and also can analyze and infer privacy information from the eavesdropped data.

Furthermore, users may intend to access data that he has not access right. Since most of the storage and search work are completed by the data center, i.e. the data server, it is assumed that the data server will not collude with users, nor will it collude with malicious outside attackers.

3.3 Design Goals

Our AP-SCF-PECKS scheme is designed to achieve the following goals.

1) ***Conjunctive keyword search with autonomous path delegation function***. The scheme should provide multiple keywords search for ciphertext and allow delegator autonomous set up delegation path. The delegator delegates the search rights of his private information to the delegatees in the path without exposing his private key. Furthermore, the private information of the delegator should be kept confidential for unauthorized users and cloud data center.
2) ***Delegation strictly follows the delegation path.*** Another challenge of the scheme is that the ciphertext delegation is strictly follows the delegation path and cannot be converted to other paths. The delegator sets up a delegation path, the re-encryption ciphertext along the path cannot be converted and inserted into other paths through meaningful decryption operations. At the same time, the original ciphertext cannot be converted and inserted into other paths through meaningful decryption operations.
3) ***Security goals***. The security of the scheme are summarized as follows. 1) *resist chosen keyword and chosen ciphertext attack*: we will prove the scheme indistinguishable against chosen keyword and chosen ciphertext attack (IND-AP-SCF-CKCA). 2) *resist KG attack*: since the partial keywords of EHR are always chosen from a small space, the scheme should be indistinguishable against keyword guessing attack (IND-KGA). 3) *standard model*: The security of the scheme proved in standard model is stronger than that in random oracle model. The proposed scheme needs to be verified in standard model to ensure higher security.

4 PRELIMINARIES

4.1 Bilinear Map

Definition1: (**Bilinear Map**) Let G_1 and G_2 be two multiplicative groups of the same prime order p, g be a generator of G_1, $\tilde{g}$ be a generator of G_2. Assume that the discrete logarithm problems in G_1 and G_2 are intractable. We say that e: $G_1 \times G_2 \to G_T$ is a bilinear map, if it satisfies the following properties:

1) *Bilinear*: For all $a, b \in Z_p$, $e(g^a, \tilde{g}^b) = e(g, \tilde{g})^{ab}$.
2) *Non-degenerate*: $e(g, \tilde{g}) \neq 1$.
3) *Computable*: For all $g_1 \in G_1$ and $\tilde{g}_2 \in G_2$, there is an efficient algorithm to compute $e(g_1, \tilde{g}_2)$.

We represent $Setup_{BM}(k)$ as an algorithm, which inputs the security parameter k, and outputs the bilinear mapping parameters $\{p, G_1, G_2, G_T, e, g, \tilde{g}\}$.

4.2 Computational Assumption

Definition2: (**Decisional Diffie-Hellman Assumption (DDH)**) Let G be a multiplicative group of prime order p, g be a generator of G. The advantage function $Adv_{G,\mathcal{A}}^{DDH}(k)$ of an adversary $\mathcal{A}$ is defined as

$$\left|\mathcal{P}r\left[\mathcal{A}(g, g^a, g^b, g^{ab}) = 1\right] - \mathcal{P}r\left[\mathcal{A}(g, g^a, g^b, g^r) = 1\right]\right|$$

where $a, b, r \in Z_p$ are randomly chosen. We say that the DDH assumption holds, if $Adv_{G,\mathcal{A}}^{DDH}(k)$ is negligible for any probabilistic polynomial time (PPT) $\mathcal{A}$.

Definition3: (**Symmetric External Diffie-Hellman Assumption (SXDH)**) On input the security parameter k, $Setup_{BM}(k)$ outputs the bilinear mapping parameters $\{p, G_1, G_2, G_T, e, g, \tilde{g}\}$. The bilinear map is e: $G_1 \times G_2 \to G_T$. We define the advantage function $Adv_{G_1,\mathcal{A}}^{SXDH}(k)$ and $Adv_{G_2,\mathcal{A}}^{SXDH}(k)$ of an adversary $\mathcal{A}$ respectively as

$$\left|\mathcal{P}r\left[\mathcal{A}(g, g^a, g^b, g^{ab}) = 1\right] - \mathcal{P}r\left[\mathcal{A}(g, g^a, g^b, g^r) = 1\right]\right|$$
$$\left|\mathcal{P}r\left[\mathcal{A}(\tilde{g}, \tilde{g}^a, \tilde{g}^b, \tilde{g}^{ab}) = 1\right] - \mathcal{P}r\left[\mathcal{A}(\tilde{g}, \tilde{g}^a, \tilde{g}^b, \tilde{g}^r) = 1\right]\right|$$

where $a, b, r \in Z_p$ are randomly chosen. We say that the SXDH assumption holds if both $Adv_{G_1,\mathcal{A}}^{SXDH}(k)$ and $Adv_{G_2,\mathcal{A}}^{SXDH}(k)$ are negligible for any PPT $\mathcal{A}$.

Definition4: (**Decisional Bilinear Diffie-Hellman Assumption (DBDH)**) On input the security parameter k, $Setup_{BM}(k)$ outputs the bilinear mapping parameters $\{p, G_1, G_2, G_T, e, g, \tilde{g}\}$. The bilinear map is e: $G_1 \times G_2 \to G_T$. We define the advantage function $Adv_{\mathcal{A}}^{DBDH}(k)$ of an adversary $\mathcal{A}$ as

$$\left|\mathcal{P}r\left[\mathcal{A}(g, g^a, g^b, g^c, \tilde{g}, \tilde{g}^a, \tilde{g}^b, \tilde{g}^c, e(g, \tilde{g})^{abc}) = 1\right]\right.$$
$$\left.- \mathcal{P}r\left[\mathcal{A}(g, g^a, g^b, g^c, \tilde{g}, \tilde{g}^a, \tilde{g}^b, \tilde{g}^c, e(g, \tilde{g})^r) = 1\right]\right|$$

where $a, b, c, r \in Z_p$ are randomly chosen. We say that the DBDH assumption holds if $Adv_{\mathcal{A}}^{DBDH}(k)$ is negligible for any PPT $\mathcal{A}$.

Definition5: (**Truncated Decisional *n*-Augmented Bilinear Diffie-Hellman Exponent Assumption (*n*-ABDHE)**) On input the security parameter k, $Setup_{BM}(k)$ outputs the bilinear mapping parameters $\{p, G_1, G_2, G_T, e, g, \tilde{g}\}$. The bilinear map is e: $G_1 \times G_2 \to G_T$. We define the advantage function $Adv_{\mathcal{A}}^{n-ABDHE}(k)$ of an adversary $\mathcal{A}$ as

$$\left|\mathcal{P}r\left[\mathcal{A}(g, g^a, \cdots g^{a^n}, \tilde{g}, \cdots \tilde{g}^{a^n}, \tilde{g}^z, \tilde{g}^{za^{n+2}}, e(g, \tilde{g})^{za^{n+1}}) = 1\right]\right.$$
$$\left.-\mathcal{P}r\left[\mathcal{A}(g, g^a, \cdots g^{a^n}, \tilde{g}, \tilde{g}^a, \cdots \tilde{g}^{a^n}, \tilde{g}^z, \tilde{g}^{za^{n+2}}, e(g, \tilde{g})^r) = 1\right]\right|$$

where $a, z, r \in Z_p$ are randomly chosen. We say that the n-ABDHE assumption holds if $Adv_{\mathcal{A}}^{n-ABDHE}(k)$ is negligible for any PPT $\mathcal{A}$.

5 DEFINITION AND SECURITY MODEL FOR AP-SCF-PECKS

In this section, we formally define the conjunctive keyword search with secure channel free and autonomous path delegation function (AP-SCF-PECKS), then we give the security model. The symbols used in this paper shown in Table 1.

TABLE 1: Summary of Notations

Notation	Description
GP	Global parameter
R_i, R_j	User (delegator), Authorized user (delegatee)
pk_s, sk_s	Public and private keys of data server
pk_i, sk_i	Public and private keys of R_i
pk_j, sk_j	Public and private keys of R_j
W, Q	Keyword set
Pa_i	Autonomous delegation path of R_i
l_i	The number of delegatees in Pa_i
$rk_{j-1\to j}$	Proxy re-encryption key for R_j
$C_i(C_j)$	Ciphertext for $R_i(R_j)$
$T_{i,Q}, (T_{j,Q})$	Trapdoor of $R_i(R_j)$ on Q

5.1 Definition of AP-SCF-PECKS

- $GlobalSetup(k)$: On input the security parameter k, this function generates a global parameter *GP*.
- $KeyGen_{Ser}(GP)$: On input the global parameter *GP*, this algorithm generates a public and private key pair (pk_s, sk_s) for data server.
- $KeyGen_R(GP)$: On input the global parameter *GP*, this algorithm generates a public and private key pair (pk_i, sk_i) for user R_i.
- $PECK(GP, pk_s, pk_i, W)$: On input the global parameter *GP*, public key pk_s of the data server, public key pk_i of the user R_i, a keyword set $W = (w_1, w_2, \cdots, w_n)$, this algorithm returns a ciphertext C_i of W for R_i.
- $Trapdoor_{R_i}(GP, pk_s, sk_i, Q)$: On input the global parameter *GP*, public key pk_s of the data server, private key sk_i of the user R_i, a keyword query set $Q = (q_1, q_2, \cdots, q_m), m \le n$. Then it outputs a trapdoor $T_{i,Q}$ for Q generated by R_i.
- $Test(GP, pk_s, sk_s, T_{i,Q}, C_i)$: On input the global parameter *GP*, key pair (pk_s, sk_s) of the data server, trapdoor $T_{i,Q}$ and ciphertext C_i. If W includes Q, it returns 1, otherwise it returns 0.

If R_i sets a delegation path, the following operations are executed. (Note that for the convenience of description, we represent R_i as the delegator which will set a delegation path, and R_j as the delegatee in the path. The key pair (pk_j, sk_j) generation of R_j is same as that of R_i.)

- $CreatPath(GP, R_i, R_j, j = 1, \cdots, l_i)$: On input the global parameter *GP*, delegator R_i, delegatees R_j where $j = 1, \cdots, l_i$. It outputs a delegation path $Pa_i = (R_0 = R_i, R_1, R_2, \cdots, R_{l_i})$. It's means that the path Pa_i is designated by the delegator R_i, and the path contains l_i delegatees, authorized user R_1 is the first delegatee, authorized user R_2 is the second and so on, authorized user R_{l_i} is the last.
- $ReKeyGen(GP, Pa_i, pk_s, sk_i, sk_j, j = 1, \cdots, l_i)$: On input the global parameter *GP*, delegation path Pa_i, public key pk_s of the data server, private key sk_i of the delegator R_i, and private key sk_j of delegatees R_j where $j = 1, \cdots, l_i$. This algorithm outputs l_i pairs proxy re-encryption key $rk_{j-1\to j} = (rk^1_{j-1\to j}, rk^2_{j-1\to j}, rk^3_{j-1\to j})$, $j = 1, \cdots, l_i$, then sends them to corresponding proxies via a secure channel. For $j = 1$, $rk_{j-1\to j} = rk_{0\to 1}$ denotes the re-encryption key from the delegator R_i to the first delegatee R_1. For $j > 1$, $rk_{j-1\to j}$ denotes the re-encryption key from the delegatee R_{j-1} to the next delegatee R_j.
- $RePECK(GP, Pa_i, rk_{j-1\to j}, C_{j-1}, 1 \le j \le l_i)$: On input the global parameter *GP*, delegation path Pa_i, re-encryption key $rk_{j-1\to j}$, and ciphertext C_{j-1}, $1 \le j \le l_i$. To re-encrypt the ciphertext under the public key pk_{j-1} to the one under pk_j in the path Pa_i, this algorithm first checks whether $(pk_{j-1}, pk_j) \in Pa_i$ and outputs '$\perp$' if not. Otherwise, it outputs re-encrypted ciphertext C_j for the delegatee R_j. For $j = 1$, ciphertext C_j generated by re-encrypting the original ciphertext C_0 (i.e. C_i), For $j > 1$, C_j generated by re-encrypting the re-encrypted ciphertext C_{j-1}.
- $Trapdoor_{R_j}(GP, pk_s, sk_j, Q)$: On input the global parameter *GP*, public key pk_s of the data server, private key sk_j of a delegatee R_j, a keyword query set $Q = (q_1, q_2, \cdots, q_m), m \le n$. Then it outputs a trapdoor $T_{j,Q}$ for Q generated by R_j.
- $Test_{Re}(GP, pk_s, sk_s, T_{j,Q}, C_j)$: On input the global parameter *GP*, key pair (pk_s, sk_s) of the data server, trapdoor $T_{j,Q}$ and ciphertext C_j. If W includes Q, it returns 1, otherwise it returns 0.

Correctness:

For any user R_i, we have that

$$Test \to \left\{ \begin{array}{ll} 1 & Q \subseteq W \\ 0 & otherwise \end{array} \right\}$$

If a delegator R_i creates a delegation path Pa_i, and a delegatee R_j belong to the path Pa_i, we have that

$$Test_{Re} \to \left\{ \begin{array}{ll} 1 & Q \subseteq W \\ 0 & otherwise \end{array} \right\}$$

5.2 Security Model of AP-SCF-PECKS

In this section, we give the game-based security definition of AP-SCF-PECKS scheme. The scheme is proved to be IND-AP-SCF-CKCA secure (indistinguishable against chosen keyword and chosen ciphertext attack) and IND-KGA secure (indistinguishable against keyword guessing attack).

5.2.1 *Security Model for IND-AP-SCF-CKCA Security*

In the following, we adopt a common security model IND-AP-SCF-CKCA security for chosen ciphertext attack and chosen keyword attack, in which the test query were added in IND-AP-SCF-CKA [21]. It guarantees that the server without the trapdoors for given keywords cannot distinguish which ciphertext encrypts which keyword, and the outside attacker without server's private key cannot make any decisions about the ciphertexts, even if the attacker obtains all the trapdoors of the keywords. The attack models for these two types of attackers are described as $Game_{Server}$ and $Game_{Receiver}$ respectively.

For the game $Game_{Server}$, $\mathcal{A}$ is assumed to be a server. For the game $Game_{Receiver}$, $\mathcal{A}$ is assumed to be an outside attacker including the receiver. If there is no PPT adversary

$\mathcal{A}$ can win the games $Game_i$ below with non-negligible advantage, where i is server or receiver, we say our scheme meets IND-AP-SCF-CKCA security. If not specified, it is applicable to both the game $Game_{Server}$ and $Game_{Receiver}$. In the games, $\mathcal{B}$ is challenger, k is security parameter. We consider the following games.

- ***Setup***: $GlobalSetup$ and $KeyGen_{Ser}$ algorithms are executed. The global parameter *GP* and key pair (pk_s, sk_s) are generated. For the game $Game_{Server}$, $\mathcal{B}$ sends (GP, pk_s, sk_s) to $\mathcal{A}$. For the game $Game_{Receiver}$, $\mathcal{B}$ sends (GP, pk_s) to $\mathcal{A}$.
- ***Query phase 1***: $\mathcal{A}$ can adaptively make the following queries.
  1) ***Public key queries***: $\mathcal{A}$ queries public key for any R_i or R_j, $\mathcal{B}$ runs algorithm $KeyGen_R$ to generate key pair (pk_i, sk_i) or (pk_j, sk_j). Then, $\mathcal{B}$ returns public key pk_i or pk_j to $\mathcal{A}$.
  2) ***Private key queries***: $\mathcal{A}$ queries private key for any R_i or R_j, For the game $Game_{Receiver}$, $\mathcal{B}$ returns private key sk_i or sk_j to $\mathcal{A}$. For the game $Game_{Server}$, $\mathcal{A}$ is assumed to be a server that $\mathcal{A}$ will not collude with users without access right, so $\mathcal{A}$ cannot makes this query.
  3) ***Trapdoor queries for delegator***: $\mathcal{A}$ queries any keyword set Q of his choice to generate trapdoor for R_i. $\mathcal{B}$ runs algorithm $Trapdoor_{R_i}$ and responds the trapdoor $T_{i,Q}$ to $\mathcal{A}$.
  4) ***Test queries for delegator***: $\mathcal{A}$ queries any keyword set Q and any *PECK* ciphertext C_i of his choice for the test query. $\mathcal{B}$ first makes a trapdoor query on Q to get trapdoor $T_{i,Q}$, then runs algorithm $Test$ and responds the result to $\mathcal{A}$.
  If $\mathcal{A}$ queries delegation path for any ciphertext, runs the queries 5) to 9).
  5) ***Delegation path queries***: $\mathcal{A}$ queries delegation path Pa_i for any *PECK* ciphertext C_i of R_i. If it's the first time to query delegation path for the ciphertext C_i, $\mathcal{B}$ runs algorithms $CreatPath$ and $ReKeyGen$ to create delegation path Pa_i and generate re-encryption key for Pa_i, then responds "Pa_i is created." to $\mathcal{A}$. Otherwise, $\mathcal{B}$ outputs '$\perp$' to indicate that the query is invalid.
  6) ***Re-encryption key queries***: $\mathcal{A}$ queries re-encryption key $rk_{j-1\to j}$ for the delegatee R_j in Pa_i. $\mathcal{B}$ first checks whether the path Pa_i is created, and whether $(pk_{j-1}, pk_j) \in Pa_i$. If not, $\mathcal{B}$ outputs '$\perp$' to indicate that the query is invalid. Otherwise, $\mathcal{B}$ responds re-encryption key $rk_{j-1\to j}$ to $\mathcal{A}$.
  7) ***Re-encryption queries***: $\mathcal{A}$ queries ciphertext C_{j-1} to generate re-encrypted ciphertext C_j in the path Pa_i. $\mathcal{B}$ first checks whether Pa_i is created, and whether $(pk_{j-1}, pk_j) \in Pa_i$. If not, $\mathcal{B}$ outputs '$\perp$' to indicate that the query is invalid. Otherwise, $\mathcal{B}$ runs algorithm $RePECK$ to generate re-encrypted ciphertext C_j, then returns C_j to $\mathcal{A}$.
  8) ***Trapdoor queries for delegatee***: $\mathcal{A}$ queries any keyword set Q to generate trapdoor for delegatee R_j. $\mathcal{B}$ runs algorithm $Trapdoor_{R_j}$ and responds the trapdoor $T_{j,Q}$ to $\mathcal{A}$.
  9) ***Test queries for delegatee***: $\mathcal{A}$ queries any keyword set Q and any *RePECK* ciphertext C_j for the test query. $\mathcal{B}$ first makes a trapdoor query on Q to get trapdoor $T_{j,Q}$, then runs algorithm $Test_{Re}$ and responds the result to $\mathcal{A}$.
- ***Challenge***: Once $\mathcal{A}$ finishes the above queries, it outputs two keyword sets Q_0^* and Q_1^* of equal length, a delegator $R_{i_1}^*$ and it's delegation path $Pa_{i_1}^*$, a user $R_{i_2}^*$. (Notice that for the game $Game_{Server}$, neither Q_0^* nor Q_1^* has been queried to obtain the corresponding trapdoor in *Query phase 1*). After receiving them, $\mathcal{B}$ chooses a random $\gamma \in \{0, 1\}$, creates target ciphertext set $C^* = (C_{i_1}^*, C_{i_2}^*)$ for Q_γ^*, where $C_{i_1}^*$ and $C_{i_2}^*$ are the ciphertext of $R_{i_1}^*$ and $R_{i_2}^*$ respectively. then returns them to $\mathcal{A}$.
- ***Query phase 2***: $\mathcal{A}$ can ask the same types of queries as in *Query phase 1*, except for the following.
  1) ***Trapdoor queries***: For the game $Game_{Server}$, the queries of $Trapdoor_{R_i}$ and $Trapdoor_{R_j}$ are not allowed if the generated trapdoor is distinguishable for Q_0^* and Q_1^*.
  2) ***Test queries***: The queries of $Test$ and $Test_{Re}$ are not allowed if the query result is distinguishable for (C^*, Q_0^*) and (C^*, Q_1^*).
- ***Guess***: $\mathcal{A}$ outputs the guess $\gamma' \in \{0, 1\}$. We say that $\mathcal{A}$ wins the game if $\gamma' = \gamma$.

The advantage of $\mathcal{A}$ in $Game_i$ is defined as $Adv_{\mathcal{A}}^{Game_i}(k) = \left|\mathcal{P}r\left[\gamma' = \gamma\right] - 1/2\right|$.

The AP-SCF-PECKS scheme is said to be IND-AP-SCF-CKCA secure if $Adv_{\mathcal{A}}^{Game_i}(k)$ is negligible, where i is server or receiver.

5.2.2 Security Model for IND-KGA Security

In the following, we define the notion of IND-KGA security for AP-SCF-PECKS scheme. Specifically, IND-KGA guarantees that an outside adversary (neither server nor receiver), that has obtained the trapdoor for challenge keyword set cannot observe the relationship between the trapdoor and any keyword set.

If there is no PPT outside adversary $\mathcal{A}$ (neither server nor receiver) can win the game below with non-negligible advantage, we say the AP-SCF-PECKS scheme meets IND-KGA security. In the game, $\mathcal{B}$ is challenger, k is security parameter. We consider the following game.

- ***Setup***: $GlobalSetup$ and $KeyGen_{Ser}$ algorithms are executed. The global parameter *GP* and key pair (pk_s, sk_s) are generated. Then, $\mathcal{B}$ sends (GP, pk_s) to $\mathcal{A}$.
- ***Query phase 1***: $\mathcal{A}$ can adaptively make the following queries.
  1) ***Public key queries***: $\mathcal{A}$ queries public key for any R_i or R_j, $\mathcal{B}$ runs algorithm $KeyGen_R$ to generate key pair (pk_i, sk_i) or (pk_j, sk_j). Then, $\mathcal{B}$ returns public key pk_i or pk_j to $\mathcal{A}$.
  2) ***Trapdoor queries for delegator***: $\mathcal{A}$ queries any keyword set Q of his choice to generate trapdoor for R_i. $\mathcal{B}$ runs algorithm $Trapdoor_{R_i}$ and responds the trapdoor $T_{i,Q}$ to $\mathcal{A}$.
  If $\mathcal{A}$ queries delegation path for any ciphertext, runs the queries 3).

3) ***Trapdoor queries for delegatee***: $\mathcal{A}$ queries any keyword set Q to generate trapdoor for the delegatee R_j. $\mathcal{B}$ runs algorithm $Trapdoor_{R_j}$ and responds the trapdoor $T_{j,Q}$ to $\mathcal{A}$.

- ***Challenge***: Once $\mathcal{A}$ finishes the above queries, it outputs two keyword sets Q_0^* and Q_1^* of equal length, a delegator $R_{i_1}^*$ and it's delegation path $Pa_{i_1}^*$, a user $R_{i_2}^*$. (Notice that neither Q_0^* nor Q_1^* has been queried to obtain the corresponding trapdoor in *Query phase 1*). After receiving them, $\mathcal{B}$ randomly chooses $\gamma \in \{0,1\}$, creates target trapdoor set $T_{i,Q_\gamma^*}^* = (T_{i_1,Q_\gamma^*}^*, T_{i_2,Q_\gamma^*}^*)$ for $R_{i_1}^*$ and $R_{i_2}^*$, then returns the target trapdoor set $T_{i,Q_\gamma^*}^*$ to $\mathcal{A}$.
- ***Query phase 2***: $\mathcal{A}$ can ask the same types of queries as in *Query phase 1*, except for the following.
  1) ***Trapdoor queries***: The queries of $Trapdoor_{R_i}$ and $Trapdoor_{R_j}$ are not allowed if the generated trapdoor is distinguishable for Q_0^* and Q_1^*.
- ***Guess***: $\mathcal{A}$ outputs the guess $\gamma' \in \{0,1\}$. We say that $\mathcal{A}$ wins the game if $\gamma' = \gamma$.

The advantage of $\mathcal{A}$ in IND-KGA game is defined as $Adv_{\mathcal{A}}^{IND-KGA}(k) = \left|\mathcal{P}r\left[\gamma' = \gamma\right] - 1/2\right|$.

The AP-SCF-PECKS scheme is said to be IND-KGA secure if $Adv_{\mathcal{A}}^{IND-KGA}(k)$ is negligible.

6 Construction of AP-SCF-PECKS and Its Application in Healthcare Cloud

In this section, we describe the construction of AP-SCF-PECKS in detail and prove its correctness, then apply AP-SCF-PECKS to healthcare cloud.

6.1 Construction of AP-SCF-PECKS

In the system, the files are encrypted by a symmetric encryption algorithm, and symmetric key is encapsulated with the user's public key by key encapsulation mechanism. Keywords extracted from the file are encrypted by the AP-SCF-PECKS scheme. The scheme in the following focus on searchable keywords encryption and autonomous path delegation function.

- $GlobalSetup(k)$: Let k be the security parameter, the algorithm randomly chooses a generator g of group G_1 and a generator $\tilde{g}$ of group G_2, sets a bilinear map $e: G_1 \times G_2 \to G_T$, a hash function $H: \{0,1\}^* \to Z_p^*$, and a cryptographic hash function $H': G_T \to G_1$. (By Z_p^* and G_2^*, we denote $Z_p^*: Z_p \backslash \{0\}$ where 0 is identity element of Z_p, and $G_2^*: G_2 \backslash \{1\}$ where 1 is identity element of G_2.) Finally, the global parameter is $GP = \{p, G_1, G_2, G_T, e, g, \tilde{g}, H, H'\}$.
- $KeyGen_{Ser}(GP)$: This algorithm randomly chooses $\tilde{V} \in G_2^*$ and $x \in Z_p^*$, computes $X = g^x$ and outputs data server's public and private key pair $(pk_s, sk_s) = ((X, \tilde{V}), x)$.
- $KeyGen_R(GP)$: This algorithm randomly chooses value $y_i \in Z_p^*$, computes $Y_i = \tilde{g}^{y_i}$, and outputs public and private key pair $(pk_i, sk_i) = (Y_i, y_i)$ for user R_i.
- $PECK(GP, pk_s, pk_i, W)$: This algorithm selects a keyword set $W = (w_1, w_2, \cdots, w_n)$ for the user R_i's outsourcing file, and randomly chooses value $\tau \in Z_p^*$. An (n+1)-degree polynomial $f(x) = \eta_{n+1}x^{n+1} + \eta_n x^n + \cdots + \eta_1 x + \eta_0$ should be constructed to make $H(w_1), H(w_2), \cdots, H(w_n)$ and τ to be (n+1) roots of the equation $f(x) = 1$. Then the algorithm randomly chooses $s, r \in Z_p^*$, computes $t = e(X, \tilde{V})^s$, $C_{i,1} = g^s, C_{i,2} = t \cdot e(X, Y_i)^r, C_{i,3} = \tilde{g}^r, B_\varphi = C_{i,3}^{\eta_\varphi}$ for $0 \le \varphi \le (n+1)$. Then it outputs ciphertext $C_i = (C_{i,1}, C_{i,2}, C_{i,3}, B_\varphi)$ where $0 \le \varphi \le (n+1)$.
- $Trapdoor_{R_i}(GP, pk_s, sk_i, Q)$: The user R_i randomly chooses $T_{i,-1}, \zeta \in Z_p^*$, computes $T_{i,-2} = g^\zeta$, $T_{i,\varphi} = g^{m^{-1} \cdot T_{i,-1} \cdot y_i \cdot \sum_{\mu=1}^m H(q_\mu)^\varphi} \cdot X^\zeta$ for $0 \le \varphi \le (n+1)$, $Q = (q_1, q_2, \cdots, q_m)$, $m \le n$. Then, the algorithm outputs trapdoor $T_{i,Q} = (T_{i,-1}, T_{i,-2}, T_{i,\varphi})$ where $0 \le \varphi \le (n+1)$.
- $Test(GP, pk_s, sk_s, T_{i,Q}, C_i)$: After receiving trapdoor $T_{i,Q}$ and ciphertext C_i, the server first computes $t = e(C_{i,1}, \tilde{V})^x$, then checks whether the following equation holds

  $$t^{T_{i,-1}} \cdot \prod_{\varphi=0}^{n+1} e(T_{i,\varphi}/T_{i,-2}^x, B_\varphi)^x = C_{i,2}^{T_{i,-1}}$$

  If the equation holds, it outputs 1. Otherwise, it outputs 0.

If R_i sets a delegation path, the following operations are executed.

- $CreatPath(GP, R_i, R_j, j = 1, \cdots, l_i)$: The delegator R_i outputs a delegation path $Pa_i = (R_0 = R_i, R_1, R_2, \cdots, R_{l_i})$ which contains l_i delegatees.

The delegator R_i sends a delegation notice $\langle R_i, Pa_i, Sig_{sk_i}(R_i, Pa_i)\rangle$ to TTP, where $Sig_{sk_i}(R_i, Pa_i)$ is the signature of the information (R_i, Pa_i) generated by R_i. Sig is a cryptographic secure signature algorithm, which is not specified in this scheme. After the TTP receive the delegation notice $\langle R_i, Pa_i, Sig_{sk_i}(R_i, Pa_i)\rangle$, signature $Sig_{sk_i}(R_i, Pa_i)$ can be verified by using the public key of R_i. If the signature verification fails, delegation request will be rejected. Otherwise, the following delegation will be executed.

- $ReKeyGen(GP, Pa_i, pk_s, sk_i, sk_j, j = 1, \cdots, l_i)$: After receiving the delegation path Pa_i, this algorithm randomly select $s_j \in Z_p^*$, $K_j \in G_T$ for delegatees R_j where $j = 1, \cdots, l_i$, computes $rk_{j-1\to j} = (rk_{j-1\to j}^1, rk_{j-1\to j}^2, rk_{j-1\to j}^3), j = 1, \cdots, l_i$ in which
  $rk_{j-1\to j}^1 = g^{s_j}$,
  $rk_{j-1\to j}^2 = H'(K_1 \cdot K_2 \cdots K_j) \cdot X^{y_i - y_j}$, and
  $rk_{j-1\to j}^3 = \begin{cases} K_j \cdot e(X, \tilde{V})^{s_j}, j = 1 \\ K_j \cdot e(X, \tilde{V})^{s_j - s_{j-1}}, j > 1 \end{cases}$.
  Then, the algorithm sends re-encryption key $rk_{j-1\to j}, j = 1, \cdots, l_i$ to proxy server.
- $RePECK(GP, Pa_i, rk_{j-1\to j}, C_{j-1}, 1 \le j \le l_i)$: To re-encrypt a ciphertext under the public key pk_{j-1} to the one under pk_j in the path Pa_i, this algorithm first checks whether $(pk_{j-1}, pk_j) \in Pa_i$ and outputs '$\perp$' if not. Otherwise, it computes
  $C_{j,1} = \begin{cases} C_{i,1}, j = 1 \\ C_{j-1,1}, j > 1 \end{cases}$, $C_{j,2} = \begin{cases} C_{i,2}, j = 1 \\ C_{j-1,2}, j > 1 \end{cases}$,
  $C_{j,3} = \begin{cases} C_{i,3}, j = 1 \\ C_{j-1,3}, j > 1 \end{cases}$, $C_{j,4} = rk_{j-1\to j}^1, C_{j,5} =$

$e(rk^2_{j-1\to j}, C_{j,3})$, and
$C_{j,6} = \begin{cases} rk^3_{j-1\to j}, j = 1 \\ C_{j-1,6} \cdot rk^3_{j-1\to j}, j > 1 \end{cases}$.
Then, the algorithm outputs re-encrypted ciphertext $C_j = (C_{j,1}, C_{j,2}, C_{j,3}, C_{j,4}, C_{j,5}, C_{j,6}, B_\varphi)$, $0 \le \varphi \le (n+1)$ for the delegatee R_j.

- $Trapdoor_{R_j}(GP, pk_s, sk_j, Q)$: The delegatee R_j randomly chooses value $T_{j,-1}, \zeta \in Z_p^*$, computes $T_{j,-2} = g^\zeta$, $T_{j,\varphi} = g^{m^{-1} \cdot T_{j,-1} \cdot y_j \cdot \sum_{\mu=1}^{m} H(q_\mu)^\varphi} \cdot X^\zeta$ for $0 \le \varphi \le (n+1)$, $Q = (q_1, q_2, \cdots, q_m)$, $m \le n$. Then, the algorithm outputs trapdoor $T_{j,Q} = (T_{j,-1}, T_{j,-2}, T_{j,\varphi})$ where $0 \le \varphi \le (n+1)$.
- $Test_{Re}(GP, pk_s, sk_s, T_{j,Q}, C_j)$: After receiving trapdoor $T_{j,Q}$ and ciphertext C_j, the server first computes $t = e(C_{j,1}, \tilde{V})^x$ and $K = \frac{C_{j,6}}{e(C_{j,4}, \tilde{V})^x}$, then checks whether the following equation holds

$$[t \cdot C_{j,5}]^{T_{j,-1}} \cdot \prod_{\varphi=0}^{n+1} e\left(T_{j,\varphi}/T_{j,-2}^x, B_\varphi\right)^x = [C_{j,2} \cdot e\left(H'(K), C_{j,3}\right)]^{T_{j,-1}}$$

If the equation holds, it outputs 1. Otherwise, it outputs 0.

Correctness:

1) $Test(GP, pk_s, sk_s, T_{i,Q}, C_i)$: The server first computes $t = e(C_{i,1}, \tilde{V})^x = e(g, \tilde{V})^{sx} = e(X, \tilde{V})^s$, then checks the following equation

$$t^{T_{i,-1}} \cdot \prod_{\varphi=0}^{n+1} e(T_{i,\varphi}/T_{i,-2}^x, B_\varphi)^x = C_{i,2}^{T_{i,-1}}$$

where

$$\begin{aligned} & t^{T_{i,-1}} \cdot \prod_{\varphi=0}^{n+1} e(T_{i,\varphi}/T_{i,-2}^x, B_\varphi)^x \\ = \; & t^{T_{i,-1}} \cdot \prod_{\varphi=0}^{n+1} e\left(g^{m^{-1} \cdot T_{i,-1} \cdot y_i \cdot \sum_{\mu=1}^{m} H(q_\mu)^\varphi}, \tilde{g}^{r \cdot \eta_\varphi}\right)^x \\ = \; & t^{T_{i,-1}} \cdot e(g, \tilde{g})^{x \cdot r \cdot m^{-1} \cdot T_{i,-1} \cdot y_i \cdot \sum_{\varphi=0}^{n+1} [\eta_\varphi \cdot \sum_{\mu=1}^{m} H(q_\mu)^\varphi]} \\ = \; & t^{T_{i,-1}} \cdot e(g, \tilde{g})^{x \cdot r \cdot T_{i,-1} \cdot y_i} \\ = \; & C_{i,2}^{T_{i,-1}} \end{aligned}$$

the equation holds, it outputs 1.

2) $Test_{Re}(GP, pk_s, sk_s, T_{j,Q}, C_j)$: The server first computes $t = e(C_{j,1}, \tilde{V})^x = e(g, \tilde{V})^{sx} = e(X, \tilde{V})^s$ and K. when $j = 1$,

$$K = \frac{C_{j,6}}{e(C_{j,4}, \tilde{V})^x} = \frac{rk^3_{j-1\to j}}{e(rk^1_{j-1\to j}, \tilde{V})^x} = \frac{K_j \cdot e(X, \tilde{V})^{s_j}}{e(g^{s_j}, \tilde{V})^x} = K_j = K_1$$

and when $j > 1$,

$$\begin{aligned} K \; = \; & \frac{C_{j,6}}{e(C_{j,4}, \tilde{V})^x} \\ = \; & \frac{C_{j-1,6} \cdot rk^3_{j-1\to j}}{e(rk^1_{j-1\to j}, \tilde{V})^x} \\ = \; & \frac{[K_1 \cdot e(X, \tilde{V})^{s_1}] \cdot [K_2 \cdot e(X, \tilde{V})^{s_2 - s_1}] \cdots [K_j \cdot e(X, \tilde{V})^{s_j - s_{j-1}}]}{e(g^{s_j}, \tilde{V})^x} \\ = \; & \frac{K_1 \cdot K_2 \cdots K_j \cdot e(X, \tilde{V})^{s_j}}{e(X, \tilde{V})^{s_j}} \\ = \; & K_1 \cdot K_2 \cdots K_j \end{aligned}$$

it means that $K = K_1 \cdot K_2 \cdots K_j, j \ge 1$.
Then checks the following equation

$$[t \cdot C_{j,5}]^{T_{j,-1}} \cdot \prod_{\varphi=0}^{n+1} e\left(T_{j,\varphi}/T_{j,-2}^x, B_\varphi\right)^x = [C_{j,2} \cdot e\left(H'(K), C_{j,3}\right)]^{T_{j,-1}}$$

where

$$\begin{aligned} & [t \cdot C_{j,5}]^{T_{j,-1}} \cdot \prod_{\varphi=0}^{n+1} e\left(T_{j,\varphi}/T_{j,-2}^x, B_\varphi\right)^x \\ = \; & t^{T_{j,-1}} \cdot e(rk^2_{j-1\to j}, C_{j,3})^{T_{j,-1}} \cdot e(g, \tilde{g})^{x \cdot r \cdot T_{j,-1} \cdot y_j} \\ = \; & t^{T_{j,-1}} \cdot e(H'(K_1 \cdot K_2 \cdots K_j), \tilde{g})^{r \cdot T_{j,-1}} \cdot e(g, \tilde{g})^{x \cdot r \cdot T_{j,-1} \cdot y_i} \end{aligned}$$

and

$$\begin{aligned} & [C_{j,2} \cdot e\left(H'(K), C_{j,3}\right)]^{T_{j,-1}} \\ = \; & t^{T_{j,-1}} \cdot e(X, Y_i)^{r \cdot T_{j,-1}} \cdot e(H'(K), \tilde{g})^{r \cdot T_{j,-1}} \\ = \; & t^{T_{j,-1}} \cdot e(g, \tilde{g})^{x \cdot r \cdot T_{j,-1} \cdot y_i} \cdot e(H'(K_1 \cdot K_2 \cdots K_j), \tilde{g})^{r \cdot T_{j,-1}} \end{aligned}$$

apparently, the equation holds, it outputs 1.

The above 1) and 2) show that our AP-SCF-PECKS scheme is correct.

6.2 Revocation

In this subsection, we provide revocation services for the AP-SCF-PECKS scheme in three different situations: user revocation, authorized user revocation in the path, and path revocation.

User revocation: When TTP generates key pair for users, it will display the users' public key in its' public bulletin board for use by other entities, and save the users' private key. When TTP revokes a user, it can remove the revoked user's public key from it's public bulletin board, and remove the revoked user's private key. In this way, the data provider cannot provide encryption for the revoked user. Other users cannot perform path delegation for the revoked user, and the revoked user also cannot execute path delegation for himself (Due to TTP cannot generate proxy re-encryption key for the revoked user). At the same time, all existing original ciphertext, proxy re-encryption ciphertext, proxy re-encryption key about the revoked user will be deleted. If the revoked user in a delegation path, the proxy re-encryption key about the next authorized user of the revoked user in the path will be updated (same as "*Authorized user revocation in the path*"). In this way, the revoked user cannot search for and match ciphertext, even if he can generate trapdoor with his private key.

Authorized user revocation in the path: A user R_i generates a delegation path $Pa_i = (R_0 = R_i, R_1, \cdots, R_k, \cdots, R_{l_i})$. When R_i revokes an authorized user R_k in the path Pa_i. First, R_i updates the path as $Pa_i = (R_0 = R_i, R_1, \cdots, R_{k-1}, R_{k+1} \cdots, R_{l_i})$. Second, if necessary (the re-encryption ciphertext C_k for the revoked user R_k is generated), the proxy sever re-encrypts C_k by using the re-encryption key $rk_{k\to k+1}$, generates the re-encryption ciphertext C_{k+1} for the next authorized user R_{k+1}, where $C_{k+1} = RePECK_{rk_{k\to k+1}}(C_k)$, and sends C_{k+1} to the data center, then the data center updates C_k to C_{k+1} . Finally, TTP updates the proxy re-encryption key $rk_{k-1\to k+1}$ for the next authorized user R_{k+1} in the path Pa_i, where
$rk_{k-1\to k+1} = (rk^1_{k-1\to k+1}, rk^2_{k-1\to k+1}, rk^3_{k-1\to k+1})$,
$rk^1_{k-1\to k+1} = rk^1_{k\to k+1}$, $rk^2_{k-1\to k+1} = rk^2_{k\to k+1}$,
$rk^3_{k-1\to k+1} = rk^3_{k-1\to k} \cdot rk^3_{k\to k+1}$,
and sends $rk_{k-1\to k+1}$ to the proxy server, then proxy sever delete the proxy re-encryption key $rk_{k-1\to k}$, $rk_{k\to k+1}$. The proxy re-encryption key of other unrevoked users (except R_{k+1}) in the path remains the same.

Path revocation: A user R_i generates a delegation path Pa_i, he can set a deadline t_i for the path. When the proxy

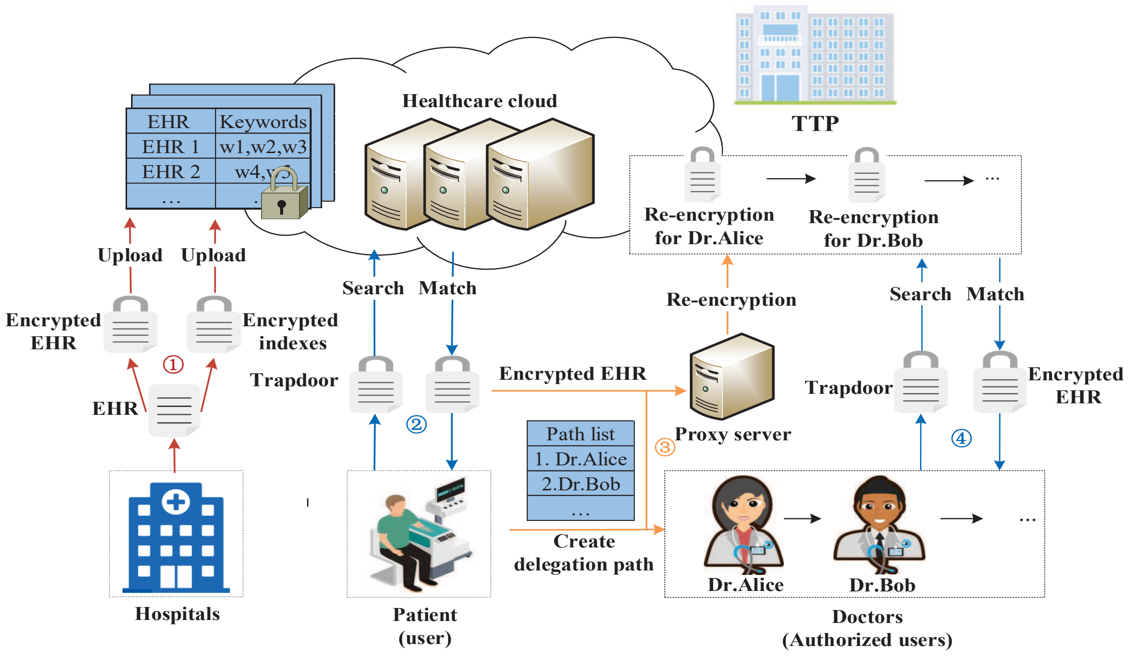

Fig. 2: Application Model for AP-SCF-PECKS

server generates proxy re-encryption ciphertext for authorized users in the path, and sends the proxy re-encryption ciphertext to the data center together with the deadline t_i. The data center compares the current time with the deadline t_i of the proxy re-encryption ciphertext, and deletes the expired proxy re-encryption ciphertext. In this way, the path beyond the deadline is invalid. If the user R_i revokes the path Pa_i, the proxy re-encryption key and the generated proxy re-encryption ciphertext corresponding to the path Pa_i will be deleted, all users in the revocation path Pa_i will not be able to search and match the corresponding proxy re-encryption ciphertext.

It can be ovserved that, the revocation mechanism of the proposed scheme does not introduce any significant additional overhead.

6.3 Application of AP-SCF-PECKS in Healthcare Cloud

In this subsection, we apply the AP-SCF-PECKS scheme to healthcare cloud. As shown in Fig. 2, there are five types of entities: TTP (trusted third party), hospital (data provider), patient/doctor (user/authorized user), healthcare cloud (third party data center) and proxy server. TTP generates keys for patient, doctor, healthcare cloud, and generates re-encryption keys according to delegation path. When hospitals, patients and doctors join in the system, they first need authorization to ensure their legitimacy [27], [28]. Then, hospitals outsource patient's encrypted EHRs to healthcare cloud, patient can search his EHRs over the healthcare cloud. When the patient want to make appointment with some doctors, he can act as a delegator to sets a multi-hop delegation path in advance according to his preference, so that doctors in the path have search and access rights of EHRs with priority from high to low.

Step 1: Hospitals store encrypted EHRs on the healthcare cloud. When the hospital uploads EHR, the hospital extracts keywords from the EHR, such as the patient's identity, hospital, department or disease, type of examination report, etc. Then the hospital encrypts keywords into searchable encrypted indexes by using patient's public key with the AP-SCF-PECKS scheme, encrypts the EHR into ciphertext by using a symmetric encryption algorithm, and uploads the encrypted indexes and corresponding ciphertext to healthcare cloud.

Step 2: The patient wants to obtain his EHRs from the healthcare cloud. He first generates trapdoor using query keywords and his private key, then sends the trapdoor to healthcare cloud for search. After receiving search request, healthcare cloud returns the matching ciphertexts to the patient. If the query keyword is the patient's identity, the patient can search and obtain all his EHRs. If the query keywords are the patient's identity and X-ray examination, the patient can search and obtain his X-ray examination, no matter in which hospital.

Step 3: The patient wants some doctors search and access his EHRs with priority from high to low. He can obtain his encrypted EHRs like *Step 2*, and send them to the proxy. Then he act as a delegator to create a multi-hop delegation path to delegate these EHRs. The delegation path such as "1.Dr.Alice, 2.Dr.Bob, . . . ". It indicates that the access right is delegated to Dr.Alice firstly, if Dr.Alice is unable to search and query, the access right will be transferred to Dr.Bob, and so on. The delegation will be terminated when there is a search query by a doctor in the path, or there is no search query after traversing the path. At the same time, the proxy server will re-encrypt ciphertexts using re-encryption keys.

Step 4: The authorized doctor wants to search and access EHRs of the patient. He generates trapdoor using query keywords and his private key, then sends the trapdoor to healthcare cloud for search. After receiving search request, healthcare cloud returns the matching ciphertexts to the authorized doctor. Authorized doctor are similar to the patient, according to the different keywords of query, authorized doctor can obtain different range of medical records of the patient.

For the convenience of description, the above described the situation of setting a delegation path for a patient. However, the scheme can be applied to multiple users and multiple paths scenarios.

In practical application, each patient can be assigned a personal cloud space in healthcare cloud, and the EHRs of a patient can be centrally stored in his personal cloud space. When the patient wants to obtain his EHRs from the healthcare cloud, he can access his personal medical account and search EHRs from his personal cloud space. In addition, the data provider also can be the patient himself.

7 SECURITY ANALYSIS

We give security proofs for AP-SCF-PECKS scheme in this section. In general, the solution should meets the following security requirements.

- ***Confidentiality***: The confidentiality of AP-SCF-PECKS scheme means that user's sensitive information in EHR must be kept secret from unauthorized users and cloud data center. The EHR file is protected by a strong symmetric encryption algorithm, and keywords of the EHR are encrypted by the $PECK$ and $RePECK$ algorithms. Only the data server can execute the $Test$ and $Test_{Re}$ algorithms to match the ciphertext by using it's private key, but it cannot decrypt the matching ciphertext, nor can it obtain any information about the keywords and secret keys of user.
- ***Strictly following delegation path***: If a user R_i creates a delegation path $Pa_i = (R_0 = R_i, R_1, R_2, \cdots, R_{l_i})$, the access right of the ciphertext will be delegated strictly following the path, and cannot be converted to other paths through meaningful decryption operations. It means that the re-encrypted ciphertext C_j for delegatee R_j in the path Pa_i ($j \neq i$) cannot be converted and inserted into another path Pa' generated by R_j or R_i. Similarly, the original ciphertext C_i generated by R_i also cannot be converted and inserted into other paths.

7.1 Security Proof for IND-AP-SCF-CKCA Security

Theorem 1: Assuming that n-ABDHE and DBDH problem are intractable, the AP-SCF-PECKS scheme is IND-AP-SCF-CKCA security in standard model. If there is a PPT adversary $\mathcal{A}$, who breaks $\varepsilon-$IND-AP-SCF-CKCA security of AP-SCF-PECKS scheme, then we can construct a PPT adversary $\mathcal{B}$ using $\mathcal{A}$ to break ε_1 n-ABDHE problem and ε_2 DBDH problem with $\varepsilon_1 \geq \varepsilon$, $\varepsilon_2 \geq \varepsilon$.

Proof: The proof of *theorem 1* is as follows.

$Game_{Server}$: $\mathcal{A}$ is assumed to be a server.

We first let the challenger set the groups G_1, G_2 and G_T with an efficient bilinear map e, a generator g of G_1 and a generator $\tilde{g}$ of G_2. Suppose that $\mathcal{B}$ accepts a properly distributed n-ABDHE instance $(g, g^a, \cdots g^{a^n}, \tilde{g}, \tilde{g}^a, \cdots \tilde{g}^{a^n}, \tilde{g}^z, \tilde{g}^{za^{n+2}}, T)$ and has to distinguish $T = e(g, \tilde{g})^{za^{n+1}}$ from a random element in G_T.

- *Setup*: $\mathcal{B}$ generates and outputs the scheme's global parameters $GP = \{p, G_1, G_2, G_T, e, g, \tilde{g}, H, H'\}$, where $H : \{0,1\}^* \rightarrow Z_p^*$ is a hash function, $H' : G_T \rightarrow G_1$ is a cryptographic hash function. $\mathcal{B}$ randomly chooses $x \in Z_p^*$, $\tilde{V} \in G_2^*$, computes $X = g^x$, let $(pk_s, sk_s) = ((X, \tilde{V}), x)$ be the data server's public and private key pair. Then, $\mathcal{B}$ sends (GP, pk_s, sk_s) to $\mathcal{A}$.
- ***Query phase 1***: $\mathcal{A}$ makes the following queries.
  1) ***Public key queries***: $\mathcal{A}$ adaptively queries a public key for R_i or R_j. $\mathcal{B}$ randomly chooses $y_i \in Z_p^*$, and defines the public key $Y_i = (\tilde{g}^a)^{y_i} = \tilde{g}^{ay_i}$, which means the private key of R_i is ay_i that is unknown to $\mathcal{B}$. The key pair of R_i is $(pk_i, sk_i) = (Y_i, ay_i)$. The key generation of R_j is similar to that of R_i. Then, $\mathcal{B}$ sends pk_i or pk_j to $\mathcal{A}$.
  2) ***Trapdoor queries for delegator***: $\mathcal{A}$ adaptively queries a keyword set $Q = (q_1, q_2, \cdots, q_m), m \leq n$ to obtain a trapdoor for R_i. $\mathcal{B}$ randomly chooses $T_{i,-1}, \zeta \in Z_p^*$, computes $T_{i,-2} = g^{\zeta}$,
  $$T_{i,\varphi} = (g^a)^{y_i \cdot m^{-1} \cdot T_{i,-1} \cdot \sum_{\mu=1}^{m} H(q_\mu)^{\varphi}} \cdot X^{\zeta} = g^{m^{-1} \cdot T_{i,-1} \cdot (ay_i) \cdot \sum_{\mu=1}^{m} H(q_\mu)^{\varphi}} \cdot X^{\zeta},$$
  where $0 \leq \varphi \leq (n+1)$. $\mathcal{B}$ has successfully simulated the trapdoor $T_{i,Q} = (T_{i,-1}, T_{i,-2}, T_{i,\varphi})$, $0 \leq \varphi \leq (n+1)$ for R_i, then $\mathcal{B}$ responds $T_{i,Q}$ to $\mathcal{A}$.
  3) ***Test queries for delegator***: $\mathcal{A}$ adaptively queries any keyword set Q and any *PECK* ciphertext C_i for the test query. $\mathcal{B}$ first makes a trapdoor query on Q to get trapdoor $T_{i,Q}$, then runs algorithm $Test$ and responds the result to $\mathcal{A}$.
  If $\mathcal{A}$ queries delegation path for any ciphertext, runs the queries 4) to 8).
  4) ***Delegation path queries***: $\mathcal{A}$ adaptively queries delegation path for a *PECK* ciphertext C_i of R_i.
  If it's the first time to query delegation path for the ciphertext C_i, $\mathcal{B}$ runs algorithm $CreatPath$ to create path $Pa_i = (R_i, R_1, R_2, \cdots, R_{l_i})$. Then, $\mathcal{B}$ selects $s_j \in Z_p^*$, $K_j \in G_T$ for $j = 1, \cdots, l_i$, and computes $rk_{j-1 \rightarrow j} = (rk_{j-1 \rightarrow j}^1, rk_{j-1 \rightarrow j}^2, rk_{j-1 \rightarrow j}^3)$, in which $rk_{j-1 \rightarrow j}^1 = g^{s_j}$, $rk_{j-1 \rightarrow j}^2 = H'(K_1 \cdot K_2 \cdots K_j) \cdot (g^a)^{(y_i - y_j) \cdot x} = H'(K_1 \cdot K_2 \cdots K_j) \cdot X^{ay_i - ay_j}$,
  $$rk_{j-1 \rightarrow j}^3 = \begin{cases} K_j \cdot e(X, \tilde{V})^{s_j}, j = 1 \\ K_j \cdot e(X, \tilde{V})^{s_j - s_{j-1}}, j > 1 \end{cases}.$$
  $\mathcal{B}$ has successfully simulated the re-encryption key, then responds "Pa_i is created." to $\mathcal{A}$.
  Otherwise, $\mathcal{B}$ outputs '$\perp$' to indicate an invalid query.
  5) ***Re-encryption key queries***: $\mathcal{A}$ adaptively queries re-encryption key $rk_{j-1 \rightarrow j}$ for the delegatee R_j in Pa_i. $\mathcal{B}$ first checks whether the path Pa_i is created, and whether $(pk_{j-1}, pk_j) \in Pa_i$. If not, $\mathcal{B}$ outputs '$\perp$' to indicate that the query is invalid. Otherwise, $\mathcal{B}$ returns re-encryption key $rk_{j-1 \rightarrow j}$ to $\mathcal{A}$.
  6) ***Re-encryption queries***: $\mathcal{A}$ adaptively queries ciphertext C_{j-1} to generate re-encrypted ciphertext C_j in the path Pa_i. $\mathcal{B}$ first checks whether the path Pa_i is created, and whether $(pk_{j-1}, pk_j) \in Pa_i$. If not, $\mathcal{B}$ outputs '$\perp$' to indicate that the query is invalid. Otherwise, $\mathcal{B}$ runs algorithm $RePECK$ to generate

re-encrypted ciphertext C_j as an honest user does, then responds the re-encrypted ciphertext C_j to $\mathcal{A}$.
7) ***Trapdoor queries for delegatee***: $\mathcal{A}$ adaptively queries a keyword set $Q = (q_1, q_2, \cdots, q_m), m \leq n$ to obtain a trapdoor for R_j. $\mathcal{B}$ randomly chooses $T_{j,-1}, \zeta \in Z_p^*$, computes $T_{j,-2} = g^{\zeta}$,

$$T_{j,\varphi} = (g^a)^{y_j \cdot m^{-1} \cdot T_{j,-1} \cdot \sum_{\mu=1}^{m} H(q_\mu)^\varphi} \cdot X^\zeta = g^{m^{-1} \cdot T_{j,-1} \cdot (ay_j) \cdot \sum_{\mu=1}^{m} H(q_\mu)^\varphi} \cdot X^\zeta,$$

where $0 \leq \varphi \leq (n+1)$. Then, $\mathcal{B}$ responds the trapdoor $T_{j,Q} = (T_{j,-1}, T_{j,-2}, T_{j,\varphi}), 0 \leq \varphi \leq (n+1)$ to $\mathcal{A}$.
8) ***Test queries for delegatee***: $\mathcal{A}$ adaptively queries any keyword set Q and any *RePECK* ciphertext C_j for the test query. $\mathcal{B}$ first makes a trapdoor query on Q to get trapdoor $T_{j,Q}$, then runs algorithm $Test_{Re}$ and responds the result to $\mathcal{A}$.

- ***Challenge***: Once $\mathcal{A}$ finishes the above queries, it outputs two keyword sets Q_0^* and Q_1^* of equal length, a delegator $R_{i_1}^*$ and it's delegation path $Pa_{i_1}^*$, a user $R_{i_2}^*$. Upon receiving them, $\mathcal{B}$ chooses a random $\gamma \in \{0,1\}$, let $Q_\gamma^* = (Q_{\gamma,1}^*, Q_{\gamma,2}^*, \cdots, Q_{\gamma,m}^*)$.
For $R_{i_1}^*$, let $F(x) = x^{n+1}$ and $\mathcal{R}(x) = (F(x) - F(\prod_{\phi=1}^{m} H(Q_{\gamma,\phi}^*)))/\ (x - \prod_{\phi=1}^{m} H(Q_{\gamma,\phi}^*))$, which is a polynomial of degree n. Then $\mathcal{B}$ randomly selects $s^*, s_j^*, \tau^* \in Z_p^*$, $K^* \in G_T$, computes $t^* = e(X, \tilde{V})^{s^*}$, and constructs a polynomial $f^*(x) = \eta_{n+1}^* x^{n+1} + \eta_n^* x^n + \cdots + \eta_1^* x + \eta_0^*$ such that $H(Q_{\gamma,1}^*), H(Q_{\gamma,2}^*), \cdots, H(Q_{\gamma,n}^*)$ and τ^* are $(n+1)$ roots of the equation $f^*(x) = 1$. Then $\mathcal{B}$ sets

$$C_{i,1}^* = g^{s^*},$$
$$C_{i,2}^* = t^* \cdot T^{x \cdot y_i} \cdot e(\textstyle\prod_{\phi=1}^{n} g^{\mathcal{R}_{\phi-1} \cdot a^\phi}, (\tilde{g}^z)^{x \cdot y_i}),$$
$$C_{i,3}^* = (\tilde{g}^z)^{\mathcal{R}(a)}, B_\varphi^* = (C_{i,3}^*)^{\eta_\varphi^*}, 0 \leq \varphi \leq (n+1).$$

and

$$C_{j,1}^* = C_{i,1}^* = g^{s^*},$$
$$C_{j,2}^* = C_{i,2}^* = t^* \cdot T^{x \cdot y_i} \cdot e(\textstyle\prod_{\phi=1}^{n} g^{\mathcal{R}_{\phi-1} \cdot a^\phi}, (\tilde{g}^z)^{x \cdot y_i}),$$
$$C_{j,3}^* = C_{i,3}^* = (\tilde{g}^z)^{\mathcal{R}(a)}, C_{j,4}^* = g^{s_j^*},$$
$$C_{j,5}^* = e\left(H'(K^*), (\tilde{g}^z)^{\mathcal{R}(a)}\right) \cdot e\left((g^a)^{(y_i - y_j) \cdot x}, (\tilde{g}^z)^{\mathcal{R}(a)}\right) = e\left(H'(K^*), \tilde{g}^{z\mathcal{R}(a)}\right) \cdot e\left(g^{(ay_i - ay_j) \cdot x}, \tilde{g}^{z\mathcal{R}(a)}\right),$$
$$C_{j,6}^* = K^* \cdot e(X, \tilde{V})^{s_j^*},$$

where $\mathcal{R}_\phi$ is the coefficient of x^ϕ in $\mathcal{R}(x)$. Let $r^* = z\mathcal{R}(a)$, if $T = e(g, \tilde{g})^{za^{n+1}}$, then

$$C_{j,2}^* = C_{i,2}^* = t^* \cdot e(g, \tilde{g})^{za^{n+1} \cdot x \cdot y_i} \cdot e(g, \tilde{g})^{x \cdot ay_i \cdot z \sum_{\phi=0}^{n-1} \mathcal{R}_\phi \cdot a^\phi} = t^* \cdot e(g, \tilde{g})^{x \cdot ay_i \cdot za^n} \cdot e(g, \tilde{g})^{x \cdot ay_i \cdot z \sum_{\phi=0}^{n-1} \mathcal{R}_\phi \cdot a^\phi} = t^* \cdot e(g, \tilde{g})^{x \cdot ay_i \cdot z \sum_{\phi=0}^{n} \mathcal{R}_\phi \cdot a^\phi} = t^* \cdot e(X, Y_i)^{r^*},$$
$$C_{j,3}^* = C_{i,3}^* = \tilde{g}^{r^*},$$
$$C_{j,5}^* = e\left(H'(K^*), \tilde{g}^{r^*}\right) \cdot e\left(g^{(ay_i - ay_j) \cdot x}, \tilde{g}^{r^*}\right) = e\left(H'(K^*) \cdot X^{(ay_i - ay_j)}, \tilde{g}^{r^*}\right).$$

For $R_{i_2}^*$, the original ciphertext generation is similar to $R_{i_1}^*$. Then $\mathcal{B}$ sends challenge ciphertext set $C^* = (C_{i_1}^*, C_{i_2}^*)$ to $\mathcal{A}$. Both of them are valid encryption for Q_γ^*.

- ***Query phase 2***: $\mathcal{A}$ can ask the same types of queries as in *Query phase 1*, except for the following.
1) ***Trapdoor queries***: The queries of $Trapdoor_{R_i}$ and $Trapdoor_{R_j}$ are not allowed if the generated trapdoor is distinguishable for Q_0^* and Q_1^*.
2) ***Test queries***: The queries of $Test$ and $Test_{Re}$ are not allowed if the query result is distinguishable for (C^*, Q_0^*) and (C^*, Q_1^*).
- ***Guess***: $\mathcal{A}$ outputs a guess $\gamma' \in \{0,1\}$. If $\gamma' = \gamma$, then $\mathcal{B}$ outputs 1 meaning $T = e(g, \tilde{g})^{za^{n+1}}$. Otherwise, it outputs 0 meaning $T \neq e(g, \tilde{g})^{za^{n+1}}$ but a random element in G_T.

Probability Analysis: In the guess phase, the adversary $\mathcal{A}$ is capable to break the scheme with the advantage $Adv_{\mathcal{A}}^{Game_{Server}}(k) = \varepsilon$, the advantage of $\mathcal{B}$ against the n-ABDHE problem is $Adv_{\mathcal{B}}^{n-ABDHE}(k) = \varepsilon_1$. If $T = e(g, \tilde{g})^{za^{n+1}}$, $\mathcal{A}$ must satisfy $|\mathcal{P}r[\gamma' = \gamma] - 1/2| \geq \varepsilon$. Else T is uniform in G_T, and $s^*, s_j^*, \tau^* \in Z_p^*$, $K^* \in G_T$ are randomly chosen, $(C_{i,1}^*, C_{i,2}^*, C_{i,3}^*, B_\varphi^*)$ and $(C_{j,1}^*, C_{j,2}^*, C_{j,3}^*, C_{j,4}^*, C_{j,5}^*, C_{j,6}^*, B_\varphi^*)$ are uniformly random from $\mathcal{A}$'s view, thus the original ciphertext and the re-encrypted ciphertext for $R_{i_1}^*$ and $R_{i_2}^*$ cannot reveal any information regarding the bit γ, then $|\mathcal{P}r[\gamma' = \gamma]| = 1/2$. Therefore, when a is uniform in Z_p^*, T is uniform in G_T, we have

$$\Big|\mathcal{P}r\left[\mathcal{B}(g, \cdots g^{a^n}, \tilde{g}, \cdots \tilde{g}^{a^n}, \tilde{g}^z, \tilde{g}^{za^{n+2}}, e(g, \tilde{g})^{za^{n+1}}) = 1\right] - \mathcal{P}r\left[\mathcal{B}(g, \cdots g^{a^n}, \tilde{g}, \cdots \tilde{g}^{a^n}, \tilde{g}^z, \tilde{g}^{za^{n+2}}, T) = 1\right]\Big| \geq |(1/2 \pm \varepsilon) - 1/2| = \varepsilon$$

as required.

We summarize the above statements into a bound on adversary's advantage in $Game_{Server}$ is that $\varepsilon \leq \varepsilon_1$.

$Game_{Receiver}$: $\mathcal{A}$ is assumed to be an outside attacker including the receiver.

We first let the challenger set the groups G_1, G_2 and G_T with an efficient bilinear map e, a generator g of G_1 and a generator $\tilde{g}$ of G_2. Suppose that $\mathcal{B}$ accepts a properly distributed DBDH instance $(g, g^a, g^b, g^c, \tilde{g}, \tilde{g}^a, \tilde{g}^b, \tilde{g}^c, T)$ and has to distinguish $T = e(g, \tilde{g})^{abc}$ from a random element in G_T.

- ***Setup***: $\mathcal{B}$ generates and outputs the scheme's global parameters $GP = \{p, G_1, G_2, G_T, e, g, \tilde{g}, H, H'\}$, where $H: \{0,1\}^* \to Z_p^*$ is a hash function, $H': G_T \to G_1$ is a cryptographic hash function. Define public key $X = g^a$, $\tilde{V} = \tilde{g}^b$, which means the private key of data server is a that is unknown to $\mathcal{B}$. The data server's key pair is $(pk_s, sk_s) = ((X, \tilde{V}), a)$. Then $\mathcal{B}$ sends (GP, pk_s) to $\mathcal{A}$.
- ***Query phase 1***: $\mathcal{A}$ makes the following queries.
1) ***Public key queries***: $\mathcal{A}$ adaptively queries a public key for R_i or R_j. $\mathcal{B}$ randomly chooses $y_i \in Z_p^*$, computes $Y_i = g^{y_i}$, the public and private key pair of R_i is $(pk_i, sk_i) = (Y_i, y_i)$. The key generation of R_j is similar to that of R_i. Then, $\mathcal{B}$ sends pk_i or pk_j to $\mathcal{A}$.
2) ***Private key queries***: $\mathcal{A}$ adaptively queries a private key for R_i or R_j. $\mathcal{B}$ responds the private key $sk_i = y_i$ or $sk_j = y_j$ to $\mathcal{A}$.
3) ***Trapdoor queries for delegator***: $\mathcal{A}$ adaptively

queries a keyword set $Q = (q_1, q_2, \cdots, q_m), m \leq n$ to obtain a trapdoor for R_i. $\mathcal{B}$ randomly chooses $T_{i,-1}, \zeta \in Z_p^*$, computes $T_{i,-2} = g^{\zeta}$,

$$T_{i,\varphi} = g^{m^{-1} \cdot T_{i,-1} \cdot y_i \cdot \sum_{\mu=1}^{m} H(q_\mu)^{\varphi}} \cdot (g^a)^{\zeta} = g^{m^{-1} \cdot T_{i,-1} \cdot y_i \cdot \sum_{\mu=1}^{m} H(q_\mu)^{\varphi}} \cdot X^{\zeta},$$

where $0 \leq \varphi \leq (n+1)$. $\mathcal{B}$ has successfully simulated the trapdoor $T_{i,Q} = (T_{i,-1}, T_{i,-2}, T_{i,\varphi})$, $0 \leq \varphi \leq (n+1)$ for R_i, then $\mathcal{B}$ responds $T_{i,Q}$ to $\mathcal{A}$.

4) ***Test queries for delegator***: $\mathcal{A}$ adaptively queries any keyword set Q and any *PECK* ciphertext C_i for the test query. $\mathcal{B}$ first makes a trapdoor query on Q to get trapdoor $T_{i,Q}$, then runs algorithm $Test$ and responds the result to $\mathcal{A}$.

If $\mathcal{A}$ queries delegation path for any ciphertext, runs the queries 5) to 9).

5) ***Delegation path queries***: $\mathcal{A}$ adaptively queries delegation path for a *PECK* ciphertext C_i of R_i.

If it's the first time to query delegation path for the ciphertext C_i, $\mathcal{B}$ runs algorithm $CreatPath$ to create path $Pa_i = (R_i, R_1, R_2, \cdots, R_{l_i})$. Then, $\mathcal{B}$ selects $s_j \in Z_p^*$, $K_j \in G_T$ for $j = 1, \cdots, l_i$, computes $rk_{j-1\to j} = (rk^1_{j-1\to j}, rk^2_{j-1\to j}, rk^3_{j-1\to j})$, in which $rk^1_{j-1\to j} = g^{s_j}$, $rk^2_{j-1\to j} = H'(K_1 \cdot K_2 \cdots K_j) \cdot (g^a)^{(y_i - y_j)}$, and

$$rk^3_{j-1\to j} = \begin{cases} K_j \cdot e(g^a, \tilde{g}^b)^{s_j}, j = 1 \\ K_j \cdot e(g^a, \tilde{g}^b)^{s_j - s_{j-1}}, j > 1 \end{cases}.$$

$\mathcal{B}$ has successfully simulated the re-encryption key, then $\mathcal{B}$ responds "Pa_i is created." to $\mathcal{A}$.

Otherwise, $\mathcal{B}$ outputs '$\perp$' to indicate an invalid query.

6) ***Re-encryption key queries***: $\mathcal{A}$ adaptively queries re-encryption key $rk_{j-1\to j}$ for the delegatee R_j in Pa_i. $\mathcal{B}$ first checks whether Pa_i is created, and whether $(pk_{j-1}, pk_j) \in Pa_i$. If not, $\mathcal{B}$ outputs '$\perp$' to indicate that the query is invalid. Otherwise, $\mathcal{B}$ returns re-encryption key $rk_{j-1\to j}$ to $\mathcal{A}$.

7) ***Re-encryption queries***: $\mathcal{A}$ adaptively queries ciphertext C_{j-1} to generate re-encrypted ciphertext C_j in the path Pa_i. $\mathcal{B}$ first checks whether Pa_i is created, and whether $(pk_{j-1}, pk_j) \in Pa_i$. If not, $\mathcal{B}$ outputs '$\perp$' to indicate that the query is invalid. Otherwise, $\mathcal{B}$ runs algorithm $RePECK$ to generate re-encrypted ciphertext C_j as an honest user does, then responds the re-encrypted ciphertext C_j to $\mathcal{A}$.

8) ***Trapdoor queries for delegatee***: $\mathcal{A}$ adaptively queries a keyword set $Q = (q_1, q_2, \cdots, q_m), m \leq n$ to obtain a trapdoor for R_j. $\mathcal{B}$ randomly chooses $T_{j,-1}, \zeta \in Z_p^*$, computes $T_{j,-2} = g^{\zeta}$,

$$T_{j,\varphi} = g^{m^{-1} \cdot T_{j,-1} \cdot y_j \cdot \sum_{\mu=1}^{m} H(q_\mu)^{\varphi}} \cdot (g^a)^{\zeta} = g^{m^{-1} \cdot T_{j,-1} \cdot y_j \cdot \sum_{\mu=1}^{m} H(q_\mu)^{\varphi}} \cdot X^{\zeta},$$

where $0 \leq \varphi \leq (n+1)$. Then, $\mathcal{B}$ responds the trapdoor $T_{j,Q} = (T_{j,-1}, T_{j,-2}, T_{j,\varphi})$, $0 \leq \varphi \leq (n+1)$ to $\mathcal{A}$.

9) ***Test queries for delegatee***: $\mathcal{A}$ adaptively queries any keyword set Q and any *RePECK* ciphertext C_j for the test query. $\mathcal{B}$ first makes a trapdoor query on Q to get trapdoor $T_{j,Q}$, then runs algorithm $Test_{Re}$ and responds the result to $\mathcal{A}$.

- ***Challenge***: Once $\mathcal{A}$ finishes the above queries, it outputs two keyword sets Q_0^* and Q_1^* of equal length, a delegator $R^*_{i_1}$ and it's delegation path $Pa^*_{i_1}$, a user $R^*_{i_2}$. Upon receiving them, $\mathcal{B}$ randomly chooses $\gamma \in \{0,1\}$, let $Q^*_\gamma = (Q^*_{\gamma,1}, Q^*_{\gamma,2}, \cdots, Q^*_{\gamma,m})$.
  For $R^*_{i_1}$, $\mathcal{B}$ randomly selects $r^*, s^*_j, \tau^* \in Z_p^*$, $K^* \in G_T$, and constructs a polynomial $f^*(x) = \eta^*_{n+1}x^{n+1} + \eta^*_n x^n + \cdots + \eta^*_1 x + \eta^*_0$ such that $H(Q^*_{\gamma,1})$, $H(Q^*_{\gamma,2}), \cdots, H(Q^*_{\gamma,n})$ and τ^* are $(n+1)$ roots of the equation $f^*(x) = 1$. Then $\mathcal{B}$ sets

$$t^* = T, C^*_{i,1} = g^c, C^*_{i,2} = T \cdot e(g^a, Y_i)^{r^*}, C^*_{i,3} = \tilde{g}^{r^*}, B^*_\varphi = (C^*_{i,3})^{\eta^*_\varphi}, 0 \leq \varphi \leq (n+1),$$

  and

$$C^*_{j,1} = C^*_{i,1} = g^c, C^*_{j,2} = C^*_{i,2} = T \cdot e(g^a, Y_i)^{r^*}, C^*_{j,3} = C^*_{i,3} = \tilde{g}^{r^*}, C^*_{j,4} = g^{s^*_j}, C^*_{j,5} = e\left(H'(K^*) \cdot (g^a)^{(y_i - y_j)}, C^*_{j,3}\right), C^*_{j,6} = K^* \cdot e(g^a, \tilde{g}^b)^{s^*_j}.$$

  If $T = e(g,\tilde{g})^{abc}$, then $t^* = e(g,\tilde{g})^{abc} = e(X, \tilde{V})^c$.
  For $R^*_{i_2}$, the original ciphertext generation is similar to $R^*_{i_1}$. Then $\mathcal{B}$ sends the challenge ciphertext set $C^* = (C^*_{i_1}, C^*_{i_2})$ to $\mathcal{A}$. Both of them are valid encryption for Q^*_γ.
- ***Query phase 2***: $\mathcal{A}$ can ask the same types of queries as in *Query phase 1*, except for the following.
  1) ***Test queries***: The queries of $Test$ and $Test_{Re}$ are not allowed if the query result is distinguishable for (C^*, Q_0^*) and (C^*, Q_1^*).
- ***Guess***: $\mathcal{A}$ outputs a guess $\gamma' \in \{0,1\}$. If $\gamma' = \gamma$, then $\mathcal{B}$ outputs 1 meaning $T = e(g,\tilde{g})^{abc}$. Otherwise, it outputs 0 meaning $T \neq e(g,\tilde{g})^{abc}$ but a random element in G_T.

Probability Analysis: In the guess phase, the adversary $\mathcal{A}$ is capable to break the scheme with the advantage $Adv_{\mathcal{A}}^{Game_{Receiver}}(k) = \varepsilon$, the advantage of $\mathcal{B}$ against the DBDH problem is $Adv_{\mathcal{B}}^{DBDH}(k) = \varepsilon_2$. If $T = e(g,\tilde{g})^{abc}$, $\mathcal{A}$ must satisfy $|\mathcal{P}r[\gamma' = \gamma] - 1/2| \geq \varepsilon$. Else T is uniformly random, and $t^* = T$ is uniform in G_T. Since $r^*, s^*_j, \tau^* \in Z_p^*, K^* \in G_T$ are also randomly chosen, $(C^*_{i,1}, C^*_{i,2}, C^*_{i,3}, B^*_\varphi)$ and $(C^*_{j,1}, C^*_{j,2}, C^*_{j,3}, C^*_{j,4}, C^*_{j,5}, C^*_{j,6}, B^*_\varphi)$ are uniformly random from $\mathcal{A}$'s view, thus the original ciphertext and the re-encrypted ciphertext for $R^*_{i_1}$ and $R^*_{i_2}$ cannot reveal any information regarding the bit γ, then $|\mathcal{P}r[\gamma' = \gamma]| = 1/2$. Therefore, when a, b, c are uniform in Z_p^*, T is uniform in G_T, we have

$$\left|\mathcal{P}r\left[\mathcal{B}(g, g^a, g^b, g^c, \tilde{g}, \tilde{g}^a, \tilde{g}^b, \tilde{g}^c, e(g,\tilde{g})^{abc}) = 1\right] - \mathcal{P}r\left[\mathcal{B}(g, g^a, g^b, g^c, \tilde{g}, \tilde{g}^a, \tilde{g}^b, \tilde{g}^c, e(g,\tilde{g})^{r}) = 1\right]\right| \geq |(1/2 \pm \varepsilon) - 1/2| = \varepsilon$$

as required.

We summarize the above statements into a bound on adversary's advantage in $Game_{Receiver}$ is that $\varepsilon \leq \varepsilon_2$.

This completes the proof of *Theorem 1*.

7.2 Security Proof for IND-KGA Security

Theorem 2: The AP-SCF-PECKS scheme is IND-KGA security in the standard model assuming that the SXDH problem is intractable. If there is a PPT adversary $\mathcal{A}$, who breaks $\varepsilon-$IND-KGA security of AP-SCF-PECKS scheme, then we can construct a PPT adversary $\mathcal{B}$ using $\mathcal{A}$ to break the ε' SXDH problem with $\varepsilon' \geq \varepsilon$.

Proof: The proof of *theorem 2* is as follows.

We first let the challenger set the groups G_1, G_2 and G_T with an efficient bilinear map e, a generator g of G_1, a generator $\tilde{g}$ of G_2. Suppose that $\mathcal{B}$ accepts a properly distributed DDH (SXDH problem in G_1) instance (g, g^a, g^b, T) and has to distinguish $T = g^{ab}$ from a random element in G_1.

- ***Setup***: $\mathcal{B}$ generates and outputs the scheme's global parameters $GP = \{p, G_1, G_2, G_T, e, g, \tilde{g}, H, H'\}$, where $H : \{0,1\}^* \rightarrow Z_p^*$ is a hash function, $H' : G_T \rightarrow G_1$ is a cryptographic hash function. $\mathcal{B}$ randomly chooses $\tilde{V} \in G_2^*$, defines $X = g^b$, let $(pk_s, sk_s) = ((X, \tilde{V}), b)$ be the data server's public and private key pair, and the private key b is unknown to $\mathcal{B}$. Then $\mathcal{B}$ sends (GP, pk_s) to $\mathcal{A}$.
- ***Query phase 1***: $\mathcal{A}$ makes the following queries.
  1) ***Public key queries***: $\mathcal{A}$ adaptively queries a public key for R_i or R_j. $\mathcal{B}$ randomly chooses $y_i \in Z_p^*$, computes $Y_i = g^{y_i}$, the public and private key pair of R_i is $(pk_i, sk_i) = (Y_i, y_i)$. The key generation of R_j is similar to that of R_i. Then, $\mathcal{B}$ sends pk_i or pk_j to $\mathcal{A}$.
  2) ***Trapdoor queries for delegator***: $\mathcal{A}$ adaptively queries a keyword set $Q = (q_1, q_2, \cdots, q_m), m \leq n$ to obtain a trapdoor for R_i. $\mathcal{B}$ randomly chooses $T_{i,-1}, \zeta \in Z_p^*$, computes $T_{i,-2} = g^{\zeta}$,
  $$T_{i,\varphi} = g^{m^{-1} \cdot T_{i,-1} \cdot y_i \cdot \sum_{\mu=1}^{m} H(q_\mu)^{\varphi}} \cdot (g^b)^{\zeta} = g^{m^{-1} \cdot T_{i,-1} \cdot y_i \cdot \sum_{\mu=1}^{m} H(q_\mu)^{\varphi}} \cdot X^{\zeta},$$
  where $0 \leq \varphi \leq (n+1)$. $\mathcal{B}$ has successfully simulated the trapdoor $T_{i,Q} = (T_{i,-1}, T_{i,-2}, T_{i,\varphi})$, $0 \leq \varphi \leq (n+1)$, then $\mathcal{B}$ responds the trapdoor $T_{i,Q}$ to $\mathcal{A}$.
  If $\mathcal{A}$ queries delegation path for any ciphertext, runs the queries 3).
  3) ***Trapdoor queries for delegatee***: $\mathcal{A}$ adaptively queries a keyword set $Q = (q_1, q_2, \cdots, q_m), m \leq n$ to obtain a trapdoor for R_j. $\mathcal{B}$ randomly chooses $T_{j,-1}, \zeta \in Z_p^*$, computes $T_{j,-2} = g^{\zeta}$,
  $$T_{j,\varphi} = g^{m^{-1} \cdot T_{j,-1} \cdot y_j \cdot \sum_{\mu=1}^{m} H(q_\mu)^{\varphi}} \cdot (g^b)^{\zeta} = g^{m^{-1} \cdot T_{j,-1} \cdot y_j \cdot \sum_{\mu=1}^{m} H(q_\mu)^{\varphi}} \cdot X^{\zeta},$$
  where $0 \leq \varphi \leq (n+1)$. Then $\mathcal{B}$ responds the trapdoor $T_{j,Q} = (T_{j,-1}, T_{j,-2}, T_{j,\varphi})$, $0 \leq \varphi \leq (n+1)$ to $\mathcal{A}$.
- ***Challenge***: Once $\mathcal{A}$ finishes the above queries, it outputs two keyword sets Q_0^* and Q_1^* of equal length, a delegator $R_{i_1}^*$ and it's delegation path $Pa_{i_1}^*$, a user $R_{i_2}^*$. Upon receiving them, $\mathcal{B}$ randomly chooses $\gamma \in \{0,1\}$, let $Q_\gamma^* = (Q_{\gamma,1}^*, Q_{\gamma,2}^*, \cdots, Q_{\gamma,m}^*)$.
  For $R_{i_1}^*$, $\mathcal{B}$ randomly selects $T_{i,-1}^* \in Z_p^*$ and sets
  $$T_{i,-2}^* = g^a, T_{i,\varphi}^* = g^{m^{-1} \cdot T_{i,-1}^* \cdot y_i \cdot \sum_{\mu=1}^{m} H(Q_{\gamma,\mu}^*)^{\varphi}} \cdot T, \quad 0 \leq \varphi \leq (n+1).$$
  If $T = g^{ab}$, then
  $$T_{i,\varphi}^* = g^{m^{-1} \cdot T_{i,-1}^* \cdot y_i \cdot \sum_{\mu=1}^{m} H(Q_{\gamma,\mu}^*)^{\varphi}} \cdot g^{ab} = g^{m^{-1} \cdot T_{i,-1}^* \cdot y_i \cdot \sum_{\mu=1}^{m} H(Q_{\gamma,\mu}^*)^{\varphi}} \cdot X^a.$$
  For $R_{i_2}^*$, the trapdoor generation is similar to $R_{i_1}^*$. Then $\mathcal{B}$ sends the challenge trapdoor set $T_{i,Q_\gamma^*}^* = (T_{i_1,Q_\gamma^*}^*, T_{i_2,Q_\gamma^*}^*)$ to $\mathcal{A}$. Both of them are valid trapdoor for Q_γ^*.
- ***Query phase 2***: $\mathcal{A}$ can ask the same types of queries as in *Query phase 1*, except for the following.
  1) ***Trapdoor queries***: The queries of $Trapdoor_{R_i}$ and $Trapdoor_{R_j}$ are not allowed if the generated trapdoor is distinguishable for Q_0^* and Q_1^*.
- ***Guess***: $\mathcal{A}$ outputs a guess $\gamma' \in \{0,1\}$. If $\gamma' = \gamma$, then $\mathcal{B}$ outputs 1 meaning $T = g^{ab}$. Otherwise, it outputs 0 meaning $T \neq g^{ab}$ but a random element in G_1.

Probability Analysis: In the guess phase, the adversary $\mathcal{A}$ is capable to break the scheme with the advantage $Adv_{\mathcal{A}}^{IND-KGA}(k) = \varepsilon$, the advantage of $\mathcal{B}$ against the SXDH problem is $Adv_{G_1,\mathcal{B}}^{SXDH}(k) = \varepsilon'$. If $T = g^{ab}$, $\mathcal{A}$ must satisfy $|\mathcal{P}r[\gamma' = \gamma] - 1/2| \geq \varepsilon$. Else T is a random element in G_1, thus $T_{i,\varphi}^*$ for $0 \leq \varphi \leq (n+1)$ are uniformly random and independent element in G_1. Since $T_{i,-1}^* \in Z_p^*$ is also randomly chosen, $(T_{i,-1}^*, T_{i,-2}^*, T_{i,\varphi}^*)$ for $0 \leq \varphi \leq (n+1)$ are uniformly random from $\mathcal{A}$'s view, thus $T_{i,Q_\gamma^*}^*$ cannot reveal any information regarding the bit γ, then $|\mathcal{P}r[\gamma' = \gamma]| = 1/2$. Therefore, when a, b are uniform in Z_p^*, T is a random element in G_1, we have

$$\left|\mathcal{P}r\left[\mathcal{B}(g, g^a, g^b, g^{ab}) = 1\right] - \mathcal{P}r\left[\mathcal{B}(g, g^a, g^b, g^r) = 1\right]\right| \geq |(1/2 \pm \varepsilon) - 1/2| = \varepsilon$$

as required.

We summarize the above statements into a bound on adversary's advantage is that $\varepsilon \leq \varepsilon'$.

This completes the proof of *Theorem 2*.

8 PERFORMANCE ANALYSIS

In this section, we first compare theoretical performance of AP-SCF-PECKS with other relevant schemes. Then, we give the simulation results to evaluate the practical performance.

8.1 Comparison

In this subsection, we have compared AP-SCF-PECKS scheme with other relevant schemes. The comparison results about function, communication and computation overhead are shown in Table 2, Table 3 and Table 4.

8.1.1 Functionality Comparison

In Table 2, we show the functions comparison as follows.

- ***Autonomous path delegation:*** Autonomous path delegation enables that each delegatee to be trusted and expected by the delegator, the delegatees in the path have search and access rights of the encrypted data with priority from high to low. At the same time, the delegation strictly follows the delegation path and cannot be transferred to other paths. [26] and our scheme can support autonomous path delegation function, but [26] is a single proxy re-encryption scheme and cannot support ciphertext retrieval.

TABLE 2: Functions Comparison with Other Schemes

Scheme	F1	F2	F3	F4	F5	F6	F7
[2]	No	No	No	No	No	No	No
[5]	No	No	No	Yes	No	No	No
[6]	No	No	No	Yes	No	No	No
[7]	No	Yes	Yes	Yes	Yes	Yes	Yes
[10]	No	Yes	No	No	No	Yes	No
[15]	No	Yes	No	Yes	No	Yes	Yes
[16]	No	Yes	No	No	Yes	Yes	Yes
[19]	No	No	No	No	Yes	No	No
[20]	No	No	No	No	Yes	No	No
[21]	No	No	No	No	Yes	Yes	No
[23]	No	No	No	No	Yes	No	No
[26]	Yes	No	No	No	No	No	No
Our	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Note: F1: Autonomous path delegation; F2: Conjunctive keyword; F3: Keyword subset search; F4: Proxy search; F5: Against KG attack; F6: Standard model; F7: Revocation.

- ***Conjunctive keyword:*** Conjunctive keyword search function can provide multiple keywords search, instead of performing keyword search multiple times and getting the need through intersection calculation. Searchable encryption schemes in [7], [10], [15], [16] and our scheme can provide conjunctive keyword search function.
- ***Keyword subset search:*** Keyword subset search function makes the number of keywords in a search is unrestricted, as long as the search keyword set is a subset of the encrypted keyword set. The scheme in [7] and our scheme can provide keyword subset search function. Unfortunately, [7] cannot support path delegation fuction.
- ***Proxy search:*** The proxy re-encryption technology will greatly facilitate users to delegate the search and access rights of their ciphertext. [5], [6], [7], [15] and our scheme can provide proxy search function for users by introducing proxy re-encryption technology. Among them, [15] adopts proxy re-encryption technology, which is mainly used to realize role inheritance and user revocation.
- ***Against KG attack:*** Outside attackers can easily realize KG attack by stealing communication information on the public transmission channel. Most of the existing schemes such as [2], [5], [6], [10] and [15] cannot resist KG attacks. The schemes in [7], [16], [19], [20], [21], [23] and our scheme can support secure channel free and resist KG attacks.
- ***Standard model:*** The schemes in [2], [5], [6], [19], [20], [23] and [26] have been proved to be secure in random oracle model. However, the security proved in standard model is higher than that in random oracle model. The schemes in [7], [10], [15], [16], [21] and our scheme have been proven secure in standard model.
- ***Revocation:*** Revocation is essential for searchable encryption scheme that support multi-user scenarios. The scheme in [7], [15], [16] and our scheme support revocation. Among them, [7] implements revocation by embedding the deadline into the re-encrypted ciphertext. [15] supports complete user revocation and role-level revocation, [16] supports user revocation and user's attribute revocation. Our scheme support user revocation, authorized user revocation in delegation path and path revocation.

Thus, our AP-SCF-PECKS scheme has multiple useful functions and high security.

8.1.2 Communication and Computation Overhead Comparison

In Table 3 and Table 4, we show the comparison of communication and computation overhead. Among them, [2], [5], [6], [7], [15], [20], [23] and [26] are based on symmetric bilinear pairing e: $G \times G \rightarrow G_T$. [16], [21] and our scheme are based on asymmetric bilinear pairing e: $G_1 \times G_2 \rightarrow G_T$. The ciphertext size $aG_1+bG_2+cG_T+dZ_p+e\{0,1\}^{log_p}+f\{0,1\}^k$ means that a ciphertext consists of a elements in G_1 and b elements in G_2 (symmetric bilinear pairing expressed as G), c elements in G_T, d elements in Z_p, e elements in $\{0,1\}^{log_p}$ and f elements in $\{0,1\}^k$, where a, b, c, d, e, f are positive integers. n is the size of keyword set, l_i is the number of delegatees in the delegation path. n_s and $|S_{ID_u}|$ are the number of organizations and roles associated with a user, $|\Gamma_\Phi|$, $|\Gamma|$ are the number of organizations and roles associated with a ciphertext in [15]. n_{or} is the number of "OR" gates in an access policy, $|S_{ID_u}|$, n_t are the number of attributes associated with a user and a trapdoor in [16]. t_{e1} and t_{e2} are the time costs for computing an exponentiation in G_1 and G_2 (symmetric bilinear pairing expressed as t_e). t_g and t_p are the time costs for computing an exponentiation in G_T and a bilinear pairing. t_h is the time cost for computing a hash function $H' : G_T \rightarrow G_1$ in our scheme, a hash function $H' : G_T \rightarrow G$ in [26], a hash function $H : \{0,1\}^* \rightarrow G$ in [2], [5], [6], [20], and the running time of these hash functions is almost the same. t_m is the time cost for computing a map-to-point hash in [23]. t_d is the time cost for computing a division. The running time for all the other basic operations include t_d will be ignored since they are much more efficient than t_p, t_e, t_{e1}, t_{e2}, t_g and t_h.

- The size and generation overhead of key in [15] are linear with the number n_s of organizations and the number $|S_{ID_u}|$ of roles associated with a user. In [16], they are linear with the number $|S_{ID_u}|$ of attributes associated with a user. In [21], they are linear with number n of keywords. However, the size of key in our scheme are only one element in G_2 and Z_p, and the generation overhead of our scheme also only needs an exponentiation operation in G_2.
- The size and computation overhead of ciphertext in [15] are linear with the number n of keywords, the number $|\Gamma|$ of roles and the number $|\Gamma_\Phi|$ of organizations associated with a ciphertext. In [16], they are linear with the number n of keywords and the number n_{or} of "OR" gates in an access policy. In [7] and our scheme, they are linear with the number n of keywords. Only [7], [15], [16] and our scheme can support conjunctive keyword search.
- The size and computation overhead of trapdoor in [15] are linear with the number $|S_{ID_u}|$ of roles associated with a user. Similarly, the computation overhead of search in [15] are linear with the number $|S_{ID_u}|$ of roles associated with a user. In [16], the size and

TABLE 3: Communication Overhead Comparison with Other Schemes

Scheme	T1	T2	T3	T4	T5	T6
[2]	G	Z_p	–	$G+\{0,1\}^{log_p}$	–	G
[5]	G	Z_p	Z_p	$3G+G_T+\{0,1\}^k$	$3G+G_T+\{0,1\}^k$	G
[6]	G	Z_p	Z_p	$G+\{0,1\}^k$	$G+\{0,1\}^k$	G
[7]	G	Z_p	Z_p	$(n+2)G+G_T$	$(n+5)G+G_T$	delegator:$(n+2)G+Z_p$ delegatee:$(n+3)G+Z_p$
[15]	n_sG	$Z_p+(n_s+2\lvert S_{ID_u}\rvert)G$	Z_p	$(2\lvert\Gamma\rvert+2n+4)G+G_T$	–	$(2\lvert S_{ID_u}\rvert+3)G+Z_p$
[16]	G_2	$(\lvert S_{ID_u}\rvert+2)G_1+$ $(\lvert S_{ID_u}\rvert+1)G_2$	–	$nG_1+2G_2+(n_{or}+1)Z_p$	–	$(n_t+2)(G_1+G_2)$
[20]	$2G$	Z_p	–	$G+\{0,1\}^k$	–	$2G$
[21]	$(n+1)G_1+2G_2$	$(n+3)Z_p$	–	$G_1+2G_2+2G_T$	–	G_1+Z_p
[23]	G	Z_p	–	$G+\{0,1\}^k$	–	G
[26]	G	Z_p	$l_i(2G_1+G_T)$	$G+G_T$	$l_i(2G+2G_T)$	–
Our	G_2	Z_p	$l_i(2G_1+G_T)$	$G_1+(n+3)G_2+G_T$	$l_i(2G_1+(n+3)G_2+3G_T)$	delegator:$(n+3)G_1+Z_p$ delegatee:$(n+3)G_1+Z_p$

Note: T1: Public key size; T2: Private key size; T3: Re-encryption key size; T4: Ciphertext size; T5: Re-encrypted ciphertext size; T6: Trapdoor size.

TABLE 4: Computation Overhead Comparison with Other Schemes

Scheme	T7	T8	T9	T10	T11	T12
[2]	t_e	–	$t_p+2t_e+t_h$	–	t_e+t_h	t_p
[5]	t_e	t_d	$2t_p+3t_e+2t_g+3t_h$	t_e	t_e+t_h	t_p
[6]	t_e	t_d	$t_p+2t_e+t_h$	t_e	t_e+t_h	t_p
[7]	t_e	t_d	$(n+2)t_e+2t_g$	$(n+5)t_e$	$(n+3)t_e$	$(n+2)t_p+t_e+3t_g$
[15]	$(2n_s+2\lvert S_{ID_u}\rvert+1)t_e$	–	$(2n+\lvert\Gamma\rvert+\lvert\Gamma_\Phi\rvert+4)t_e+t_g$	–	$(2\lvert S_{ID_u}\rvert+3)t_e$	$(\lvert S_{ID_u}\rvert+2)t_p+(\lvert S_{ID_u}\rvert+1)t_e$
[16]	$(\lvert S_{ID_u}\rvert+2)(t_{e1}+t_{e2})$	–	$(n_{or}+1)t_p+2t_{e_2}+$ $(2n+n_{or}+1)t_{e_1}$	–	$2t_p+(n_t+3)t_{e1}+(n_t+2)t_{e2}$	$4t_p+3t_{e_1}$
[20]	$2t_e$	–	$t_p+2t_e+t_h$	–	$3t_e+2t_h$	$t_p+2t_e+t_h$
[21]	$(n+1)t_{e1}+2t_{e2}$	–	$3t_p+t_{e1}+4t_{e2}+3t_g+t_h$	–	t_{e1}	$4t_p+2t_{e2}+2t_g$
[23]	t_e	–	$2t_p+2t_e+t_m+t_g$	–	$t_p+3t_e+2t_m$	t_p
[26]	t_e	$l_i(t_p+2t_e+2t_h)$	$t_p+t_e+t_g$	l_it_p	–	–
Our	t_{e2}	$l_i(2t_{e1}+t_g+t_h)$	$t_p+2t_{e1}+(n+3)t_{e2}+t_g$	l_it_p	$(n+4)t_{e_1}$	$(n+3)t_p+2t_{e1}+3t_g$

Note: T7: Key generation; T8: Re-encryption key generation; T9: Encryption; T10: Re-encryption; T11: Trapdoor; T12: Test/Search.

computation overhead of trapdoor are linear with the number n_t of attributes in a trapdoor. In [7] and our scheme, they are linear with the number n of keywords, while only [7] and our scheme can support keyword subset search.

- The size and computation overhead of re-encryption key and re-encrypted ciphertext in [26] and our scheme are linear with number l_i of delegatees in delegation path, since they need to generate re-encryption key and re-encrypted ciphertext for each delegatee. The size and computation overhead of re-encryption key and re-encrypted ciphertext for each delegatee are similar to other schemes.
- The schemes in [2], [5], [6], [20], [21] and [23] may have less computation overhead in encryption, trapdoor generation, test or search operation, but they only can support single keyword search.

In conclusion, the efficiency of AP-SCF-PECKS scheme is moderate, and we can support multiple useful functions and high security.

8.2 Evaluation

We evaluate AP-SCF-PECKS and [7], [15], [16] by implementing the scheme on an experimental workbench. The experiments have been executed by using the pairing based cryptography (PBC) library [29] on a PC with running Windows10, I5-8265U 1.6-GHz processor, and 8 GB memory. To make the result more accurate, we have executed each algorithm multiple times and calculated the average execution time as the result. At the same time, we have compared the experimental time with the theoretical time, that the experimental time are almost the same as the theoretical time.

Fig.3 shows the experimental results for each algorithm of our scheme. The experiment were carried out in four different elliptic curves: MNT159, MNT201, MNT224, and BN160, and the pairings Type D159, D201, D224, and F on these four curves are asymmetric Type 3 pairings. As shown in Fig.3, the MNT159 curve for each algorithm has less computation time than MNT201 and MNT224 curves. Although the cost of BN160 curve is less than that of MNT159 curve for the algorithm $PECK$, $Trapdoor_{R_i}$ and $Trapdoor_{R_j}$, but considering the overall cost of AP-SCF-PECKS scheme, M-NT159 curve has lower cost and better performance. Therefore, the cost of the scheme will be evaluated based on the MNT159 curve, which can provide security level of 80-bit. The average execution time of $GlobalSetup$, $KeyGen_{Ser}$, $KeyGen_R$ algorithm are approximately 0.2s, 19.8ms and 18.4ms. The execution time of $GlobalSetup$ algorithm includes the time of two bilinear pairing operations $e(g, \tilde{g})$ and $e(X, \tilde{V})$. These two operations do not consume additional computation time in other phases.

As shown in Fig.3, we demonstrate the execution time

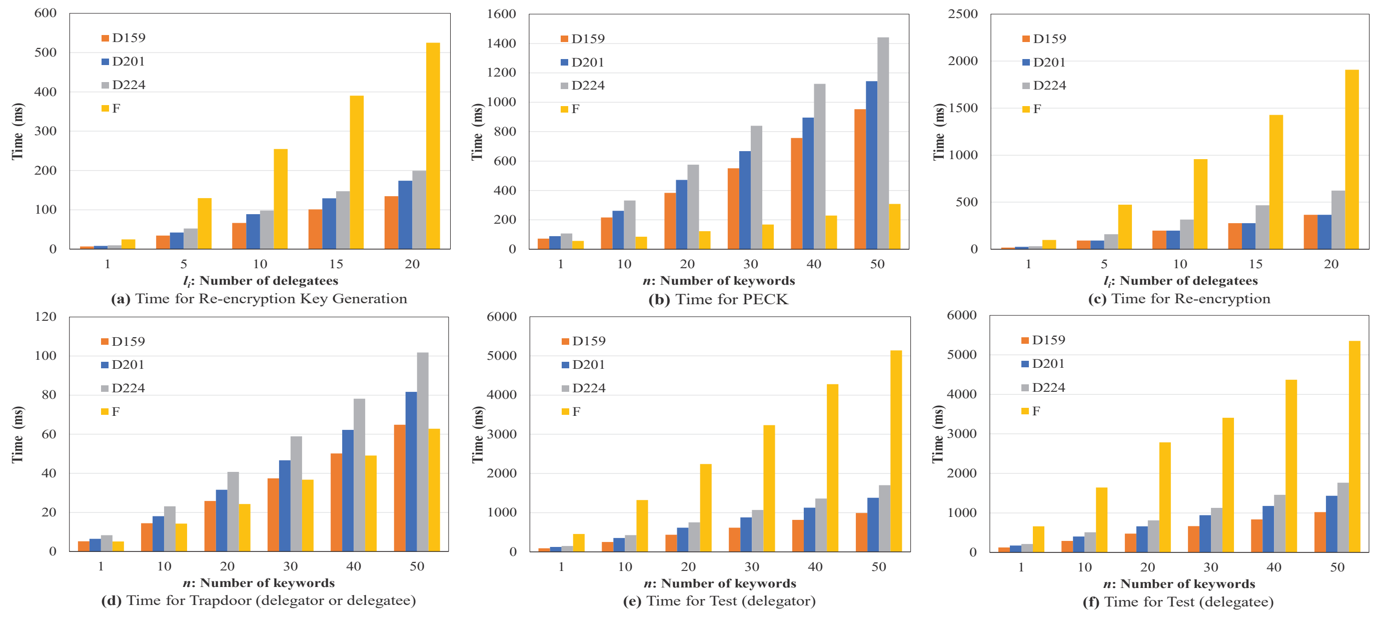

Fig. 3: Operation Time of the Algorithms in AP-SCF-PECKS

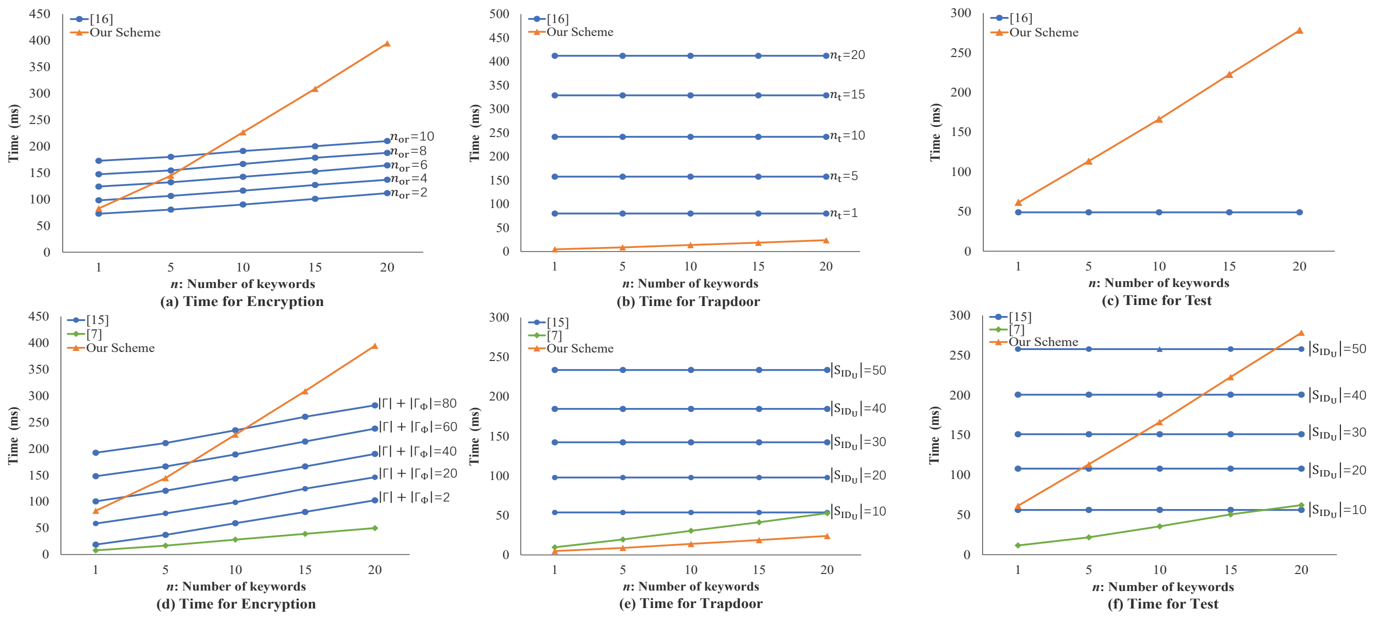

Fig. 4: Operation Time in AP-SCF-PECKS, Yang et al.'s Scheme [7], Sultan et al.'s Schemes [15] and [16][1]

of each algorithm with different values of n and l_i. Among them, $PECK$, $Trapdoor_{R_i}$ and $Trapdoor_{R_j}$, $Test$, $Test_{R_j}$ algorithms in (b) (d) (e) (f) are increase linearly with the number n of keywords. (a) (c) show the execution time of $ReKeyGen$ and $RePECK$ algorithms, they increase linearly with the number l_i of delegatees in delegation path. In general, the number of query keywords and delegatees are normally less than 10. It can be seen from Fig.3 that when n and l_i are taken as 10, the total execution time of our scheme is close to 1s. Since the computation cost of each entity is one-time, the execution time of the simulation is acceptable.

Fig.4 shows the experimental comparison of the proposed scheme with the related works [7], [15], [16], and we consider the most frequently operations *Encryption*, *Trapdoor Generation*, and *Search/Test*. For [16] and our scheme, we use asymmetric Type D159 pairing. For [7] and [15], we use symmetric Type A pairing.

The asymmetric Type D159 pairing has large calculation overhead for pairing and exponentiation in G_2, and has small calculation overhead for exponentiation in G_1. Therefore, as shown in Fig.4 (a) (b) (c), the computation overhead of our scheme is smaller in trapdoor generation phase and larger in encryption and testing phase compared with [16]. However, our scheme can support keyword subset search,

1. (a)(b)(c) are the comparison of computation overhead for AP-SCF-PECKS and [16], which all use the asymmetric Type D159 pairing.

(d)(e)(f) are the comparison of computation overhead for AP-SCF-PECKS, [7] and [15], where AP-SCF-PECKS uses the asymmetric Type D159 pairing, [7] and [15] use the symmetric Type A pairing.

i.e. the number of search keywords is unrestricted. For the case of single keyword search, the cost of our scheme is close to [16].

Similarly, due to the performance of asymmetric Type D159 pairing, as shown in Fig.4 (d) (e) (f), the computation overhead of our scheme is smaller in trapdoor generation phase and larger in encryption and testing phase compared with [7] and [15]. In summary, the asymmetric pairing has lower computational efficiency and higher security than the symmetric pairing. Therefore, our scheme has larger computational overhead in some stages compared with [7] and [15], but can guarantee higher security.

In conclusion, the proposed scheme focus on different authorization entities and scenarios compared with the existing searchable encryption schemes, and the experimental results show that the efficiency of the proposed scheme is moderate for the healthcare cloud application.

9 CONCLUSION

In this paper, we have proposed a novel conjunctive keyword search scheme which can realize autonomous path delegation function, then we apply it to healthcare cloud. It can achieve that patient search and access his EHRs through single or multiple keywords. The patient also can autonomously delegate the search and access right of his EHRs to doctors with the priority from high to low, and the delegation cannot be transfered to other unauthorized users. The security has been formally proved in standard model, which indicates that our scheme guarantees a higher level security. It is also able to support secure channel free and resist KG attacks. The theoretical and experimental evaluation show that the efficiency of the proposed scheme is moderate for healthcare cloud application, which can support multiple useful functions and high security.

ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China Research under Grant 62072371, the Key Research and Development Program of Shaanxi Province under 2021ZDLGY06-02, and Basic Research Program of Qinghai Province under 2020-ZJ-701.

REFERENCES

[1] P. Y. Wu, C. W. Cheng, C. D. Kaddi, J. Venugopalan, R. Hoffman, and M. D. Wang, "comic and electronic health record big data analytics for precision medicine," *IEEE Transactions on Biomedical Engineering*, vol. 64, no. 2, pp. 263–273, 2017.

[2] B. Dan, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Advances in Cryptology - EUROCRYPT 2004, Interlaken, Switzerland, May 2-6*, 2004.

[3] M. Blaze, G. Bleumer, and M. Strauss, "Divertible protocols and atomic proxy cryptography," in *Advances in Cryptology - EUROCRYPT '98, Espoo, Finland, May 31 - June 4*, 1998.

[4] X. A. Wang, X. Huang, X. Yang, L. Liu, and X. Wu, "Further observation on proxy re-encryption with keyword search," *Journal of Systems & Software*, vol. 85, no. 3, pp. 643–654, 2012.

[5] J. Shao, Z. Cao, X. Liang, and L. Huang, "Proxy re-encryption with keyword search," *Information Sciences*, vol. 180, no. 13, pp. 2576–2587, 2010.

[6] W. C. Yau, C. W. Phan, S. H. Heng, and B. M. Goi, "Proxy re-encryption with keyword search: New definitions and algorithms," in *Security Technology, Disaster Recovery and Business Continuity - International Conferences, Held as Part of the Future Generation Information Technology Conference, Jeju Island, Korea, December 13-15*, 2010.

[7] Y. Yang and M. Ma, "Conjunctive keyword search with designated tester and timing enabled proxy re-encryption function for e-health clouds," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 4, pp. 746–759, 2016.

[8] P. Xu, S. He, W. Wang, W. Susilo, and H. Jin, "Lightweight searchable public-key encryption for cloud-assisted wireless sensor networks," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 8, pp. 3712–3723, 2018.

[9] X. Zhang, C. Xu, R. Xie, and C. Jin, "Designated cloud server public key encryption with keyword search from lattice in the standard model," *Chinese Journal of Electronics*, vol. 27, no. 2, pp. 304–309, 2018.

[10] H. Cui, Z. Wan, R. H. Deng, G. Wang, and Y. Li, "Efficient and expressive keyword search over encrypted data in cloud," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 3, pp. 409–422, 2018.

[11] P. Golle, J. Staddon, and B. R. Waters, "Secure conjunctive keyword search over encrypted data," in *Applied Cryptography and Network Security, Second International Conference, Yellow Mountain, China, June 8-11*, 2004.

[12] J. P. Dong, K. Kim, and P. J. Lee, "Public key encryption with conjunctive field keyword search," in *Information Security Applications, 5th International Workshop, Jeju Island, Korea, August 23-25*, 2004.

[13] Y. Miao, J. Ma, X. Liu, J. Weng, H. Li, and H. Li, "Lightweight fine-grained search over encrypted data in fog computing," *IEEE Transactions on Services Computing*, vol. 12, no. 5, pp. 772–785, 2019.

[14] O. Farràs and J. Ribes-González, "Provably secure public-key encryption with conjunctive and subset keyword search," *International Journal of Information Security*, vol. 18, no. 5, pp. 533–548, 2019.

[15] N. H. Sultan, M. Laurent, and V. Varadharajan, "Securing organization's data: A role-based authorized keyword search scheme with efficient decryption," *IEEE Transactions on Cloud Computing, Early access, April. 6*, 2021, Doi: 10.1109/TCC.2021.3071304.

[16] N. H. Sultan, N. Kaaniche, M. Laurent, and F. A. Barbhuiya, "Authorized keyword search over outsourced encrypted data in cloud environment," *IEEE Transactions on Cloud Computing, Early access, July. 30*, 2019, Doi: 10.1109/TCC.2019.2931896.

[17] W. B. Jin, H. S. Rhee, H. A. Park, and H. L. Dong, "Off-line keyword guessing attacks on recent keyword search schemes over encrypted data," in *Secure Data Management, Third VLDB Workshop, Seoul, Korea, September 10-11*, 2006.

[18] W. C. Yau, S. H. Heng, and B. M. Goi, "Off-line keyword guessing attacks on recent public key encryption with keyword search schemes," in *Autonomic and Trusted Computing, 5th International Conference, Oslo, Norway, June 23-25*, 2008.

[19] J. Baek, R. Safavi-Naini, and W. Susilo, "Public key encryption with keyword search revisited," in *Computational Science & Its Applications-iccsa, International Conference, Perugia, Italy, June 30-july 3*, 2008.

[20] H. S. Rhee, rnJong Hwan Park, rnWilly Susilo, and rnDong Hoon Lee, "Trapdoor security in a searchable public-key encryption scheme with a designated tester," *Journal of Systems & Software*, vol. 83, no. 5, pp. p.763–771, 2010.

[21] L. Fang, W. Susilo, C. Ge, and J. Wang, "Public key encryption with keyword search secure against keyword guessing attacks without random oracle," *Information Sciences*, vol. 238, no. Complete, pp. 221–241, 2013.

[22] T. Suzuki, K. Emura, and T. Ohigashi, "A generic construction of integrated secure-channel free peks and pke and its application to emrs in cloud storage," *Journal of medical systems*, vol. 43, no. 5, pp. 1–15, 2019.

[23] Y. Lu, J. Li, and Y. Zhang, "Secure channel free certificate-based searchable encryption withstanding outside and inside keyword guessing attacks," *IEEE Transactions on Services Computing*, vol. PP, no. 99, pp. 1–1, 2020.

[24] H. Guo, Z. Zhang, J. Xu, N. An, and L. Chen, "Non-transferable proxy re-encryption," *The Computer Journal*, vol. 62, no. 4, pp. 490–506, 2019.

[25] X. A. Wang, F. Xhafa, W. Hao, and W. He, "Non-transferable unidirectional proxy re-encryption scheme for secure social cloud

storage sharing," in *International Conference on Intelligent Networking and Collaborative Systems, Ostrava, Czech Republic, Sept 7-9*, 2016.

[26] Z. Cao, H. Wang, and Y. Zhao, "Ap-pre: Autonomous path proxy re-encryption and its applications," *IEEE Transactions on Dependable and Secure Computing*, vol. 16, no. 5, pp. 833–842, 2019.

[27] C. Wang, D. Wang, Y. Tu, G. Xu, and H. Wang, "Understanding node capture attacks in user authentication schemes for wireless sensor networks," *IEEE Transactions on Dependable and Secure Computing, Early access, February. 17*, 2020, Doi: 10.1109/TDSC.2020.2974220.

[28] S. Qiu, D. Wang, G. Xu, and S. Kumari, "Practical and provably secure three-factor authentication protocol based on extended chaotic-maps for mobile lightweight devices," *IEEE Transactions on Dependable and Secure Computing, Early access, September. 8*, 2020, Doi: 10.1109/TDSC.2020.3022797.

[29] B. Lynn, "The stanford pairing based crypto library," *[Online]. Available:http://crypto.stanford.edu/pbc*, accessed May 1, 2020.

Qian Wang received the B.S. degree in measurement and control technology and instrument and the M.S. degrees in information security from Xi'an University of Posts and Telecommunications, in 2012 and 2016, respectively. Then she is with School of Computer Science and Technology, Xi'an University of Posts and Telecommunications, Xian, China. She is currently pursuing the Ph.D. degree with School of Computer, Qinghai Normal University, Qinghai, China. Her research interests include cryptography and privacy preservation.

Chengzhe Lai received his B.S. degree in information security from Xi'an University of Posts and Telecommunications in 2008 and a Ph.D. degree from Xidian University in 2014. He was a visiting Ph.D. student with the Broadband Communications Research (BBCR) Group, University of Waterloo from 2012 to 2014. At present, he is with Xi'an University of Posts and Telecommunications and the National Engineering Laboratory for Wireless Security, Xi'an, China. His research interests include wireless network security and privacy preservation.

Rongxing Lu received the Ph.D. degree from the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada, in 2012. He was an Assistant Professor with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, from 2013 to 2016. He has been an Associate Professor with the Faculty of Computer Science (FCS), University of New Brunswick (UNB), Fredericton, NB, Canada, since 2016. He was a Post-Doctoral Fellow with the University of Waterloo, from 2012 to 2013. Dr. Lu was a recipient of the Governor Generals Gold Medal for his Ph.D. degree from the Department of Electrical and Computer Engineering, University of Waterloo, the 8th IEEE Communications Society (ComSoc) AsiaPacific Outstanding Young Researcher Award in 2013, and the 2016 to 2017 Excellence in Teaching Award from FCS, UNB. He is currently serves as the ViceChair (Publication) of IEEE ComSoc CIS-TC.

Dong Zheng received an M.S. degree in mathematics from Shaanxi Normal University, Xi'an, China, in 1988, and a Ph.D. degree in communication engineering from Xidian University, in 1999. He was a professor in the School of Information Security Engineering, Shanghai Jiao Tong University. He is currently a professor at Xi'an University of Posts and Telecommunications and is also connected with the National Engineering Laboratory for Wireless Security, Xi'an, China. His research interests include provable security and new cryptographic technology.