Block-Level Message Encryption for Secure Large File to Avoid De-duplication

Bhagyashree Bhoyane[1] , Snehal Kalbhor[2] , Sneha Chamle[3], Sandhya Itkapalle[4], Prof. P.M. Gore[5]

[1,2,3,4]*Student, Computer Department, Padmabhushan Vasantdada Patil Institute of Technology, Pune, Maharashtra*
[5] *Professor, Computer Department, Padmabhushan Vasantdada Patil Institute of Technology, Pune, Maharashtra*

***

Abstract: *Cloud computing is a great way of computing as a utility. Besides all the benefits of the cloud computing security of the stored data need to be considered while storing sensitive data in the cloud. Cloud users cannot rely only on cloud service provider for the security of their sensitive data stored in the cloud.*

To achieve optimal usage of storage resources, many cloud storage providers perform de-duplication, which exploits data redundancy and avoids storing duplicated data from multiple users. The system proposes a new approach to achieve more efficient deduplication for (encrypted) large files. Our approach, named Block-Level Message-Locked Encryption (BL-MLE), can achieve file-level and block-level deduplication, block key management, and proof of ownership simultaneously using a small set of metadata. The BL-MLE method can be simply completed to hold confirmation of storage that makes it multi-purpose for secure cloud storage.

***KEYWORDS*: Third Party Authenticator, AES Algorithm, RSA Algorithm, SHA 512 Algorithm, deduplication, Block-Level Message-Locked Encryption.**

1. INTRODUCTION

Uploading large files would consume extensive bandwidth; source-based deduplication seems to be a better choice for large file outsourcing.

The user firstly sends a file identifier to the server for file redundancy checking. If the file to-be-stored is duplicated in the server, the user should convince the server that he/she indeed owns the file. Otherwise, the user uploads the identifiers/tag of all the file blocks to the server for block-level deduplication checking. Finally, the user uploads data blocks which are not stored in the server.

Deduplication [1] [2] is a popular technique widely used to save storage spaces in the cloud. To achieve secure deduplication of encrypted files, a new cryptographic primitive named Message-Locked Encryption (MLE) scheme[2][1][8] can be extended to obtain secure deduplication for large files, it requires a lot of metadata maintained by the end user and the cloud server.

A hybrid cloud approach [3] to ensure security in deduplication which involves private cloud for providing tokens to access encrypted data in the cloud. Data encryption method working here is convergent encryption [5] and PoW [4] is used to make sure ownership eligibility to deduplicate the file.

Homomorphic encryption is used as one of the key managing schemes in [6]. Data encryption key is first computed by the initial file uploader and further distributed consequent verified uploader by the key server. Data encryption key used for encryption are further encrypted with the hash of file content. Data encrypted with data encryption keys are sent to the storage server. HEDup ensures privacy while enabling deduplication. Key server may become a bottleneck when the number of clients increases in case of large-scale deployment, and a decentralized deployment of the key server is supposed as a solution.

Proof-of-Ownership (PoW)[4] [7]is necessary for source-based deduplication. PoW is an interactive protocol between a prover (file owner) and a verifier (data server). By executing the protocol, the prover convinces the verifier that he/she is an owner of a file stored by the verifier. PoW protocol in which presents three schemes that differ in terms of security and performance.

2. PROPOSED SYSTEM

Deduplication is basically a compression method for removing unnecessary data. Fig 1 explains the deduplication process.

1. Proof of Ownership

Data Owner uploads document, metadata, the checksum on a cloud after encryption using keys from Data Owner and Cloud Service Provider. Also, a copy of metadata and checksum is sent to Auditor.

2. Data Access Via Permission model

Registered users send access request and receive encrypted file if authorized. User calculates checksum to compare with original and reports to Data Owner if checksum mismatch occurs.

3. Prevention De-duplication

De-duplication at File Level and Block Level is avoided by Maintaining the checksum of file data and block of file data and compare at the time of file upload to avoid Deduplication. In this, the AES algorithm is used for File Encryption in System, And Sha-1is used for sharing the files. After checking the file level Deduplication if no duplicate found then it will check it for blocks. It divides the file in a number of blocks and performs the deduplication.

4. Third Party Authenticator (TPA)

Auditor Receives metadata after upload. Performs periodic or on-Demand integrity checks by sending challenges to Cloud Service Provider. On response from Cloud Service Provider, Auditor confirms response and reports status to Data Owner.

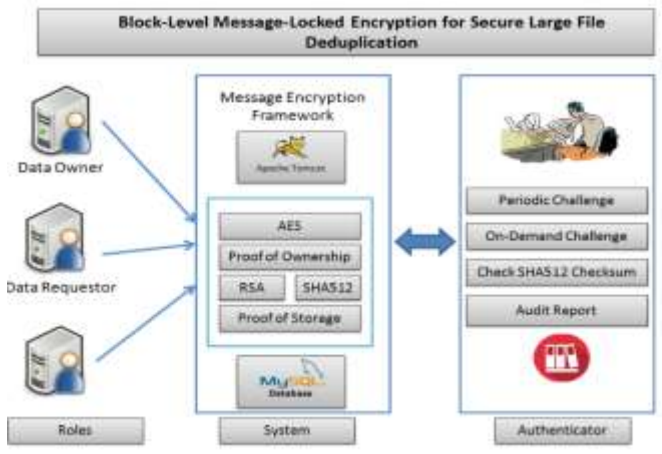

Fig-1: Architecture of Proposed System [1]

3. PROPOSED METHOD

Algorithm Used:

A. AES Algorithm

The substitution-permutation network is a basic design principle of AES and is fast in software and hardware. Contrasting its predecessor DES, AES does not use a Feistel network.

The AES operates on a 4×4 column-major order matrix of bytes, term as the state, while some versions of Rijndael have a larger block size and have additional columns in the state. Most AES calculations are done in a special finite field.

We use 128 bit key for an AES cipher which specifies the number of repetitions should be 10 cycles' transformation rounds that convert the input, called the plaintext, into the final output, called the ciphertext. Each round consists of several processing steps, each containing four similar but different stages, including one that depends on the encryption key itself. A set of reverse rounds are used to transform ciphertext back into the original plaintext using the same encryption key.

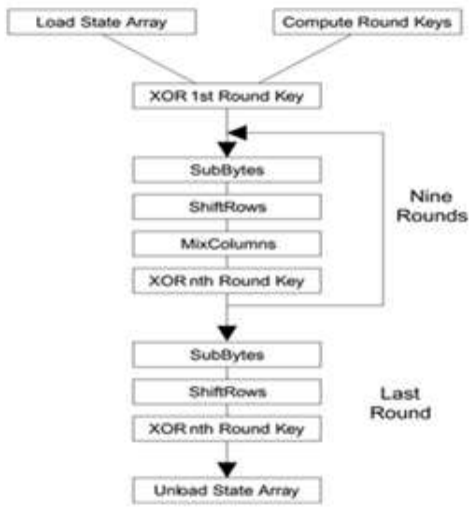

Fig-2: AES Algorithm Flowchart

B. RSA Algorithm

The following steps used to generate the RSA algorithm keys:

1. Choose two different prime numbers p and q.

For security purposes, the integer's p and q should be superior at random, and should be similar in magnitude but 'differ in length by a few digits to make factoring harder. Prime integers can be professionally created using a primality test.

2. Compute n = pq.

- n is used as the modulus for both the public and private keys. Its length often expressed in bits, is the key length.

3. Compute $\varphi(n) = \varphi(p)\varphi(q) = (p - 1)(q - 1) = n - (p + q - 1)$, where φ is Euler's totient function. This value is kept private.

4. Prefer an integer e such that $1 < e < \varphi(n)$ and $\gcd(e, \varphi(n)) = 1$; i.e., e and φ(n) are coprime.

5. Determine d as $d \equiv e-1 \pmod{\varphi(n)}$; i.e., d is the modular multiplicative inverse of e (modulo φ(n))

- This is more clearly stated as: solve for d given $d \cdot e \equiv 1 \pmod{\varphi(n)}$

- e having a small bit-length and small Hamming weight results in more capable encryption – most usually 216 + 1 = 65,537. On the other hand, a large amount of smaller values of e (such as 3) have been shown to be fewer protected in some settings.

- e is released as the public key exponent.
- d is kept as the private key exponent.

The public key consists of the modulus n and the public (or encryption) exponent e. The confidential input consists of the modulus n and the private (or decryption) example d, which must be kept secret. p, q, and φ(n) must also be kept secret because they can be used to calculate d.

C. SHA 512 Algorithm

The United States National Security Agency designed a cryptographic hash function that is SHA-1 (Secure Hash Algorithm 1) and it is a U.S. Federal Information Processing Standard published by the United States NIST. SHA-1 produces a 160-bit (20-byte) hash value known as a message digest. A SHA-1 hash value is usually rendered as a hexadecimal number, 40 digits long.

Image Description: One iteration within the SHA-1 compression function: A, B, C, D, and E are 32-bit words of the state; F is a nonlinear function that changeable; n denotes a left bit iteration by n places; n changed for each operation; Wt is the extended message word of round t; Kt is the round constant of round t; denotes addition modulo 232.

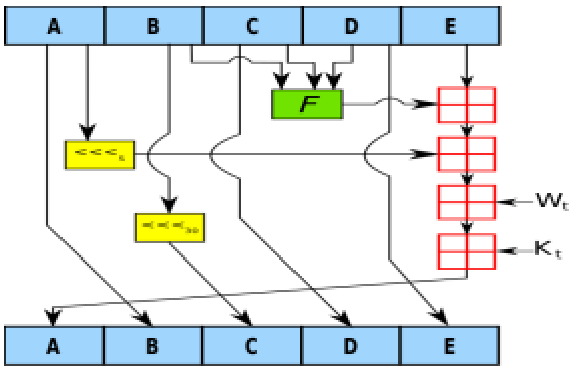

Fig-3: SHA Algorithm

4. EXPERIMENTAL RESULT

To accomplish best possible procedure of storage resources, many cloud storage providers perform de-duplication, which exploits data redundancy and avoids storing duplicated data from multiple users. Data Owner uploads document, metadata, the checksum on a cloud after encryption using keys.

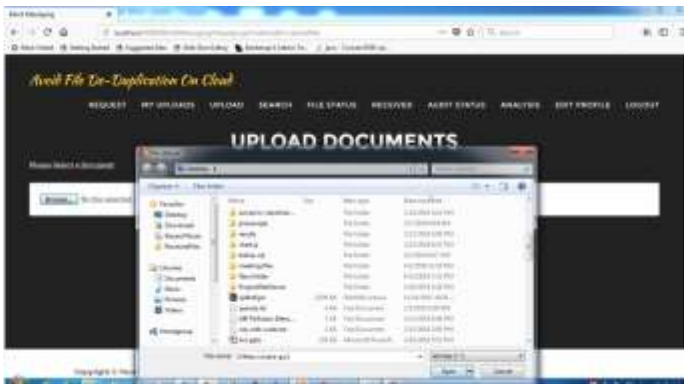

Fig-4: Upload Document

The User has to register first on the system with all the details to get the access and receive an encrypted file. The user profile is shown in the figure below.

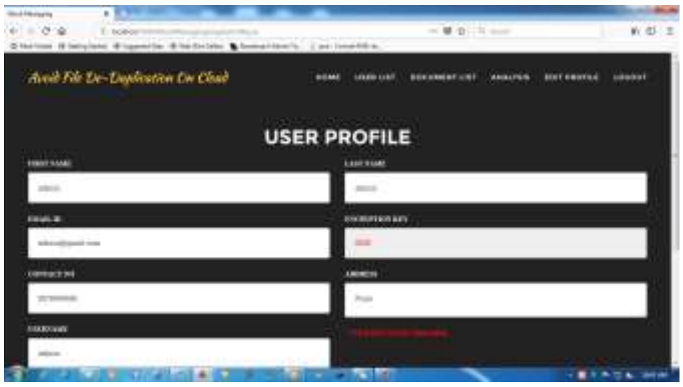

Fig-5: User Profile

Auditor Receives metadata after upload. Performs periodic or on-Demand integrity checks and generate audit reports.

Fig-6: Audit Report

After the entire authentication, the file access is given to the user with the decryption key for accessing the files uploaded by the owner.

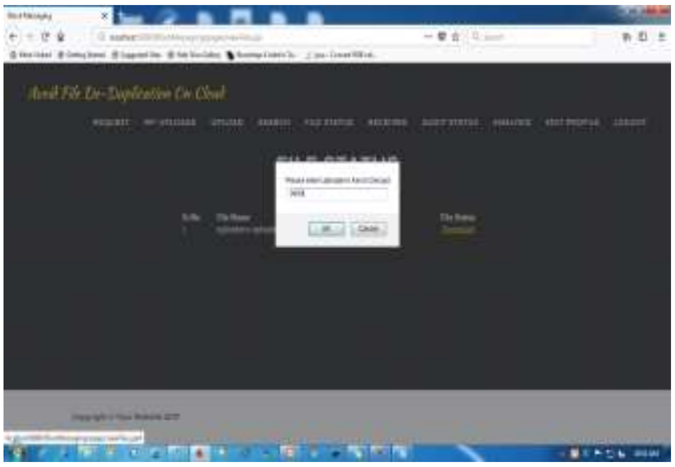

Fig-7: Granted File Access

The graphical analysis of duplicate and unique content is shown in figure 8.

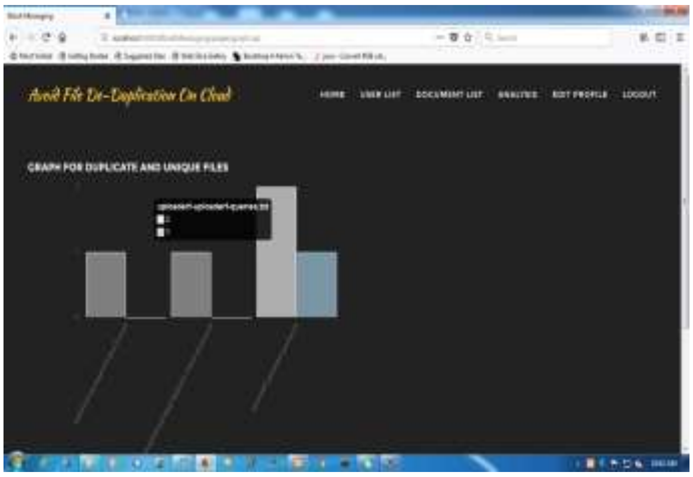
Avoid File De-Duplication On Cloud
HOME
USER LIST
DOCUMENT LIST
ANALYSIS
EDIT PROFILE
LOGOUT
GRAPH FOR DUPLICATE AND UNIQUE FILES

Fig-8: Graph Analysis

5. CONCLUSION

We have developed a system i.e. block-level deduplication which can provide more space savings than file-level deduplication does in large file storage. It is worth noting that for block-level deduplication, the block size can be either fixed or variable. This system exploits data redundancy and avoids storing duplicated data from multiple users System focus on the block-level deduplication with fixed block size.

REFERENCES

[1] Bhagyashree Bhoyane, Snehal Kalbhor, Sneha Chamle, Sandhya Itkapalle, "Block-Level Message-Locked Encryption for Secure Large File De-duplication", International Research Journal of Engineering and Technology(IRJET), Volume: 04 Issue: 12 | Dec-2017.

[2] Mihir Bellare, Sriram Keelveedhi, Thomas Ristenpart "Message-Locked Encryption and Secure Deduplication", preliminary version of this paper appears in the proceedings of Eurocrypt 2013.

[3] Li, Jin, Yan Kit Li, Xiaofeng Chen, Patrick PC Lee, and Wenjing Lou. "A hybrid cloud approach for secure authorized deduplication." Parallel and Distributed Systems, IEEE Transactions on 26, no. 5 (2015): 1206 – 1216.

[4] Yang, Chao, Jianfeng Ma, and Jian Ren. "Provable Ownership of Encrypted Files in De-Duplication Cloud Storage." Ad Hoc & Sensor Wireless Networks 26.1 - 4 (2015): 43 – 72.

[5] J. Dou ceur, A. Adya, W. Bolosky, D. Simon, and M. Theimer. "Reclaiming space from duplicate files in a serverless distributed file system. In Distributed Computing Systems", 2002. Proceedings. 22nd International Conference on pages 617{624. IEEE, 2002.

[6] Miguel, Rodel, and Khin Mi Mi Aung. "HEDup: Secure Deduplication with Homomorphic Encryption." In Networking, Architecture, and Storage (NAS), 2015 IEEE International Conference on, pp. 215 - 223. IEEE, 2015.

[7] Yang, Chao, Jianfeng Ma, and Jian Ren. "Provable Ownership of Encrypted Files in De-Duplication Cloud Storage." Ad Hoc & Sensor Wireless Networks 26.1 - 4 (2015): 43 - 72.

[8] Rongmao Chen, Yi Mu, "BL-MLE: Block-Level Message-Locked Encryption for Secure Large File Deduplication", IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY.