


Article

Public Key Encryption with Equality Test in a Cloud Environment

Ping Zhang ¹ , Jinbo Li ^{1,*}  and Zhumu Fu ²

¹ School of Mathematics and Statistics, Henan University of Science and Technology, Luoyang 471003, China; zping@haust.edu.cn

² College of Information Engineering, Henan University of Science and Technology, Luoyang 471003, China; fuzhumu@haust.edu.cn

* Correspondence: 200320100641@stu.haust.edu.cn; Tel.: +86-184-3790-7066

Abstract: With the rapid development and wide application of cloud computing and 5G communication, the number of mobile users is increasing rapidly, meaning that cloud storage services are receiving more and more attention. The equality test technology of retrievable encrypted data has become a hot research topic among scholars in recent years. In view of the problem of offline keyword-guessing attacks (KGAs) caused by collusion between internal servers and users, a public key encryption with equality test scheme (RKGA-CET) with higher security against KGAs is proposed. Based on the assumed difficulty of the discrete logarithm problem (DLP) and the properties of bilinear mapping, a specific encryption algorithm that encrypts the keyword twice is designed. In the first encryption stage, we convert the keyword according to the property of isomorphism of a finite field. In the second encryption stage, we encrypt the converted keyword vector and embed the user's private key, and then perform the equality test. The algorithm ensures that the adversary cannot generate legal ciphertexts and implement KGAs when the secondary server is offline. At the same time, the algorithm also supports two authorization modes, in which case users can flexibly choose the corresponding authorization mode according to their own needs. Performance analysis shows that this scheme has overall superiority compared with other similar ones.

Keywords: equality test; discrete logarithm; bilinear mapping; offline keyword-guessing attack



Citation: Zhang, P.; Li, J.; Fu, Z.

Public Key Encryption with Equality Test in a Cloud Environment.

Information **2022**, *13*, 265. <https://doi.org/10.3390/info13060265>

Academic Editor: Georgios Kambourakis

Received: 10 February 2022

Accepted: 19 April 2022

Published: 24 May 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

In recent years, the rapid development of the Internet and information technology has laid a solid foundation for the application of cloud computing. With the wide application of cloud computing technology, more and more enterprises and individuals prefer to outsource data to cloud servers for storage, so as to reduce the occupation of storage space of local computers. However, there is no such thing as a free lunch. While enjoying the convenient services provided by the cloud servers, the users have to pay attention to the problems of data security, privacy, identity information security, etc. [1–4]. The privacy data of the users stored in the cloud may be leaked due to the access of incompletely trusted cloud server providers. Therefore, the data owner needs to encrypt their sensitive data before uploading or sharing private files and subsequently sending them to the cloud server. Although traditional encryption technology can protect data privacy, it does not support the retrieval of encrypted data, and the files taking the form of ciphertexts makes their operation more difficult. When searching for the required file, the users must download all of the ciphertexts to the local computer first, and then find the data they need after decryption. However, this undoubtedly occupies a lot of bandwidth and local storage space, which is obviously unreasonable.

In order to solve the problem of keyword search in ciphertexts, the concept of public key encryption with keyword search has been proposed [5–7]. The users first use the standard public key encryption system to encrypt the plaintexts, and then use the searchable

encryption mechanism to encrypt the keyword and, finally, attach the keyword ciphertext to the plaintext ciphertext and send them to the cloud server for storage. When a user wants to search for a keyword, they can use their private key to generate a trapdoor corresponding to the keyword and send it to the cloud server. Once the cloud server receives this trapdoor, it uses the trapdoor to test the ciphertext of the keyword. If the test is successful, it indicates that the ciphertext contains the keyword required by the user. Then, the server returns the ciphertext containing the keyword to the user. After receiving the ciphertext, the user decrypts it to obtain the required information. Searchable encryption technology solves the problem of searching for keywords in ciphertexts in a single-user environment (or single-public-key environment); that is, this technology can only process the data encrypted by a single public key. However, in a multiuser environment (or multi-public-key environment), searching for keywords encrypted with different public keys in ciphertexts requires equality test technology. The equality test technology, which not only supports keyword retrieval on ciphertexts in multiuser environments, but can also be used to compare whether ciphertexts of different users contain the same keyword, can be regarded as a variant of the searchable encryption technology. Due to the large number of the users in the cloud environment, equality test technology is more practical than searchable encryption technology. As mentioned above, the advantage of equality test technology is that it allows a tester to check whether any two ciphertexts encrypted by different public keys contain the same plaintext without decryption. However, it also has some shortcomings that are exposed in the application process, such as poor flexibility of authorization and too-idealistic proof under a random oracle model. The first point means that users cannot adaptively authorize any number of ciphertexts to testers, which makes the generation and storage of ciphertext-level authorization trapdoors very cumbersome. At the same time, user-level authorization to restrict testers' testing behaviors is difficult. The second point means that a scheme based on an ideal model cannot accurately reflect reality, so it cannot show that instantiation in the real world is safe. Therefore, the research on equality test technology has important theoretical significance and application value.

Equality test technology can be used by medical systems to protect patients' historical cases. Suppose the *Alice* and *Bob* are patients with the same symptoms who belong to the same medical institution, each of whom holds multiple medical records encrypted with their own public keys. The medical institution uses the public cloud server as the running environment of the medical record management system where patients can upload the encrypted data, and often wants to classify the encrypted medical label *Tag* in the medical record management system, so as to find potential diseases by comparing the same diseases. At this time, *Alice* and *Bob* can authorize the medical institution to perform equality tests on their own encrypted medical records, with the aim of classifying and storing these data with the medical institution. Even if these data have been encrypted with different public keys, with the authorization of the patients, the medical institution can compare and classify the encrypted medical records, and then choose the most suitable treatment scheme for their disease. However, if the medical institution is allowed to arbitrarily detect whether the patients' encrypted medical records have the same plaintext messages, this will violate the patients' privacy to a certain extent. Therefore, it is necessary to increase the authorization mechanism to protect the privacy of patients. In this way, when a large number of large-scale equality tests need to be performed on encrypted medical records, patients can directly grant user-level authorization to the medical institution, thereby reducing the waste of time and space in the authorization process, and improving efficiency. If only the label *Tag* needs to be tested, the ciphertext-level authorization can be generated directly for the corresponding label, and only the medical institution is allowed to perform equality tests on these ciphertexts. A secure equality test technology should be able to prevent KGAs by internal and external adversaries—especially internal adversary attacks. However, if the KGA is initiated by the medical institution itself, patients' private data are more likely to be leaked, because the information entropy of the keyword space of the

disease is relatively small, which may even affect patients' lives. Therefore, the medical system should adopt a ciphertext equality test technology with higher security [8,9].

1.1. Our Work

Under this background, our paper proposes a RKGA-CET scheme that can resist offline KGAs. Firstly, the scheme encrypts a user's message according to the isomorphism of the finite field $F_p[x] \bmod f(x)$ generated by the irreducible polynomial $f(x)$ of degree n over the finite field F_p , and the finite field generated by the companion matrix derived from $f(x)$, which is equivalent to the conversion of the keyword. Secondly, based on the DLP assumption of bilinear mapping, the encrypted message is re-encrypted. Finally, the equality test is performed. This scheme makes the probability of the polynomial-time adversary guessing the keyword through the exhaustive method negligible. This is because when the auxiliary server is offline, even if the adversary obtains the legal ciphertext (the converted keyword), the polynomial discrete logarithm problem over the finite field F_p needs to be solved in order to decrypt the corresponding plaintext message from the ciphertext. Therefore, the adversary cannot implement offline KGAs. In addition, our paper also designs two kinds of granularity authorization algorithm. Under the random oracle model, it is proven that our scheme is a one-way (OW-CCA) and indistinguishable (IND-CCA) secure encryption scheme against chosen ciphertext attacks.

In order to explain the highlights of our work more generally, it is necessary to clarify the benefits of secondary encryption. We note that most of the existing schemes are encrypted once; that is, they encrypt the keyword w into the ciphertext c . During the test, a tester needs to input the ciphertexts c_1, c_2 and the trapdoor T , and then calculate whether $H(c_1, T)$ and $H(c_2, T)$ are equal based on a hash function H . Here, we use a relatively abstract expression. The specific situation may be different, but there is no difference in essence. Furthermore, the calculation result of $H(c_1, T)$ is $H(w_1)$, and the calculation result of $H(c_2, T)$ is $H(w_2)$. If $w_1 = w_2$, then it must be the case that $H(w_1) = H(w_2)$, so that the tester can judge whether the plaintexts corresponding to the ciphertexts are the same without decryption. The premise of algorithm security is that the hash function is a random oracle or anti-collision mechanism. However, it is difficult to meet such strong requirements in real applications, and the number of keywords is often limited, which increases the probability of the adversary successfully breaking the algorithm when executing KGAs. Considering this situation, we carry out an "intermediate process" in our scheme; that is, before encrypting the keyword w into the ciphertext c , we first encrypt the keyword w into the ciphertext v , and then encrypt v into c . Through such processing, the calculation result of $H(c_1, T)$ is $H(v_1)$, and the calculation result of $H(c_2, T)$ is $H(v_2)$. If $w_1 = w_2$, then it must be the case that $H(v_1) = H(v_2)$. It should be noted that keywords are no longer required to participate in the equality test, but are required for the results of the "intermediate process". When an adversary carries out KGAs, they can only encrypt the keyword w into v first, and then perform the equality test. However, it can be seen from the construction process of our scheme that the adversary needs to successfully obtain the random number b used by the user in the encryption, with a probability of $1/p^n - 1$ to obtain the correct v , which indicates that the adversary has only a small probability to obtain the result of the "intermediate process", so the probability of successfully executing KGAs is also small. We contend that our scheme has higher security, which is based on this "intermediate process".

1.2. Related Work

In order to enable the users to search ciphertexts directly in ciphertext space, Boneh et al. [10] proposed a public key encryption with keyword search (PEKS) scheme, in which users can search for the required data only by generating the corresponding query trapdoor with their own private keys and without decrypting the ciphertexts. Since the PEKS scheme was proposed, it has attracted extensive interest from researchers. Baek et al. [11] proposed the SCF-PEKS scheme, which removes the secure channel between the server

and each user assumed by Boneh et al. in the PEKS scheme. However, the users of this scheme have to expose their private keys to a third party, leading to security risks. After that, Rhee et al. [12] improved the scheme proposed by Baek et al. by using a more complex public key structure to avoid exposing users' private keys and strengthen security. In the same year, Fang et al. [13,14] implemented a keyword search encryption scheme without a secure channel in the standard model, and subsequently proposed a secure public key encryption with keyword search scheme against KGAs in the standard model. Since the keyword space is much smaller than the key space, however, most of the existing keyword search schemes have potential KGA threats. After the internal server colludes with malicious users to obtain the trapdoor, in the event of the auxiliary server being offline, it can guess the information of the trapdoor and keywords by exhaustively guessing the keywords, which is not expected by honest users. Therefore, in recent years, academic research has mainly focused on how to design an encryption scheme that can resist KGAs. In 2019, Zeng Qi et al. [15] designed a PKE+PEKS scheme that could resist KGAs, which achieved security by generating a secret value and binding this secret value with keywords. In the same year, Wang Shaohui et al. [16] designed an encryption scheme against inside keyword-guessing attacks (IKGAs) by using a non-deterministic algorithm and introducing random numbers into keyword ciphertexts and search trapdoors. In 2020, Du Ruizhong et al. [17] also designed an encryption scheme against IKGAs by introducing random numbers, and combined their scheme with blockchain technology to enhance security. Then, Zhang Yulei et al. [18] designed a certificateless authentication searchable encryption scheme against IKGAs in multiuser environments by combining public key authentication encryption and proxy re-encryption technology. In 2021, Chen Ningjiang et al. [19] designed an encryption scheme against IKGAs by constructing an inverted index structure of ciphertext and embedding the user's private key. Meanwhile, Li Zhiyi et al. [20] designed a dynamic PEKS scheme that could resist KGAs based on the ElGamal encryption algorithm. However, PEKS does not support searching for data encrypted with different public keys in multiuser environments, so it has some limitations. Therefore, Yang et al. [21] first proposed a public key encryption with equality test (PKEET) scheme for multiuser environments. This scheme can use the property of bilinear pairing to test whether two ciphertexts encrypted by different public keys contain the same plaintext without decryption. However, this scheme lacks an authorization mechanism, meaning that anyone could perform an equality test on the ciphertexts, resulting in certain security risks. Subsequent research has focused on providing authorization mechanisms, and researchers have put forward different PKEET schemes [22–25]. Tang [26,27] proposed an FG-PKEET scheme that supports user-level authorization; that is, only when the tester receives the authorization trapdoors of two designated users can they have the authority to perform an equality test for the arbitrary ciphertexts of the two users. Subsequently, Huang et al. [28] proposed a PKE-AET scheme that supports ciphertext-level authorization; that is, only when the tester receives the authorization trapdoors of two designated users can they have the authority to perform an equality test for the designated ciphertexts of the two users. In 2021, Xu Yan et al. [29] put forward an identity-based equality test scheme against IKGAs. In their scheme, the blind signature algorithm is used to convert keywords, and then the ciphertexts of the converted keywords are outsourced to the cloud server. However, the test algorithm in their scheme is slightly high.

1.3. Organizational Structure

The organizational structure of this paper is as follows: Section 1 is the Introduction. In Section 2, we introduce the relevant knowledge used in the paper and the definitions of two types of security threats and security models. Section 3 introduces the concrete construction process of our scheme, including the encryption algorithm, decryption algorithm, authorization algorithm, and test algorithm. In Section 4, we prove the correctness and security of the proposed scheme. Section 5 compares the performance and security of our

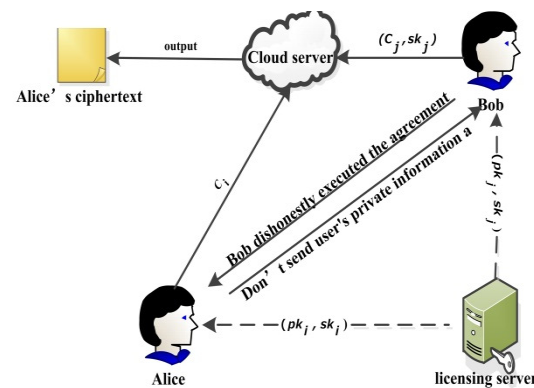


Figure 2. The collusion process between the cloud server and Bob.

2.6. Security Model

The RKGA-CET scheme proposed in this paper considers the following two types of adversaries:

- (1) Type-I adversaries: After obtaining the trapdoor corresponding to the challenge ciphertext, try to guess the plaintext corresponding to the challenge ciphertext.
- (2) Type-II adversaries: Fail to obtain the trapdoor corresponding to the challenge ciphertext, but try to distinguish which plaintext the challenge ciphertext corresponds to.

Game 1 (OW-CCA game): For a Type-I adversary A_1 , the OW-CCA secure model is defined by the following Game 1:

① Initialization phase: The challenger C runs $setup(\lambda)$ to generate the system parameter pp , runs $KeyGen(pp)$ to get k pairs of public and private keys $(pk_i, sk_i) (1 \leq i \leq k)$, and sends pp and all public keys pk_i to the adversary A_1 , whose target user is $U_t (1 \leq t \leq k)$.

② Inquiry phase 1: The adversary A_1 is allowed to make the following four kinds of inquiries to the challenger C in polynomial time:

Hash query: The list is initialized to be empty. The adversary asks the oracle for x (x represents the input of the inquired oracle), and the challenger C randomly selects y (y represents the output of the inquired oracle) and sends y to the adversary.

Private key query: Send the sequence number i . If $i \neq t$, the challenger C will send the private key sk_i corresponding to the public key pk_i to the adversary A_1 .

Decryption query: Send the sequence number i and the ciphertext c_i . The challenger C sends the decryption result $Dec(c_i, sk_i)$ to the adversary A_1 .

Authorization query: Send the sequence number i or (i, C_i) . The challenger C returns the trapdoor td_i or $td_{(i, C_i)}$ to the adversary A_1 .

③ Challenge phase: When the adversary A_1 finishes the inquiries in inquiry phase 1, the challenger C randomly selects a plaintext $Q^* \in F_p^{n \times n}$, runs $Enc(Q^*, pk_t)$ to get the ciphertext C^* , and sends C^* as the challenge ciphertext to the adversary A_1 .

④ Inquiry phase 2: The adversary A_1 is allowed to continue to query the challenger C with four types of queries of inquiry phase 1 in polynomial time, but is not allowed to query the sequence number t to obtain the private key sk_t during the private key query. They are also not allowed to query the sequence number t and the ciphertext C^* to get the corresponding decryption result during the decryption query.

⑤ Guess phase: The adversary A_1 outputs $Q' \in F_p^{n \times n}$. If $Q' = Q^*$, A_1 wins Game 1.

Definition 1. If $Adv_{A_1}^{OW-CCA}(\lambda) = Pr(Q' = Q^*)$ is negligible for all polynomial-time Type-I adversaries, the RKGA-CET scheme is OW-CCA secure under the above model.

Game 2 (IND-CCA game): For a Type-II adversary A_2 , the IND-CCA secure model is defined by the following Game 2:

① Initialization phase: The challenger C performs the same operation as the initialization phase of Game 1.

② Inquiry phase 1: The adversary A_2 is allowed to make four kinds of inquiries of the inquiry phase 1 of Game 1 to the challenger C in polynomial time, and is subject to the same restrictions.

③ Challenge phase: The adversary A_2 selects two pieces of plaintext $Q_1, Q_2 \in F_p^{n \times n}$ with equal length and sends them to C . The challenger C randomly selects $b \in \{0, 1\}$, runs $Enc(Q_b, pk_t)$ to obtain the ciphertext C^* , and sends C^* as the challenge ciphertext to the adversary A_2 .

④ Inquiry phase 2: The adversary A_2 is allowed to continue to query the challenger C with four kinds of queries of inquiry phase 1 in polynomial time, and is subject to the same restrictions. In addition, they are not allowed to query the sequence number t or (t, C_t) to get the corresponding trapdoor during the authorization query.

⑤ Guess phase: The adversary A_2 outputs $b' \in \{0, 1\}$. If $b' = b$, A_2 wins Game 2.

Definition 2. If $Adv_{A_2}^{IND-CCA}(\lambda) = \left| Pr(b' = b) - \frac{1}{2} \right|$ is negligible for all polynomial-time Type-II adversaries, the RKGA-CET scheme is IND-CCA secure under the above model.

3. Encryption and Authorization Scheme

Our scheme can not only realize the basic equality test function, but also solve the hidden security risks of collusion between cloud servers and users in traditional schemes. The new scheme uses discrete logarithms to construct the encryption algorithm, greatly enhancing the security of users' private information. At the same time, it can also prevent untrusted cloud servers and adversaries from colluding to steal the users' private information. The specific algorithm is as follows:

3.1. System Initialization

Input the security parameter λ and return the public parameter pp , where $pp = \{e, G, G_1, G_T, g, g_1, g^\alpha, g_1^\alpha, H_1, H_2, H_3, H_4, H_5\}$. Use bilinear pairing to construct a mapping $e = G \times G_1 \rightarrow G_T$ (G, G_1, G_T are three multiplicative cyclic groups with the prime number p as the order; g and g_1 are generators of G and G_1 , respectively). For any $\alpha \in Z_p^*$, calculate g^α, g_1^α . Hash functions $H_1 : F_p^{n \times n} \rightarrow G_1$, $H_2 : \{0, 1\}^* \rightarrow \{0, 1\}^{l_1}$, $H_3 : G_1 \rightarrow \{0, 1\}^{(n+1)l_1}$, $H_4 : \{0, 1\}^{nl_1} \rightarrow G$, $H_5 : Z_p^* \rightarrow G_1$ (l_1 represents the length of a single element in Z_p).

3.2. Key Generation Algorithm

Input the public parameter pp . Randomly select an irreducible polynomial $f_i(x)$ of degree n and a positive integer a_i ($a_i < p^n - 1$) over F_p , and calculate $g_i(x) = x^{a_i} \bmod f_i(x)$, where x is a pure variable notation. Then, randomly select $x_i, y_i \in Z_p^*$, and calculate $X_i = g_1^{x_i}, Y_i = g_1^{y_i}$. The public key of the user U_i is $pk_i = \{X_i, Y_i, f_i(x), g_i(x), p\}$, and the private key is $sk_i = \{x_i, y_i, a_i\}$.

3.3. Encryption Algorithm

For any plaintext message $M \in [0, p^{n^2-1})$ that needs to be encrypted, it is expressed

in the form of the matrix $Q = \begin{bmatrix} q_{1,1} & q_{1,2} & \cdots & q_{1,n} \\ q_{2,1} & q_{2,2} & \cdots & q_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ q_{n,1} & q_{n,2} & \cdots & q_{n,n} \end{bmatrix}$. For each element in Q , $q_{i,j} \in F_p$,

so $Q \in F_p^{n \times n}$. Given the plaintext matrix $Q_i \in F_p^{n \times n}$, the encryption of U_i is divided into two phases:

Phase 1:

The user U_i randomly selects a positive integer b_i ($b_i < p^n - 1$), and calculates $v_i(x) = x^{b_i} \equiv a_{i,n-1}x^{n-1} + \cdots + a_{i1}x + a_{i0} \bmod f_i(x)$ and $h_i(x) = (g_i(x))^{b_i} = x^{a_i b_i} \equiv a'_{i,n-1}x^{n-1} + \cdots + a'_{i1}x + a'_{i0} \bmod f_i(x)$ over F_p .

- (1) Let B be the companion matrix of $f_i(x)$. According to the isomorphism theorem of finite fields we have $R_i = h_i(B) = \sum_{j=0}^{n-1} a_{ij}' B^j$.
- (2) Let $C_i' = R_i + Q_i$. Then, generate the ciphertext $C_i'' = (v_i, C_i')$, where v_i is an n -dimensional vector composed of coefficients of $v_i(x)$, and C_i' is an $n \times n$ matrix in which every element belongs to F_p , so the ciphertext space after the first encryption is $(F_p^n, F_p^{n \times n})$.

Phase 2:

The user U_i encrypts v_i . First, convert $v_i = (a_{i0}, a_{i1}, \dots, a_{i,n-1})^T$ to $v_i' = a_{i0} \parallel a_{i1} \parallel \dots \parallel a_{i,n-1} \in \{0, 1\}^{n \cdot l_1}$. Then, randomly select $r_{i,1}, r_{i,2} \in Z_p^*$ to generate the ciphertext $C_i = (C_{i,1}, C_{i,2}, C_{i,3}, C_{i,4}, C_{i,5})$ as follows:

$$C_{i,1} = g_1^{r_{i,1}}, C_{i,2} = g^{r_{i,2}}, C_{i,3} = H_3(X_i^{r_{i,1}}) \oplus (v_i' \parallel r_{i,1}),$$

$$C_{i,4} = Y_i^{r_{i,2}} \cdot (H_4(v_i'^{a_i}))^{r_{i,1}}, C_{i,5} = H_2(C_{i,1} \parallel C_{i,2} \parallel C_{i,3} \parallel C_{i,4} \parallel v_i' \parallel r_{i,1}).$$

Finally, output the ciphertext $C_i''' = (C_i, C_i')$.

3.4. Decryption Algorithm

Corresponding to the encryption phases, the decryption of the ciphertext C_i''' is divided into two phases:

Phase 1:

Input the private key sk_i of U_i to process the ciphertext $C_i = (C_{i,1}, C_{i,2}, C_{i,3}, C_{i,4}, C_{i,5})$. Calculate $C_{i,3} \oplus H_3(C_{i,1}^{x_i}) \rightarrow (v_i' \parallel r_{i,1})$. If both $C_{i,1} = g_1^{r_{i,1}}$ and $C_{i,5} = H_2(C_{i,1} \parallel C_{i,2} \parallel C_{i,3} \parallel C_{i,4} \parallel v_i' \parallel r_{i,1})$ hold, return v_i' ; otherwise, return the error symbol \perp .

Phase 2:

First, convert $v_i' = a_{i0} \parallel a_{i1} \parallel \dots \parallel a_{i,n-1}$ to $v_i = (a_{i0}, a_{i1}, \dots, a_{i,n-1})^T$, construct $v_i(x) = a_{i,n-1}x^{n-1} + \dots + a_{i1}x + a_{i0} \bmod f_i(x)$ from the vector v_i , and calculate the polynomial $h_i(x) = (v_i(x))^{a_i} = \sum_{j=0}^{n-1} a_{ij}' x^j \bmod f_i(x)$. Then, use the companion matrix B of $f_i(x)$ to

calculate the matrix $R_i = h_i(B) = \sum_{j=0}^{n-1} a_{ij}' B^j$. Finally, restore the plaintext $Q_i = C_i' - R_i$.

3.5. Authorization Algorithm

According to the security requirements of the users, they can choose two kinds of authorization with different granularities:

- (1) $Aut(sk_i) \rightarrow td_i$: User-level authorization. All of the ciphertexts of the users can be tested. The authorization algorithm is as follows: input the private key sk_i of user U_i , and output a trapdoor $td_i = y_i$;
- (2) $Aut(sk_i, C_i''') \rightarrow td_{i,C_i}$: Ciphertext-level authorization. The ciphertexts designated by the users can be tested. Let the ciphertext to be tested be $C_i''' = (C_i, C_i')$. The authorization algorithm is as follows: input the private key sk_i of user U_i and the designated ciphertext C_i''' , and output a trapdoor $td_{i,C_i} = (y_i, C_{i,2}^{y_i})$.

3.6. Test Algorithm

The equality test algorithm is implemented by the cloud server. After receiving the user's equality test request, the cloud server can test ciphertexts according to the corresponding authorization type provided by the user, and return the result to the user. Suppose that the users who need to perform equality test are U_i and U_j . The cloud server first judges the authorization type according to the obtained trapdoor, and then performs different calculations according to different types of authorization (assuming that the users

disclose the random numbers of encryption phase 2 to the cloud tester; it can be seen in the following security analysis that this does not affect the security of the scheme at all).

- (1) $Test(C_i''', td_i, C_j''', td_j) \rightarrow result$: If the authorization is user-level authorization, the cloud server obtains the trapdoor $td_i = y_i, td_j = y_j$. First, calculate $K_i = \frac{C_{i,4}}{C_{i,2}^{y_i}}$, $K_j = \frac{C_{j,4}}{C_{j,2}^{y_j}}$, and then judge whether $e(K_i, C_{i,1}) = e(K_j, C_{j,1})$ is true. If not, the server outputs 0; otherwise, the server outputs 1. Then, continue to judge whether $e(K_i^{r_{i,1}}, H_1(C_i')) = e(K_j^{r_{j,1}}, H_1(C_j'))$ is true. If so, it means that the plaintext messages corresponding to the two ciphertexts are equal, and the server outputs 1. If not, it indicates that the plaintext messages are not equal, and the server outputs 0.
- (2) $Test(C_i''', td_{i,C_i}, C_j''', td_{j,C_j}) \rightarrow result$: If the authorization is ciphertext-level authorization, the cloud server obtains the trapdoor $td_{i,C_i} = (y_i, C_{i,2}^{y_i}), td_{j,C_j} = (y_j, C_{j,2}^{y_j})$. First, calculate $K_{i,C_i} = \frac{e(C_{i,4}^{r_{i,1}}, H_5(y_i y_j))}{e(C_{i,2}^{y_i}, H_5(y_i y_j))}, K_{j,C_j} = \frac{e(C_{j,4}^{r_{j,1}}, H_5(y_i y_j))}{e(C_{j,2}^{y_j}, H_5(y_i y_j))}$, and then judge whether $K_{i,C_i} = K_{j,C_j}$ is true. If not, the server outputs 0; otherwise, the server outputs 1. Then, continue to judge whether $e(C_{i,2}^{r_{i,2}}, H_1(C_i')) = e(C_{j,2}^{r_{j,2}}, H_1(C_j'))$ is true. If so, it means that the plaintext messages corresponding to the two ciphertexts are equal, and the server outputs 1. If not, it indicates that the plaintext messages are not equal, and the server outputs 0.
- (3) $Test(C_i''', td_i, C_j''', td_{j,C_j}) \rightarrow result$: If one of the authorizations is user-level authorization and the other is ciphertext-level authorization (it may be assumed that the user U_i submits the user-level authorization and the user U_j submits the ciphertext-level authorization), the cloud server obtains the trapdoor $td_i = y_i, td_{j,C_j} = (y_j, C_{j,2}^{y_j})$. First, calculate $K_i = \frac{C_{i,4}}{C_{i,2}^{y_i}}, K_{j,C_j} = \frac{e(C_{j,4}^{r_{j,1}}, H_5(y_i y_j))}{e(C_{j,2}^{y_j}, H_5(y_i y_j))}$, and then judge whether $e(K_i^{r_{i,1}}, H_5(y_i y_j)) = K_{j,C_j}$ is true. If not, the server outputs 0; otherwise, the server outputs 1. Then, continue to judge whether $e(C_{i,2}^{r_{i,2}}, H_1(C_i')) = e(C_{j,2}^{r_{j,2}}, H_1(C_j'))$ is true. If so, it means that the plaintext messages corresponding to the two ciphertexts are equal, and the server outputs 1. If not, it indicates that the plaintext messages are not equal, and the server outputs 0.

4. Scheme Analysis

4.1. Correctness Analysis

The RKGA-CET scheme satisfies the correctness condition, and its correctness is proven as follows:

Proof.

- (1) (Decryption correctness): For any plaintext message $Q_i \in F_p^{n \times n}$, there is $Dec(Enc(Q_i, pk_i), sk_i) = Q_i$.
- (2) (Test consistency): For any $Q_i, Q_j \in F_p^{n \times n}$, if $Q_i = Q_j$ and $td_{j,C_j} \leftarrow Aut(sk_j, \cdot), td_{i,C_i} \leftarrow Aut(sk_i, \cdot)$, then $\Pr[Test(Enc(Q_i, pk_i), td_{i,C_i}, Enc(Q_j, pk_j), td_{j,C_j}) = 1] = 1$.
- (3) (Test reliability): For any $Q_i, Q_j \in F_p^{n \times n}$, if $Q_i \neq Q_j$ and $td_{i,C_i} \leftarrow Aut(sk_i, \cdot), td_{j,C_j} \leftarrow Aut(sk_j, \cdot)$, then $\Pr[Test(Enc(Q_i, pk_i), td_{i,C_i}, Enc(Q_j, pk_j), td_{j,C_j}) = 1] \leq negl(\lambda)$.
- (4) For any ciphertext $C_i = (C_{i,1}, C_{i,2}, C_{i,3}, C_{i,4}, C_{i,5}), v'_i$ can be recovered.

$$\begin{aligned} C_{i,3} \oplus H_3(C_{i,1}^{x_i}) &= H_3(X_i^{r_{i,1}}) \oplus (v'_i \parallel r_{i,1}) \oplus H_3(g_1^{r_{i,1} x_i}) \\ &= H_3(g_1^{x_i r_{i,1}}) \oplus (v'_i \parallel r_{i,1}) \oplus H_3(g_1^{r_{i,1} x_i}) \\ &= v'_i \parallel r_{i,1} \end{aligned}$$

- (5) A logical equivalent expression can be derived from the construction process of the scheme as follows: $R_i = R_j \Leftrightarrow V_i(x)^{a_i} = V_j(x)^{a_j} \Leftrightarrow v_i^{a_i} = v_j^{a_j} \Leftrightarrow v_i'^{a_i} = v_j'^{a_j}$. Given the ciphertext $C_i''' = (C_i, C_i')$, $C_j''' = (C_j, C_j')$, it can be calculated in three cases:

① For a user-level authorization trapdoor $td_i = y_i$, $td_j = y_j$, calculate:

$$\begin{aligned} K_i &= \frac{C_{i,4}}{C_{i,2}^{y_i}} = \frac{Y_i^{r_{i,2}}(H_4(v_i'^{a_i}))^{r_{i,1}}}{g^{r_{i,2}y_i}} = \frac{g^{y_i r_{i,2}}(H_4(v_i'^{a_i}))^{r_{i,1}}}{g^{r_{i,2}y_i}} = (H_4(v_i'^{a_i}))^{r_{i,1}} \\ K_j &= \frac{C_{j,4}}{C_{j,2}^{y_j}} = \frac{Y_j^{r_{j,2}}(H_4(v_j'^{a_j}))^{r_{j,1}}}{g^{r_{j,2}y_j}} = \frac{g^{y_j r_{j,2}}(H_4(v_j'^{a_j}))^{r_{j,1}}}{g^{r_{j,2}y_j}} = (H_4(v_j'^{a_j}))^{r_{j,1}} \\ e(K_i, C_{j,1}) &= ((H_4(v_i'^{a_i}))^{r_{i,1}}, g_1^{r_{j,1}}) = (H_4(v_i'^{a_i}), g_1)^{r_{i,1}r_{j,1}} \\ e(K_j, C_{i,1}) &= ((H_4(v_j'^{a_j}))^{r_{j,1}}, g_1^{r_{i,1}}) = (H_4(v_j'^{a_j}), g_1)^{r_{j,1}r_{i,1}} \end{aligned}$$

If $e(K_i, C_{j,1}) = e(K_j, C_{i,1})$, then $v_i'^{a_i} = v_j'^{a_j} \Rightarrow R_i = R_j$. Continue to calculate:

$$\begin{aligned} e(K_i^{r_{j,1}}, H_1(C_i')) &= e((H_4(v_i'^{a_i}))^{r_{i,1}r_{j,1}}, H_1(C_i')) = e(H_4(v_i'^{a_i}), H_1(C_i'))^{r_{i,1}r_{j,1}} \\ e(K_j^{r_{i,1}}, H_1(C_j')) &= e((H_4(v_j'^{a_j}))^{r_{j,1}r_{i,1}}, H_1(C_j')) = e(H_4(v_j'^{a_j}), H_1(C_j'))^{r_{j,1}r_{i,1}} \end{aligned}$$

If $e(K_i^{r_{j,1}}, H_1(C_i')) = e(K_j^{r_{i,1}}, H_1(C_j'))$, then $C_i' = C_j'$. Therefore, $Q_i = Q_j$ can be obtained from $C_i' - R_i = C_j' - R_j$; that is, $Test(C_i, td_i, C_j, td_j) = 1$ holds.

② For a ciphertext-level authorization trapdoor $td_{i,C_i} = (y_i, C_{i,2}^{y_i})$, $td_{j,C_j} = (y_j, C_{j,2}^{y_j})$, calculate:

$$\begin{aligned} K_{i,C_i} &= \frac{e(C_{i,4}^{r_{j,1}}, H_5(y_i y_j))}{e(C_{i,2}^{y_i}, H_5(y_i y_j))} \\ &= \frac{e(Y_i^{r_{i,2}}(H_4(v_i'^{a_i}))^{r_{i,1}r_{j,1}}, H_5(y_i y_j))}{e(g^{r_{i,2}y_i}, H_5(y_i y_j))} \\ &= \frac{e(g^{y_i r_{i,2}}(H_4(v_i'^{a_i}))^{r_{i,1}r_{j,1}}, H_5(y_i y_j))}{e(g^{r_{i,2}y_i}, H_5(y_i y_j))} \\ &= e(H_4(v_i'^{a_i}), H_5(y_i y_j))^{r_{i,1}r_{j,1}} \\ K_{j,C_j} &= \frac{e(C_{j,4}^{r_{i,1}}, H_5(y_i y_j))}{e(C_{j,2}^{y_j}, H_5(y_i y_j))} \\ &= \frac{e(Y_j^{r_{j,2}}(H_4(v_j'^{a_j}))^{r_{j,1}r_{i,1}}, H_5(y_i y_j))}{e(g^{r_{j,2}y_j}, H_5(y_i y_j))} \\ &= \frac{e(g^{y_j r_{j,2}}(H_4(v_j'^{a_j}))^{r_{j,1}r_{i,1}}, H_5(y_i y_j))}{e(g^{r_{j,2}y_j}, H_5(y_i y_j))} \\ &= e(H_4(v_j'^{a_j}), H_5(y_i y_j))^{r_{j,1}r_{i,1}} \end{aligned}$$

If $K_{i,C_i} = K_{j,C_j}$, then $v_i'^{a_i} = v_j'^{a_j} \Rightarrow R_i = R_j$. Continue to calculate:

$$\begin{aligned} e(C_{i,2}^{r_{j,2}}, H_1(C_i')) &= e(g^{r_{i,2}r_{j,2}}, H_1(C_i')) = e(g, H_1(C_i'))^{r_{i,2}r_{j,2}} \\ e(C_{j,2}^{r_{i,2}}, H_1(C_j')) &= e(g^{r_{j,2}r_{i,2}}, H_1(C_j')) = e(g, H_1(C_j'))^{r_{j,2}r_{i,2}} \end{aligned}$$

If $e(C_{i,2}^{r_{j,2}}, H_1(C_i')) = e(C_{j,2}^{r_{i,2}}, H_1(C_j'))$, then $C_i' = C_j'$. Therefore, $Q_i = Q_j$ can be obtained from $C_i' - R_i = C_j' - R_j$; that is, $Test(C_i, td_{i,C_i}, C_j, td_{j,C_j}) = 1$ holds.

③ For two different types of authorization, where one is user-level authorization and the other is ciphertext-level authorization (let user U_i submit the user-level authorization trapdoor $td_i = y_i$, and user U_j submit the ciphertext-level authorization trapdoor $td_{j,C_j} = (y_j, C_{j,2}^{y_j})$), calculate:

$$K_i = \frac{C_{i,4}}{C_{i,2}^{y_i}} = (H_4(v_i'^{a_i}))^{r_{i,1}},$$

$$K_{j,C_j} = \frac{e(C_{j,4}^{r_{j,1}}, H_5(y_i y_j))}{e(C_{j,2}^{y_j}, H_5(y_i y_j))} = e(H_4(v_j'^{a_j}), H_5(y_i y_j))^{r_{j,1} r_{i,1}},$$

$$e(K_i^{r_{j,1}}, H_5(y_i y_j)) = e((H_4(v_i'^{a_i}))^{r_{i,1} r_{j,1}}, H_5(y_i y_j)) = e(H_4(v_i'^{a_i}), H_5(y_i y_j))^{r_{i,1} r_{j,1}}.$$

If $e(K_i^{r_{j,1}}, H_5(y_i y_j)) = K_{j,C_j}$, then $v_i'^{a_i} = v_j'^{a_j} \Rightarrow R_i = R_j$. Continue to calculate:

$$e(C_{i,2}^{r_{j,2}}, H_1(C_i')) = e(g^{r_{i,2} r_{j,2}}, H_1(C_i')) = e(g, H_1(C_i'))^{r_{i,2} r_{j,2}},$$

$$e(C_{j,2}^{r_{i,2}}, H_1(C_j')) = e(g^{r_{j,2} r_{i,2}}, H_1(C_j')) = e(g, H_1(C_j'))^{r_{j,2} r_{i,2}}.$$

If $e(C_{i,2}^{r_{j,2}}, H_1(C_i')) = e(C_{j,2}^{r_{i,2}}, H_1(C_j'))$, then $C_i' = C_j'$. Therefore, $Q_i = Q_j$ can be obtained from $C_i' - R_i = C_j' - R_j$; that is, $\text{Test}(C_i, td_{i,C_i}, C_j, td_{j,C_j}) = 1$ holds. \square

4.2. Security Analysis

Theorem 1. For probabilistic polynomial-time adversaries, the RKGA-CET scheme proposed in this paper can resist offline keyword-guessing attacks.

Lemma 1. For probabilistic polynomial-time adversaries, the RKGA-CET scheme is OW-CCA secure for user-level authorization and ciphertext-level authorization based on the DLP difficulty problem under the random oracle model.

Proof. Let A_1 be a polynomial-time adversary to break the OW-CCA security. Suppose that A_1 requests at most H_1 hash query for q_{H_1} times, H_2 hash query for q_{H_2} times, H_3 hash query for q_{H_3} times, H_4 hash query for q_{H_4} times, H_5 hash query for q_{H_5} times, private key query for q_K times, decryption query for q_D times, and authorization query (ciphertext-level authorization and user-level authorization) for q_{Aut} times. Note that if the same query is executed multiple times under a random oracle, the same result will be obtained. The game of adversary A and challenger C can be simulated as follows:

(1) System establishment phase:

The adversary A_1 selects a user U_t ($1 \leq t \leq k$) to attack. The challenger C runs $setup(\lambda)$ to get the system parameter pp , runs $KeyGen(pp)$ to get the public key pk_i and private key sk_i ($1 \leq i \leq k$), and sends pp and all public keys pk_i to the adversary A_1 . The challenger C responds to the hash query initiated by A_1 to the random oracle by managing and maintaining the list $H_1 - List, H_2 - List, H_3 - List, H_4 - List, H_5 - List$. The initialization of these lists is empty.

(2) Inquiry phase 1:

① Hash – 1 query (v_1): Given a new $v_1 \in F_p^{n \times n}$, the challenger C randomly selects $h_1 \in G_1$, puts (v_1, h_1) as a new item in the list $H_1 - List$, and outputs h_1 to A_1 as the answer.

② Hash – 2 query (v_2): Given a new $v_2 \in \{0, 1\}^*$, the challenger C randomly selects $h_2 \in \{0, 1\}^l$, puts (v_2, h_2) as a new item in the list $H_2 - List$, and outputs h_2 to A_1 as the answer.

③ *Hash* – 3 query (v_3): Given a new $v_3 \in G_1$, the challenger C randomly selects $h_3 \in \{0, 1\}^{(n+1)l_1}$, puts (v_3, h_3) as a new item in the list $H_3 - List$, and outputs h_3 to A_1 as the answer.

④ *Hash* – 4 query (v_4): Given a new $v_4 \in \{0, 1\}^{nl_1}$, the challenger C randomly selects $h_4 \in G$, puts (v_4, h_4) as a new item in the list $H_4 - List$, and outputs h_4 to A_1 as the answer.

⑤ *Hash* – 5 query (v_5): Given a new $v_5 \in Z_p^*$, the challenger C randomly selects $h_5 \in G_1$, puts (v_5, h_5) as a new item in the list $H_5 - List$, and outputs h_5 to A_1 as the answer.

⑥ Private key query (K): Send the sequence number i . If $i \neq t$, the challenger C sends the private key sk_i corresponding to the public key pk_i to the adversary A_1 .

⑦ Decryption query (D): Send the sequence number i and the ciphertext C_i''' . If $i \neq t$, C runs the algorithm $Dec(C_i''', sk_i)$ to decrypt the ciphertext, and sends the decryption result to A_1 . If $i = t$, C inputs $C_{i,1}^{x_i}$ to request H_3 and outputs a result h_3 . Then, C calculates $C_{i,3} \oplus H_3(C_{i,1}^{x_i})$ to get $v_i' || r_{i,1}$, and verifies whether the following equation is true:

$$C_{i,1} = g_1^{r_{i,1}}, C_{i,5} = H_2(C_{i,1} || C_{i,2} || C_{i,3} || C_{i,4} || v_i' || r_{i,1})$$

If not, C returns \perp to A_1 ; otherwise, C decrypts v_i' to get Q_i and sends it to A_1 .

⑧ Authorization query (*Aut*): For the two kinds of authorization mentioned in this paper:

- When receiving the sequence number i , C returns the trapdoor td_i to A ;
- When receiving the sequence number i and the ciphertext C_i , C returns the trapdoor $td_{(i,C_i)}$ to A_1 .

(3) Challenge phase:

After A_1 finishes the query in query phase 1, C randomly selects a plaintext $Q^* \in F_p^{n \times n}$, encrypts the plaintext through phase 1 of the encryption algorithm to obtain C^{**} , cascades the elements in v^* into $v^{*'}$, and then randomly selects $r_1^*, r_2^* \in Z_p^*$ and calculates:

$$C_1^* = g_1^{r_1^*}, C_2^* = g^{r_2^*}, C_3^* = H_3(x^{r_1^*}) \oplus (v^{*'} || r_1^*), \\ C_4^* = Y^{r_2^*} \cdot (H_4(v^{*'} || r_1^*))^{r_1^*}, C_5^* = H_2(C_1^* || C_2^* || C_3^* || C_4^* || v^{*'} || r_1^*).$$

Finally, return the challenge ciphertext C^* to A_1 , where $C^* = (C_1^*, C_2^*, C_3^*, C_4^*, C_5^*)$, and the public key is pk_t .

(4) Inquiry phase 2:

The adversary A_1 makes the same inquiry as in inquiry phase 1, but they are not allowed to inquire the sequence number t to obtain the private key sk_t in the private key query phase, and they are not allowed to inquire the sequence number t and the ciphertext C^* to obtain the corresponding decryption result in the decryption query phase.

(5) Guess phase:

A_1 outputs $Q' \in F_p^{n \times n}$. If $Q' = Q^*$, A_1 guesses successfully.

A_1 's advantage of winning the game is $Adv_{A, RKGA-CET}^{OW-CCA}(\lambda) = \Pr(Q' = Q^*) = \Pr(R' = R^*)$. Now we prove that $\Pr(R' = R^*) \leq \text{negl}(\lambda)$.

The matrix R is pseudo-random because it is generated by the public key pk_t and the random number b ; that is, it is generated by solving the polynomial $h(x) = g^b(x) = v^a(x) = x^{ab}$. To calculate $h(x) = x^{ab} \bmod f(x)$, however, $g(x) = x^a \bmod f(x)$ and $v(x) = x^b \bmod f(x)$ must be calculated. This problem is the *Diffie – Hellman* problem over the polynomial finite field $F_p[x]/\langle f(x) \rangle$, and the *Diffie – Hellman* problem can be reduced to the *DLP* problem over the finite field in polynomial time. However, the *DLP* problem is computationally infeasible. This contradicts the original hypothesis. \square

Lemma 2. For probabilistic polynomial-time adversaries, the RKGA-CET scheme is IND-CCA secure for user-level authorization and ciphertext-level authorization based on the DLP difficulty problem under the random oracle model.

Proof. Let A_2 be a polynomial-time adversary to break the IND-CCA security. The game simulation of the adversary A_2 and the challenger C is the same as above, but the adversary is not allowed to query the sequence number t or (t, C_t) to obtain the corresponding trapdoor in the inquiry phase 2. The adversary selects two pieces of plaintext $Q_1, Q_2 \in F_p^{n \times n}$ with equal length and sends them to the challenger. The challenger randomly selects $b \in \{0, 1\}$, runs $Enc(Q_b, pk_i)$ to get the ciphertext C^* , and sends C^* to the adversary as the challenge ciphertext. The adversary outputs $b' \in \{0, 1\}$. If $b' = b$, the adversary breaks the IND-CCA security.

Let S_1 represent the event that the adversary can correctly output $b' = b$. The advantage of the adversary is:

$$Adv_{A, RKGA-CEM}^{IND-CCA}(\lambda) = \left| \Pr[S_1] - \frac{1}{2} \right| \quad (1)$$

Let S_2 represent the event that the adversary breaks the hash function H_3 with advantage ε_1 . Then, the adversary can calculate $C_{i,3} \oplus H_3(C_{i,1}^{x_i})$ to get $v'_i || r_{i,1}$. According to the forking lemma, there is:

$$|\Pr[S_1] - \Pr[S_2]| \leq \varepsilon_1 \quad (2)$$

Let S_3 represent the event that the adversary obtains the private key a_i with advantage ε_2 , and then recovers the plaintext by calculating $(V_i(x))^{a_i} = h_i(x) \rightarrow R_i = h_i(B) \rightarrow Q_i = C'_i - R_i$. According to the forking lemma, there is:

$$|\Pr[S_2] - \Pr[S_3]| \leq \varepsilon_2 \quad (3)$$

Since a_i is a random element over the finite field F_p , we have $\Pr[S_3] = \frac{1}{2}$.

Combining (1)–(3), we can get:

$$\begin{aligned} Adv_{A, RKGA-CEM}^{IND-CCA}(\lambda) &= \left| \Pr[S_1] - \frac{1}{2} \right| \\ &= |\Pr[S_1] - \Pr[S_3]| \\ &\leq |\Pr[S_1] - \Pr[S_2]| + |\Pr[S_2] - \Pr[S_3]| \\ &\leq \varepsilon_1 + \varepsilon_2 \end{aligned}$$

According to the hypothesis of the random oracle and the difficulty of the DLP problem, $\varepsilon_1, \varepsilon_2$ must be negligible. Hence, $Adv_{A, RKGA-CEM}^{IND-CCA}(\lambda)$ must be negligible. \square

By Lemmas 1 and 2, Theorem 1 is proven.

5. Performance Analysis

We compared the computational overhead and security of the RKGA-CET scheme with those proposed in [5–7,15,29], and the results are shown in Table 1. Since the system establishment algorithm is executed only once, its computational overhead can be negligible. The performance comparison mainly focuses on the bilinear operation e , the modular exponentiation operation m , and the hash operation h , which are the most resource-consuming in the practical operation. The security comparison is to verify whether the scheme has the ability to resist KGA attacks.

Analysis of Table 1 shows that in the encryption phase, the five schemes compared all contain bilinear operations, while the RKGA-CET scheme does not need it. The number of modular exponentiation operations performed by RKGA-CET is more than that in [6,15,29], but less than that in [5,7]. The number of hash operations of the six schemes is essentially the same. Overall, RKGA-CET only needs 11 operations, which is essentially the same as the number of operations in [29], and less than in the other four schemes. In the decryption phase, RKGA-CET has the same number of operations as the schemes in [7,15], and has a lower decryption cost compared with [29], but slightly higher than those in [5,6]. In the test phase, since RKGA-CET performs different equality tests according to different

authorization types, we choose the largest test cost as the test cost of RKGA-CET—that is, the test cost when the authorization type is ciphertext-level authorization. It can be seen that the test cost of RKGA-CET is significantly higher than that of the other five schemes. This is because we first use the isomorphism of finite fields to transform keywords, so that the adversary cannot use the weakness of the relatively small information entropy in the keyword space to carry out KGA attacks, and then outsource the ciphertexts of the converted keywords to the cloud server. While enhancing security, RKGA-CET also increases the test cost, so it is suitable for occasions that require high safety but low testing cost, such as medical systems that are related to the safety of patients' lives. However, it should also be noted that when the authorization type is user-level authorization, the test cost of RKGA-CET is $4m + 2h + 4e$, which is close to the scheme in [5] and lower than that of [29]. Although the test cost of the schemes in [6,7] is better than that of RKGA-CET, none of them can resist KGA attacks in terms of security. Considering the encryption process, decryption process, and test process as a whole, the number of bilinear operations of RKGA-CET is the least, which is $6e$. The modular exponentiation times of RKGA-CET are $10m$ more than those of the schemes in [6,15], $5m$ more than that of the scheme in [29], and essentially equal to those of the schemes in [5,7]. The number of hash operations of the six schemes is generally similar. According to the experimental results in [8], a modular exponentiation operation takes 0.01 ms, a hash operation takes 0.19 ms, and a bilinear operation takes about 1.5 ms. Therefore, in general, the efficiency of RKGA-CET is higher than that of the schemes proposed in the other five references with more bilinear operations.

Table 1. Computational overhead and security comparison.

Scheme	Encryption Cost	Decryption Cost	Test Cost	Resist KGAs
Scheme in [5]	$9m + 3e + 3h$	$2m$	$5m + 4e + h$	Yes
Scheme in [6]	$4m + 6e + 2h$	h	$e + 2m + h$	No
Scheme in [7]	$13m + 10e + 4h$	$2m + e$	$3m + 2e$	No
Scheme in [15]	$4m + 8e + 5h$	$e + 2h$	$2m + e$	Yes
Scheme in [29]	$6m + 2e + 2h$	$m + 2e + 2h$	$4m + 6e + 4h$	Yes
RKGA-CET	$8m + 3h$	$2m + h$	$6m + 6e + 3h$	Yes

In order to visually show the comparison results, we conducted simulation experiments of the encryption algorithm and the test algorithm based on the PBC library. The experiment used a 64-bit Windows operating system, the processor was an Intel (R) Core (TM) i5-9500 CPU @ 3.00 GHz, and the running memory was 8 GB. We chose a type A elliptic curve, whose equation is $y^2 = x^3 + x$ and embedding degree is 2. The order q is a prime number of 512 bit. We selected SHA-1 as the hash function, and the bit length of the hash value was 160 bit. The specific experimental results are shown in Figures 3 and 4.

The simulation results show that the RKGA-CET scheme has obvious advantages in the overhead of the encryption algorithm, but the overhead of the test algorithm is slightly higher, which is acceptable because RKGA-CET reduces the encryption burden of the server while enhancing security. In summary, RKGA-CET has certain advantages in terms of computational overhead and security.

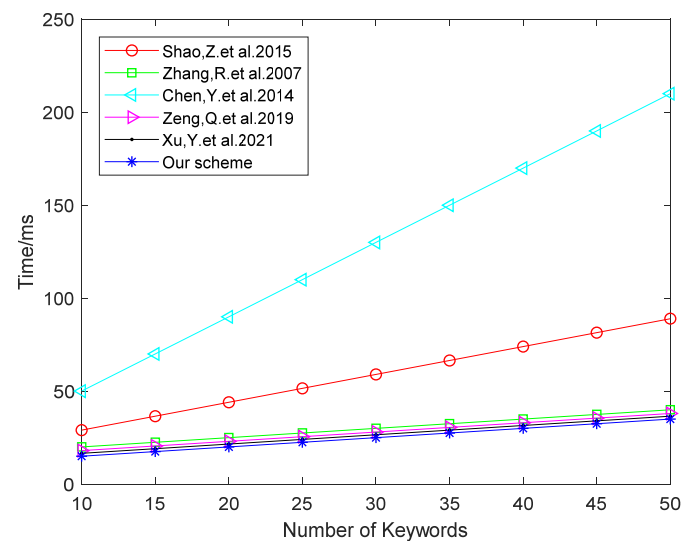


Figure 3. Time of encryption algorithm [5–7,15,29].

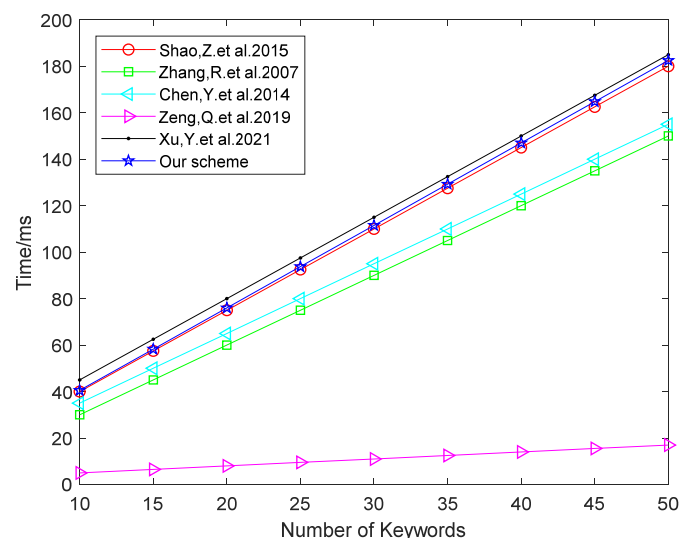


Figure 4. Time of test algorithm [5–7,15,29].

6. Application in Medical System

In the “Introduction”, we introduced the idea that the equality test technology can be applied to the medical system to protect patients’ historical cases. In this section, the technology designed by our team is applied to the medical system. It is still assumed that *Alice* and *Bob* are patients with the same symptoms who belong to the same medical institution. Firstly, the key-generation center (KGC) initializes the system parameters, and generates a public–private key pair for each patient according to the key-generation algorithm. Secondly, each patient encrypts their own medical record with the public key according to the encryption algorithm, and then uploads the encrypted medical record to the medical record management system. Patients choose different granularity authorization according to the authorization algorithm, and then the medical institution performs equality tests on the ciphertexts according to the type of authorization. Figure 5 shows the specific application model of the RKGA-CET scheme, including four entities: a trusted KGC, patients, a medical record management system, and a medical institution.

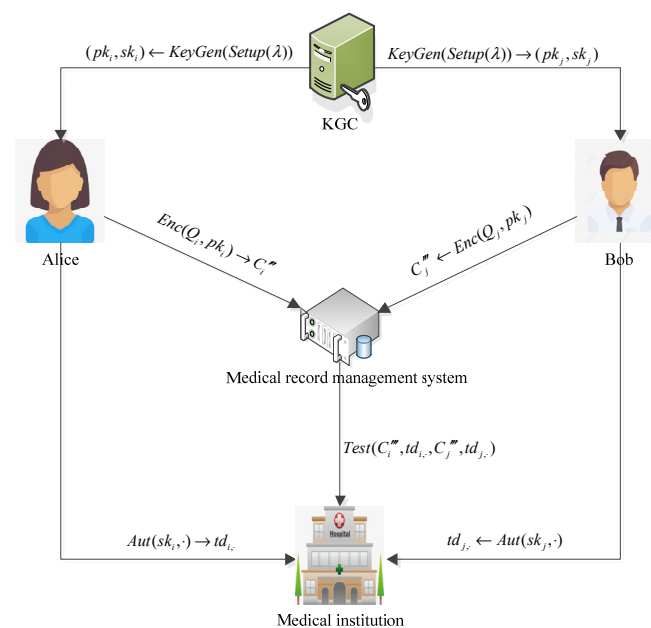


Figure 5. The process of equality testing.

7. Conclusions

With the rapid development and wide application of cloud computing and 5G communication, the research of public key encryption with equality test algorithms has become a hot topic. Based on the existing public key encryption with equality test schemes, RKGA-CET is proposed in this paper, where we first use the property of finite-field isomorphism to transform the keyword, and then encrypt the converted keyword vector. The function of the first encryption is that the adversary can only test the encrypted keyword, so that the adversary can only guess the encrypted keyword space or directly crack the first encrypted key, which will not reveal any keyword space information, so the adversary cannot use the weakness of the relatively small information entropy of the keyword space to perform KGAs. Meanwhile, the property of isomorphism of finite fields ensures that cracking the first encrypted key is equivalent to solving the discrete logarithm problem over a finite field. Security analysis shows that the RKGA-CET scheme achieves both OW-CCA security and IND-CCA security, thus resisting the offline KGAs caused by the collusion between the cloud server and the dishonest users. Furthermore, while ensuring data confidentiality, RKGA-CET also supports two different authorization modes. Finally, performance analysis shows that RKGA-CET is practical in terms of computational overhead and security. This paper only makes theoretical innovation on the basis of previous research. Later, we will further study how to reduce the overhead of the test algorithm while enhancing security, and design encryption schemes with better properties in combination with the needs of practical applications, so as to effectively strengthen the security and privacy of user data and promote the development of cloud computing.

Author Contributions: Conceptualization, P.Z. and J.L.; methodology, P.Z. and J.L.; software, J.L.; validation, P.Z. and J.L.; formal analysis, J.L.; investigation, J.L. and Z.F.; resources, P.Z.; data curation, Z.F.; writing—original draft preparation, J.L.; writing—review and editing, P.Z.; visualization, P.Z.; funding acquisition, P.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Science and Technology Project of Henan Educational Department, grant number 20A520012.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors acknowledge Alexander Wang from the Faculty of Engineering, Built Environment, and Information Technology, SEGi University, Malaysia, for his support and assistance with this paper.

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

References

1. Bao, G.; Wang, S.; Li, Y. Research on data security protection method based on privacy perception in cloud computing. *Netinfo Secur.* **2017**, *17*, 84–89.
2. Wang, Y.; Zhang, J.; Cheng, S.; Xiaohui, T. An improved design of homomorphic encryption for Cloud Computing. *Netinfo Secur.* **2017**, *17*, 21–26.
3. Li, Z.; Hu, X. Research on data security and privacy protection in the background of big data. *Internet Things* **2020**, *10*, 76–78.
4. Wu, H. Research on privacy data security for Internet of things. *Wirel. Internet Technol.* **2020**, *17*, 21–22.
5. Shao, Z.; Yang, B. On security against the server indesignated tester public key encryption with keyword search. *Inf. Process. Lett.* **2015**, *115*, 957–961. [\[CrossRef\]](#)
6. Zhang, R.; Imai, H. Generic combination of public key encryption with keyword search and public key encryption. In Proceedings of the International Conference on Cryptology and Network Security, Singapore, 8–10 December 2007; pp. 159–174.
7. Chen, Y.; Zhang, J.; Lin, D. Generic construction of integrated PHE and PEKS. *J. Des. Codes Cryptogr.* **2014**, *78*, 493–526. [\[CrossRef\]](#)
8. Yang, N.; Zhou, Q.; Xu, S. Searchable encryption scheme with unpaired public key authentication. *J. Comput. Res. Dev.* **2020**, *57*, 2125–2135.
9. Ming, Y.; Wang, E. Identity-Based Encryption with Filtered Equality Test for Smart City Applications. *J. Sens.* **2019**, *19*, 3046. [\[CrossRef\]](#)
10. Dan, B.; Crescenzo, G.D.; Ostrovsky, R.; Persiano, G. Public key encryption with keyword search. In Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, 2–6 May 2004.
11. Baek, J.; Safavi-Naini, R.; Susilo, W. Public key encryption with keyword search revisited. In Proceedings of the International Conference on Computational Science and Its Applications, Perugia, Italy, 30 June–3 July 2008; Springer: Berlin/Heidelberg, Germany, 2008.
12. Rhee, H.S.; Park, J.H.; Susilo, W.; Lee, D.H. Improved searchable public key encryption with designated tester. In Proceedings of the 4th International Symposium on Information, Computer, and Communications Security, Sydney, Australia, 10–12 March 2009.
13. Fang, L.; Susilow, W.; Ge, C.; Wang, J. A secure channel free public key encryption with keyword search scheme without random oracle. In Proceedings of the International Conference on Cryptology and Network Security, Kanazawa, Japan, 12–14 December 2009; Springer: Berlin/Heidelberg, Germany, 2009.
14. Fang, L.; Susilow, W.; Ge, C.; Wang, J. Public key encryption with keyword search secure against keyword guessing attacks without random oracle. *Inf. Sci.* **2013**, *238*, 221–241. [\[CrossRef\]](#)
15. Zeng, Q.; Han, X.; Cao, Y. An encryption scheme combining public key encryption and keyword searchable encryption. *Comput. Mod.* **2019**, 103–107. [\[CrossRef\]](#)
16. Wang, S.H.; Zhang, Y.X.; Wang, H.Q.; Xiao, F.; Wang, R.C. Efficient public key searchable encryption scheme against internal keyword guessing attack. *Comput. Sci.* **2019**, *46*, 91–95.
17. Du, R.; Tan, A.; Tian, J. Blockchain-based public key searchable encryption scheme. *J. Commun.* **2020**, *41*, 114–122.
18. Zhang, Y.; Wen, L.; Wang, H. Searchable encryption scheme without certificate authentication in multi-user environment. *J. Electron. Inf. Technol.* **2020**, *42*, 1094–1101.
19. Chen, N.; Liu, C.; Huang, R. Fast public key searchable encryption scheme against internal keyword guessing attack in cloud environment. *J. Electron. Inf. Technol.* **2021**, *43*, 467–474.
20. Li, Z.; Wang, X.; Wang, Z. A dynamic asymmetric searchable encryption scheme resistant to keyword guessing attack. *Comput. Mod.* **2021**, 100–106. [\[CrossRef\]](#)
21. Yang, G.; Tan, C.H.; Huang, Q.; Wong, D.S. Probabilistic public key encryption with equality test. In Proceedings of the Cryptographers’ Track at the RSA Conference, San Francisco, CA, USA, 1–5 March 2010; Springer: Berlin/Heidelberg, Germany, 2010.
22. Ma, S.; Huang, Q.; Zhang, M.; Yang, B. Efficient public key encryption with equality test supporting flexible authorization. *IEEE Trans. Inf. Forensics Secur.* **2015**, *10*, 458–470. [\[CrossRef\]](#)
23. Lin, X.J.; Qu, H.; Zhang, X. Public Key Encryption Supporting Equality Test and Flexible Authorization without Bilinear Pairings. *IACR Cryptol. Eprint Arch.* **2016**, *2016*, 277. [\[CrossRef\]](#)
24. Lee, H.T.; Ling, S.; Seo, J.H.; Wang, H. CCA2 Attack and Modification of Huang et al.’s Public Key Encryption with Authorized Equality Test. *Comput. J.* **2016**, *59*, 1689–1694. [\[CrossRef\]](#)
25. Wu, L.; Zhang, Y.; Choo, K.R.; He, D. Efficient Identity-based Encryption Scheme with Equality Test for Smart City. *IEEE Trans. Sustain. Comput.* **2018**, *3*, 44–55. [\[CrossRef\]](#)
26. Tang, Q. Towards Public Key Encryption Scheme Supporting Equality Test with Fine-grained Authorization. In Proceedings of the 16th Australasian Conference on Information Security and Privacy, Melbourne, Australia, 11–13 July 2011; Springer: Berlin/Heidelberg, Germany, 2011.

27. Tang, Q. Public Key Encryption Supporting Plaintext Equality Text and User-specified Authorization. *Secur. Commun. Netw.* **2012**, *5*, 1351–1362. [[CrossRef](#)]
28. Huang, K.; Tso, R.; Chen, Y.C.; Rahman, S.; Almogren, A.; Alamri, A. PKE-AET: Public key Encryption with Authorized Equality Test. *Comput. J.* **2015**, *58*, 2686–2697. [[CrossRef](#)]
29. Xu, Y.; Wang, M.; ZHong, H.; ZHong, S. IBEEET-AOK: ID-based encryption with equality test against off-line KGAs for cloud medical services. *Front. Comput. Sci.* **2021**, *15*, 178–180. [[CrossRef](#)]
30. Hu, L.; Li, X.; Lu, L. An identity-based cryptosystem and elliptic curve Tate pair. *Inf. Netw. Secur.* **2005**, 64–66. [[CrossRef](#)]
31. Roger, A.H.; Charles, R.J. *Matrix Analysis*, 2nd ed.; China Machine Press: Beijing, China, 2014; Volume 9, pp. 86–92.
32. Li, K.; Liu, H. On the trace of k elements in a finite field. *Pure Math. Appl. Math.* **2019**, *35*, 394–407.