This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSUSC.2019.2914954, IEEE Transactions on Sustainable Computing

A Fog-centric Secure Cloud Storage Scheme

M A Manazir Ahsan, *Student Member, IEEE*, Ihsan Ali, *Student Member, IEEE*, Muhammad Imran, Mohd Yamani Idna Bin Idris, Suleman Khan, Anwar Khan

Abstract— Cloud computing is now being utilized as a prospective alternative for catering storage service. Security issues of cloud storage are a potential deterrent in its widespread adoption. Privacy breach, malicious modification and data loss are emerging cyber threats against cloud storage. Recently, a fog server based three-layer architecture has been presented for secure storage employing multiple clouds. The underlying techniques used are Hash-Solomon code and customized hash algorithm in order to attain the goal. However, it resulted in loss of smaller portion of data to cloud servers and failed to provide better modification detection and data recoverability. This paper proposes a novel fog-centric secure cloud storage scheme to protect data against unauthorized access, modification, and destruction. To prevent illegitimate access, the proposed scheme employs a new technique $Xor - Combination$ to conceal data. Moreover, $Block - Management$ outsources the outcomes of $Xor - Combination$ to prevent malicious retrieval and to ensure better recoverability in case of data loss. Simultaneously, we propose a technique based on hash algorithm in order to facilitate modification detection with higher probability. We demonstrate robustness of the proposed scheme through security analysis. Experimental results validate performance supremacy of the proposed scheme compared to contemporary solutions in terms of data processing time.

Index Terms—Cloud storage, fog server, Xor-Combination, CRH, privacy

1 Introduction

CLOUD computing, a prominent computing paradigm was introduced in SES 2006 (Search Engine Strategies 2006) and was formally defined by NIST (National Institute of Standards and Technology) [1] in 2009. Since then, this technique has resulted in attracting increased market share with its powerful computing, storage and communication facilities [2, 3]. Its infrastructure resources are not only scalable on demand but also available at an economical price following convenient payment policy, pay as you go. Along with individual and enterprise customers, cloud computing also draws attention of many research communities who exert massive efforts towards its gradual maturity. Hence, cloud computing has many functionalities and cloud storage technique is becoming increasingly important for growing volume of data.

Volume of user's data is rising exponentially with the rise of network bandwidth [4]. Almost every internet user has his/her own cloud storage ranging from GB's to TB's. Local storage fails to fulfil this immense storage requirement alone. Most importantly, people have inherent need for ubiquitous access to their data. Consequently, people are finding new mediums to store their data. Giving preference to powerful storage capacity, a growing number of users have switched to cloud storage; they even prefer to save their private data to the cloud. Storing data on a commercial public cloud server will be a prevalent trend in the near future. Getting inspired by the fact, many organizations, such as Dropbox, Google Drive, iCloud and Baidu cloud are providing a variety of storage services to their users. However, advantages of cloud storage are accompanied with a set of cyber threats [5-8]. Privacy issue is one of the major threats in addition to loss of data, malicious modification, server crash are some examples of cyber threats. There are some prominent cyber incidents in the history, for example, Yahoo's three billion accounts exposure by hackers in 2013, Apple's iCloud leakage in 2014, Dropbox data privacy breach in 2016, particularly iCloud's leakage event, where numerous Hollywood actresses' private photos were exposed and caused massive outcry. Such incidents affect company's reputation fervently [9-11].

In traditional cloud computing scenario, once users outsource their data to the cloud, they can no longer protect it physically. Cloud Service Provider (CSP) can access, search or modify their data stored in the cloud storage. At the same time, the CSP may loss the data unintentionally due to some technical faults. Alternatingly, a hacker can violate the privacy of the user data. Using some cryptographic mechanisms (such as encryption, hash chain), confidentiality or integrity can be protected [12]. However, cryptographic approach cannot prevent internal attacks, no matter how much the algorithm improves [13]. To protect data confidentiality, integrity and availability (CIA), several research communities introduced the idea of Fog Computing placing fog devices in between the user and the cloud

- *M A Manazir Ahsan, Ihsan Ali, Mohd Yamani Idna Bin Idris are with the Faculty of Computer Science and Information Technology, University of Malaya, 50603 Kuala Lumpur, Malaysia.* E-mail: manazir.ahsan@csebuet.org, ihsanalichd@siswa.um.edu.my, yamani@um.edu.my
- *Muhammad Imran is with the College of Computer and Information Sciences, King Saud University, Saudi Arabia. E-mail: dr.m.imran@ieee.org.*
- Suleman Khan is with the School of Information Technology, Monash University, 47500 Selangor, Malaysia. E-mail: suleman.khan@monash.edu.
- *Anwar Khan is with the Department of Electronics, Quaid-i-Azam University, Islamabad 44000, Pakistan. E-mail: arkhan@upesh.edu.pk.*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSUSC.2019.2914954, IEEE Transactions on Sustainable Computing

server. One of the prominent and recent works in this field is proposed by Wang et al. They utilized *Reed Solomon code* and hash digest centric customized algorithms to preserve confidentiality and integrity of the data respectively [12]. They also formulated the computational intelligence (CI) to determine the portion of data to be stored in cloud, fog and user's local machine. They maintained a rating system for cloud server so that user can rate the cloud servers and the cloud servers tend to act responsively. Nonetheless, this scheme reveals that some portion of data (not the entire data) to the cloud and their customized hash algorithm, despite taking extra computation/storage overhead, adds no value over standard hash algorithm (i.e. MD5) in terms of collision resistance. In this paper, we propose a fog-based cloud storage scheme for data confidentiality, integrity and availability. For confidentiality and availability (even after malicious events), we propose a method referred to as $Xor-Combination$ that splits the data into several blocks, combine multiple blocks using Xor operation and outsource the resulted blocks to different cloud/fog servers. In order to prevent any individual cloud server to retrieve a portion of original data, the proposed technique $Block-Management$ selects the cloud server to store each particular data blocks. $Xor-Combination$ along with $Block-Management$ helps to protect data and to retrieve data from multiple sources even when some blocks are missing. At the same time, we propose a noble hashing mechanism titled as $Collision\ Resolving\ Hashing\ (CRH)$ operation based on traditional hash algorithm (i.e., SHA256, MD5) that withstands collision in hashing [14] and security features. The proposed scheme thrives to be a robust solution for efficient and secure cloud storage. The main contributions of the paper can be summarized as follows:

- We proposed a secure cloud storage scheme based on fog computing employing $Xor-Combination$, $Block-Management$ and CRH operation. $Xor-Combination$ together with $Block-Management$ contributes to maintain privacy and to prevent data loss. CRH operation ensures detection of data modification.
- Theoretical security analysis proves the privacy guarantee, data recoverability, and modification detection of the proposed scheme.
- We implemented a prototype version of the scheme and conducted experiments to verify its performance in comparison with the contemporary scheme. Results prove its efficiency in terms of time and memory usage.

Organization: Rest of the paper is organized as follows: Section 2 discusses some related works and section 3 defines the system model, threat model and design goal. Section 4 presents the proposed scheme in details. Section 5 analyses theoretical security/privacy, recoverability, modification detection of the proposed scheme. Experiment and performance analysis are illustrated in section 6. Section 7 concludes the paper with discussing the result and some future research directions.

2 Related Work

Importance of cloud storage draws attention of researchers from both academia and industry. Improving the performance of the cloud storage as well as maintaining the security level are the main research domains. Security issues are always the focus of research in order to enhance credibility of the storage mechanisms [15]. A range of survey papers [16-19] indicated that privacy breaches, malicious modification (or integrity violation), data loss are the main cyber threats of cloud storage. Kaufman argued that, to cope with the aforementioned experienced security threats, cloud servers have to establish coherent and effectual policy [20]. Zissis et al. evaluated cloud security by identifying unique security requirements and presented a conceptual solution using trusted third party (TTP). As underlying cryptographic tool they used public key cryptography to ensure confidentiality, integrity and authenticity of data and communication while addressing specific vulnerabilities [21]. Wang et al. focused on integrity protection on cloud computing and proposed public auditability scheme as a counter measure [22]. They set two goals of their work, one was the efficient public auditing without requiring local copy of data and the other one was not to cause any vulnerability of the data. They utilized homomorphic authenticator with random masking for privacy preserving public auditing of cloud data. However, public key centric homomorphic authenticator caused computational burden and this work did not focus on partial/entire data loss. An efficient public auditing protocol using sampling block-less verification was proposed in [23]. At the core of their proposed protocol there was a noble dynamic data structure which consisted of doubly linked info table and a location array. This structure reduces the computation/communication cost substantially. Conversely, like previous scheme, it does not address cyber threats other than integrity checks.

Xia et al. proposed a mechanism titled Content Based Image Retrieval (CBIR) to protect image outsourced to cloud server relying on locality sensitive hashing (LSH) and secure k-nearest-neighbors (kNN) algorithms [24]. It is equally applicable to other data types (i.e., text) as well. It preserves privacy of sensitive images and ensures efficient retrieval but does not guarantee integrity or elimination of an image (or other type of data). Arora et al. enlisted and compared some cryptographic primitives

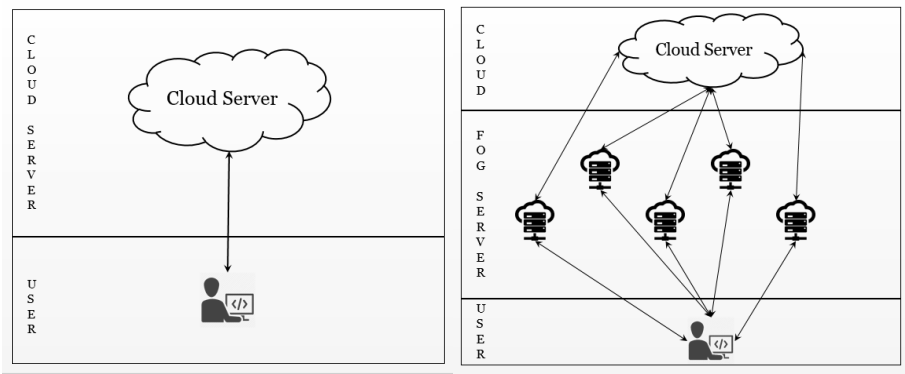
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSUSC.2019.2914954, IEEE Transactions on Sustainable Computing

for preservation of privacy and integrity of cloud storage [25]. This comparison is also befitting for other computing architecture. One recent work reported by Shen et al. used cloud infrastructure for urbanization. Their proposal illustrated cloud to share data between urban people and/or applications [26]. To protect privacy of shared data they used attributed based encryption (ABT). However, they concentrated on the privacy of data and relied on cloud server for integrity and data loss prevention. Precisely, Khan et al. emphasized trust in cloud computing. The challenges of trust in cloud and how a cloud server can achieve trust of its customer have been discussed in their article [27]. The previous researches discussed so far are most commonly related to integrity preservation by various public/private auditing frameworks. Privacy, on the other hand, is best protected by encryption, though the encryption techniques making the searching operation difficult. Hence, different searchable encryption schemes came into existence related to searching on encrypted cloud data [28-30].

In contrast with different third party auditor (TTA) based solutions, fog server centric solutions have upper hand in terms of preventing cyber threats. For example, TTA solutions are good for finding malicious modification but they do little for privacy preservation. Similarly, searchable encryption related solutions work well for preserving privacy and (comparatively) efficient retrieval of data but they have issues like data loss and modification. Conversely, fog server which is an extension of cloud server at proximity of the user can provide a prospective solution to fight against various cyber threats. However, fog based solutions to combat cyber threats are yet to be explored in detail. Tian et al. proposed a new scheme of cloud storage resorting to fog server in order to protect against different attacks [13]. They adopted three-layered architecture, kept the fog server in between the cloud server and the users. Considering fog server being trusted by the user, they presented a noble scheme for privacy preservation, modification detection, and data loss prevention. They encode the data utilizing Reed-Solomon code and deduce Computation Intelligence (CI) to determine the amount of data to be outsourced to cloud/fog servers so that no individual cloud server can reconstruct the data. However, fraction of data gets exposed to each outsourcing cloud server. On the other hand, they formulate Malicious Modification Detection (MMD) to detect malicious modification that has no advantage over traditional hashing algorithms to detect malicious modification. Another recent work proposed [12] also undertook the similar work with same architecture. These papers recommend the use of fog based solutions for secure cloud storage and more importantly, to protect against the cyber threats, those directed towards cloud data. In this paper, the authors propose a secure cloud storage scheme on the basis of fog server considering Tian et al.'s scheme as the benchmark.

3 Problem Formulation

This section formally defines the problem. System model, threat model and design goal of the paper have been described in detail. Additionally, a short description of fog computing and the notations used in this paper have been presented.



a. Traditional Cloud b. Fog-based Cloud

Fig. 1. Comparative computing architecture

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSUSC.2019.2914954, IEEE Transactions on Sustainable Computing

3.1 System Model

In traditional cloud computing realm, cloud server supports the user with computation, storage, and networking facilities. In this scenario, user directly uploads their data either for safe keeping or for processing with scalable computing/storage resources which is illustrated in Fig. 1(a). However, outsourcing data to the cloud may breach the privacy of the data. Furthermore, there are situations where large amount of data gets accumulated from a particular location and is processed in real time to generate some result [31, 32]. Sending data to a centralized infrastructure (i.e. cloud server) may cause transmission delays. Fog computing can resolve this issue. Fog computing is smaller version of cloud computing that is placed between cloud server and the user. Fig. 1(b) shows the scenario of fog-based cloud storage system. As user needs a trustworthy storage to save data, the proposed scheme considers an architecture where user has full control over fog devices. Users can rely on fog computing/storage devices for the management of their data. Fog computing devices further communicate with multiple clouds for advanced storage requirements. Besides, long-thick channel between cloud-fog and short-thin channel between fog-user contributes to resolve the communication issue (i.e. transmission delay) [33]. As demonstrated in Fig. 1(b), user uploads the data to the fog devices, fog device utilizes the techniques of proposed scheme to split the data into different blocks and send the different blocks to different cloud servers. Fog server can store several blocks of data to its own storage system. While retrieving data, user requests the fog server and the fog server brings corresponding blocks from cloud server, combines to form the requested data and send it back to the user. Here, proposed scheme employs different techniques for privacy preservation, data loss detection and disaster recoverability.

There are three entities in our system model: user, fog server, cloud server. Trust level of these entities is different. Like prior schemes [12, 13], we consider following reliability of the entities:

User: User is the owner of data. Privacy, disaster recoverability, modification detection of user's data is ultimate goal of this paper.

Fog Server: Fog server is trusted to user. User relies on fog server with his data. Close proximity of fog devices to the user, robust physical security, proper authentication, secure communication, intrusion detection ensures fog server's reliability to the user.

Cloud Server: Cloud server is considered as *honest but curious* [34, 35]. This means that cloud server follows the Service Level Agreement (SLA) properly, but has an intention to analyse user's data. Conversely, cloud server may pretend to be good but acts as a potential adversary. In that case, cloud server may modify data in order to forge as original data. Similarly, cloud server may hide/loss the data resulting in permanent data loss of the user. Furthermore, hardware/software failure may result in data modification or permanent loss as well.

3.2 Threat Model

There are different kinds of cyber threats in cloud storage. Aligning with cryptographic features of data protection i.e. confidentiality, integrity and availability (commonly referred to as CIA-triad), cyber threats can be categorised into three broad categories: *privacy breaches*, *malicious modification* and *data loss* respectively. Once data is entered into the cloud server, user cannot protect it any more. Apart from saving data in cloud storage, cloud computation requires data as data is an essential part of computation. Thus, whenever cloud server gets data, the privacy of data can be jeopardized by internal employees of the cloud. Alternatively, an external attacker can attack cloud server to violate the privacy of the user data. In either cases, privacy of data is susceptible to internal and/or external attackers. Over the past years, there have been several incidents of privacy violation from renowned cloud servers. On the other hand, inside and/or outside attacker can modify sensitive data purposefully. It may lead to false data to appear as right. Lastly, cloud server can hide data intentionally which may result in permanent data loss of the user. Alternately, cloud storage may crash causing data loss from the server. The proposed scheme attempts to combat the above mentioned three threats of cloud storage using fog based techniques.

3.3 Design Goal

Design goal of this paper comes from the cyber threats described in the *Threat Model* (sub-section 3.2). Goal of the proposed scheme is to prevent privacy breaches, malicious data modification and data loss. This sub-section enlists design goals of the proposed scheme and later in *Security Analysis* section, the authors analyse if the proposed scheme achieves the target goals or not. The goals are listed below:

A Privacy Preservation

Sensitive data outsourced to the cloud is susceptible to the inside or outside attacker. Hence, information leakage takes place. Encryption can protect such leakage. Encryption transform data into random text and no operation can be performed upon encrypted text. Therefore, encryption resolves the privacy of stationary data but cannot protect privacy of computing data. Some researchers proposed homomorphic encryption but homomorphic encryption is computationally infeasible [36]. Besides, sometimes encryption is subject to cracking. Therefore, privacy preservation is the foremost important issue in cloud storage while it stores sensitive data.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSUSC.2019.2914954, IEEE Transactions on Sustainable Computing

B Data Recoverability

Data recoverability is another important goal that should receive attention from the research community. Though, cloud server is supposed to maintain redundancy for disaster recoverability, but users can take pre-emptive steps to recover data of paramount importance. Cloud server can conceal data purposefully or data can be lost permanently due to hardware/software failure. Hence, data should be recoverable in case of any malicious or catastrophic data loss.

C Modification Detection

Malignant cloud server may modify the actual data and forge it to appear as original. User, without any modification detection technique, is susceptible to take wrong decisions based on fabricated data. Consequently, modification detection is another essential feature that fog based cloud storage scheme should achieve.

Before describing the proposed scheme, we illustrate fog computing in the following sub-section.

3.4 Fog Computing

Cloud computing is a sophisticated technology used to cater computing, storage and communication service over the internet with flexibility and efficiency. Nonetheless, there are cases where massive amount of data spanning in a large geographical area needs to be stored, processed and analysed efficiently. Furthermore, privacy guarantee of the collected/processed data is sometimes critically important. To fulfil the gap, the concept of fog computing emerged which is able to extend cloud computing in more proximity to the user that it serves [37]. In contrast with cloud, fog computing can provide computing and storage facilities at the edge of the network, hence, it is alternatively called edge computing.

A fog computing node can be any network device with the capability of storage, computing, and network connectivity (routers, switches, video surveillance cameras, servers, etc.) [38]. Security and privacy issues in fog computing infrastructures prevail [39-41]. The security and privacy issues can be mitigated by counter technologies mentioned such as proper authentication, access control, secure channel, intrusion detection, trust management. While all these techniques are in place, fog computing can be considered as a trusted device upon which the users can rely for processing, storing and managing data. This paper presents fog computing as a trusted device. On the basis of fog computing, the present research proposes a secure cloud storage scheme.

3.5 Notations

This subsection explains the notations used in this paper.

L: length of fixed-length data block. A data is splitted into one or more fixed length blocks.

$\boldsymbol{B_i}$**:** i^{th} Data block.

$\boldsymbol{C_{i,j}}$**:** $Xor - combination$of blocks B_i, B_j. That is, $C_{i,j} = B_i \oplus B_j$. Alternatingly, it refers to as 2-blocks-combination.

$\boldsymbol{C_{i,j,k}}$**:** Xor combination of blocksB_i, B_j, B_k. That is, $C_{i,j,k} = B_i \oplus B_j \oplus B_k$.Alternatively, it is 3-blocks-combination.

$\boldsymbol{Combined\ block}$: $\boldsymbol{C_{i,j,k}}$& $\boldsymbol{C_{i,j}}$ are commonly known as $Combined\ block$.

$< i, \boldsymbol{j}, \boldsymbol{B_i} \oplus \boldsymbol{B_j} >$**:** Tuple with two blocksB_i, B_j.

$< i, \boldsymbol{j}, \boldsymbol{k}, \boldsymbol{B_i} \oplus \boldsymbol{B_j} \oplus \boldsymbol{B_k} >$**:** Tuple with three blocksB_i, B_j, B_k.

$\boldsymbol{hash}(\boldsymbol{message})$**:** Hash digest of the text message.

$\boldsymbol{message_1}$ $||\boldsymbol{message_2}$**:** concatenation of text: $message_1$ and $message_2$.

$|$**message**$|$**:** size of text message.

4 Fog based Secure Cloud Storage

Security is one of the essential features of cloud computing. Furthermore, security of data is the prime need for cloud storage and it comprises of data privacy, integrity and availability. To enhance the credibility of cloud, data security has always been the focus of research of relevant research community [42]. At the same time, users are more concerned with the security of the outsourced data to the cloud. Hence, cloud with more security degree will attract more clients. Therefore, both research and business communities are testing the boundaries of cloud security. At the centre of cloud data security, there are three aspects: confidentiality, integrity and availability. We address these issues of cloud storage using fog based scheme referring the three issues as privacy protection, modification detection and recoverability respectively. In this section, we will elaborate how the proposed scheme works to preserve privacy, detect malicious modification and to ensure recoverability.

4.1 Proposed Scheme

With a view to protecting cloud data, we proposed secure cloud data storage scheme based on fog computing. In the proposed scheme, fog server, furnished with some computing, storage and communication capabilities, is assumed to be reliable for the user. Reliable fog computing can be implemented by employing proper authentication, access control and intrusion detection mechanism. Close proximity of fog device to the user enhances its credibility as a secure computing infrastructure. Apart from the fog computing, proposed scheme utilizes its own techniques$Xor - Combination$, $Block - Management$ and

$Collision\ Resisting\ Hashing\ (CRH)$ to preserve privacy, ensure recoverability and to detect data modification for the data deployed in the cloud storage. By utilizing cryptographic techniques, cloud storage can combat external attacks. However, data is more vulnerable to inside attackers, specially, when cloud server itself turns

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSUSC.2019.2914954, IEEE Transactions on Sustainable Computing

into malicious adversary. The proposed scheme intends to counteract the inside attacks considering fog device as a reliable computing base and outsourcing smaller parts of data to different cloud servers. Previous schemes [12, 13] outsourced smaller parts of the data to different clouds in plain text format, making it susceptible to certain information leakage. Moreover, these schemes cannot withstand actual collision of hash functions (if any). Conversely, we split data into smaller blocks using *Xor-Combination* before outsourcing to multiple clouds. $Xor - Combination$ conceals original content and ensures full data recovery even in case of malignant (or non-malignant) data loss from some cloud servers. Besides, we propose a noble technique of CRH that intends to minimize collision of a hash function.

From the perspective of design, proposed scheme aims to protect privacy, ensure recoverability and to detect malicious modification. For this purpose, fog centric architecture resorts to three techniques:$Xor - Combination$, $Block - Management$ and CRH. Particularly, $Xor - Combination$ plays an important role to preserve privacy and to reconstruct data if many portions of it is missing. On the other hand, CRH contributes to detect any malicious modification of data even if the underlying hash function fails to resist collision [43]. Fig. 2 illustrates the steps of data processing before outsourcing data to the cloud's storage. Once a user has the data (i.e. a document or a file) for safe keeping into the cloud storage and sends it to his reliable fog server device, then the fog device processes the steps as follows:

- It pads the data if the data is not (exact) multiple of fixed length (L) block.
- Afterwards it executes *Xor-Combination* on the padded data which results in number of 2-block-combinations and n number of 3-block-combinations.
- After that, $Block - Management$ decides which blocks are to be preserved in which clouds and sends the blocks to corresponding clouds. Different Meta data (i.e. data number, block tag, ID, cloud number) is preserved into Table 1.
- Simultaneously, fog server executes CRH operation on each of the data blocks which produces$DataDigest$. It computes hash digest of a particular data block, generates a random number R, computes hash digest of the data concatenated with the random number R.

$$DataDigest = hash(data - block)$$
$$RandomDigest = hash(R \,||\, data - block)$$

- Preserve the Business ID (i.e., unique Data ID & Block Tag combination), $DataDigest$, random number (R), $RandomDigest$ into fog database as shown in Table 2.
- Finally, fog server outsources the different blocks to different cloud servers as shown in Table 1.

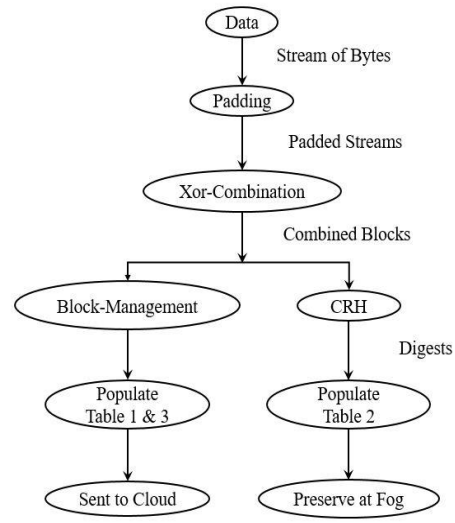


Fig. 2. Data processing flow

- Each cloud server receives the Business ID (to uniquely identify a data block) and block content and preserves it into its storage. It also maintains a record set as shown in Table 3.

Table 1. Data-Block-Cloud

Data / Document / File ID	Block Tag	Business ID	Cloud Server
…	…	…	…
…	…	…	…
…	…	…	…

Table 2. Hash-Digest-Table

Business ID	Hash Digest	Random Number	Random Digest
…	…	…	…
…	…	…	…
…	…	…	…

Table 3. Cloud-Server's-Table

Business ID	Block Content
…	…
…	…
…	…

For proper demonstration of the proposed scheme, we describe the inner mechanism of $Xor - Combination$, $Block - Management$ and CRH as follows.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSUSC.2019.2914954, IEEE Transactions on Sustainable Computing

4.1.1 Xor-Combination

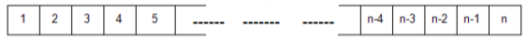


Fig. 3. Splitting data into fixed sized block

Xor-Combination is a noble approach used for privacy preservation and data loss recoverability simultaneously. It receives the padded data as an input and returns two sets of tuples as output where each tuple consists of a $block\ tag$ and fixed length (L) blocks. Comma separated block number is termed as $block\ tag$. Each set contains n = $\left\lceil \frac{|data|}{|L|} \right\rceil$ number of tuples. Upon receiving padded input, this algorithm splits it into $\left\lceil \frac{|data|}{|L|} \right\rceil$ numbers of data blocks with size L each such as $B_1, B_2, B_3, \ldots\ \ldots\ \ldots, B_n$ as shown in Fig. 3. Here, $|B_1| = |B_2| = |B_3| = \ldots = |B_n| = L$. Afterwards, it generates 2-block-combinations ($\boldsymbol{C_{i,j}}$) and 3-block-combinations ($\boldsymbol{C_{i,j,k}}$) with consecutive data blocks. Considering first data block comes after the last data block like round robin . Multiple data blocks combination is commonly termed as $combined\ block$. Firstly, it takes each pair of consecutive blocks, combines in a Xor operation i.e. $C_{i,(i\%n)+1} = B_i \oplus B_{(i\%n)+1}$ and constructs a tuple$< i, (i\%n) + 1,\ C_{i,(i\%n)+1} >$.Similarly, tuples with 3-block-combinations are generated. Finally, two sets of tuples $\{< 1,2, C_{1,2} >, < 2,3, C_{2,3} >, < 3,4, C_{3,4} >,\ \ldots\ \ldots\ \ldots,$ $< n, 1, C_{n,1} >\}$ and $\{< 1,2,3, C_{1,2,3} >, < 2,3,4, C_{2,3,4} >$ $, < 3,4,5, C_{3,4,5} >,\ \ldots\ \ldots\ \ldots, < n-1, n, 1, C_{n-1,n,1} >,$ $< n, 1,2, C_{n,1,2} >\}$ are returned as output of *Xor-Combination* algorithm. Algorithm 1 shows the total working procedure of Xor-Combination below:

Algorithm: Xor-Combination
Input: Data as block of bytes.
Output: Two sets of tuples.
Procedure:
Receive padded data as input;
$Set2 \leftarrow \Phi$; // initialize with null.
$Set3 \leftarrow \Phi$; // initialize with null.
$n \leftarrow \left\lceil \frac{|data|}{|L|} \right\rceil$;
Split the data into L length blocks i.e. $B_1,\ B_2,\ B_3, \ldots\ \ldots\ \ldots, B_n$;
For each $i \leftarrow 1$ to n do
$Set2\ \cup = < i, (i\%n) + 1,\ B_i \oplus B_{(i\%n)+1} >$;
$Set3\ \cup = < i, (i\%n) + 1,\ \big((i\%n) + 2\big)\%n, B_i \oplus B_{(i\%n)+1} \oplus B_{((i\%n)+2)\%n} >$;
End for;
Return $Set2$ and$Set3$;
End Procedure;

$Xor - Combination$, actually, is a series of code that splits and combines any number of consecutive blocks to facilitate privacy preservation and recoverability in case of data loss. In the above algorithm, we resort to 2-block-combinations and 3-block-combinations. In this case, this technique can be referred to as *2/3-Xor-Combination*. As per system needs, one can construct *3/4-Xor-Combination*, *2/3/4-Xor-Combination* and so on.

Finally, a close observation reveals that, each block can be retrieved using five different interactions between combined blocks in *2/3-Xor-Combination*. Such as, one can retrieve B_i as follows:
$B_i = C_{i,i+1,i+2} \oplus C_{i+1,i+2}$
$B_i = C_{i-1,i,i+1} \oplus C_{i-1,i} \oplus C_{i,i+1}$
$B_i = C_{i-2,i-1,i} \oplus C_{i-2,i-1}$
$B_i = C_{i-3,i-2,i-1} \oplus C_{i-3,i-2} \oplus C_{i-1,i}$
$B_i = C_{i+1,i+2,i+3} \oplus C_{i,i+1} \oplus C_{i+2,i+3}$

4.1.2 Collision Resisting Hashing (CRH)

$Collision\ Resisting\ Hashing$ is a proposed technique based on a standard hashing algorithm (i.e. MD5, SHA256) that successfully checks consistency even if there exists a collision. For example, hash digest of $OriginalText$ (i.e. $hash(OriginalText)$) is preserved in order to detect any malicious modification and $ModifiedText$ has the same hash digest as that of $OriginalText$ (i.e. $hash(OriginalText) = hash(ModifiedText)$). CRH is able to distinguish between

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSUSC.2019.2914954, IEEE Transactions on Sustainable Computing

$OriginalText$ and $ModifiedText$, despite having such collision.

CRH utilizes chaos of a hash function: small difference in initial state yields diverging outcomes for later state. First of all, it generates a random number R. Then it computes hash digests of $OriginalText$ and $OriginalText$ prepended with R, namely, $OriginalDigest$ and $RandomDigest$ respectively, (for example, $OriginalDigest = hash(OriginalText)$ and $RandomDigest = hash(R \,||\, OriginalText)$). Finally, it stores the random number R, $OriginalDigest$ and $RandomDigest$ into database as shown Table 3. While detecting any modification of $VarifiableText$, it computes $hash(VarifiableText)$ and $hash(R \,||\, VarifiableText)$ and cross checks with $OrigianlDigest$ and $RandomDigest$ stored in the database in a systematic manner. Equality in both cases indicates "no modification" with higher probability where any inequality indicates "malicious modification" with certainty.

In order to improve the probability of detection, one can increase the number of random numbers and random digest pairs. However, given the property of standard collision resistant hash function, we prefer to conduct with one pair of random numbers and random digest in our scheme.

4.1.3 Block Management

$Block - Management$ technique assists to figure out which blocks are to be stored in which cloud servers. It works on the *combined blocks* (i.e. 2-block-combinations and 3-block-combinations) generated from $Xor - Combination$. The block management along with $Xor - Combination$ has two goals to attain, one is privacy of the data and the other is recoverability of the data in case of data loss.

A closer observation of $Xor - Combination$ reveals that any 2-block-combination (or 3-block-combination) alone is not able to retrieve the original content of a block in plain text form. Therefore, storing all 2-block-combination together and all 3-block-combination together in two non-colluding cloud servers ensures the privacy of the data blocks and in turn preserves the privacy of the data.

Algorithm: CRH.processing
Input: Text
Output: Random number, two hash digests.
Procedure:
Receive Text, namely, $OriginalText$ as input;
Generate random number R;
Compute $OriginalDigest = hash(OriginalText)$;
Compute $RandomDigest = hash(R \,||\, OriginalText)$;
Store R, $OriginalDigest$ and $RandomDigest$ into database;
Return$R$, $OriginalDigest$ and $RandomDigest$;
End procedure;

Algorithm: CRH.verification
Input:$VerifiableText$
Output:true or false.
Procedure:
Retrieve corresponding R, $OriginalDigest$ and $RandomDigest$ from database.
Compute $VerifyDigest = hash(VerifiableText)$ and $RandomVerifyDigest = hash(R \,||\, VerifiablText)$;
If ($OriginalDigest == VerifiableDigest$ and $RandomDigest == VerifiableRandomDigest$) {
Return true;
} else {
Return false;
}
Return$R$, $OriginalDigest$ and $RandomDigest$;
End procedure;

From the above discussion, *CRH* technique has two different algorithms: *CRH.Processing* and *CRH.verification*. *CRH.processing* generates and stores the random number R, $OriginalDigest$ and the $RandomDigest$. *CRH.verification*, on the other hand, detects the malicious modification.

On the other hand, to ensure data recoverability, we need to retrieve all the blocks (in plain text form) of data. Each plain text block (i.e.B_i) can be retrieved using 5 different ways in 2/3-Xor-Combination as follows:

$$B_i = C_{i-3,i-2,i-1} \oplus C_{i-3,i-2} \oplus C_{i-1,i}$$
$$B_i = C_{i-2,i-1,i} \oplus C_{i-2,i-1}$$
$$B_i = C_{i-1,i,i+1} \oplus C_{i-1,i} \oplus C_{i,i+1}$$
$$B_i = C_{i,i+1,i+2} \oplus C_{i+1,i+2}$$
$$B_i = C_{i+1,i+2,i+3} \oplus C_{i,i+1} \oplus C_{i+2,i+3}$$

Considering 3-block-combination, as it has higher number of blocks, in 5 different ways there are 5 different consecutive 3-block-combinations. To maximize block recoverability, we can put these 5 consecutive 3-block-combinations in 5 different cloud servers so that any server crash will not prevent the reconstruction of the data block using other servers. Thus, as a rule of thumb, we can put every 6th 3-block-combination together in same cloud server and every 7th 2-block-combination in same cloud server. And also to confirm that, no 2-block-combination and 3-block-combination containing same data block (i.e.B_i) falls in the same server.

Furthermore, this $Block - Management$step populates Meta data into fog server's database and cloud servers' database tables as shown in Table 1 and Table 3 correspondingly.

4.2 Putting things together

This subsection describes the process of storing and retrieving data (e.g. file or document) to and from cloud server. When a user uploads a file to cloud storage, he follows $Storing\ Procedure$ and $Retrieval\ Procedure$ in

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSUSC.2019.2914954, IEEE Transactions on Sustainable Computing

case of downloading data from cloud server. In both cases, the user resorts to fog server for secrecy.

4.2.1 Storing Procedure:

Storing procedure takes a file to be uploaded to cloud server securely. It has several steps and most crucial steps take place in fog server. Fig. 4 shows its different steps and its description is given in the following section. When the user intends to upload a data file, he sends the file to the fog server through some secure channel. Then, fog server starts processing the file. Fig. 4(a) demonstrates the $Storing\ Procedure$.

4.2.1.1 Splitting File:

Fog server pads the file as per needs based on system policy. After that fog server splits the file into several fixed length blocks and combines them using $Xor-Combination$ algorithm. At the end of this step, we get two sets of 2-block-combinations and 3-block-combinations together known as $combined\ blocks$.

4.2.1.2 Integrity Processing:

For each $combined\ block$, fog server generates random number, hash digest and random hash digest using CRH.processing algorithm and stores this information into fog database (like Table 2) for future integrity check.

4.2.1.3 Block Management:

At this step, fog server determines which block to be stored to which cloud server using $Block-Management$ technique, stores this metadata into fog database (like Table 1) and sends the blocks to respective cloud servers.

4.2.1.4 Cloud Storage:

Cloud server receives and stores the blocks along with metadata (like Table 3) into its storage.

4.2.2 Retrieval Procedure:

Retrieval procedure takes a request of a file, collects necessary $combined\ blocks$ from various cloud servers, and checks their integrity. If integrity check fails then it requests faulty blocks from other cloud servers. When all the necessary combined blocks pass integrity check, the fog server reconstructs the entire file and sends it back to the user. Fig. 4(b) illustrates the $Retrieval\ Procedure$.

4.2.2.1 Look up:

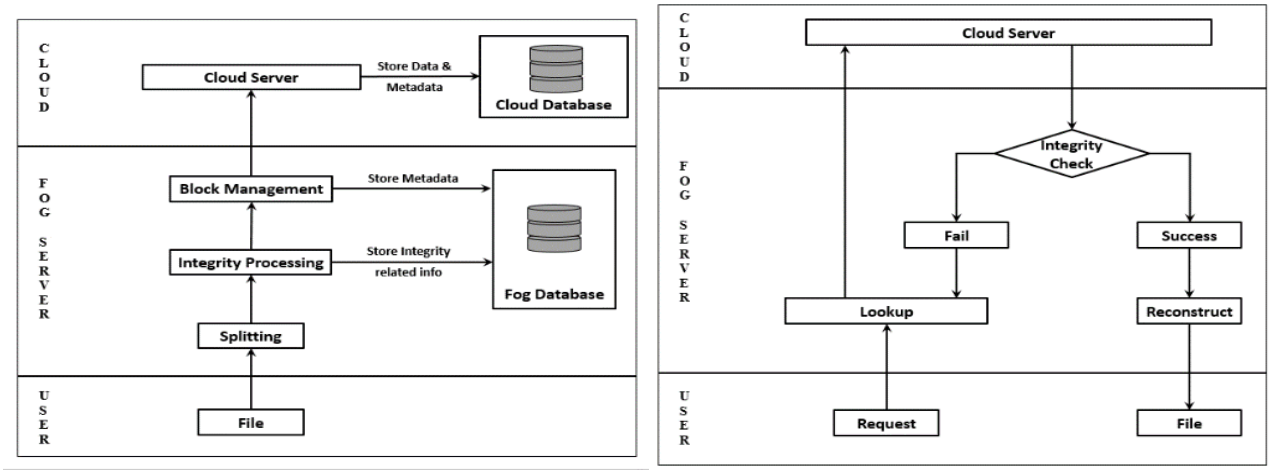
Once a user requests for a file to the fog server, the fog server looks up the relevant $combined\ blocks$ to construct the file into its metadata database. Afterwards it sends request to corresponding cloud servers holding the $combined\ blocks$.

4.2.2.2 Integrity check:

When the cloud servers send *combined blocks* back to the fog server, fog server checks integrity of each *combined block* using CRH.verification algorithm. If integrity check fails for a $combined\ block$ then the fog server discards it and tries to reconstruct data block using other $combined\ block$s stored in other cloud server.

4.2.2.3 Reconstruction:

Once the fog sever gets all the necessary combined blocks to derive the data blocks, it reconstructs the entire file. Finally, it sends the file back to the user.



a. Storing Procedure

b. Retrieving Procedure

Fig. 4. File Processing

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSUSC.2019.2914954, IEEE Transactions on Sustainable Computing

5 Security Analysis

In the section, we analyse the strength of the proposed scheme and investigate if it achieves the goals defined in sub-section 3.3. First of all, *privacy guarantee* which refers to attacker's inability of cracking the hidden text. Secondly, *data recoverability* tries to retrieve data even in case of permanent data loss from some cloud servers. Lastly, *modification detection* implies finding out any malicious data changes.

A Privacy Preservation

In the proposed scheme, trusted fog server processes the data, stores the metadata into its storage and uploads the data (hidden by *Xor-Combination* algorithm) to the different clouds' storages. Therefore, cloud server only gets hidden data and without collaboration with fog server, it cannot retrieve the actual data. Besides, fog server uploads different portions of data to different clouds. Thus, even if a cloud server is able to retrieve the data, it only gets a fraction of data. However, the proposed scheme wishes to no information leakage to the cloud server. Prior schemes [12, 13] using Reed-Solomon code or Reed-Solomon derived code cannot hide small portions of data from the cloud servers storing them. In contrast, we propose a noble technique $Xor-Combination$ to attain the goals.

Due to smaller word space, cloud server, namely, the attacker can run the following cryptanalysis on $Xor-Combination$. Suppose there are two words w_1 = "jack" and w_2 = "sparrow". The attacker slides one word against another as shown in Fig. 5 and computes Xor of aligned letters (like Fig. 5). In this manner, he gets $|w_1|+|w_2|$ = $4+7=11$ outcomes of Xor operations. When the attacker gets a *combined block*, he finds, if any of the outcomes exists as a substring in the *combined block*. If it exists then the attacker concludes possible existence of w_1 and w_2 with some probability. Conversely, as language has enough redundancy, so there is no certainty of the findings. Again, latest edition of Merriam Webster dictionary has 225000 ($\approx 2^{18}$) word definitions [44]. Thus, there will be 2^{36} possible keyword pairs, making it computationally infeasible to check this way. In order to make this cryptanalysis harder, one can use *3/4-Xor-Combination* that requires 2^{54} word pairs. Then, the attacker has to deal with three word combinations such as "captain", "jack", "sparrow". Or *4/5-Xor-Combination* that requires 2^{72} word pairs and so on. Finally, encrypting the

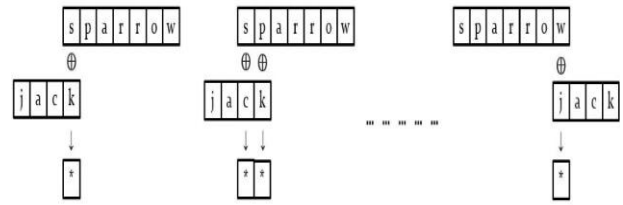


Fig. 5. Cryptanalysis of Xor-Combination

$Xor-Combination$ plays a crucial role to hide and recover data in the proposed scheme. Now, we will demonstrate, how it withstands against information breaches. $Xor-Combination$ splits the data into fixed length (L) data blocks and combines consecutive data blocks by Xor operation. Here, each *data block* has 2^L options and the outcome of Xor operation has 2^L options. Thus, an attacker can pre-compute all the $\frac{2^{2L}}{2}=2^L$ pairs of operands (*data blocks*) of 2^L Xor outcomes (*combined blocks*). Then, when the attacker finds a *combined block*, he has 2^L different options of possible pair of operands. As no consecutive *combined block* is available to the attacker (i.e. the cloud server), it leads to confusion about deterministic selection of pair of *data blocks* from *combined block*.

combined blocks before sending to cloud server further preserves the privacy that creates computation overhead though.

B Data Recoverability

As mentioned earlier, *2/3-Xor-Combination* allows to retrieve a *data block* (B_i) in five different ways. $Xor-Combination$, along with $Block-Management$ ensures data recoverability. $Xor-Combination$ generates *combined blocks* and $Block-Management$ saves it in diversified clouds' storages so that it can be recovered while some clouds either hides or losses the *combined blocks*. Following are the five different calculations for a data block retrieval:

1. $B_i = C_{i-3,i-2,i-1} \oplus C_{i-3,i-2} \oplus C_{i-1,i}$
2. $B_i = C_{i-2,i-1,i} \oplus C_{i-2,i-1}$
3. $B_i = C_{i-1,i,i+1} \oplus C_{i-1,i} \oplus C_{i,i+1}$
4. $B_i = C_{i,i+1,i+2} \oplus C_{i+1,i+2}$
5. $B_i = C_{i+1,i+2,i+3} \oplus C_{i,i+1} \oplus C_{i+2,i+3}$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSUSC.2019.2914954, IEEE Transactions on Sustainable Computing

Given the rule-of-thumb of $Block-Management$ to maximize recoverability, every 6th 3-block-combination is to be stored in the same cloud server and every 7th 2-block-combination is to be stored in the same cloud server, we preserve the combined blocks in 11 different cloud servers as shown in Fig. 6. From Fig. 6, data block B_i can be retrieved using *combined blocks* $C_{i-1,i,i+1}$, $C_{i-1,i}$ and $C_{i,i+1}$ which are preserved in CSP 3, CSP 8 and CSP 9 respectively. Even if other CSPs hide or loss data, *data block* B_i can be recovered. In the same manner, despite disasters in any of the cloud server, entire data can be reconstructed.

Storing combined blocks in several cloud servers is financially advantageous due to cloud servers' pay-as-you-go payment policy [45]. $Combined\ blocks$ are smaller building blocks of a data and preserving them in single or multiple cloud server(s) costs almost the same.

C Modification Detection

Standard collision resisting and one way hashing algorithms (e.g. SHA-256, MD5) work fine for cross checking the integrity of a file or password. However, hashing algorithms take arbitrary length input and generate fixed size output as illustrated in Fig. 7. Therefore, there exist multiple inputs which are mapped to same output according to the pigeonhole principal [46] and one can get a collision in a hash function incidentally which is very unlikely for standard hash functions. To address the problem, previous schemes replaced the hash digest (i.e. output of hash function) with random numbers from particular location and stored the digest with the random number for future integrity checks [13]. However, it has no extra effect on detection of malicious modification. In contrast, we propose a noble technique titled as $Collision\ Resisting\ Hashing\ (CRH)$ utilizing chaotic property of a hash function[47].

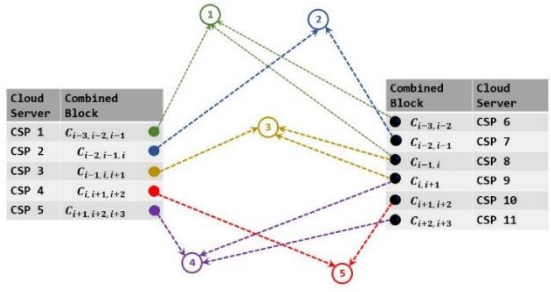


Fig. 6.Data Block Recovery using $\boldsymbol{Xor-Combiation}$ & $\boldsymbol{Block-Management}$

Because of chaotic property of hash algorithm, a small change in the input results in massive changes in the output. Fig. 7 depicts the fact vividly. According to the figure, despite smaller distance between two inputs a and b the outputs $hash(a)$ and $hash(b)$ have much more distance between them. CRH utilizes this property and stores a random number R, hash digest of the given text (namely, $OriginalText$) concatenated with the random number (namely, $RandomDigest$) and the original hash digest ($OriginalDigest$) of the $OriginalText$.

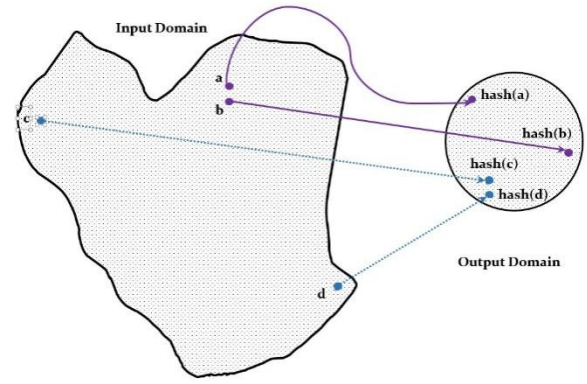


Fig. 7. Hash functions input/output mapping

Random number =R
Original text = $OriginalText$
$OriginalDigest = hash(OriginalText)$
$RandomDigest = hash(R \,||\, OriginalText)$

Suppose, we want to detect the modified text, ModifiedText, which has a collusion with $OriginalText$, that means, $OriginalText \neq ModifiedText$ and $hash(OriginalText) = hash(ModifiedText)$.

Now, $hash(R \,||\, OriginalText) \neq hash(R \,||\, ModifiedText)$ will be true with very high probability. It enables CRH to detect the modification with high probability.

If needed, one can resort to multiple random numbers $(R_1, R_2, R_3, \ldots\ \ldots\ \ldots)$ and multiple random digests $(RandomDigest_1, RandomDigest_2, RandomDigest_3, \ldots\ \ldots\ \ldots)$ to detect malicious modification almost with certainty.

6 Experiment & Performance Analysis

This section lays down experimental comparison of the proposed scheme with prior work of Wang et al. [13]. In order to make the logical comparison, we tried to align different aspects of environment such as block size, communication speed and so on. For data privacy, proposed scheme uses *Xor-Combination*, whereas Wang et al. depends on Reed-Solomon code. Similarly, to detect malicious modification, proposed scheme and Wang et al.'s scheme resorts to *CRH* and MMD (Maliciously Modified Detection) algorithms in respective manner. Therefore, comparison between the proposed scheme and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSUSC.2019.2914954, IEEE Transactions on Sustainable Computing

Wang et al.'s scheme implies comparison between respective algorithms/techniques.

We conducted the experiment in a machine with Windows 10 and 64-bit based operating system, Intel Core 2 duo processor with clock speed 3.33 GHz each, x64 based processor, 6 GB of RAM. We implemented the schemes using Java (jdk-8u141) and in Eclipse Juno IDE. Since, most of the computations take place in Fog Server, we experimented and analysed the various functionalities of fog server.

A: Data Processing

We selected files with size of 100KB to 1MB at increment of 100KB at each step, pad sufficiently and processed using *2/3-Xor-Combination* and a popular Reed-Solomon code *RS(255, 223)*. Splitting operation is same for both schemes, therefore, splitting is kept out of comparison in our experiment. The experiment takes each *data block*, processes it using respective algorithms and writes down the execution time. The result is shown in Fig. 8.

Fig. 8 demonstrates *Log* of elapsed time to encode different size of blocks using $Xor - Combination$ and Reed-Solomon code, namely, *RS(255, 223)*. It shows that, $Xor - Combination$ is three dimensional and faster than RS(255, 223). It can be attributed to the fact that, $Xor - Combination$ requires only Xor operation where the processor computes it directly inside its circuitry. Conversely, Reed-Solomon codes in software require *Galois Field* arithmetic operation and general purpose processor does not support it directly. For example, to implement a Galois field multiply in software, requires a test for 0, two log table look-ups, modulo add and anti-log table look-up. Therefore, in our implementation, it requires more execution time. However, in all cases, $Xor - Combination$ is supposed to work better than Reed-Solomon code, since it requires light weight computation.

Similarly, Fig. 9 illustrates the comparative time of decoding between $Xor - Combination$ and RS(255, 223). In this case, $Xor - Combination$ has same encoding/decoding time as it requires similar kind of operations. In contrast, RS(255, 223) requires much longer execution time for decoding. Besides, a close observation of Fig. 8 and 9 reveals that, in our experiment, encoding of *RS(255, 223)* takes longer time to decode . The reason is attributable to the fact that, we retrieve the data by decoding without incurring any error in the encoded text.

B: Communication Cost

There are two sorts of communication in our system model as indicated in Fig. 1(b): communication between user – fog server and communication between fog server – cloud servers. In communication between user – fog server, the proposed scheme and the benchmark scheme has no significant difference i.e. both schemes send almost same amount of payload from user to fog server. Conversely, the difference happens in the communication between fog server and cloud server. So, we conducted an experiment on the communication between fog and cloud servers. Besides, the communication speed, quality of service vary instalment to instalment, therefore, to measure the two schemes in a same scale, we compared the payloads they exchange between fog and cloud servers.

A detailed investigation revealed that, $Xor - Combination$ generates output with size two times the size of the original data. The total size of all *2-blocks-combinations* (of a particular file) is equal to the size of the original file. e.g. if a file is splitted into B_1, B_2, B_3, …, B_n then *2-block-combinations* are $B_1 \oplus B_2$, $B_2 \oplus B_3$, $B_3 \oplus B_4$, …, $B_n \oplus B_1$. There is n number of such *2-block-combinations* each with size $\left\lceil \frac{|file|}{n} \right\rceil$ making the total size$|file|$. Similarly, total size of all 3-block-combinations (of the file) is equal to the size of the file. Hence, output of $Xor - Combination$ has the size of $2\times|file|$.On the other hand, any *RS(n, k)* Reed-Solomon code generates output with size $\frac{n}{k} \times |file|$, where k and n is the size of input and output of a Reed-Solomon code respectively. For

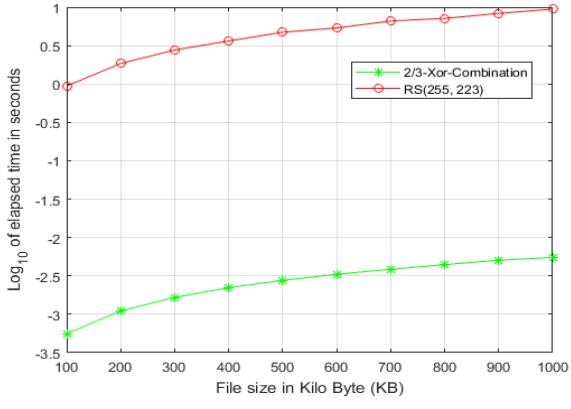


Fig. 8. Encoding time comparison between Xor-Combination and Reed-Solomon code

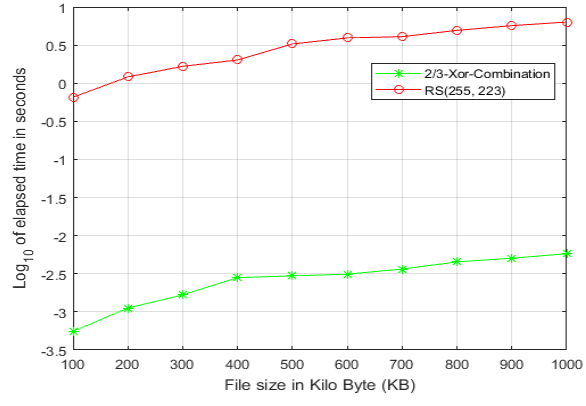


Fig. 9. Decoding time comparison between Xor-Combination and Reed-Solomon code

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSUSC.2019.2914954, IEEE Transactions on Sustainable Computing

example, *RS(255, 223)* produces output $\frac{255}{223}$ time the size of the original file. Consequently, to upload a file, the proposed scheme (which uses $Xor - Combination$) sends 2×$|file|$ amount of payload from fog to cloud servers and Wang et al.'s scheme sends $\frac{n}{k} \times |file|$ amount of payload. Fig. 10 illustrates the fact for different sizes of data file and Wang et al.'s scheme using *RS (255, 223)*. As the figure shows, in case of uploading data from fog server to various cloud servers, proposed scheme has to send larger size of payload in comparison with prior scheme. Hence, it bears higher communication cost. However, this is a trade-off for faster computation and better recoverability.

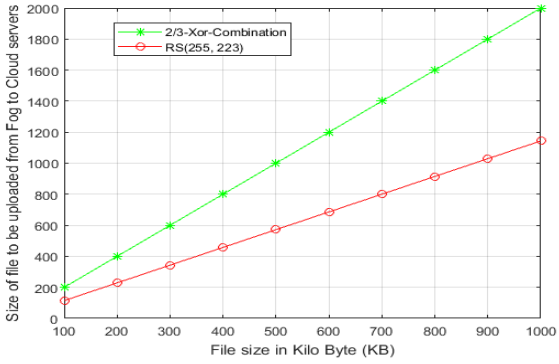


Fig. 10. Uploading time comparison using payload size

In case of downloading a file from cloud server to fog server, proposed scheme has some advantages in a special case. For example, in best case scenario when no malicious modification and/or data loss occur, only then a fraction of all *combined blocks* need to be downloaded from clouds to fog server. However, in case of malicious modification or data loss, fog server may need to incrementally download all the *combined blocks* of a file. Fig. 11 demonstrates the facts. In the best case of 2/3-Xor-Combination, cloud server needs to download only 1.2 times the size of the original file and in the worst scenario, it may incrementally require to download 2 times the original size. In all cases, RS (255, 223) code has the same downloading cost and it is similar to the cost of uploading file, numerically it is 1.14 times of size of the file.

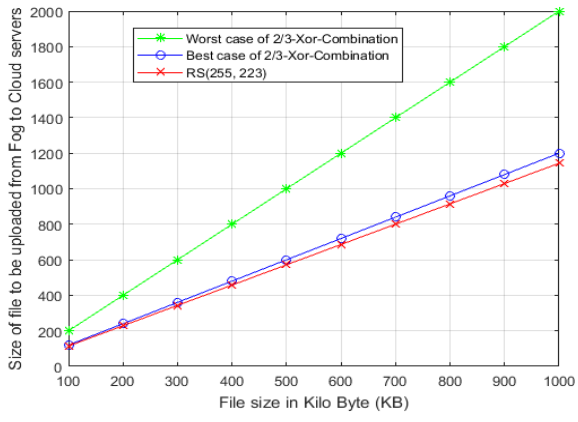


Fig. 11. Downloading time comparison using payload size

C *Modification Detection Cost*

To detect any malicious modification in a data file/document, proposed scheme uses *CRH* and the prior scheme resorts to using *MMD* techniques. However, both techniques depend on hash digest computation along with other light weight operations (i.e. substring operation, random number generation etc.). Hence, there is very insignificant difference between costs of the two schemes. There is only qualitative difference (i.e. robust collusion detection property in the proposed *CRH* algorithm).

7 Conclusion

The emergence of cloud computing has brought numerous advantages to the computing arena. The storage service is excellent unless users outsource their sensitive data to cloud storage server. Cloud server gets full access and control over user's data once data is outsourced to the cloud. It can read or search through the user's data. Moreover, data is susceptible to many cyber-attacks and cloud hardware or software malfunction may damage the data permanently. Fog based three-layer architecture befits to a secure solution for robust cloud storage against cyber threats. This article proposed a scheme that undertakes preventive activities to a trusted fog server and puts the actual data in twisted format to multiple cloud servers. As preventive measures s, this paper presents $Xor - Combination$, CRH and $Block - Management$ approaches. $Xor - Combination$ prepares a dataset for outsourcing by splitting and combining into fixed length blocks. As encryption is vulnerable to cracking and causes computational overhead, the proposed scheme does not rely on encryption technology. $Block - Management$ decides which combined blocks to be outsourced to which cloud server so that no individual cloud can retrieve the original data or a piece of data. At the same time,$Xor - Combination$, along with $Block - Management$, contributes to reconstruction of any data block in case of malicious modification or data loss. Finally, CRH supports the detection of any modification. Unlike the prior scheme, the proposed scheme twists the data before outsourcing it to cloud using $Xor - Combination$ so that no cloud server gets a smaller piece of data in plain text format. Similarly, $Xor - Combination$ enables better data recoverability and CRH facilitates integrity checks almost with certainty. Security analysis proves that it is computationally hard to extract plain text

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSUSC.2019.2914954, IEEE Transactions on Sustainable Computing

from a *combined block* which is outcome of $Xor-Combination$. Similarly, CRH overcomes the collision of a hash function (if any) with high probability and detects almost any malicious detection. Extensive comparative experiments indicate that its performance is effective as compared to the prior schemes. Future work in this domain can be summarized as follows:

1. To enhance the efficiency of fog based cloud storage service.
2. To improve the security of fog server for a robust fog centric cloud computing infrastructure.
3. To enable cloud server to compute cryptic data without revealing any information from it.

8 Acknowledgment

This research work was partially supported by the Faculty of Computer Science and Information Technology, University of Malaya under a special allocation of the Post Graduate Fund for RP036-15AET project. Imran's work is supported by the Deanship of Scientific Research, King Saud University, Saudi Arabia through Research Group No. 1435-051.

References

[1] P. Mell and T. Grance, "The NIST definition of cloud computing," Communications of the ACM, vol. 53, no. 6, p. 50, 2010.

[2] X. Liu, S. Zhao, A. Liu, N. Xiong, and A. V. Vasilakos, "Knowledge-aware proactive nodes selection approach for energy management in Internet of Things," Future generation computer systems, 2017.

[3] Y. Liu, A. Liu, S. Guo, Z. Li, Y.-J. Choi, and H. Sekiya, "Context-aware collect data with energy efficient in Cyber–physical cloud systems," Future Generation Computer Systems, 2017.

[4] J. Chase, R. Kaewpuang, W. Yonggang, and D. Niyato, "Joint virtual machine and bandwidth allocation in software defined network (SDN) and cloud computing environments," in Communications (ICC), 2014 IEEE International Conference on, 2014, pp. 2969-2974: IEEE.

[5] B. Martini and K.-K. R. Choo, "Distributed filesystem forensics: XtreemFS as a case study," Digital Investigation, vol. 11, no. 4, pp. 295-313, 2014.

[6] N. D. W. Cahyani, B. Martini, K. K. R. Choo, and A. Al-Azhar, "Forensic data acquisition from cloud-of-things devices: windows Smartphones as a case study," Concurrency and Computation: Practice and Experience, vol. 29, no. 14, 2017.

[7] J. Fu, Y. Liu, H.-C. Chao, B. Bhargava, and Z. J. I. T. o. I. I. Zhang, "Secure data storage and searching for industrial IoT by integrating fog computing and cloud computing," 2018.

[8] C. F. Tassone, B. Martini, and K. K. R. Choo, "Visualizing digital forensic datasets: a proof of concept," Journal of forensic sciences, vol. 62, no. 5, pp. 1197-1204, 2017.

[9] C. Hooper, B. Martini, and K.-K. R. Choo, "Cloud computing and its implications for cybercrime investigations in Australia," Computer Law & Security Review, vol. 29, no. 2, pp. 152-163, 2013.

[10] D. Quick and K.-K. R. Choo, "Digital forensic intelligence: Data subsets and Open Source Intelligence (DFINT+ OSINT): A timely and cohesive mix," Future Generation Computer Systems, vol. 78, pp. 558-567, 2018.

[11] Y.-Y. Teing, A. Dehghantanha, K.-K. R. Choo, and L. T. Yang, "Forensic investigation of P2P cloud storage services and backbone for IoT networks: BitTorrent Sync as a case study," Computers & Electrical Engineering, vol. 58, pp. 350-363, 2017.

[12] T. Wang et al., "Fog-based storage technology to fight with cyber threat," Future Generation Computer Systems, 2018.

[13] T. Wang, J. Zhou, X. Chen, G. Wang, A. Liu, and Y. Liu, "A Three-Layer Privacy Preserving Cloud Storage Scheme Based on Computational Intelligence in Fog Computing," IEEE Transactions on Emerging Topics in Computational Intelligence, vol. 2, no. 1, pp. 3-12, 2018.

[14] T. Wang et al., "Data collection from WSNs to the cloud based on mobile Fog elements," Future Generation Computer Systems, 2017.

[15] M. Xie, U. Bhanja, J. Shao, G. Zhang, and G. Wei, "LDSCD: A loss and DoS resistant secure code dissemination algorithm supporting multiple authorized tenants," Information Sciences, vol. 420, pp. 37-48, 2017.

[16] S. Basu et al., "Cloud computing security challenges & solutions-A survey," in Computing and Communication Workshop and Conference (CCWC), 2018 IEEE 8th Annual, 2018, pp. 347-356: IEEE.

[17] D.-G. Feng, M. Zhang, Y. Zhang, and Z. Xu, "Study on cloud computing security," Journal of software, vol. 22, no. 1, pp. 71-83, 2011.

[18] R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, fog et al.: A survey and analysis of

security threats and challenges," Future Generation Computer Systems, vol. 78, pp. 680-698, 2018.

[19]H. Takabi, J. B. Joshi, and G.-J. Ahn, "Security and privacy challenges in cloud computing environments," IEEE Security & Privacy, vol. 8, no. 6, pp. 24-31, 2010.

[20]L. M. Kaufman, "Data security in the world of cloud computing," IEEE Security & Privacy, vol. 7, no. 4, 2009.

[21]D. Zissis and D. Lekkas, "Addressing cloud computing security issues," Future Generation computer systems, vol. 28, no. 3, pp. 583-592, 2012.

[22]C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for data storage security in cloud computing," in Infocom, 2010 proceedings ieee, 2010, pp. 1-9: Ieee.

[23]J. Shen, J. Shen, X. Chen, X. Huang, and W. Susilo, "An efficient public auditing protocol with novel dynamic structure for cloud data," IEEE Transactions on Information Forensics and Security, vol. 12, no. 10, pp. 2402-2415, 2017.

[24]Z. Xia, N. N. Xiong, A. V. Vasilakos, and X. Sun, "EPCBIR: An efficient and privacy-preserving content-based image retrieval scheme in cloud computing," Information Sciences, vol. 387, pp. 195-204, 2017.

[25]R. Arora, A. Parashar, and C. C. I. Transforming, "Secure user data in cloud computing using encryption algorithms," International journal of engineering research and applications, vol. 3, no. 4, pp. 1922-1926, 2013.

[26]J. Shen, D. Liu, J. Shen, Q. Liu, and X. Sun, "A secure cloud-assisted urban data sharing framework for ubiquitous-cities," Pervasive and mobile Computing, vol. 41, pp. 219-230, 2017.

[27]K. M. Khan and Q. Malluhi, "Establishing trust in cloud computing," IT professional, vol. 12, no. 5, pp. 20-27, 2010.

[28]Z. Fu, X. Wu, Q. Wang, and K. Ren, "Enabling central keyword-based semantic extension search over encrypted outsourced data," IEEE Transactions on Information Forensics and Security, vol. 12, no. 12, pp. 2986-2997, 2017.

[29]C. Guo, X. Chen, Y. Jie, F. Zhangjie, M. Li, and B. Feng, "Dynamic Multi-phrase Ranked Search over Encrypted Data with Symmetric Searchable Encryption," IEEE Transactions on Services Computing, 2017.

[30]Y. Yang, X. Liu, and R. Deng, "Multi-user Multi-Keyword Rank Search over Encrypted Data in Arbitrary Language," IEEE Transactions on Dependable and Secure Computing, 2017.

[31]J. Feng, L. T. Yang, G. Dai, W. Wang, and D. J. I. T. o. B. D. Zou, "A Secure Higher-Order Lanczos-Based Orthogonal Tensor SVD for Big Data Reduction," 2018.

[32]J. Feng, L. T. Yang, and R. J. I. C. C. Zhang, "Tensor-based Big Biometric Data Reduction in Cloud," vol. 5, no. 4, pp. 38-46, 2018.

[33]T. H. Luan, L. Gao, Z. Li, Y. Xiang, G. Wei, and L. Sun, "Fog computing: Focusing on mobile users at the edge," arXiv preprint arXiv:1502.01815, 2015.

[34]J. Shen, T. Zhou, X. Chen, J. Li, and W. Susilo, "Anonymous and traceable group data sharing in cloud computing," IEEE Transactions on Information Forensics and Security, vol. 13, no. 4, pp. 912-925, 2018.

[35]S. Zawoad, A. K. Dutta, and R. Hasan, "Towards building forensics enabled cloud through secure logging-as-a-service," IEEE Transactions on Dependable and Secure Computing, vol. 13, no. 2, pp. 148-162, 2016.

[36]H. Feng, J. Liu, Q. Wu, and W. Liu, "Predicate Fully Homomorphic Encryption: Achieving Fine-Grained Access Control over Manipulable Ciphertext," in International Conference on Information Security and Cryptology, 2017, pp. 278-298: Springer.

[37]I. Stojmenovic and S. Wen, "The fog computing paradigm: Scenarios and security issues," in Computer Science and Information Systems (FedCSIS), 2014 Federated Conference on, 2014, pp. 1-8: IEEE.

[38]J. Ni, K. Zhang, Y. Yu, X. Lin, X. S. J. I. T. o. D. Shen, and S. Computing, "Providing task allocation and secure deduplication for mobile crowdsensing via fog computing," 2018.

[39]F. Chen, T. Xiang, X. Fu, and W. Yu, "User differentiated verifiable file search on the cloud," IEEE Transactions on Services Computing, 2016.

[40]Z. Chen et al., "A cloud computing based network monitoring and threat detection system for critical infrastructures," Big Data Research, vol. 3, pp. 10-23, 2016.

[41]X. Fu, Z. Ling, W. Yu, and J. Luo, "Cyber Crime Scene Investigations (C²SI) through Cloud Computing," in Distributed Computing Systems Workshops (ICDCSW), 2010 IEEE 30th International Conference on, 2010, pp. 26-31: IEEE.

[42] M. Z. A. Bhuiyan, T. Wang, T. Hayajneh, and G. M. Weiss, "Maintaining the Balance between Privacy and Data Integrity in Internet of Things," in Proceedings of the 2017 International Conference on Management Engineering, Software Engineering and Service Sciences, 2017, pp. 177-182: ACM.

[43] X. Wang and H. Yu, "How to break MD5 and other hash functions," in Annual international conference on the theory and applications of cryptographic techniques, 2005, pp. 19-35: Springer.

[44] Y.-C. Chang and M. Mitzenmacher, "Privacy preserving keyword searches on remote encrypted data," in International Conference on Applied Cryptography and Network Security, 2005, pp. 442-455: Springer.

[45] H. J. Syed, A. Gani, R. W. Ahmad, M. K. Khan, and A. I. A. Ahmed, "Cloud Monitoring: A Review, Taxonomy, and Open Research Issues," Journal of Network and Computer Applications, 2017.

[46] D. J. Fernández-Bretón, "Hindman's Theorem is only a countable phenomenon," Order, vol. 35, no. 1, pp. 83-91, 2018.

[47] J. Wang, T. Zhang, N. Sebe, and H. T. Shen, "A survey on learning to hash," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 40, no. 4, pp. 769-790, 2018.