JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Data Integrity Verification Scheme with Deduplication on Encrypted Data

Meena Kumari
Department of Computer Science and Applications
Kurukshetra University, Kurukshetra, Haryana, India.

Jyoti Rani
Department of Computer Science and Applications
Kurukshetra University, Kurukshetra, Haryana, India.

Abstract*— With the continuous and exponential increase in data size, cloud computing (CC) has received a proliferating attention due to its storage service. Along with a wide range of benefits offered by CC, it suffers from various security concerns such as data confidentiality, integrity, and diminished storage capacity etc. Hence, this paper proposes a data integrity verification scheme which supports data deduplication on encrypted data to mitigate adverse effects of these security concerns. It uses the techniques of proxy re-encryption (PRE) and one-way hashing to solve the issues of ownership verification and data integrity. Data deduplication eliminates the need to store duplicate copies of repeating data while preserving the confidentiality of sensitive data. Extensive experimental and simulation results exhibit that the proposed method is efficient and incurs minimal computational and communication overhead.

Keywords*— Deduplication, hashing, integrity.

I. INTRODUCTION

The cloud computing (CC) is a paradigm which offers a uniform exposure to distributed network spread over wide area as and when needed [1]. CC comes with inherent features such as, flexibility, scalability and reliability in contrast to its traditional IT implementations. Even though, in CC, services, does not reside in locally managed premises yet it is regulated by cloud provider. This particular phenomena fetches various security issues along with storage capacity concerns.

With the hypothetically unlimited storage space provided by cloud providers, users can take on lease as much space as they require and cloud service providers tend to minimalize redundant data and amplify space savings. A scheme which comes to rescue in this regard is deduplication. Data deduplication is a technique which eliminates the need to store replicated data more than once. Only a single copy is savedand further requests for storage of similar data is fulfilled by only adding the user to owners list of that data.

Due to space and cost savings, deduplication has attracted a wide attention among cloud storage providers. Though data deduplication fetches plenty of benefits but is in conflict with encryption which is essential to maintain data confidentiality. The encrypted form of replicated data belonging to different users is distinct, hence rendering deduplication unfeasible. Convergent encryption propositioned by [6] is to achieve viability of data confidentiality and deduplication. In convergent encryption, data is encrypted or decrypted with a convergent key, which is generated by applying hashing on data. Encrypted data is sent for storage while preserving the key. The key used for encryption is obtained from the data itself, therefore, replicas produce the same convergent key and cipher text. Although convergent encryption seems secure and effective, it suffers from security attacks [7, 8] like dictionary attack and brute force attacks.

This paper proposes a data integrity verification scheme which supports data deduplication on encrypted data. The proposed work renders the concept of re-encryption and owner verification. Purposely, the significance of this paper can be presented as below:

- Support users data integrity stored at cloud storage. The user can challenge CSP to verify its data integrity.
- Support of deduplication on Encrypted data to guarantee privacy and storage efficiency at the same time
- A lightweight approach for verification of data ownership and efficient duplication check by using a security challenge.
- Support data dynamic operations of the update, insert and delete.
- Assessment of security and performance of the proposed method is arbitrated through experimental study and evaluation. The results generated through experiments and simulation exhibits its efficiency and applicability.

The rest of the paper is organized as follows: Section II elaborate related work, Section III provides preliminary of the notations used throughout the paper. Proposed system's model is introduced in section IV and algorithms in section V. section VI analyzes the security and performance of the proposed algorithms followed by experimental results and discussion in section VII. In last section concluding remarks are presented.

II. RELATED WORK

Douceur et al. [6, 19], Wen et al. [14] and P. Meye et al. [15]had used the concept of convergent encryption. According to this concept, data was encrypted by using hash digest generated from data as a key. A unique key was used for encrypting duplicate data, which resulted in the identification

of duplicates. [14] had used this scheme to deduplicate images whereas, [15] had presented a two-phase deduplication scheme for intra-user and inter-user deduplication. Their technique suffers from various security threats like dictionary attack and brute force attacks. Pasquale [9] has proposed ClouDedup, to cope with the limitations of convergent encryption. ClouDedup was a block-level scheme for deduplication. Their scheme had included a separate component for performing an additional encryption operation and key management. Their scheme could not handle data and key management issues raised after data deletion. As user had encryption key even after data deletion, it can lead to data access in future. Yu Chia-Mu [11] had proposed ZEUS (zero-knowledge deduplication response) framework. They had developed ZEUS and ZEUS+, two privacy-aware deduplication protocols. Both of these frameworks worked on delimiting leakage of status information of data existence during deduplication check. Their proposed work incurred additional communication overhead. Zhe SUN's [13] proposed deduplication scheme was comprised of major two components, a deduplication application at outside and HDFS at the back end. HDFS was incorporated for building up a quick indexing mechanism. Their proposed work does not handle deduplication on encrypted data.

Yang Chao et al. [10] had presented a scheme in which a client provides a proof of ownership by using a whole file instead of part of it. Their method uses the concept of spot checking in which the client only needs to know dynamic coefficients and randomly picked index of files owned. G. Ateniese et al. [12] had proposed numerous re-encryption techniques to show the benefits of proxy re-encryption as a

means of providing access control to a secure file system. Fan et al. [26] had proposed a hybrid data deduplication scheme along with partial semantic security. Their scheme supports deduplication on both plaintext and cipher text, but cannot support deduplication on encrypted data. In their scheme CSP is familiar with the encryption keys hence raises security concerns.

Existing one-way hash algorithms involved use of a singular matrix for hash generation. Distinct methods were employed by algorithms in [16], [18], [20] and [22] for singular key matrix generation. Later on, this generated matrix is utilized for final hash generation depending upon matrix multiplication operation. High algorithm execution time which further rises with increase in file sizes were the major limitations of these algorithms. Moreover, the execution time involved only in matrix multiplication operation only is counted in [16]. Although, a considerable amount of time is consumed in initial steps of data preprocessing. In [21], compression technique is applied on final hash which results in data loss. Kavuri Satheesh et al. [17] has encouraged the involvement of TPA for integrity verification. But TPA is also vulnerable to security risks and require full user's trust. N Gowtham Kumar
[23] had recommended a signature scheme for data integrity verification by incorporating conventional network security measures. Rashmi M. Jogdand et al. [24], Li Yantao [25] do not consider practical storage and retrieval of data to Cloud server. They additionally don't support deduplication of encrypted data.

In lieu of the above-described schemes and their shortcomings, this paper proposes a data integrity verification scheme which also supports ownership verification and data deduplication while preserving its privacy through encryption.

III. Notations and Terminology

This section defines the notations and terminology used in this paper.

A. Symmetric Encryption

Symmetric encryption utilizes a shared key named as SymmKey to encrypt and decrypt User's data. This operation involves following three functions:

- KG( )→SymmKey, is the key generation algorithm that produces SymmKey for encrypting or decrypting data.
- Enc(Ptext, SymmKey) → Ctext, is the encryption function that takes the secret key SymmKey and User's Plaintext data Ptext as input and then outputs the ciphertext Ctext.
- Dec(Ctext, Symmkey) → Ptext is the decryption function that consumes secret key SymmKey and ciphertext Ctext as input and then produces the original Plaintext data Ptext.

B. Proxy Re-encryption

- PREKG( ) → (PK_i , SK_i) is the key generation algorithm for PRE which generates public and Private key pair for entity i.
- EncPRE($Ptext_i$, PK_i) → $Ctext_i$ is the PRE encryption algorithm which takes as input user's Plaintext data $Ptext_i$ and PK_i and produces as output Ciphertext $Ctext_i$.
- DecPRE($Ctext_i$, SK_i) → $Ptext_i$ is the PRE decryption algorithm which takes as input $Ctext_i$ and SK_i and generates user's Plaintext data $Ptext_i$.
- PREReKG(PK_i, SK_i, PK_j) → $RK_{i→j}$ is Re-encryption Key generation algorithm for PRE algorithm which takes as input PK_i , SK_i , and PK_j and produces Re- encryption key for the proxy.
- PREReEnc($RK_{i→j}$, $Ctext_i$) → $Ctext_j$ is Re-encryption algorithm for PRE algorithm which on input $RK_{i→j}$ and $Ctext_i$ produce $Ctext_J$ as output. PREReEnc($RK_{i→j}$, $Ctext_i$)= EncPRE(Ptext, PK_j)= $Ctext_j$, this can be decrypted using Secret key SK_j

C. Hash generation

- HF(Ptext, Hkey) → Hashtag is a hash function which on input Ptext And Hkey generate Hashtag for integrity and duplication check.

IV. BACKGROUND

This section illustrates various entities involved in data storage, integrity verification and encrypted deduplication checking in CC environment.

(i) Cloud Storage Service Provider (CSSP). This entity is responsible for providing data storage service. To minimize the storage cost, it avoids the duplicated data storage and maintains only unique data and its associated hashtags. This is assumed that CSSP has a huge storage capability and computation resources and it never colludes with user or Auditor.

(ii) Data Users. A user (Ui) is an entity that wants to store its private data to the Cloud storage site and access it later. It is assumed that system supports multiple users (Ui's, I = 1 to n). One user is data owner who actually owns data and others are designated as naive users who wish to access data. Each user has issued a set of access rights during system setup phase of the system.

(iii) Auditor Party. This party is fully trusted by the data users. As Cloud users lack computing resources and infrastructure, hence this entity is introduced to assist in verifying data ownership and handling data There can be a hybrid approach of BiCloud which consists of a public and a private cloud [2]. A company might archive its data to the public cloud and continue to use services of private cloud for performing functions of a third party.

The data owner sends its encrypted data to CSSP along with the Hashtag generated from HF( ) and is used for data duplication check. The data owner encrypts SymmKey with PK_{AP} and issues the encrypted key to CSSP. The data owner during data upload generates a unique Hashtag before uploading data by using Hash function HF ( ). This Hashtag is used to identify any unintended changes in data during storage as Hashtag of changed data does not match the stored Hashtag.

Data duplication check is performed at the time when a naïve user U_i tends to save replica of previously stored data. This is checked by CSSP by comparing hashtags. In case of positive result, CSSP communicates with AP by issuing the hashtag and the data owner's PRE key. The AP ensures the data ownership of naive user and then passes a re-encryption key to the user to convert the encrypted SymmKey in a form which can only be decrypted by the entitled naive user.

Proposed algorithm also support data dynamic operations of data insert, update and delete.

(i) Data Deletion. When a naïve user deletes data from the repository, CSSP maintains the index for redundant data users and deletes that naïve user from the list of eligible users. If the remaining entries are there in the index, the CSSP will not remove the stored data, but restrict data access of that naïve user that requests data deletion. In case of rest records fall vacant, the encrypted data will be removed by CSSP.

(ii) Data Update. If stored data and SymmKey is updated by a data owner with data' and SymmKey' and the new encrypted data' is sent to CSSP to supersede old data, CSSP releases the new re-encrypted SymmKey' to all eligible naïve users by contacting AP.

(iii) Data Insert. This operation is performed in the same way as discussed above.

AP challenge user U_i to ensure that it is the actual data owner. CSSP performs duplication check by ensuring if the Hashtag X_i=HF(HF(Ptext×P) of data Ptext has stored earlier or not. This scheme makes sure that CSSP does not reveal HF(Ptext) to any unauthorized entity. If similar Hashtag persists, CSSP hand over task of further verification to AP, which arbitrary selects C, where C€R (0; . . . ; 2ˡ-1) where R is a random number and challenges U_i by C. U_i ensure that $0 \le C \le 2^{l}-1$, and calculate Y = HF(Ptext)+(S_i×C), where S_i€R (0; . . . ; 2ˡ-1) is the ECC secret key of user Ui and sends $Enc(PK_{AP}, Y)$ and Vi to AP. $V_i = -S_i \times P$ is the public key and γ is a security constraint. AP calculates HF(YP+ CV_i) and matches it with X_i. If HF(YP+ CV_i) = X_i ,AP generates reencryption key $ReEncKey_{AP \rightarrow ui}$ and issues it to CSSP. CSSP re-encrypts $ReEnc(PK_{AP};SymmKey)$ and provides the re- encrypted key to U_i.

Data owner U_i encrypts its secret key $SymmKey_i$ with $Enc(PK_{AP},SymmKey_i)$ and publishes it along with encrypted data $Enc(SymmKey_i, Ptext)$ to CSSP. If a naïve user U_j tries to store similar encrypted data using a different key, CSSP calls AP to challenge ownership and issue $SymmKey_i$ to the eligible naïve user U_j .

V. PROPOSED ALGORITHM

This section discusses the algorithms and procedures followed during system design. Let's assume data owner U_j archive its data $Ptext_j$ to CSSP by encrypting it with $SymmKey_j$. U_k is a naïve user who wishes to upload duplicate data. The flowchart shown in Fig.1 illustrates the working of the Proposed algorithm.

Following sections describe the different phases of the proposed algorithm.

A. *Setup Phase*

Each Cloud data user Ui generates PK_i and SK_i for PRE. The public key PKi is consumed to generate re-encryption key for Ui by AP. P is taken as a base point in Elliptic curve cryptography (ECC) which is distributed between system entities, Si€R (0; . . . ; 2-1) is the ECC secret key of user U_i and $V_i = -S_i \times P$ is the public key and γ is a security constraint. The keys (PK_i,SK_i) or (V_i, S_i) of U_i are obligated to a distinct identifier of the user. This binding uniquely identifies a particular user in the system. AP produce PK_{AP} and SK_{AP} for PRE and distribute PK_{AP} to all CSSP users.

B. *Integrity Checking Phase*

To generate a unique Hashtag and to guarantee data integrity, this paper incorporates the author's prior proposed One-way hash algorithm [3, 4]. The algorithm works in two phases namely hash key HKey generation and hashtag generation. Algorithm 1 shows the steps involved in hashtag generation. The generated Hashtag is used in Data Deduplication phase to check whether duplicate data exists or not.

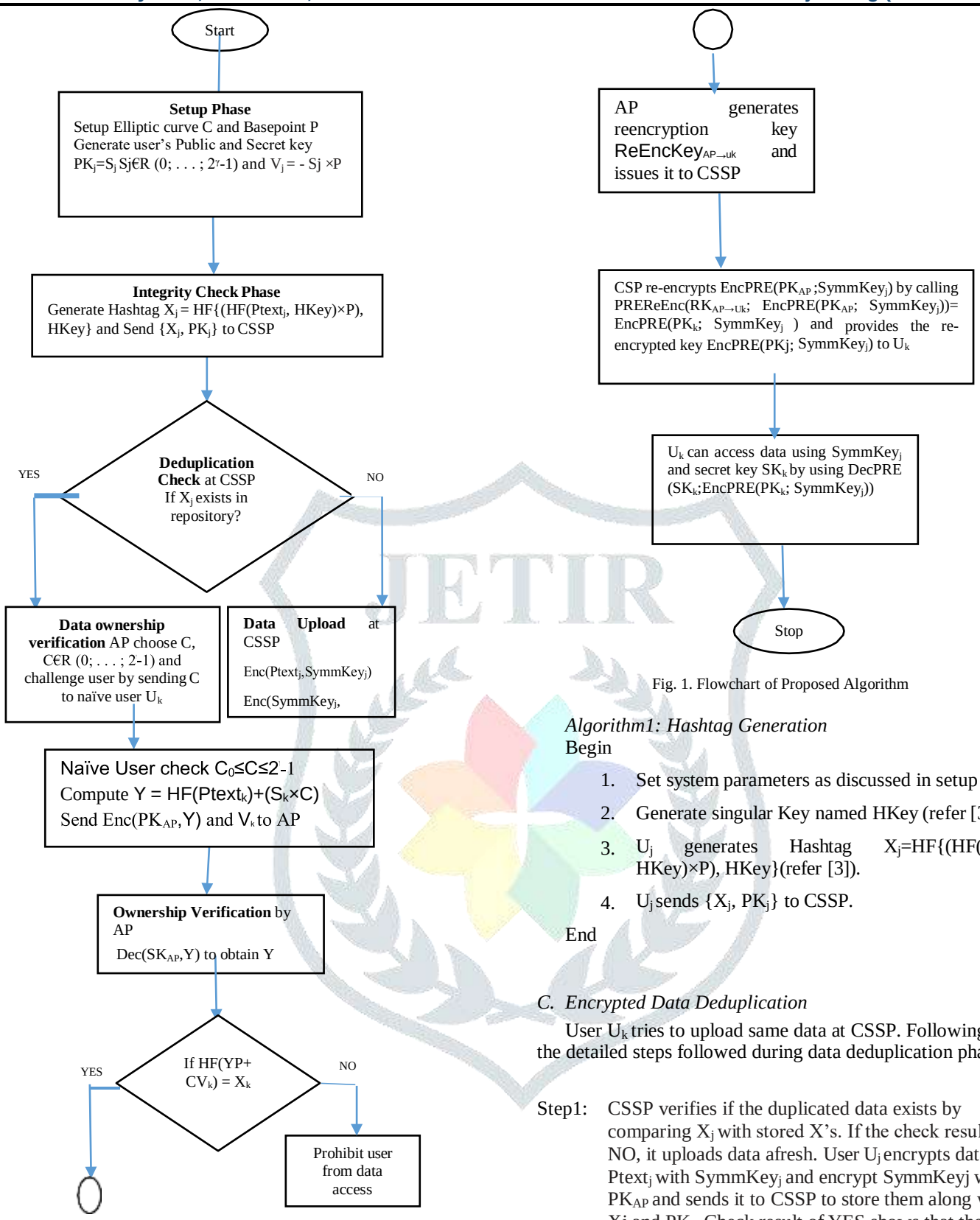

Fig. 1. Flowchart of Proposed Algorithm

Algorithm1: Hashtag Generation

Begin

1. Set system parameters as discussed in setup phase.
2. Generate singular Key named HKey (refer [3]).
3. U_j generates Hashtag X_j=HF{(HF($Ptext_j$, HKey)×P), HKey}(refer [3]).
4. U_j sends {X_j, PK_j} to CSSP.

End

C. Encrypted Data Deduplication

User U_k tries to upload same data at CSSP. Following are the detailed steps followed during data deduplication phase:

Step1: CSSP verifies if the duplicated data exists by comparing X_j with stored X's. If the check results in NO, it uploads data afresh. User U_j encrypts data $Ptext_j$ with $SymmKey_j$ and encrypt SymmKeyj with PK_{AP} and sends it to CSSP to store them along with Xj and PK_j. Check result of YES shows that the previously stored data belongs to the same user, and informs that user about this situation. If data replica

belongs to a different user, go to Step 3 for further action.

Step 2: *Duplication check:* Naïve User U_k at a later stage wish to store similar data Ptext at CSSP after processing Step 1 That is, U_k delivers $\{X_k; PK_k\}$ to CSSP. Duplication due to X_k existence, so CSSP issues $\{X_k; PK_k\}$ to AP.

Step 3: *Challenge Ownership:* AP questions the ownership of U_k by arbitrarily selecting C, C€R (0; . . . ; 2-1) and passes it to U_k. U_k ensures whether $0 \leq C \leq 2\text{-}1$ and calculate $Y = HF(Ptext)+(S_k \times C)$ and delivers $Enc(PK_{AP}, Y)$ and Vi to AP. AP again calculates $HF(YP+ cV_k)$ and match it with X_k. If challenge proof is positive, i.e. $HF(yP+ cV_k)= X_k$, the ownership challenge is successful, AP issues reencryption key $ReEncKey_{AP \rightarrow Uk}$ and sends it to CSSP.

Step 4: *Deduplication:* CSSP re-encrypts $EncPRE(PK_{AP}$;$SymmKey_j)$ by calling $PREReEnc(RK_{AP \rightarrow Uk}; EncPRE(PK_{AP}; SymmKey_j))= EncPRE(PK_k; SymmKey_j)$ and issues the re-encrypted key $EncPRE(PKj; SymmKey_j)$ to U_k. Then U_k can obtain $SymmKey_j$ by using its secret key SK_k. U_k informs the CSSP about the success of ownership verification, which in turns keep a record of corresponding naïve user's statistics in the index pertaining to that file. Now both U_j and U_k can access the same data stored at CSSP. The difference lies is that the User U_j uses $SymmKey_j$ directly, while U_k can get $SymmKey_j$ by calling $DecPRE(SK_k; EncPRE(PK_k; SymmKey_j))$.

D. Data Dynamic operations at CSSP

1) Data Deletion

When data user U_j wishes to delete data from the repository, it performs steps as follows:

Step1: U_j sends deletion request of X_j to CSSP.

Step2: CSSP after verifying whether user U_j is entitled to this request or not, deletes deduplication record belongs to U_j from the index.

Step3: CSSP also blocks U_j's future access to data.

Step4: CSSP also verifies if the deduplication record is vacant or not. If yes, it removes encrypted data together with the associated index.

2) Data Update

A data owner U_i can also request for data modification.

Step1: User U_i sends an update request: $\{X_i$; updated ciphertext'; updated cipherkey'; update $Ctext_i\}$.

Step2: CSSP saves updated ciphertext and updated cipher key together with X_i and PK_i.

Step3: CSSP communicates with AP deduplication for existing naïve user's if their re-encryption keys are not acknowledged. AP examines its policy for issuing re-encryption keys consumed by CSSP to accomplish re-encryption.

Step4: These re-encrypted keys were distributed to all authorized naïve users for further data access.

E. Data Owner Management

In scenarios where data owner U_i stores data after naïve user U_j, CSSP can administer data storage by data owner by challenging ownership and allow storage sharing. CSSP communicates with AP by issuing all naïve user's PKi's. AP provides $ReEncKey_{AP \rightarrow Ui}$ to CSSP if ownership challenge is positive. CSSP re-encrypts $CipherKey_i$ generates re-encrypted $SymmKey_i$, and distribute it to all authorized naïve user's. It removes $Ciphertext_j$ and $CipherKey_j$ and replaces it with U_i's version of $Ciphertext_i$ and $CipherKey_i$. CSSP further updates associated index records pertaining to stored data.

VI. Experimental Validation of the Proposed Algorithm

This section presents the experimental setup used in order to implement the proposed algorithm. Table I shows the hardware and software specifications of the system used. For experiment purpose, distinct data files of different file sizes were taken as datasets. The proposed algorithm is implemented in Java NetBeans IDE 8.2.

TABLE I. System Specification

System Specifications	
Hardware	Intel® Core™ i7-3537U CPU@ 2.00 GHz 2.50GHz
Memory	4.00 GB RAM
Operating System	Ubuntu v14.04, Windows 8.1
Programming Environment	NetBeans IDE 8.2, Java
Library Used	Pairing-Based Cryptography (http://crypto.stanford.edu/pbc/) Java cryptography API JCA
Database	MySQL v5.5.41

VII. Security and Performance Analysis

This section deals with the performance analysis of the proposed system. For this purpose, a number of experiments are conducted to evaluate performance of each major steps involved in the proposed algorithm.

A. Data Encryption or Decryption

In order to judge the efficiency of data encryption and decryption by AES256 and DES, different datasets of sizes 64KB to 100MB were selected. An average of ten attempts of execution is recorded and tabulated in Fig. 2 and Fig. 3 shows key generation time for symmetric encryption.

For selected datasets, AES or DES takes less than 32ms for encryption or decryption and less than 320ms for key generation. As the file size increases, the operation time of AES and DES also increases, which is unavoidable in any system. In existing papers, only time for encryption and decryption is taken into consideration, while much time is involved in key generation. Hence total time for encryption/decryption should be time involved for SymmKey generation and encryption/decryption.

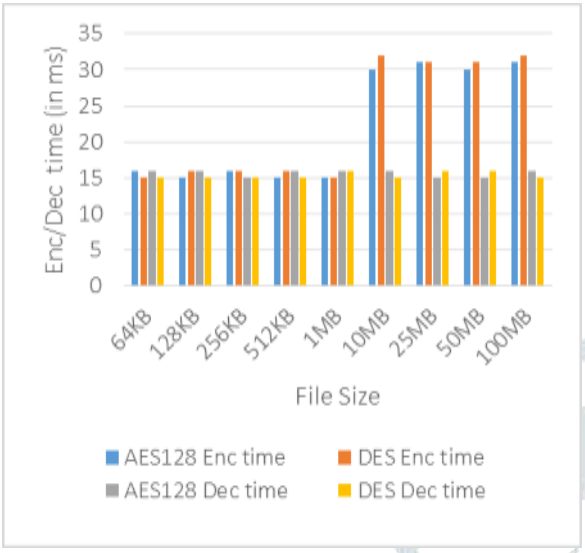

Fig. 2. The efficiency of Encryption/Decryption

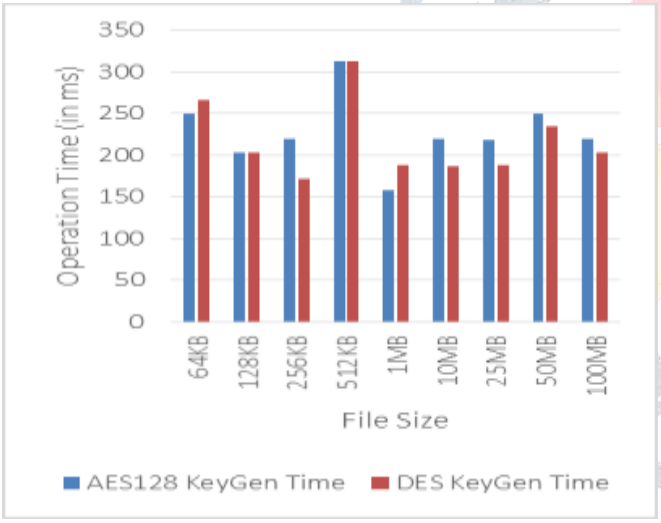

Fig. 3. Key Generation time for encryption or decryption.

B. Data Deduplication

For the purpose of evaluation of the deduplication ratio effect, two distinct test file sets having 10 files of 512KB size were selected. The first set is uploaded one by one and for uploading the second set, a portion of 10 files was taken, according to the given deduplication ratio, from the first set as duplicate files and remaining files from the second set as unique files. The average time of storing the files from the second set is presented in Figure 4. In case of replicas, uploading and encryption would be skipped. The time consumed during both these operations decreases with increase in deduplication ratio. The time spent on duplicate check also minimizes as the searching would be stopped if the first replica is found. A little time is spent on maintaining an index, which stores information related to file's unique ID, its version (as for version changes if the file is modified) and user's specific access permissions. As for duplication ratio increases, relatively time for index maintenance also increases, which is minimal and is inevitable in any system, hence can be neglected.

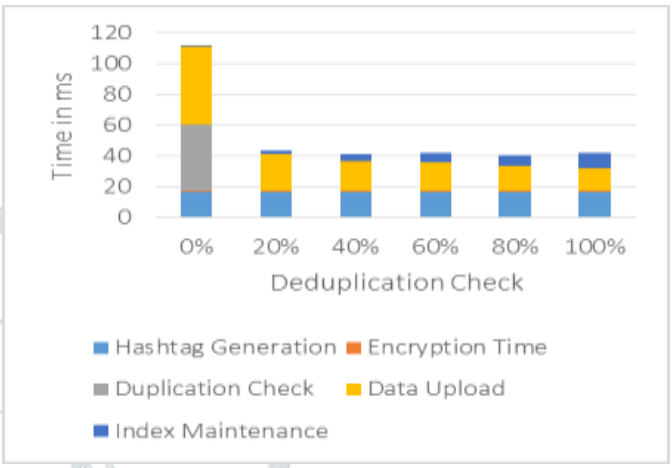

Fig. 4. Effect of Deduplication Ratio

C. One-Way Hash Function

Efficiency of proposed one way hash function can be judged from [3], [4] and [27]. The proposed algorithm shows results similar to standard algorithm of MD5, SHA160 and is better than SHA512. Moreover, due to involvement of concept of elliptic curve, hashtag can't be obtained by any malicious user. The hashtag is never stored at CSSP or AP site, only data token is stored at CSSP site. It brings about storage efficiency in the system.

D. PRE Algorithm

In order to test the efficiency of 1024bit PRE by using AES algorithm, the proposed algorithm is executed 10 times by using data set of 512K. The average operation time of PRE key generation, PRE encryption, PRE Re-encryption, and PRE decryption are calculated and shown in Fig.5.

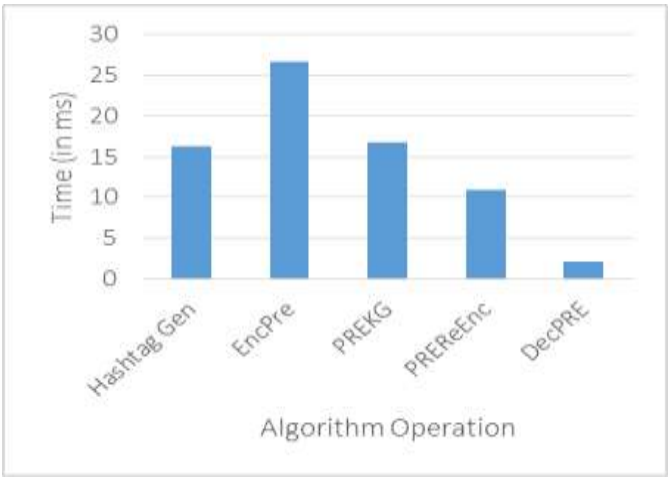

Fig. 5. Efficiency of PRE algorithm

PRE associated data deduplication operations are not affected by the size of datasets. This shows that the proposed scheme is efficient with respect to different sizes of data. In spite of the fact that proposed hash function takes slightly added execution time as compared to [5] but it brings out other advantages as specified in the previous section.

E. Data Ownership Challenge Verification

Mainly naïve user and AP were involved in the ownership challenge verification operation. For ownership verification, the 192-bit field of an elliptic curve, AES256, 1024-bit PRE and 512KB of the dataset is selected. Operational time of ownership response given by the naïve user and challenge verification performed by AP is tabulated in Table II.

TABLE II. DATA OWNERSHIP CHALLENGE VERIFICATION

Ownership Challenge Verification Operation	Time (in ms)
challenge response	28.63
challenge verification	0.3

As challenge response is given by the user, it depends upon the computation facilities available at user's site. While AP is responsible for challenge verification, having rich computational amenities, consumes less time as compared to [5].

It has been noted that most time-consuming operation of the proposed algorithm is data uploading, but it is inescapable in every security system. However, the data ownership challenge verification step of the proposed scheme is very efficient.

VIII. CONCLUSION

This paper has presented the algorithm for handling of encrypted data with deduplication by using PRE and ownership verification in Cloud Computing paradigm. It has been proved that the proposed integrity check algorithm's statistical tests show efficient results similar to MD5 and SHA160 and better than SHA512. Experimental and simulation results show the efficient performance of PRE and deduplication by introducing minimal computational or communication overhead. For replicas of data, a single copy is stored and is encrypted only once, resulting in reduced computational cost. In addition, it has been demonstrated that data ownership challenge verification phase is a very lightweight program that bears no added load on the system.

REFERENCES

[1] Verma Amandeep et al.,"Cloud Computing Security Issues and Challenges: A Survey", Springer-Verlag Berlin Heidelberg, acc 2011, Part IV, CCIS 193, pp. 445–454, 2011.

[2] Li jin et al.," A Hybrid Cloud Approach for Secure Authorized Deduplication", IEEE Transactions On Parallel and Distributed Systems, 2014.

[3] Kumari Meena Et Al., "A Secure And Flexible One Way Hash Function For Data Integrity Verification In Cloud Computing Environment", 3rd International Conference On Next Generation Computing Technologies, Oct 2017.

[4] Kumari Meena et al., "A Secure One Way Hash For Integrity Verification In Cloud Computing Environment", International Journal of Advanced Studies in Computer Science And Engineering, Volume 6 No 11, Nov 2017.

[5] Yan Zheng et al.,"Deduplication On Encrypted Big Data In Cloud", IEEE Transactions On Big Data, Vol. 2, No. 2, pp. 138-150, 2016.

[6] John R Douceur et al.," Reclaiming Space From Duplicate Files In A Serverless Distributed File System", 22nd International Conference On Distributed Computing Systems, pp 617–624, 2002.

[7] Mihir Bellar et al., "Message-Locked Encryption And Secure Deduplication", Advances In Cryptology–Eurocrypt, pp 296–312, 2013.

[8] D. Perttula et al., "Attacks On Convergent Encryption.", 2016. [Online]. Available: Http://Bit.Ly/Yqxyvl

[9] Puzio Pasquale et al.," Cloudedup: Secure Deduplication With Encrypted Data For Cloud Storage", **IEEE** 5th International Conference on Cloud Computing Technology and Science (Cloudcom), 2013.

[10] Yang Chao et al., "Provable Ownership Of File In De-Duplication Cloud Storage", Communication And Information System Security Symposium, pp 695-700, 2013.

[11] Yu Chia-Mu Et Al., "Privacy-Aware Data Deduplication for Side-Channel In Cloud Storage", Journal Of Latex Class Files, Vol. 14, No. 8, pp 1-13, August 2015.

[12] G. Ateniese et al.," Improved Proxy Re-Encryption Schemes With Applications To Secure Distributed Storage", ACM Transactions on Information And System Security, Vol. 9, No. 1, February 2006.

[13] Z. Sun et al., "Dedu: Building A Deduplication Storage System Over Cloud Computing," IEEE International Conference on Computer Supported Cooperative Work Des., 2011, pp. 348–355.

[14] Z. C. Wen et al., "A Verifiable Data Deduplication Scheme in Cloud Computing", International Conference on Intelligent Networking Collaborative Systems, 2014, pp. 85–90.

[15] P. Meye et al., "A Secure Two phase Data Deduplication Scheme", Proc. Hpcc/CSS/Access, 2014, pp. 802–809.

[16] Abutaha Mohammed et al.," New One Way Hash Algorithm Using Non-Invertible Matrix", In Proc. Of International Conference on Computer Medical Applications (Iccma), IEEE, 2013.

[17] Kavuri Satheesh et al.," Data Authentication and Integrity Verification Techniques for Trusted/Untrusted Cloud Servers", In Proc. Of International Conference On Advances In Computing, Communications And Informatics (Icacci), pp 2590 - 2596, 24-27 Sept. 2014.

[18] Acharya Bibhudendra et al.," Novel Methods Of Generating Self-Invertible Matrix For Hill Cipher Algorithm", International Journal Of Security, Volume 1, Issue 1, 2007, pp 14-21.

[19] Z. O. Wilcox, "Convergent Encryption Reconsidered," 2011. [Online]. Available: Http://Www.Mailarchive.Com /Cryptography@Metzdowd.Com/Msg08949.Html.

[20] Abutaha Mohammed et al.," A Practical One Way Hash Algorithm Based On Matrix Multiplication", International Journal of Computer Applications, Volume 23– No.2, Pp 34-38, June 2011.

[21] S.G.Srikantaswamy et al., "Hash Function Design Using Matrix Multiplication, Ex-Or, Checksum Generation and Compression Technique Approach", International Journal Of Computer Science And Information Technology & Security, Vol. 3, No.1, February 2013.

[22] Hamamreh A. Rushdi et al.,"Design Of A Robust Cryptosystem Algorithm For Non-Invertible Matrices Based On Hill Cipher", Ijcsns International Journal Of Computer Science And Network Security, Vol.9 No.5, May 2009.

[23] Kumar N Gowthamet al.," Hash-Based Approach For Providing Privacy And Integrity In Cloud Data Storage Using Digital Signatures", International Journal Of Computer Science And Information Technologies, Vol. 5 (6), Pp 8074-8078, 2014.

[24] M. Jogdand Rashmi et al., "Enabling Public Verifiability and Availability for Secure Data Storage in Cloud Computing", Springer Evolving Systems, 26th Sep 2013.

[25] Li Yantao et al., "Parallel Hash Function Construction Based On Chaotic Maps with Changeable Parameters", Neural Computing and Applications, Springer, 2011.

[26] C. Fan et al., "Hybrid Data Deduplication in Cloud Environment", International Conference on Information Security Intelligent Control, 2012, pp. 174–177.

[27] Kumari Meena et al., "Analytical Performance of Modified One-Way Hash Algorithm for Data Integrity Verification in Cloud Computing.", International Journal of Green Computing 9.2 (2018): 16-26. Web. 12 Mar. 2019.