Received May 27, 2017, accepted June 6, 2017, date of publication June 19, 2017, date of current version July 24, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2716971

Verifiable Public Key Encryption Scheme With Equality Test in 5G Networks

YAN XU[1], MING WANG[1], HONG ZHONG[1], JIE CUI[1], LU LIU[2], AND VIRGINIA N. L. FRANQUEIRA[2]

[1]School of Computer Science and Technology, Anhui University, Hefei 230601, China

[2]Department of Computing and Mathematics, University of Derby, Derby DE22 1GB, U.K.

Corresponding author: Hong Zhong (zhongh@ahu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61572001 and Grant 61502008, in part by The Natural Science Foundation of Anhui Province under Grant 1508085QF132 and Grant 1708085QF136, and in part by the Doctorial Research Start-Up Foundation of Anhui University.

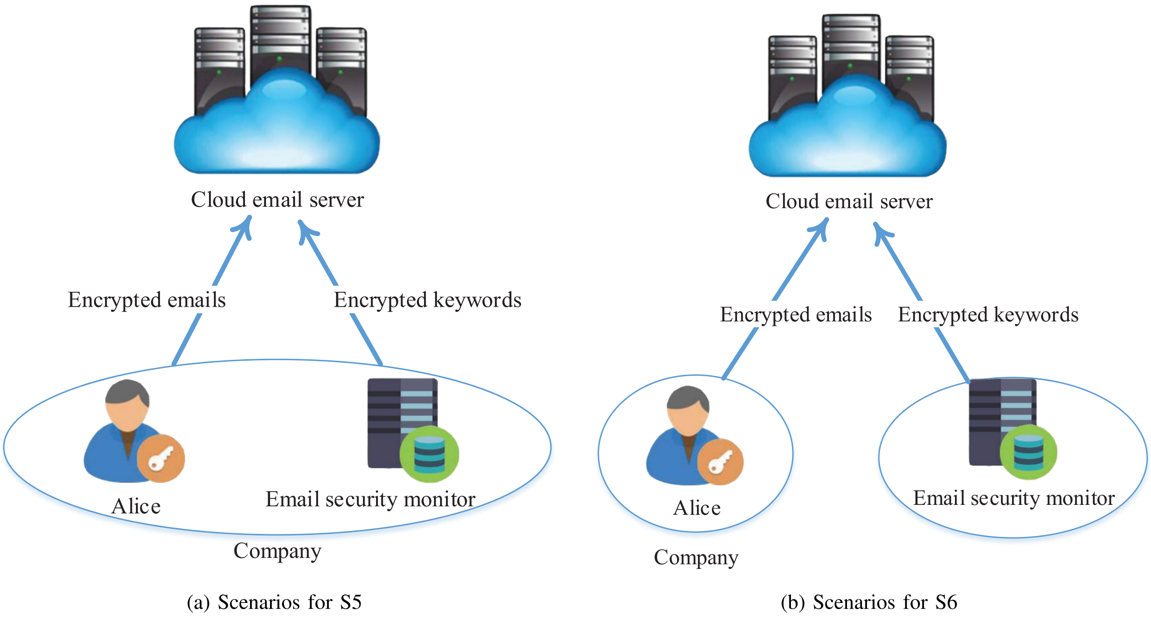
ABSTRACT The emergence of 5G networks will allow cloud computing providers to offer more convenient services. However, security and privacy issues of cloud services in 5G networks represent huge challenges. Recently, to improve security and privacy, a novel primitive was proposed by Ma *et al.* in TIFS 2015, called public key encryption with equality test supporting flexible authorization (PKEET-FA). However, the PKEET scheme lacks verification for equality test results to check whether the cloud performed honestly. In this paper, we expand the study of PKEET-FA and propose a verifiable PKEET (V-PKEET) scheme, which, to the best of our knowledge, is the first work that achieves verification in PKEET. Moreover, V-PKEET has been designed for three types of authorization to dynamically protect the privacy of data owners. Therefore, it further strengthens security and privacy in 5G networks.

INDEX TERMS Public key encryption, equality test, privacy-preserving, verification, cloud computing, 5G networks.

I. INTRODUCTION

Empowered by new Radio Access Technologies (RATs), 5G wireless access solutions are designed to support a variety of applications. This will enable a significant growth in mobile use of cloud-based services, and an upwards trend in outsourcing of privacy sensitive data to cloud servers. 5G networks exacerbate the privacy problems intrinsic to Cloud Computing. For example, cloud storage can reduce the client storage burden but, in order to prevent the cloud from modifying the stored data, its integrity should always be verifiable [1], [2]. For the purpose of security and privacy-preservation, this data also needs to be encrypted before it is stored in cloud servers. For this reason, a large number of researchers have lately concentrated in supporting operations on encrypted data, such as, searchable encryption [3]–[5] and outsourced computing [6], [7]. In this paper, we focus on public key encryption with equality test (PKEET), which was firstly proposed by Yang *et al.* [8] in CT-RSA 2010. PKEET is used to test whether two messages encrypted by using (possibly) two different public keys are identical. In such application of PKEET scheme as outsourced database, once receiving the trapdoors from users, the database manager can then partition the database in accordance with the encrypted messages without the need to interact with data users. PKEET has also various applications in Cloud Computing. For example, the property of equality test can be used to search ciphertexts and classify encrypted messages.

The paper uses as example an email system able to classify all emails by encrypted keywords generated by an email security monitor. Therefore, the monitor is in charge of producing encrypted sensitive keywords, and the cloud email server uses PKEET to search if such sensitive keywords are involved in the mails or not, without knowledge of the emailsą́ content. Suppose that Alice encrypts emails using her public key pk_A, encryption will conceal content, and prevent some crucial information from being searched. To increase search precision, the encrypted emails are attached to a small number of keywords encrypted in PKEET system, and a single ciphertext expressed as C=Enc$(M, pk_A)||$PKEET$(W_1, pk_A)||$ PKEET$(W_2, pk_A)||...||$PKEET(W_n, pk_A). Finally, Alice transfers the cipertext C to the cloud email server for further classification. The key point of the PKEET scheme is that the cloud email server can check if the encrypted keywords are attached to an email including some given keywords generated by the email security monitor.



(a) Scenarios for S5

(b) Scenarios for S6

FIGURE 1. Application scenarios for V-PKEET. (a) Scenarios for S5. (b) Scenarios for S6.

However, Yang *et al.* [8] showed that anyone has the ability to test if two ciphertexts are encrypted under the same plaintext without any authorization. In other words, the attacker can choose an user as target and generate a ciphertext using its public key, however, due to the function of equality test so that the attacker can guess the plaintext under the ciphertext. Therefore, it is essential to implement an authorization strategy to protect data ownersąŕ privacy. Only after the users send delegations to the cloud, can the cloud provider perform the equality test. Furthermore, different applications demand different delegation strategies to dynamically protect the privacy of data owners. Recently, Ma *et al.* [9] designed a PKEET-FA scheme without verification for equality test results. However, in practice, the cloud may be malicious and, therefore, may compute and deliver to users a wrong result. Furthermore, the PKEET-FA scheme supports four scenarios S1-S4 with different delegation granularity levels. There are, however, two other scenarios S5-S6 with different levels of delegation which are not supported. One is receiver-specific level authorization(S5), i.e. all ciphertexts of Alice can only make a comparison with all ciphertexts of a specific receiver (e.g. Bob), but could not make a comparison with any ciphertext of anyone other than Bob. The other one is receiver-specific ciphertext-to-reveiver level authorization(S6), i.e a specific ciphertext of Alice could only make a comparison with all ciphertexts of a specific receiver such as Bob, but could not make a comparison with any ciphertext of any receiver other than Bob.

S5. Assume that the receiver (Alice) and the email security monitor work in the same company. By obtaining the delegations from Alice and the email security monitor, respectively, the cloud email server has the ability to test whether the emails encrypted by Aliceąŕs public key include the keywords encrypted under the email security monitorąŕs public key. The key is that Alice intends to affirm that the equality test is surely performed on her ciphertexts and the specified monitorąŕs ciphertexts (see Fig.1(a)). In other words, Alice and a specific entity produce a trapdoor each to perform equality test in regard to all their ciphertexts.

S6. Assume that the email security monitor belongs to a public institution, different than the company where Alice works, and the institution works in a pay-per-keyword manner. The cloud email server only has the ability to test whether the specific email encrypted by Aliceąŕs public key is contained in keywords encrypted under the specific monitorąŕs public key (see Fig.1(b)). In other words, Alice produces a trapdoor to perform the equality test on her specific ciphertext while the monitor produces a trapdoor to perform the equality test on all of its ciphertexts.

A. OUR CONTRIBUTION

The proposed scheme supports three types of authorization(See Fig.2) simultaneously, which are described in detail in the following. Type-1 authorization is the same as that in [9], however, in our construction, this type is more efficient. Type-2 authorization requires no proxies other than that in FG-PKEET [10] and FG-PKEET+ [11]. Type-3 authorization is novel and has not been implemented in earlier works.

1) Type-1
Receiver level authorization: All ciphertexts of Alice (receiver) can make a comparison with all ciphertexts of any other receiver. (This type represents a coarse-grained delegation.)

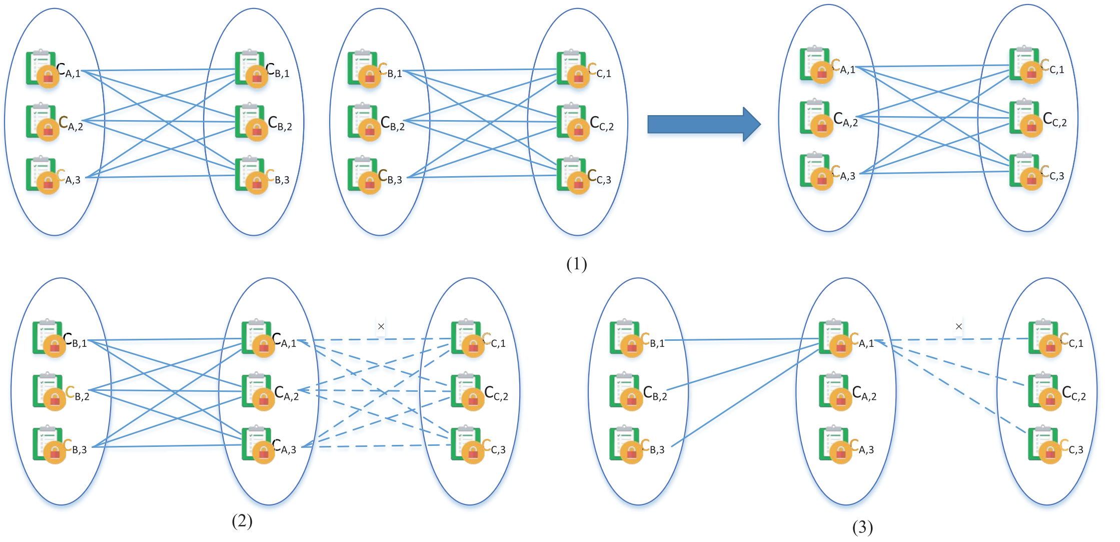


FIGURE 2. Three types of authorizations of V-PKEET.(1) Receiver level authorization (S1);(2) Receiver-specific level authorization (S5);(3) Receiver-specific ciphertext-to-reveiver (S6).

2) Type-2
Receiver-specific level authorization: All ciphertexts of Alice can only make a comparison with all ciphertexts of a specific receiver (e.g. Bob). However, it could not make a comparison with ciphertext of anyone other than Bob. (This type represents a fine-grained delegation)
3) Type-3
Receiver-specific ciphertext-to-receiver level authorization(or receiver-specific receiver-to-ciphertext level authorization): A specific ciphertext of Alice could only make a comparison with all ciphertexts of a specific receiver. such as Bob. However, it could not make a comparison with any ciphertext of any receiver other than Bob (or vice versa).

Furthermore, we propose the concept of enhanced PKEET, called verifiable PKEET (V-PKEET). It supports verification of results from an untrusted cloud server, in other words, we consider the cloud is malicious, i.e., that could compute a wrong result to users. In our system, the user checks whether the cloud server has devotedly performed the authorized equality test, which, to the best of our knowledge, has not been researched previously. Finally, the proposed scheme completes a satisfying security.

B. RELATED WORKS

1) PUBLIC KEY ENCRYPTION WITH KEYWORD SEARCH(PEKS)

The primitive of public key encryption with keyword search was originally introduced by Boneh *et al.* [12]. In this research, a sender encrypts plaintexts using the receiverąŕs public key enabling the receiver to search over the ciphertexts in the asymmetric encryption system. The receiver generates a tag to check if the keyword embedded in the token is equal to the message underlying the ciphertext. Bellare *et al.* [13] showed an efficient searchable scheme based on a deterministic encryption algorithm to effectively complete retrieval of encrypted data. Boneh and Waters [14] presented a new searchable public key encryption(PKE) scheme which provided the functions of comparison search, ordinary subset search and connectives search. Later Rhee *et al.* [15] proposed a secure searchable scheme, called dPEKS, which enhanced the security model of PEKS by taking advantage of a serverąŕs public key in the encryption process. Before long, Yau *et al.* [16] used the technique of proxy re-encryption to achieve bidirectional Re-dPEKS and Guo *et al.* [17] enhanced the scheme to be more efficient. These PKE schemes could produce a token associated with the keyword, and one would be capable of classifying the ciphertexts in accordance with the plaintexts, taking advantage of the token. Nevertheless, this approach has a weakness: a same message encrypted using different public keys cannot be divided into one category. To address this problem, we concentrate on studying PKEET.

2) PUBLIC KEY ENCRYPTION WITH EQUALITY TEST (PKEET)

The concept of public key encryption with equality test (PKEET) was firstly introduced by Yang *et al.* [8]. The study offers a new function to test whether two ciphertexts that could be generated under different public keys are the same plaintext. However, the scheme has no delegation mechanism for data owners to assign someone to test. To realize authorization on PKEET, Tang [10] proposed a novel scheme which provided fine-grained authorization strategy (FG-PKEET) for users. However, in this scheme, the two users need to interact to produce a random value. Besides, Tang [18] came up with a new primitive, called AoN-PKEET, to complete a coarse-grained delegation. Furthermore, Tang [11] applied FG-PKEET to a two-proxy system in which two agent servers needed to work together to achieve an equality test (FG-PKEET+). However, the proposal requires a protocol to interact between two agent servers, and this is too costly. In this paper, we improve the FG-PKEET+ and present the Type-2 authorization, where a proxy server is not required to participate such that it can reduce the communication cost. Huang *et al.* [19] introduced a new proposal, called PKE-AET, which can achieve the equality test between authorized ciphertexts and users through warrants. Lee *et al.* [20] identified a serious flaw and modified the PKE-AET so as to achieve the IND-CCA2 secure. Recently, Ma *et al.* [9] has proposed a scheme that supports various levels of authorizations simultaneously, called PKEET-FA. All the above-mentioned approaches lack verification for the result of equality test. This means that receivers cannot confirm whether the cloud server has faithfully performed the protocol. However, if a malicious cloud server deceives the user for the purpose of self-interest, the equality test is meaningless. To innovate it, it is imperative to verify the return result of equality test from the malicious cloud.

3) VERIFIABLE COMPUTATION

Verifiable computation plays a particularly important role in outsourced computing, especially in a malicious cloud environment. Generally speaking, verifiable computation protocols [21]–[24] enable a client with weak computation ability to outsource the processed data to the cloud. The cloud then returns corresponding results, as well as proofs that the data has been processed correctly. Gennaro *et al.* proposed the concept of non-interactive verifiable computation in CRYPTO 2010 [21]. In this case, the client can verify the outsourced result through a proof. Shortly afterwards, Benabbas *et al.* [22] conceived the first practical verifiable computation scheme for large datasets. Fiore and Gennaro [23] made use of a pseudo-random function with closed-form efficiency and designed two novel constructions for publicly verification: one was evaluation of large polynomials and another was matrix multiplication. Parno *et al.* [24] combined the concepts of verifiable calculation and attributed-based encryption to complete public delegation and public verification. Recently, Zheng *et al.* [25] presented a verifiable authorized set intersection scheme over ciphertexts. They expanded the single-accumulator scheme [26], [27] to test whether members belong to the set. In our paper, we consider the malicious cloud may compute the inaccurate result to users and use variant of the technique in [25] to verify its authenticity.

C. ORGANIZATION

In the next part of the paper, we introduce the preliminaries. Then we give the system model and introduce its formal definition and threat models in Section 3. Furthermore, a construction of V-PKEET is presented in Section 4, and security analyses are showed in the next part. Section 6 presents performance analyses. Finally, we conclude this paper in Section 7.

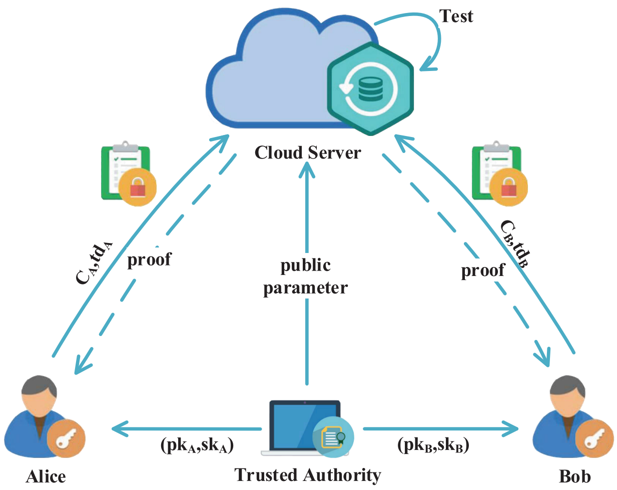


FIGURE 3. System Model of V-PKEET.

II. PRELIMINARIES

A. BILINEAR MAP

Let G, G_1 and G_T be three cyclic groups of prime order p. We denote that g and g_1 are generators of G and G_1, respectively. A bilinear map $e : G \times G_1 \longrightarrow G_T$ owns the following properties:

1) Bilinear: For any $S \in G$ and $T \in G_1$ and $u, v \in Z_p$, $e(S^u, T^v) = e(S, T)^{uv}$.
2) Non-degenerate: $e(S, T) \neq 1$.
3) Computable: There is an efficient algorithm to compute $e(S, T)$ for any $S \in G$ and $T \in G_1$.

B. COMPUTATIONAL DIFFIE-HELLMAN (CDH) ASSUMPTION

Given $\{g, g^a, g^b\}$, where g is the generator of G with prime order p and $a, b \in Z_p$. It is intractable to calculate g^{ab}.

C. DECISIONAL DIFFIE-HELLMAN (DDH) ASSUMPTION

Given $\{g^a, g^b, g^c\}$, where g is the generator of G with prime order p and $a, b, c \in Z_p$. It is intractable to distinguish $\{g^a, g^b, g^{ab}\}$ from $\{g^a, g^b, g^c\}$.

III. DEFINITION OF V-PKEET

A. SYSTEM MODEL

In our system (see Fig.3), there are several entities involved: trusted authority (TA), data owners described as Alice and Bob, cloud server (CS).

1) Trusted Authority (TA). The TA is in charge of generating system public parameters that will be used by the CS and data owners, and of generating public/private key pairs for data owners
2) Data Owners. Alice and Bob use their respective public key to encrypt their private data, called M_A and M_B, then outsource the resulting ciphertexts denoted

by C_A and C_B to the CS. At any point in time when the data owner intends to authorize the equality test, he (or she) will generate a trapdoor to delegate the CS to perform the test.

3) Cloud Server (CS). The CS stores the resulting ciphertexts, and performs the equality test algorithm but without decrypting ciphertexts. After the cloud has performed the equality test, it returns a proof to Alice (or Bob) which is used to verify the validity of the result.

B. DEFINITION OF V-PKEET

Definition 1: A V-PKEET cryptosystem has the following algorithms:

1) ***Setup***(λ): It inputs a security parameter λ, and generates a public parameter pp as output.
2) ***KeyGen(pp)***: It inputs the public parameter pp, and generates a public/secret key pair(pk_i, sk_i) for receiver U_i as output.
3) ***Enc***(M_i, pk_i): It inputs a message M_i and the public key pk_i of receiver U_i, and generates a ciphertext C as output.
4) ***Dec***(C_i, sk_i): It inputs a ciphertext C_i and the secret key sk_i of receiver U_i, and returns a plaintext M_i as output.

In order to complete $Type - \gamma(\gamma = 1, 2, 3)$ delegation for U_i and U_j, we give the definitions of $Aut_\gamma(\gamma = 1, 2, 3)$ algorithm to compute trapdoors for $U_i's/U_j's$ ciphextext (or ciphertexts), and $Test_\gamma(\gamma = 1, 2, 3)$ algorithm to test if the two ciphertexts are encrypted under the identical plaintext.

Type-1 Authorization:

Type-1-1 $Aut_1(sk_i)$: This algorithm inputs the secret key sk_i, and calculates a trapdoor $td_{(1,i)}$ for U_i as output.

Type-1-2 $Auxiliary_1(M_i, pk_j)$: This algorithm uses $U_i's$ message Mi and $U_j's$ public key as inputs and produces some auxiliary information Dig_i and au_i as output, then gives au_i to the cloud.

Type-1-3 $Test_1(C_i, td_{(1,i)}, au_i, C_j, td_{(1,j)}, au_j)$: This algorithm inputs $U_i's$ ciphertext C_i, the trapdoor $td_{(1,i)}$, the auxiliary information au_i, $U_j's$ ciphertext C_j, the trapdoor $td_{(1,j)}$ and the auxiliary information au_j, and outputs 1 if C_i and C_j are encrypted under the same plaintext and 0 otherwise. After that the cloud calculates a proof for U_i(or U_j) and returns it to U_i(or U_j) which can verify if the cloud has devotedly executed the protocol.

Type-2 Authorization:

Type-2-1 $Aut_2(sk_i, pk_j)$: This algorithm inputs $U_i's$ secret key sk_i and $U_j's$ public key pk_j, and calculates a trapdoor $td_{(2,i,j)}$ for (U_i, U_j) as output.

Type-2-2 $Auxiliary_2(M_i, pk_j)$: This algorithm uses $U_i's$ message M_i and $U_j's$ public key pk_j as inputs, and produces some auxiliary information Dig_i and au_i as output, then gives au_i to cloud.

Type-2-3 $Test_2(C_i, td_{(2,i,j)}, au_i, C_j, td_{(2,j,i)}, au_j)$: It inputs $U_i's$ ciphertext C_i, the trapdoor $td_{(2,i,j)}$, the auxiliary information au_i, $U_j's$ ciphertext C_j, the trapdoor $td_{(2,j,i)}$ and the auxiliary information au_j, and outputs 1 if C_i and C_j are encrypted under the same plaintext and 0 otherwise. After that the cloud calculates a proof for U_i(or U_j) and returns it to U_i(or U_j) which can verify if the cloud has devotedly executed the protocol.

Type-3 Authorization:

Type-3-1(1) $Aut_{(3,i)}(sk_i, C_i, pk_j)$: This algorithm inputs $U_i's$ secret key sk_i, its ciphertext C_i and $U_j's$ public key pk_j, and computes a trapdoor $td_{(3,i,C_i,j)}$ for (U_i, C_i, U_j) as output.

Type-3-1(2) $Aut_{(3,j)}(sk_j, pk_i)$: This algorithm inputs $U_j's$ secret key sk_j, $U_i's$ public key pk_i, and calculates a trapdoor $td_{(3,j,i)}$ for (U_j, U_i) as output.

Type-3-2(1) $Auxiliary_{(3,i)}(M_i, pk_j)$: This algorithm inputs $U_i's$ message M_i and $U_j's$ public key pk_j, and produces some auxiliary information Dig_i and au_i as output, and gives au_i to cloud.

Type-3-2(2) $Auxiliary_{(3,j)}(M_j, pk_i)$: This algorithm inputs $U_j's$ message M_j and $U_i's$ public key pk_i, and produces some auxiliary information Dig_j and au_j as output, and gives au_j to cloud.

Type-3-3 $Test_3(C_i, td_{(3,i,C_i,j)}, au_i, C_j, td_{(3,j,i)}, au_j)$: It uses $U_i's$ ciphertext C_i, the trapdoor $td_{(3,i,C_i,j)}$, the auxiliary information au_i, $U_j's$ ciphertext C_j the trapdoor $td_{(3,j,i)}$ and the auxiliary information au_j as inputs, and outputs 1 if C_i and C_j are encrypted under the same plaintext or 0 otherwise. At the same time, the cloud returns a proof to U_i(or U_j) which can verify if the cloud has devotedly executed the protocol.

5) ***Prove***: The cloud server generates a proof which allows verification whether it has faithfully performed the protocol. If C_i and C_j are encrypted under the same plaintext, the cloud server executes $Prove1$, or else the cloud server executes $Prove2$. In $Prove1/Prove2$, $H(T_i)$ is generated in Test algorithm as $U_i's$ ciphertext conversion section.

   (1)$Prove1(H(T_i), H(T_j))$: This algorithm inputs $H(T_i)$, $H(T_j)$, and outputs a proof to U_i(or U_j).

   (2)$Prove2(H(T_i), H(T_j))$: This algorithm inputs $H(T_i)$ and $H(T_j)$, and outputs a proof sends to U_i(or U_j).
6) ***Verify***$(sk_i, Dig_i, rslt_i, proof_i)$***:*** This algorithm inputs $U_i's$ secret key sk_i, digest Dig_i, result $rslt_i$, and the proof $proof_i$, and outputs 1 if the cloud has faithfully executed the protocol and 0 otherwise.

C. SECURITY MODELS

In this paper, we consider two types of adversaries for the security model of the V-PKEET scheme:

1)*Type-I adversary:* For $Type - \gamma(\gamma = 1, 2, 3)$ authorization, the adversary owns $Type - \gamma$ trapdoor and is looking forward to disclose the message under the challenge ciphertext.

2)*Type-II adversary:* For $Type - \gamma(\gamma = 1, 2, 3)$ authorization, the adversary does not obtain $Type - \gamma$ trapdoor and is looking forward to tell apart that C_i^* is the encrypted value of which plaintext.

In the following, we give the definition of OW-CCA security for $Type - \gamma(\gamma = 1, 2, 3)$ authorization against Type-I adversary in V-PKEET.

Game 1: Let A_1 is a Type-I adversary and the underlying receivers with the subscript $k(1 \leq k \leq n)$. The game between the adversary A_1 and the challenger C is described as below:

1) *Setup*: C first executes the *Setup* algorithm and gives A_1 the outcoming public parameter pp. Then C executes *KeyGen* to produce n pairs of public/secret keys $(pk_i, sk_i)(1 \leq i \leq n)$ and gives all pk_i to A_1.
2) *Phase1*: the adversary A_1 is permitted to ask all sorts of queries for a polynomial number of times, as follows. The restriction is that $< t >$ does not appear in the key retrieve queries.
    - Key retrieve queries $< i >$: C returns sk_i to the adversary A_1.
    - Decryption queries $< i, C_i >$: C executes *Dec* algorithm on input (C_i, sk_i) to decrypt C_i using sk_i and returns the output to the adversary A_1.
    - Authorization queries: For $Type - \gamma(\gamma = 1, 2, 3)$ authorization,
      –when inputs $< i >$, C returns $td_{(1,i)}$;
      –when inputs $< i, j >$, C returns $td_{(2,i,j)}$;
      –when inputs $< i, C_i, j >$, C returns $td_{(3,i,C_i,j)}$.
3) *Challenge*: C selects a random message $M_t \in M$, executes $C_t^* = Enc(M_t, pk_t)$ and sends C_t^* to the adversary A_1 as the challenge ciphextext.
4) *Phase* 2: A_1 requests the same as the queries in Phase 1. The restrictions are that
    - $< t >$ is not used as an input of the key retrieve queries;
    - $< t, C_t^* >$ is not used as an input of the decryption queries.
5) *Guess:* A_1 returns $M_t^{'} \in M$. In the game, when $M_t^{'} = M_t$, A_1 is the winner. The probability that A_1 is the winner is denoted by

$$Adv^{OW-CCA,Type-\gamma}_{V-PKEET,A_1}(\lambda) = Pr[M_t = M_t^{'}](\gamma = 1, 2, 3) \quad (1)$$

Definition 2: We say that the V-PKEET is OW-CCA secure for $Type-\gamma(\gamma = 1, 2, 3)$ *authorization iff for any PPT adversaries* A_1, *its advantage*$Adv^{OW-CCA,Type-\gamma}_{V-PKEET,A_1}(\lambda)$ *is negligible in the security parameter* λ.

Now we give the definition of the IND-CCA security for the $Type - \gamma(\gamma = 1, 2, 3)$ authorization against Type-II adversary in V-PKEET below.

Game 2: Let A_2 is a Type-II adversary and the underlying receiver with the subscript $k(1 \leq k \leq n)$. The game between the adversary A_2 and the challenger C is described next.

1) *Setup:* C first executes the *Setup* algorithm and gives A_2 the outcoming public parameter pp. Then C executes *KeyGen* to produce n pairs of public/secret keys $(pk_i, sk_i)(1 \leq i \leq n)$ and gives all pk_i to A_2.
2) *Phase1:* A_2 requests the same as the queries in **Game 1**.
3) *Challenge:* A_2 submits two plaintexts $M_0, M_1 \in M$ of same length. C randomly selects a bit $b \in \{0, 1\}$, executes and returns $C_t^* = Enc(M_b, pk_t)$ to the A_2 as the challenge ciphextext and returns C_t^* to the A_2 as the challenge ciphextext.
4) *Phase 2*: A_2 requests the same as the queries in *Phase* 1. The restrictions are that
    - $< t >$ is not used as an input in the key retrieve queries.
    - $< t, C_t^* >$ is not used as an input in the decryption queries.
    - For $Type - \gamma(\gamma = 1, 2, 3)$ authorization queries,
      $-\gamma = 1 :< t >$is not used as an input in the authorization queries;
      $-\gamma = 2 :< t, C_t^* >$ is not used as an input in the authorization queries;
      $-\gamma = 3 :< t, C_t^*, \cdots >$ is not used as an input in the authorization queries.
5) *Guess:* A_2 returns $b' \in \{0, 1\}$. In this game, when $b = b'$, A_2 is the winner. The probability that A_2 is the winner is denoted by

$$Adv^{IND-CCA,Type-\gamma}_{V-PKEET,A_2}(\lambda) = |Pr[b = b'] - \frac{1}{2} \qquad (\gamma = 1, 2, 3) \qquad (2)$$

Definition 3: We say that the V-PKEET is IND-CCA secure for $Type-\gamma(\gamma = 1, 2, 3)$ *authorization iff for any PPT adversaries* A_2, *its advantage* $Adv^{IND-CCA,Type-\gamma}_{V-PKEET,A_2}(\lambda)$ *is negligible in the security parameter* λ.

IV. OUR V-PKEET SCHEME

Now we present an effective V-PKEET scheme.

1) *Setup*(λ): It inputs a security parameter λ and returns the public parameters pp. Firstly, the algorithm constructs a bilinear map, where the G, G_1, and G_T are three groups with prime order p and g, g_1 are the generators of G, G_1 respectively and randomly chooses $\alpha \in Z_p^*$ and calculates g^α, g_1^α. Then it selects four cryptographic hash functions: $H : G_T \rightarrow Z_p, H_1 : \{0, 1\}^* \rightarrow \{0, 1\}^l, H_2 : G_1 \rightarrow \{0, 1\}^{l_1+l_2}, H_3 : G \rightarrow G$, where l_1 and l_2 represent lengths of element of G and Z_p , respectively. $pp = \{p, e, G, G_1, G_T, g, g_1, g^\alpha, g_1^\alpha, H, H_1, H_2, H_3\}$.
2) *KeyGen*(pp): It inputs pp and randomly chooses $x_i, y_i \in Z_p^*$, then outputs the key pair for receiver U_i:

$$(pk_i, sk_i) = ((X_i = g_1^{x_i}, Y_i = g^{y_i}), (x_i, y_i))$$

3) *Enc*(M_i, pk_i): It inputs $U_i^{'}s$ plaintext M_i and public key pk_i and randomly selects $r_{i,1}, r_{i,2} \in Z_p^*$, then generates a ciphertext $C_i = (C_{i,1}, C_{i,2}, C_{i,3}, C_{i,4}, C_{i,5})$ as following:

$$C_{i,1} = g_1^{r_{i,1}}, C_{i,2} = g^{r_{i,2}}$$
$$C_{i,3} = H_2(X_i^{r_{i,1}}) \oplus (M_i || r_{i,1})$$
$$C_{i,4} = Y_i^{r_{i,2}} \cdot H_3(M_i)$$
$$C_{i,5} = H_1(C_{i,1}||C_{i,2}||C_{i,3}||C_{i,4}||M_i||r_{i,1})$$

4) *Dec*(C_i, sk_i): It inputs a ciphertext $C_i = (C_{i,1}, C_{i,2}, C_{i,3}, C_{i,4}, C_{i,5})$ and private key sk_i, then computes $M_i||r_{i,1}$. If $C_{i,1} = g_1^{r_{i,1}}$ and $C_{i,5} =$

$H_1(C_{i,1}||C_{i,2}||C_{i,3}||C_{i,4}||M_i||r_{i,1})$ both hold, it returns M_i; otherwise, it returns an error indicator $\perp$.

Type-1 Authorization:

Type-1-1 $Aut_1(sk_i)$: The algorithm returns a trapdoor $td_{(1,i)} = y_i$.

Type-1-2 $Auxiliary_1(M_i, pk_j)$: U_i generates auxiliary information $au_i = (cph_i, \sigma_i)$ for each message of U_i. Firstly, U_i computes $H(e(H_3(M_i), g_1))$ for each message of U_i and $Dig_i = g^{H(e(H_3(M_i),g_1))+\alpha}$ for verification. Then U_i transforms Dig_i to cph_i using $U_j's$ public key pk_j, $cph_i = (g^{w_{i,1}}, g^{w_{i,2}}, Dig_i Y_j^{(w_{i,1}+w_{i,2})})$, where $w_{i,1}, w_{i,2} \leftarrow Z_p$. Finally, U_i signs cph_i and gets $\sigma_i \leftarrow sigSign(sigSk, cpk_i)$to prevent the cloud from tampering cph_i which could bring a wrong ciphertext. Similarly, U_j can run the same algorithm to generate a trapdoor and auxiliary information $au_j = (cph_j, \sigma_j)$.

Type-1-3 $Test_1(C_i, td_{(1,i)}, au_i, C_j, td_{(1,j)}, au_j)$: Firstly, it calculates

$$C_{i,4}/C_{i,2}^{y_i} = H_3(M_i)$$
$$C_{j,4}/C_{j,2}^{y_j} = H_3(M_j)$$

If $H_3(M_i) = H_3(M_j)$ holds set $T_i = e(H_3(M_i), g_1)$, $T_j = e(H_3(M_j), g_1)$ return 1 and then run $Prove1(H(T_i), H(T_j)) \rightarrow proof_i$ for U_i, or return 0 and run $Prove2(H(T_i), H(T_j)) \rightarrow proof_i$ for U_i otherwise. (Similarly, the cloud can generate $proof_j$ for U_j)

Type-2 Authorization:

Type-2-1 $Aut_2(sk_i, pk_j)$: The algorithm returns a trapdoor $td_{(2,i,j)} = (X_j^{x_i}, X_j^{x_i y_i})$.

Type-2-2 $Auxiliary_2(M_i, pk_j)$: U_i computes $H(e(H_3(M_i), X_j^{x_i}))$ for each message of U_i and lets $Dig_i = g^{H(e(H_3(M_i),X_j^{x_i}))+\alpha}$ for verification. Then U_i transforms Dig_i to cph_i using $U_j's$ public key pk_j, $cph_i = (g^{w_{i,1}}, g^{w_{i,2}}, Dig_i Y_j^{(w_{i,1}+w_{i,2})})$, where $w_{i,1}, w_{i,2} \leftarrow Z_p$. Finally, U_i runs $\sigma_i \leftarrow sigSign(sigSk, cph_i)$ to obtain a signature σ_i on ciphertext cph_i. U_i generates auxiliary information $au_i = (cph_i, \sigma_i)$ for each message of U_i. Similarly, U_j can run the same algorithm to generate a trapdoor $td_{(2,j,i)} = (X_i^{x_j}, X_i^{x_j y_j})$ and auxiliary information $au_j = (cph_j, \sigma_j)$.

Type-2-3 $Test_2(C_i, td_{(2,i,j)}, au_i, C_j, td_{(2,j,i)}, au_j)$: The algorithm calculates

$$T_i = \frac{e(C_{i,4}, X_j^{x_i})}{e(C_{i,2}, X_j^{x_i y_i)}} = e(H_3(M_i), g_1)^{x_i x_j}$$

$$T_j = \frac{e(C_{j,4}, X_i^{x_j})}{e(C_{j,2}, X_i^{x_j y_j)}} = e(H_3(M_j), g_1)^{x_i x_j}$$

If $T_i = T_j$ holds, return 1 and run $Prove1(H(T_i), H(T_j)) \rightarrow proof_i$ for U_i, or return 0 and run $Prove2(H(T_i), H(T_j)) \rightarrow proof_i$ for U_i otherwise. (Similarly, the cloud can generate $proof_j$ for U_j)

Type-3 Authorization:

Type-3-1(1) $Aut_{(3,i)}(sk_i, C_i, pk_j)$: The algorithm calculates a trapdoor $td_{(3,i,C_i,j)} = (C_{i,2}^{y_i} Y_j^{y_i}, X_j^{x_i}, X_j^{x_i y_i})$.

Type-3-1(2) $Aut_{(3,j)}(sk_j, pk_i)$: The algorithm calculates a trapdoor $td_{(3,j,i)} = (X_i^{x_j}, X_i^{x_j y_j})$.

Type-3-2(1) $Auxiliary_{(3,i)}(M_i, pk_j)$: U_i computes $\tau = e(H_3(M_i), X_j^{x_i})$ for message of specific C_i and lets $Dig_i = g^{H(\tau)+\alpha}$ for verification. Then U_i transforms Dig_i to cph_i using $U_j's$ public key pk_j , $cph_i = (g^{w_{i,1}}, g^{w_{i,2}}, Dig_i Y_j^{(w_{i,1}+w_{i,2})})$, where $w_{i,1}, w_{i,2} \leftarrow Z_p$. Finally, U_i runs $\sigma_i \leftarrow sigSign(sigSk, cpk_i)$to obtain a signature σ_i on ciphertext cph_i. U_i generates auxiliary information $aui = (cph_i, \sigma_i)$ for the message of specific C_i.

Type-3-2(2) $Auxiliary_{(3,j)}(M_j, pk_i)$: U_j computes $\tau = e(H_3(M_j), X_i^{x_j})$ for each message of U_j and lets Dig_j=$g^{H(\tau)+\alpha}$ for verification. Then U_j transforms Dig_j to cph_j using $U_i's$ public key pk_i , $cph_j = (g^{w_{j,1}}, g^{w_{j,2}}, Dig_j Y_i^{(w_{j,1}+w_{j,2})})$, where $w_{j,1}, w_{j,2} \leftarrow Z_p$. Finally, U_i runs $\sigma_j \leftarrow sigSign(sigSk, cph_j)$to obtain a signature σ_j on ciphertext cph_j. U_j generates auxiliary information $au_j = (cph_j, \sigma_j)$ for each message of U_j.

Type-3-3 $Test_3(C_i, td_{(3,i,C_i,j)}, au_i, C_j, td_{(3,j,i)}, au_j)$: The algorithm calculates

$$T_i = \frac{e(C_{i,4}, X_j^{x_i}) \cdot e(Y_j, X_j^{x_i y_i})}{e(C_{i,2}^{y_i} \cdot Y_j^{y_i}, X_j^{x_i)}} = e(H_3(M_i), g_1)^{x_i x_j}$$

$$T_j = \frac{e(C_{j,4}, X_i^{x_j})}{e(C_{j,2}, X_i^{x_j y_j)}} = e(H_3(M_i), g_1)^{x_i x_j}$$

If $T_i = T_j$ holds, return 1 and run $Prove1(H(T_i), H(T_j)) \rightarrow proof_i$ for U_i, or return 0 and run $Prove2(H(T_i), H(T_j)) \rightarrow proof_i$ for U_i otherwise. (Similarly, the cloud can generate $proof_j$ for U_j)

5) *Prove*:

(1) $Prove1(H(T_i), H(T_j)) \rightarrow proof_i$:
In this case, $H(T_i) = H(T_j)$, $rslt_i = C_i$.
–Let $P(x) = x$ and compute $g^{P(\alpha)}$ by replacing x with α.
–Let $M(x) = x$ and $N(x) = H(T_j) + x$, and find two polynomials $M'(x)$, $N'(x)$ so as to $M(x)M'(x) + N(x)$ $N'(x)$=1 mod p by satisfying $gcd(M(x), N(x)) = 1$. Then computing $(g^{M(\alpha)}, g_1^{M'(\alpha)}, g_1^{N'(\alpha)})$ by replacing x with α. Witness=$(g^{M(\alpha)}, g_1^{M'(\alpha)}, g_1^{N'(\alpha)}, g^{P(\alpha)})$, set $proof_i = (Witness, cph_j, \sigma_j, rslt_i)$.

(2)$Prove2(H(T_i), H(T_j)) \rightarrow proof_i$:
In this case, $H(T_i) \neq H(T_j)$, $rslti = \phi$
–Let $P(x) = x + H(T_j)$ and calculate $g^{P(\alpha)}$ by replacing x with α.
–Let $M(x) = x + H(T_i)$ and $N(x) = x + H(T_j)$, and find two polynomials $M'(x)$, $N'(x)$ so as to $M(x)M'(x) + N(x)N'(x) = 1$ mod p by utilizing $gcd(M(x), N(x)) = 1$. Then computing $(g^{M(\alpha)}, g_1^{M'(\alpha)}, g_1^{N'(\alpha)})$ by replacing x with α. Witness=$(g^{M(\alpha)}, g_1^{M'(\alpha)}, g_1^{N'(\alpha)}, g^{P(\alpha)})$, $proof_i = (Witness, cph_j, \sigma_j, rslt_i)$.

6) $Verify(sk_i, Dig_i, rslt_i, proof_i)$:

On inputs $rslt_i$ and $proof_i$, U_i validates whether the cloud server has trustily performed the V-PKEET scheme as follows:

–Validate the completeness of cph_j by executing $sigVerify(sigpk, cph_j, \sigma_j)$. If the verification is not passed, then aborts; otherwise continue to execute the following.

–Decrypt cph_j using private key sk_i according to

$$Dig_j = Dig_j g^{y_i(t_{j,1}+t_{j,2})}/(g^{t_{j,1}} \cdot g^{t_{j,2}})^{y_i}$$

–If $rslt_i$ is empty, let $Y_i = \phi$. Otherwise decrypt $rslt_i$ to obtain the message M_i, and compute Y_i according to each type of authorization as follows:

- For Type-1 Authorization: compute $Y_i = H(e(H_3(M_i), g_1))$;
- For Type-2 Authorization: compute $Y_i = H(e(H_3(M_i), g_1^{x_j x_i}))$;
- For Type-3 Authorization: compute $Y_i = H(e(H_3(M_i), g_1^{x_j x_i}))$.

–Continue to verify the result. If the verification is not passed, it returns 0; otherwise returns 1.

- If $Y_i \neq \phi$, compute $g_1^{P'(\alpha)}$ according to $P'(x) = x + Y_i$. Otherwise, set $P'(x) = 1$ and $g_1^{P'(\alpha)} = g_1$.
- If $e(g^{M(\alpha)}, g_1^{p'(\alpha)}) \neq e(Dig_i, g_1)$, return 0. Otherwise keep on running the following.
- If $e(g^{p(\alpha)}, g_1^{p'(\alpha)}) \neq e(Dig_j, g_1)$, return 0. Otherwise keep on running the following.
- If $e(g^{M(\alpha)}, g_1^{M'(\alpha)}) \cdot e(Dig_j, g_1^{N'(\alpha)}) \neq e(g, g_1)$, return 0. Otherwise return 1.

If it outputs 1, it means that the two users' ciphertexts are the same plaintext and the cloud faithfully executes the protocol. Similarly, U_j can perform the same algorithm to check if the cloud has devotedly executed the equality test.

V. SECURITY ANALYSIS

Theorem 1: Our V-PKEET has OW-CCA security for Type $-\gamma(\gamma = 1, 2, 3)$ authorization against Type $- I$ adversary under the CDH assumption with the random oracle model.

Proof: Assume an adversary with advantage ε in the game given in **Game 1**.

Let A_1 is a probabilistic polynomial time $Type{-}I$ adversary breaking the OW-CCA security. We assume that A_1 requests at most $q_{H_1} H_1$ hash queries, $q_{H_2} H_2$ hash queries, $q_{H_3} H_3$ hash queries, q_K key retrieve queries, q_D decryption queries and q_{Aut} authorization queries for $Type - \gamma(\gamma = 1, 2, 3)$ authorization. Note that if the same request is made many times in an oracle query, the same reply will be obtained. The OW-CCA security is proven by a string of games as follows.

Game 1.0: We give the game defined in **Definition 3** as follows.

1) $pp \leftarrow \{p, e, G, G_1, G_T, g, g_1, g^\alpha, g_1^\alpha, H, H_1, H_2, H_3\}$, $\forall 1 \leq i \leq n, x_i, y_i \leftarrow Z_p^*, X_i = g_1^{x_i}, Y_i = g^{y_i}$.
2) $state \leftarrow A_1^{o_{H_1}, o_{H_2}, o_{H_3}, o_K, o_D, o_{Aut}}(\{X_i, Y_i\})_{i=1}^n$ these oracles are defined as below.
   o_{H_1} **query** $< v_1 >$: Given a new $v_1 \in \{0, 1\}^*$, the challenger C randomly selects a value h_1 uniformly from the set $\{0, 1\}^l$ and delivers h_1 to the adversary A_1.
   o_{H_2} **query** $< v_2 >$: Given a new $v_2 \in G_1$, C selects a random value h_2 uniformly from the set $\{0, 1\}^{l_1+l_2}$ and gives h_2 to the adversary A_1.
   o_{H_3} **query** $< v_3 >$: Given a new $v_3 \in G_T$, C randomly selects a value h_3 uniformly from the field Z_p and gives h_3 to the adversary A_1.
   o_K **query** $< i >$čžC gives sk_i to the adversary A_1.
   o_D **query** $< i, C_i >$čžThe challenger inputs sk_i to execute the Dec algorithm to decrypt C_i, then returns M_i to A_1.
   o_{Aut} **query**: The three types of authorization queries described as follows:
   $\gamma = 1$: given $< i >$, C inputs sk_i to execute Aut_1 algorithm, then gives $td_{(1,i)} = y_i$ to the adversary A_1.
   $\gamma = 2$: given $< i, j >$, C inputs sk_i to execute Aut_2 algorithm, then returns $td_{(2,i,j)} = (X_j^{x_i}, X_j^{x_i y_i})$ to the adversary A_1.
   $\gamma = 3$: given $< i, C_i, j >$, C inputs sk_i to execute Aut_3 algorithm, then gives $td_{(3,i,C_i,j)} = (C_{i,2}^{y_i} Y_j^{y_i}, X_j^{x_i}, X_j^{x_i y_i})$ to the adversary A_1.
3) $M_t \leftarrow G^*, r_{t,1}, r_{t,2} \leftarrow Z_p^*, C_t^* = (C_{t,1}^*, C_{t,2}^*, C_{t,3}^*, C_{t,4}^*, C_{t,5}^*)$ is defined as follows:
$$C_{t,1}^* = g_1^{r_{t,1}}, \quad C_{t,2}^* = g^{r_{t,2}}$$
$$C_{t,3}^* = H_2(X_t^{r_{t,1}}) \oplus (M_t || r_{t,1})$$
$$C_{t,4}^* = Y_t^{r_{t,2}} \cdot H_3(M_t)$$
$$C_{t,5}^* = H_1(C_{t,1}^* || C_{t,2}^* || C_{t,3}^* || C_{t,4}^* || M_t || r_{t,1})$$
4) $M_t' \leftarrow A_1^{o_{H_1}, o_{H_2}, o_{H_3}, o_K, o_D, o_{Aut}}(state, C_t^*)$. Denoted by $\mathbf{S}_{1,0}$ the event that $M_t = M_t'$ in **Game 1.0**. We have that

$$\begin{aligned} \varepsilon_0 &= Adv_{V-PKEET,A_1}^{OW-CCA,Type-\gamma}(q_{H_1}, q_{H_2}, q_{H_3}, q_K, q_D, q_{Aut}) \\ &= Pr[S_{1,0}](\gamma = 1, 2, 3) \end{aligned} \tag{3}$$

Clearly,

$$\varepsilon_0 = \varepsilon \tag{4}$$

holds. *Game 1.1:*

1) $pp \leftarrow \{p, e, G, G_1, G_T, g, g_1, g^\alpha, g_1^\alpha, H, H_1, H_2, H_3\}$, $\forall 1 \leq i \leq n, x_i, y_i \leftarrow Z_p^*, X_i = g_1^{x_i}, Y_i = g^{y_i}$.
2) $state \leftarrow A_1^{o_{H_1}, o_{H_2}, o_{H_3}, o_K, o_D, o_{Aut}}(\{X_i, Y_i\})_{i=1}^n$ these oracles are defined as below.
   o_{H_1} **query** $< v_1 >$: It is identical with that in **Game 1.0**.
   o_{H_2} **query** $< v_2 >$: It is identical with that in **Game 1.0**.
   o_{H_3} **query** $< v_3 >$: It is identical with that in **Game 1.0**.
   o_K **query** $< i >$: It is identical with that in **Game 1.0**.
   o_D **query** $< i, C_i >$: C executes a request to H_2 on input $(C_{i,1}^{x_i})$ and outputs the answer h_2. Then the

challenger computes $C_{i,3} \oplus h_2$ to obtain $M_i||r_{i,1}$ and tests whether the following equations hold:

$$C_{i,1}=g_1^{r_{i,1}},\ C_{i,5}=H_1(C_{i,1}||C_{i,2}||C_{i,3}||C_{i,4}||M_i||r_{i,1})$$

If the test fails, the challenger returns $\perp$ to A_1; otherwise, it returns M_i to A_1. Denoted by $\mathbf{E}_1$ event that in some ciphertexts C_i, there is a new request $C_{i,1}||C_{i,2}||C_{i,3}||C_{i,4}||M_i||r_{i,1}$ to H_1 results in $C_{i,5}$, in other words the oracle gives the same reply when the input is different.

o_{Aut} **query**: It is identical with that in **Game 1.0**.

3) $M_t \leftarrow G^*, r_{t,1}, r_{t,2} \leftarrow Z_p^*, W_1^* \leftarrow \{0,1\}^{l_1+l_2}, C_t^* = (C_{t,1}^*, C_{t,2}^*, C_{t,3}^*, C_{t,4}^*, C_{t,5}^*)$ is defined as follows:

$$\begin{aligned} C_{t,1}^* &= g_1^{r_{t,1}}, \quad C_{t,2}^* = g^{r_{t,2}} \\ C_{t,3}^* &= W_1^* \oplus (M_t||r_{t,1}) \\ C_{t,4}^* &= Y_t^{r_{t,2}} \cdot H_3(M_t) \\ C_{t,5}^* &= H_1(C_{t,1}^*||C_{t,2}^*||C_{t,3}^*||C_{t,4}^*||M_t||r_{t,1}) \end{aligned}$$

Add the tuple $((C_{t,1}^*)^{x_i}, W_1^*)$ to table T_2 for H_2.

4) $M_t^{'} \leftarrow A_1^{o_{H_1},o_{H_2},o_{H_3},o_K,o_D,o_{Aut}}(state, C_t^*)$ Denoted by $\mathbf{S}_{1.1}$ the event that $M_t = M_t^{'}$ in **Game 1.1**. Let

$$\varepsilon_1 = Pr[S_{1.1}] \tag{5}$$

Because of the idealness of the random oracle, $\Pr[\mathbf{E}_1]$ is negligible. Refer to the Difference Lemma in [28], we have

$$|\varepsilon_1 - \varepsilon_0| = Pr[E_1] \tag{6}$$

Game 1.2:

1) $pp \leftarrow \{p, e, G, G_1, G_T, g, g_1, g^\alpha, g_1^\alpha, H, H_1, H_2, H_3\}$, $\forall 1 \le i \le n, x_i, y_i \leftarrow Z_p^*, X_i = g_1^{x_i}, Y_i = g^{y_i}$.

2) $state \leftarrow A_1^{o_{H_1},o_{H_2},o_{H_3},o_K,o_D,o_{Aut}}(\{X_i, Y_i\})_{i=1}^n$ these oracles are defined as below.

o_{H_1} **query** $< v_1 >$: It is identical with that in **Game 1.1**.

o_{H_2} **query** $< v_2 >$: It is identical with that in **Game 1.1**, except that if A_1 queries $(C_{t,1}^*)^{x_i}$, the game is over. The event is denoted by $\mathbf{E}_2$.

o_{H_3} **query** $< v_3 >$: It is identical with that in **Game 1.1**.

o_K **query** $< i >$: It is identical with that in **Game 1.1**.

o_D **query** $< i, C_i >$: It is identical with that in **Game 1.1**, unless that getting the challenge ciphertext C_t^*(see below), A_1 requires to decrypt the ciphertext $(C_{t,1}^*, C_{t,2}^*, C_{t,3}^*, C_{t,4}^*, C_{t,5}^*)$, where $C_{t,3}^{'*} \neq C_{t,3}^*$, C returns $\perp$ to the adversary A_1.

o_{Aut} **query**: It is identical with that in **Game 1.1**.

3) $M_t \leftarrow G^*, r_{t,1}, r_{t,2} \leftarrow Z_p^*, W_2^* \leftarrow \{0,1\}^{l_1+l_2}, C_t^* = (C_{t,1}^*, C_{t,2}^*, C_{t,3}^*, C_{t,4}^*, C_{t,5}^*)$ is defined as follows:

$$\begin{aligned} C_{t,1}^* &= g_1^{r_{t,1}}, \quad C_{t,2}^* = g^{r_{t,2}} \\ C_{t,3}^* &= W_2^*, C_{t,4}^* = Y_t^{r_{t,2}} \cdot H_3(M_t) \\ C_{t,5}^* &= H_1(C_{t,1}^*||C_{t,2}^*||C_{t,3}^*||C_{t,4}^*||M_t||r_{t,1}) \end{aligned}$$

Add the tuple $((C_{t,1}^*)^{x_t}, W_2^* \oplus (M_t||r_{t,1}))$ to table T_2 for H_2.

4) $M_t^{'} \leftarrow A_1^{o_{H_1},o_{H_2},o_{H_3},o_K,o_D,o_{Aut}}(state, C_t^*)$ Denoted by $\mathbf{S}_{1.2}$ the event that $M_t = M_t^{'}$ in **Game 1.2**. Let

$$\varepsilon_2 = Pr[S_{1.2}] \tag{7}$$

The game is the same as the distribution in **Game 1.1** unless the event $\mathbf{E}_2$ occurs, in other words $g_1^{r_{t,1}x_t}$ is requested to the random oracle H_2. It is worth noting that the secret key x_t is never acted as the response that the attacker requests. Consequently, $\mathbf{Pr}[\mathrm{E}_2]$ is negligible under the CDH problem in G_1. From the Difference Lemma in [28], the following inequality holds

$$|\varepsilon_2 - \varepsilon_1| = Pr[E_2] \tag{8}$$

Because H_1 and H_3 are viewed as random oracles, the ε_2 is negligible. According to the aforementioned analysis, combining Eqs.(3)-(8), we have

$$\varepsilon \le Pr[E_1] + Pr[E_2] + \varepsilon_2 \tag{9}$$

The equation is negligible under the CDH problem in G_1. The *Theorem 1* has been proved completely.

Theorem 2: Our V-PKEET has IND-CCA security for $Type - \gamma(\gamma = 1, 2, 3)$ authorization against $Type - II$ adversary under CDH assumption and DDH assumption with the random oracle model.

Proof: Assume an adversary has the advantage ε in the game given in **Game 1**.

Let A_2 is a probabilistic polynomial time $Type - II$ adversary breaching the IND-CCA security of our scheme. A_2 requests at most $q_{H_1}H_1$ hash queries, $q_{H_2}H_2$ hash queries, $q_{H_3}H_3$ hash queries, q_K key retrieve queries, q_D decryption queries and q_{Aut} authorization queries for $Type - \gamma(\gamma = 1, 2, 3)$ authorization. Note that if the same request is made many times in an oracle query, the same reply will be obtained. The security is proven by a battery of games as follows.

Game 2.0: We give the game that defined in **Definition 3** as follows.

1) $pp \leftarrow \{p, e, G, G_1, G_T, g, g_1, g^\alpha, g_1^\alpha, H, H_1, H_2, H_3\}$, $\forall 1 \le i \le n, x_i, y_i \leftarrow Z_p^*, X_i = g_1^{x_i}, Y_i = g^{y_i}$.

2) $(M_0, M_1) \leftarrow A_2^{o_{H_1},o_{H_2},o_{H_3},o_K,o_D,o_{Aut}}(\{X_i, Y_i\})_{i=1}^n$ these oracles are issued the same as Game 1.0.

3) $b \leftarrow \{0,1\}, r_{t,1}, r_{t,2} \leftarrow Z_p^*, C_t^* = (C_{t,1}^*, C_{t,2}^*, C_{t,3}^*, C_{t,4}^*, C_{t,5}^*)$ is computed as follows:

$$\begin{aligned} C_{t,1}^* &= g_1^{r_{t,1}}, \quad C_{t,2}^* = g^{r_{t,2}} \\ C_{t,3}^* &= H_2(X_t^{r_{t,1}}) \oplus (M_b||r_{t,1}) \\ C_{t,4}^* &= Y_t^{r_{t,2}} \cdot H_3(M_b) \\ C_{t,5}^* &= H_1(C_{t,1}^*||C_{t,2}^*||C_{t,3}^*||C_{t,4}^*||M_b||r_{t,1}) \end{aligned}$$

4) $b' \leftarrow A_2^{o_{H_1}, o_{H_2}, o_{H_3}, o_K, o_D, o_{Aut}}(C_t^*)$ We called the event $b = b'$ $\mathbf{S}_{2,0}$ in this game. So we have

$$\begin{aligned}\varepsilon_0 &= Adv_{V-PKEET,A_2}^{IND-CCA,Type-\gamma}(q_{H_1}, q_{H_2}, q_{H_3}, q_K, q_D, q_{Aut}) \\ &= Pr[S_{2,0}](\gamma = 1, 2, 3) \qquad (10)\end{aligned}$$

Clearly,

$$|\varepsilon_0 - 1/2| = \varepsilon \qquad (11)$$

holds.

Game 2.1:

1) $pp \leftarrow \{p, e, G, G_1, G_T, g, g_1, g^\alpha, g_1^\alpha, H, H_1, H_2, H_3\}$, $\forall 1 \leq i \leq n, x_i, y_i \leftarrow Z_p^*, X_i = g_1^{x_i}, Y_i = g^{y_i}$.
2) $(M_0, M_1) \leftarrow A_2^{o_{H_1}, o_{H_2}, o_{H_3}, o_K, o_D, o_{Aut}}(\{X_i, Y_i\})_{i=1}^n$ these oracles are issued the same as **Game 1.1**.
3) $b \leftarrow \{0, 1\}, r_{t,1}, r_{t,2} \leftarrow Z_p^*, W_1^* \leftarrow \{0, 1\}^{l_1+l_2}, C_t^* = (C_{t,1}^*, C_{t,2}^*, C_{t,3}^*, C_{t,4}^*, C_{t,5}^*)$ is computed as follows:

$$\begin{aligned}&C_{t,1}^* = g_1^{r_{t,1}}, \quad C_{t,2}^* = g^{r_{t,2}} \\ &C_{t,3}^* = W_1^* \oplus (M_b || r_{t,1}) \\ &C_{t,4}^* = Y_t^{r_{t,2}} \cdot H_3(M_b) \\ &C_{t,5}^* = H_1(C_{t,1}^* || C_{t,2}^* || C_{t,3}^* || C_{t,4}^* || M_b || r_{t,1})\end{aligned}$$

Add the tuple $((C_{t,1}^*)^{x_t}, W_1^*)$ to table T_2 for H_2.
4) $b' \leftarrow A_2^{o_{H_1}, o_{H_2}, o_{H_3}, o_K, o_D, o_{Aut}}(C_t^*)$ We called the event $b = b'$ $\mathbf{S}_{2,1}$ in this game. Let

$$\varepsilon_1 = Pr[S_{2,1}] \qquad (12)$$

Because of the idealness of the random oracle, $\mathbf{Pr}[\mathrm{E}_1]$ is negligible. Refer to the Difference Lemma in [28], the following inequality holds

$$|\varepsilon_1 - \varepsilon_0| = Pr[E_1] \qquad (13)$$

Game 2.2:

1) $pp \leftarrow \{p, e, G, G_1, G_T, g, g_1, g^\alpha, g_1^\alpha, H, H_1, H_2, H_3\}$, $\forall 1 \leq i \leq n, x_i, y_i \leftarrow Z_p^*, X_i = g_1^{x_i}, Y_i = g^{y_i}$.
2) $(M_0, M_1) \leftarrow A_2^{o_{H_1}, o_{H_2}, o_{H_3}, o_K, o_D, o_{Aut}}(\{X_i, Y_i\})_{i=1}^n$ these oracles are issued the same as Game 1.1.
3) $b \leftarrow \{0, 1\}, r_{t,1}, r_{t,2} \leftarrow Z_p^*, W_2^* \leftarrow \{0, 1\}^{l_1+l_2}, C_t^* = (C_{t,1}^*, C_{t,2}^*, C_{t,3}^*, C_{t,4}^*, C_{t,5}^*)$ is computed as follows:

$$\begin{aligned}&C_{t,1}^* = g_1^{r_{t,1}}, \quad C_{t,2}^* = g^{r_{t,2}} \\ &C_{t,3}^* = W_2^*, \quad C_{t,4}^* = Y_t^{r_{t,2}} \cdot H_3(M_b) \\ &C_{t,5}^* = H_1(C_{t,1}^* || C_{t,2}^* || C_{t,3}^* || C_{t,4}^* || M_b || r_{t,1})\end{aligned}$$

Add the tuple $((C_{t,1}^*)^{x_t}, W_2^* \oplus (M_b || r_{t,1}))$ to table T_2 for H_2.
4) $b' \leftarrow A_2^{o_{H_1}, o_{H_2}, o_{H_3}, o_K, o_D, o_{Aut}}(C_t^*)$ We called the event $b = b'$ $\mathbf{S}_{2,2}$ in this game. Let

$$\varepsilon_2 = Pr[S_{2,2}] \qquad (14)$$

The game is the same as the distribution in **Game 2.1** except that the event $\mathbf{E}_2$ occurs, in other words $g_1^{r_{t,1}x_t}$ is requested to the oracle H_2. It is worth noting that the secret key x_t is never acted as the response that the attacker requests. Consequently, $\mathbf{Pr}[\mathrm{E}_2]$ is negligible under the CDH problem in G_1. Refer to the Difference Lemma in [28], the following inequality holds

$$|\varepsilon_2 - \varepsilon_1| = Pr[E_2] \qquad (15)$$

Game 2.3:

1) $pp \leftarrow \{p, e, G, G_1, G_T, g, g_1, g^\alpha, g_1^\alpha, H, H_1, H_2, H_3\}$, $\forall 1 \leq i \leq n, x_i, y_i \leftarrow Z_p^*, X_i = g_1^{x_i}, Y_i = g^{y_i}$.
2) $(M_0, M_1) \leftarrow A_2^{o_{H_1}, o_{H_2}, o_{H_3}, o_K, o_D, o_{Aut}}(\{X_i, Y_i\})_{i=1}^n$, these oracles are defined as below.

   o_{H_1} **query** $< v_1 >$: Same as that in **Game 2.2** except that if $v_1 = (C_{t,1}^* || C_{t,2}^* || C_{t,3}^* || C_{t,4}^* || M_b || r_{t,1})$, the game is over. The event is denoted by $\mathbf{E}_3$.

   o_{H_2} **query** $< v_2 >$, o_{H_3} **query** $< v_3 >$, o_K **query** $< i >$, o_D **query** $< i, C_i >$, o_{Aut} **query**: these queries are issued identical to **Game 2.2**.
3) $b \leftarrow \{0, 1\}, r_{t,1}, r_{t,2} \leftarrow Z_p^*, W_{3,1}^* \leftarrow \{0, 1\}^{l_1+l_2}$, $W_{3,2}^* \leftarrow \{0, 1\}^l$, $C_t^* = (C_{t,1}^*, C_{t,2}^*, C_{t,3}^*, C_{t,4}^*, C_{t,5}^*)$ is computed as follows:

$$\begin{aligned}&C_{t,1}^* = g_1^{r_{t,1}}, \quad C_{t,2}^* = g^{r_{t,2}}, \ C_{t,3}^* = W_{3,1}^*, \\ &C_{t,4}^* = Y_t^{r_{t,2}} \cdot H_3(M_b), \quad C_{t,5}^* = W_{3,2}^*\end{aligned}$$

Add the tuple $((C_{t,1}^*)^{x_t}, W_{3,1}^* \oplus (M_b || r_{t,1}))$ to table T_2 for H_2 and $(C_{t,1}^* || C_{t,2}^* || C_{t,3}^* || C_{t,4}^* || M_b || r_{t,1})$ to the table T_1 for H_1.
4) $b' \leftarrow A_2^{o_{H_1}, o_{H_2}, o_{H_3}, o_K, o_D, o_{Aut}}(C_t^*)$. We called the event $b = b'$ $\mathbf{S}_{2,3}$ in this game. Let

$$\varepsilon_3 = Pr[S_{2,3}] \qquad (16)$$

The game is the same as the distribution in **Game 2.2** only if the event $\mathbf{E}_3$ occurs, in other words $(C_{t,1}^* || C_{t,2}^* || C_{t,3}^* || C_{t,4}^* || M_b || r_{t,1})$ is requested to the random oracle H_1. We have

$$|\varepsilon_3 - \varepsilon_2| \leq Pr[E_3] \qquad (17)$$

is negligible under the CDH problem in G_1.

On referring to the proof of semantic security of ElGamal scheme [28], we note that it is obvious to validate that $\varepsilon_3 - 1/2$ is negligible under the DDH problem in G. According to the aforementioned analysis, combining Eqs.(10)-(17), we have $|\varepsilon_0 - \varepsilon_3| \leq Pr[E_1 + Pr[E_2] + Pr[E_3]$. The equation is negligible in the oracle under the CDH problem in G_1 and DDH problem in G. It is worth noting that $\varepsilon = |\varepsilon_0 - 1/2|$ and $|\varepsilon_3 - 1/2|$ is negligible, so ε is negligible. The *Theorem 2* has been proved completely.

VI. COMPARISON

Table I compares the computational complexity between previous and our PKEET schemes in terms of Type-1 authorization (coarse-grained authorization). We observe that our design has mildly smaller space in terms of the public key, the secret key and the ciphertexts than PKEET-FA (Type-1) [9] and costs a bit less than PKEET-FA(Type-1) in terms of encryption, decryption and test, noting that we only consider the equality test rather than the cost of generating the proof for verification. Moreover, in our authorization,

TABLE 1. Comparisons between existing works and our scheme in coarse-grained authorization.

	[18]	[9](1)	Ours(1)
$\vert pk\vert$	$2\vert G\vert$	$3\vert G\vert$	$2\vert G\vert$
$\vert sk\vert$	$2\vert Z_p\vert$	$3\vert Z_p\vert$	$2\vert Z_p\vert$
$\vert C\vert$	$4\vert G\vert + \vert Z_p\vert + l_0$	$5\vert G\vert + \vert Z_p\vert$	$4\vert G\vert + \vert Z_p\vert + l_0$
C_{Enc}	5Exp	6Exp	4Exp
C_{Dec}	2Exp	5Exp	2Exp
C_{Aut}	0	0	0
C_{Test}	4Exp	2Pairing+2Exp	2Exp
Sec/wa	OW-CCA	OW-CCA	OW-CCA
Sec/woa	IND-CCA	IND-CCA	IND-CCA
Gra	①	①	①
verify	×	×	✓

Legends:$\vert pk\vert$, $\vert sk\vert$ and $\vert C\vert$: the size of public key, the secret key and the ciphertext; C_{Enc}, C_{Dec}, C_{Aut} and C_{Test}: the computational complexity of algorithms for encryption, decryption, authorization and test; Sec/wa: authorized security; Sec/woa: unauthorized security; Gra: authorization granularity with descending grades: ①>②>③; Ours(γ)($\gamma = 1, 2, 3$): our scheme with Type-γ authorization; Exp: exponentiation; Pairing: pairing evaluation.(next table is the same as)

TABLE 2. Comparisons between existing works and our scheme in fine-grained authorizations.

	[11]	Ours(2)	Ours(3)
$\vert pk\vert$	$2\vert G\vert$	$2\vert G\vert$	$2\vert G\vert$
$\vert sk\vert$	$2\vert Z_p\vert$	$2\vert Z_p\vert$	$2\vert Z_p\vert$
$\vert C\vert$	$4\vert G\vert + \vert Z_p\vert + l_0$	$4\vert G\vert + \vert Z_p\vert + l_0$	$4\vert G\vert + \vert Z_p\vert + l_0$
C_{Enc}	4Exp	4Exp	4Exp
C_{Dec}	2Exp	2Exp	2Exp
C_{Aut}	3Exp	4Exp	6Exp
C_{Test}	4Pairing	4Pairing	5Pairing
Sec/wa	OW-CCA	OW-CCA	OW-CCA
Sec/woa	IND-CCA	IND-CCA	IND-CCA
Gra	②	②	③
verify	×	✓	✓

we add the function of verification for the result from the cloud server.

Table II compares the computational complexity between previous and our PKEET schemes in the other two authorizations (fine-grained authorizations). Compared with [11], our Type-2 authorization requires no interactive protocol to produce the trapdoor td and completes an additional function of verification. Type-3 authorization is firstly presented in our scheme, which could protect the users' privacy more finely-grained. Our Type-2 and Type-3 authorizations have very small computational overheads in the light of Aut algorithm. Furthermore, the Type-2 and Type-3 authorizations have considerably larger computational overheads over equality tests. Because they need several bilinear pairing operations and a bilinear pairing costs about four times longer than an exponentiation in a conventional desktop computer environment according to the experimental results in [29] and [30]. With respect to the security, same as [9], [11], and [18], V-PKEET accomplishes OW-CCA authorized security and IND-CCA unauthorized security. Our scheme achieves verification, while the existing schemes do not take this into account.

VII. CONCLUSION

In this research, we expand the study of PKEET-FA, and design a verifiable PKEET scheme, called V-PKEET, which, to the best of our knowledge, is the first PKEET scheme providing a verification mechanism to check whether the cloud server has devotedly executed the equality test. Furthermore, V-PKEET achieves three types of authorization to protect the dynamic privacy of data owners. Finally, we prove that our scheme achieves OW-CCA authorized security and IND-CCA unauthorized security under the CDH assumption and DDH assumption in a random oracle model. This proposed scheme can make the cloud computing in 5G networks more secure.

ACKNOWLEDGMENT

The authors are very grateful to the anonymous referees for their detailed comments and suggestions regarding this paper.

REFERENCES

[1] J. Yu, K. Ren, C. Wang, and V. Varadharajan, "Enabling cloud storage auditing with key-exposure resistance," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 6, pp. 1167–1179, Jun. 2015.

[2] J. Yu, K. Ren, and C. Wang, "Enabling cloud storage auditing with verifiable outsourcing of key updates," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 6, pp. 1362–1375, Jun. 2016.

[3] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. Lou, "Fuzzy keyword search over encrypted data in cloud computing," in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 1–5.

[4] B. Wang, S. Yu, W. Lou, and Y. T. Hou, "Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud," in *Proc. IEEE INFOCOM*, Apr. 2014, pp. 2112–2120.

[5] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 1, pp. 222–233, Jan. 2014.

[6] M. Green *et al.*, "Outsourcing the decryption of abe ciphertexts," in *Proc. USENIX Secur. Symp.*, 2011, pp. 1–16.

[7] X. Liu, R. H. Deng, K.-K. R. Choo, and J. Weng, "An efficient privacy-preserving outsourced calculation toolkit with multiple keys," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 11, pp. 2401–2414, Nov. 2016.

[8] G. Yang, C. H. Tan, Q. Huang, and D. S. Wong, "Probabilistic public key encryption with equality test," in *Proc. Cryptographers' Track RSA Conf.*, 2010, pp. 119–131.

[9] S. Ma, Q. Huang, M. Zhang, and B. Yang, "Efficient public key encryption with equality test supporting flexible authorization," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 3, pp. 458–470, Mar. 2015.

[10] Q. Tang, "Towards public key encryption scheme supporting equality test with fine-grained authorization," in *Proc. Austral. Conf. Inf. Secur. Privacy*, 2011, pp. 389–406.

[11] Q. Tang, "Public key encryption schemes supporting equality test with authorisation of different granularity," *Int. J. Appl. Cryptogr.*, vol. 2, no. 4, pp. 304–321, 2012.

[12] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.*, 2004, pp. 506–522.

[13] M. Bellare, A. Boldyreva, and A. O'Neill, "Deterministic and efficiently searchable encryption," in *Proc. Annu. Int. Cryptol. Conf.*, 2007, pp. 535–552.

[14] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," in *Proc. Theory Cryptograph. Conf.*, 2007, pp. 535–554.

[15] H. S. Rhee, J. H. Park, W. Susilo, and D. H. Lee, "Improved searchable public key encryption with designated tester," in *Proc. 4th Int. Symp. Inf., Comput., Commun. Secur.*, 2009, pp. 376–379.

[16] W. C. Yau, R. C.-W. Phan, S.-H. Heng, and B.-M. Goi, "Proxy reencryption with keyword search: New denitions and algorithms," in *Proc. Int. Conf. SecTec*, 2010, pp. 149–160.

[17] L. Guo, B. Lu, X. Li, and H. Xu, "A verifiable proxy re-encryption with keyword search without random oracle," in *Proc. 9th Int. Conf. Comput. Intell. Secur.*, Dec. 2013, pp. 474–478.

[18] Q. Tang, "Public key encryption supporting plaintext equality test and user-specified authorization," *Secur. Commun. Netw.*, vol. 5, no. 12, pp. 1351–1362, Dec. 2012.

[19] K. Huang, R. Tso, Y.-C. Chen, S. M. M. Rahman, A. Almogren, and A. Alamri, "PKE-AET: Public key encryption with authorized equality test," *Comput. J.*, vol. 58, no. 10, pp. 2686–2697, 2015.

[20] H. T. Lee, S. Ling, J. H. Seo, and H. Wang, "CCA2 attack and modification of Huang et al.Šs public key encryption with authorized equality test," *Comput. J.*, vol. 59, no. 11, pp. 1689–1694, 2016.

[21] R. Gennaro, C. Gentry, and B. Parno, "Non-interactive verifiable computing: Outsourcing computation to untrusted workers," in *Proc. Annu. Cryptol. Conf.*, 2010, pp. 465–482.

[22] S. Benabbas, R. Gennaro, and Y. Vahlis, "Verifiable delegation of computation over large datasets," in *Proc. Annu. Cryptol. Conf.*, 2011, pp. 111–131.

[23] D. Fiore and R. Gennaro, "Publicly verifiable delegation of large polynomials and matrix computations, with applications," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2012, pp. 501–512.

[24] B. Parno, M. Raykova, and V. Vaikuntanathan, "How to delegate and verify in public: Verifiable computation from attribute-based encryption," in *Proc. Theory Cryptography Conf.*, 2012, pp. 422–439.

[25] Q. Zheng and S. Xu, "Verifiable delegated set intersection operations on outsourced encrypted data," in *Proc. IEEE Int. Conf. Cloud Eng.*, Mar. 2015, pp. 175–184.

[26] L. Nguyen, "Accumulators from bilinear pairings and applications," in *Proc. Cryptographers' Track RSA Conf.*, 2005, pp. 275–292.

[27] I. Damgård and N. Triandopoulos, "Supporting non-membership proofs with bilinear-map accumulators," in *Proc. IACR Cryptol. ePrint Arch.*, 2008, p. 538.

[28] V. Shoup, "Sequences of games: A tool for taming complexity in security proofs," *IACR Cryptol. EPrint Archive*, 2004, p. 332.

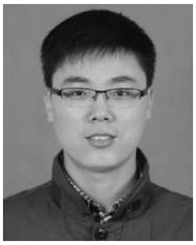
[29] K. Lauter, "The advantages of elliptic curve cryptography for wireless security," *IEEE Wireless Commun.*, vol. 11, no. 1, pp. 62–67, Feb. 2004.

[30] M. Yoshitomi, T. Takagi, S. Kiyomoto, and T. Tanaka, "Efficient implementation of the pairing on mobilephones using BREW," *IEICE Trans. Inf. Syst.*, vol. 91, no. 5, pp. 1330–1337, 2008.

YAN XU received the B.S. degree from Shandong University, Jinan, China, in 2004, and the Ph.D. degree in computer science and technology from the University of Science and Technology of China, China, in 2015. She is currently a Lecturer with Anhui University. Her research interests include network and information security.

MING WANG received the B.S. degree in computer science from Anhui University, Hefei, China, in 2014, where she is currently pursuing the M.S. degree with the School of Computer Science and Technology. Her research interests include data security and privacy protection technology.

HONG ZHONG received the Ph.D. degree from the University of Science and Technology of China in 2005. She has been a Professor since 2009 and the Executive Dean of the School of Computer Science and Technology, Anhui University, China. Her research interests cover network and information security.



JIE CUI received the Ph.D. degree from the University of Science and Technology of China in 2012. He is currently an Associate Professor with the School of Computer Science and Technology, Anhui University. He has authored over 20 papers. His research interests include network and information security.

LU LIU received the M.Sc. degree in data communication systems from Brunel University, U.K., and the Ph.D. degree from the University of Surrey, U.K. (funded by DIF DTC). He is currently a Professor of distributed computing with the University of Derby, U.K. His research interests are in the areas of cloud computing, service computing, computer networks, and peer-to-peer networking. He is a fellow of the British Computer Society.

VIRGINIA N. L. FRANQUEIRA received the Ph.D. degree in computer science from NL in 2009, and the M.Sc. degree from the Federal University of Espirito Santo (BR). Prior to that, she was a Lecturer with the University of Central Lancashire, U.K, held a post-doctoral research position with the University of Twente (NL), and was an Information Security Consultant in the U.K. She is currently a Senior Lecturer with the University of Derby, U.K. Her topics of research interest include security engineering, risk management and estimation, attack modeling, and external insider threat. She is a member of the IEEE Computer Society and the British Computer Society.