Privacy Preserving on Searchable Encrypted Data in Cloud

Deepak Sharma and Avtar Singh

Abstract Cloud computing is the latest paradigm which offers different cloud services by rearranging of resources and provides them on the bases of user's demand. Users are using cheaper data storage and computation offered in the cloud environment, and they are also facing many problems about reliability, privacy-preserving, and optimized searching of their outsourced data. Proposed scheme "Privacy-preserving keyword search" allows users to encrypt their stored data while preserving some search capabilities and also seek few efforts to consider the reliability of the searchable encrypted data outsourced to the clouds. Client-side encryption and server-side searching of encrypted data without decryption in server-side enable more efficient and eliminate privacy issues between client and server.

Keywords Privacy-preserving keyword search · Optimized searching · Secret sharing · Reliability · Cloud computing

1 Introduction

Cloud computing is the latest paradigm which plays a significant role in the IT industry and the government sector. It provides various computing services over the Internet on the basis of pay-per-use, e.g., storage, server, services, etc. Cloud computing helps users to access database resources via the Internet from anywhere. Confidentiality of data can be maintained by storing the encrypted form of data. Cloud data security and efficient searching are becoming more important for the future development of cloud computing technology in government, industry, and business sectors. The cloud storage [1] offers plenty of new opportunities to cloud consumers. Cloud consumer enjoys powerful computation and storage capabilities

D. Sharma (✉) · A. Singh
National Institute of Technology Jalandhar, Jalandhar 144011, India
e-mail: deepaksharma.51000@gmail.com

A. Singh
e-mail: avtarz@gmail.com

G. S. Hura et al. (eds.), *Advances in Communication and Computational Technology*, Lecture Notes in Electrical Engineering 668,
https://doi.org/10.1007/978-981-15-5341-7_64

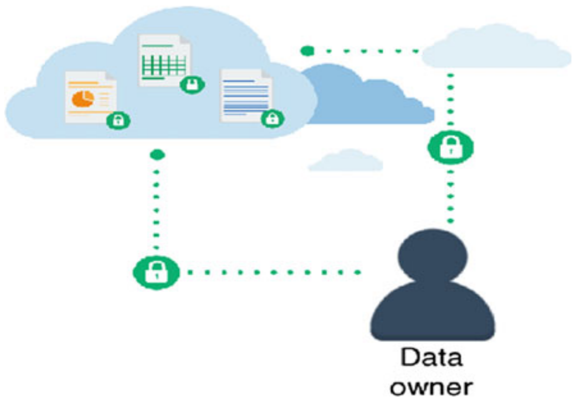
offered by clouds and consumers also suffer from more complicated issues related to reliability, privacy-preserving and complex analysis problems of their cloud data, so it is difficult to obtain full reliability guarantee on the CSP.

In cloud storage, confidentiality of data can be maintained by storing the encrypted form of the data in the cloud. Encryption on data assures security but imposes issues in case of performing any operations on such data. These security issues lead to direct exposure of sensitive data to the adversary. Each time when it is necessary to apply any operations on cloud data, then decryption of encrypted data is required. Hence, it is required to apply operations directly on to encrypted data. But it is not practically operational on encrypted data [2]. By applying operations to partially encrypted data, make things practically operational.

For making accurate and optimized searching, there are comparison and partition base searching and sorting techniques. Efficiency is one of the most important challenges in the cloud domain. Storage as a service efficiency is an important factor occur when to encrypt/decrypt the files and searches that files in the server side. Similarly, the privacy of data also plays a vital role in cloud computing. Server-side operations are most vulnerable to risk. There are so many trust issues between cloud service providers and the client in the cloud. Executing most of the operations in client-side helps to make data more secure and also avoid trust issues between the server and the client. Client-side encryption and decryption provide transfer security and data storage security in server. But clients are not rich with resources and not faster enough to execute large data at a faster rate. It is very important to do things efficiently on client-side. Searching time depends on how the data is stored in server-side and encryption/decryption time depends on how fast the cryptographic algorithm. The suggestion is to use searchable encryption to execute a keyword-based search directly on encrypted data to reduce the costly operation in cloud computing and efficient searching algorithm for optimized performance and QoS.

As shown in Fig. 1, data owner stores their encrypted data to Cloud to achieve data protection but it is not a cost-efficient approach. Above approach has the main purpose to achieve efficient and time-saving encryption/decryption for privacy protection of

Fig. 1 Cloud architecture



sensitive data from a third party (CSPs) in the user side. Achieve efficient retrieval of documents containing the queried keyword in increasing network traffic with the help of index which is constructed and encrypted to search directly on encrypted formats and learn nothing from the encrypted index and encrypted documents.

The overall paper is organized as follows: Sect. 2 describes the recent study which shows what is already done in privacy protection and searching of data and how the motivations are arising about security, privacy protection, and performance efficiency. Section 3 shows the modified AES algorithm. Section 4 shows some approaches which are used to solve the research objective, and Sect. 5 shows the implementation and result analysis.

2 Recent Study

The cloud infrastructure is owned by single CSP or multiple-CSPs and provides different on-demand services like resource allocation, memory, storage, computation, etc. In cloud computing, availability of data gives rise to the vulnerabilities and unauthorized breaches to use sensitive data in cloud storage. Data security and privacy protection are one of the main factors of the user's concerns about cloud technology. There are different approaches that make use of encryption techniques to achieve privacy in the cloud. The encryption system not only preserves the privacy of the file but also provides keyword search over encrypted data and high performance. Initially how to do keyword search over encrypted file effectively was arise in Song et al. [3]. Since then there are lots of work conducted in this field. In 2004, Bonech et al. [4] proposed public-key encryption with keyword searching (PEKS), which enables a way of checking the keywords in an email without learning any information about the email and keywords.

In 2009, Liu et al. [5] allow users to efficiently access files containing certain keywords in a cloud anytime and propose a scheme efficient privacy-preserving keyword searching (EPPKS) in the cloud which improves the performance of previous one. But in 2012, Liu et al. [6] propose a scheme of secure privacy-preserving keyword searching (SPKS) which allows the CSP to participate in the decipherment, so users pay less computation overhead for decryption. But that arises the security concern related to exposé of sensitive data to un-trusted CSPs.

Wang et al. [7] provide a solution for supporting efficient ranked-based keyword search for achieving efficient utilization of cloud-stored encrypted data. The proposed scheme provides a security guarantee and efficiency of the solution. Sajid et al. [8] propose a scheme where fast, efficient ranked based searching over data without decryption overhead, which minimize the searching time. But they are not talking about minimizing the encryption and decryption time at the user side. In cloud environment, most users are not capable to encrypt/decrypt the large size data in faster speed which is the biggest problem in those scenarios. To solving that problem, proposed scheme should need less computation power and less time overhead to make thing simple and compatible.

Related work on searchable encryption focus on Boolean keyword search and single keyword search, and rarely sort the result. Cao et al. [9] propose a solution which solves the problem of privacy-preserving multi-keyword ranked search over encrypted data (MRSE).

Efficient similarity searching over encrypted data scheme shows how they create trapdoor to search user query over a highly secure data index list. They will use locality-sensitive hashing (LSH) for fast nearest neighbor search in high-dimensional spaces. LSH hashes input items so that similar items map to the same bucket metric with high probability.

3 Modified AES Algorithm

First of all, the privacy of data is archived by some encryption algorithms which provide confidentiality and consume lesser time. AES is a cryptographic algorithm which provides high security in lesser time. But to seduce time furthermore, our security parameter is also going downward. By stabilizing the security but continues reduction in time gives efficiency in cloud computing when try to encrypt and decrypt data in the user side. So, the proposed algorithm makes existing AES more secure and less time-consuming. Figure 2 shows the modified AES encryption, and Fig. 2 shows the modified AES decryption. These figures show different modifications made inside the basic AES algorithms. Modifications inside AES algorithm are briefly decried.

3.1 Cyclic Encryption

Cyclic encryption performs different operations over data to provide diffusion and confusion, and S-box is fundamental component used in symmetric key cryptography. Generally, it takes m bits as input and generates n bits as output. This is sometimes often called $m * n$ S-box. Key adding is also a main component in diffusion. Here, key is XOR with data linearly and all the process of cyclic encryption is given in this algorithm. In cyclic encryption, three functions are evoked in sequence: S-box, cyclic key addition, and linear XORing. S-box is the common permutation function using S-box/inverse S-box table [10]. Linear XOR is used to mix the data value and form new result. In XOR, firstly initialize the initialization vector IV. This vector IV is XOR with first byte of data and the outcome is then XOR with next byte and procedure stays proceeded as far as possible.

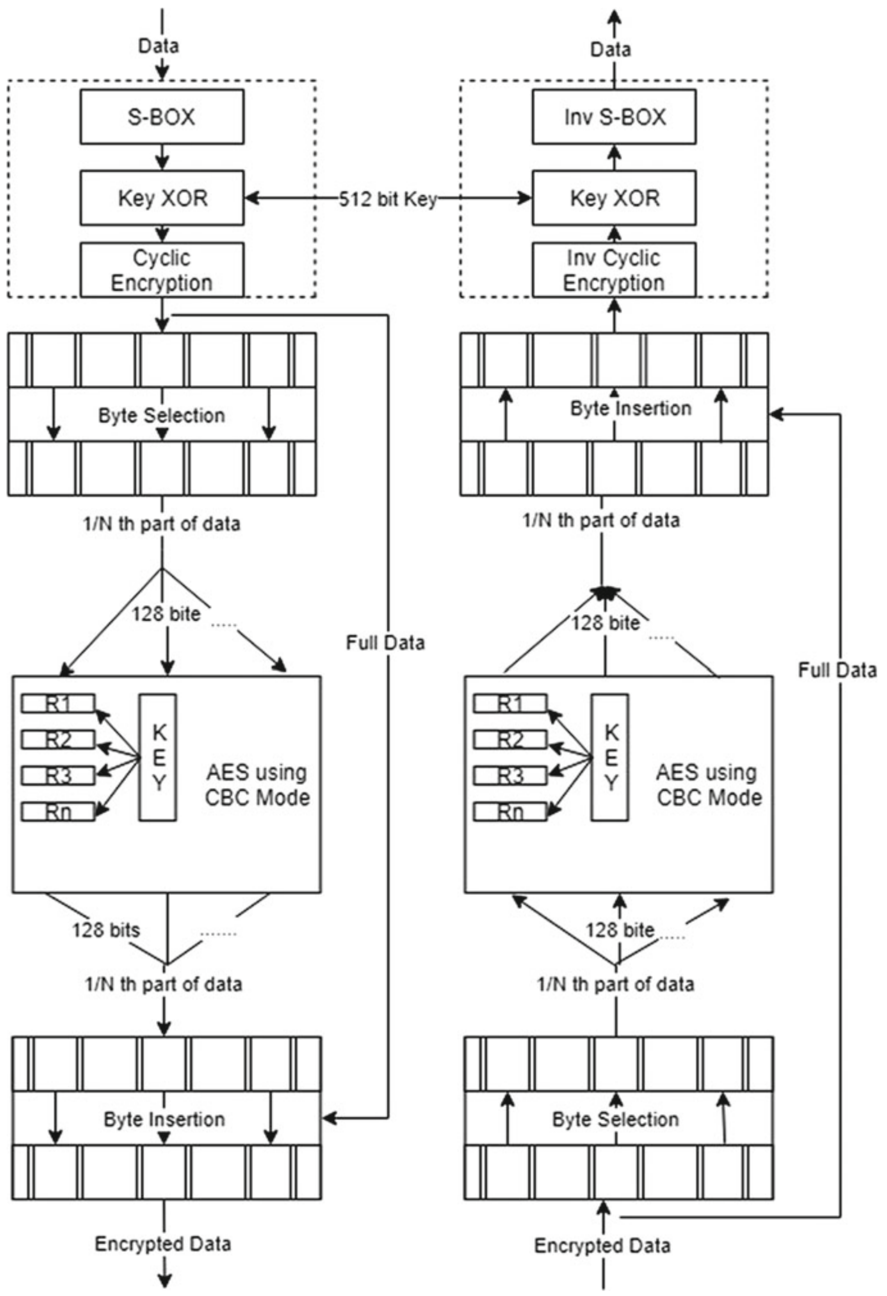


Fig. 2 Encryption process and decryption process

```
Algorithm: Cyclic Encryption

Require: Key← secret key 256 bits D: Data items
Cyclic Encryption:
1. S-box of data
2. Cyclic Key adding
3. Linear XORing
Cyclic Key adding
1. for all $Dt_i$–>D do
2.     $Dt_i \leftarrow Dt_i \oplus key_j$
3. end for
Linear XOR
1. for all $Dt_i$–> D do
2.     $IV \leftarrow Dt_i \oplus IV$
3.     $Dt_i \leftarrow IV$
4. end for
```

3.2 Inverse Cyclic Encryption

Inverse cyclic encryption performs some inverse operation of cyclic encryption and same operations. Inverse S-box is fundamental component used in symmetric key cryptography. In key adding, key is XOR with data linearly.

```
Algorithm: Inverse Cyclic Encryption

Require: Key← secret key 256 bits D: Data items
Inverse Cyclic Encryption:
1. Inverse Linear XORing
2. Cyclic Key adding
3. Inverse S-box of data
Cyclic Key adding
1.     for all $Dt_i$–>D do
2.         $Dt_i \leftarrow Dt_i \oplus key_j$
3.     end for
Inverse Linear XOR
1. for all $Dt_i$–> D do
2.     $Temporary_i \leftarrow Dt_i \oplus IV$
3.     $IV \leftarrow Dt_i$
4.     $Dt_i \leftarrow Temporary_i$
5. end for
```

In inverse linear XOR, initialize vector IV is XOR with first byte of data and the outcome is stored. But previous dt_i XOR with next byte and procedure stays proceeded as far as possible.

3.3 Byte Abstraction

It is not possible to predict the user data size. It may vary from small to large. To encrypt a large file encryption, algorithm takes a larger amount of time. Encryption algorithm is not applied over the whole data. So only 1/*N*th part is encrypted that is discussed further.

3.4 AES Implementation

It is a symmetric encryption calculation and utilizations substitution and stage approach. Key sizes of 256 bits AES encryption take 14 rounds separately. All rounds preparing is same with the exception of the last round.

3.5 Byte Embedding

In this Nth part that is encrypted is placed on its original place of data. Modified AES algorithm is useful in achieving the proposed scheme goals. This provides optimized encryption/decryption and maintains the security level as well. As shown in Figs. 2, extra function is added in front of encryption of AES and backside of decryption of AES. These two addition functions add extra security factor to algorithm but increase time consumption also but after applying N factor, that addition functions provide nearly same security with lesser time consumption. That N factor is described in Sect. 3.

4 Approach

To provide the privacy-preserving and optimized searching, use reliable cloud storage and privacy-preserving keyword search scheme in a cloud environment. Use of searchable encryption is to execute keyword-based search directly on encrypted data and to reduce the costly operation in cloud computing. Generation of an index which embedded the file features and keywords is used for keyword search. The server performs the searching of keywords on the index without learning anything about the data.

4.1 Data Encryption/Decryption

Data is unstructured, and the service provider does not know about the properties of data. Service provider gets the data in encrypted form. The client provides the data items D and encrypts the data items say D by using key to form the encrypted collection. From encryption, AES 512-bit keys are used for storing data in a cloud server. Generally, AES has various key sizes (128, 192, and 256) for encryption. But 512-bit key size AES is more secure and reliable as compared to rest. Similarly, AES-512 has mostly 22 numbers of rounds for encryption, which is mapped with key size.

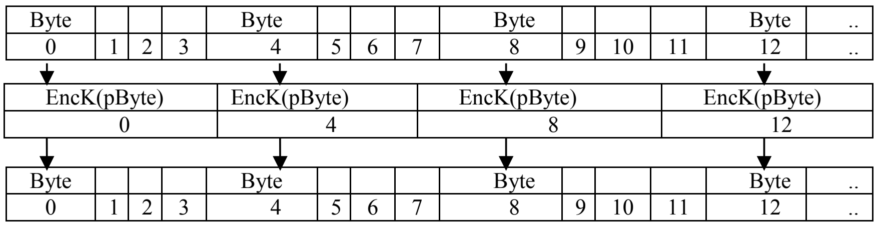


Fig. 3 An example for encryption of files $N = 4$

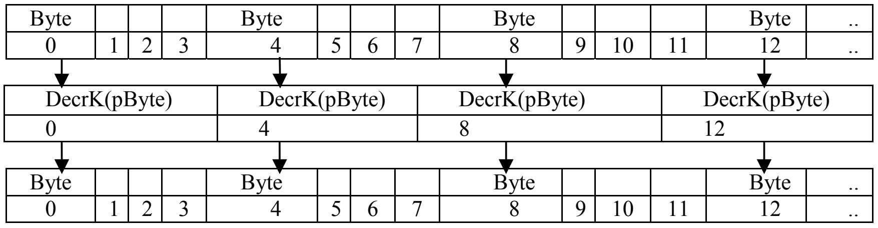


Fig. 4 An example for decryption of files $N = 4$

User data sizes are not predictable, which are either larger or small. If AES-512 encryption is applied over large size data, it will take a lot of time to implement. So, encrypting whole data is not a good choice. Encryption applies over one/Nth part of a file only and embeds that one/Nth part into that particular file. Using that way hiding of sensitive data from unauthorized people and secure the file from tempering.

For data encryption files are in a byte array format and took every Nth byte (mean index multiple of N) from the byte array and encrypt that streamed byte array. This process is part of byte abstraction in Fig. 2 and in byte embedding encrypted bytes array again rearranges to their actual position from where they are extracted. Like the above example, Fig. 3 shows how bytes are abstracted, which are multiple of N and encrypt them sequentially and place encrypted byte again to their original position using byte embedding. After that whole encrypted files are stored along with their encrypted index. Similarly, decryption of file is carried out when server retrieves the file which is searched using appropriate query keyword. In decryption of file, firstly byte abstraction and decrypt them sequentially and place encrypted byte again to their original position using byte embedding shown in Fig. 4.

4.2 Feature Extraction

In feature extraction, frequent keywords are extracted from file data item $D_{\text{item}}(D_i, \ldots, D_n)$ which contains their features $f(f_i,\ldots,f_n)$ (keywords or sensitive data). Feature extraction maps D_i to f_i. For feature extraction, there is a different algorithm which is optimized and reliable for extraction.

4.3 Index Construction

Construction of hash function $H(h_1, h_n)$ is according to the requirement of our application. Mapping each feature f_i to hash function g_i. Like $g_i(f_i)$ be the output by applying hash function g_i over feature f_i. Then, $g_i(f_i)$ is one bucket identifier of the data item index and the data items that contain characteristic are the member of it. Content is simply a bit matrix of size L where L is the total number of data file stored on the server. Each data item has assigned an identifier from 1 to L. As always storing of data index in matrix form is not space-efficient but it required lesser time to search the index in matrix. So, storing data index in matrix format is efficient and simple. Hashing and matrix construction are given in build index algorithm.

4.4 Index Encryption

After index construction, secure index by applying encryption on bit matrix identifiers and contents. Only users should able to encrypt that bit identifiers during the query process. Data owner will not share that secret key with other entrusted person. Otherwise, the adversary could apply that encryption and find out any index of file easily. Similarly, the whole bit matrix of indexs is encrypted to hide the leakage of important information.

There is the algorithm which is used to build index using secret keys (Key_{id}, $\text{Key}_{\text{paylod}}$), and $\text{EncKey}_{\text{id}}$, $\text{EncKey}_{\text{payload}}$ be an encryption technique like AES-512 in CTC mode. D_{item} denotes data items, Max: maximum possible number of features, F_i denotes extracted feature of data items, $h_i(f_{ij})$ denotes hashing implemented over feature f_i, Mk denotes matrix identifier, and V_{Mk} is the bit of matrix. After encryption, fake records are added into the index to hide the actual number of features.

Algorithm: Buid Index

```
    Require: $D_{item}$: data items, h: hash functions, Max: maximum possible number of features
1.  $Key_{id}$ ← Key generation, $Key_{payload}$ ← Key generation
2.  for all $D_i$ –>$D_{item}$ do
3.      $F_i$ ← extract characteristic of Di
4.          for all $f_{ij}$ -> $F_i$ do
5.              $F_{ij}$ ← apply hash $_{function}$ on $f_{ij}$
6.              if h($f_{ij}$ ) != bucket identifier list then
7.                  add $h_i(f_{ij})$ to the bucket identifier list
8.              end if
9.              $Vh(f_{ij})[id(D_i)]$ ← 1
10.         end for
11.  end for
12.  for all $M_k$ -> bucket identifier list do
13.      $V_{Mk}$ ← retrieve payload of $M_k$
         $\pi M_k$ ← $EncKey_{id}$ ($M_k$), $\sigma V_{Mk}$ ← $EncKey_{payload}$ ($V_{Mk}$ )
14.      add ($\pi_{Mk}$, $\sigma V_{Mk}$) to I
15. end for
16. return I
```

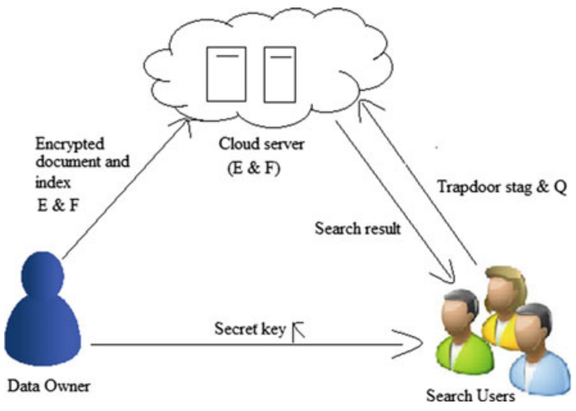


Fig. 5 Condition trapdoor security for partial encrypted patterns

4.5 Trapdoor Construction

Initially it applies a hash function over feature f_i to construct the plain query ($h_i(f_i)$), and then applies Enckey$_{id}$($h_i(f_i)$) on each component of the plain query. Once the trapdoor T_f is constructed, it is sent to the server. Figure 5 shows that data owner stores encrypted data and index in the cloud server, and a different user can search their query and retrieve result using the secret key, which is shared by the data owner.

4.6 Keyword Search

As shown in the algorithm search keyword, where client generates the trapdoor using LSH algorithm and sends query trapdoor to server. Server gets the trapdoor T_f from user and performs an operation for each "Query Trapdoor" on the index, and sends back the result which contains the encrypted bit vector.

Algorithm: Search Keyword

```
    CLIENT:
   Require: h: hash functions, Max: maximum possible number of features
1.     Key$_{id}$ ← secret key, f: characteristics to search
2.     for all $f_i$ –>f do
3.        $F_{di}$ ← EncKey$_{id}$(h($f_f$))
4.     end for
5.     Trapdoor $T_f$←($f_1$,……,$f_{dn}$)
6.     Send $T_f$ to Server
   SERVER
Require: I: index, $T_f$: Trapdoor
7.      for all $f_{di}$ -> $T_f$ do
8.           If ($f_{di}$,$V_{Mk}$)->I then
9.         Send $V_{Mk}$ to Client
10.     end if
11.    end for
```

Table 1 Tools and platform used

IDE	Net beans
Programming language	Java 1.8
Operating system	Windows 10 Pro 64-bit
Analytical tool	SigmaPlot
Processor	Intel Core I7 3rd Gen, 2.4 Ghz
RAM	4.00 GB, 1600 MHz
Graphics	1.00 GB, Nvidia Geforce 650 M

Both the owners' cloud data and data index contain important information and required protection against adversary. So, system should fulfill this security requirement:

- **Data privacy**: Secret key is required to breach the security of cloud data. Including cloud server, no one can able to learn any about private data and index data.
- **Search privacy**: Most important requirement of users is to protect the searching criteria they are using on data. These searchable criteria are not reducible from trapdoor query and data/index sent by user.

5 Implementation and Result Analysis

Proposed scheme takes place in two stages. In the first stage, storage of data in cloud takes place by encryption and indexing of files. Then, encrypted file along with encrypted index (which is constructed by feature extraction from data items Di) is sent to cloud. In the second stage, retrieval of files is done by keyword searching (take place in cloud side).

5.1 Simulation Settings

See Table 1.

5.2 Throughput Analysis

Performance is evaluated over different sizes of data with different key sizes. Simulation result shows encryption/decryption time against file size and encryption parameter N.

Different sizes of data demonstration of results in graphical form are shown in Figs. 6 and 7. A graph of encryption and decryption with respect to changing value of $N = \{4, 8, 16, \text{and } 32, \ldots, n\}$. Figures analyze that if larger size N is taken then

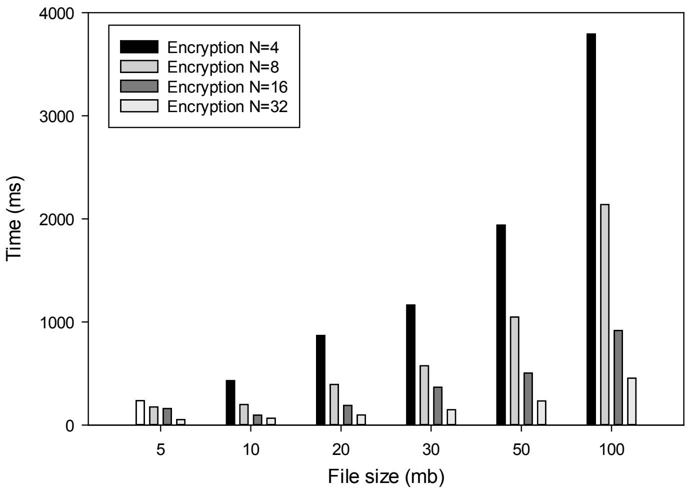


Fig. 6 Encryption ($Data_{byte}/Time_{millisec}$)

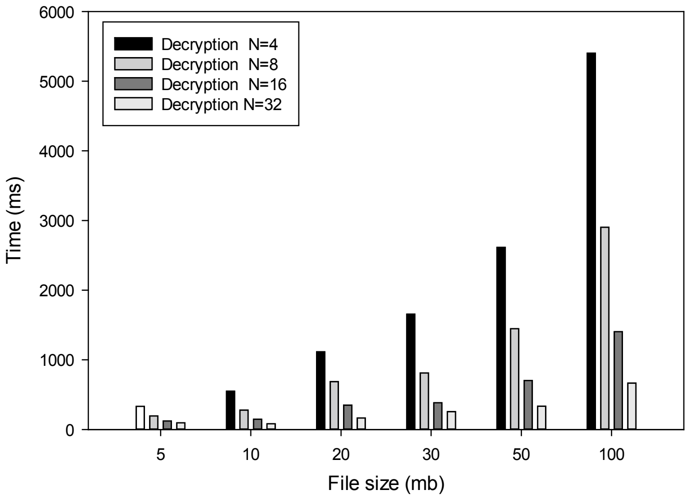


Fig. 7 Decryption ($Data_{byte}/Time_{millisec}$)

encryption time for $N = \{4, 8, 16, 32\}$ are decreasing (N value used in selection of byte for encryption/decryption).

Figure 8 shows the throughput (Datab/Timemillisec) and shows how much data in byte form is encrypted or decrypted in one millisecond. This also shows as the increase in $N = \{4, 8, 16, 32, \ldots, n\}$ the throughput of encryption increases.

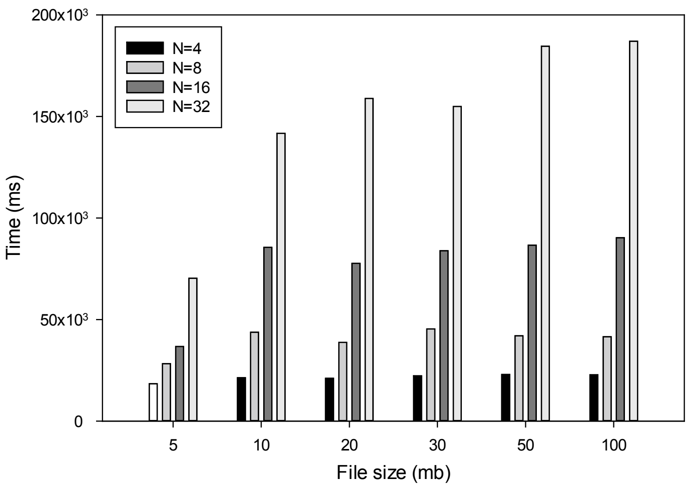


Fig. 8 Throughput ($Data_{byte}/Time_{millisec}$)

5.3 *Storage and Retrieval of Data from Cloud*

Experiment 1–10 mb of data is taken for comparison of data upload and download time of proposed scheme and client-side deduplication scheme (CSDS) [11] with respect to average time (ms). Figure 9 shows that the proposed scheme consumes less time in both upload and download of files.

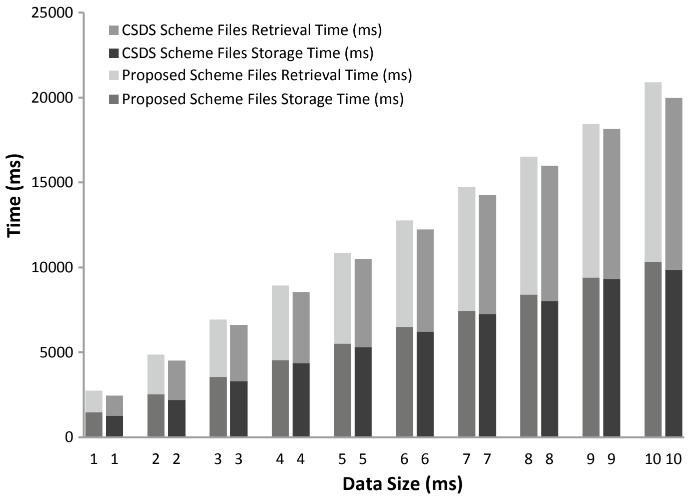


Fig. 9 Files storage time and retrieval time (Data mb/Time ms)

Figure 10 shows encryption time and index construction time with respect to *N* value. *N* value is the value which shows amount of data is encrypted. Figure 10 shows decryption time and searching time with respect to *N* value. As shown in Figs. 10 and 11, encryption times are more as compared to decryption time. Index construction time and searching time are linear as compared to encryption/decryption time. Here, encryption and decryption take place in user's side which are not capable

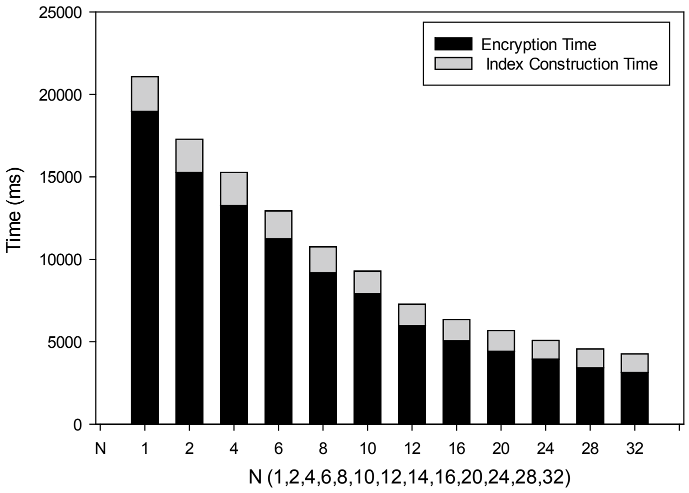


Fig. 10 Storage of file

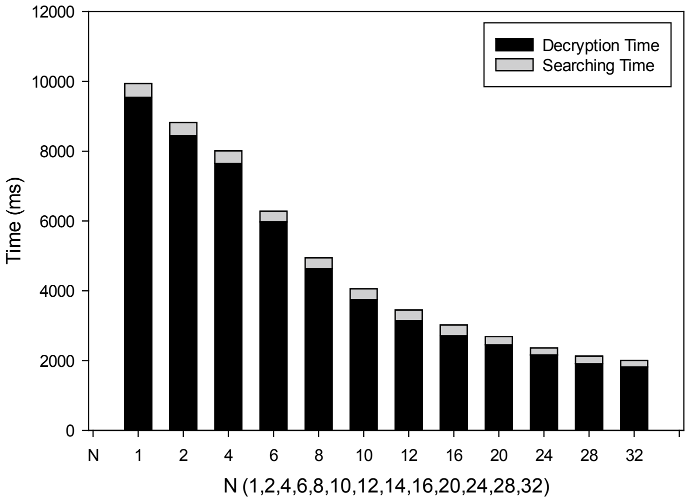


Fig. 11 Retrieval of file

Table 2 Algorithms with avalanche effect (%)

N	Avalanche effect of basic AES (%)	Avalanche effect of propose AES (%)	Proposed AES avalanche effect (%) rate of change
1	46.88	56.01000	0
2	24.431	51.99631	7.1433
4	12.190	46.99631	9.6160
6	6.700	41.03711	12.680
8	7.259	37.93308	7.5639
10	7.994	35.06430	7.5628
12	7.590	31.96428	8.8409
16	7.035	30.96428	3.1284
20	6.812	30.56428	1.2918
24	7.37	30.26428	0.9815
28	5.69	30.06428	0.6608
32	7.705	30.00010	0.2136

of encrypting/decrypting large size data and also have low hardware configurations. Parameter n trying to make possible to execute large data with low hardware configurations.

Avalanche effect shows the property of security which is crucial for block cipher. Avalanche effect measures the change of one bit in plaintext and leads to how much change in ciphertext. That is totally depending on amount of confusion and diffusion in cryptographic algorithm.

Original AES [12] has lesser avalanche effect as compared to the proposed avalanche effect when value of *N* is 1. As decreasing of *N* value propose AES have high avalanche effect then original AES (shown in Table 2).

Figure 11 shows percentage value of avalanche effect with respect to *N* value and Fig. 12 is the graph which shows us the change of values of Figs. 9 and 11 with respect to *N*. As shown in Fig. 12, avalanche effect is not changed as much as compared to encryption time. Purpose of the proposed scheme is to lower the encryption/decryption overhead on user side but not compromising the security. Above graph shows that purpose of the proposed scheme is fulfilled in some extend (Fig. 13).

6 Conclusion

Cloud storage as a service is one of the main services provided by cloud. Proposed scheme maintains the confidentiality of sensitive data and minimizes the time consumption in encryption/decryption and enables direct searching over encrypted data without compromising much privacy guarantee of sensitive data. Matrix-based index construction and comparison-based searching obtain better efficiency

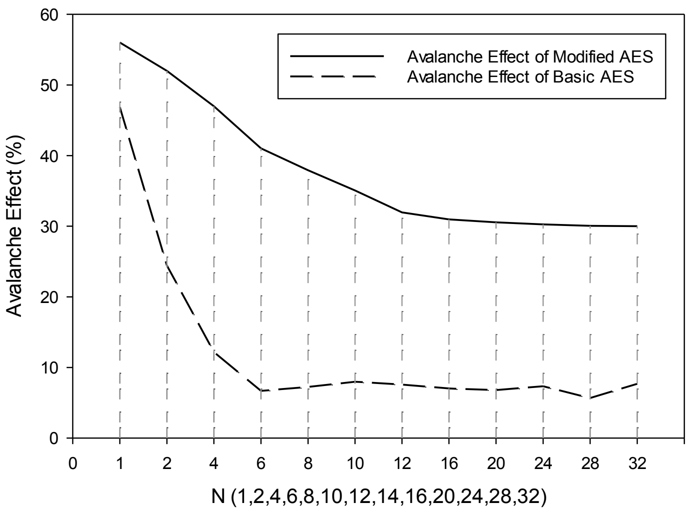


Fig. 12 Avalanche effect of data [(%)/N (1, 2, 4, 6, 8, 10, 12, 16, 20, 24, 28, 32)]

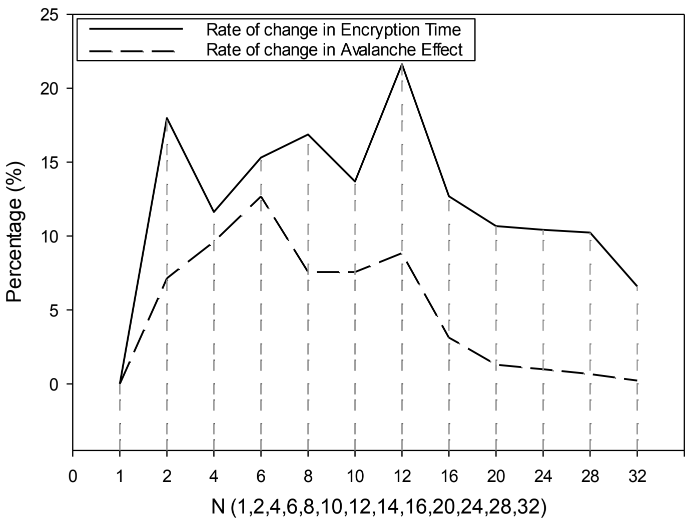


Fig. 13 Rate of change in encryption time with respect to change in avalanche effect

in searching. There are still many challenges in the proposed scheme like there is future work on dynamic deletion/insertion of files and large-sized data. There are many security challenges in a multi-user scheme. Firstly, all the users share secret key for indexing and trapdoor generation. In this scenario, user revocation is the big challenge. In the future work, improvement is required in the proposed scheme to handle these challenges.

References

1. Li J, Lin D, Squicciarini AC, Li J, Jia C (2017) Towards privacy-preserving storage and retrieval. IEEE Trans Cloud Comput 5(3):499–509
2. Chatterjee A, Sengupta I (2015) Searching and sorting of fully homomorphic encrypted data on cloud. In: IACR cryptology ePrint archive. IACR, p 981
3. Song DX, Wagner D, Perrig A (2000) Practical techniques for searches on encrypted data. In: IEEE symposium on security and privacy. S&P. IEEE, pp 44–55
4. Boneh D, Di Crescenzo G, Ostrovsky R, Persiano G (2004) Public key encryption with keyword search. In: International conference on the theory and applications of cryptographic techniques. Springer, pp 506–522
5. Liu Q, Wang G, Wu J (2009) An efficient privacy preserving keyword search scheme in cloud computing. In: International conference on IEEE computational science and engineering, 2009. CSE'09. IEEE, pp 715–720
6. Liu Q, Wang G, Wu J (2012) Secure and privacy preserving keyword searching for cloud storage services. J Netw Comput Appl 35(3):927–933
7. Wang C, Cao N, Ren K, Lou W (2012) Enabling secure and efficient ranked keyword search over outsourced cloud data. IEEE Trans Parallel Distrib Syst 23(8):1467–1479
8. Khan MS, Wang C, Kulsoom A, Ullah Z (2013) Searching encrypted data on cloud. Int J Comput Sci Issues IJCSI 10(6):230
9. Cao N, Wang C, Li M, Ren K, Lou W (2013) Privacy-preserving multi-keyword ranked search over encrypted cloud data. IEEE Trans Parallel Distrib Syst 222–233
10. https://en.wikipedia.org/wiki/Rijndael_S-box
11. Kaaniche N, Laurent M (2014) A secure client side deduplication scheme in cloud storage environments. In: 2th international conference on new technologies, mobility and security (NTMS). IEEE, pp 1–7
12. Sriperumbudur Tamil Nadu (2018) A block cipher algorithm to enhance the avalanche effect using dynamic key-dependent S-box and genetic operations. Int J Pure Appl Math 119(10):399–418
13. Kartheeban K, Durai Murugan A (2017) Privacy preserving data storage technique in cloud computing. In: IEEE international conference on intelligent techniques in control, optimization and signal processing (INCOS). IEEE, pp 1–6
14. Al-Mamun A, Rahman S, Ahmed Shaon T, Hossain M (2017) Security analysis of AES and enhancing its security by modifying S-box with an additional byte. Int J Comput Netw Commun IJCNC 9(2)
15. Shankarwar MU, Pawar AV (2015) Security and privacy in cloud computing: a survey. In: Proceedings of the 3rd international conference on frontiers of intelligent computing theory and applications (FICTA). Springer, pp. 1–11
16. Li J, Lin D, Squicciarini A, Jia C (2015) STRE: privacy-preserving storage and retrieval over multiple clouds. In: International conference on security and privacy in communication networks, pp 36–44)
17. Kamara S, Papamanthou C, Roeder T (2012) Dynamic searchable symmetric encryption. In: Proceedings of the 2012 ACM conference on computer and communications security. ACM, pp 965–976
18. Kuzu M, Islam MS, Kantarcioglu M (2012) Efficient similarity search over encrypted data. In: IEEE 28th international conference on data engineering. IEEE, pp 1156–1167
19. Ananthi S, Sadish Sendil M, Karthik S (2011) Privacy preserving keyword search over encrypted cloud data. In: International conference on advances in computing and communications. Springer, pp 480–487
20. Owens D (2010) Securing elasticity in the cloud. Commun ACM 53(6):46–51
21. Chang Y-C, Mitzenmacher M (2005) Privacy preserving keyword searches on remote encrypted data. In: International conference on applied cryptography and network security. Springer, pp 442–455