A secure search method for cloud data storage

Nguyen Dao Truong[1*], Doan Thi Bich Ngoc[2]

[1]Academy of Cryptography Technique, 141 Chien Thang, Phuong Mai, Hanoi, Vietnam;
[2]University of Information and Communication Technology, Thai Nguyen University, Z115 Road, Quyet Thang, Thai Nguyen, Vietnam.
[*]Corresponding author: truongnd-it@actvn.edu.vn
Received 5 Jun. 2025; Revised 7 Aug. 2025; Accepted 20 Sep. 2025; Published 2 Oct. 2025.
DOI: https://doi.org/10.54939/1859-1043.j.mst.106.2025.137-144

ABSTRACT

Storing data on cloud computing platforms offers significant benefits in terms of performance and scalability; however, it also poses serious challenges related to security and privacy. One of the key issues is enabling users to search over encrypted data without revealing its contents to the cloud service provider. In this paper, we propose a secure search method based on searchable encryption combined with a novel digital signature scheme, allowing users to perform keyword queries without decrypting the data. The proposed method is implemented and evaluated in a cloud environment, with results demonstrating acceptable performance and strong security against common attacks such as query analysis.

Keywords: Cloud storage; Searchable encryption; Secure search; Digital signature; Encrypted data search.

1. INTRODUCTION

In the context of data explosion and the trend of shifting to cloud storage, the issue of protecting data privacy and integrity is becoming increasingly urgent. One common solution is to encrypt data before storing it to prevent unauthorized access. However, encryption also significantly reduces the ability to process and query information, especially searching. Searchable encryption schemes [1-3] were created to solve this contradiction: allowing users to perform search queries directly on encrypted data while still ensuring security.

Searable Encryption (SE) provides a mechanism that combines encryption and searchability, allowing the system to return correct results without decrypting the entire data. There are two main types of SE: symmetric searchable encryption (SSE) [1, 4] and public-key searchable encryption (PEKS) [5-10], which are suitable for different application scenarios. Modern SE models also support searching by multiple conditions, approximate searching, or range searching.

With its outstanding advantages, SE is being widely applied in secure storage systems such as digital libraries, cloud storage services, and electronic medical records systems. However, there are still many challenges, such as balancing performance and security, the ability to resist query analysis attacks, or supporting more complex queries in real environments. In the following sections of the paper, we combine a searchable symmetric encryption method with a generative stream cipher, combined with a novel digital signature [11] to counter these attacks.

2. SEARCH SCHEME ON ENCRYPTED DATA

2.1. Basic search scheme

The basic scheme depicted in figure 1 provides a basic introduction to the method of encrypting a document before storing the data on the server. During the initialization process, the user uses a digital signature algorithm [11] to generate a signature, which is then appended to the document to create a signed document, D, and then chooses a block size n. Each word w in the document D is padded to length n or, if w is longer than the block size, it is split into multiple blocks. During the encrypting process, for each word w, a string of bits S of length n-m is generated using a pseudorandom G, where m is the output length of the pseudorandom function. When the word w_i

at position *i* is encrypted, S_i is generated and $T_i = \langle S_i, F_{k_i}(S_i) \rangle$ with the secret key k_i is generated. The ciphertext C_i is calculated by applying XOR operation on T_i and w_i. By this scheme, the user can choose to use the same *k* for all the words in the document or use a different key for each word.

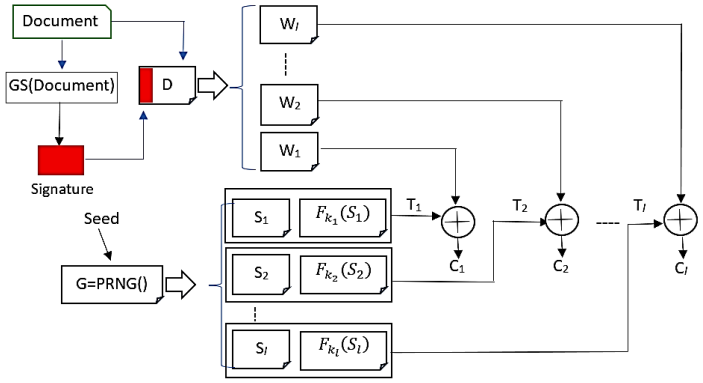


Figure 1. *The basic search scheme.*

When a user wants to search for a word in a document, the user sends the word to be searched for *w* and the key *k* to the server (cloud). Next, the server checks each word in each document to see if the current encrypted word is XORed with the search word in the form $T_i = \langle S_i, F_k(S_i) \rangle$. The encrypted word matches the search word if a valid *T* is found. The server can then return the document to the user. The user uses a signature verification algorithm [11] to verify the signature to ensure that the returned document is correct.

When the user wants to decrypt the returned document, he applies the decryption algorithm to recover the plaintext document. Since the XOR operator returns the original input if this operation is used twice, the user can reconstruct the keystream S (the user knows the seed), compute *T*, and XOR it with the ciphertext *C* to obtain the plaintext document.

There are two problems with this basic search scheme. First, the search term is displayed to the server in plaintext. Second, the user must provide the key *k* to the server when searching. The server decrypts the entire document if the same key *k* was used for all words. However, if the user chose a different key *k* for each word, then they would have to know the exact position where the word might appear and the key k for that word. This defeats the purpose of the search, so to correct this, the paper moves towards a second scheme in the next section.

2.2. Enhanced search scheme

The enhanced search scheme (depicted in figure 2) provides a simple extension to the previous controlled search scheme to hide searches from the server. This is achieved by pre-encoding the search term w using a block cipher in ECB mode before applying the stream cipher. The key k'' used for the block cipher must also be kept secret by the user. The key k is now calculated from $f_{k'}(X)$ where $X = E_{k''}(w)$.

We can see that by pre-encrypting each word with a block cipher in ECB mode before the main encryption, the user does not need to decrypt the documents anymore. This is because if users generate keys, $k_i = f_{k'}(E_{k''}(w_i))$, they will need to know $E_{k''}(w_i) = X_i$. This makes no sense because there would be no point in generating $T_i = \langle S_i, F_{k'}(S_i) \rangle$ to find X_i if the user knows X_i. In the completed search scheme (in the next section) depicted in figure 3, we have overcome this problem without compromising the security of the algorithm.

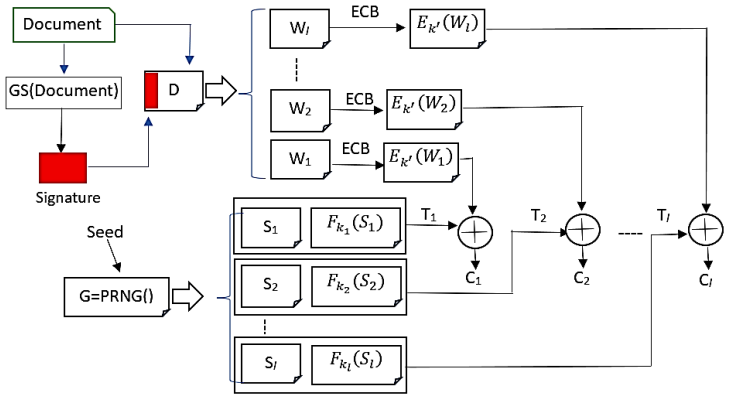


Figure 2. *Enhanced search scheme.*

2.3. Completed search scheme

The completed search scheme (as shown in figure 3) provides a simple fix for the problem described in the enhanced search scheme. In this scheme, the user must split the word X into two parts, the left half, L, and the right half, R. The left half, L, must have length $n-m$, the same length as S, and the right half, R, must have length m, the same length as $F_k(S)$. Next, instead of calculating $k = f_{k'}(X)$, the user has to calculate $k = f_{k'}(L)$.

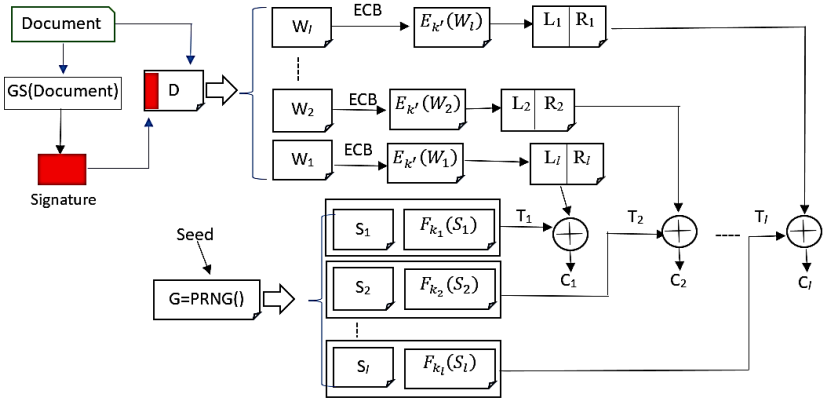


Figure 3. *The completed search scheme.*

The completed search scheme is summarized as follows: First, the user divides the signed document D into a set of words w of size n, where n is the block size. In English text, this usually means dividing the text per space character and padding the words to a length n. The user then iterates through the set of words and for each word w_i at position i, performs pre-encrypted it: $X_i = E_{k''}(w_i)$. The pre-encrypted word X_i is split into two halves L_i and R_i. The user then generates S_i as the $n-m$ length output of a pseudorandom generator G, with a randomly chosen secret seed, which is stored locally for use during decryption. The user computes $k_i = f_{k'}(L_i)$ and sets $T_i = \langle S_i, F_{k_i}(S_i) \rangle$. The current ciphertext block C_i is computed by XORing X_i and T_i.

When users want to search for a word, they pre-encrypt the word to search for w: $X = E_{k''}(w)$, then divide it into two parts, the left part L and the right part R, respectively, just like in the encryption process. Then, the user must calculate the key $k = f_{k'}(L)$ and send $\langle X, k \rangle$ to the server

(cloud). The server iterates over each block in each document and partially decrypts it to check whether the block has a valid, *T*. For each block of ciphertext C_i at position *i*, the server splits the block into two parts, a left part C_{i1} and a right part C_{i2}. Similarly, for the pre-encrypted search word, $X->[L,R]$. Then the server calculates $S_i = C_{i1}\text{XOR}\ L$. Next, the server calculates $F_k(S_i)$ and check if $F_k(S_i)\text{XOR}\,C_{i2}$ equals *R* (checks the validity of *T*). If so, then a match was found and the document is returned to the user.

When decrypting, the user generates S_i using the pseudorandom generator *G* and their secret seed, then computes $L_i = S_i\text{XOR}\ C_{i1}, k_i = f_{k'}(L_i), f_{k_i}(S_i)$ and $R_i = C_{i2}\text{XOR}\ F_{k_i}(S_i)$. The user then combines L_i and R_i to get X_i. Next, the block cipher is used to decrypt X_i into w_i.

3. PROVING THE SECURITY OF THE SEARCH SCHEME

Definition 3.1. Define F_k to be a mapping $F_k : N\times D\to Q\times R$ is defined as follows: $F_k(w)=(G_k(w),H_k(w))$, where we write $B_k(i)$ as a short form of the i-th block of $G_k(w)$.

Definition 3.2. We say that a distribution D over key $K = k_1, k_2, ..., k_n$ is mutually binding if for every index *i*, one of the following two conditions is satisfied:

1. (*Link to previous key*): There exists an index $j < i$ such that $\Pr[k_i = k_j] = 1$ i.e., the key at position *i* is exactly the same as the key at the previous position with probability 100%.

2. (*Select random independently*): The key k_i is chosen completely randomly from the set of possible keys, i.e., $k_i \sim U(K)$, where $U(K)$ represents a uniform distribution over the set of keys, *K*.

Lemma 3.1. If the key K is a pseudo-random function $(T,\varepsilon)-$ secure and G is a pseudorandom number generator $(T,\varepsilon')-$secure, then the function F_k is a $(T',\varepsilon'')-$secure pseudorandom number generator, where $T' = T - O(n), \varepsilon'' = \varepsilon + \varepsilon'$, and the constant $O(n)$ are negligible compared to *T*.

Proof: First, we define two probability distributions:

P_F : The probability distribution generated by the algorithm uses *K* and *G*.

P_U : Truly random distribution.

We need to show that there exists no algorithm that can distinguish these two distributions with an advantage greater than ε''.

Suppose there exists an algorithm *A* that can distinguish P_F and P_U with probability greater than ε'', i.e. $|\Pr[A(P_F)=1]-\Pr[A(P_U)=1]| > \varepsilon''$.

This means that algorithm *A* can tell the difference between output of F_k and completely random data, with an accuracy better than some ε'' value.

From here, we can construct an algorithm *B* to distinguish between the two components of F_k:

- Or distinguish *K* from a truly random function.
- Or distinguish *G* from a true random number generator.

Step 1. Prove that if A can distinguish P_F and P_U, then G can be distinguished by a true random number generator. Consider an algorithm B, which takes input from a random number generator G or a truly random number generator. B will run algorithm A on that input dataset. If the algorithm A can distinguish P_F from P_U, then this implies that B can also distinguish G from a true random number generator, with an advantage of at least ε'. This contradicts the assumption that G is a secure pseudorandom number generator.

Step 2. Prove that if the algorithm A can distinguish P_F and P_U, then it can distinguish K from a true pseudorandom function.

Similarly, suppose B' is an algorithm that can distinguish a pseudorandom function K from a truly random function. B' will use algorithm A to detect the difference. If A can distinguish between P_F and P_U, then B' can also distinguish between K with a true pseudorandom function with an advantage of at least ε. Therefore, by the reductio ad absurdum principle, algorithm A cannot exist. This proves that F_k is indeed a secure pseudorandom number generator.

Lemma 3.2. If Π is a pseudorandom function $(T,\varepsilon)-$secure, and G is a secure pseudorandom number generator $(T,\varepsilon')-$secure, then the function F_k is a pseudorandom number generator $(T',\varepsilon'')-$secure, where $T' = T - \mathrm{O}(n), \varepsilon'' = \varepsilon + \varepsilon'$, and the constant $\mathrm{O}(n)$ are negligible compared to T.

Proof: The proof is similar to lemma 3.1, except that the keys are chosen independently instead of being generated from a pseudorandom function.

Step 1: Define the two distributions to be compared..

Consider two probability distributions:

P_F : Output distribution of the algorithm using Π and G.

P_U : Distribution of a truly random number generator.

We need to prove that there is no algorithm that can distinguish P_F and P_U with probability greater than ε''.

Suppose there exists an algorithm A that can distinguish these two distributions with an advantage greater than ε'', that is: $\left|\Pr\left[A(P_F)=1\right] - \Pr\left[A(P_U)=1\right]\right| > \varepsilon''$.

Similar to the proof of Lemma 3.1, if algorithm A can distinguish P_F and P_U, then we can construct an algorithm B that can distinguish G from a true pseudorandom number generator or distinguish Π from a truly pseudorandom function..

Step 2: Analyze the difference between the key independent variant and the regular form.

In lemma 3.1, the keys are chosen through a pseudo-random function, i.e., there is a connection between the keys at different positions. However, in Lemma 3.1, the keys at each position are chosen completely independently, i.e.:

k_i is randomly selected from $\{0,1\}^n$ independently at each position i.

This independent key selection does not reduce the security of the scheme, since each key remains random to the attacker. Hence, similar to Lemma 3.1, if algorithm A can distinguish P_F

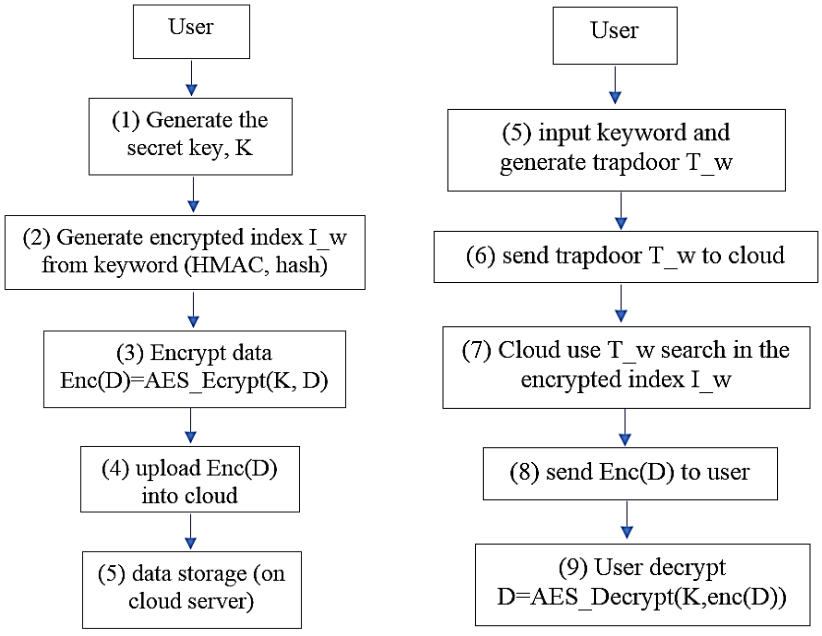
and P_U, we can construct an algorithm B that can break the security of either G or Π, which contradicts the assumption that both G and Π are secure. Therefore, by the reductio ad absurdum principle, algorithm A cannot exist, proving that F_k with independently chosen keys is still a secure pseudorandom number generator.

In addition, to defend against repeated query analysis attacks, we incorporated the generation of pseudorandom numbers in the construction of search tokens. Specifically, each token T_i is generated by XOR-ing a pseudorandom value with the ciphertext of the search keyword. As a result, the likelihood of successful repeated query analysis is significantly reduced in our proposed model.

4. EXPERIMENT

4.1. Deployment architecture

In the implementation of the algorithm, we use a client-server model solution. A server can have multiple clients and each client connects to a single server. The dataset used for testing consists of the first 128 poems from William Shakespeare's The Sonnets [12] collection as separate text files. The smallest file among the files is 659 bytes and the largest file is 719 bytes. The total size of all 128 files is 81.6 KB.



a. Cloud upload process *b. Search process*

Figure 4. *Cloud data storage and search deployment model.*

4.2. Experimental results

To provide more realistic results, the paper performed tests on the technical implementation itself. The tests analyzed the main functions of the scheme, specifically as follows:

Upload function: Performs a test encoding of 128 text files and sends them to the server.

Search function, when no documents match the query: Performs a test including creating a trapdoor with a fake word “abundances”, sending it to the server and performing a search query.

Search function, in case there are 96 matching files: The test includes creating a trapdoor with the word “and”, sending it to the server, performing a search query, returning 96 matching files and decoding them.

***Table 1**. Function execution time when executing with 128 files.*

Block size	*Upload*	*Search 0 file match (ms)*	*Search 96 file match (ms)*
16	2754	213	411
32	3592	401	579

Table 1 shows the execution time during encryption and upload of 128 text files, search with 0 matching files and 96 matching files on 128 encrypted text files with block sizes of 16 and 32. The results are the average of 10 separate executions.

***Table 2.** Storage space required for storing 128 files.*

Block size	*File size (KB)*	*Percent*
Plaintext	81.6	0%
16	232.4	285%
32	454.5	557%

Table 2 shows the storage space required to store 128 encrypted files with different block sizes. The smallest of the files is 659 bytes, and the largest is 719 bytes. The total size of all 128 files is 81.6 KB.

It is clear from the storage space requirements in table 2 that using a block size of 32 on this dataset will add a large amount of redundant data. Whether this is worth it is unclear and requires further testing. When most of the words in the dataset are short, choosing a block size of 16 can save a lot of storage space. However, if the dataset includes longer words, choosing a block size of 32 can save some storage space. This is because the last two bytes in each block are reserved for padding. This means that with a block size of 32, 30 bytes are reserved for the actual word, compared to 14 bytes for a block size of 16. A 30-byte word, when encoded with a block size of 32, requires 1 block of 32 bytes, while the same word encoded with a block size of 16 would require 3 blocks of 16 bytes.

In comparison with the work referenced in [4], we provide a detailed comparison of the number of operations required, as presented in table 3.

***Table 3.** Comparison of computational cost.*

Scheme	*Keyword update*		*Search*	*Data update*	*Client storage*
	Server	*Client*			
NDSS [4]	$O(N)$	$O(1)$	$O(a_{max})$	$O(1)$	$O(W\log F)$
Ours	$O(N)$	$O(1)$	$O(a_w)$	$O(\log N)$	$O(1)$

N denotes the total number of keyword-and-file-identifier pairs, W denotes the number of distinct keywords, F denotes the number of files. For keyword w, a_{max} is the padding constant used for hiding the real search result size, a_w is the total number of data update queries the client has issued.

5. CONCLUSIONS

Searchable encryption presents a promising solution for balancing data security and accessibility in cloud computing environments. By allowing keyword-based search over encrypted data without requiring decryption, SE significantly enhances data privacy while reducing the risk of leakage. Our proposed approach, which integrates searchable encryption with a novel digital signature mechanism, demonstrates both practical performance and strong resistance to common attacks such as query analysis.

Despite these advantages, several challenges remain regarding efficiency, scalability, and robustness against advanced threats, particularly in real-world distributed systems. Looking forward, the integration of searchable encryption with advanced techniques such as secure machine learning, blockchain, or homomorphic encryption holds great potential for building more secure

and efficient data systems. Notably, our current implementation focuses on English keyword search, and expanding support to Vietnamese remains an important direction for future work.

REFERENCES

[1]. Li, F. et al. *"A Survey on Searchable Symmetric Encryption"*, ACM Computing Surveys, 56(5), Article 119, (2023). https://dl.acm.org/doi/10.1145/3617991

[2]. Yan, L., Wang, G., Yin, T., Liu, P., Feng, H., Zhang, W., Hu, H., & Pan, F. *"Attribute-Based Searchable Encryption: A Survey"*, Electronics, 13(9), 1621, (2024). https://doi.org/10.3390/electronics13091621

[3]. Kishiyama, B., & Alsmadi, I. *"A Review on Searchable Encryption Functionality and the Evaluation of Homomorphic Encryption"*, International Journal of Science, Technology and Society, 12(2), (2024).

[4]. Chen, T et al. *"The Power of Bamboo: On the Post-Compromise Security for Searchable Symmetric Encryption"*, NDSS, (2023). https://doi.org/10.14722/ndss.2023.24725

[5]. Shi, D et al.. *"SPKSE: Secure Public Key Searchable Encryption Withstand Keyword Guessing Attacks"*, Scientific Reports, 15, 19658, (2025). https://doi.org/10.1038/s41598-025-01454-9

[6]. Cai, J., Zhao, X., Li, D., Li, H., & Fan, K. *"Efficient and Expressive Public Key Authenticated Encryption with Keyword Search in Multi-user Scenarios"*, Cryptography and Security (cs.CR), (2025). https://doi.org/10.48550/arXiv.2503.16828

[7]. Xu, Y., Cheng, H., Li, J., & Liu, X. *"Lightweight Multi-User Public-Key Authenticated Encryption With Keyword Search"*, IEEE Transactions on Information Forensics and Security, (2025). https://doi.org/10.1109/TIFS.2025.3550054

[8]. Zhang, K., Hu, B., Ning, J., Gong, J., & Qian, H. *"Pattern Hiding and Authorized Searchable Encryption for Data Sharing in Cloud Storage"*, IEEE Transactions on Knowledge and Data Engineering, 37, 2802–2815, (2025). https://doi.org/10.1109/TKDE.2025.3537613

[9]. Xu, S., Cao, Y., Chen, X., Guo, Y., Yang, Y., Guo, F., & Yiu, S.-M. *"Lattice-based Public Key Encryption with Authorized Keyword Search: Construction, Implementation, and Applications"*, Cryptology ePrint Archive, (2023).

[10]. Zhu, T., Wang, J., Xiao, Y., Gao, Y., Zhou, Y., & Weng, J. *"Fully-incremental Public Key Encryption with Adjustable Timed-release Keyword Search"*, Information Sciences, 702, 121887, (2025).

[11]. Ngoc, D. T. B., & Truong, N. D. *"Developing a digital signature scheme avoids attacks based on the order of the generator element"*, Journal of Military Science and Technology, 101, 131–139, (2025).

[12]. *"The Sonnets"*, (n.d.). https://ocw.mit.edu/ans7870/6/6.006/s08/lecturenotes/files/t8.shakespeare.txt

TÓM TẮT

Về một phương pháp tìm kiếm an toàn trong lưu trữ dữ liệu đám mây

Lưu trữ dữ liệu trên nền tảng điện toán đám mây mang lại nhiều lợi ích về hiệu năng và khả năng mở rộng, tuy nhiên cũng đặt ra những thách thức nghiêm trọng về bảo mật và quyền riêng tư. Một trong những vấn đề quan trọng là làm sao cho phép người dùng có thể tìm kiếm dữ liệu đã được mã hóa mà không tiết lộ nội dung cho nhà cung cấp dịch vụ đám mây. Trong bài báo này, chúng tôi đề xuất một phương pháp tìm kiếm an toàn dựa trên mã hóa có thể tìm kiếm kết hợp với chữ ký số mới, cho phép người dùng thực hiện truy vấn từ khóa mà không cần giải mã dữ liệu. Phương pháp được triển khai thử nghiệm trên môi trường đám mây với kết quả cho thấy hiệu năng chấp nhận được và độ an toàn cao đối với các tấn công phổ biến như phân tích truy vấn.

Từ khóa: Cloud storage; Searchable encryption; Secure search; Digital signature; Encrypted data search.