Verifiable Ranked Search Over Dynamic Encrypted Data in Cloud Computing

Presenter: Xiaohong Nie†

Joint work with

Qin Liu†, Xuhui Liu†, Tao Peng §, and Jie Wu¶

† College of Computer Science and Electronic Engineering, Hunan University, P. R. China, 410082

§ School of Information Science and Engineering, Central South University, P. R. China, 410083

¶ Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122, USA

Outline

- A : Introduction
- B : Preliminaries
- C : Main Idea
- D : Scheme Implementation
- E : Experiment Results
- F : Conclusion

A Introduction

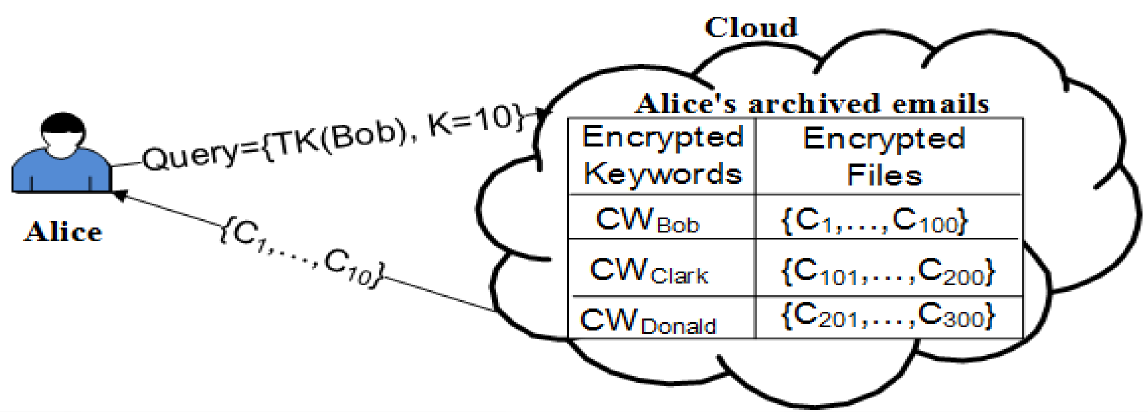
Introduction

- Existing research suggests encrypting data before outsourcing and adopting Searchable Symmetric Encryption (SSE) to facilitate keyword-based searches on the ciphertexts.

- However, no prior SSE constructions can achieve ***sublinear search time, efficient update and verification*, and *on-demand file retrieval***.

To address this, we propose our scheme.

Design Goal (Our scheme)



(1) *Ranked search*. The user is allowed to perform a top-K search to retrieve the best matched files.

(2) *Dynamic*. The user is able to update (add and delete) files stored in the cloud.

(3) *Verifiability*. The malicious CSP may delete encrypted files not commonly used to save memory space, or it may forge the search results to deceive the user.

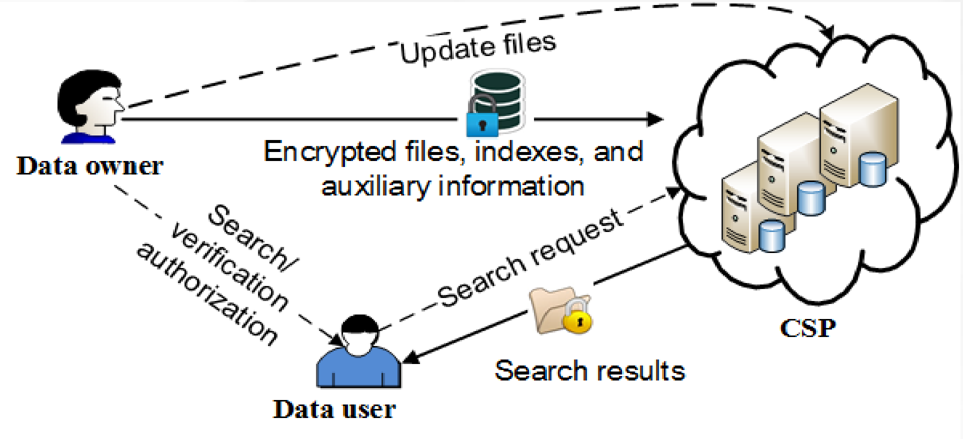
Contributions

- A verifiable, ranked, and dynamic SSE scheme to preserve big data security in a cloud environment.
- Allowing the user to efficiently update the file collection and verify the correctness of a top-K search while preserving user privacy from the CSP.

B

Preliminaries

System Model



1. The data owner creates ciphertexts $C = \{C_1, \ldots, C_n\}$ for each of file D_i , and then builds an encrypted index **I** and a verifiable matrix V from D, and the universal keywords $W = \{w_1, \ldots, w_m\}$.
2. The data owner performs updates (add/delete) on ciphertexts and retrieve the data of interest on *demand* in a *verifiable* way.
3. The CSP provides data storage and query services. The cloud users pay the services residing on the cloud or deploy their applications/systems in the cloud.

RSA Accumulator

RSA accumulator works as follows:

---For a set $E=\{y_1, y_2, ..., y_n\}$ with $y_i \in \{0,1\}^\lambda$,

---For each y_i, Alice chooses a prime $x_i \in \{0,1\}^{3\lambda}$ randomly.

Let prime(y_i) denote such a prime x_i.

$x_i = \text{prime}(y_i)$

--- Alice computes accumulated value of set E as

$Acc(E) = g^{x_1 x_2 \cdots x_n} \bmod N$ and sends $Acc(E)$ to Bob.

Later, Alice proves that $y_j \in E$ to Bob as follows：

--- She computes

$\pi_j = g^{x_1 x_2 \cdots x_{j-1} x_{j+1} \cdots x_n} \bmod N$

$x_j = \text{prime}(y_j)$

and sends π_j and x_j to Bob

--- Bob verifies that

$Acc(E) = \pi_j^{x_j} \bmod$

C

Main Idea

Main Idea

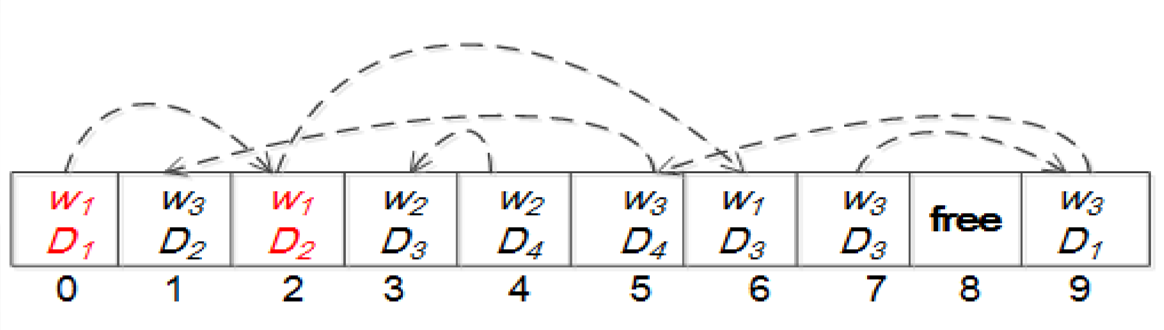
The search information in **I**:
build **a ranked inverted index I** from a collection of files to facilitate top-K searches.

The rank information in **V** :
build **a *verifiable matrix* V** for verifiable updates and searches.

Specifically, **I** contains multiple inverted lists, each linking a set of nodes that corresponds to one keyword. A list of nodes is chained according to their ranks for a specific keyword . The node’s prior/following neighbor will be recorded in **V** with the RSA accumulator.

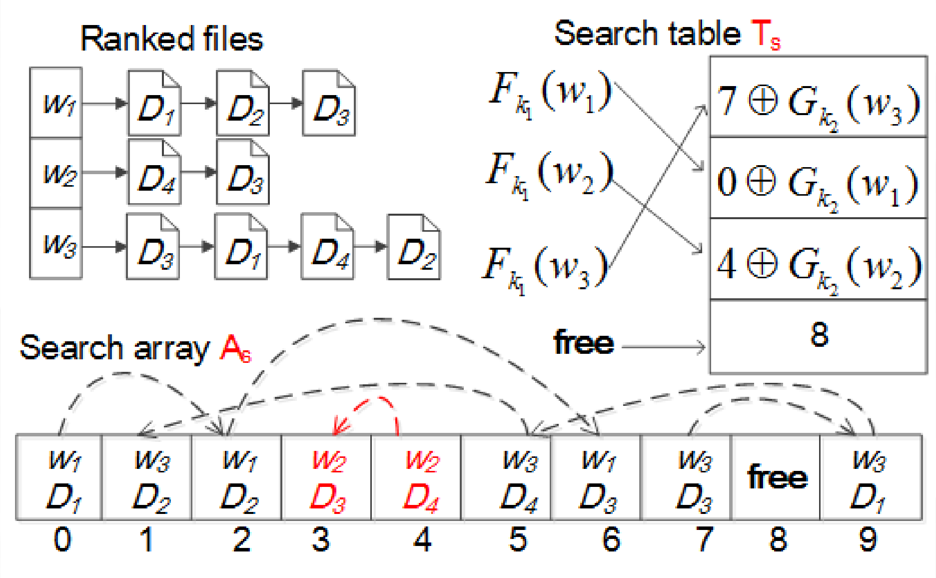
Main Idea (Ranked linked list)

L_w is composed of #w nodes $(N_1, \ldots, N_{\#w})$ and defined $N_j = \langle id_j, addr_s(N_{j+1})\rangle$, where $id_j \in ID(w)$ is the identifier of the rank-j file for keyword w and $addr_s(N_{j+1})$ is the address of node N_{j+1} in the search array A_s. In the special case, $N_{\#w} = \langle id_{\#w}, \mathbf{0}\rangle$.



Main Idea(Ranked Inverted Index)

- I= {T_s, A_s}:
- The ranked inverted index, where for each word w ∈ W, a list L_w of #w nodes are randomly stored in the search array A_s and the pointer to the head of L_w is included in the search table T_s.



Main Idea (Verifiable Matrix)

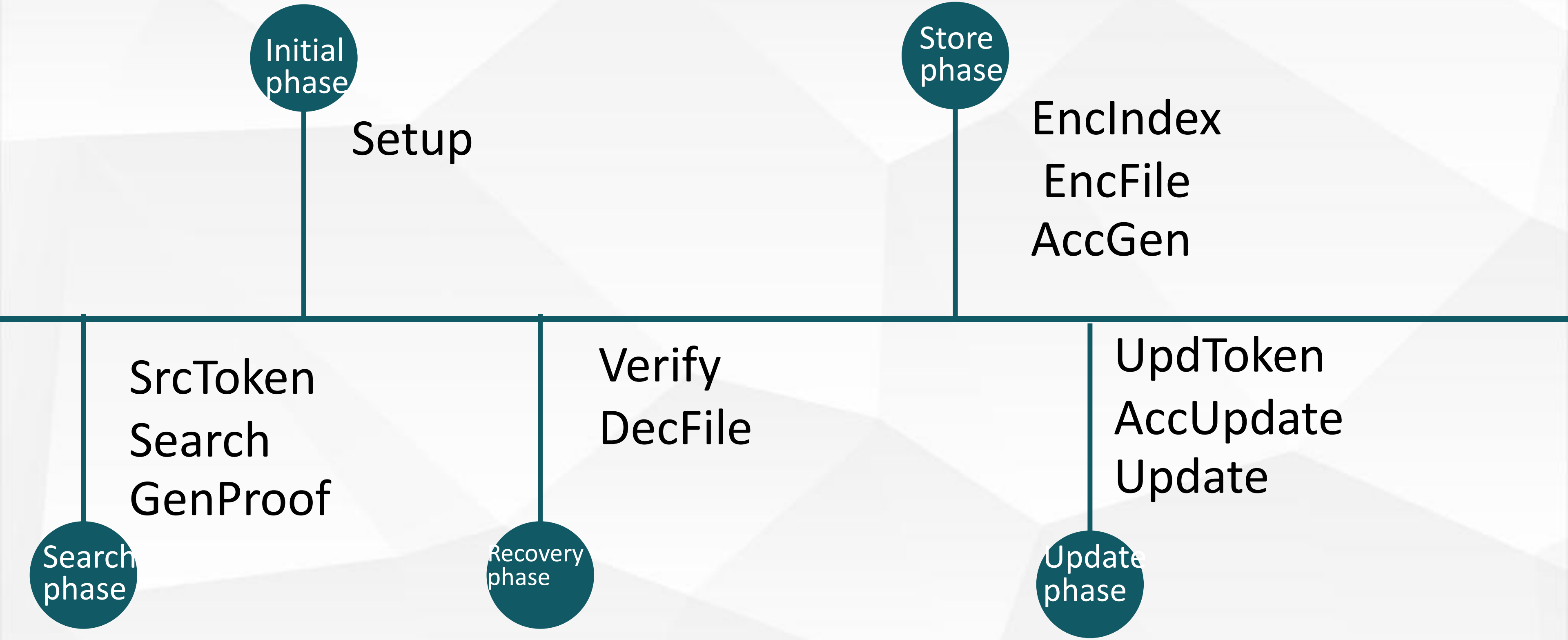
Since a keyword appears in n files at most , the verifiable matrix V is an $m \times n$ matrix, where row $i \in [1, m]$ corresponds to a keyword $w \in W$, and column $j \in [1, n]$ corresponds to a rank $j \in [1, n]$. The relationship between the row i and the keyword w is determined by the key-value pairs of the search table T_s.

$$
\begin{array}{c c}
 & \begin{array}{cccc} \text{rank-1} & \text{rank-2} & \text{rank-3} & \text{rank-4} \end{array} \\
\begin{array}{c} F_{k_1}(w_1) \\ F_{k_1}(w_2) \\ F_{k_1}(w_3) \end{array} &
\begin{pmatrix}
H(\mathbf{0},3,1) \oplus S_{k_4}(w_3) & H(3,1,4) \oplus S_{k_4}(w_3) & H(1,4,2) \oplus S_{k_4}(w_3) & H(4,2,\mathbf{0}) \oplus S_{k_4}(w_3) \\
H(\mathbf{0},1,2) \oplus S_{k_4}(w_1) & H(1,2,3) \oplus S_{k_4}(w_1) & \oplus S_{k_4}(w_1) & \mathcal{R}_{2,4} \\
H(\mathbf{0},4,3) \oplus S_{k_4}(w_2) & H(4,3,\mathbf{0}) \oplus S_{k_4}(w_2) & \mathcal{R}_{3,3} & \mathcal{R}_{3,4}
\end{pmatrix}
\end{array}
$$

D

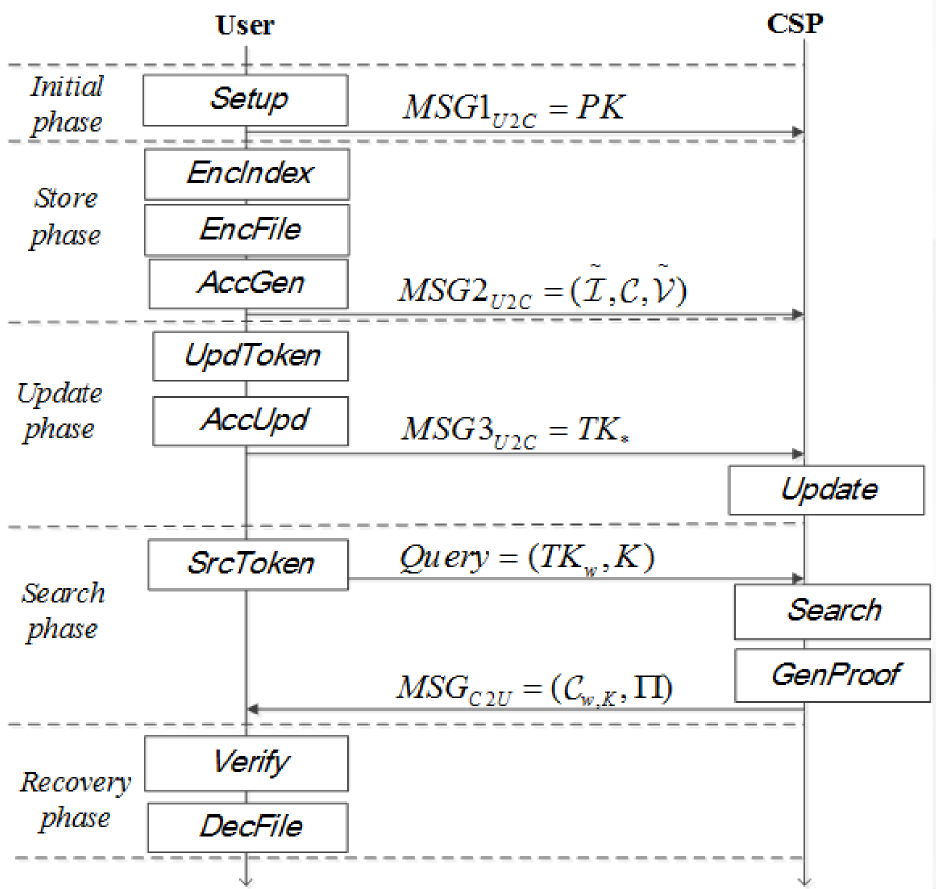
Implementation Scheme

Implementation Scheme



Implementation Scheme

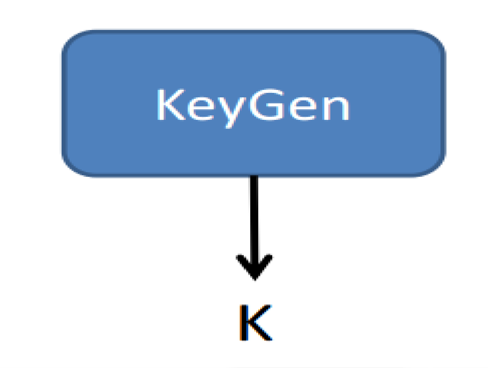
- (1) Initial phase
- (2) Store phase
- (3) Search phase
- (4) Recovery phase
- (5) Update phase



Initial phase

- The user randomly chooses four κ-bit strings k_1, k_2, k_3, k_4 as keys of PRFs, runs SKE.Gen(1^κ) to generate k_e, and generates ($\mathbf{N} = pq, g$) . Let $P(y)$ be a random prime x such that $f(x) = y$. We have

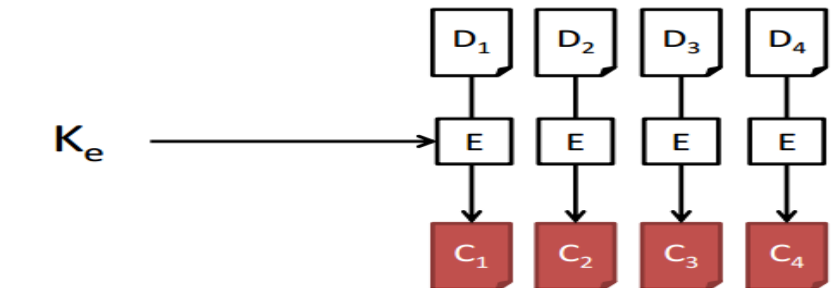
  PK = ($\mathbf{N}$, g, f)
  SK = (p, q, k_e, k_1, k_2, k_3, k_4)



Store phase

1:

For each file $D_i \in D$, the user runs $SKE.Enc(k_e, D_i)$ to generate the ciphertext C_i



2: The user computes:

$$A_c = g^{\prod_{i=1}^{4} \mathrm{prime}(H(i,H(c_i)))} \bmod N$$

$$A_I = g^{\prod_{i=1}^{3} \prod_{j=1}^{4} \mathrm{prime}(H(i,v[i][j]))} \bmod N$$

where A_C and A_I will be kept locally.

Search phase

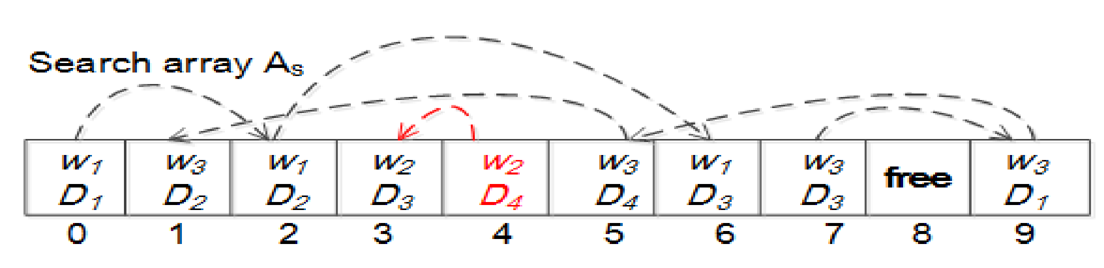
1:

Suppose that the user wants to retrieve top-1 files containing keyword w_2 :

She will send Query = $\{TK_{w_2}, 1\}$ to the CSP, where $TK_{w_2} = \{F_{k_1}(w_2), G_{k_2}(w_2), P_{k_3}(w_2)\}$ for $F_{k_1}(w_2) = 3$.

2:

The CSP locates $Ts[F_{k_1}(w_2)]$ and recovers the address of the first node containing keyword w_2 in As by computing $4 \leftarrow Ts[F_{k_1}(w_2)] \oplus G_{k_2}(w_2)$.



Search phase (GenProof)

3:

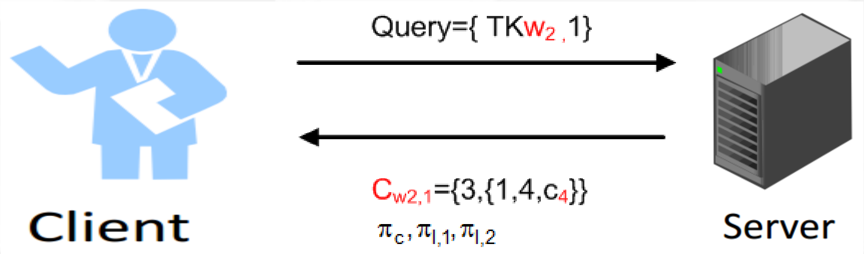
The CSP calculates proofs $\Pi = \{\pi_C, \pi_{I,1}, \pi_{I,2}\}$:

$$\pi_c = g^{\prod_{i=1,2,3} p(H(i,H(C_1)))}$$

$$\pi_{I,1} = g^{\prod_{j=2,3,4} p(H(3,V[i][j]))}$$

$$\pi_{I,2} = g^{\prod_{i\neq 3}\prod_{j=1}^{4} p(H(3,V[i][j]))}$$

The message returned to the user is $\{Cw_{2,1}, \Pi\}$



Recovery phase

Verify :

The user computes $x = P(H(4, H(C_4)))$ and checks if:

$$A_c = \pi_c^{x} \bmod N$$

She reconstructs V[3][1] $=H(0, 4, 3) \oplus S_{k4}(w_2)$ from $Cw_{2,1}$, computes $z = P(H(3, V[3][1]))$, and checks if:

$$A_l = (\pi_{l,2})^{z \bullet \pi_{l,1} \bmod (p-1)(q-1)} \bmod N$$

DecFile:

The Verify algorithm is 1, the user runs SKE.Dec(k_e ,C_4)

Update phase

Update(I, C, V, TK_*) → (I′, C′, V′) : If the update token $TK_*(D)$ = $TK_{del}(D)$ = (i, **delete**), the CSP replaces the ciphertext C_i with **delete**. Otherwise, given $TK_*(D)$ = $TK_{add}(D)$ = {($n + 1$, C_{n+1}), C, τ_v, τ_a}, the CSP first adds C as the last column of the verifiable matrix, and then updates C to C′ by adding ($n + 1$, C_{n+1}) to C.

E

Experiment Results

Experiment Results

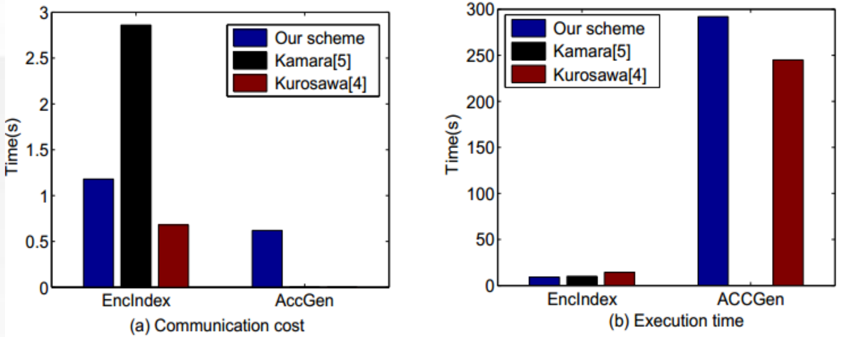


Fig. 1. Comparison in store phase.

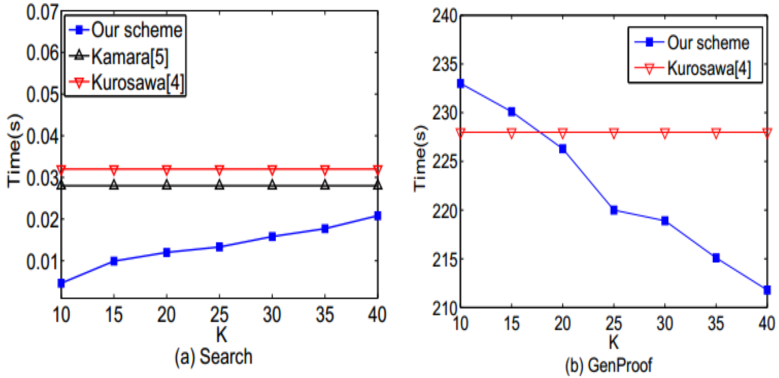


Fig. 2. Execution time in the search phase.

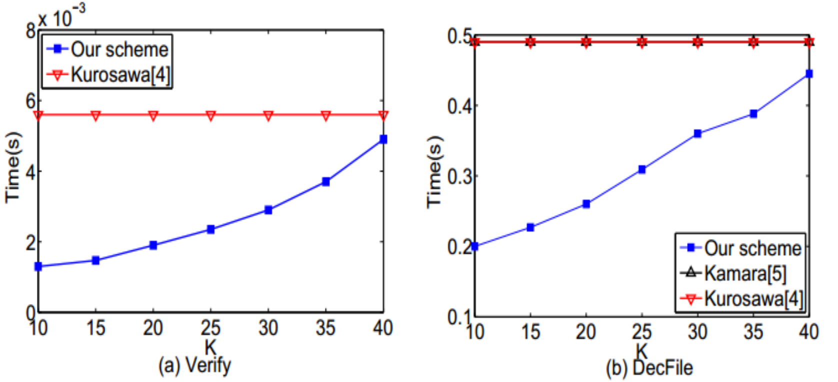


Fig. 3. Execution time in the recovery phase.

Experiment Results

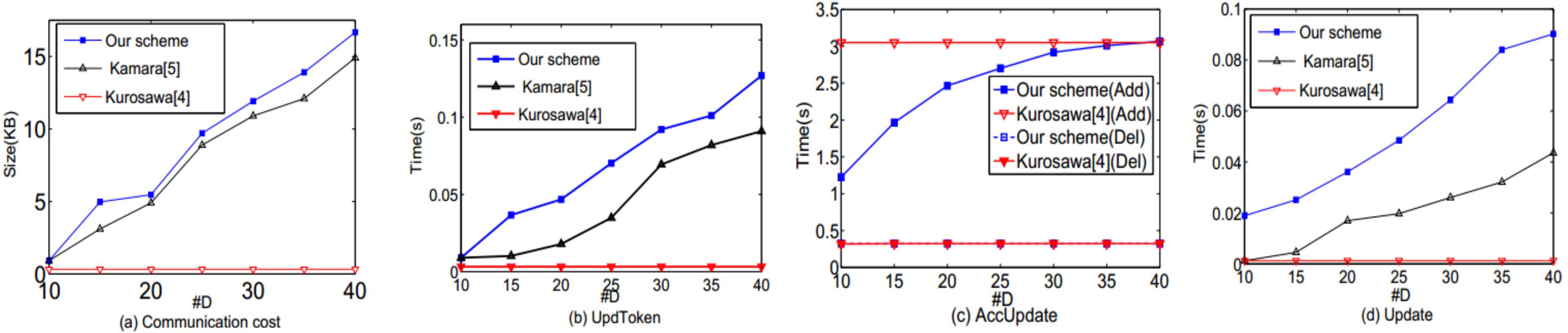


Fig. 4. Communication cost and execution time in the update phase.

F Conclusion

Conclusion

- A verifiable, ranked, and dynamic SSE scheme in a cloud environment.
- Verify the correctness of the top-K search and the integrity of a set of dynamic files .

- However, our VRSSE scheme supports only single-keyword searches. As part of our future work, we will try to design a multi-keyword VRSSE scheme to achieve conjunctive keyword searches.

THANKS