IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 4, 2020, accepted January 23, 2020, date of publication February 3, 2020, date of current version February 14, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2971089

A Parallel and Forward Private Searchable Public-Key Encryption for Cloud-Based Data Sharing

BIWEN CHEN[1,2], LIBING WU[3,4], (Senior Member, IEEE), LI LI[3], KIM-KWANG RAYMOND CHOO[5], (Senior Member, IEEE), AND DEBIAO HE[2,3]

[1]School of Computer Science, Wuhan University, Wuhan 430072, China
[2]Guangdong Provincial Key Laboratory of Data Security and Privacy Protection, Guangzhou 510632, China
[3]School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China
[4]Shenzhen Research Institute, Wuhan University, Shenzhen 518057, China
[5]Department of Information Systems and Cyber Security, The University of Texas at San Antonio, San Antonio, TX 78249, USA

Corresponding author: Libing Wu (whuwlb@126.com)

This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFC1315404, in part by the National Natural Science Foundation of China under Grant 61772377, Grant 61972294, and Grant 61932016, in part by the Natural Science Foundation of Hubei Province of China under Grant 2017CFA007, in part by the Science and Technology Planning Project of Shenzhen under Grant JCYJ20170818112550194, and in part by the Opening Project of Guangdong Provincial Key Laboratory of Data Security and Privacy Protection under Grant 2017B030301004-11.

ABSTRACT Data sharing through the cloud is flourishing with the development of cloud computing technology. The new wave of technology will also give rise to new security challenges, particularly the data confidentiality in cloud-based sharing applications. Searchable encryption is considered as one of the most promising solutions for balancing data confidentiality and usability. However, most existing searchable encryption schemes cannot simultaneously satisfy requirements for both high search efficiency and strong security due to lack of some must-have properties, such as parallel search and forward security. To address this problem, we propose a variant searchable encryption with parallelism and forward privacy, namely the parallel and forward private searchable public-key encryption (PFP-SPE). PFP-SPE scheme achieves both the parallelism and forward privacy at the expense of slightly higher storage costs. PFP-SPE has similar search efficiency with that of some searchable symmetric encryption schemes but no key distribution problem. The security analysis and the performance evaluation on a real-world dataset demonstrate that the proposed scheme is suitable for practical application.

INDEX TERMS Data sharing, cloud storage, searchable encryption, parallel search, forward privacy.

I. INTRODUCTION

Cloud-based data sharing has emerged as a promising solution for convenient and on-demand access to large amounts of data shared. Its numerous benefits, including lower cost, better resource utilization, and greater agility, have attracted extensive attention in industry or academia. The cloud-based data sharing systems are already widely applied in a lot of different industries such as education, logistics, healthcare, finance. The classical application scenarios of cloud-based data sharing are shown in FIGURE 1. However, with the continuous occurrence of security issues (e.g., celebrity photos being leaked in iCloud), users are increasingly concerned about privacy protection while enjoying the convenience of cloud storage. Therefore, secure mechanisms balancing privacy and utilization of data are urgently needed to facilitate the widespread application of cloud-based data sharing.

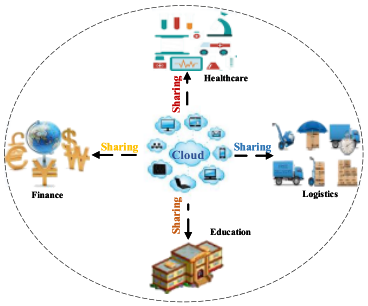


FIGURE 1. The classical application scenarios of cloud-based data sharing.

The associate editor coordinating the review of this manuscript and approving it for publication was Kuo-Hui Yeh.

Searchable encryption (SE), searching on encrypted data without decrypting them, seems to be one of the most desirable techniques to provide privacy-preserving without sacrificing data availability. Currently, there are already abundant works under different threat models and various functionality requirements. Most existing SE schemes can be divided roughly into two broad categories: searchable symmetric-key encryption (SSE) and searchable public-key encryption (SPE). Although SSE has high computation performance and low storage costs, it requires that all users share a symmetric key to generate ciphertexts and search keywords, which incurs the complex key distribution problem. In contrast to SSE, SPE avoids the above drawbacks of SSE and can realize the flexible access control. But the most existing SPE schemes are still not impractical for cloud-based data sharing systems because they may be vulnerable to security threats [1], [2] (e.g., keyword guessing attack and file-injection attack) and their performance is far from being practical [3].

Recently, a few SPE schemes [1], [4]–[7] have been designed to better protect users' query privacy. Huang and Li [1] proposed a variant SPE with authentication function to resist inside keyword guessing attack and Li *et al.* [4] extended the work to the identity-based setting for addressing the key management problem. Unfortunately, their schemes cannot resist the file-injection attack introduced by Zhang *et al.* [2], in which an attacker can recover the keywords corresponding to users' trapdoors by injecting a small number of documents (usually less than 100). The attack may be devastating for the existing SE schemes, particularly with SPE schemes, because the injection files in the SPE schemes are easier than that of in SSE scheme. But Zhang *et al.* also pointed out that forward privacy as an important security property can be used to resist the above attack. However, most existing schemes [5]–[7] supporting forward privacy belong to the SSE, while most SPE schemes do not support this security property.

The low performance of SPE schemes is one of the greatest inhibitors of their practical applications. Recently, Xu *et al.* [3] designed a lightweight SPE scheme with hidden structures, and their scheme reduces the number of computation-intensive operations so that its search performance close to that of some SSE schemes. However, Xu's scheme still exists two significant drawbacks due to its inherent design. Firstly, their scheme cannot supports the parallel keyword search. The ciphertexts containing the same keyword exist a hidden relation of chain and can only be sought out in series. Secondly, forward privacy is not supported. The hidden data structure is implemented through a forward singly list, the search server always finds the first ciphertext by a common and public parameter.

A. MOTIVATION AND CONTRIBUTIONS

Our investigation started with a question: can we construct a parallel SPE scheme with forward privacy? Fortunately, inspired by the work of Xu *et al.*, we present a positive answer to this question. We design a parallel and forward private SPE scheme (PFP-SPE) following the hidden data structure used by Xu, but with a very different implement. PFP-SPE uses a hidden star-chain data structure, in which every update generates a new state, which is like the pointer that points all the newly inserted files and the states are related by a symmetric key primitive. The search server can only decrypt the previous states by the current state and the corresponding key and then finds all matching files, but it can not predict the next state. The main contributions of this paper are summarized as follows:

- We introduce a new variant searchable public-key encryption scheme–**P**arallel and **F**orward **P**rivate **S**earchable **P**ublic-Key **E**ncryption (PFP-SPE). PFP-SPE achieves the parallel search and forward privacy by leveraging the hidden data structure.
- We give a concrete construction of PFP-SPE and present a formal security proof under the computable bilinear Diffie-Hellman assumption.
- We implement our proposed scheme based on the popular cryptographic library and measure the performance of that over a real-world dataset. Theoretical analysis and experimental results demonstrate the practicability of our scheme.

B. ORGANIZATION

The remainder of the paper is organized as follows. Section II introduces the related works by emphasizing their features. Section III, IV present the preliminaries and the system overview, respectively. Section V gives a concrete construction of PFP-SPE and the correctness analysis of its, followed by its formal security proof in Section VI. Section VII presents the theoretical analysis and some experimental evaluations about the construction of the proposed scheme. The conclusion is drawn in Section VIII.

II. RELATED WORK

Searchable encryption schemes have been extensively investigated in recent years. This section will introduce some significant works related to this paper.

Security: It is most important for searchable encryption schemes to have strong security. Recently, several attacks [2], [16], [17] have been designed to recover the information of keyword in the search trapdoors. These attacks are performed by leveraging the information leaked during the searching and updating processes. For example, the work [2] shows even small leakage can be exploited by passive adversaries and highlights the importance of forward privacy. Stefanov *et al.* [18] firstly proposed the concept of forward privacy and constructed a forward private DSSE scheme based on ORAM. Likewise, Garg *et al.* [19] also proposed an efficient DSSE scheme via the round-optimal oblivious RAM. But, their schemes are impractical because of the efficiency problem. Bost [5] designed a forward private DSSE scheme, which only relies on trapdoor permutations. However, Song *et al.* [7] found that Bost *et al.*'s scheme exists

two significant deficiencies: his design is based on public key cryptography and the search operation is not I/O efficient [20]. Then, the work of Song *et al.* [7] presented two effective schemes: FAST and FASTIO, and these schemes combine efficiency and forward privacy. Recently, Ghareh Chamani [21] presented a new construction for forward and backward privacy SSE scheme, which supports two security features: forward and backward privacy. Unfortunately, most existing schemes supporting forward privacy [5], [7], [21] are based on the symmetric-key cryptosystem, and the key management and distribution problems may limit the application scenarios of them, such as Internet of Things. In contrast to, the SPE schemes [1], [4], [8], [22]–[25] can effectively avoid the above problems and support the flexible access control. However, how to construct a SPE scheme with forward privacy is still an open hard problem.

Search Efficiency: The search efficiency of SE plays a key role in practical applications. The scheme proposed by Song *et al.* [26] takes search time that is linear in the size of the dataset. Then, Curtmola *et al.* [27] designed the first index-based SSE scheme achieving sub-linear search complexity. Currently, most SPE schemes are designed based on the inverted index [1] and the search time of these SPE schemes [22], [28], [29] is linear with the total number of keywords. To improve the search efficiency, Xu *et al.* [30] proposed a new SPE scheme for keyword search as fast as possible, called searchable public-key ciphertexts with hidden structures. Their scheme improves search efficiency by defining the hidden relationship among the ciphertexts of keywords, and this relationship can only be disclosed by a legal trapdoor of keyword. Recently, some schemes [31]–[33] with high search efficiency are designed based on the hidden structures. In addition, most existing SPE schemes [22], [30] generally use many computation-intensive operations to generate ciphertexts and search the matching ciphertexts, which limits their scheme to be widely used. Along this way, some lightweight schemes [3], [34], [35] have been designed. For example, Xu *et al.* [3] designed a more efficient scheme by reducing some computation-intensive operations. Designing an efficient mechanism to improve the performance of SPE schemes remains an important challenge.

III. PRELIMINARIES

All key notations in the paper can be referred in Table 1.

Bilinear Pairing: Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic groups with the same order q, and P be a generator of $\mathbb{G}_1$. There exists a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$, which satisfies the following characteristics:

- e is computable: for all elements $P_1, P_2 \in \mathbb{G}_1$, there exists a polynomial-time algorithm to compute $e(P_1, P_2) \in \mathbb{G}_2$.
- e is bilinear: for all elements $P_1, P_2 \in \mathbb{G}_1$, $x, y \in \mathbb{Z}_q$, $e(P_1^x, P_2^y) = e(P_1, P_2)^{xy}$.
- e is non-degenerate: $e(P, P) \neq 1$.

[1] A keyword is mapped to a set of indexes of files containing this keyword

TABLE 1. Notations in this paper.

Notation	Description
$x \xleftarrow{R} \mathbb{X}$	an element x is sampled uniformly at random from a set $\mathbb{X}$
$\lvert\mathbb{X}\rvert$	$\mathbb{X}$'s cardinality
λ	a security parameter
$(q, \mathbb{G}_1, \mathbb{G}_2, P, e)$	the public system parameter about the pairing operation
(H_0, H_1, H_2, H_3)	the hash function sets (mapping a group element to a string)
(h_1, h_2, h_3, h_4)	the hash function sets (mapping a string to other string)
F/F^{-1}	a pseudo-random permutation and the inverse permutation
$(PK_i, sk_i)_{i=do,dr}$	the public/private key pair of the users
(ST, VI)	two key-value maps secretly maintained by the users
ind_i	the index of i-th file
$DB(w)$	the index set of files containing the keyword w
$EDB(w)$	the ciphertexts of $DB(w)$
st_c, k_c, c	a state under the counter c and the corresponding key
(Sa_w, Sv_w)	the state ciphertext pair of keyword w
$(Ia_i, Iv_i))_{i=1,\ldots}$	the indexes ciphertext set
(Ha_{dr}, Hv_{dr})	the pointer ciphertext pair of the data receiver
v_w	the version information of keyword w
c_w	the version ciphertext of v_w
T_w	the trapdoor of keyword w
p_j	the j-th thread

CBDH Assumption: The security of our proposed scheme is based on the computable bilinear Diffie-Hellman assumption, which is defined as below:

Given the parameter $param = (q, P, \mathbb{G}_1, \mathbb{G}_2, e)$ and $(aP, bP, cP) \in \mathbb{G}_1$, where $(a, b, c) \in \mathbb{Z}_q$ are chosen and unknown. The CBDH assumption is that there no exists an efficient polynomial-time algorithm to compute $e(P, P)^{abc}$. If we assume an attacker A can compute $e(P, P)^{abc}$ with the advantage $Adv_A^{CBDH}(1^\lambda)$, then the advantage $Adv_A^{CBDH}(1^\lambda)$ must be negligible if the CBDH assumption holds.

Hidden Structure: The concept of the hidden structure was first introduced to improve the search efficiency in work [30]. Although the first work [30] has sub-linear search complexity, lots of the computation-intensive operations make it impractical with the increasing the data size. Later, based on the hidden single linked list structure, Xu *et al.* [3] gave a lightweight solution with high search performance, which is the first SPE scheme as efficient as a practical SSE scheme. Unlike the linked list structure, the work in this paper adopts a new data structure, namely star-chain.

FIGURE 2 shows the differences between the linked list and star-chain through a simple example. In contrast to the single linked list structure, the star-chain structure has two significant properties. Firstly, when inserting some new files, the single linked list appends to the end of the list, while these files will become the new header of the chain in the star-chain structure. The property makes the star-chain structure support forward privacy. Secondly, when searching on a keyword, all matching messages only be found one by one in a single linked list, but multiple messages can be found at once in the star-chain structure because these messages are attached to

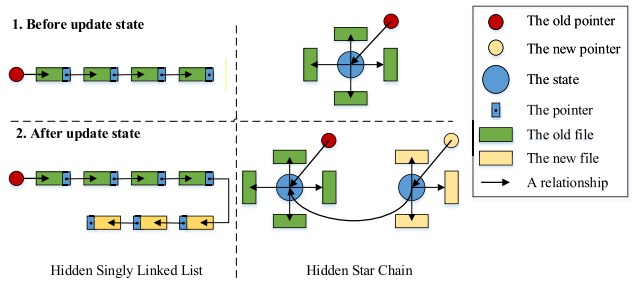


FIGURE 2. The hidden data structure.

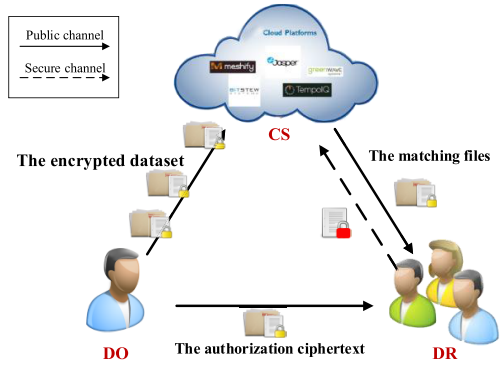


FIGURE 3. System model of PFP-SPE.

a state, like a star. The property provides effective help for parallel search.

IV. SYSTEM OVERVIEW

In our cloud-based data sharing system, there mainly involves three entities, namely Data Owner (DO, e.g. data service provider), Data Receiver (DR, e.g. data customer) and Cloud Servicer (CS). The system model is shown in FIGURE 3. The responsibilities of every entity are given as follows:

- **DO**: After extracting the keyword-indexes pairs from the outsourced dataset, the DO needs to encrypt them and builds an encrypted outsourced dataset. Then, (s)he uploads the encrypted dataset to CS, together with the encrypted keyword-indexes pairs. Meanwhile, (s)he also needs to send the authorization ciphertexts to the data receiver.
- **DR**: An authorized DR first submits a trapdoor to obtain the matching ciphertexts, and then enjoys the relevant data services.
- **CS**: When receiving a search query, the CS finds all matching ciphertexts and returns them to the DR.

A. FORMALIZED DEFINITION

According to the above description, the PFP-SPE should include five algorithms: **Setup**, **KeyGen**, **Encryption**, **Trapdoor** and **Search**, respectively.

- **Setup**(λ): This algorithm is a fundamental one. Given a security parameter λ, it outputs the system parameter *param*.
- **KeyGen**(*param*): This algorithm is called by every users to generate public/private key pairs. Given the system parameter *param*, it outputs a public/private key pair (PK, sk).
- **Encryption**($param$, $DB(w)$, w, sk_{do}, PK_{dr}, ST): This algorithm is only invoked by the DO. Given the system parameter *param*, the dataset $DB(w)$, the secret key sk_{do} of the DO, the public key PK_{dr} of the DR and a state map ST maintained by the DO, it outputs the encrypted dataset $EDB(w)$ and the updated state map ST.
- **Trapdoor**($param$, sk_{dr}, PK_{do}, w, VI): This algorithm is invoked by the DR. Given the system parameter *param*, the secret key sk_{dr} of the DR, the public key PK_{do} of the DO, a keyword w to be searched and a version map VI maintained by the DR, it outputs an authorization trapdoor T_w.
- **Search**($param$, T_w, EDB): This algorithm is executed by the CS. Given the system parameter *param*, a trapdoor T_w and the encrypted dataset EDB, it outputs the matching result R.

B. DESIGN GOALS

Our design goals are to enforce the following application requirements:

- **Correctness**. This is a fundamental requirement. The data receiver should be able to find the matching indexes if the trapdoor given is correct.
- **Confidentiality**. This requirement consists of two security properties: ciphertext indistinguishability and forward privacy.
- **Forward Privacy**. This requirement ensures that an adversary can not use a previous trapdoor to test the newly updated files.
- **Parallelism**. This requirement requires that our scheme should be able to support the parallel search.

C. SECRUITY MODEL

Depending on what security requirements the system model needs to have, a PFP-SPE scheme should ensure that the keyword-indexes pairs ciphertexts do not reveal their corresponding keyword information. Besides, a secure PFP-SPE scheme must prevent the previous trapdoor from querying for newly updated data, which can resist the file-injection attack to obtain the relations of newly updated data and previously searched keywords. We formally define the security notions of the PFP-SPE, including ciphertext indistinguishability against the chosen-plaintext attack and forward privacy, by leveraging the following security game takes place between an adversary A and a challenger C.

Ciphertext Indistinguishability: The chosen-plaintext attack game consists of the following five phases:

- **Setup**. The challenger C first initializes the whole system, including generating the system parameter and publishing them to the attacker A. Then, the challenger performs the **KeyGen** algorithm to generate the public/private key pairs (PK_{do}, sk_{do}), (PK_{dr}, sk_{dr}) and sends the public keys (PK_{do}, PK_{dr}) to A. The phase

mainly simulates that the adversary A knows all public information.

- **Query 1**. The phase is mainly the interaction between A and C. The attacker A can adaptively issue the encryption queries. With regard to each keyword-indexes pair $(w, DB(w))$, the challenger C performs the **Encryption**$(param, DB(w), w, sk_{do}, PK_{dr}, ST)$ algorithm to output the encrypted dataset $EDB(w)$ and a new state map ST^*. The challenger C sends $EDB(w)$ to A and keeps the updated map ST^* secretly. Meanwhile, the attacker A can also issue the search trapdoor query. With regard to each trapdoor query about subscription keyword w, C performs the **Trapdoor**$(param, sk_{dr}, PK_{do}, w, VI)$ algorithm to generate a trapdoor T_w and sends it to A. Assume before the trapdoor query is issued, the encryption query about w has been issued. This phase mainly simulates A can obtain some information during the execution of PFP-SPE scheme.
- **Challenge**. In this phase, the attacker A choose two challenging keyword-indexes pairs $(w_0, DB(w_0))$, $(w_1, DB(w_1))$ as his attack target. When receiving the challenging pairs $(w_0, DB(w_0))$, $(w_1, DB(w_1))$, the challenger C first selects a random bit $b \xleftarrow{R} \{0, 1\}$, then C executes $(EDB(w_b), ST^*) =$ **Encryption**$(param, DB(w), w, sk_{do}, PK_{dr}, ST)$ to encrypt w_b. Finally, C sends $EDB(w_b)$ to A.
- **Query 2**. The phase is the same as the **Query 1**. But there exists a restriction that the attacker A is not allowed to make trapdoor queries for (w_0, w_1).
- **Guess**. In this phase, the attacker A choses a bit $b^* \in \{0, 1\}$ as a guess about b. If $b^* = b$, A wins the game; Otherwise, A fails.

The A's advantage in winning above game is defined as:

$$Adv_A^{CPA}(1^\lambda) = |Pr[b^* = b] - \frac{1}{2}|$$

Definition 1: FPF-SPE scheme achieves the ciphertext indistinguishability against chosen-plaintext attack if there exist no PPT adversaries which can win the above game with a non-negligible advantage under the CBDH assumption.

Forward Privacy: An adversary can break the privacy of a searchable encryption scheme, which does not support forward privacy, by launching the file-injection attack. The process is as follows: 1)The adversary first generates the encrypted index containing certain chosen keywords. 2) (S)He searches the encrypted dataset by the previous trapdoor. 3) The adversary can infer the keyword corresponding to the previous trapdoor from the returned results [2]. Therefore, forward privacy needs to prevent the leakage of the update operation in searchable public-key encryption and ensures that a previous trapdoor cannot be used to search the newly inserted files.

Definition 2: FPF-SPE scheme achieves the forward privacy if the update operation leaks no information about the keywords.

V. OUR SCHEME

In this section, we first give a construction of SBSE-PF scheme. Then, we prove the correctness of our construction.

A. CONSTRUCTION

Our PFP-SPE scheme is a tuple of several algorithms, namely **Setup**, **KeyGen**, **Encryption**, **Trapdoor** and **Search** respectively. The details of these algorithms are as follows:

Setup(λ): It takes a security parameter λ as input, and generates the system parameter $param = (q, \mathbb{G}_1, \mathbb{G}_2, P, e, F, F^{-1}, H_0, H_1, H_2, H_3, h_1, h_2, h_3, h_4)$. Let $(\mathbb{G}_1, \mathbb{G}_2)$ be two cyclic groups with the same order q and P be a generator of group $\mathbb{G}_1$. Let e be a bilinear pairing operation $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. Let $H_0 : \{0,1\}^* \to \mathbb{G}_1$, $H_1, H_2 : \mathbb{G}_2 \to \{0,1\}^\lambda$, $H_3 : \mathbb{G}_1 \to \{0,1\}^{2\lambda}$, $h_1 : \{0,1\}^\lambda \to \{0,1\}^\lambda$, $h_2 : \{0,1\}^\lambda \to \{0,1\}^{3\lambda}$ and $h_3, h_4 : \{0,1\}^\lambda \times \mathbb{N} \to \{0,1\}^\lambda$. Let $F : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^\lambda$ be a pseudo-random permutation (e.g., DES, AES) and F^{-1} is the inverse permutation of F.

Besides, both the DO and the DR need to maintain two maps $(ST, VI \leftarrow$ **empty map**$)$ formed as $(key, value)$, respectively.

KeyGen$(param)$: It takes the system parameter $param$ as input, every user selects a random number $s \in \mathbb{Z}_q^*$ as his secret key, and then computes the public key $PK = sP$. Let (PK_{dr}, sk_{dr}) and (PK_{do}, sk_{do}) be the public/private key pairs of the DR and the DO, respectively.

Encryption$(param, DB(w), w, sk_{do}, PK_{dr}, ST)$: It takes the system parameter $param$, the index set of files $DB(w) = \{ind_1, \ldots, ind_m\}$ containing the keyword w, where $m = |DB(w)|$, the private key sk_{do} of the DO, the public key PK_{dr} of the DR and the state map ST saved secretly by the DO. Then, the DO executes the following steps (Algorithm 1):

1 Retrieves the state $(st_c, k_c, c) \leftarrow ST[w]$ by w from the map ST, where c is a counter. If $ST[w] = \perp$, then sets $st_c \xleftarrow{R} \{0,1\}^*$, $k_c \xleftarrow{R} \{0,1\}^*$ and $c = 0$.
2 Computes $k_{c+1} = F(sk_{do}, k_c)$ and $st_{c+1} = F(k_{c+1}, st_c)$. Then sets $Sa_w = h_1(st_{c+1})$ and $Sv_w = (0||m||k_{c+1}) \oplus h_2(st_{c+1})$.
3 For $i \in \{1, m\}$, computes $Ia_i = h_3(st_{c+1}, i)$ and $Iv_i = h_4(st_{c+1}, i) \oplus (ind_i)$.
4 Updates the state map $ST[w] = (st_{c+1}, k_{c+1}, c+1)$.
5 Computes $Ha_{dr} = H_1(e(sk_{do}H_0(w), PK_{dr})^{v_w})$, and $Hv_{dr} = H_2(e(sk_{do}H_0(w), PK_{dr})^{v_w}) \oplus st_c$, where $v_w \xleftarrow{R} \mathbb{Z}_q^*$.
6 Computes $c_w = (c_w^1, c_w^2) = (rP, H_3(rPK_{dr}) \oplus (w||v_w))$, where $r \xleftarrow{R} \mathbb{Z}_q^*$.

Finally, the DO sends (Sa_w, Sv_w), (Ia_i, Iv_i), (Ha_{dr}, Hv_{dr}) to the CS and publishes c_w to the DR respectively. The CS stores the ciphertexts as $EDB[Sa_w] = Sv_w$, $EDB[Ha_{dr}] = Hv_{dr}$, $EDB[Ia_i] = Iv_i$, where $i = \{1, 2, \ldots, m\}$. And the DR obtains v_w by decrypting the ciphertext c_w and inserts the map $VI[w] = v_w$. The process can be done off-line.

Trapdoor$(param, sk_{dr}, PK_{do}, w, VI)$: It takes the system parameter $param$, the public key PK_{do} of the DO, the secret key sk_{dr} of the DR, a keyword w and the map VI as inputs, and performs the following steps:

Algorithm 1 The Construction of PFP-SPE Scheme

Encryption($param$, $DB(w)$, sk_{do}, PK_{dr}, ST): (DB(w) = $\{ind_1, \ldots, ind_m\}$)

1: $(st_c, k_c, c) \leftarrow ST[w]$.
2: **if** $(st_c, k_c, c) == \perp$ **then**
3: $st_c \xleftarrow{R} \{0,1\}^*$, $k_c \xleftarrow{R} \{0,1\}^*$, $c = 0$.
4: **end if**
5: $k_{c+1} = F(sk_{do}, k_c)$
6: $st_{c+1} = F(k_{c+1}, st_c)$.
7: $ST[w] \leftarrow (st_{c+1}, k_{c+1}, c+1)$.
8: $Sa_w = h_1(st_{c+1})$.
9: $Sv_w = (0||m||k_{c+1}) \oplus h_2(st_{c+1})$.
10: **for all** i such that $1 \le i \le m$ **do**
11: $Ia_i = h_3(st_{c+1}, i)$.
12: $Iv_i = h_4(st_{c+1}, i) \oplus (ind_i)$
13: **end for**
14: $v_w \xleftarrow{R} \mathbb{Z}_q^*$.
15: $Ha_{dr} = H_1(e(sk_{do}H_0(w), PK_{dr})^{v_w})$.
16: $Hv_{dr} = H_2(e(sk_{do}H_0(w), PK_{dr})^{v_w}) \oplus st_{c+1}$.
17: send (Sa_w, Sv_w), $(Ia_i, Iv_i)_{i=1,\ldots,m}$, (Ha_{dr}, Hv_{dr}) to **CS**.
18: $r \xleftarrow{R} \mathbb{Z}_q^*$.
19: $c_w = (c_w^1, c_w^2) = (rP, H_3(rPK_{dr}) \oplus (w||v_w))$
20: send c_w to **DR**.

CS:

21: $EDB[Sa_w] = Sv_w$.
22: **for all** i such that $1 \le i \le m$ **do**
23: $EDB[Ia_i] = Iv_i$.
24: **end for**

DR:

25: $(w||v_w) = H_3(sk_{dr}c_w^1) \oplus c_w^2 = H_3(sk_{dr}rP) \oplus H_3(rPK_{dr}) \oplus (w||v_w)$.
26: $VI[w] = v_w$.

Algorithm 2 The Construction of PFP-SPE Scheme

Search($param$, T_w, EDB):

1: $R \leftarrow \emptyset$
2: $Ha_{dr}^* = H_1(T_w) = H_1(e(sk_{dr}PK_{do}, v_wH_0(w))) = H_1(e(sk_{do}H_0(w), PK_{dr})^{v_w})$.
3: $Hv_{dr}^* \leftarrow EDB[Ha_{dr}^*]$
4: **if** $Hv_{dr}^* == \perp$ **then**
5: return R
6: **end if**
7: $st = Hv_{dr}^* \oplus H_2(T_w)$.
8: $j = 0$.
9: **while** 1 **do**
10: $Sa_w^* = h_1(st)$.
11: $Sv_w^* = EDB[Sa_w^*]$.
12: **if** $Sv_w^* == \perp$ **then**
13: return R.
14: **end if**
15: $(0||m||k) = Sv_w^* \oplus h_2(st)$.
16: **Thread** p_j = **New thread**().
17: $p_j(param, st, m, EDB)$.
18: $st = F^{-1}(k, st)$.
19: **end while**

Thread $p_j(param, st, m, EDB)$:

20: **for all** i such that $1 \le i \le m$ **do**
21: $Ia_i^* = h_3(st, i)$.
22: $Iv_i^* = EDB[Ia_i^*]$
23: $ind_i = h_4(st, i) \oplus Iv_i^*$.
24: $R \leftarrow R \cup in_i$.
25: return R
26: **end for**

1 Retrieves the version information $v_w = VI[w]$ by w from the version map VI. If $VI[w] = \perp$, the algorithm terminates.
2 Computes the trapdoor $T_w = e(sk_{dr}PK_{do}, v_wH_0(w))$.
3 Sends the trapdoor T_w to the CS.

Search($param$, T_w, EDB): It takes the system parameter $param$, the ciphertext database EDB, and the trapdoor T_w of w as inputs, and performs the following steps (Algorithm 2):

1 Computes $Ha_{dr}^* = H_1(e(sk_{dr}PK_{do}, v_wH_0(w))) = H_1(e(sk_{do}H_0(w), PK_{dr})^{v_w}) = H_1(T_w)$, and then retrieves $Hv_{dr}^* = EDB[Ha_{dr}^*]$.
2 If $Hv_{dr}^* = \perp$, terminates this algorithm and return $R_w = \emptyset$. Otherwise, obtains a state information $st_c = H_2(T_w) \oplus Hv_{dr}^*$.
3 Computes $Sa_w^* = h_1(st_c)$ and obtains $(0||m||k) = Sv_w^* \oplus h_2(st_c)$ where $Sv_w^* = EDB[Sa_w^*]$. If $Sv_w^* = \perp$, returns R_w to the DR.
4 For $i = m$ to 1, the CS computes $Ia_i^* = h_3(st_c, i)$ and obtains $Iv_i^* = EDB[Ia_i^*]$, and adds the index $(ind_i = h_4(st, i) \oplus Iv_i^*)$ into $R_w = R_w \cup ind_i$,
5 Computes $st_{c-1} = F^{-1}(k, st_c)$ and sets $st_c = st_{c-1}$, and go to step 3.

It is worth noting that the number m of returned indexes is exposed as a tradeoff for supporting the parallel search.

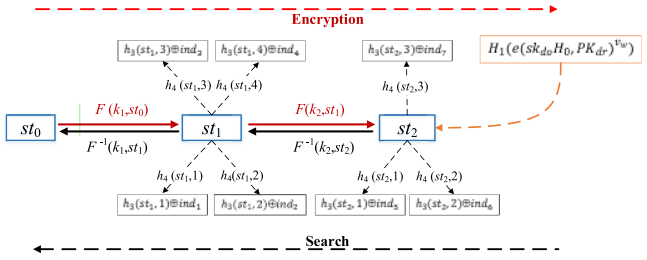


FIGURE 4. An example of PFP-SPE.

B. CORRECTNESS

Example of PFP-SPE: To illustrate the correctness of our construction, we present a simple example, which is shown in FIGURE 4. In this example, there are 7 file indexes $(ind_i)_{i=1,\ldots,7}$, which are encrypted twice (the first time, 4 file indexes are encrypted, and 3 file indexes are encrypted at the second time). The latest pointer ciphertext pair $(Ha_{dr}, Hv_{dr}) = (H_1(e(sk_{do}H_0(w), PK_{dr})^{v_w}), H_2(e(sk_{do}H_0(w), PK_{dr})^{v_w}) \oplus st_2)$ can be used to find the latest state st_2, where v_w is the latest version information of w. We assume that the DR has inserted the version v_w into the map $Vi[w] = v_w$. When the DR needs to search a keyword w, (s)he computes the trapdoor $T_w = e(sk_{dr}PK_{do}, v_wH_0(w))$ and sends it to the CS. The CS computes $Ha^*_{dr} = H_1(T_w)$ and obtains the state $st_2 = H_2(T_w) \oplus Hv^*_{dr}$, where $Hv^*_{dr} = EDB[Ha^*_{dr}]$. The CS can obtain the number $m = 3$ of indexes corresponding to the state st_2 and a secret key k_2 by decrypting the state ciphertext pair $(Sa_w = h_1(st_2), Sv_w = (0||3||k_2) \oplus h_2(st_2))$. Then, the CS starts two threads, one is responsible to find the previous state $st_1 = F^{-1}(k_2, st_2)$, one is responsible to search the indexes $\{ind_5, ind_6, ind_7\}$ corresponding to the state st_2 by computing $h_4(st_2, 1), h_4(st_2, 2), h_4(st_2, 3)$ respectively. By repeat above process, CS can find all matching indexes $R = \{ind_1, \ldots, ind_7\}$. From the above process, we know that the trapdoor as a pointer points to the latest state.

Correctness: The above example shows the CS can find all matching ciphertexts if the DR provides a correct trapdoor. Here, we will prove primarily that the CS cannot search any matching ciphertext if the DR provides an incorrect trapdoor.

Theorem 1: PFP-SPE is correct if the hash functions are collision-resistant.

Proof: Suppose the hash functions selected are collision-resistant and the latest state is st about the keyword w_i and the corresponding pointer ciphertext pair $Ha_{dr} = H_1(e(sk_{do} \cdot H_0(w_i), PK_{dr})^{v_{w_i}}), Hv_{dr} = H_2(e(sk_{do} \cdot H_0(w_i), PK_{dr})^{v_{w_i}}) \oplus st$, where v_{w_i} denotes the latest version.

If there exists a PPT adversary A who can break the correctness of the PFP-SPE, which means that A can find the right results even though (s)he provides an incorrect trapdoor. In other words, an incorrect trapdoor $T^*_{w_i}$ is not equal to $T_{w_i} = e(sk_{dr}PK_{do}, v_{w_i}H_0(w_i))$, but the equation $H_1(T^*_{w_i}) = H_1(e(sk_{do}H_0(w_i), PK_{dr})^{v_{w_i}}) = H_1(T_{w_i})$ holds. That means the adversary A breaks the collision-resistant property of hash function H_1, which is in contradiction to the assumption. Thus, the PFP-SPE is correct if the hash functions are collision-resistant.

Parallel: In a singly list structure [3], [30], every index only corresponds to a state, the CS searches the matching indexes one by one. In our scheme, every state corresponds to multiple indexes like a star. After the CS finds a state, a thread is started to search all indexes corresponding to the current state, while the CS can still compute the next state. Thus, the search task can be achieved in parallel. It is worth noting that if the DO just inserts one index at a time, the search efficiency of our scheme will be similar to that of the singly-linked list schemes. Thus, the more indexes the DO updates each time, the higher the search efficiency of our proposed scheme.

VI. SECURITY PROOF

In this section, we prove the security properties including ciphertext indistinguishability and forward privacy.

A. CIPHERTEXT INDISTINGUISHABILITY

The security of our proposed scheme relies on the CBDH assumption, which means that PFP-SPE achieves ciphertext indistinguishability secure if the CBDH assumption holds.

Theorem 2: Let $\mathcal{O}_E$ and $\mathcal{O}_T$ be two oracles to response queries of the algorithms **Encryption**, **Trapdoor**, respectively. We assume that the adversary A breaks the chosen-plaintext attack game with the advantage $Adv^{CPA}_A(1^\lambda)$ and makes at most q_{H_0} hash function queries to H_0, at most q_{H_1} hash function queries to H_1, at most q_{H_2} hash function queries to H_2, at most q_E oracle queries to $\mathcal{O}_E$ and at most q_T oracle queries to $\mathcal{O}_T$. Then, the challenger C solves the CBDH problem with probability:

$$Adv^{CBDH}_C(1^\lambda) \geq \frac{Adv^{CPA}_A(1^\lambda)}{2\hat{e}(q_E + q_T + 1)(q_{H_1} + q_{H_2})}$$

where $\hat{e}$ is the base of the natural logarithm.

Proof: In order to take advantage of the adversary's ability, the challenger C plays the chosen-plaintext attack game with the adversary.

1. **Setup**. Firstly, the challenger C selects an instance of CBDH problem $(P, \alpha P, \beta P, \gamma P)$ and aims to computes $e(P, P)^{\alpha\beta\gamma}$, where (α, β, γ) are three unknown random numbers. C publishes the system parameter $param = (q, \mathbb{G}_1, \mathbb{G}_2, P, e, F, F^{-1}, H_0, H_1, H_2, H_3, h_1, h_2, h_3, h_4)$ to A and sets $(PK_{do} = \alpha P, PK_{dr} = \beta P)$ as the public keys of the DO and the DR, respectively.
2. **Query** 1. The adversary A can adaptively issue to oracles $\mathcal{O}_E, \mathcal{O}_T$ and hash functions H_0, H_1, H_2. We assume that the same issue is queried only one time and the encryption query was made before the trapdoor query.
   - H_0 queries: at any time the adversary A can query the hash function H_0 as the random oracle. To respond to $H_0(w_i)$ queries, C performs the following steps:
     1) If w_i already is queried, C searches the tuple $(w_i, h_{w_i}, x_{w_i}, coin)$ from H_0 list and responds with $H_1(w_i) = h_{w_i}$.

2) Otherwise, C selects a random $x_{w_i} \in \mathbb{Z}_q^*$ and picks a random bit $coin \xleftarrow{R} \{0,1\}$, where $Pr[coin = 1] = \frac{1}{q_E+q_T+1}$. If $coin = 0$, C computes $h_{w_i} = x_{w_i}P$; if $coin = 1$, C computes $h_{w_i} = x_{w_i}\gamma P \in \mathbb{G}_1$. Then, C inserts $(w_i, h_{w_i}, x_{w_i}, coin)$ into the H_0-list, which is initially empty.

- H_1 queries: In each query, the adversary A issues an element $U \in \mathbb{G}_2$. To respond to the query $H_1(U)$, C picks a new random value $V \in \{0,1\}^\lambda$ as its response. And then C inserts (U, V) into H_1-list.
- H_2, H_3 queries: This query is similar to the H_1 query. Given an element $U \in \mathbb{G}_{1,2}$, C returns the random string $Z \in \{0,1\}^*$. Then C inserts the pair (U, Z) into $H_{2,3}$-lists.
- **Encryption** queries: When A issues an encryption query corresponding to the keyword $(w_i, DB(w_i)$, C responds as follows:
  1) Executes the steps $7-19$ in algorithm 1, in which C selects $k^*_{c+1} \xleftarrow{R} \{0,1\}^*$ instead of computing $k_{c+1} = F(sk_{do}, k_c)$. Due to the randomness of the pseudo-random permutation F, the distributions of k^*_{c+1} and k_{c+1} are indistinguishable. Meanwhile, C sends the encrypted data $(Sa_{w_i}, Sv_{w_i}), (Ia_i, Iv_i)_{i=1,\ldots,m}$ to the adversary and inserts the tuple $(st_{c+1}, k^*_{c+1}, c+1)$ to the lists $K[w_i][c+1] = k^*_{c+1}, ST[w_i] = st_{c+1}$ secretly.
  2) Retrieves $(w_i, h_{w_i}, x_{w_i}, coin)$ by w_i from H_0-List, If the record $(w_i, *, *, *) \notin H_0$-List, C issues the query $H_0(w_i)$.
  3) If $coin = 0$, C computes $U = e(x_{w_i}PK_{do}, PK_{dr})^{v_w} = e(sk_{do}H_0(w_i), PK_{dr})^{v_{w_i}}$ and issues the query $H_1(U)$ to obtain the value $V = H_1(U)$, where $v_w \xleftarrow{R} \mathbb{Z}_q^*$. Then C stores the pair (w_i, v_{w_i}) as the version to the list $VI[w_i] = v_{w_i}$ and issues the query $H_2(U)$ to obtain the value $Z = H_2(U)$. C returns $(Ha_{dr} = V, Hv_{dr} = Z \oplus st_{c+1})$ to the adversary, where st_{c+1} is retrieved by w_i from the map $ST[w_i] = st_{c+1}$. In addition, C executes the steps $24-25$ to generate c_w and returns it to the adversary.
  5) Otherwise, reports failure and terminates.
- **Trapdoor** queries: When A issues an trapdoor query corresponding to the keyword w_i, the challenger C responds as follows:
  1) Retrieves $(w_i, h_{w_i}, x_{w_i}, coin)$ by w_i from H_0-List, If record $(w_i, *, *, *) \notin H_0$-List, C issues the query $H_0(w_i)$.
  2) If $coin = 0$, C retrieves x_{w_i}, v_{w_i} by w_i from H_0-list and VI, respectively. And then C computes $T_{w_i} = e(x_{w_i}PK_{do}, PK_{dr})^{v_{w_i}} = e(sk_{dr}PK_{do}, H_0(w_i))^{v_{w_i}}$. Finally, C returns the trapdoor T_{w_i} as the response.
  3) Otherwise, reports failure and terminates.
- **Challenge**. The adversary A adaptively selects two keyword-indexes pairs $(w_0, DB(w_0))$, $(w_1, DB(w_1))$ that it wishes to be challenged on. C generates the challenge authorization information as follows:
  1) C runs the H_0 queries twice to obtain $h_0, h_1 \leftarrow \mathbb{G}_1$ such that $H_0(w_0) = h_0$ and $H_0(w_1) = h_1$. For $i = 0, 1$, let $(w_i, h_i, x_i, coin_i)$ be the corresponding tuples on the H_0-list. If both $coin_0 = 0$ and $coin_1 = 0$, then C reports failure and terminates.
  2) C randomly selects a bit $b \in \{0,1\}$ such that $coin_b = 1$.
  3) C executes the step 1) in the **Encryption** queries and keeps the tuple $(st^*_{c+1}, k^*_{c+1}, c+1)$, which denotes the latest state information of the keyword w_b.
  4) C computes $(Ha_{dr}, Hv_{dr}) = (V^*_{w_b}, Z^*_{w_b} \oplus st^*_{c+1})$, where $V^*_{w_b}, Z^*_{w_b} \xleftarrow{R} \{0,1\}^*$. Note that this challenge implicitly defines $V^*_{w_b} = H_1(e(sk_{do}PK_{dr}, H_0(w_b))^{v_{w_b}})$ and $Z^*_{w_b} = H_2(e(sk_{do}PK_{dr}, H_0(w_b))^{v_{w_b}})$.
- **Query** 2. The phase is the same as the **query** 1 phase, where the only restriction is that the adversary can not issue w_0, w_1 to $\mathcal{O}_E$ nor $\mathcal{O}_T$.
- **Guess**. The attacker A sends a bit b' as its guess. At this point, C picks a random pair (U, V) from the H_1-list and outputs $\frac{U}{e(P,P)^{x_b v_{w_b}}}$ as its guess for the CBDH problem, where x_b and v_{w_b} are the values used in the challenge phase. The reason this works is that A must have issued a query for either $H_1(e(sk_{do}PK_{dr}, H_0(w_0))^{v_{w_0}})$ or $H_1(e(sk_{do} \cdot PK_{dr}, H_0(w_1))^{v_{w_1}})$. Thus, with probability $\frac{1}{2}$, the H_1-List contains a pair whose left hand side is $U = e(P,P)^{\alpha\beta\gamma x_{w_b} v_{w_b}}$. If C picks the right pair (U, V) from the H_1-list, then $\frac{U}{e(P,P)^{x_{w_b} v_{w_b}}}$ as its solution for CBDH problem.

In the following content, we will show the probability that C correctly solves the CBDH problem in the above game. To do so, we define four events:

- ε_1: C does not failure as a result of any of A's encryption queries.
- ε_2: C does not failure as a result of any of A's trapdoor queries.
- ε_3: C does not failure during the challenge phase.
- ε_4: A issues hash query H_1 with element $e(P,P)^{\alpha\beta\gamma x_{w_0} v_{w_0}}$ or $e(P,P)^{\alpha\beta\gamma x_{w_1} v_{w_1}}$.

To solve the CBDH problem, C needs to ensure that the simulation process is successful. According to the above game, C may terminates in the *Encryption* queries, *Trapdoor* queries or in the challenge phase, respectively. Moreover, those cases are all independent.

Claim 1: The probability of both event ε_1 and the event ε_2 happening at the same time is at least $\frac{1}{\hat{e}}$, where $\hat{e}$ is the base of the natural logarithm.

Proof: We assume that w_i denotes the keyword of the A's i-th encryption query and w_j denotes the keyword of

the A's j-th trapdoor query. We also assume that the tuples $(w_i, h_{w_i}, x_{w_i}, coin_i)$ and $(w_j, h_{w_j}, x_{w_j}, coin_j)$ corresponds to the H_0-list, H_1-List, respectively.

We know that if one **Encryption** query (or one **Trapdoor** query) fails, which means $coin_i = 1$ (or $coin_j = 1$). Because the probability of $coin_i = 1$ (or $coin_j = 1$) is equal to $\frac{1}{q_E+q_T+1}$, the probability of one encryption query (or one trapdoor query) failing to failure is equal to $\frac{1}{q_E+q_T+1}$. Since the adversary A makes at most q_E **Encryption** queries and at most q_T **Trapdoor** queries respectively, the probability of both event ε_1 and the event ε_2 happening at the same time is at least $(1 - \frac{1}{q_E+q_T+1})^{(q_E+q_T)} \geq \frac{1}{\hat{e}}$, where $\hat{e}$ is the base of the natural logarithm.

Claim 2: The probability of the event ε_3 happening is at least $\frac{1}{q_E+q_T+1}$.

Proof: The event ε_3 does not happen if A is able to generate w_0, w_1 with the following property: $coin_0 = coin_1 = 0$, where the bits $coin_0, coin_1$ belong to two tuples $(w_i, h_{w_i}, x_{w_i}, coin_i)$. Since $Pr[coin_i = 1] = \frac{1}{q_E+q_T+1}$ and the values $coin_i, coin_j$ are independent of each other, we have that $Pr[coin_0 = coin_1 = 0] = (1 - \frac{1}{q_E+q_T=1})^2 \leq 1 - \frac{1}{q_E+q_T+1}$. Hence, the probability that the event ε_3 happening is at least $\frac{1}{q_E+q_T+1}$.

Claim 3: The probability of the event ε_4 happening is at least $\frac{1}{2}Adv_A^{CPA}(1^\lambda)$, where $Adv_A^{CPA}(1^\lambda)$ is the advantage of A winning the above game. Hence, $Pr[\varepsilon_4] \geq \frac{1}{2}Adv_A^{CPA}(1^\lambda)$

Proof: According to the definition of the chosen-plaintext attack game, we have $Adv_A^{CPA}(1^\lambda) = Pr[b^* = b] - \frac{1}{2}$. If A does not issue the H_1 hash query with element $e(P,P)^{\alpha\beta\gamma x_{w_0} v_{w_0}}$ or $e(P,P)^{\alpha\beta\gamma x_{w_1} v_{w_1}}$, then the attacker A has no advantage to win the game. The main reason is that the challenge ciphertext is independent from all challenge keyword pairs. Thus, we have that:

$$\begin{aligned} Adv_A^{CPA}(1^\lambda) &= Pr[b^* = b] - \frac{1}{2} \\ &= Pr[b^* = b|\varepsilon_4]Pr[\varepsilon_4] \\ &\quad + Pr[b^* = b|\neg\varepsilon_4]Pr[\neg\varepsilon_4] - \frac{1}{2} \\ &= Pr[b^* = b|\varepsilon_4] - \frac{1}{2} \end{aligned}$$

Therefore, we have that $Pr[\varepsilon_4] > Adv_A^{CPA}(1^\lambda)$. In addition, since A has the same probability to issue H_2 hash query with element $e(P,P)^{\alpha\beta\gamma x_{w_0} v_{w_0}}$ and $e(P,P)^{\alpha\beta\gamma x_{w_1} v_{w_1}}$. We finally have that

$$Pr[\varepsilon_4] \geq \frac{1}{2}Adv_A^{CPA}(1^\lambda)$$

According to the above game, there are at most $q_{H_1} + q_{H-2}$ records in the H_1-list and H_2-list. If C does not abort in the above chosen-plaintext attack game, **Claim 3** shows that C has a probability of greater than $\frac{1}{2(q_{H_1}+q_{H_2})}Adv_A^{CPA}(1^\lambda)$ to select $U = e(P,P)^{\alpha\beta\gamma x_{w_b} v_{w_b}}$ correctly.

Finally, according to the above claims, we have that

$$Adv_C^{CBDH}(1^\lambda) \geq \frac{Adv_A^{CPA}(1^\lambda)}{2\hat{e}(q_E + q_T + 1)(q_{H_1} + q_{H_2})}$$

B. FORWARD PRIVACY

The forward privacy of PFP-SPE can be proven in the random oracle model, and the process is similar to that of the work [7] or the work [5]. We will only give a brief explanation here and a similar process can be seen in the literature [7]. In our PFP-SPE scheme, the DO keeps a state st for each keyword w and updates the state st when inserting an index on the server that contains the keyword w. The new state st^* is used to encrypt the index to be updated, which means the previous trapdoors are outdated. Because the search algorithm only finds out the ciphertexts which are encrypted by the corresponding state and cannot be used to search the new ciphertext. After receiving the latest trapdoor pointing to the current state information st^*, the CS can recover all previous states and find all matching indexes. The PFP-SPE uses a pair of pseudo-random permutation functions to ensure that the CS cannot predict the future state without the help of the secret key of DO. Thus, if there are new updates, the CS learns nothing and forward privacy is achieved.

VII. COMPARISONS AND EVALUATION

In this section, we first compare our scheme with another similar scheme [3] in terms of the computation cost, the storage cost, security and so on. Then, we evaluate the performance of our proposed scheme based on the actual dataset.

A. SCHEME COMPARISON

Like the work [3], our scheme is also a special searchable public-key encryption with hidden structure [30]. In contrast to [3], PFP-SPE has considerably more security and has higher search efficiency.

Table 2 shows that our scheme improves the results of work [3] in several ways. N is the total number of indexes containing the keyword w, e denotes one pairing operation, SM denotes the scalar multiplication in group $\mathbb{G}_1$ and $Mul_{\mathbb{G}_2}/Div_{\mathbb{G}_2}/Exp_{\mathbb{G}_2}$, denotes the multiplication/division/exponentiation operations in group $\mathbb{G}_2$, FP denotes the forward privacy and Pa denotes the scheme can perform parallel search operation. The symbol × denotes that the scheme does not have this property and ✓ denotes that the scheme supports this property. The computation costs of encryption and search algorithms of our scheme are not only related to the number of matching indexes N, but also to the number of updates M. The worst case is M is equal to N. Fortunately, the main operations in encryption and search phase are some hashing operations, the time costs of them can be ignored.

Both schemes need to store some information about the keywords to retrieve the corresponding indexes, and the storage cost is also an important indicator to evaluate the storage costs of searchable schemes. In practice, we generally have $|\lambda| \in [80, 128, 192, 256]$ and $|G2| \in [160, 256, 512, 1024]$. Therefore, our scheme is more efficient than Xu *et al.* [3] in terms of storage costs under the same level of security. Besides, since our scheme supports forward privacy and

TABLE 2. Compared with the existing scheme.

Scheme	Computation Cost			Storage Cost	Security&Functionality	
	Encryption	Trapdoor	Search	DO	FP	Pa
Xu [3]	$e + H_0 + SM + N \cdot (H_2 + Mul_{\mathbb{G}_2})$	$H_0 + SM$	$e + (N+1)H_2 + NDiv_{\mathbb{G}_2}$	O($\|\mathbb{G}_2\|$)	×	×
PFP-SPE	$e+H_0+3SM^*+2F+h_1+h_2+H_1+H_2+H_3+N(h_3+h_4)$	$e + SM$	$H_1 + H_2 + h_1 + h_2 + F^{-1} + N(h_3 + h_4)$	O(λ)	✓	✓

* $sk_{do}PK_{dr}$ can be computed off-line.

TABLE 3. Test environment.

System configuration	
DO/CS	Intel(R) Core(TM) i5-4210U CPU @ 1.70GHz
Compiler	VS 2017
Program Library	Miracl 7.0
Elliptic Curve	
EC	$y^2 = x^3 - 3x$
Group Order	$2^{255} + 2^{41} + 1$
Security Level	AES_SECURITY 128

TABLE 4. Testing dataset.

Keyword	The number of documents
Mann-k	8926
Germany-c	5128
Shackleton-s	4407
Hass-e	3030
Beck-s	2674
Fossum-d	2067
Stclair-c	1328
Campbell-l	667

achieves parallel search keywords, both security and search efficiency are improved compared with that of Xu [3].

B. EXPERIMENTAL EVALUATION

In this subsection, we experimentally compare our scheme with Xu's scheme in terms of encrypting indexes and searching keywords. Table 3 shows the system configuration and the chosen elliptic curve (the security level is equal to that of the symmetric cipher key lengths with 128 bits). Specifically, we code our scheme and Xu's scheme using the Miracl library (C++, version-7.0) [2] and the chosen super-singular curve defined over $GF(q)$. We use a laptop (ThinkPad T440, 64 bits Windows 7, 4GB RAM) as the hardware environment. The pseudo-random permutation F is achieved by implementing 128-bit AES-CBC (F^{-1} is implemented by the decryption algorithm of AES-CBC). The basic cryptographic hash function used is SHA-256.

We select Enron Email Dataset (2015 version, about 423 Mb) [3] as our testing data. The public dataset consists of emails sent and received by 150 senior management. As shown in Table 4, we extract 8 keywords corresponding to 33438 documents. Table 5 shows the time costs of each main operations,in which the hash-to-point is the most time consuming. Combining Table 2, Table 4 and Table 5, we roughly evaluate the performance of Xu's scheme and our scheme, and the results are shown in FIGURE 5.

[2]Miracl: online at https://www.miracl.com/

[3]Enron Email Dataset: online at https://www.cs.cmu.edu/ enron/

TABLE 5. The main operations.

Operation	e	H_0	$H_{1,2,3}$	$Mul/Div/Exp_{\mathbb{G}_2}$	SM
Time Cost (ms)	187.5	332.2	0.3	0.063/0.069/15.3	52.5

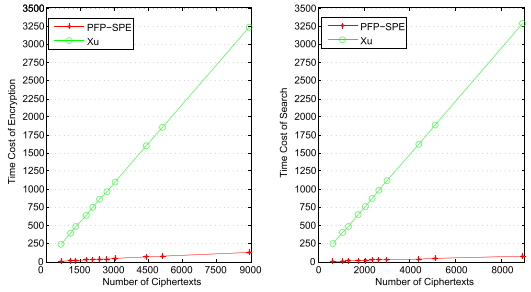


FIGURE 5. Time costs of encryption and search.

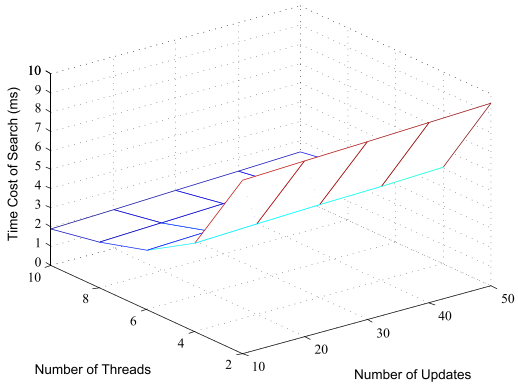


FIGURE 6. Relationships among the search time, the number of threads and the number of updates.

As shown in Figure 5, the more indexes the dataset has, the better the performance of our scheme will be. For example, to search the keywords "$Mann - k$", "$Beck - s$" and "$Campbell - l$", Xu's scheme takes 3.29 s, 0.986 s and 0.246 s respectively, whereas our scheme only takes 0.124 s, 0.037 s, and 0.009 s, respectively. It is clear that the time cost of our scheme is approximately 25 times less than that of Xu's scheme for searching all matching indexes.

FIGURE 6 shows the relationships among the search time, the number of threads and the number of updates, and the results demonstrate that the fewer the number of updates,

the more the number of threads, the more efficient the search. In addition, we also note that efficiency gains are slow when the number of threads reaches a certain level. The main reason is that the main factor affecting search efficiency is not the search for indexes, but the search for the state of each update. Thus, our scheme is better suited to avoid frequent updates.

VIII. CONCLUSION

In this paper, we investigate the new architecture of practical dynamic searchable encryption scheme with efficiency and strong security. Fortunately, we also present a new cryptographic primitive, namely parallel and forward private searchable public-key encryption, and give a concrete construction. In our scheme, the star-chain data structure makes it more search effective and supports forward privacy. The experimental results show our scheme is more suitable for the practical application than Xu *et al.*'s scheme.

ACKNOWLEDGMENT

The authors would like to thank all anonymous reviewers for their suggestions and comments for improvement of the paper.

REFERENCES

[1] Q. Huang and H. Li, "An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks," *Inf. Sci.*, vols. 403–404, pp. 1–14, Sep. 2017.

[2] Y. Zhang, J. Katz, and C. Papamanthou, "All your queries are belong to us: The power of file-injection attacks on searchable encryption," in *Proc. USENIX Secur. Symp.*, 2016, pp. 707–720.

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

[4] H. Li, Q. Huang, J. Shen, G. Yang, and W. Susilo, "Designated-server identity-based authenticated encryption with keyword search for encrypted emails," *Inf. Sci.*, vol. 481, pp. 330–343, May 2019.

[5] R. Bost, "Σοφος forward secure searchable encryption," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 1143–1154.

[6] K. S. Kim, M. Kim, D. Lee, J. H. Park, and W.-H. Kim, "Forward secure dynamic searchable symmetric encryption with efficient updates," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2017, pp. 1449–1463.

[7] X. Song, C. Dong, D. Yuan, Q. Xu, and M. Zhao, "Forward private searchable symmetric encryption with optimized i/o efficiency," *IEEE Trans. Dependable Secure Comput.*, to be published.

[8] M. Ma, D. He, N. Kumar, K.-K.-R. Choo, and J. Chen, "Certificateless searchable public key encryption scheme for industrial Internet of Things," *IEEE Trans. Ind. Inf.*, vol. 14, no. 2, pp. 759–767, Feb. 2018.

[9] L. Wu, B. Chen, K.-K.-R. Choo, and D. He, "Efficient and secure searchable encryption protocol for cloud-based Internet of Things," *J. Parallel Distrib. Comput.*, vol. 111, pp. 152–161, Jan. 2018.

[10] M. Ma, D. He, H. Wang, N. Kumar, and K.-K.-R. Choo, "An efficient and provably secure authenticated key agreement protocol for fog-based vehicular ad-hoc networks," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8065–8075, Oct. 2019.

[11] R. Bost and P.-A. Fouque, "Security-efficiency tradeoffs in searchable encryption," *on Privacy Enhancing technol.*, vol. 2019, no. 4, pp. 132–151, Oct. 2019.

[12] L. Liu, J. Su, X. Liu, R. Chen, K. Huang, R. H. Deng, and X. Wang, "Toward highly secure yet efficient KNN classification scheme on outsourced cloud data," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 9841–9852, Dec. 2019.

[13] P. Vijayakumar, V. Chang, L. Jegatha Deborah, B. Balusamy, and P. Shynu, "Computationally efficient privacy preserving anonymous mutual and batch authentication schemes for vehicular ad hoc networks," *Future Gener. Comput. Syst.*, vol. 78, pp. 943–955, Jan. 2018.

[14] W. Kong, J. Shen, P. Vijayakumar, Y. Cho, and V. Chang, "A practical group blind signature scheme for privacy protection in smart grid," *J. Parallel Distrib. Comput.*, vol. 136, pp. 29–39, Feb. 2020.

[15] Y. Yang, X. Zheng, W. Guo, X. Liu, and V. Chang, "Privacy-preserving smart IoT-based healthcare big data storage and self-adaptive access control system," *Inf. Sci.*, vol. 479, pp. 567–592, Apr. 2019.

[16] M. S. Islam, M. Kuzu, and M. Kantarcioglu, "Access pattern disclosure on searchable encryption: Ramification, attack and mitigation," in *Proc. Ndss*, vol. 20, Feb. 2012, p. 12.

[17] D. Cash, P. Grubbs, J. Perry, and T. Ristenpart, "Leakage-abuse attacks against searchable encryption," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2015. pp. 668–679.

[18] E. Stefanov, C. Papamanthou, and E. Shi, "Practical dynamic searchable encryption with small leakage," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, vol. 71, 2014, pp. 72–75,

[19] S. Garg, P. Mohassel, and C. Papamanthou, "Tworam: Efficient oblivious ram in two rounds with applications to searchable encryption," in *Proc. Annu. Cryptol. Conf.* Berlin, Germany: Springer, 2016, pp. 563–592.

[20] D. Cash, S. Jarecki, C. Jutla, H. Krawczyk, M.-C. Roşu, and M. Steiner, "Highly-scalable searchable symmetric encryption with support for Boolean queries," in *Advances in Cryptology-CRYPTO 2013* Berlin, Germany: Springer, 2013, pp. 353–373.

[21] J. Ghareh Chamani, D. Papadopoulos, C. Papamanthou, and R. Jalili, "New constructions for forward and backward private symmetric searchable encryption," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Jan. 2018, pp. 1038–1055.

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

[23] Y. Yang and M. Ma, "Conjunctive keyword search with designated tester and timing enabled proxy re-encryption function for E-health clouds," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 4, pp. 746–759, Apr. 2016.

[24] J. Li, Y. Shi, and Y. Zhang, "Searchable ciphertext-policy attribute-based encryption with revocation in cloud storage," *Int. J. Commun. Syst.*, vol. 30, no. 1, p. e2942, Jan. 2017.

[25] H. Xiong, H. Zhang, and J. Sun, "Attribute-based privacy-preserving data sharing for dynamic groups in cloud computing," *IEEE Syst. J.*, vol. 13, no. 3, pp. 2739–2750, Sep. 2019.

[26] D. Xiaoding Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symp. Secur. Privacy.*, Nov. 2002, pp. 44–55.

[27] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," *J. Comput. Secur.*, vol. 19, no. 5, pp. 895–934, Nov. 2011.

[28] R. Chen, Y. Mu, G. Yang, F. Guo, and X. Wang, "A new general framework for secure public key encryption with keyword search," in *Proc. Australas. Conf. Inf. Secur. Privacy*. Berlin, Germany: Springer, 2015 pp. 59–76.

[29] R. Chen, Y. Mu, G. Yang, F. Guo, X. Huang, X. Wang, and Y. Wang, "Server-aided public key encryption with keyword search," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 12, pp. 2833–2842, Dec. 2016.

[30] P. Xu, Q. Wu, W. Wang, W. Susilo, J. Domingo-Ferrer, and H. Jin, "Generating searchable public-key ciphertexts with hidden structures for fast keyword search," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 9, pp. 1993–2006, Sep. 2015.

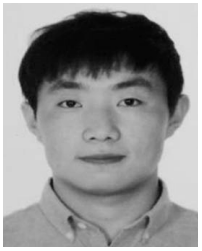
[31] P. Xu, X. Tang, W. Wang, H. Jin, and L. T. Yang, "Fast and parallel keyword search over public-key ciphertexts for cloud-assisted IoT," *IEEE Access*, vol. 5, pp. 24775–24784, 2017.

[32] P. Xu, X. Gao, W. Wang, W. Susilo, Q. Wu, and H. Jin, "Dynamic searchable public-key ciphertexts with fast performance and practical security," in *Proc. IACR*, 2017, p. 741.

[33] P. Xu, S. Tang, P. Xu, Q. Wu, H. Hu, and W. Susilo, "Practical multi-keyword and Boolean search over encrypted e-mail in cloud server," *IEEE Trans. Services Computing*, to be published.

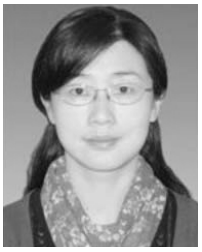
[34] S. Tahir, S. Ruj, Y. Rahulamathavan, M. Rajarajan, and C. Glackin, "A new secure and lightweight searchable encryption scheme over encrypted cloud data," *IEEE Trans. Emerg. Topics Comput.*, vol. 7, no. 4, pp. 530–544, Oct. 2019.

[35] Y. Yang, X. Zheng, and C. Tang, "Lightweight distributed secure data management system for health Internet of Things," *J. Netw. Comput. Appl.*, vol. 89, pp. 26–37, Jul. 2017.



BIWEN CHEN received the M.S. degree in computer science from the Hubei University of Technology, Wuhan, China, in 2016. He is currently pursuing the Ph.D. degree with the Computer School, Wuhan University, Wuhan. His main research interests include cryptography and information security, in particular cryptographic protocols.

LIBING WU (Senior Member, IEEE) was born in 1972. He received the B.S. and M.S. degrees in computer science from Central China Normal University, Wuhan, China, in 1994 and 2001, respectively, and the Ph.D. degree in computer science from Wuhan University, China, in 2006. He is currently a Professor with the School of Cyber Science and Engineering, Wuhan University. His areas of research interests include distributed computing, trusted software, and wireless sensor networks. He is a Senior Member of the CCF.



LI LI received the Ph.D. degree in computer science from the Computer School, Wuhan University. She is currently an Associate Professor with the School of Software, Wuhan University. Her research interests include data security and privacy, applied cryptography, and security protocols.

KIM-KWANG RAYMOND CHOO (Senior Member, IEEE) received the Ph.D. degree in information security from the Queensland University of Technology, Australia, in 2006. He currently holds the Cloud Technology Endowed Professorship with The University of Texas at San Antonio (UTSA). In 2016, he was named the Cybersecurity Educator of the Year (APAC). He and his team won the Digital Forensics Research Challenge organized by Germany's University of Erlangen–Nuremberg, in 2015. He was a recipient of the 2008 Australia Day Achievement Medallion, the British Computer Society's Wilkes Award, in 2008, the Fulbright Scholarship, in 2009, the 2014 Highly Commended Award by the Australia New Zealand Policing Advisory Agency, the 2014 Highly Commended Award by the Australia New Zealand Policing Advisory Agency, the ESORICS 2015 Best Research Paper Award, the IEEE TrustCom 2018 Best Paper Award, the 2018 UTSA College of Business Col. Jean Piccione and Lt. Col. Philip Piccione Endowed Research Award for Tenured Faculty, the 2019 IEEE Technical Committee on Scalable Computing (TCSC) Award for Excellence in Scalable Computing (Middle Career Researcher), the Outstanding Associate Editor of 2018 for IEEE Access, the British Computer Society's 2019 Wilkes Award Runner-Up, the 2019 *EURASIP Journal on Wireless Communications and Networking* (JWCN) Best Paper Award, the Korea Information Processing Society's *Journal of Information Processing Systems* (JIPS) Survey Paper Award (Gold) 2019, and the IEEE Blockchain 2019 Outstanding Paper Award.

DEBIAO HE received the Ph.D. degree in applied mathematics from the School of Mathematics and Statistics, Wuhan University, in 2009. He is currently a Professor with the School of Cyber Science and Engineering, Wuhan University. His main research interests include cryptography and information security, in particular cryptographic protocols.