Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.18576/amis/130312

Secure and Efficient Deduplication Scheme based on Ownership Challenge for Mobile Cloud Environment

Sebastian Annie Joice[1,*] *and M. A. Maluk Mohamed*[2]

[1] Department of Computer Science and Engineering, Government College of Engineering,Srirangam,Tiruchirappalli, India
[2] Department of Computer Science and Engineering, M.A.M College of Engineering,Tiruchirappalli, India

Received: 20 Nov. 2018, Revised: 3 Feb. 2019, Accepted: 6 Feb. 2019
Published online: 1 May 2019

Abstract: Cloud Computing plays a vital job in providing storage, infrastructure, and processing services. The demand for storing data in cloud is increasing day by day due to the large number of users. To protect the data stored in the cloud, data is often stored in an encrypted form. Cloud storage includes a large amount of duplicated data under different encryption schemes by different users. Existing solutions that deal with encrypted deduplication do not support controlled access, ownership revocation and file modification require uploading the entire file in encrypted form. In this paper, a scheme based on Incremental Proxy Re-Encryption (IPRE) scheme for deduplication with ownership challenge integrated with improved file modification operation in mobile cloud environment is proposed. The proposed scheme is compared with Proxy Re-Encryption (PRE) scheme based on turnaround time and energy consumption. During file modification operation, the proposed scheme shows remarkable improvement in results using the restricted storage and processing speed of mobile devices.

Keywords: Cloud Computing, Deduplication, Incremental Proxy Re-encryption, Mobile Cloud, Proof of Ownership

1 Introduction

With the advent of smartphones and social networking, data is generated and updated every second. The storage problem arising due to the exponential growth of data is solved by the storage services provided by cloud computing. Still there exists a problem of data replication on multiple servers, which increases the cost of storage space. Data deduplication technique in cloud computing helps to increase storage efficiency in cloud computing. Hadoop Distributed File System (HDFS) plays a vital role in cloud computing providing data reliability, with the burden of increased storage space.

Data privacy [1,2] is also important while storing users' data in the cloud. In order to protect the private data from adversaries as well as from Cloud Service Providers (CSP), the companies and customers may encrypt and store encrypted data in the cloud [3,4,5]. Encrypting the data stored in the cloud by different cloud users by using different keys and encryption techniques results in different ciphertext which is proved to be difficult to identify duplicates by the cloud server. Client-side encryption techniques to encrypt the file using the private key can be used to perform deduplication but increase the cost of key management issues [6].Another method to encrypt the file using a public key can be provided by CSP. This is not suitable if the cloud server is semi-trusted or untrusted.

Convergent encryption [7] is a cryptosystem that solves the problem of key management as the key to encrypt file is obtained from the file itself. The encryption key is the hash value generated from the file. The user encrypts the file using the hash value obtained from the file, uploads the file to the cloud server and retains the encryption key. Convergent encryption technique provides identical cipher text from an identical plain text file. The cloud storage performs deduplication over the ciphertext stored and any owner of the file can able to download the file from the cloud storage and decrypt using the encryption key obtained from the file. Convergent encryption provides better privacy and is open to file attacks.

Data deduplication is used to reduce the storage cost [8] and processing overhead. Due to the changing need of users, usage of data in cloud varies. To cope with the need of users, a dynamic deduplication scheme that supports

* Corresponding author e-mail: anniejoicegce@outlook.com

data ownership challenge is needed. Deduplication scheme should be independent of the size of data so that it is applicable for big data as well.

Deduplication can be performed in two ways. One that can be done on the server side. Server-side deduplication is easy to perform. The cloud server receives the file from the user and checks whether a duplicate of the file is already stored or unavailable in cloud. If it is available in the cloud server, it discards the file, otherwise it keeps the file in the cloud storage. On the other hand, deduplication is performed on the client side, in which the user calculates the hash value from the file. The client sends the hash value computed from the file to the cloud server. The cloud server verifies whether the hash value is already available or not. If it is available, the cloud server links the user with the existing file. Otherwise, the user is asked to upload the file. Client-side deduplication plays a vital role in mobile cloud computing environment, in which the mobile client accesses the cloud storage. Most of the public cloud storage services adapt client-side deduplication technique to reduce storage and bandwidth rate.

An important security threat in deduplication is the attacker gaining the hash value of the file and the cloud server adds the attacker as an additional owner of the file. This is due to the claim of ownership by the attacker with a single piece of gained hash value. Proof of Ownership (PoW) plays a vital role and the user should prove file ownership in addition to possession of file.

The contribution of the proposed scheme is to provide dynamic deduplication scheme for encrypted data in cloud storage that supports ownership management.

The organization of the paper is as follows. Section 2 gives a brief overview about related work. Section 3 discusses system architecture and security model. Section 4 describes the incremental proxy re-encryption Scheme for deduplication. System implementation and testing environment are discussed in section 5.

2 Related Work

2.1 Deduplication over encrypted data

To preserve the privacy of data stored in the cloud, and protect it from adversaries, users encrypt the data and store in cloud [9]. If the client encrypts the data and stores in the cloud server, deduplication cost will increase. This is due to the problem of using different encryption keys by different users. Message-Locked Encryption (MLE) [10] can solve this problem where the key used to encrypt and decrypt the file is generated from the file to be stored in cloud. Using cryptographic hash function, the user computes the hash value of the file to be stored and encrypts the file using the hash generated from that file. This technique is called Convergent Encryption (CE) introduced by Douceur et.al [7]. In CE technique, identical ciphertext is generated from identical plain text. The CE function is deterministic so that any user encrypting the plain text generates the same hash and encrypted data. CE is susceptible to brute-force attacks. ClouDedup [11] achieves confidentiality and performs block-level deduplication based on CE. It focuses mainly on two main operations: storing file or block and retrieving file or block. The operations such as removal, modification, and update were not taken into consideration. Bellare et.al [12] proposed DupLESS to overcome brute-force attacks in CE by using Key Server (KS) to derive keys. The secret key is generated by KS and other users are inaccessible to the key. The user obtained the message-derived key from KS by sending the hash computed from the file. An Oblivious Pseudo Random Function (OPRF) protocol is used to ensure that cloud server learns nothing about user input or resulting output and user learns nothing about secret key. Li et al. [13] proposed Dekey, a convergent key management mechanism using Ramp Secret Sharing Scheme (RSSS) and the computation overhead for encoding and decoding is less. In this method, the convergent key is distributed across multiple cloud servers. The key space overhead is high as the number of users increases. Data integrity is also an important consideration in cloud storage [14].

2.2 Proof of Ownership Verification and Others

Harnik et al. [6] proposed a randomized solution that does not reveal more facts about the file stored on the server. In randomized solution method, deduplication is performed on the stored file, when the number of duplicates of the file exceeds a threshold value. Client-side deduplication is a very effective mechanism that can be used in mobile devices that reduces storage cost and bandwidth usage. Due to the security vulnerability in client side deduplication, the user should prove the cloud storage server in which he/she owns the file rather than holding a hash value of the file. Halevi et al. [15] introduced the scheme of proof of ownership (PoW) using Merkle tree. The user computes the encoding X=E(F), by applying erasure coding E on the file F. Then the Merkletree(MT) is constructed over the encoded file MT(X) . The root of the Merkle tree and the total number of leaves in the tree is kept with the user. By examining the sibling paths of all the leaf nodes the PoW is verified. By using Merkle tree method Proof of Ownership (PoW) is achieved by sending the path of all leaf nodes to the cloud resulting in more computation and communication overhead. Yu et al. [16] proposed a probabilistic data structure, the bloom filter [17] which reduces the computation cost and overhead due to communication. The rate of growth of bloom filter may increase with the number of files stored in the cloud. A cryptographically secure and efficient scheme for Proof of Ownership is proposed by Yang et. al [18]. In this method, the client reveals only partial information about it to the server without uploading the

file, to prove the ownership. Dynamic spot checking technique is used to generate proof of ownership which reduces the computation overhead and communication cost between data uploader (holder) and CSP. A unique evidence is produced in each ownership challenge by consolidating the parts of original file which is randomly-sampled with the dynamic coefficients. Approved data deduplication is performed in hybrid cloud architecture [19].

3 Problem Formulation

3.1 System and Network Model

We propose a system to achieve deduplication of encrypted data at CSP based on ownership challenge. Even in the absence of data holders the proposed scheme can be able to perform deduplication. The system model of the proposed scheme is shown in figure 1.

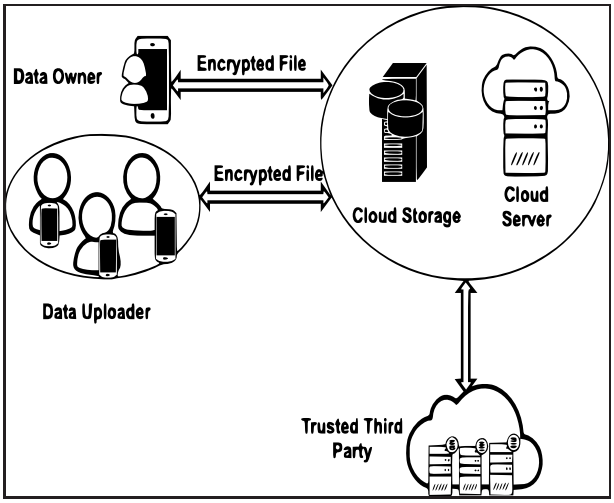

Fig. 1: System Model

The proposed system consists of three entities. (i) a semi-trusted CSP that offers storage services. (ii) data uploader is a user that can upload and store its data at CSP. One user is designated as data owner who creates the file. Data owner has a higher priority than other data uploaders. It is possible to have a number of data uploaders who are mobile users u_i,i=A,B,..... The data uploaders in the system could store the encrypted data in CSP. (iii) A fully-trusted third party (TTP) to verify data ownership and performs data deduplication. Data uploaders make the encrypted hash codes available to TTP for ownership verification.

4 Scheme for Deduplication

The various notations used in the proposed system are presented in table 1.

Table 1: System Notations

Notation	Description
G	Random number g belonging to group G_1 of prime order q
Z	Random number e(g,g) belonging to group G_T
(Pk_A,Sk_A)	Public key and secret key of user A
MAC_F	Hash value of the file F
r	A random number
R	Proxy Re-Encryption algorithm
C	Encrypted File F
C_A	Number representing $g^{x_A r}$
$rk_{A\rightarrow B}$	Re-encryption key from user A to user B
B	File Block
K	Total number of blocks
F	File F
$\neg$	Binary Negation
$\wedge$	Bitwise AND
$\vee$	Bitwise OR

4.1 System Setup

The proxy re-encryption [**?**] works on two cryptographic groups G_1 and G_T of prime order q with a bilinear map e: G_1 X $G_1 \rightarrow G_T$ symmetric pairing with the property of bilinearity. The random generators are $g \in G_1$ and $Z=e(g,g) \in G_T$ that can be used for encrypting the message M, decrypting the message C and re-encryption of the message. During system setup, the TTP generates a key pair for every data holder u_A on the cloud storage for proxy re-encryption.

$Sk_A=x_A$, $Pk_A=g^{x_A}$, where $x_A \in Z_q^*$

The re-encryption key is generated for data holder u_A at TTP by using the public key Pk_A. We select an elliptic curve H(a,b) of genus $g \geq 2$ over a finite field GF(p), where p is a prime number and its base point G of order q. The private key $D_A \in_R \{0,...., 2^{\sigma}-1\}$, and public key $P_A=\neg D_A \wedge G$ are generated and σ is a security parameter. To verify the identity of the mobile user the keys (Pk_A,Sk_A) and (P_A, D_A) are crucial and the values are bound to the unique identity of the user. The TTP also generates (PK_{TTP}, SK_{TTP}) independently for proxy re-encryption and broadcasts its public key PK_{TTP} to CSP and mobile users.

4.2 Token Generation

The mobile user generates data token (MAC) for the file F as follows: $MAC_F=H(H(F) \wedge G)$.

4.3 Encrypted Data Upload

Let's assume mobile user (data owner) A wants to store file F on the cloud. To encrypt the file, the mobile user A generates random number r$\in$ Z_q*. Then the mobile user encrypts the file F using the private key as below:

$$C_A = g^{x_A r} \tag{1}$$

$$C = F \cdot r \tag{2}$$

During encryption, the file F is divided into blocks of fixed size and encryption is performed on each block and finally, all the blocks are concatenated to get the original file.

$$F = ||_{k=1}^{n} B^k \tag{3}$$

The mobile user sends (C, C_A, MAC_F) to CSP.

4.4 Ownership Challenge Verification using Re-encryption

The scheme to verify data ownership challenge is depicted in figure 2. To download the file F from the cloud, the mobile user B requests the TTP. TTP challenges the mobile user B for data ownership by randomly choosing $s\in_R \{0,...., 2^{\sigma}\text{-}1\}$ and sends the value to B. The mobile user checks the value s such that $0 \leq s \leq 2^{\sigma}$-1 and computes y=H(F) $\vee$ (DB $\wedge$ s). To protect H(F), if y is known by a malicious node encrypting y using Pk_{TTP} $E(y,Pk_{TTP})$ and sends y and P_B to TTP. TTP computes H(y $\wedge$G $\wedge$s $\wedge P_B$) and computes MAC of file F. If H(y$\wedge$ G $\wedge$ s $\wedge P_B$) =MAC_F, the TTP will download and re-encrypt the message C_A as follows:

$$\begin{aligned} C_B &= e(g^{x_A r}, g^{\frac{x_B}{x_A}}) \\ &= e(g,g)^{r x_B} \\ &= Z^{r x_B} \end{aligned} \tag{4}$$

TTP sends the re-encrypted message (C, C_B) to mobile user B.

4.5 Decryption

By using C_B, the mobile user B decrypts the message as below:

$$(Z^{r x_B})^{\frac{1}{x_B}} = Z^r \tag{5}$$

$$F = (\frac{F.Z^r}{Z^r}) \tag{6}$$

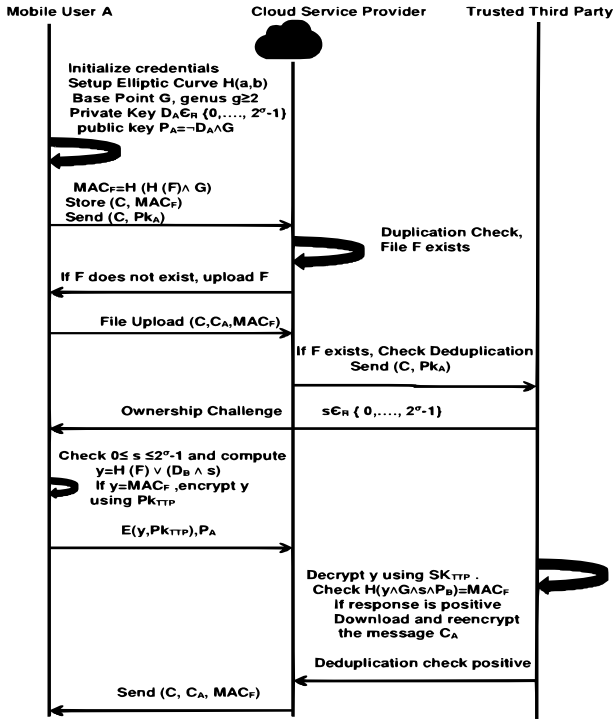

Fig. 2: Ownership Challenge and Verification

4.6 Duplication Check

The deduplication procedure is depicted in figure 3. Let's say mobile user A wishes to upload a file F. CSP verifies Pk_A and searches the cloud storage for existence of the file using MAC_F. If the duplication check is negative, it requests the mobile user to upload the file F. Mobile user A encrypts the file F using the private key C_A to get C. A sends (C, C_A, MAC_F) to CSP. If the duplication check is proved to be positive and the pre-stored file is from the same mobile user, the prevailing situation is informed to the mobile user. If the file upload request is from a different mobile user B, duplication is performed as follows:

$$\begin{aligned} C_B &= e(g^{x_A r}, g^{\frac{x_B}{x_A}}) \\ &= e(g,g)^{r x_B} \\ &= Z^{r x_B} \end{aligned} \tag{7}$$

The mobile user receives the secret key Sk_B=x_B and confirms the success of deduplication to CSP. Now, both the mobile users A and B can access the same file F stored at CSP.

4.7 Data Deletion

Let's assume the mobile user A (data owner) wants to delete the file F from CSP. The mobile user sends deletion

request to CSP along with (Pk_A, x_A, MAC_F). CSP verifies the validity of the request, removes the file F and blocks the mobile user A from later access to the file F. CSP deletes the encrypted file and related records if the deduplication record is empty.

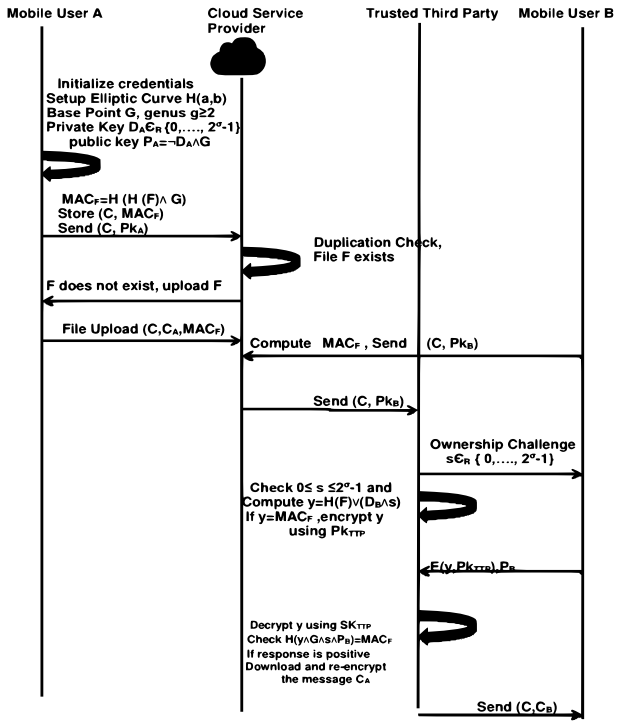

Fig. 3: Deduplication Procedure

4.8 Data Owner Management

Let's say the real data owner (mobile user A) stores the file later than the data holder (mobile user B). CSP can be able to store the encrypted file by data owner and share the cloud storage.

To verify the data ownership CSP contacts TTP by sending all mobile users public key PK_i, i=1,2,.....n, if CSP does not know the re-encryption key of a particular mobile user. TTP issues the re-encryption key by downloading the file and re-encrypts the message and sends it to CSP if the result of ownership challenge is proved to be positive. CSP deletes the encrypted file of mobile user B and replaces it with the encrypted file of mobile user A and updates the deduplication records corresponding to file F.

4.9 Encrypted Data Update

Let's say mobile user B (data holder) could update encrypted file stored at CSP by downloading (C, C_B, MAC_F) from the cloud storage. Mobile user B sends an update request to CSP: { x_B, Update C}. CSP contacts TTP for ownership challenge and the mobile user B performs update operation as follows:

$$\begin{aligned}(C_B, g^{\frac{1}{x_B}}) &= (\frac{g^{x_B r}}{C_B}, g^{\frac{1}{x_B}}) \\ &= (g,g)^{x_B r \cdot \frac{1}{x_B}} \\ &= (g,g)^{r} \\ &= Z^{r}\end{aligned} \tag{8}$$

$$C_{Update} = F_{Update} \cdot r \tag{9}$$

The mobile user computes the new MAC for the file F and sends (C, C_B, MAC_F) to CSP.

5 Results and Discussion

5.1 Implementation and Test Environment

The mobile application (mobile cloud client) is developed, tested and debugged using Android SDK. The mobile cloud client is deployed on the hardware specification in Table 2. Google App Engine (GAE) is used to host a web instant with 2.4GHz processing and 512MB RAM capacity.

Table 2: Hardware Specification

Hardware Specification	
(Sony Xperia S smartphone)	CPU: Dual Core 11.5 GHz
	RAM: 1GB
	Storage: 32GB
	Operating System: Android OSV4.0.4
	Battery: 1750mAh
	Mobile Application Development
	toolkit: Android SDK
Software Environment Library	Java Pairing Based
	Cryptography Library (JPBC)

5.2 Security Analysis

The proposed scheme provides a secure way to deduplicate and store data in the cloud by preventing the CSP and TTP to access the original file. The security is achieved through incremental proxy re-encryption scheme and Elliptic Curve Cryptography (ECC).

Proposition 1: The proposed scheme guarantees that only the legitimate users can access the original file F from the cloud and the file F can be deduplicated and stored in a secure manner and free from collusion attack.

Proof: The file to be stored in the cloud is always in encrypted form, so it is not possible for CSP to access the

file content. Although TTP gains encrypted file C and the public key of the user, TTP is prevented from accessing F, since access to file F is blocked by CSP.
Confidentiality is achieved by incremental proxy re-encryption. The original file F is impossible to obtain from the hash function H (H (F)∧ G). Another way to disclose F is through obtaining from ciphertext C, which is impossible for incremental proxy re-encryption. To achieve integrity, the MAC of the downloaded file is compared with the calculated MAC.
CSP cannot gain the number representing to get the file F, because it does not know anything about the secret key of any mobile user. Only the authorized mobile user can be able to obtain the file F from the re-encrypted message from TTP. This guarantees that F can be accessed only by legitimate users.

Proposition 2: The deduplication check is performed with the hash code H (H (F)∧ G) and it cannot be obtained by any unauthorized cloud users.

Proof: H (H (F)∧ G) plays a key role for data deduplication check. When the token is captured by some malicious mobile clients, H(F) is still secured due to elliptic curve discrete logarithmic problem. CSP can be able to get information about the token but nothing else. Though TTP can be able to know the value y=H (F) ∨ (D_B ∧ s), it cannot obtain D_B and H(F). H(F) cannot be transmitted between any unauthorized mobile users and is not obtained by any other mobile users. Even when CSP conspires with any malicious mobile clients, the malicious mobile user can only obtain H (H (F)∧ G), not H(F) and thus it is impossible to access the file F stored at cloud by succeeding the ownership verification performed by TTP.

Proposition 3: A mobile cloud user should have the file F to succeed the ownership verification of TTP.

Proof: The mobile user can generate H(F) with the original file F and compute y=H (F) ∨ (D_B ∧ s) with D_B and s provided by TTP. TTP computes H(y∧G ∧ s ∧ P_B) =H((H (F) ∨ (D_B ∧ s) ∧ G ∧ s ∧ P_B)=H((H(F)∧ G) ∨ D_B ∧ s∧ G ∧ s ∧ ¬ D_B ∧ G =MAC_F, which completes the ownership verification of TTP.

5.3 Computational Complexity

In the proposed system, the mobile user (data owner) is responsible for system setup, encryption, and token generation. ECC key generation needs one-point multiplication, and system setup is performed only once for all file storage operations. The time taken to compute hash of the file depends on the file size and it is very fast, so it cannot be considered. Token generation involves two hash operation and one-point multiplication. Thus the computation complexity for both setup and file upload is *O(1)*.
A mobile user uploads a file to CSP by generating token H (H (F)∧ G). The CSP saves the file only if the token does not exist already. If a mobile user (data holder) uploads the same file, CSP re-encrypts the message with the help of TTP to verify ownership challenge. The re-encryption operation requires one pairing. The computation complexity is *O(n)* if the same file is uploaded by n mobile users (data holders).
A mobile user (data holder) wants to upload the same file that has been stored already in the cloud, a token is generated by the data holder H (H (F)∧ G). During ownership challenge, the data holder computes y=H (F) ∨ (D_A ∧ s). It is not necessary for the data holder to encrypt the file for upload. The encryption E(Pk_{TTP},y) requires 2 exponentiations. So the computation complexity of data holder is *O(1)*.
TTP is responsible for re-encrypting the message by challenging the data ownership. TTP randomly selects the value s and compares H(y∧ G ∧ s ∧ P_B) =MAC_F which requires two-point multiplications. TTP re-encrypts the message which requires one pairing. TTP re-encrypts the message for all authorized mobile users (data holders) and the computational complexity is *O(n)*.

Table 3: Data set used to calculate the total number of operations (Block size-64 bytes)

S.No.	File Size(in bytes)	No.of Files	Total No. of operations
1.	5120	50	4000
2.	10240	50	8000
3.	15360	50	12000
4.	20480	50	16000
5.	25600	50	20000

5.4 Encryption and Decryption

Java Pairing-Based Cryptography Library (JPBC) is used by the mobile client to encrypt the files. During encryption, file is divided into blocks and $F \cdot Z^r$ multiplication operations is performed and to decrypt a file ($F \cdot Z^r/Z^r$) division operation is performed and presented in table 3. The pairing operation is constructed on the curve $y^2=x^3+x$ over the prime field F_q. The time taken for each basic operation is presented in table 4. The computation complexity of the proposed scheme is compared with Yan Zheng et al. [23] and the comparison is presented in table 5. The proposed scheme is evaluated and its performance is tested based on the algorithms for deduplication, encryption, and decryption. The time taken to perform encryption and decryption on different file size is shown in figure 4 and 5 respectively. The encryption operation on the dataset and uploading operation consumes more time on the mobile device; however, the proposed scheme significantly improves file modification operation. The turnaround time is calculated for a various

Table 4: Time taken for basic operation

S.No.	Basic Operation	Time Taken (millisecond)
1.	Pairing	5.87
2.	EXP(G_1)	3.86
3.	EXP(G_T)	0.67
4.	Point Multiplication	0.6
5.	Multiplication	0.31
6.	Division	0.27

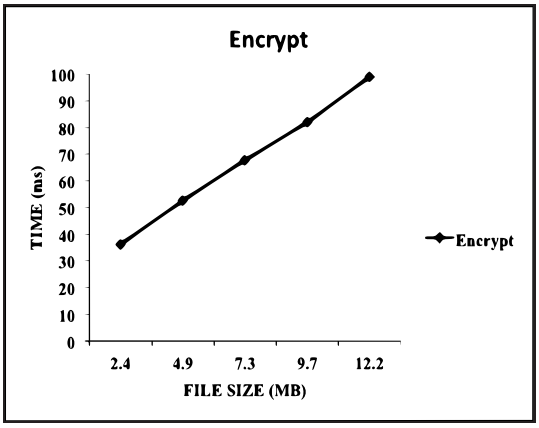

Fig. 4: Time taken to encrypt file

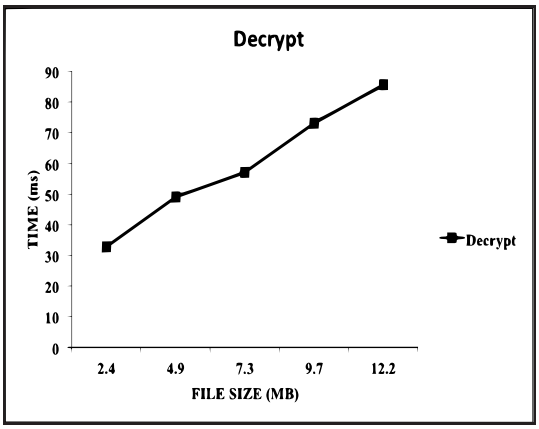

Fig. 5: Time taken to decrypt file

number of block insertion/modification operation and it includes the time taken to read the file and the time taken to encrypt the file. The uploading and downloading time are not taken into account. The operation is performed on blocks of various size 1,2,3,4,5, and 6 and compared with Proxy Re-encryption (PRE) scheme and the results are presented in figure 6. The energy consumption for a various number of block insertion/modification operation

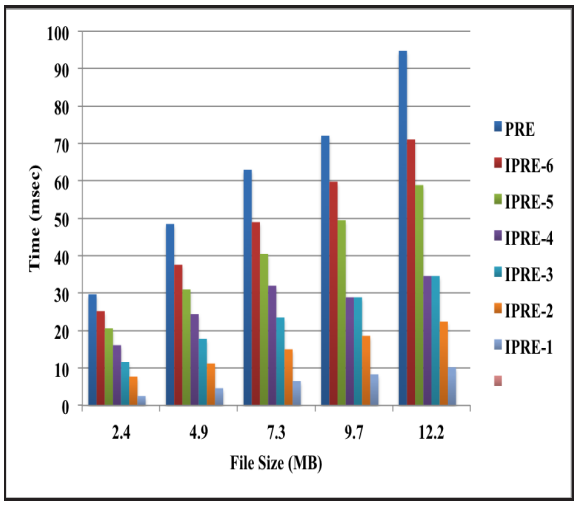

Fig. 6: Turnaround time-Block Modification/Insertion

is calculated and presented in figure 7. The energy consumption for file uploading and downloading operation is not considered.

5.5 CPU Utilization and Memory Utilization

During the encryption of 30 files of size 1,024,000 bytes, the utilization of CPU is evaluated in percentage, Private Memory (PM) used is evaluated in kilobytes (KB) and Proportional Set Size memory (PSS) is measured in kilobytes. PM is the memory that is used by a process completely on its own and it is used in resident memory. Proportional set size is the portion of memory (Random Acess Memory) used by a process and it includes the amount of private memory used by a process and proportion of memory shared by one or more processes. PSS can be calculated as

$$PSS = \frac{SM}{N} \tag{10}$$

SM is the memory shared among at least one process on the mobile device and N represents the total number of processes utilizing the shared memory. A mobile application to evaluate the utilization of resources that executes android top command for each second is used to get the information about the utilization of resources for each of the running processes in the mobile device. The experiment setup is repeated four times and the same system configuration is used. The average results of the experiment are presented in table 6.

Table 5: Computation complexity of data owner, CSP, data holder by comparing with (Yan Zheng et al., [23])

System Entity	Procedure	Proposed Scheme	Zheng et al. Scheme	Complexity
Mobile user	Setup	1Pt.M+1Exp	1Pt.M+1Exp	***O(1)***
(Data Owner)	File Upload	1Pt.M+1Exp+1mult	2Exp+1Pt.M	
CSP	Re-encryption	1Pair	1Pair	***O(n)***
	System Setup	1Pt.M+1Exp	1Pt.M+1Exp	
Mobile User	Ownership Challenge	Bitwise AND and	2Exp+1Pt.M	***O(1)***
(Data Holder)	Response	Bitwise OR		
	Data Upload	-	-	
	Decryption	1 Division	1Exp	
TTP	System setup	1Exp	1Exp	***O(n)***
	Ownership Challenge	Bitwise AND and	2Exp+2Pt.M	
		Bitwise OR		

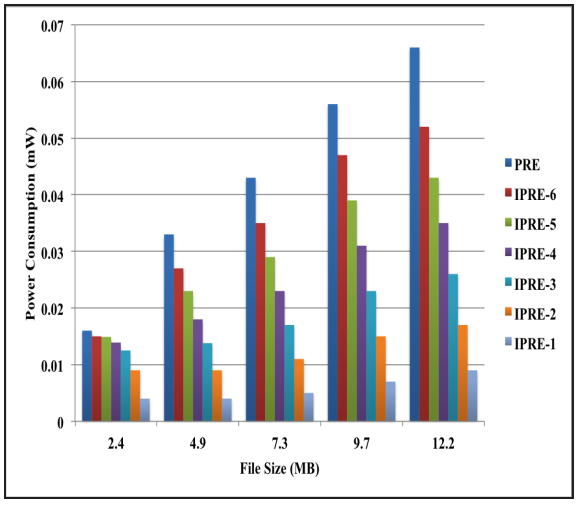

Fig. 7: Energy consumption- Block Modification/Insertion

6 Conclusion

Deduplication plays a significant role in reducing the storage cost, network resources, and bandwidth. In this paper, encrypted data with deduplication based on ownership challenge is proposed for mobile cloud environment. The incremental proxy re-encryption provides a significant improvement in block insertion/modification operations. The incremental proxy re-encryption is also compared with proxy re-encryption based on utilization of CPU, memory usage in terms of primary memory and shared memory, turn around time, and energy consumption on the mobile device. The performance improvement is evaluated based on the number of blocks the mobile user (data owner) updates in the cloud. Mobile users can securely access encrypted data stored in the cloud because only authorized data holders can decrypt the file.

Table 6: CPU utilization and memory usage - to encrypt 30 files of size 1,024,000 bytes

CPU Utilization	Time (msecs)	Private Memory(KB)	Shared Memory (KB)
PRE-30 files of size 1,024,000 bytes			
53.00	105,382.60	16,422.31	22,580.14
54.13	105,335.86	14,818.54	20,932.57
52.82	105,094.24	16,027.52	21,856.15
53.76	105,238.42	15,756.12	21,789.61
IPRE-30 files of size 1,024,000 bytes(8)			
54.47	106,948.83	16,243.62	18,209.18
53.45	106,942.71	16,977.31	18,837.03
54.15	106,944.87	17,424.85	19,281.43
54.02	106,945.47	16,881.93	18,775.87
IPRE-30 files of size 1,024,000 bytes(64)			
55.34	107,044.67	16,975.63	17,967.52
55.40	107,059.61	16,920.35	17,985.95
54.83	106,991.52	17,028.10	18,136.30
55.19	107,031.93	16,974.69	18,029.92
IPRE-30 files of size 1,024,000 bytes(128)			
56.41	107,160.68	17,076.51	18,009.23
56.37	107,155.13	17,071.73	18,012.82
57.07	107,166.89	17,106.92	19,192.10
56.61	107,160.90	17,085.06	18,404.71
IPRE-30 files of size 1,024,000 bytes(256)			
57.38	107405.51	17,266.30	19270.90
57.39	107506.53	17,133.32	18398.16
58.04	107559.36	17,351.68	19260.63
57.60	107490.47	17,250.43	18976.57

References

[1] Y. Shin, D. Koo, and J. Hur, A Survey of Secure Data Deduplication Schemes for Cloud Storage Systems, ACM Computing Surveys, Vol.49, No.4, pp.1-38 (2017).

[2] K. Hashizume, D. Rosado, E. Fernandez-Medina, and E. Fernandez, An analysis of security issues for cloud computing, Journal of Internet Services and Applications, Vol.4, No.5, pp.1-13 (2013).

[3] N. Baracaldo, E. Androulaki, J. Glider, and A. Sorniotti, Reconciling End-to-End Confidentiality and Data Reduction In Cloud Storage, Proceedings of the 6th Edition of the ACM Workshop on Cloud Computing Security - CCSW-14, pp.21-32 (2014).

[4] W.K. Ng, Y. Wen, and H. Zhu, Private data deduplication protocols in cloud storage, ACM Symposium on Applied Computing, pp.441-446 (2012).

[5] M.W. Storer, K. Greenan, D.D.E. Long, and E. L. Miller, Secure data deduplication, 15th ACM Conference on Computer and Communications, pp.1-10 (2008).

[6] D. Harnik, B. Pinkas, and A. Shulman-Peleg, Side channels in cloud services: Deduplication in cloud storage, IEEE Security and Privacy, Vol.8, No.6, pp.40-47 (2010).

[7] J.R. Douceur, A. Adya, W.J. Bolosky, D. Simon, M. Theimer, and P. Simon, Reclaiming space from duplicate files in a serverless distributed file system, ICDCS 2002: Proceedings of the 22nd International Conference on Distributed Computing Systems, pp.617-624 (2002).

[8] D.T. Meyer, and W.J. Bolosky, A study of practical deduplication, ACM Transactions on Storage, Vol.7, No.4, pp.1-20 (2012).

[9] M.Ali, R. Dhamotharan, E. Khan, S.U. Khan, A.V. Vasilakos, K. Li, and A.Y. Zomaya, SeDaSC: Secure Data Sharing in Clouds, IEEE Systems Journal, Vol.11, No.2, pp.395-404 (2017).

[10] M. Bellare,S. Keelveedhi, and T. Ristenpart, Message-Locked Encryption and Secure Deduplication, Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp.296-312 (2013).

[11] P. Puzio, R. Molva, M. Onen, and S. Loureiro, ClouDedup: Secure deduplication with encrypted data for cloud storage, Proceedings of the 5th International Conference on Cloud Computing Technology and Science, CloudCom, pp.363-370 (2013).

[12] M. Bellare, S. Keelveedhi, and T. Ristenpart, DupLESS: Server-Aided Encryption for Deduplicated Storage, IACR Cryptology ePrint Archive, Report No.2013/429 (2013).

[13] J. Li, X. Chen, M. Li, J. Li, P.P.C. Lee, and W. Lou, Secure deduplication with efficient and reliable convergent key management, IEEE Transactions on Parallel and Distributed Systems, **25**(6), 1615-1625 (2014).

[14] Q. Zheng, and S. Xu, Secure and efficient proof of storage with deduplication. Proceedings of the Second ACM Conference on Data and Application Security and Privacy - CODASKY'12, pp.1-12 (2012).

[15] S. Halevi, D. Harnik, B. Pinkas, and A. Shulman-Peleg, Proofs of Ownership in Remote Storage Systems,Proceedings of the 18th ACM Conference on Computer and Communications Security, pp.491-500 (2011).

[16] C. M. Yu, C. Y. Chen, and H. C. Chao, Proof of ownership in deduplicated cloud storage with mobile device efficiency. IEEE Network, Vol.29, No.2, pp.51-55 (2015).

[17] J. Blasco, R. Di Pietro, A. Orfila, and A. Sorniotti, A tunable proof of ownership scheme for deduplication using Bloom filters, 2014 IEEE Conference on Communications and Network Security, CNS 2014, 481-489 (2014).

[18] J.R. Chao Yang, Jian Ren, and Jianfeng Ma, Provable ownership of files in deduplication cloud storage, Security and Communication Networks, Vol.8, No.14, pp.2457-2468 (2015).

[19] Jin Li, Yan Kit Li, Xiaofeng Chen, Patrick P. C. Lee, W. Lou, A Hybrid Cloud Approach for Secure Authorized Deduplication, IEEE Transactions on Parallel and Distributed Systems, Vol.26, No.5, pp.1206-1216 (2015).

[20] G. Ateniese, K. Fu, M.Green, and S.Hohenberger, Improved proxy re-encryption schemes with applications to secure distributed storage, ACM Transactions on Information and System Security, Vol.9, No.1, pp.1-30 (2006).

[21] A.N. Khan, M.L.M. Kiah, S.A. Madani, M. Ali, A. Khan, R. ur and S. Shamshirband, Incremental proxy re-encryption scheme for mobile cloud computing environment, The Journal of Supercomputing, Vo.68, No.2, pp.624-651 (2014).

[22] P.K. Tysowski, and M.A. Hasan, Re-Encryption-Based Key Management Towards Secure and Scalable Mobile Applications in Clouds, IACR Cryptology ePrint Archive, Report No.2011/668 (2011).

[23] Z. Yan, W. Ding, X. Yu, H. Zhu, and R.H. Deng, Deduplication on Encrypted Big Data in Cloud. IEEE Transactions on Big Data, Vol.2, No.2, pp.138-150 (2016).

Sebastian Annie Joice is a part-time Ph.D. student and working as Assistant Professor in the Department of Computer Science and Engineering in Government College of Engineering, Srirangam. She received B.E Computer Science and Engineering from Bharathidasan University, Tiruchirappalli and received M.E. Computer Science and Engineering from Anna University, Chennai. Her research interests include Security, Cloud Computing,and Automata Theory.

M. A. Maluk Mohamed obtained his Ph.D. degree from IIT Madras, Chennai in the year 2006. He is a Professor of M.A.M. College of Engineering, Affiliated to Anna University, Chennai. He has(co-)authored over 80 research articles published in refereed journals and conferences, and is a frequent invited speaker at conferences and institutions all over India. His current research focus is on Distributed Computing, Mobile Computing, wireless Sensor Networks, Cluster Computing, Grid Computing, etc. He is a member of the ACM, IEEE, ISA, IARCS and life member of the Computer Society of India.