applied sciences

MDPI

Article

VR-PEKS: A Verifiable and Resistant to Keyword Guess Attack Public Key Encryption with Keyword Search Scheme

Yingying Tang , Yuling Chen *, Yun Luo, Sen Dong and Tao Li

State Key Laboratory of Public Big Data, College of Computer Science and Technology, Guizhou University, Guiyang 550025, China; gs.yytang21@gzu.edu.cn (Y.T.); gs.yunluo20@gzu.edu.cn (Y.L.); midmountain@sina.com (S.D.); litao_2019@qfnu.edu.cn (T.L.)

* Correspondence: ylchen3@gzu.edu.cn

Abstract: Public key encryption with keyword search (PEKS) allows users to perform keyword searches of ciphertext on untrusted cloud storage servers, protecting data privacy while sharing data. However, it faces several security problems in practical applications. First, an attacker can launch a keyword guessing attack to obtain keywords of interest to users, causing the leakage of their sensitive information. Second, untrusted cloud servers may return incorrect or incomplete results. In addition, with the continuous development of quantum computers, existing PEKS schemes face the problem of quantum attacks. Since cloud servers are mostly untrusted, verifiable search has become a hot research topic among scholars. However, most of the current schemes are based on bilinear pairing constructions, which are vulnerable to quantum attacks. To solve these problems, we propose a new ciphertext retrieval scheme based on fully homomorphic encryption (FHE), called VR-PEKS. This scheme implements verifiable search and is able to solve the problems of keyword guessing attacks and quantum attacks. We propose to improve the security of the scheme by using the oblivious pseudorandom function to randomize keywords and then encrypt them using FHE. An encrypted verified index structure is constructed and exposed by the data owner, enabling the data recipient to achieve verification of the correctness and integrity of the retrieved results without relying on a trusted third party. We demonstrate the security of the proposed scheme in a stochastic prediction model, and prove that our scheme satisfies keyword ciphertext indistinguishability and keyword trapdoor indistinguishability under adaptive keyword selection attacks. The comparison shows that our scheme is secure and feasible.

Keywords: cloud storage; public key encryption with keyword search; fully homomorphic encryption; keyword guess attack; verifiable

check for updates

Citation: Tang, Y.; Chen, Y.; Luo, Y.; Dong, S.; Li, T. VR-PEKS: A Verifiable and Resistant to Keyword Guess Attack Public Key Encryption with Keyword Search Scheme. *Appl. Sci.* **2023**, *13*, 4166. https://doi.org/10.3390/app13074166

Academic Editors: Peng Jiang and Jingwei Li

Received: 8 March 2023
Revised: 23 March 2023
Accepted: 23 March 2023
Published: 24 March 2023

1. Introduction

In the era of rapid Internet development, people are increasingly concerned about the privacy of their personal information [1,2]. In order to reduce the pressure of local storage and protect users' privacy, data are usually encrypted and uploaded to cloud servers for storage. However, the issue of how to retrieve ciphertext data efficiently is a challenging problem. The emergence of Searchable Encryption (SE) [3] provides a solution for ciphertext retrieval, which can effectively solve the problem of privacy leakage caused by storing data in plaintext and the difficulty of retrieving ciphertext caused by encrypting the data. SE is an encryption primitive that enables users to search ciphertext data securely by keyword. SE can be divided into symmetry searchable encryption (SSE) and public key encryption with keyword search (PEKS). SSE uses a symmetric encryption algorithm, which requires complex key management and distribution when expanding to multi-user scenarios; thus, it is mainly applicable to single-user scenarios and not conducive to multi-user data sharing [4]. Therefore, Boneh et al. [5] proposed PEKS technology, which can simultaneously realize data sharing and ciphertext retrieval.

PEKS technology can effectively protect the privacy of sensitive user information. It can be applied to cloud healthcare services, enabling the sharing of electronic medical records among multiple hospitals or departments within the same hospital. When a patient visits a hospital, the hospital extracts key information, such as their name, gender, and disease name, to construct a medical record index using these keywords. The hospital then stores the index and encrypted medical records on a cloud server. When Hospital A needs to share medical records with Hospital B, it uses B's public key to encrypt the keywords and medical files and creates a medical record index. These contents are then packaged and uploaded to the cloud server for storage. B's hospital staff can generate keyword traps using a private key and send them to the cloud server for searching. Once the cloud server completes the search, it returns the matching encrypted files. After decryption by B's hospital staff, they can obtain the plaintext files of the patient's medical records.

However, Andola et al. [6] stated that the keyword guess attack (KGA) in the PEKS scheme is still an unresolved problem due to the low entropy property of keywords. In KGA, an attacker intercepts the keyword search trapdoor of the data receiver, encrypts guessed keywords with the receiver's public key, and matches the ciphertext with the trapdoor to obtain the keywords of interest to the user, leading to a serious privacy violation [7]. KGA can be divided into internal KGA and external KGA. External KGA means that the attacker is a malicious party unrelated to the cloud service provider, and they obtain the trap gate by eavesdropping on the public channel between the cloud server and the receiver. In internal KGA, the attacker is a semi-trusted or malicious cloud server, and they can obtain the trapdoor directly from the receiver and can execute $Test(W, T_W)$ to check whether the keyword matches the trapdoor. Therefore, the attack capability of the internal adversary is more powerful. Many researchers have proposed different protocols to resist KGA [8–11].

Although SE can guarantee the privacy of sensitive data, risks due to hardware failures, network attacks, and the system vulnerabilities of cloud servers still exist [12,13]. In particular, malicious cloud servers have powerful attack capabilities. As Li et al. [14] conclude, attackers with more power have no incentive to attack. Malicious cloud servers are not only capable of performing KGA, but may also perform actions such as intentionally dropping, corrupting, or tampering with cloud data, or incorrectly performing search operations. Therefore, it is necessary to introduce a verification mechanism in PEKS schemes in order to ensure the correctness of the search results and detect the dishonest or malicious behavior of cloud servers. Therefore, some scholars have begun to study verifiable keyword search schemes [15–17].

However, with the continuous development of quantum computers, the above PEKS schemes constructed based on traditional number theory and hard problem assumptions are vulnerable to quantum attacks [18]. Gentry's fully homomorphic encryption (FHE) scheme [19] based on the lattice hard problem is resistant to quantum computer attacks, and homomorphic addition and homomorphic multiplication operations can be used for the computation of ciphertexts. Akavia et al. [20,21] and Wen et al. [22] studied the search for FHE encrypted data.

By considering the KGA problem prevalent in PEKS cryptosystems, the verification problem of untrusted cloud servers returning incomplete results, and the quantum attack problem, in this paper, we propose a new ciphertext retrieval scheme based on FHE, called VR-PEKS. Our main contributions are as follows.

- We propose a new PEKS scheme based on FHE, called VR-PEKS, and make the scheme resistant to keyword guessing attacks by internal and external adversaries by using the oblivious pseudorandom function (OPRF) to blind keywords. The OPRF keys are securely stored and used by data owners and data users, so that malicious cloud servers cannot generate a valid keyword trapdoor for *Test* algorithm.
- We design an encrypted authentication index structure, which is created and disclosed by the data owner, so that the data user can verify the correctness and integrity of the search results, so as to prevent the malicious cloud server from forging, tampering with, or discarding the stored cloud data, or performing the search task incorrectly.

- In the random prediction model, we prove that our scheme satisfies keyword ciphertext indiscriminability and keyword trapdoor indiscriminability under an adaptive keyword selection attack, and compare the security of the scheme with that of other PEKS schemes.

The rest of this paper is organized as follows. In Section 2, the work related to PEKS is presented. In Section 3, the relevant background knowledge that is used in this paper is presented. Section 4 presents the system model, scheme definition, and security model of this scheme. Section 5 describes the proposed scheme in this paper in detail and proves the security and correctness of the scheme. In Section 6, the proposed scheme in this paper is compared and analyzed with other PEKS schemes. Finally, Section 7 concludes this paper.

2. Related Work

Boneh et al. [5] initially proposed a PEKS scheme based on bilinear mapping. However, subsequent studies have shown that the scheme has serious security problems, such as the need for secure channels to transmit trapdoors [23], keyword guessing attacks [7], and other problems. To address these problems, several researchers have proposed improvement schemes. Baek et al. [23] designed an SCF-PEKS scheme that transmits keyword trapdoors without a secure channel. Tang et al. [24] introduced the concept of registered keywords and required the sender to register the keyword with the receiver before generating the keyword ciphertext. Rhee et al. [25] introduced the concept of trapdoor indistinguishability and proposed a public key searchable encryption scheme (dPEKS) with a designated tester. Recently, Li et al. [26] proposed a hierarchical PEKS scheme (dDHPEKS) with decodable encryption for designated testers to satisfy security against external keyword guessing attacks. However, due to the powerful attack capabilities of cloud servers, the above schemes cannot address internal KGAs from malicious servers. Therefore, researchers have started to investigate PEKS schemes that can resist internal KGAs. Xu et al. [9] constructed a public key encryption with fuzzy keyword search scheme (PEFKS), where each keyword corresponds to an exact keyword search trapdoor and a fuzzy keyword search trapdoor, which is effective against internal KGA but has a high communication overhead. Chen et al. [10] proposed a server-assisted scheme (SA-PEKS). To solve the problem of internal KGA, some researchers have introduced the concept of public key authenticated encryption for keyword search (PAEKS). In 2021, Pan et al. [27] proposed a new public key authenticated encryption with keyword search scheme that achieves both multi-ciphertext and multi-trapdoor indistinguishability. Qin et al. [28] introduced an improved cipher-keyword (CI security) model for PAEKS to guarantee the indistinguishability of multiple cipher-keywords in a multi-user environment. Cheng et al. [11] proposed a certificateless public key authentication encryption with keyword search scheme (CLPAEKS), which is free of certificates and key management, while solving internal keyword attacks.

In a system, an effective mechanism or policy is needed to ensure correct operation [29]. Verifiable keyword search is a technique used to ensure the accuracy of retrieval results and detect the dishonest or malicious behavior of cloud servers. Several researchers have proposed different verifiable keyword search schemes, such as Zheng et al.'s verifiable attribute-based keyword search (VABKS) [30] and Sun et al.'s efficient verifiable connected keyword search (VCKS) for encrypted cloud data [31]. Chen et al. [32] also developed a verifiable keyword search scheme with fine-grained authorization control using reversible Bloom lookup tables and Merkle hash trees. However, most of these schemes are vulnerable to keyword guessing attacks (KGA). To address this issue, Miao et al. [33] created a basic verifiable search framework (VSEF) that solves both verifiable search and internal KGA problems.

3. Preliminaries

3.1. Searchable Encryption

Searchable encryption is divided into symmetric searchable encryption (SSE) and public key encryption with keyword search (PEKS). In this paper, we focus on PEKS.

PEKS, proposed by Boneh et al., is an asymmetric cryptosystem that enables the keyword-based retrieval of encrypted data. It involves the generation of a public key and a private key by the data receiver, the encryption of a keyword by the data owner, the generation of a trapdoor by the data receiver, and the testing of the keyword by the cloud server.

PEKS consists of four algorithms: *keyGen*, *Enc*, *Trapdoor*, and *Test*.

(1) $(pk, sk) \leftarrow KeyGen(\lambda)$: *KeyGen* is used by the data receiver to generate public and private keys (pk, sk) with a security parameter λ.

(2) $C_w \leftarrow Enc(pk, w)$: *Enc* is used by the data sender to encrypt a keyword and generate a ciphertext C_w for the keyword.

(3) $T_w \leftarrow Trapdoor(sk, w)$: *Trapdoor* is used by the data receiver to generate a trapdoor T_w for a given keyword w.

(4) $b \leftarrow Test(pk, C_w, T_{w'})$: *Test* is used by the cloud server to test whether the trapdoor $T_{w'}$ for a given keyword w' corresponds to the same keyword as the keyword ciphertext C_w in the index, i.e., whether w is equal to w'.

3.2. BFV

Fully homomorphic encryption(FHE) [19,34,35] supports the calculation of ciphertext without decrypting it. The calculation result is also saved and transmitted in ciphertext, and the result of ciphertext decryption is the same as that of plaintext computation, i.e., $f(Enc(m)) = Enc(f(m))$. In the BFV scheme [34], the ciphertext consists of polynomials in the ring R_q. The plaintext is a polynomial in the ring R_t, and the BFV encryption scheme relies on several system parameters, including d, q, t, and σ, to ensure both correctness and security. These parameters are chosen carefully to balance the level of security with the efficiency of the encryption process.

The BFV encryption scheme includes the following algorithms.

(1) $(pk, sk) \leftarrow BFV.KeyGen(1^\lambda)$: Randomly sample $\mathbf{s} \overset{R}{\leftarrow} R_2$ and set the private key $sk = \mathbf{s}$. Then, randomly sample $\mathbf{pk}_1 \overset{R}{\leftarrow} R_q$ and a noise vector $\mathbf{e} \overset{R}{\leftarrow} \chi$. Compute and obtain the public key $pk = (\mathbf{pk}_0, \mathbf{pk}_1) = ([-(\mathbf{pk}_1 \cdot \mathbf{s} + \mathbf{e})]_q, \mathbf{pk}_1)$.

(2) $ct \leftarrow BFV.Enc(pk, \mathbf{m})$: To encrypt a message $\mathbf{m}$ that belongs to the polynomial ring R_t, random vectors $\mathbf{u}$, $\mathbf{e}_1$, and $\mathbf{e}_2$ are sampled uniformly at random from the sets R_2 and χ, respectively. Let $\triangle = \lfloor q/t \rfloor$, and compute

$$CT = (CT[0], CT[1]) = ([\mathbf{pk}_0 \cdot \mathbf{u} + \mathbf{e}_1 + \triangle \cdot \mathbf{m}]_q, [\mathbf{pk}_1 \cdot \mathbf{u} + \mathbf{e}_2]_q)$$

(3) $BFV.Dec(CT)$: Compute

$$\left[\left\lfloor \frac{t \cdot [CT[0] + CT[1] \cdot sk]_q}{q} \right\rceil\right]_t$$

(4) $BFV.Add(CT_1, CT_2)$: The *Add* algorithm adds up the ciphertexts CT_1 and CT_2, returning

$$([CT_1[0] + CT_2[0]]_q, [CT_1[1] + CT_2[1]]_q)$$

(5) $BFV.Mul(CT_1, CT_2, rlk)$: The *Mul* algorithm multiplies the corresponding terms of the two ciphertexts CT_1 and CT_2 and then reduces the ciphertext using the relinearized key rlk.

3.3. Oblivious Pseudorandom Function

Freedman et al. [36] introduced the oblivious pseudorandom function (OPRF). The sender privately chooses a key k. OPRF allows the receiver to combine its own input information x with the sender's key k, which is converted into the corresponding number after a series of operations, and the receiver can learn $F_k(x)$. In this process, the receiver cannot know the receiver's k, and the sender does not know the final result $F_k(x)$. Each input x_i can be calculated differently from the other inputs, and these numbers can then be considered as pseudorandom numbers.

Stanislaw Jarecki et al. [37] proposed an OPRF protocol based on the Diffie–Hellman assumption. It computes $OPRF_k(x) = H'(H(x)^k)$, where H is a random prediction function in the range of values in the group $\mathbb{Z}_q^*$. Let G be a cyclic group of order q. The sender samples the key k from $\mathbb{Z}_q^*$. Then, the receiver inputs $x \in \{0,1\}^*$. The receiver first randomly chooses $k' \leftarrow \mathbb{Z}_q^*$, and then sends $H(x)^{k'}$ to the sender, who replies $\left(H(x)^{k'}\right)^k$ to the receiver. The receiver can output $H'\left(H(x)^k\right)$, where H' is used to map the group elements to a sufficiently long bit string.

3.4. FHE-Based Secure Search

Akavia et al. proposed the definition of *Secure Search* [20], i.e., using FHE to search encrypted data. The client encrypts the data through FHE and uploads them to the cloud server for storage. When it needs to retrieve the data, it sends an encrypted query request $[\![q]\!]$. Due to the complete homomorphism of FHE, the server can correctly perform the search. First, the client runs the homomorphic encryption key generation algorithm to generate a public key pk and a private key sk. The client publishes the public key pk, while saving the private key sk. Next, the client uploads n encrypted items $x = (x_1, x_2, \cdots, x_n)$ to the server using the public key pk. The encrypted data stored in the server are denoted as $[\![x]\!] = ([\![x_1]\!], [\![x_2]\!], \cdots, [\![x_n]\!])$. When the client wants to make a query, they send an encrypted query $[\![q]\!]$ to the server. The server then performs homomorphic evaluation on each record $[\![x_i]\!]$ to obtain the encrypted matching result $[\![b]\!] = ([\![b_1]\!], [\![b_2]\!], \cdots, [\![b_n]\!])$. If record $[\![x_i]\!]$ satisfies the query $[\![q]\!]$, then b_i is 1; otherwise, it is 0. After computing $[\![b]\!]$, the server can then fetch the matching record by homomorphically computing $[\![i^*]\!]$ and obtaining $[\![x_i]\!]$. The server sends $([\![i^*]\!], [\![x_{i^*}]\!])$ to the client for decryption. Here, i^* corresponds to the index of the first matching record.

The bottleneck of the FHE-based security search framework is the homomorphic multiplication in the acquisition step. Homomorphic multiplication in [20] was $O(nlog^2n)$, which was subsequently optimized to $O(n \log n)$ in [21]. Wen et al. [22] proposed a new $LEAF$ protocol using three methods of positioning, extraction, and reconstruction, which reduced the homomorphic multiplication times to $O(n)$, which is more advantageous for weak power device deployment. The LEAF protocol is used in the scheme proposed in this paper.

4. VR-PEKS Scheme and Security Definition

4.1. System Model

Our VR-PEKS scheme consists of four different entities, namely a key generation center (KGC), a data sender (DS), a data receiver (DR), and a cloud server (CS). Figure 1 shows the relationships and interactions between the entities.

(1) Key Generation Center (KGC): This is a trusted third party. It is responsible for generating the system parameters and the keys of the sender and receiver, including the public–private key pair (pk, sk) of FHE to encrypt keywords and documents, and an OPRF key k.

(2) Data Sender (DS): The DS encrypts his document using FHE encryption algorithm, $Enc(F, pk)$. In addition, he extracts the associated keywords $W = \{w_1, w_2, \cdots, w_m\}$ from each document and encrypts these keywords, $Enc(W, pk, k)$, and generates an encrypted keyword index. Then, the encrypted file data and the searchable index are sent to the cloud server for storage. Finally, he generates an encrypted verified index structure to make it public for the receiver to verify the search results.

(3) Data Receiver (DR): The DR sends the keyword trapdoor to the cloud server to search for his interested keywords. After obtaining the search result, he verifies it locally and decrypts it to acquire the plaintext file.

(4) Cloud Server (CS): This can be an untrusted entity. It has powerful data storage and computing capabilities to provide storage and search services for users.

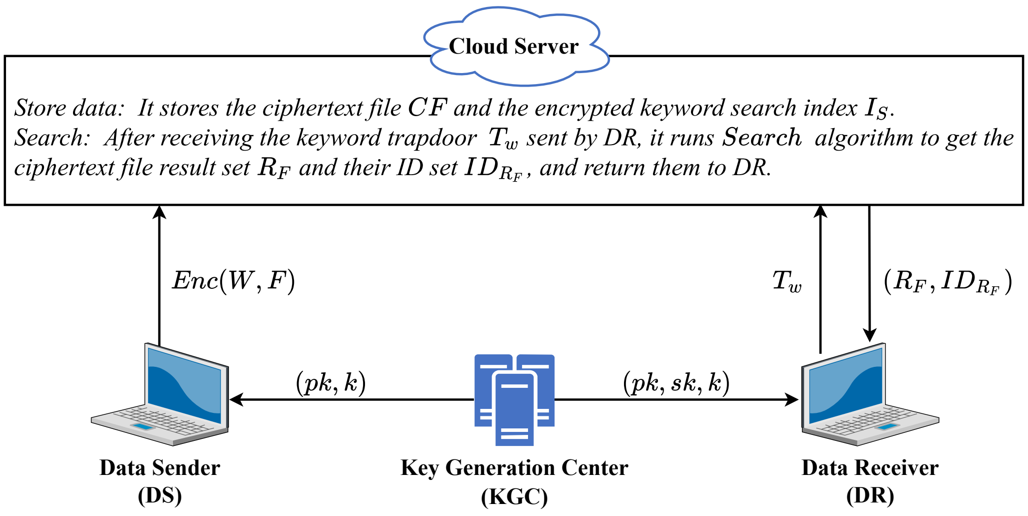
Cloud Server
Store data: It stores the ciphertext file CF and the encrypted keyword search index I_S.
Search: After receiving the keyword trapdoor T_w sent by DR, it runs $Search$ algorithm to get the ciphertext file result set R_F and their ID set ID_{R_F}, and return them to DR.
$Enc(W, F)$
T_w
(R_F, ID_{R_F})
(pk, k)
(pk, sk, k)
Data Sender (DS)
Key Generation Center (KGC)
Data Receiver (DR)

Figure 1. System model for the proposed VR-PEKS scheme.

4.2. Scheme Definition

There are six polynomial time algorithms in the VR-PEKS scheme, described as follows.

(1) $(pk, sk, k) \leftarrow KeyGen(1^\lambda)$: The algorithm is executed by KGC. On inputting a security parameter λ, it outputs public and private keys (pk, sk) of the FHE encryption algorithm and a key k of $OPRF$.

(2) $(CF, I_S, I_V) \leftarrow Enc(pk, k, W, F)$: The algorithm is executed by the DS. On input keys (pk, k), keywords $W = \{w_1, w_2, \cdots, w_m\}$, and plaintext document set $F = \{F_1, F_2, \cdots, F_n\}$, it outputs encrypted data file set CF, a searchable index I_S, and a verification index I_V.

(3) $T_{\mathbf{w}} \leftarrow GenTrapdoor(pk, \mathbf{w}, k)$: The algorithm is executed by DR. On input public key pk, key k, and keyword $\mathbf{w}$, it outputs keyword trapdoor $T_{\mathbf{w}}$.

(4) $(R_F, ID_{R_F}) \leftarrow Search(T'_{\mathbf{w}}, I_S)$: The algorithm is executed by CS. On input searchable index I_S and the trapdoor $T'_{\mathbf{w}}$ of keyword $\mathbf{w}'$, it runs this algorithm to test whether the keyword ciphertexts $CT_{\mathbf{w}}$ and $T'_{\mathbf{w}}$ correspond to the same keyword. If $\mathbf{w} = \mathbf{w}'$, it returns the corresponding ciphertext file result set R_F and the file name set ID_{R_F}.

(5) $Verify(R_F, ID_{R_F}, I_V)$: The algorithm is executed by DR. On input ciphertext file result set R_F, file name set ID_{R_F}, and encrypted verification index I_V, it outputs the file validation result. If it is true, it returns 1; otherwise, the result is 0.

(6) $Dec(R_F, sk)$: The algorithm is executed by DR. On input ciphertext file result set R_F and private key sk, it outputs the plaintext of R_F.

4.3. Security Model

Secure cryptographic schemes generally have keyword ciphertext indistinguishability under adaptive selection attacks, i.e., an adversary cannot determine with more than $1/2$ probability, given any two plaintexts and the ciphertext of any one of them, which plaintext the given ciphertext is generated from. There are two main types of adversaries in the PEKS scheme, namely external adversary $\mathcal{A}_1$ and internal adversary $\mathcal{A}_2$. $\mathcal{A}_1$ can intercept the trapdoor from the public channel between CS and DR, and then perform KGA using DR's public key. $\mathcal{A}_2$ is a more aggressive adversary than $\mathcal{A}_1$, which possesses the ciphertext of files and keyword trapdoors, and is able to execute search algorithms. An adversary can obtain the keyword trapdoor by some means and can use the data user's public key to generate the ciphertext of his guessed plaintext keyword, thus conducting a keyword guessing attack and causing the violation of the user's privacy. Therefore, in the PEKS scheme, it is also necessary to ensure the indistinguishability of having keyword trapdoors. It follows that a PEKS scheme that is secure and resistant to external and internal keyword guessing attacks should satisfy keyword ciphertext indistinguishability under an adaptive selection attack (KC-IND-CKA) and keyword trapdoor indistinguishability under an adaptive selection attack (KT-IND-CKA). KC-IND-CKA and KT-IND-CKA security is defined by the following two games.

4.3.1. KC-IND-CKA Security

The KC-IND-CKA security is defined by the following interactive game between adversary $\mathcal{A}$ and challenger $\mathcal{C}$.

(1) The challenger $\mathcal{C}$ inputs a security parameter λ and calls the *KeyGen* algorithm to generate DR's key pair (pk, sk) and key k. Then, he sends pk to adversary $\mathcal{A}$.

(2) The challenger $\mathcal{C}$ generates a keyword ciphertext CT_w for keyword w chosen by the adversary $\mathcal{A}$ and sends it to $\mathcal{A}$.

(3) The challenger $\mathcal{C}$ generates the keyword trapdoor T_w for keyword w chosen by the adversary $\mathcal{A}$ and sends it to $\mathcal{A}$.

(4) Adversary $\mathcal{A}$ sends two keywords w_0, w_1 to challenger $\mathcal{C}$ that he wishes to challenge, with the restriction that adversary $\mathcal{A}$ has not requested trapdoor T_{w_0} or T_{w_1} before. After receiving the keywords, the challenger $\mathcal{C}$ chooses a random bit $b \in \{0,1\}$ and sends the ciphertext CT_{w_b} to adversary $\mathcal{A}$.

(5) The adversary $\mathcal{A}$ can continue to adaptively choose any keyword w' to request its ciphertext $CT_{w'}$ and trapdoor $T_{w'}$ as long as $w' \neq w_0, w_1$.

(6) The adversary $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$ of the keyword ciphertext. When $b' = b$, the adversary $\mathcal{A}$ wins the game. The possibility that adversary $\mathcal{A}$ can win the game is defined as

$$Adv_{\mathcal{A}}^{KC-IND-CKA}(\lambda) = \left| \Pr\left[b' = b\right] - \frac{1}{2} \right|$$

4.3.2. KT-IND-CKA Security

The KT-IND-CKA security is defined by the following interactive game between adversary $\mathcal{A}$ and challenger $\mathcal{C}$.

(1) The challenger $\mathcal{C}$ inputs a security parameter λ and calls the *KeyGen* algorithm to generate DR's key pair (pk, sk) and key k. Then, he sends pk to adversary $\mathcal{A}$.

(2) The adversary $\mathcal{A}$ can perform the queries in steps (2) and (3) of the KC-IND-CKA game.

(3) The adversary $\mathcal{A}$ sends two keywords w_0, w_1 to challenger $\mathcal{C}$ that he wishes to challenge, with the restriction that $\mathcal{A}$ has not previously requested trapdoor T_{w_0} or T_{w_1}. After receiving the keywords, $\mathcal{C}$ chooses a random bit $b \in \{0,1\}$ and sends the trapdoor T_{w_b} to $\mathcal{A}$.

(4) The adversary $\mathcal{A}$ can continue to adaptively choose any keyword w' to request its ciphertext $CT_{w'}$ and trapdoor $T_{w'}$ as long as $w' \neq w_0, w_1$.

(5) The adversary $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$ of the keyword trapdoor. When $b' = b$, the adversary $\mathcal{A}$ wins the game. The possibility that adversary $\mathcal{A}$ can win the game is defined as

$$Adv_{\mathcal{A}}^{KT-IND-CKA}(\lambda) = \left| \Pr\left[b' = b\right] - \frac{1}{2} \right|$$

5. The Proposed VR-PEKS Scheme

In this section, we describe the proposed VR-PEKS scheme and prove its security and correctness.

5.1. Construction of the Scheme

We describe the construction of VR-PEKS based on the fully homomorphic encryption BFV scheme [34]. The working process is as follows: KGC runs the *KeyGen* algorithm to generate the public–private key pair (pk, sk) of BFV and key k of $OPRF$. Then, it securely sends (pk, k) and (pk, sk, k) to DS and DR, respectively. DS encrypts the file and keywords using the BFV encryption algorithm and creates a searchable index and an encrypted verified index, $(CF, I_S, I_V) \leftarrow Enc(pk, k, W, F)$. Then, he uploads the ciphertext file CF and searchable index I_S to the CS for storage, and exposes the encrypted verification index I_V. We use pk and k to encrypt document keywords, which has two benefits. Firstly, after pre-processing by the OPRF, we can use highly optimized FHE parameters during the search operation and do not need to worry about noise flooding because OPRF already provides sufficient protection. Secondly, CS and external adversaries are not aware of the key k, making them unable to generate a valid keyword search trapdoor. Therefore, the scheme is able to defend against KGA by malicious cloud servers and external adversaries. When DR wants to retrieve files, he runs the GenTrapdoor algorithm to generate a valid keyword trapdoor and sends it to CS. Then, CS runs a search algorithm to search the corresponding ciphertext files and returns the result to DR. DR verifies the retrieval results locally, and decrypts them to obtain plaintext data after verification is passed. The schematic flow of the algorithm is shown in Figure 2.

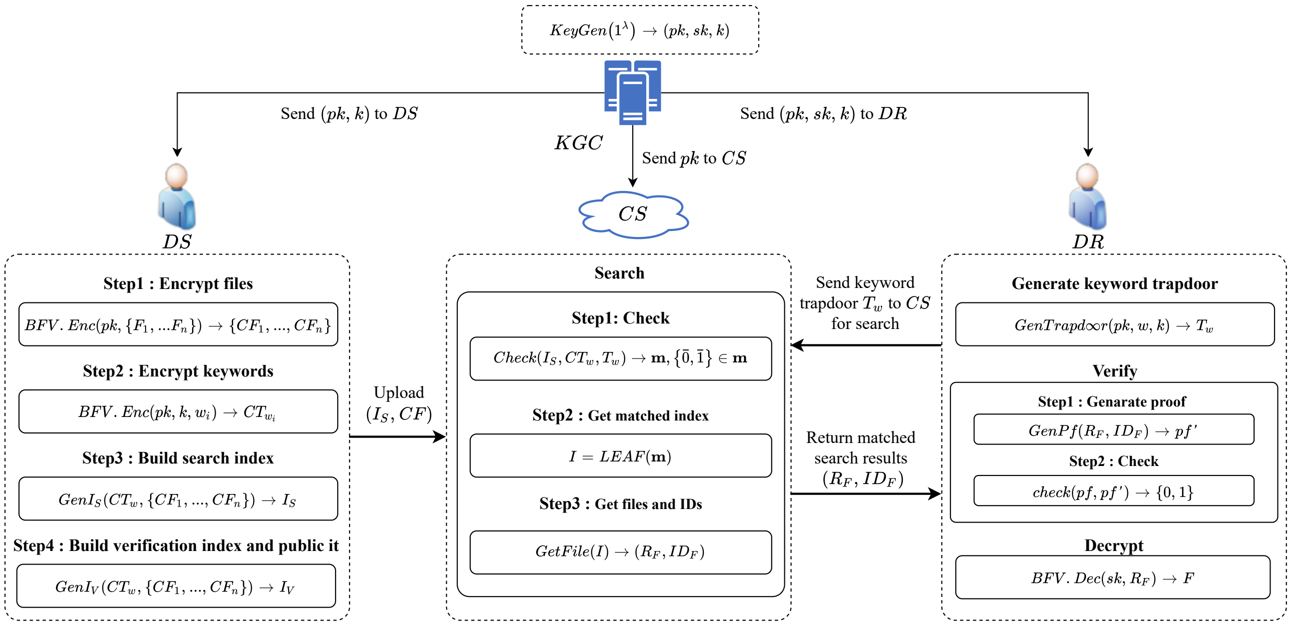
$KeyGen(1^\lambda) \rightarrow (pk, sk, k)$
Send (pk, k) to DS
Send (pk, sk, k) to DR
KGC
Send pk to CS
CS
DS
DR
Step1 : Encrypt files
$BFV.Enc(pk, \{F_1, ...F_n\}) \rightarrow \{CF_1, ..., CF_n\}$
Step2 : Encrypt keywords
$BFV.Enc(pk, k, w_i) \rightarrow CT_{w_i}$
Step3 : Build search index
$GenI_S(CT_w, \{CF_1, ..., CF_n\}) \rightarrow I_S$
Step4 : Build verification index and public it
$GenI_V(CT_w, \{CF_1, ..., CF_n\}) \rightarrow I_V$
Upload (I_S, CF)
Search
Step1: Check
$Check(I_S, CT_w, T_w) \rightarrow \mathbf{m}, \{\bar{0}, \bar{1}\} \in \mathbf{m}$
Step2 : Get matched index
$I = LEAF(\mathbf{m})$
Step3 : Get files and IDs
$GetFile(I) \rightarrow (R_F, ID_F)$
Send keyword trapdoor T_w to CS for search
Return matched search results (R_F, ID_F)
Generate keyword trapdoor
$GenTrapdoor(pk, w, k) \rightarrow T_w$
Verify
Step1 : Genarate proof
$GenPf(R_F, ID_F) \rightarrow pf'$
Step2 : Check
$check(pf, pf') \rightarrow \{0, 1\}$
Decrypt
$BFV.Dec(sk, R_F) \rightarrow F$

Figure 2. Algorithm flow chart of the proposed VR-PEKS scheme.

The VR-PEKS algorithm is as follows.

(1) $(pk, sk, k) \leftarrow KeyGen(1^\lambda)$: The algorithm is executed by KGC to generate the keys required by the system. Given security parameter λ, KGC runs BFV's key generation algorithm [34] to obtain the public key pk and private key sk. It randomly samples $\mathbf{s} \leftarrow \chi$, lets private key $sk = \mathbf{s}$, and samples $\mathbf{a} \leftarrow R_2$, $\mathbf{e} \leftarrow \chi$, and then computes the public key $pk = ([-(\mathbf{a} \cdot \mathbf{s}) + \mathbf{e}]_q, \mathbf{a})$. In addition, KGC needs to sample a key k for OPRF $F : \{0,1\}^* \longrightarrow \{0,1\}^\kappa$. Then, KGC sends securely (pk, k) to DS and sends (sk, pk, k) to DR. DR discloses pk.

(2) $(CF, I_S, I_V) \leftarrow Enc(pk, k, W, F)$: The algorithm is executed by DS to encrypt files and keywords and create indexes. DS encrypts files $F = \{F_1, F_2, \cdots, F_n\}$ using a BFV encryption algorithm, generates ciphertext file set $CF = \{CF_1, CF_2, \cdots, CF_n\}$, encrypts keywords $W = \{w_1, w_2, \cdots, w_m\}$ associated with the file set, and establishes keyword searchable index I_S and encrypted verification index I_V. Finally, DS uploads the ciphertext file set CF and searchable index I_S to CS for storage, and makes the encrypted verification index I_V publicly available so that DR can use it to verify the correctness and integrity of the search results. The specific steps are as follows.

(a) Generate encrypted files: Given pk and plaintext file set $F = \{F_1, F_2, \cdots, F_n\}$, DS generates the set of ciphertext files

$$CF \leftarrow BFV.Enc(pk, F)$$

Sets $\mathbf{p}_0 = pk[0]$, $\mathbf{p}_1 = pk[1]$, samples $\mathbf{u}_1, \mathbf{e}_1, \mathbf{e}_2 \leftarrow \chi$, for each message $\mathbf{m}$ in F, generate ciphertext

$$CT = ([\mathbf{p}_0 = pk[0] \cdot \mathbf{u}_1 + \mathbf{e}_1 + \triangle \cdot \mathbf{m}]_q, [\mathbf{p}_1 \cdot \mathbf{u}_1 + \mathbf{e}_2]_q)$$

(b) Generate keyword ciphertext $CT_{\mathbf{w}}$: Given a t-bit keyword $\mathbf{w} = w_1 w_2 \cdots w_t$, DS first pseudorandomizes $\mathbf{w}$ using the OPRF function: $\mathbf{w}' = F_k(\mathbf{w})$. Then, he encodes $\mathbf{w}'$ as a polynomial $\mathbf{w}'$ in $Z_2[x]/(x^d + 1)$, where d is a parameter in the BFV encryption scheme. Samples $\mathbf{u_2}, \mathbf{e}_3, \mathbf{e}_4 \leftarrow \chi$. DS generates keyword searchable ciphertext

$$CT_{\mathbf{w}} = BFV.Enc\left(\left[\mathbf{p}_0 \cdot \mathbf{u_2} + \mathbf{e}_3 + \triangle \cdot \mathbf{m}'\right]_q, [\mathbf{p}_1 \cdot \mathbf{u_2} + \mathbf{e}_4]_q\right)$$

(c) Build the search index I_S: Assume that the outsourced ciphertext file set CF contains n encrypted data files, which are associated with m keywords. DS establishes a reverse searchable index I_S. This allows him to place ciphertext files associated with the same keywords on one line. Given keywords $w_i (i = 1, \cdots, m)$ and their associated file set $F_j (j = 1, \cdots, n)$, DR builds encrypted keyword index I_S, and the index structure is shown in Figure 3.

(d) Build and expose verification index I_V: DS constructs an encrypted verification index I_V, as shown in Figure 4. I_V is a two-dimensional table with $m \times (n+2)$. The first column stores the keyword ciphertext $CT_{w_i} (i = 1, \cdots, m)$. The $2 \sim (n+1)$ columns store size $L_{ij} = Enc(CF_j.length)(i = 1, \cdots, m, j = 1, \cdots, n)$ of encrypted files $CF_j (j = 1, \cdots, n)$. If ciphertext file CF_j is not associated with the keyword $CT_{\mathbf{w}_i}$, L_{ij} is empty. The last column stores the verification proof pf_i corresponding to the keyword ciphertext $CT_{\mathbf{w}_i}$,

$$pf_i = \Big(\sum_{j \in [1,n]} ID_{CF_j}, \sum_{j \in [1,n]} L_{ij}\Big),\ L_{ij} \neq null$$

(3) $T_{\mathbf{w}} \leftarrow GenTrapdoor(pk, \mathbf{w}, k)$: When DR wants to search the ciphertext file corresponding to t-bit keyword $\mathbf{w} = w_1 \cdots w_m$, he first uses the OPRF function to pseudorandomize $\mathbf{w}$ to obtain $\mathbf{w}' = F_k(\mathbf{w})$, and encodes $\mathbf{w}'$ as a polynomial in $Z_2[x]/(x^d + 1)$. Then, DR samples $\mathbf{u}_3, \mathbf{e}_5, \mathbf{e}_6 \leftarrow \chi$ to generate keyword trapdoor

$$T_{\mathbf{w}} = BFV.Enc\left(\mathbf{w}', pk\right) = \left(\left[\mathbf{p}_0 \cdot \mathbf{u}_3 + \mathbf{e}_5 + \triangle \cdot \mathbf{w}'\right]_q, [\mathbf{p}_1 \cdot \mathbf{u}_3 + \mathbf{e}_6]_q\right)$$

(4) $(R_F, ID_{R_F}) \leftarrow Search(T_{\mathbf{w}'}, I_S)$: When CS receives the trapdoor $T_{\mathbf{w}'}$ from DR, CS runs the *Search* algorithm to retrieve index I_S to obtain the matching ciphertext file result set R_F and file name set ID_{R_F}. Then, it returns the result to DR. The specific steps are as follows.

(a) Determine whether the keyword ciphertext $CT_{\mathbf{w}_i}$ matches the trapdoor $T_{\mathbf{w}'}$. We use the exact match function to determine whether the search trapdoor matches the keyword ciphertext and obtain the matching array $\mathbf{m}$. The encrypted 1s or 0s are stored in $\mathbf{m}$, denoted by $\bar{1}$ and $\bar{0}$. $\bar{1}$ indicates that the trapdoor $T_{\mathbf{w}'}$ corresponds to the same keyword as the keyword ciphertext $CT_{\mathbf{w}_i}$, $\bar{0}$ on the contrary. For $CT_{\mathbf{w}_i}, T_{\mathbf{w}'} \in \{0,1\}^m$,

$$\mathbf{m}[i] = IsEqual(CT_{\mathbf{w}_i}, T_{\mathbf{w}'}) = \prod_{i \in [m]} (1 + CT_{\mathbf{w}_i}[i] + T_{\mathbf{w}'}[i])\ mod\ 2,\ i = 1, 2, \cdots, m$$

(b) Retrieve indexes securely. We use the LEAF protocol [22] to securely obtain the matching index $I = LEAF(\mathbf{m})$.

(c) Obtain related ciphertext file set R_F and file name set ID_{R_F} according to index I. Then, send R_F and ID_{R_F} to DR.

(5) $Verify(R_F, ID_{R_F}, I_V)$: When DR receives the retrieval result R_F, ID_{R_F} from CS, he completes following verification steps.

(a) Generate verification certificate

$$pf' = -(\sum ID_{CF}, \sum L_{CF}),\ CF \in R_F$$

where ID_{CF} is the name of each ciphertext file in R_F and L_{CF} is the size of each ciphertext file.

(b) Check $pf + pf' = 0$, If it is equal, it outputs 1. Otherwise, it outputs 0 and discards the files.

(6) $Dec(CF, sk)$: DR uses private key sk to decrypt the ciphertext file result set R_F and obtains the corresponding file data in plaintext. Here, $\mathbf{s} = sk$, $\mathbf{c}_0 = ct[0]$, $\mathbf{c}_1 = ct[1]$, and he computes

$$\left[\left\lfloor \frac{t \cdot [\mathbf{c}_0 + \mathbf{c}_1 \cdot \mathbf{s}]_q}{q} \right\rceil\right]_t$$

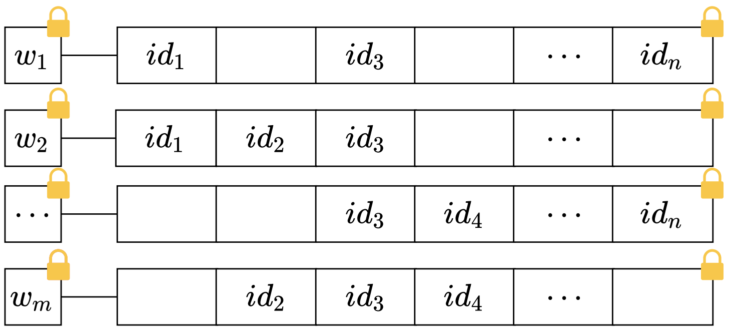

Figure 3. Reverse encrypted keyword index structure.

		1	2	$\cdots$	n	$n+1$
1	CT_{w_1}	$Enc(20)$		$\cdots$	$Enc(30)$	pf_1
2	CT_{w_2}	$Enc(10)$	$Enc(15)$	$\cdots$	$Enc(11)$	pf_2
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
m	CT_{w_m}		$Enc(101)$	$\cdots$	$Enc(201)$	pf_m

Figure 4. The encrypted verification index structure.

5.2. Correctness

Correctness. For the whole scheme, if each entity performs correctly according to the algorithms, DR can generate a valid keyword trapdoor to obtain the matching ciphertext files and can verify the correctness and integrity of the result.

For keyword ciphertext $CT_{\mathbf{w}}$ and trapdoor $T_{\mathbf{v}}$,

$$\begin{aligned}
CT_{\mathbf{w}} + T_{\mathbf{v}} &= \left(\left[\mathbf{p}_0 \cdot \mathbf{u}_2 + \mathbf{e}_3 + \triangle \cdot \mathbf{w}'\right]_{q'}, \left[\mathbf{p}_1 \cdot \mathbf{u}_2 + \mathbf{e}_4\right]_q\right) \\
&\quad + \left(\left[\mathbf{p}_0 \cdot \mathbf{u}_3 + \mathbf{e}_5 + \triangle \cdot \mathbf{v}'\right]_{q'}, \left[\mathbf{p}_1 \cdot \mathbf{u}_3 + \mathbf{e}_6\right]_q\right) \\
&= \left(\left[\mathbf{p}_0 \cdot (\mathbf{u}_2 + \mathbf{u}_3) + \underbrace{(\mathbf{e}_3 + \mathbf{e}_5)}_{\text{noise}} + \underbrace{\triangle \cdot (\mathbf{w}' + \mathbf{v}')}_{\text{message}} \right]_q, \right. \\
&\quad \left. \left[\mathbf{p}_1 \cdot (\mathbf{u}_2 + \mathbf{u}_3) + \underbrace{(\mathbf{e}_4 + \mathbf{e}_6)}_{\text{noise}} \right]_q \right)
\end{aligned}$$

Since the scheme uses fully homomorphic encryption, the keyword ciphertext and trapdoor have additive homomorphism. As seen from the above equation, the result of adding $CT_{\mathbf{w}}$ and $T_{\mathbf{v}}$ ciphertext only exists in scaled ciphertext, and the form of the added result is same as that of $\mathbf{w}' + \mathbf{v}'$, and only new noise is added. $\mathbf{w}'$ and $\mathbf{v}'$ are the oblivious pseudorandom values of $\mathbf{w}$ and $\mathbf{v}$, respectively, $\mathbf{w}' = F_k(\mathbf{w}), \mathbf{v}' = F_k(\mathbf{v})$. Then, if $\mathbf{w}$ and $\mathbf{v}$ correspond to the same keyword,

$$IsEqual(CT_{\mathbf{w}}, T_{\mathbf{v}}) = \prod_{i \in [m]} (1 + CT_{\mathbf{w_i}}[i] + T_{\mathbf{v}}[i]) \; mod \; 2 \longrightarrow \bar{1}$$

Otherwise, $IsEqual(CT_{\mathbf{w}}, T_{\mathbf{v}}) \to \bar{0}$.

It is easy to find that only users who know the key k can generate a valid keyword search trapdoor. CS matches $T_{\mathbf{v}}$ with each keyword ciphertext in the search index I_S to obtain array $\mathbf{m}$. Then, CS securely retrieves the index I_S using the LEAF protocol (the protocol's correctness has been proven in [22]), and obtains the matching index to obtain the corresponding ciphertext file result set R_F and file name set ID_{R_F}.

After the data recipient receives the retrieval result, the data recipient can generate the verification proof pf' with the pre-proof pf in the verification index for calculation and determine whether the cloud server returns the correct result,

$$pf' + pf = -(\sum ID_{CF}, \sum L_{CF}) + (\sum_{j \in [1,n]} ID_{CF_j}, \sum_{j \in [1,n]} L_{ij}), \; CF \in R_F$$

By addition homomorphism, the calculation result is $\bar{0}$ or other. After decryption with the private key sk, the data receiver can determine whether the size and name of the ciphertext file set returned by the cloud server are correct.

5.3. Security

Our VR-PEKS scheme is implemented using a fully homomorphic encryption BFV scheme [34], which is implemented based on the RLWE problem. RLWE is a ring-based version of the LWE problem [38], which is defined as follows.

Definition 1 (RLWE). For security parameter λ, let $f(x)$ be a cyclotomic polynomial $\Phi_m(x)$ with $deg(f) = \varphi(m)$ depending on λ and set $R = \mathbb{Z}[x]/f(x)$. Let $q = q(\lambda) \geq 2$ be an integer. For a random element $\mathbf{s} \in R_q$ and distribution $\chi = \chi(\lambda)$ over R, denote with $A^q_{\mathbf{s},\chi}$ the distribution obtained by choosing a uniformly random element $\mathbf{a} \leftarrow R_q$ and a noise term $\mathbf{e} \leftarrow R_q$ and outputting $(\mathbf{a}, [\mathbf{a} \cdot \mathbf{s} + \mathbf{e}]_q)$. The $Decision - RLWE_{d,q,\chi}$ problem is to distinguish between the distribution $A^q_{\mathbf{s},\chi}$ and the uniform distribution $U(R_q^2)$. The $Search - RLWE_{d,q,\chi}$ problem is to find $\mathbf{s}$ by given $(\mathbf{a}, A^q_{\mathbf{s},\chi})$.

Theorem 1. *Assuming that the RLWE problem is hard and the oblivious pseudorandom function used is a random oracle function, VR-PEKS can safely resist KGA.*

Lemma 1. *Assuming that the RLWE problem is hard and the oblivious pseudorandom function used is a random oracle function, then VR-PEKS satisfies KC-IND-CKA security.*

Proof. The proof against KC-IND-CKA will be composed of the following three games. $\mathcal{A}$ is the adversary against KC-IND-CKA security. In $Game_i$, let $\mathcal{A}$ go to attack KC-IND-CKA security. The event $\mathcal{A}$ guessed correctly is defined as S_i (namely, $b' = b$).

$Game_1$. This is the game that we originally designed. The advantage of adversary $\mathcal{A}$ winning the game is

$$Adv(\mathcal{A}) = \left|\Pr\left[b' = b\right] - \frac{1}{2}\right| = \left|\Pr\left[S_1\right] - \frac{1}{2}\right|$$

$Game_2$. This is the same as $Game_1$, except that the challenge is terminated when the following events occur: $F_k(w) = F_k(w_b), w \neq w_b$. It is not difficult to find that in order to terminate the challenge, there must be an enemy $\mathcal{B}_\infty$ that can break the oblivious pseudorandom function F with a certain advantage ε_1. Therefore, according to the difference lemma, the probability of the enemy $\mathcal{A}$ guessing correctly in $Game_1$ and $Game_2$ has the following relation:

$$\left|\Pr\left[S_1\right] - \Pr\left[S_2\right]\right| \leq \varepsilon_1$$

$Game_3$. The difference between $Game_3$ and $Game_2$ lies in the means of challenging the ciphertext. Challenger $\mathcal{C}$ replaces the keyword ciphertext produced in the form $([\mathbf{p}_0 \cdot \mathbf{u} + \mathbf{e}_1 + \triangle \cdot F_k(\mathbf{w_b})]_q, [\mathbf{p}_1 \cdot \mathbf{u} + \mathbf{e}_2]_q)$ with a uniform random distribution $U(R_q^2)$. Then, $Game_2$ and $Game_3$ should be consistent, unless there is an adversary $\mathcal{B}_\in$ who can distinguish $U(R_q^2)$ and $([\mathbf{p}_0 \cdot \mathbf{u} + \mathbf{e}_1 + \triangle \cdot F_k(\mathbf{w_b})]_q, [\mathbf{p}_1 \cdot \mathbf{u} + \mathbf{e}_2]_q)$ with a non-negligible advantage (that is, to solve the RLWE problem). Thus, there is $\left|\Pr\left[S_2\right] - \Pr\left[S_3\right]\right| \leq \varepsilon_2$. Because $U(R_q^2)$ is uniformly and randomly distributed, the advantage of the adversary's correct guess in $Game_3$ is $\Pr\left[S_3\right] = 1/2$.

Finally, the adversary $\mathcal{A}$ has the advantage of winning the game as

$$Adv(\mathcal{A}) = \left|\Pr\left[S_1\right] - \frac{1}{2}\right| \leq \left|\Pr\left[S_1\right] - \Pr\left[S_2\right]\right| + \left|\Pr\left[S_2\right] - \Pr\left[S_3\right]\right| \leq \varepsilon_1 + \varepsilon_2$$

Since oblivious pseudorandom function F is a random oracle function [37], the RLWE problem is hard [34], so that ε_1 and ε_2 can be ignored. It can be concluded that $Adv(\mathcal{A})$ can be ignored. The security of KC-IND-CKA is proven. □

Lemma 2. *Assuming that the RLWE problem is hard and the oblivious pseudorandom function used is a random oracle function, then VR-PEKS satisfies KT-IND-CKA security.*

Proof. The same as above, the proof for KT-IND-CKA will consist of the following three games. The event $\mathcal{A}$ guessed correctly is defined as S_i (namely, $b' = b$).

$Game_1$. This is the game that we originally designed. The advantage of adversary $\mathcal{A}$ winning the game is

$$Adv(\mathcal{A}) = \left|\Pr\left[b' = b\right] - \frac{1}{2}\right| = \left|\Pr\left[S_1\right] - \frac{1}{2}\right|$$

$Game_2$. This is the same as $Game_2$ in KC-IND-CKA security. There is

$$\left|\Pr\left[S_1\right] - \Pr\left[S_2\right]\right| \leq \varepsilon_1$$

$Game_3$. Challenger $\mathcal{C}$ replaces the trapdoor with a uniform random distribution $U(R_q^2)$,

$$T_{\mathbf{w}_b} = ([\mathbf{p}_0 \cdot \mathbf{u}_3 + \mathbf{e}_5 + \triangle \cdot F_k(\mathbf{w}_b)]_q, [\mathbf{p}_1 \cdot \mathbf{u}_3 + \mathbf{e}_6]_q)$$

Then, $Game_2$ and $Game_3$ should be consistent, unless there is an adversary $\mathcal{B}_3$ who can distinguish $U(R_q^2)$ and $([\mathbf{p}_0 \cdot \mathbf{u} + \mathbf{e}_1 + \triangle \cdot F_k(\mathbf{w_b})]_q, [\mathbf{p}_1 \cdot \mathbf{u} + \mathbf{e}_2]_q)$ with a non-negligible advantage (that is, to solve the RLWE problem). Thus, there is $|\Pr[S_2] - \Pr[S_3]| \leq \varepsilon_3$.

Combine the above three games to obtain

$$Adv(\mathcal{A}) = \left|\Pr[S_1] - \frac{1}{2}\right| \leq |\Pr[S_1] - \Pr[S_2]| + |\Pr[S_2] - \Pr[S_3]| \leq \varepsilon_1 + \varepsilon_3$$

Since oblivious pseudorandom function F is a random oracle function [37], the RLWE problem is hard [34], so that ε_1 and ε_2 can be ignored. It can be concluded that $Adv(\mathcal{A})$ can be ignored. The security of KT-IND-CKA is proven. □

6. Comparison

In this section, we present a comparative analysis of the proposed scheme, comparing it with the classical scheme in PEKS and with recent work related to verifiable search. Our analysis focuses on the security and theoretical computational complexity of the scheme. We also perform an experimental comparison of the schemes to show their feasibility and efficiency.

6.1. Security Comparison

The security comparison is shown in Table 1. We compare our proposed scheme with the schemes BDOP-PEKS [5] and VSEF [33], and Zhang's scheme [16]. BDOP-PEKS is the classical public key searchable encryption scheme, and most of the public key searchable encryptions are based on the improvement of this scheme. However, this scheme has weak security, requires a secure channel transmission trapdoor, cannot resist keyword guessing attacks, and does not consider the verification of the retrieval results. VSEF and Zhang et al. studied verifiable search, but they only considered whether the cloud server returned the correct set of files, and only verified the file IDs without considering the size of each file, i.e., only the correctness of the search results can be verified, and the integrity of the search results cannot be verified. Our solution integrates the security, document set, and validation of each file size in the document set. In addition, the BDOP-PEKS, VSEF, and Zhang et al. schemes all rely on bilinear mapping operations, which cannot resist quantum attacks. In contrast, our VR-PEKS scheme is constructed based on the fully homomorphic encryption BFV, which is based on the RLWE hard problem and is able to resist quantum attacks [39].

Table 1. Security comparison.

Schemes	KC-IND	KT-IND	External KGA	Internal KGA	Quantum Attack	Correctness Verifiable	Integrity Verifiable	Third-Party Auditor
BDOP-PEKS [5]	yes	no	insecure	insecure	insecure	no	no	/
VSEF [33]	yes	yes	secure	secure	insecure	yes	no	required
Zhang's [16]	yes	yes	secure	secure	insecure	yes	no	required
Ours	yes	yes	secure	secure	secure	yes	yes	not required

6.2. Calculation Comparison

The computational cost comparison of the algorithms for each scheme is shown in Table 2. We consider only the time-consuming cryptographic operations. Let E denote the exponential operation, P denote the pairing operation, H denote the hash operation, M denote the homomorphic multiplication operation, and A denote the homomorphic addition operation. m denotes the number of keywords in W, n denotes the number of files in F, and ℓ denotes the number of search result files returned, while f denotes the number of data owners. U denotes the number of attributes in the system. For BDOP-

PEKS and its improved scheme, due to using a forward index (file–keyword), when performing keyword matching, almost all encrypted indexes of the ciphertext file need to be matched once, so the complexity of the search phase is $O(n)$. This leads to a serious decrease in search efficiency in systems with a large number of files. VSEF, Zhang et al.'s scheme, and ours are constructed based on the reverse index (keyword–file), and the search efficiency is proportional to the number of keywords, which is more efficient. The algorithms in [5,16,33] are based on bilinear pairing construction, which requires elliptic curve-based operations. The algorithms in ours are based on the RLWE construction, which can improve the computational speed and achieve higher security strength [40]. However, it slightly increases the storage overhead and communication overhead because the ciphertext is usually stored in matrix form.

We performed experimental simulations of searchable encryption schemes based on bilinear pairing and RLWE-based constructions to measure the practical performance of the above schemes. These experiments were conducted on Windows with a 3.40 GHz AMD Ryzen 5 2600 Six-Core processor, and Java's Pairing-Based Cryptography (JPBC) Library and Rings Library. We consider $|Z_p| = 96$ bit and $|G_1| = |G_2| = 208$ bit. The depth of the polynomial ring is 32 bits. The encrypted keyword is 2 bytes. In this paper, we show the performance characteristics of the main algorithms, namely *KeyGen*, *Enc*, *GenTrap*, and *Search*. In Figure 5, we show the computational overhead of the four main algorithms. In performing the experiments, we only considered the operations within the noise allowed. Bootstrapping is required after the noise of the fully homomorphic encryption operation reaches the upper limit, and it may take several minutes to execute bootstrapping in the BFV scheme [34]. Since the bilinear pairing-based encryption scheme involves exponential operations, while the RLWE-based encryption scheme involves only additive and multiplicative operations, the computational overhead of the RLWE-based encryption scheme is much lower than that of the bilinear pairing-based encryption scheme in the noise range allowed for decryption.

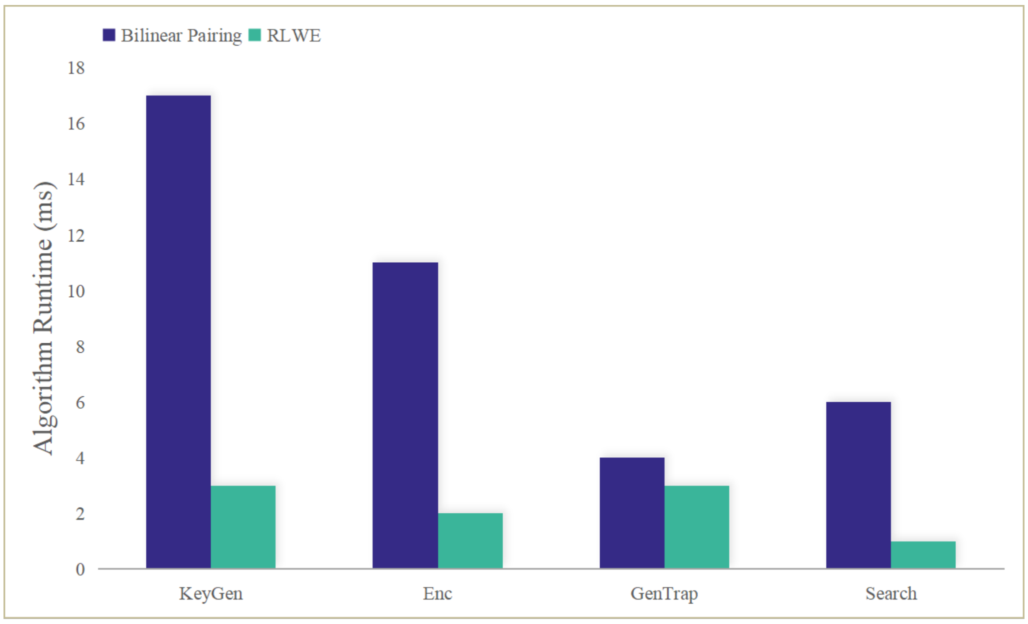

Figure 5. Reverse encrypted keyword index structure.

Table 2. Theoretical calculation cost comparison.

Schemes	Construction	$KeyGen$	Enc	$Trap$	$Search$	$Verify$
BDOP-PEKS [5]	Bilinear Pairing	$2E$	$2E + 2H + P$	$E + H$	$n(H + P)$	/
VSEF [33]	Bilinear Pairing	$2E$	$6E + 2H$	$3E + H + P$	$(m + 1)P$	$(2\ell + 1)E + \ell H + 2P$
Zhang's [16]	Bilinear Pairing	$(2U + f + 4)E + E_T + H$	$(\sum_{r=1}^{U} U_r + U + 2f + 3 + m)E + 3E_T + H$	$(2U + 1)E$	$(2U + 1)P + E_T$	$3E + 2P + qh$
Ours	RLWE	$M + A$	$2M + 3A + H$	$2M + 3A + H$	$m(2A + M)$	$(2\ell + 1)A$

7. Conclusions

In this paper, we propose a public key encryption with keyword search scheme based on fully homomorphic encryption. First, the scheme uses OPRF to blind the keywords and then encrypts them using BFV to generate an encrypted searchable index, thus enabling the cloud server to search the data without decrypting them and effectively resisting the internal keyword guessing attacks on the cloud server. Second, by constructing an encrypted verifiable index, the data sender enables the receiver to verify the correctness and integrity of the search results without relying on a trusted third-party audit server, thus improving the security of the system on an untrusted cloud server. Our scheme can be used as a reference for data security and privacy protection in other fields, such as data management and sharing in healthcare, finance, and government. In future research work, we will explore the design of more secure, efficient, and semantically richer public key searchable encryption schemes based on FHE, and also focus on RLWE-based post-quantum ciphers to cope with the threat of cracking existing cryptographic algorithms by future quantum computers, and provide more reliable and efficient solutions for data security and privacy protection in cloud computing and other fields.

Author Contributions: Conceptualization, Y.T. and Y.L.; methodology, Y.T. and Y.C.; software, Y.T.; validation, Y.T. and S.D.; formal analysis, Y.T.; investigation, T.L.; resources, T.L.; data curation, S.D.; writing—original draft preparation, Y.T.; writing—review and editing, Y.T. and Y.L.; visualization, S.D.; supervision, Y.C.; project administration, Y.L.; funding acquisition, Y.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the National Natural Science Foundation of China (61962009 and 62202118), the Natural Science Research Technology Top Talent Project of the Education Department of Guizhou Province ([2022]073), and the Vocational Education Science Research Project of the Education Department of Guizhou Province.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: No data available.

Acknowledgments: We sincerely thank the editors and reviewers for their constructive comments and suggestions on this paper.

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

References

1. Chen, Y.; Sun, J.; Yang, Y.; Li, T.; Niu, X.; Zhou, H. PSSPR: A source location privacy protection scheme based on sector phantom routing in WSNs. *Int. J. Intell. Syst.* **2022**, *37*, 1204–1221. [CrossRef]
2. Luo, Y.; Chen, Y.; Li, T.; Wang, Y.; Yang, Y.; Yu, X. An Entropy-View Secure Multiparty Computation Protocol Based on Semi-Honest Model. *J. Organ. End User Comput.* **2022**, *34*, 1–17. [CrossRef]
3. Song, D.X.; Wagner, D.; Perrig, A. Practical techniques for searches on encrypted data. In Proceedings of the 2000 IEEE Symposium on Security and Privacy, Berkeley, CA, USA, 14–17 May 2000; IEEE: Piscataway, NJ, USA, 2000; pp. 44–55.
4. Chaudhari, P.; Das, M.L. KeySea: Keyword-Based Search With Receiver Anonymity in Attribute-Based Searchable Encryption. *IEEE Trans. Serv. Comput.* **2022**, *15*, 1036–1044. [CrossRef]

5. Boneh, D.; Di Crescenzo, G.; Ostrovsky, R.; Persiano, G. Public key encryption with keyword search. In Proceedings of the Advances in Cryptology—EUROCRYPT 2004: International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, 2–6 May 2004; Springer: Berlin/Heidelberg, Germany, 2004; pp. 506–522.
6. Andola, N.; Gahlot, R.; Yadav, V.K.; Venkatesan, S.; Verma, S. Searchable encryption on the cloud: A survey. *J. Supercomput.* **2022**, *78*, 9952–9984. [CrossRef]
7. Byun, J.W.; Rhee, H.S.; Park, H.A.; Lee, D.H. Off-line keyword guessing attacks on recent keyword search schemes over encrypted data. In Proceedings of the Secure Data Management: Third VLDB Workshop, SDM 2006, Seoul, Korea, 10–11 September 2006; Springer: Berlin/Heidelberg, Germany, 2006; pp. 75–83.
8. Xu, P.; Tang, X.; Wang, W.; Jin, H.; Yang, L.T. Fast and parallel keyword search over public-key ciphertexts for cloud-assisted IoT. *IEEE Access* **2017**, *5*, 24775–24784. [CrossRef]
9. Xu, P.; Jin, H.; Wu, Q.; Wang, W. Public-key encryption with fuzzy keyword search: A provably secure scheme under keyword guessing attack. *IEEE Trans. Comput.* **2012**, *62*, 2266–2277. [CrossRef]
10. Rongmao, C.; Yi, M.; Guomin, Y.; Fuchun, G.; Xinyi, H.; Xiaofen, W.; Yongjun, W. Server-Aided Public Key Encryption With Keyword Search. *Inf. Forensics Secur. IEEE Trans. ISSN* **2016**, *11*, 1556–6013.
11. Cheng, L.; Meng, F. Certificateless public key authenticated searchable encryption with enhanced security model in IIoT applications. *IEEE Internet Things J.* **2022**, *10*, 1391–1400 [CrossRef]
12. Baror, S.O.; Venter, H. A taxonomy for cybercrime attack in the public cloud. In Proceedings of the International Conference on Cyber Warfare and Security, Stellenbosch, South Africa, 28 February–1 March 2019; Academic Conferences International Limited: Reading, UK, 2019; p. 505.
13. Bove, D.; Müller, T. Investigating characteristics of attacks on public cloud systems. In Proceedings of the 2019 6th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2019 5th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom), Paris, France, 21–23 June 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 89–94.
14. Li, T.; Wang, Z.; Chen, Y.; Li, C.; Jia, Y.; Yang, Y. Is semi-selfish mining available without being detected? *Int. J. Intell. Syst.* **2022**, *37*, 10576–10597. [CrossRef]
15. Miao, Y.; Weng, J.; Liu, X.; Choo, K.K.R.; Liu, Z.; Li, H. Enabling verifiable multiple keywords search over encrypted cloud data. *Inf. Sci.* **2018**, *465*, 21–37. [CrossRef]
16. Zhang, Y.; Zhu, T.; Guo, R.; Xu, S.; Cui, H.; Cao, J. Multi-keyword searchable and verifiable attribute-based encryption over cloud data. *IEEE Trans. Cloud Comput.* **2021**, *11*, 971–983. [CrossRef]
17. Yousefipoor, V.; Eghlidos, T. An efficient, secure and verifiable conjunctive keyword search scheme based on rank metric codes over encrypted outsourced cloud data. *Comput. Electr. Eng.* **2023**, *105*, 108523. [CrossRef]
18. Liu, Z.Y.; Tseng, Y.F.; Tso, R.; Mambo, M.; Chen, Y.C. Public-key authenticated encryption with keyword search: A generic construction and its quantum-resistant instantiation. *Comput. J.* **2022**, *65*, 2828–2844. [CrossRef]
19. Gentry, C. Fully homomorphic encryption using ideal lattices. In Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing, Bethesda, MD, USA, 31 May–2 June 2009; pp. 169–178.
20. Akavia, A.; Feldman, D.; Shaul, H. Secure search on encrypted data via multi-ring sketch. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, Toronto, ON, Canada, 15–19 October 2018; pp. 985–1001.
21. Akavia, A.; Gentry, C.; Halevi, S.; Leibovich, M. Setup-free secure search on encrypted data: Faster and post-processing free. *Cryptol. ePrint Arch.* **2018**. [CrossRef]
22. Wen, R.; Yu, Y.; Xie, X.; Zhang, Y. Leaf: A faster secure search algorithm via localization, extraction, and reconstruction. In Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, 9–13 November 2020; pp. 1219–1232.
23. Baek, J.; Safavi-Naini, R.; Susilo, W. Public key encryption with keyword search revisited. In Proceedings of the Computational Science and Its Applications—ICCSA 2008: International Conference, Perugia, Italy, 30 June–3 July 2008; Springer: Berlin/Heidelberg, Germany, 2008; pp. 1249–1259.
24. Tang, Q.; Chen, L. Public-key encryption with registered keyword search. In Proceedings of the Public Key Infrastructures, Services and Applications: 6th European Workshop, EuroPKI 2009, Pisa, Italy, 10–11 September 2009; Springer: Berlin/Heidelberg, Germany, 2010; pp. 163–178.
25. Rhee, H.S.; Park, J.H.; Susilo, W.; Lee, D.H. Trapdoor security in a searchable public-key encryption scheme with a designated tester. *J. Syst. Softw.* **2010**, *83*, 763–771. [CrossRef]
26. Li, H.; Huang, Q.; Susilo, W. A secure cloud data sharing protocol for enterprise supporting hierarchical keyword search. *IEEE Trans. Dependable Secur. Comput.* **2020**, *19*, 1532–1543. [CrossRef]
27. Pan, X.; Li, F. Public-key authenticated encryption with keyword search achieving both multi-ciphertext and multi-trapdoor indistinguishability. *J. Syst. Archit.* **2021**, *115*, 102075. [CrossRef]
28. Qin, B.; Cui, H.; Zheng, X.; Zheng, D. Improved security model for public-key authenticated encryption with keyword search. In Proceedings of the Provable and Practical Security: 15th International Conference, ProvSec 2021, Guangzhou, China, 5–8 November 2021; Springer: Berlin/Heidelberg, Germany, 2021; pp. 19–38.
29. Li, T.; Wang, Z.; Yang, G.; Cui, Y.; Chen, Y.; Yu, X. Semi-selfish mining based on hidden Markov decision process. *Int. J. Intell. Syst.* **2021**, *36*, 3596–3612. [CrossRef]

30. Zheng, Q.; Xu, S.; Ateniese, G. VABKS: Verifiable attribute-based keyword search over outsourced encrypted data. In Proceedings of the IEEE INFOCOM 2014—IEEE Conference on Computer Communications, Toronto, ON, Canada, 27 April–2 May 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 522–530.
31. Sun, W.; Liu, X.; Lou, W.; Hou, Y.T.; Li, H. Catch you if you lie to me: Efficient verifiable conjunctive keyword search over large dynamic encrypted cloud data. In Proceedings of the 2015 IEEE Conference on Computer Communications (INFOCOM), Hong Kong, China, 26 April–1 May 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 2110–2118.
32. Chen, Z.; Zhang, F.; Zhang, P.; Liu, J.K.; Huang, J.; Zhao, H.; Shen, J. Verifiable keyword search for secure big data-based mobile healthcare networks with fine-grained authorization control. *Future Gener. Comput. Syst.* **2018**, *87*, 712–724. [CrossRef]
33. Miao, Y.; Tong, Q.; Deng, R.H.; Choo, K.K.R.; Liu, X.; Li, H. Verifiable searchable encryption framework against insider keyword-guessing attack in cloud storage. *IEEE Trans. Cloud Comput.* **2020**, *10*, 835–848. [CrossRef]
34. Fan, J.; Vercauteren, F. Somewhat practical fully homomorphic encryption. *Cryptol. ePrint Archive* **2012**.
35. Cheon, J.H.; Kim, A.; Kim, M.; Song, Y. Homomorphic encryption for arithmetic of approximate numbers. In Proceedings of the Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, 3–7 December 2017; Springer: Berlin/Heidelberg, Germany, 2017; pp. 409–437.
36. Freedman, M.J.; Ishai, Y.; Pinkas, B.; Reingold, O. Keyword Search and Oblivious Pseudorandom Functions. In Proceedings of the TCC, Cambridge, MA, USA, 10–12 February 2005; Springer: Berlin/Heidelberg, Germany, 2005; Volume 3378, pp. 303–324.
37. law Jarecki, S.; Liu, X. Fast secure computation of set intersection. In Proceedings of the International Conference on Security and Cryptography for Networks, Amalfi, Italy, 13–15 September 2010; Springer: Berlin/Heidelberg, Germany, 2010; pp. 418–435.
38. Regev, O. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM* **2009**, *56*, 1–40. [CrossRef]
39. Maringer, G.; Puchinger, S.; Wachter-Zeh, A. Information- and Coding-Theoretic Analysis of the RLWE/MLWE Channel. *IEEE Trans. Inf. Forensics Secur.* **2023**, *18*, 549–564. [CrossRef]
40. Peikert, C. Lattice cryptography for the internet. In Proceedings of the Post-Quantum Cryptography: 6th International Workshop, PQCrypto 2014, Waterloo, ON, Canada, 1–3 October 2014; Springer: Berlin/Heidelberg, Germany, 2014; pp. 197–219.