Hindawi
Security and Communication Networks
Volume 2023, Article ID 8351353, 15 pages
https://doi.org/10.1155/2023/8351353

Research Article

HCV: Practical Multi-Keyword Conjunctive Query with Little Result Pattern Leakage

Xueling Zhu, Shaojing Fu, Huaping Hu, Qing Wu, and Bo Liu

College of Computer, National University of Defense Technology, Changsha, Hunan 410003, China

Correspondence should be addressed to Bo Liu; kyle.liu@nudt.edu.cn

Received 22 September 2021; Revised 13 July 2022; Accepted 3 August 2022; Published 11 July 2023

Academic Editor: Zhili Zhou

Multi-keyword conjunctive query is the most common searchable encryption (SE) scheme and gives practical search capability. This paper focuses on constructing a conjunctive query scheme with high privacy and accurate query result. For efficiency, this paper introduces a novel counter vector (CV) data structure instead of inverted index and builds the CV index database with three-tuple keywords. We use two kinds of cryptography primitives to encrypt the CV index database, respectively. One is symmetric encryption for efficiency, which gives two schemes, CVX and ICV. Experiments show that both CVX and ICV have much higher efficiency than the existing schemes. The other is BGN homomorphic encryption for privacy and gives the scheme HCV. HCV reduces much RP information leakage and even achieves "ideal" leakage for 3-keyword query. Experiments show that the HCV scheme achieves high privacy while compromising not so much storage.

1. Introduction

With the wide application of cloud computing and the commercialization of 5G, more and more people tend to outsource their data. Some remote storage systems [1, 2] help clients with limited resources to manage large amounts of data at a very low cost. Although it brings great convenience and higher efficiency, the security and privacy issues cannot be ignored. For example, famous social networking site Facebook may have leaked data for millions of users to a political firm Cambridge Analytica [3]. Encryption is a simple solution to protect data security, but it would prevent the data from being searched. Searchable encryption (SE) can address this issue by providing a way to search without decryption. Xiaodong et al. proposed the first SE scheme in 2000 [4]. They realized the retrieval of a keyword on the encrypted data. Later, a series of practical SE schemes [5–9] were proposed.

The schemes mentioned above can only support single-keyword queries. However, a practical system needs to find the documents containing a set of keywords, which was called conjunctive query. A naive method of conjunctive query is to perform single-keyword query for each keyword one by one and then filter the desired ones. Nevertheless, if the resultant document set is very large for a keyword, this method will have low efficiency. Besides, this method causes significant information leakage, as it reveals the resultant document sets for each queried keyword. A passive attacker in [10, 11] can leverage the common leakages in SSE schemes to reveal the user's query. Recently, Zhang et al. [12] proposed an even more powerful attack, in which the attacker can adaptively inject new documents. With this power, the attacker can recover the content of user's query by learning which added documents match it [3].

Some conjunctive SSE schemes are proposed to compromise security and efficiency. Golle et al. firstly proposed the conjunctive equality queries [13]. Each conjunctive query builds a set of tokens that can be used to identify matching documents in the database. Their methods only leak the set of matching documents. However, the workload of the server is heavy. Moreover, the communication complexity between server and client is high. The scalability of this solution is limited [14]. Cash et al. [14] proposed the first conjunctive query scheme with sublinear searching complexity, that is, "Oblivious Cross-Tags" (OXT). Before that, all other solutions can only work linearly in the database's size. Nevertheless, the OXT protocol leaks some "partial" information to the server, containing the queries themselves and the database contents. Result pattern (RP) leakage is one of the information leakages mentioned in OXT. The adversary can use it to steal information. The file-injection attacks [12] have exploited RP leakage to reveal all queried keywords with 100% accuracy [15].

Kamara and Moataz [16] proposed highly efficient SSE schemes with worst-case sublinear search and achieved optimal communication complexity. They used the set operations (union, intersection, and complement) for efficiency. Also, their methods can support conjunctive, disjunctive, and Boolean queries. However, set operations inevitably cause information leakage. For the sake of efficiency and functionality, Kamara's method does not prevent RP information leakage. It leaks more than many other solutions. Nevertheless, Kamara's method of building the inverted index is very worthy of our reference. Lai et al. [15] analyzed the result pattern (RP) leakage and proposed "Hidden Cross-Tags" (HXT). The HXT protocol eliminates keyword-pair result pattern (KPRP) leakage presented in the OXT protocol, leaving only the minimal and significantly smaller whole result pattern (WRP). Thus, the HXT protocol offers high security than the OXT protocol. Although the HXT protocol is efficient and practical, it cannot get the full query results because of using Bloom filter to build an index database.

Yin et al. [17] proposed an efficient and privacy-preserving multi-keyword conjunctive query over the cloud. They used the binary tree structure and homomorphic encryption to achieve high query efficiency and small privacy leakage. This method supports the multi-keyword conjunctive query and protects its privacy. However, the scale of this scheme is limited for the tree-based index. They gave the experiment result based on eight keywords, not enough in most of the settings.

The existing scheme mentioned above cannot achieve unnecessary information leakage and precise query results. We make progress on these issues and give an affirmative answer by constructing a practical SSE scheme. Our construction has been focused on the conjunctive queries, like OXT and HXT, since such queries are the most common in many practical settings. In our construction, we assume that the server is honest but curious and there are only one reader and one writer.

1.1. Our Contribution. We make progress on the multi-keyword conjunctive query and construct a method with high privacy and accuracy query result. The main contribution of this paper can be summarized as follows:

(1) First, we progress on the multi-keyword conjunctive query setting since it is a common search method for most people. Unlike the index database based on inverted index, we propose a novel data structure—counter vector (CV)—to construct the index database, which is the basis of our proposed solutions. Using CV, we built a map from three-tuple keywords to a file-path collection containing all the three keywords, and we designed an algorithm to build the CV database as quickly as possible. Compared to Kamara and Moataz's multi-map [16], CV achieves a higher search efficiency while compromising storage efficiency. Although the number of three-tuple keywords is much larger than the keyword pair, storage space's natural growth is limited due to the sparsity inverted index database. However, due to the original single-word inverted index database's sparsity, the actual growth of storage is limited. Experiments show that when the keywords' weight is less than 110 and the number of keywords is not more than 512, the CV database would add about 30% storage to the keyword-pair database, and the search efficiency improved dozens of times.

(2) Second, we propose three schemes CVX, ICV, and HCV, using two kinds of cryptography primitives and CV data structure. CVX is the basic scheme and has a much higher search efficiency than Kamara's IEX. However, CVX's search efficiency is greatly affected by the weight of keywords (i.e., the number of documents containing a keyword). For this reason, we propose the improved ICV, which is more efficient for heavy-weight keywords. For HCV, it uses BGN homomorphic encryption algorithm to reduce the RP leakage [18]. We prove that HCV achieved ideal leakage for a 3-keyword query. All of these schemes have a strong practicability. They are easy to implement and can work on a PC.

(3) Third, we analyze our scheme's security and evaluate the performance of all three schemes. The experiments show that both CVX and ICV have much more searching efficiency than Kamara's IEX. When the keywords' weight is larger, ICV has a significant advantage over CVX. For HCV, we analyze its privacy, indicating that it leaks less than Cash's OXT and gives the accuracy query results better than the probabilistic result of Lai's HXT.

We compared the performance of some schemes, and Table 1 gives the details.

1.2. Related Work. Xiaodong et al. [4] gave the first SSE scheme, whose search complexity is linear to the size of database. Later, Goh [8] introduced a search index for each file and made the search cost to be proportional to the number of files. Curtmola et al. [7] presented the inverted index to achieving sublinear search complexity. This scheme defined two formal security models and gave a formal security definition. To support expressive queries, Golle et al. firstly proposed the conjunctive equality queries [13]. In each conjunctive query, a set of tokens can be built to identify matching documents in the database. Their methods leak little information; however, the performance is not so good and the scalability of this solution is limited.

Table 1: Property comparison.

Scheme	PI	RPH	Conj	Query computation	Query comm
CGKO'06-1 [7]	—	—	—	$O(r)$	(1)
CGKO'06-2 [7]	—	—	—	$O(r)$	$O(r)$
Cash'13-OXT [14]	√	—	√	$O((q-1)r)$	$O(q\cdot r)$
Kamara'17-IEX [16]	—	—	√	$O(q^2\cdot M_1)$	$O(q^2/2)$
Kamara'17-BIEX [16]	—	—	√	$O(q^2\cdot(M_1+l\cdot M_2))$	$O(q^3/2)$
Lai's SHVE [18]	√	√	√	$O(q\cdot r)$	$O(q\cdot r)$
Yin's scheme [17]	—	√	√	$O(r)$	$O(q)$
HCV (our scheme)	—	√	√	$O(\lceil (q-1)/2\rceil\cdot r)$	$O(\lceil (q-1)/2\rceil\cdot r)$

We substitute many leakages by upper bounds and assume some search times' interaction. "PI" means probabilistic (Bloom filter) indexing, "RPH" means result pattern hiding, "Conj" means supporting conjunctive query or not, and "Query comm" means the size of message from client. For notations, q means the number of queried keywords, n = # do cuments, $N=\sum_w |\mathrm{DB}(w)|$, $m=|W|$, $M=\max_w|\mathrm{DB}(w)|$, $r=|\mathrm{DB}(w_1)|$ for the conjunctive query and $|\mathrm{DB}(w)|$ for single-word query, p = #processors, and $M_1=\max\{\#\mathrm{DB}(w_i)\}_{i\in[q]}$.

To support more scalable and expressive queries, Cash et al. [14] proposed the Oblivious Cross-Tags (OXT) protocol with worst-case sublinear search complexity. The OXT protocol divides conjunctive search process into s-term and x-terms. The s-term is for a regular single-keyword search and x-terms are used to acquire document identifiers containing multiple keywords. However, the OXT protocol cannot avoid the "keyword-pair result pattern" (KPRP) leakage, which can be exploited in recent attacks [10, 12]. Since then, a line of extensions [19–21] have been made for OXT. However, such schemes actually trade off performance, security, and functionality. To improve the efficiency of OXT, Kamara and Moataz [16] introduced two schemes LEX-2Lev and LEX-ZMF, and both achieve worst-case sublinear search complexity for Boolean query. Lai et al. [15] proposed "Hidden Cross-Tags" (HXT) protocol, which achieved conjunctive query and reduced the leakage of OXT. The HXT protocol used the "Cross-Tags Set" (XSet) data structure and a lightweight Hidden Vector Encryption (HVE) to encrypt it, and then it avoids the KPRP leakage. In fact, HXT's XSet is actually a Bloom filter, and thus it cannot give the precise query result.

Very recently, some conjunctive SSE schemes with extended functions are proposed. Wang et al. [22] pointed out that the scheme proposed in article [23] is not correct. A new SE scheme was proposed by adopting a special additive homomorphic encryption scheme to achieve the multiplicative homomorphic property efficiently. Furthermore, they enhanced the security on the user side. Ma et al. presented a practical SSE protocol that supports conjunctive queries without KPRP leakage [24]. They proposed a novel SSE protocol called "Practical Hidden Cross-Tags" (PHXT). Using subset membership check (SMC), the PHXT protocol maintains the same storage size as OXT while preserving the same privacy and functionality as HXT. Fan et al. [25] proposed a verifiable conjunctive keyword search scheme based on cuckoo filter (VCKSCF), which significantly reduces verification and storage overhead. Gan et al. also focused on the verifiable conjunctive SSE [26] and presented an efficient verifiable SSE (VSSE) scheme for conjunctive queries with sublinear search overhead. VSSE is built on the OXT protocol and completes the verification through Symmetric Hidden Vector Encryption (SHVE) and greatly reduces the computation payload in the verification process. For IoT application, Zhang et al. [27] proposed a lightweight and efficient attribute-based encryption scheme for data sharing and searching (namely, LSABE). Their scheme can significantly reduce the computing cost of IoT devices with the provision of multiple keyword searching for data users.

This paper is organized as follows. We give the preliminaries in Section 2. In Section 3, we depict the details of the CVX SSE scheme, containing the proof of correctness and evaluation of the efficiency and security. Section 4 depicts the ICV scheme. In Section 5, we introduce the HCV scheme. The experiment results are shown in Section 6. We conclude our method in Section 7.

2. Preliminary

We first depict the notations and definitions. Then, we list part of them in Table 2.

2.1. Notations. We denote all binary strings with length n as $\{0,1\}^n$ and all finite binary strings as $\{0,1\}^*$. Let $[n]=\{1,\ldots,n\}$, and $2^{[n]}$ is the corresponding power set. An element x sampled from a distribution χ is denoted as $x\longleftarrow\chi$. $x\longleftarrow A$ represents that the element x is output by algorithm A. For a tuple v of n elements, its ith element can be denoted as v_i or $v[i]$. Given an element $s\in v$, let $l^{-1}(s)$ denote the index of s in v. For a set S, we use $\#S$ to represent its cardinality. For a string s, $|s|$ means its bit length and s_i means its ith bit. Given strings s and r, $s\|r$ refers to their concatenation.

Multi-map (MM) is an abstract data type. Typically, it can be instantiated by an inverted index. MM with capacity n is a collection of n label/tuple pairs $(l_i,V_i)_{i\le n}$. Getting the tuple associated with label l_i can be denoted as $V_i=\mathrm{MM}[l_i]$. Similarly, associating the tuple V_i to label l_i can be denoted as $\mathrm{MM}[l_i]=V_i$.

The symbol $D=(D_1,\ldots,D_n)$ denotes a document collection. Each document contains a number of keywords from the universe W. The ith keyword in W can be denoted as $W[i]$, and the document identifier can be denoted as $id(D_i)$. Each multi-map can be regarded as a database and denoted as DB. The document collection containing keyword w can be written as $\mathrm{DB}(w)$, and the set of keywords in W that co-occur with w can be written as $co\mathrm{DB}(w)\subseteq W$.

Table 2: Notations and terminologies.

Notation	Meaning
k	A security parameter
id_i	The document identifier of the ith document
n	Number of documents in the database
W_i	All w contained in id_i
(l_i, V_i)	(label, value) pairs
W, K	The set of all keywords, the key used to encrypt
D	The set of all documents $(D_1, \ldots, D_n)$
DB, EDB	Database $(id_i, W_i)_{i=1}^{n}$, encrypted DB
DB (w), coDB (w)	Inverted index $\{ids\colon w \in W_{id}\}$, keywords co-occurring with w
MM, EMM	Multi-map, encrypted MM
CV, ECV	Counter vector, encrypted CV

Informally, for a private-key encryption scheme, if its ciphertexts do not reveal any partial information about the plaintext even to an adversary that can adaptively query an encryption oracle, we say it is secure against chosen-plaintext attacks (CPAs). Similarly, if its ciphertexts are computationally indistinguishable from random even to an adversary that can adaptively query an encryption oracle, we say it is random-ciphertext-secure against chosen-plaintext attacks (RCPAs) [28].

2.2. Result Pattern Hiding Searchable Encryption. Lai et al. [15] proposed result pattern (RP) hiding searchable encryption to resist RP leakage proposed by Cash et al. [14]. RP leakage is the leaked information obtained by the server during query. In [15], Lai et al. analyzed the RP leakage and gave three forms: single-keyword result pattern (SP) leakage, keyword-pair result pattern (KPRP) leakage, and multiple keyword cross-query intersection result pattern (IP) leakage.

The KPRP leakage is a "nonideal" leakage and can be eliminated. Consider an n-keyword conjunction query $w_1 \wedge \cdots \wedge w_n$; during this process, the server gets the set $\mathrm{DB}(w_1) \cap \mathrm{DB}(w_i)$ of documents containing every pair of query keywords of form $(w_1, w_i), 2 \le i \le n$, and it can acquire the final query result, which is the set $\cap_{j=1}^{n} \mathrm{DB}(w_j)$ of documents matching all n query keywords.

Only the final query result, which is called whole result pattern (WRP) leakage, cannot be avoided during this process. In addition, other leaks are intermediate links in the query process and can be reduced in some ways.

2.3. Bilinear Groups and Homomorphic Encryption

2.3.1. Bilinear Groups of Composite Order. Given a security parameter k, generate a tuple (N, g, G, G_T, e), where $N = p \cdot q$ and p, q are two k-bit prime numbers. G and G_T are two finite cyclic multiplicative groups of composite order N, $g \in G$ is a generator, and $e\colon G \times G, G \longrightarrow G_T$ is a bilinear map with the following properties:

(i) *Bilinearity.* $e(g^a, h^b) = e(g, h)^{ab}$ for any $(g, h) \in G^2$ and $a, b \in Z_N$.

(ii) *Nondegeneracy.* If g is a generator of G, then $e(g, g)$ is a generator of G_T with order N.

(iii) *Computability.* There exists an efficient algorithm to compute $e(g, h) \in G_T$ for all $(g, h) \in G$.

2.3.2. BGN Homomorphic Encryption. The Boneh–Goh–Nissim (BGN) [29] homomorphic encryption includes three algorithms: key generation, encryption, and decryption. We give the detailed depiction as follows.

(i) Key generation: Given a security parameter k, generate a tuple (N, g, G, G_T, e) as described in Section 2.3.1. Set $h = g^q$; then, h is a random generator of the subgroup of G of order p. Compute the key pair, containing the private key $sk = p$ and the public key $pk = (N, G, G_T, e, g, h)$.

(ii) Encryption: let m denote the message to be encrypted, choose a random number $\in Z_N$, and compute the ciphertext $c = E(m, r) = g^m h^r \in G$.

(iii) Decryption: Given the ciphertext $c = E(m, r) = g^m h^r \in G$, then compute $c^p = (g^m h^r)^p = (g^p)^m$. Set $\hat{g} = g^p$ and compute the discrete log of c^p base $\hat{g}$ according to Pollard's lambda method (see [30] (p.128) and [17]).

3. The Basic Scheme

In this section, we will introduce the basic multi-keyword conjunctive query scheme proposed in this paper. First of all, we give the details of a counter vector data structure.

3.1. Counter Vector and Counter Vector Database

3.1.1. Counter Vector. A counter vector cv with length n is an array of n integers. Given a three-tuple (w_1, w_2, w_3), suppose $\mathrm{DB}(w_1)$ contains seven files $fid_1, \ldots, fid_7$, and we can get counter vector cv as in Table 1. A collection of counter-vectors compose of CV database..

Given a counter vector $cv = \{x_1, \ldots, x_n\}$, we can easily query the 2-conjunctive keywords (w_1, w_2) and 3-conjunctive keywords (w_1, w_2, w_3) as follows:

(1) Initialize a result set $T = \varnothing$.

(2) For $i = 1, \ldots, n$,

(a) For 2-conjunctive queries, if $(x_i \geq 1)$, then we append fid_i in T.
(b) For 3-conjunctive queries, if $(x_i == 2)$, then we append fid_i in T.

In fact, our constructions are mainly based on the counter vector. First, we make database (DB) containing CV and MM. Then, we encrypt them separately to get EDB = (EMM, ECV). During the search, we decrypt and get the counter vectors first and then query them to get the result.

3.1.2. Counter Vector Database. In the proposed mechanism, we need to construct CV database. The details are shown in Algorithm 1.

Step 1: sort the original inverted index according to the length of DB (w).

Step 2: for each keyword pair (w_i, w_j), initial a integer vector cv with length $\#\mathrm{DB}(w_i)$, and compute $\mathrm{DB}(w_i \wedge w_j)$.

Step 3: for each three-tuple (w_i, w_j, w_k), compute $\mathrm{DB}(w_i \wedge w_j \wedge w_k)$ and get the counter vectors as in Table 1.

Once finishing all three tuples, we can get the CV database.

Here, we give a definition of symbol $\otimes$, which will be used in the following algorithm. For two counter vectors $cv_1 = (c_{1,1}, c_{1,2}, \ldots, c_{1,n})$ and $cv_2 = (c_{2,1}, c_{2,2}, \ldots, c_{2,n})$, where $c_{i,j} \in \{0,1,2\}$, $i = 1,2$, $1 \leq j \leq n$, define $c = c_1 \otimes c_2$ as follows.

If $c_{1,j} = c_{2,j} = 2$, then $c[j] = 1$; else, $c[j] = 0$.

3.2. Description of CVX Scheme. CVX mechanism is the basic scheme we proposed. It consisted of three modules: Setup, Token, and Query. The Setup generates an index database DB and encrypts it to EDB. Token outputs a token with the key and keyword set. Query returns the queried result once given the token and EDB. These modules contain several algorithms:

(i) A CV encryption scheme $\sum_{\mathrm{CV}} = (\mathrm{Setup}, \mathrm{Token}, \mathrm{Get})$, which is adaptive secure.
(ii) A black-box multi-map encryption scheme $\sum_{\mathrm{MM}} = (\mathrm{Setup}, \mathrm{Token}, \mathrm{Query})$.
(iii) A private-key encryption scheme SKE = (Gen, Enc, Dec), which is *RCPA-secure.*
(iv) A pseudo-random function F.
(v) A function GetTag $\{\cdot\}$, which is used to get tags given the counter vector and document database.

The CV structure is a multi-map data structure with different contents from the MM, so CV and MM support the same encryption algorithm, but it will be different in the process of Search. We choose a black-box MM encryption scheme as in Kamara and Moataz [16].

The project is depicted as follows.

(i) *Setup.* In the Setup modules, an index database DB was generated and encrypted to be EDB. Given the security parameter k, compute the keys for encryption. Then, given the keyword/docID pairs, we can generate an inverted index database MM and get a counter vector using Algorithm 1. MM maps each keyword $w \in W$ to encrypted identifiers in DB (w). CV maps each three-tuple (w_i, w_j, w_k) to a vector of integers. Finally, encrypt the MM and CV using algorithms $\sum_{\mathrm{MM}}$ and $\sum_{\mathrm{CV}}$, respectively. We can get encrypted multi-maps (EMM) and encrypted counter vectors (ECV). All the output of Setup mainly contains the encrypted structures EDB = (EMM, ECV) and their keys.
(ii) *Token.* Given the key and a set of queried keywords $w = (w_1, \ldots, w_q)$, the Token algorithm outputs the token TK = (*gtk*, *ltk*). gtk is the global token used to query the MM. ltk is local token used to query the CV. ltk contains $\lceil (q-1)/2 \rceil$ subtokens. Take $q = 5$ for example, gtk is calculated from w_1, and ltk contains two subtokens (stk) computed from (w_1, w_2, w_3) and (w_1, w_4, w_5), respectively.
(iii) *Query.* We can get the output result of Query using the following steps. First of all, input the EMM and global token , and we can get the encrypted multi-maps mm = DB (w_1). Second, from ECV using the local token $\mathrm{ltk} = \left\{\mathrm{stk}_1, \ldots, \mathrm{stk}_{\lceil (q-1)/2 \rceil}\right\}$, get encrypted counter vectors $\mathrm{cv}_i, i = 1, \ldots, \lceil (q-1)/2 \rceil$. Third, decrypt them using the exiting encryption schemes $\sum_{\mathrm{MM}}$ and $\sum_{\mathrm{CV}}$ separately. Subsequently, for each cv_i, get the set S_i of tags according to the counter, compute the intersection $S = \cap S_i$, and output the set S.

We detailed the CV-based conjunctive SSE scheme CVX = (Setup, Token, Query) in Algorithm 2.

3.3. Correctness and Efficiency. In this subsection, we prove the correctness and analyze the efficiency of our scheme.

3.3.1. Correctness. To show the correctness of CVX, we consider the operations of the counter vectors. For a common conjunctive query $w = (w_1 \wedge \cdots \wedge w_q)$, the output of CVX.Query (EDB, tk) is

$$T = \bigcap_{i=1}^{\lceil (q-1)/2 \rceil} T_i. \tag{1}$$

We want to get T for conjunctive queries, where

$$T = \bigcap_{i=1}^{q} \mathrm{DB}_i. \tag{2}$$

We know that

```
     Input: inverted index
     Output: CV database
(1)  sort the original inverted index according to $\#\mathrm{DB}(w)$;
(2)  for $0 \le i \le N-1, i \le j \le N$ do
(3)    compute $S_{i,j} = \mathrm{DB}(w_i) \cap \mathrm{DB}(w_j)$;
(4)    if $S_{i,j} \neq \emptyset$ then
(5)      initial the $\mathrm{tmp}_{i,j}$ of length $\#\mathrm{DB}(w)$ with 0;
(6)      while $\mathrm{tag}_l \in S_{i,j}, l \in [\#\mathrm{DB}(w)]$ do $\mathrm{CV}[l]+=1$;
(7)    end
(8)  end
(9)  for$0 \le i \le N-2, i \le j \le N-1, j \le k \le N$ do
(10)   if $(S_{i,j} \neq \emptyset) and (S_{i,k} \neq \emptyset)$ then
(11)     compute $S = S_{i,j} \cap S_{i,k}$;
(12)     if $S \neq \emptyset$ initial the $CV$ with length $\#\mathrm{DB}(w)$, let $\mathrm{CV}_{i,j,k} = \mathrm{tmp}_{i,j}$; while $\mathrm{tag}_l \in S$ do $\mathrm{CV}_{i,j,k}[l]+=1$;
(13)   end
(14) end
```

Algorithm 1: Generation of CV database.

```
(1)  —Setup
     Input: $k$, inverted index
     Output: $K$, EDB
(2)  a. initialize a multi-map $MM$, for all $w \in W$, pad the $\mathrm{MM}(w)$ according to $\mathrm{DB}(w)$;
(3)  b. initialize a counter-vector$CV$, generate the $CV$ database using Algorithm 1;
(4)  c. compute $(K_g, \mathrm{EMM}) \longleftarrow \sum_{\mathrm{MM}}.\mathrm{Setup}(1^k, \mathrm{MM})$;
(5)  d. compute $(K_l, \mathrm{ECV}) \longleftarrow \sum_{\mathrm{CV}}.\mathrm{Setup}(1^k, \mathrm{CV})$;
(6)  set $K = (K_g, K_l)$ and $= (\mathrm{EMM}, \mathrm{ECV})$;
(7)  return $(K, \mathrm{EDB})$.
(8)  —Token(K, w):
     Input: a subset $V = \{w_1, \ldots, w_q\}$
     Output: token tk
(9)  compute $\mathrm{gtk} = H\ (w_1)$;
(10) For $l = 1: \lceil (q-1)/2 \rceil$
(11)   select a three-tuple $(w_1, w_i, w_j)$;
(12)   compute $\longleftarrow H_1(w_1) \| H_2(w_2) \| H_3(w_3)$ ;
(13)   $\mathrm{stk}_l \longleftarrow H(d)$;
(14) set $\mathrm{ltk} = \{\mathrm{stk}_l\}_{l=1:\lceil (q-1)/2 \rceil}$;
(15) return $tk = \{\mathrm{gtk}, \mathrm{ltk}\}$;
(16) —Query(EDB, tk):
(17) parse EDB as $(\mathrm{EMM}, \mathrm{ECV})$;
(18) parse $tk$ as $(\mathrm{gtk}, \mathrm{ltk})$, parse ltk as $(\mathrm{stk}_1, \ldots, \mathrm{stk}_{\lceil (q-1)/2 \rceil})$;
(19) a. compute $T \longleftarrow \sum_{\mathrm{MM}}.\mathrm{Get}(\mathrm{EMM}, \mathrm{gtk}_i)$;
(20) b. for all $j = 1, \ldots, \lceil (q-1)/2 \rceil$, compute $cv_j \longleftarrow \sum_{CV}.\mathrm{Get}(\mathrm{ECV}, \mathrm{stk}_j)$;
(21) c. for $j = 1, \ldots, \lceil (q-1)/2 \rceil$, compute $t = \mathrm{cv}_1 \otimes \cdots \otimes \mathrm{cv}_{\lceil (q-1)/2 \rceil}$
(22) $T = \mathrm{GetTag}(t)$
```

Algorithm 2: Basic CV SSE.

$$
\begin{aligned}
T &= \bigcap_{i=1}^{q} \mathrm{DB}_i \\
&= (\mathrm{DB}(w_1) \cap \mathrm{DB}(w_2) \cap \mathrm{DB}(w_3)) \cap \mathrm{DB}(w_1) \cap \\
&\mathrm{DB}(w_4) \cap \mathrm{DB}(w_5) \cap \cdots \cap \mathrm{DB}(w_1) \\
&\cap \mathrm{DB}(w_{q-1}) \cap \mathrm{DB}(w_q)) \\
&= \bigcap_{i=1}^{\lceil (q-1)/2 \rceil} T_i.
\end{aligned}
\tag{3}
$$

3.3.2. Efficiency. The Query complexity of *BVX* is $O(\lceil (q-1)/2 \rceil)$. The size of tokens is $O(\lceil (q-1)/2 \rceil)$. The community complexity achieves optimal because the search result has no redundancy. The storage complexity is

$$
O\left(\mathrm{strg}\left(\sum_{w} \#\mathrm{DB}(w)\right) + \sum_{w} (C_w \times (\#co(w) + \#co(w, v)))\right).
\tag{4}
$$

C_w is a constant relating to w. $\#co(w)$ is the number of keyword sets which contain pairs $(w, v), w, v \in W$. $\#co(w, v)$ is the number of keyword sets which contain three-tuple $(w, v, u), w, v, u \in W$. strg is the storage complexity of $\sum_{\mathrm{MM}}$, which is a black-box multi-map encryption scheme used in this paper.

3.4. Security Analysis. CVX is adaptive secure on condition of controlled disclosure [6]. We mainly consider its leakage functions, which include the Setup leakage and Query leakage. The details of the CVX leakage profile is depicted below. Its Setup leakage is

$$\mathscr{L}_S^{\mathrm{cvx}}(\mathrm{DB}) = \left(\mathscr{L}_S^{mm}(\mathrm{MM}), \mathscr{L}_S^{\mathrm{cvx}}(\mathrm{CV})\right), \tag{5}$$

where $\mathscr{L}_S^{mm}(\mathrm{MM})$ and $\mathscr{L}_S^{\mathrm{cvx}}(\mathrm{CV})$ are the Setup leakages of the multi-map encryption schemes and counter vector encryption schemes, respectively. The Query leakage is

$$\begin{aligned} \mathscr{L}_Q^{\mathrm{cvx}}(\mathrm{DB}, w) = {} & \mathscr{L}_S^{mm}(\mathrm{MM}), \\ & \mathscr{L}_Q^{mm}(\mathrm{MM}, w_1), \\ & \mathscr{L}_Q^{\mathrm{cv}}(\mathrm{CV}, w_1, w_{2i}, w_{2i+1}), \\ & \ldots \\ & \mathscr{L}_Q^{\mathrm{cv}}\left(\mathrm{CV}, w_{q-1}, w_q\right), \\ & \mathrm{TagPat}_i(\mathrm{DB}, w)_{i \in [1, \ldots, \lceil (q-1)/2 \rceil]}, \end{aligned} \tag{6}$$

where for all $1 \leq i \leq \lceil (q-1)/2 \rceil$,

$$\begin{aligned} \mathrm{TagPat}_i(\mathrm{DB}, w) = \Big(& \left(f_i(id)\right)_{id \in \mathrm{DB}(w_1 \cap w_{2*i}) \cap \mathrm{DB}(w_{2*i+1})}, \ldots, \\ & \left(f_i(id)\right)_{id \in \mathrm{DB}(w_1 \cap w_{q-1}) \cap \mathrm{DB}(w_q)} \Big), \end{aligned} \tag{7}$$

and f_i is a random function from $\{0, 1\}^{|id| + \log \# W}$ to $\{0, 1\}^k$.

If the definition of adaptive security is similar to Kamara and Moataz [16] (Definition 4.2), we give the theorem as follows.

Theorem 1. *CVX is* $(\mathscr{L}_S^{cvx}, \mathscr{L}_Q^{cvx})$*-secure on the condition that* $\sum_{\mathrm{MM}}$ *and* $\sum_{\mathrm{CV}}$ *are adaptively* $(\mathscr{L}_S^{mm}, \mathscr{L}_Q^{mm})$*-secure, SKE is RCPA-secure, and F is pseudo-random.*

Proof. Suppose $\mathcal{S}_{\mathrm{CV}}$ and $\mathcal{S}_{MM}$ are the simulators guaranteed to exist from $\sum_{\mathrm{MM}}$ and $\sum_{\mathrm{CV}}$'s adaptive semantic security. To simulate EDB, the simulator $\mathcal{S}$ for CVX takes the Setup leakage as input:

$$\mathscr{L}_{\mathcal{S}}^{cvx} = \left(\mathscr{L}_{\mathcal{S}}^{mm}(\mathrm{MM}), \mathscr{L}_S^{\mathrm{cv}}(\mathrm{CV})\right), \tag{8}$$

computes $\mathrm{EMM} \longleftarrow \mathcal{S}_{\mathrm{MM}}\{\mathscr{L}_{\mathcal{S}}(\mathrm{MM})\}$, $\mathrm{ECV} \longleftarrow \mathcal{S}_{\mathrm{CV}}(\mathscr{L}_{\mathcal{S}}^{bv}(\mathrm{CV}))$, and outputs EDB = (ECV,EMM).

If a token was simulated, it takes the Query leakage as input:

$$\begin{aligned} & \mathscr{L}_{\mathcal{Q}}^{cvx}(\mathrm{EDB}, w) = \\ & \mathscr{L}_{\mathcal{S}}^{mm}(\mathrm{MM}_1), \\ & \mathscr{L}_{\mathcal{Q}}^{cv}(\mathrm{CV}, w_1, w_{2*i}, w_{2*i+1}), \\ & \quad \ldots, \\ & \mathscr{L}_{\mathcal{Q}}^{cv}\left(\mathrm{CV}, w_1, w_{q-1}, w_q\right), \\ & \mathrm{TagPat}_i(\mathrm{EDB}, w))_{1 \leq i \leq \lceil (q-1)/2 \rceil}, \end{aligned} \tag{9}$$

and then, a token $tk = (\mathrm{gtk}, \mathrm{ltk})$ can be acquired. For all $1 \leq i \leq \lceil (q-1)/2 \rceil$, we set $ltk = (\mathrm{stk}_1, \ldots, \mathrm{stk}_{\lceil (q-1)/2 \rceil})$.

The gtk can be simulated as

$$\mathrm{gtk} \longleftarrow S_{\mathrm{MM}}\left(\mathscr{L}_Q^{cv}(CV_i, w_1, w_{2*i}, w_{2*i+1}), \mathrm{cv}\right). \tag{10}$$

For all $1 \leq i \leq \lceil (q-1)/2 \rceil$, *stk* is simulated as

$$\mathrm{stk}_i = S_{\mathrm{MM}}\left(\mathscr{L}_Q^{cv}(\mathrm{CV}, w_i), (\mathrm{tag}_{id})_{id \in \mathrm{DB}(w_i) \cap \mathrm{DB}(w_j) \cap \mathrm{DB}(w_k)}\right). \tag{11}$$

As we know, for all probabilistic polynomial-time adversaries $\mathscr{A}$, the probability *Real(k)* outputs 1 and *Ideal(k)* outputs 1 is very close. That is, if the Σ_{CV} and $\Sigma_{\mathrm{MM}}$are adaptive security, the SKE is RCPA-security, and the F is pseudo-randomness, then the simulated EDB and *tk* are indistinguishable from the real EDB and *tk*. It shows that the leaked random *tag* is indistinguishable from the encrypted identifier in the *Real(k)* experiment. □

4. Improved CV Scheme (ICV)

4.1. Description of ICV. Firstly, we set $\#\mathrm{DB}(w)$ to be the weight of keyword w. When the average weight of all keywords is larger, the CVX will be less efficient. For this reason, we improve the CVX and propose ICV scheme, which is more efficient when the average weight is heavy.

Similar to CVX, ICV has three modules (Setup, Token, and Query) and five security algorithms that are Σ_{CV}, Σ_{MM}, private-key encryption scheme SKE, pseudo-random function F, and GetTag$\{\cdot\}$ function.

The biggest difference between CVX and ICV lies in the construct of DB. In CVX, the MM is the inverted index, while ICV uses two-dimensional inverted index to construct MM. That is, for each keyword pair (w_i, w_j), $1 \leq i, j \leq |W|$, compute the DB $(w_i \cap w_j)$, pad it to the value of MM $[i, j]$, and then we get a two-dimensional MM, i.e., each label $l_{i,j}$ corresponds to the set $\mathrm{DB}(w_i \cap w_j)$.

Obviously, we can query a keyword pair easily using the MM. For a single-keyword query, we only need to let $i = j$, that is, to acquire $\mathrm{DB}(w_i \cap w_i)$. For q-conjunctive query, where $q \geq 3$, we can parse it to $(q-2)$three-tuples,

$$w_1 \wedge w_2 \cdots \wedge w_q = (w_1 \wedge w_2 \wedge w_3) \wedge \cdots \wedge (w_1 \wedge w_2 \wedge w_q), \tag{12}$$

and then, query each three-tuple to get a set S_i, $1 \le i \le q-2$, and then output the result $S = \cap_{i=1}^{q-2}$.

The scheme is depicted as follows.

(i) *Setup*. In the Setup process, an index database is generated and encrypted. Firstly, given the security parameter k, we can compute the keys. Then, calculate the two-dimensional index database MM using the keyword/docID pairs. For every keyword pair $w_i, w_j \in W$, MM maps it to encrypted identifiers in DB $(w_i \cap w_j)$. Otherwise, given a new keyword w_k, compute a binary counter vector $cv_{i,j}$ with length $|\,\mathrm{DB}(w_i \cap w_j)\,|$; if $tag_{ii} \in \mathrm{DB}(w_i \wedge w_j \cdots \wedge w_k)$, set the $cv_{i,j}[ii]$ to be value "1"; otherwise, set it to "0." Up to now, CV maps each three-tuple (w_i, w_j, w_k) to a vector of binary integers with length $\#\mathrm{DB}(w_i \cap w_j)$.

Finally, encrypt the MM and CV using algorithms Σ_{MM} and Σ_{CV}, respectively. We can get EMM and ECV. All the output of Setup mainly contains the encrypted structures EDB = (EMM, ECV) and their keys.

(ii) *Token*. Given the key and a set of queried keywords $w = (w_1, \ldots, w_q)$, suppose $q \ge 3$, and the Token algorithm generates the token TK = (gtk, ltk). gtk is the global token obtained from (w_1, w_2) and is used to query the MM. ltk contains $q-2$ *subtokens*, $(stk_1, \ldots, stk_{q-2})$. Each stk_i is calculated from (w_1, w_2, w_i), $i = 3, \ldots, q$.

(iii) *Query*. We can get the results of Query through the following steps. First of all, input the EMM and global token gtk, we can get MM = DB $(w_1 \cap w_2)$. Then, parse the $ltk = (stk_1, \ldots, stk_{q-2})$, for each stk_i, query ECV and get encrypted counter vectors $ecv_i, i = 1, \ldots, q-2$, and decrypt them to cv_i using Σ_{CV}, separately. Finally, for each cv_i, run GetTag $\{\cdot\}$ to get the set S_i. If we have all the subtokens processed, compute the intersection $S = \cap S_i$ and output the set S.

The details of the ICV SSE scheme ICV = (Setup, Token, Query) are given in Algorithm 3.

4.2. Correctness and Efficiency. We now analyze the correctness and efficiency.

4.2.1. Correctness. To show the correctness of ICV, we consider the operations of the counter vectors. For a common conjunctive query $w = (w_1 \wedge \cdots \wedge w_q)$, we can get the following equation, which gives the correctness of the ICV.

$$\begin{aligned} T &= \bigcap_{i=1}^{q} \mathrm{DB}_i \\ &= (\mathrm{DB}(w_1) \cap \mathrm{DB}(w_2) \cap \mathrm{DB}(w_3)) \\ &\quad \cap (\mathrm{DB}(w_1) \cap \mathrm{DB}(w_2) \cap \mathrm{DB}(w_4)) \\ &\quad \cap \cdots \cap (\mathrm{DB}(w_1) \cap \mathrm{DB}(w_2) \cap \mathrm{DB}(w_q))) \\ &= (S \cap \mathrm{DB}(w_3)) \cap \cdots \cap (S \cap \mathrm{DB}(w_q)). \end{aligned} \tag{13}$$

4.2.2. Efficiency. The Query complexity of ICV is $O(q-2)$. The sizes of tokens are $O(q-2)$. The community complexity achieves optimal because the search result has no redundancy. The storage complexity is

$$O\left(\mathrm{strg}\left(\sum_{w,v} \#\mathrm{DB}(w \cap v)\right)\right) + \sum_{w,v}\left(C_{w,v} \times (\#co(w,v))\right), \tag{14}$$

where $\mathbf{C}_{w,v}$ is a constant relating to w, v and $strg$ is the storage complexity of Σ_{MM}, which is a black-box multi-map encryption scheme used in this paper.

4.3. Security Analysis. Because the CVX and the ICV use the same data structures MM and CV and use the same algorithm for security, their safety performance is equivalent and can be proved in the same way.

5. Hidden Result Pattern CV SSE (HCV)

In this section, we propose our new hidden result pattern conjunctive keyword query scheme. This scheme employs Yin et al.'s scheme [17] to decrease the result pattern leakage. Unlike Lai's HXT scheme, we do not use the Bloom filter to construct index database, so our HCV can achieve accurate query and even ideal leakage when $q = 3$. The details of HCV are given as follows.

5.1. Description of HCV. HCV mechanism consists of three modules: Setup, Token, and Search. In Setup module, an index database DB was generated and encrypted to EDB. Token outputs a token tk for searching with the key and keyword set. Query returns the queried result once given the token and EDB. These modules contain several algorithms:

(i) A hidden vector CV encryption scheme $\sum_{\mathrm{HCV}}$ = (Setup, Token, Get), which uses BGN homomorphic encryption technique to protect its privacy.

(ii) A black-box multi-map encryption scheme $\sum_{\mathrm{MM}}$ = (Setup, Token, Query).

(iii) A private-key encryption scheme SKE = (Gen, Enc, Dec), which is RCPA-secure.

```
(1)  —Setup
     Input: $k$, inverted index
     Output: $K$, EDB
(2)  a. initialize a multi-map $MM$,
(3)     for all $w_i, w_j \in W, 1 \le i, j \le \#W$,
(4)        compute $\mathrm{MM}(i, j) \longleftarrow \mathrm{DB}(w_i \cap w_j)$;
(5)  b. initialize a binary integer vector $cv$ with length $\#\mathrm{DB}(w_i \cap w_j)$,
(6)     for all $w_k, k \ne i, j$,
(7)        compute $S_{i,j,k} = \mathrm{DB}(w_i \cap w_j \cap w_j)$;
(8)        for each $s \in S_{i,j,k}$,
(9)           $cv[\mathrm{MM}(i, j)^{-1}(s)] = 1$;
(10) c. compute $(K_g, \mathrm{EMM}) \longleftarrow \sum_{\mathrm{MM}}.\mathrm{Setup}(1^k, \mathrm{MM})$;
(11)    compute $(K_l, \mathrm{ECV}) \longleftarrow \sum_{\mathrm{CV}}.\mathrm{Setup}(1^k, \mathrm{CV})$;
(12)    set $K = (K_g, K_l)$ and $\mathrm{EDB} = (\mathrm{EMM}, \mathrm{ECV})$;
(13)    return $(K, \mathrm{EDB})$.
(14) —Token$(K, w)$:
     Input: a subset $V = \{w_1, \ldots, w_q\}$
     Output: token tk
(15) compute global token $tk \longleftarrow \mathrm{hash}(w_1 \| w_2)$
(16) compute local token $ltk = (stk_1, \ldots, stk_{q-2})$:
(17)    for $i = 1, \ldots, q - 2$, do
(18)    $stk_i = \mathrm{has}h2(gtk \| \mathrm{hash}1(w_{i+2}))$;
(19) output $tk = (gtk, ltk)$
(20) return tk;
(21) —Query$(\mathrm{EDB}, tk)$:
(22) parse EDB as $(\mathrm{EMM}, \mathrm{ECV})$;
(23) parse $tk$ as $(gtk, ltk)$, parse $ltk$ as $(stk_1, \ldots, stk_{q-2})$;
(24) (a)compute $T \longleftarrow \sum_{\mathrm{MM}}.\mathrm{Get}(\mathrm{EMM}, gtk_i)$;
(25) (b)for all $j = 1, \ldots, q - 2$, compute
(26)    $cv_j \longleftarrow \sum_{\mathrm{CV}}.\mathrm{Get}(\mathrm{ECV}, stk_j)$;
(27) (c)compute $t = cv_1 \& \cdots \& cv_{q-2}$,
(28)    $S = \mathrm{GetTag}(t)$
(29) output $S$;
```

Algorithm 3: ICV SSE.

(iv) A pseudo-random function F.

The project is depicted as follows.

5.1.1. Setup. In the Setup process, an index database is generated encrypted. Firstly, it generates a MM and CV database using the same method as $\sum_{\mathrm{CVX}}$.Setup. Then, encrypt the MM and CV, respectively.

MM is encrypted using the existing black-box encryption scheme Σ_{MM}, outputting EMM. CV is encrypted using Σ_{HCV} based on BGN homomorphic encryption technique. All the output of Setup mainly contains the encrypted structures EDB = (EMM, ECV) and their keys.

Σ_{HCV} is depicted as follows.

(a) Initialize: Compute a pair of public key $pk = (N, G, G_T, e, g, h)$ and the private key $sk = p$, according to the given security parameter k. Further, a one-way hash function $H: \{0,1\}^* \longrightarrow Z_N$ was initialized by the data user, and three random numbers R_0, R_1, and R_2 were chosen from Z_N. The tuple $\{sk, R_0, R_1, R_2\}$ and keyword dictionary W are kept secret, and the key $\{pk, H\}$ is sent to the cloud server.

(b) Encrypt CV: for each three-tuple $w = (w_{i_1}, w_{i_2}, w_{i_3})$, we can get a cv $t_w = \{t_1, \ldots, t_n\}$, $t_i \in \{0, 1, 2\}$, and the user encrypts it as

$$c_i = e(g, g)^{H\left(R_{t_i} \| i\right) \cdot q}, \tag{15}$$

where R_0, R_1, R_2 are three random numbers owned by the data user. Then, each counter vector t_w can be encrypted to $c = \{c_1, \ldots, c_n\}$, and all of the encrypted counter vectors constitute the ECV.

(c) Encrypt files: encrypt each file $f_j \in D$ using the existing SKE encryption scheme, and then send them to the cloud server, as well as the EMM and ECV.

5.1.2. Token. It is the same as CVX.

5.1.3. Query. The conjunctive query between client and server can be done as follows.

(a) The client sends the token tk to server. Server parses it as some subtokens $\{stk_i\}$, for each i, query the ECV, and if one of the queries returns $\varnothing$, then return $\varnothing$ to the client and end the search. Otherwise, send a new query request command to client.

(b) Once received query request, client initializes a integer vector V of length n, sets each element of V to a fixed value, which can be chosen from $\{0, 1, 2\}$, chooses n random numbers $r_0, r_1, \ldots, r_n$, and encrypts the vector V as $E(V)$, where

$$E(V)[i] = g^{H(R_v \| i) + p \cdot r_i}. \quad (16)$$

Client sends the subtokens and $E(V)s$ to the server.

(c) After receiving the subtokens and $E(V)s$, the cloud server performs the search operation. For each subtoken stk_i, the server initializes a $1 \times n$ vector P_i, computes $T^i = \mathrm{ECV}[stk_i]$, and then for each $0 \le j \le n-1$, computes $e(E(V)[j], h)$ and matches it with the jth element of T^i. If $e(E(V)[j], h) == T^i[j]$, it sets $P[j] = 1$; otherwise, it sets $P[j] = 0$. Then, the server gets each P_i using the same method and computes the intersection of $P = \cap P_i$.

(d) Finally, the server computes $S = \mathrm{GetTag}\{P\}$ and outputs S as the search result.

The details of the HCV SSE scheme HCV = (Setup, Token, Query) are given in Algorithm 4.

5.2. Correctness and Efficiency. In this subsection, we prove the correctness and analyze the efficiency.

5.2.1. Correctness. The correctness of HCV can be proved using the following equation. While it is true, the check operations in $\sum_{\mathrm{HCV}}$.Query is valid, and the result is correct.

$$\begin{aligned} e(E(V)[i], h) &= e\left(g^{H(R_v\|i)+pr_i}, g^q\right) \\ &= e(g,g)^{(H(R_v\|i)+pr_i)\cdot q} \\ &= e(g,g)^{(H(R_v\|i)q+pqr_i)} \\ &= e(g,g)^{(H(R_v\|i)q)} \\ &= T[i]. \end{aligned} \quad (17)$$

5.2.2. Efficiency. The search complexity of HCV is $O(\lceil (q-1)/2 \rceil)$. The sizes of Tokens are $O(\lceil (q-1)/2 \rceil)$. The communication complexity achieves optimal because the search result has no redundancy. For CVX and HCV, they have the same CV data structure before encryption, while the encryption algorithm of CVX keeps the length of encryption unchanged, and HCV extends the length of encryption to N times, so the storage complexity of HCV is N times over CVX, where N is determined by HCV's security parameter k.

5.3. Security Analysis. In this section, the privacy and security of the HCV can be analyzed. We keep our eyes on two types of security: one is the RP leakage, and the other is the privacy of outsourcing data and conjunctive queries.

5.3.1. RP Leakage Comparison. We consider two leakage components: KPRP and WRP, and illustrate the difference. For example, given a database containing six documents labelled by $\{id_i\}_{1 \le i \le 6}$ and each document contains some keywords. The details are listed in Table 3.

Consider the conjunctive query $w_1 \wedge w_2 \wedge w_3$. Suppose the keyword w_1 is the least frequent. For the three queried keywords, the inverted indexes are listed below: $\mathrm{DB}(w_1) = \{id_1, id_4, id_5\}$, $\mathrm{DB}(w_2) = \{id_1, id_2, id_4, id_6\}$, and $\mathrm{DB}(w_3) = \{id_2, id_4, id_5, id_6\}$.

Firstly, we compute the RP leakage component in Cash's OXT:

$$\mathrm{RP} = \bigcup_{j=2}^{3} \left(\mathrm{DB}(w_1) \cap \mathrm{DB}(w_j)\right) = \{id_1, id_4\} \cup \{id_4, id_5\}. \quad (18)$$

As shown in Table 4, the RP leakage reveals 4 entries of the inverted index.

Lai's HXT protocol eliminates the "partial query" (KPRP) leakage, which is a part of RP leakage. It only has the leakage of whole result pattern (WRP). Actually, in HXT, $\mathrm{WRP} = \cap_{j=1}^{3} \mathrm{DB}(w_j)$. The HXT protocol reveals the exact result of the query, that is, only $\{id4\}$. However, the HXT protocol leaks two entries w_2 and w_3 as shown in Table 4.

Our scheme HCV reveals the exact result $\{id4\}$, which is ideal leakage. Besides, HCV reveals nothing about the entry. The server only gets the encrypted keyword-tuples. The leakage comparison is given in Table 5.

5.3.2. HCV Privacy-Preserving

(1) Outsourcing Data Is Privacy-Preserving. In HCV scheme, the outsourced data include a file collection and an index database, both of which are encrypted. The encrypted files are secure because the server does nothing to them. Consider the encrypted index, and it includes two data structures, MM and CV data structures. For MM, we use a black-box MM encryption scheme, and the security is not discussed here. For the CV structure, as described in Section 5.1, it contains n encrypted elements represented as $e(g,g)^{H(R_{t_i}\|i)q}$, $i = 1, 2, \ldots, n$, where $t_i \in \{0, 1, 2\}$. Theorem 2 shows that the cloud server can get nothing related to t_i.

Theorem 2. *If H is a secure one-way hash function, the cloud server can get nothing related to $t_i \in \{0, 1, 2\}$ from the encrypted CV database.*

```
(1) —**Setup**
    **Input**: Security Parameter $k$, Inverted Index
    **Output**: $K$, EDB
(2) Generate key $\mathbf{K} = \{K_g, K_l\}$ according to $k$;
(3) Get the $EMM$ as in $\sum_{\mathrm{CVX}}$.Setup;
(4) Compute CV database using Algorithm 1;
(5) Encrypt CV as equation (15) to acquire $ECV$;
(6) set $K = \{K_g, K_l\}$, EDB = {EMM, ECV}
(7) return $(K, \mathrm{EDB})$.
(8) —**Token**
(9) The same as CVX.
(10) —**Query**;
    **Input**: tk, EDB
    **Output**: result
(11) initialize a $1 \times n$ tag vector $S$;
(12) initialize a $1 \times n$ bool vector $P$;
(13) initialize a $1 \times n$ integer vector $T$;
(14) (a) Server parse $tk = (gtk, ltk)$, and parse $ltk = \{stk_1, \ldots, stk_{\lceil (q-1)/2 \rceil}\}$;
(15) **for** $l = 1$: $\lceil (q-1)/2 \rceil$ **do**
(16)   **if**$[stk_l] == \varnothing$ ;
(17)     **return**result = $\varnothing$;
(18) Server send *Query Request* command to Client.
(19) (b) **for** $l = 1$: $\lceil (q-1)/2 \rceil$ **do**
(20)   set a vector $V_l$ and encrypt it to $E(V_l)$ as equation (16), send $stk_l$ and $E(V_l)$ to Server.
(21) (c) Server do as follows:
(22) set each element of $P$ to integer 1 **for**$l = 1$: $\lceil (q-1)/2 \rceil$ **do**
(23)   set $T = \mathrm{ECV}[stk_l]$,
(24)   **for** $i = 0$: $n - 1$ **do**
(25)     **if** $T[i] == e(E(V_l)[i], h)$ then
(26)       $P[i] = P[i]\&1$;
(27)     **else**
(28)       $P[i] = 0$;
(29) Finally, the Server computes $S = \mathrm{GetTag}\{P\}$;
(30) **return**$S$
```

Algorithm 4: HCV conjunctive searchable encryption scheme.

Table 3: Counter vector.

$\mathrm{DB}(w_1)$	fid_1	fid_2	fid_3	fid_4	fid_5	fid_6	fid_7
$w_1 \cap w_2$	1	0	1	1	0	1	1
$w_1 \cap w_2 \cap w_3$	1	0	1	0	0	1	0
Counter vector	2	0	2	1	0	2	1

Table 4: The database.

ID	Keywords
1	w_1, w_2, w_6, w_7, w_8
2	w_2, w_3, w_4, w_5
3	w_4, w_5, w_6, w_7
4	w_1, w_2, w_3
5	w_1, w_3, w_6
6	w_2, w_3, w_7

Proof. Now we prove the correctness of the theorem.

On the one hand, for each encrypted element $e(g, g)^{H(R_{t_i} \| i)q}$, to get the value of R_{t_i}, the server can inverse the function H or exhaust the value of R_{t_i}. However, H is a one-way function, it cannot be inverted. Meanwhile, R_{t_i} is a very large integer (i.e., 1024 bit). It is impossible currently to exhaust it. To guess the value, the cloud server only has 1/3 probability to get correct result for each element, as $t_i \in \{0, 1, 2\}$.

On the other hand, for different encrypted elements, e.g., $e(g, g)^{H(R_{t_i} \| i)q}$ and $e(g, g)^{H(R_{t_j} \| j)q}$, if $i \neq j$, the server cannot know whether t_i and t_j have the same value as H is a secure one-way hash function.

The cloud server gets nothing about t_i $(1 \le i \le d)$ from the encrypted tag array [17]. □

Table 5: Leakage comparison for query $w_1 \wedge w_2 \wedge w_3$.

RP (from OXT)	(id_1, w_2), (id_4, w_2), (id_4, w_3), (id_5, w_3)
WRP (from HXT)	(id_4, w_2), (id_4, w_3)
WRP (from HCV)	$(id_4, encrypted\ \ keyword\ \ tuple)$

(2) Conjunctive Query Is Privacy-Preserving. The conjunctive query contains three steps: query request, query processing, and query response.

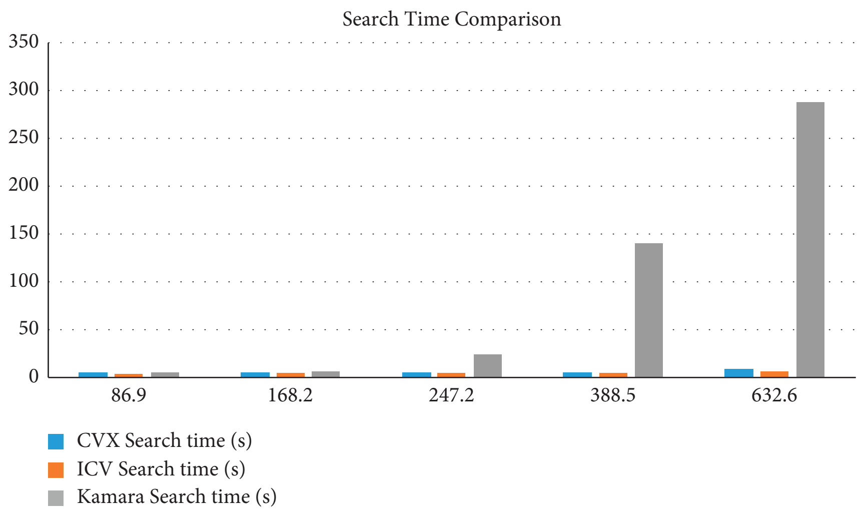

FIGURE 1: Low weight performance.

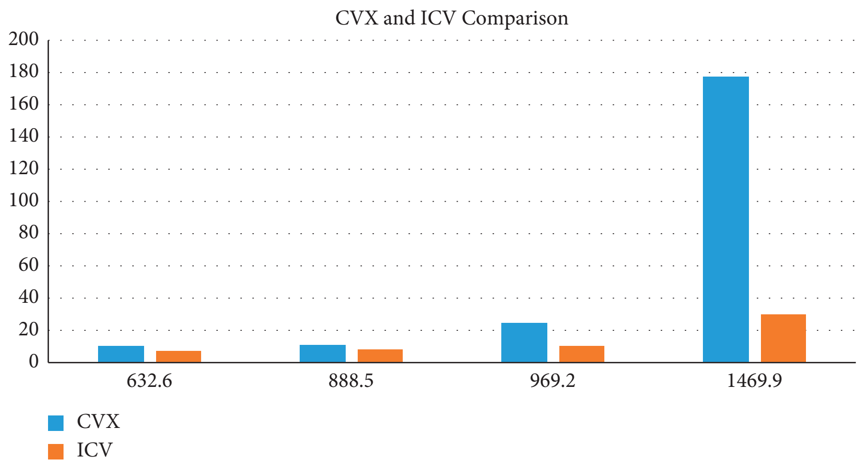

FIGURE 2: High weight performance.

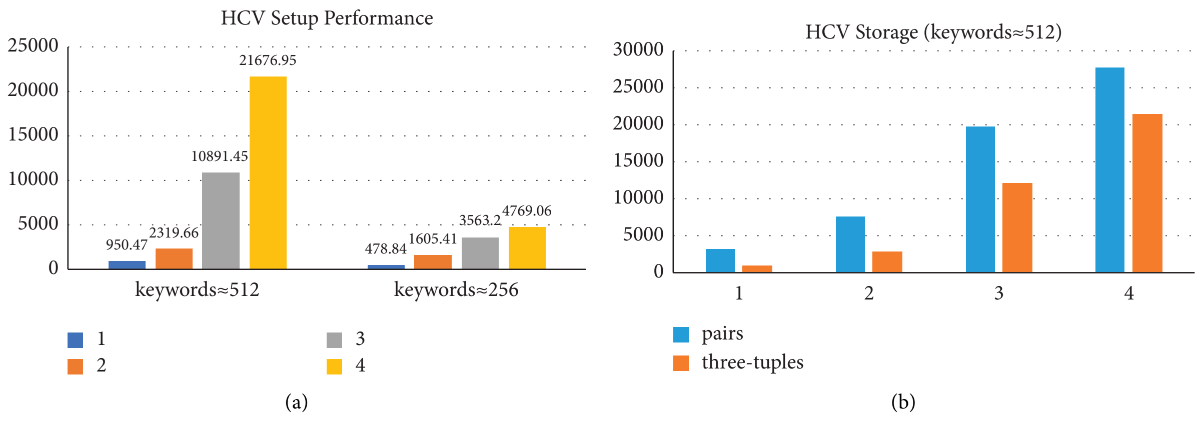

FIGURE 3: HCV performance. (a) Setup. (b) Storage.

In the query request, the query vector $Q = (k_0, k_1, \ldots, k_{d-1})$ is encrypted to be $E(Q) = (E(k_0), E(k_1), \ldots, E(k_{d-1}))$, where for each k_i, we have $E(k_i) = g^{H(R_{k_i} \| i) + p \cdot r_i}$. The encryption guarantees k_i is secure. Meanwhile, as each query uses a random value r_i, the encryption is nondeterministic. That is, for two queries, the honest-but-curious server cannot determine whether they have the same queried keywords.

For the query processing, the server receives the query vector $E(Q) = (E(k_0), E(k_1), \ldots, E(k_{n-1})$ which is encrypted, calculates $e(E(k_i), h)$, and matches the result with the ith element in the encrypted *tag* array. In this process, the server can only get information about the access pattern and search pattern, which our scheme allows.

For the query response, the search result is returned to the client by the cloud server. The search result consists of some encrypted files, and the encryption algorithm can be nondeterministic, e.g., AES-CBC. Thus, the adversary cannot correlate two queries even if they use the same queried keywords [17].

6. Experiment Evaluation

All experiments were run on Intel(R) Core(TM) i5-8400 CPU@2.80 GHz processor with 8 GB RAM. We use a commodity Windows 10 system. For CVX and ICV scheme, we implement our work in Python, while for HCV, we did the experiment in C++ for efficiency. To show our solution's practicability, the data used in the experiment are from the NSF e-mail dataset. The whole set contains 30799 keywords and 49078 files. We sampled and selected some of them for the experiment.

We use the common hash algorithm HMAC-SHA1, and the encryption algorithm for CVX and ICV is AES-CBC. In both the CVX and ICV schemes, Setup, Token, and Search algorithms are contained. While Setup and Token algorithms are implemented locally by the client, Query is implemented by the server. We assume that the client has sufficient computing resources, so only the efficiency of Query is considered here. In HCV scheme, we use the homomorphic encryption algorithm BGN and mainly experiment on the Setup performance. We generate the BGN parameters through type a1 pairing in PBC library, based on the curve $y^2 = x^3 + x$ (the group order N is a 1024 bit number).

Firstly, we test the performance of CVX and ICV, and we compare them with Kamara's IEX. To perform a search, Kamara's scheme needs to perform $q - 1$ decryption and then compute the intersection of $q - 1$ sets. CVX only needs to perform $\lceil (q-1)/2 \rceil$ decryption, and then do the intersection of $\lceil (q-1)/2 \rceil$ vectors. Take $q = 4$ for example, and we implement 2^{20} experiment. For each keyword w, we assumed that $\mathrm{DB}(w)$ represents the weight of w. Obviously, search efficiency is closely related to keywords' weight. The larger the keywords' weight, the less efficient the query operation.

For the sake of simplification, we use the mean value of all keywords' total weight as the metric to test the Query efficiency. In the case of low weight, we compare the Search efficiency of CVX and ICV with Kamara's scheme. The experiment shows that Search efficiency of CVX and ICV is not significantly different. However, both of them are dramatically better than Kamara's IEX. The experimental results are shown in Figure 1. The abscissa is the mean value of keyword weight, and the ordinate is the Query time in seconds. The keywords sampled range from 756 to 779.

Then, we find that when the weight mean value is larger, the CVX is not so efficient, so we proposed ICV. When the weight is larger, the advantage of ICV is obvious. Compared with CVX, ICV is more suitable for larger mean weight value. Figure 2 shows the experimental results. The abscissa is the mean value of keyword weight, and the ordinate is the *Query* time in seconds. The keywords sampled range from 203 to 537.

Secondly, we test the performance of HCV, which uses BGN homomorphism technology with high computational complexity to encrypt the index database. Therefore, we focus on the experiment of Setup which contains generating CV database and encrypting CV using BGN algorithm. We mainly tested the time complexity and storage complexity of Setup, both of which are closely related to the number of keywords and the weight of keywords. Our experiments are based on the sampled NSF dataset, which can be divided to two types. One contains about 512 keywords, and the other contains about 256 keywords. Our experiments mainly depict how the keyword number and weight affect the Setup efficiency. When the number of keywords is about 512, we set the mean weight as 23/51/83/112. When the sampled keyword number is about 256, we set the mean weight as 53/76/95/116. The experimental results are shown in Figure 3. Figure 3(a) describes the time complexity of HCV Setup process. The horizontal axis is the mean weight, and the vertical axis is the time of Setup process. If we fix the number of the keywords, the storage complexity depends on the number of counter vectors, mainly correlated to the keywords' weight. Figure 3(b) describes the relationship between the number of counter vectors and the keywords' weight, which can depict storage complexity. The horizontal axis is the mean weight, and the vertical axis is the number of the reserved counter vectors.

7. Conclusion

This paper proposed a novel CV data structure, which has been used in our proposed CVX, ICV, and HCV conjunctive query SE scheme. In CVX, the search efficiency is greatly improved compared with Kamara's IEX. Furthermore, when the weight of the keywords is larger, the search efficiency will decrease dramatically. So, we improved the CVX and proposed the ICV, which is more efficient but leaks more information than CVX. In HCV, we use the homomorphic encryption technology (BGN) to encrypt the CV data structure for the sake of resisting the RP information leakage. Security analysis shows that our scheme is secure, and performance evaluation also validates its efficiency. However, homomorphism encryption is used in our scheme and causes large computation and heavy storage, so our scheme can only be used for small datasets. In future work, we will study the scalable scheme and consider more security properties.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Nature Science Foundation of China under grant nos. 62072466 and U1811462.

References

[1] A. Adya, W. J. Bolosky, M. Castro et al., “FARSITE: federated, available, and reliable storage for an incompletely trusted environment,” in *5th Symposium on Operating System Design and Implementation (OSDI 2002)*, D. E. Culler and P. Druschel, Eds., pp. 9–11, USENIX Association, Boston, Massachusetts, USA, 2002.

[2] I. Clarke, O. Sandberg, B. Wiley, W. Theodore, and H.. Freenet, “A distributed anonymous information storage and retrieval system,” in *Proceedings of the Designing Privacy Enhancing Technologies, International Workshop on Design Issues in Anonymity and Unobservability*, pp. 46–66, Springer, Berkeley, CA, USA, July 2000.

[3] Y. Wang, J. Wang, S. Sun, M. Miao, and X. Chen, “Toward forward secure SSE supporting conjunctive keyword search,” *IEEE Access*, vol. 7, pp. 142762–142772, 2019.

[4] D. Xiaodong, D. Song, and W. A. Perrig, “Practical techniques for searches on encrypted data,” in *Proceedings the 2000 IEEE Symposium on Security and Privacy*, pp. 44–55, Berkeley, CA, USA, August 2002.

[5] Y.-C. Chang and M. Mitzenmacher, “Privacy preserving keyword searches on remote encrypted data,” in *Applied Cryptography and Network Security, Third International Conference, ACNS 2005*, John Ioannidis, A. D. Keromytis, and M. Yung, Eds., pp. 442–455, New York, NY, USA, 2005.

[6] M. Chase and S. Kamara, “Structured encryption and controlled disclosure,” in *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security*, M. Abe, Ed., pp. 577–594, Springer, Singapore, 2010.

[7] R. Curtmola, J. A. Garay, S. Kamara, and R. Ostrovsky, “Searchable symmetric encryption: improved definitions and efficient constructions,” in *Sabrina De Capitani di Vimercati*, A. Juels and R. N. Wright, Eds., pp. 79–88, ACM, Alexandria, VA, USA, 2006.

[8] E. Goh, “Secure indexes,” *IACR Cryptol. ePrint Arch.*vol. 216, p. 2003, 2003.

[9] S. Kamara, C. Papamanthou, and Tom Roeder, “Dynamic searchable symmetric encryption,” in *The ACM Conference on Computer and Communications Security, CCS’12*, Y. Ting, D. George, and V. D. Gligor, Eds., pp. 965–976, ACM, Raleigh, NC, USA, 2012.

[10] M. S. Islam, M. Kuzu, and M. Kantarcioglu, “Access pattern disclosure on searchable encryption: ramification, attack and mitigation,” in *19th Annual Network and Distributed System Security Symposium, NDSS 2012*, The Internet Society, San Diego, California, USA, 2012.

[11] D. Cash, P. Grubbs, J. Perry, and T. Ristenpart, “Leakage-abuse attacks against searchable encryption,” in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, I. Ray, N. Li, and C. Kruegel, Eds., pp. 668–679, ACM, Denver, CO, USA, 2015.

[12] Y. Zhang, J. Katz, and C. Papamanthou, “All your queries are belong to us: the power of file-injection attacks on searchable encryption,” *25th USENIX Security Symposium, USENIX Security 16*, pp. 707–720, USENIX Association, Austin, TX, USA, 2016.

[13] P. Golle, J. Staddon, R. Brent, and Waters, “Secure conjunctive keyword search over encrypted data,” in *Applied Cryptography and Network Security, Second International Conference, ACNS 2004*, M. Jakobsson, M. Yung, and J. Zhou, Eds., pp. 31–45, Springer, Yellow Mountain, China, 2004.

[14] 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 *Annual Cryptology Conference*, pp. 353–373, Springer, 2013.

[15] S. Lai, S. Patranabis, S. Amin et al., “Result pattern hiding searchable encryption for conjunctive queries,” in *ACM SIGSAC Conference on Computer and Communications Security, CCS 2018*, D. Lie, M. Mannan, M. Backes, and X.F. Wang, Eds., pp. 745–762, ACM, Toronto, ON, Canada, 2018.

[16] S. Kamara and T. Moataz, “Boolean searchable symmetric encryption with worst-case sub-linear complexity,” in *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 94–124, ACM, 2017.

[17] F. Yin, Y. Zheng, R. Lu, and X. Tang, “Achieving Efficient and Privacy-Preserving Multi-Keyword Conjunctive Query over Cloud,” *IEEE Access*, vol. 99, 2019.

[18] S. Lai, S. Patranabis, S. Amin, J. K. Liu, and Z. Cong, “Result pattern hiding searchable encryption for conjunctive queries,” in *Proceedings of the 2018 ACM SIGSAC Conference*, Toronto, Canada, October 2018.

[19] S. Sun, J. K. Liu, S. Amin, R. Steinfeld, and Tsz Hon Yuen, “An efficient non-interactive multi-client searchable encryption with support for boolean queries,” *IACR Cryptol. ePrint Arch.*vol. 1038, 2016.

[20] J. Wang, X. Chen, J. Li, J. Zhao, and J. Shen, “Towards achieving flexible and verifiable search for outsourced database in cloud computing,” *Future Generation Computer Systems*, vol. 67, pp. 266–275, 2017.

[21] Y. Wang, J. Wang, S. Sun et al., “Towards multi-user searchable encryption supporting boolean query and fast decryption,” *J. UCS*, vol. 25, no. 3, pp. 222–244, 2019.

[22] Y. Wang, S.-F. Sun, J. Wang, J. K. Liu, and X. Chen, “Achieving searchable encryption scheme with search pattern hidden,” *IEEE Trans. Serv. Comput.*vol. 15, no. 2, pp. 1012–1025, 2022.

[23] B. Wang, W. Song, W. Lou, and Y. Thomas Hou, “Inverted index based multi-keyword public-key searchable encryption with strong privacy guarantee,” in *Proceedings of the 2015 IEEE Conference on Computer Communications, INFOCOM 2015*, IEEE, Kowloon, Hong Kong, May 2015.

[24] C. Ma, Y. Gu, and H. Li, “Practical searchable symmetric encryption supporting conjunctive queries without keyword

pair result pattern leakage," *IEEE Access*, vol. 8, pp. 107510–107526, 2020.

[25] C. Fan, X. Dong, Z. Cao, and J. Shen, "VCKSCF: efficient verifiable conjunctive keyword search based on cuckoo filter for cloud storage," *IACR Cryptol. ePrint Arch*, vol. 1372, 2020.

[26] Q. Gan, J. K. Liu, X. Wang et al., "Verifiable searchable symmetric encryption for conjunctive keyword queries in cloud storage," *Frontiers of Computer Science*, vol. 16, no. 6, Article ID 166820, 2022.

[27] K. Zhang, J. Long, X. Wang, H. N. Dai, K. Liang, and M. Imran, "Lightweight searchable encryption protocol for industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 6, pp. 4248–4259, 2021.

[28] S. Kamara and T. Moataz, "Boolean Searchable Symmetric Encryption with Worst-Case Sub-linear Complexity," 2017, https://eprint.iacr.org/2017/126.

[29] D. Boneh, E. Goh, and K. Nissim, "Evaluating 2-dnf formulas on ciphertexts," in *Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005*, Joe Kilian, Ed., pp. 325–341, Springer, Cambridge, MA, USA, 2005.

[30] A. Menezes, C. Paul, van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, Boca Raton, FL, USA, 1996.