EFFICIENT SLEPIAN-WOLF BASED PROOF OF RETRIEVABILITY SCHEME FOR CLOUD STORAGE

TAN CHOON BENG

FACULTY OF COMPUTING AND INFORMATICS
UNIVERSITI MALAYSIA SABAH
2018

EFFICIENT SLEPIAN-WOLF BASED PROOF OF RETRIEVABILITY SCHEME FOR CLOUD STORAGE

TAN CHOON BENG

THESIS SUBMITTED IN PARTIAL FULFILLMENT FOR THE DEGREE OF MASTER OF SCIENCE

FACULTY OF COMPUTING AND INFORMATICS
UNIVERSITI MALAYSIA SABAH
2018

CERTIFICATION

NAME : **TAN CHOON BENG**

MATRIC NO. : **MI1611010T**

TITLE : **EFFICIENT SLEPIAN-WOLF BASED PROOF OF RETRIEVABILITY SCHEME FOR CLOUD STORAGE**

DEGREE : **MASTER OF SCIENCE (COMPUTER SCIENCE)**

VIVA DATE :

CERTIFIED BY:

1. MAIN SUPERVISOR **Seal and Signature**
Dr. Mohd. Hanafi Ahmad Hijazi

_____

DECLARATION

I hereby declare that the material in this thesis is my own work except for certain quotations, equations, summaries, definitions, and references, which have been duly acknowledged.

22 DECEMBER 2017 .....

TAN CHOON BENG

MI1611010T

CERTIFIED BY

_____

(SUPERVISOR)

DR. MOHD HANAFI AHMAD HIJAZI

ACKNOWLEDGEMENT

The completion of this master thesis is impossible without the kindness, help and guidance of several grateful parties. There are not much people who willing to stand by our side, encourage and support us to achieve our goals, regardless of favorable or unfavorable situations. Here, I would like take this opportunity to express my deepest gratitude and appreciation to them whom have been constantly assisting and supporting me along the way.

First of all, I would like express my sincere thankfulness to my main advisor, Dr. Mohd Hanafi Ahmad Hijazi who has been serving as a senior lecturer in Faculty of Computing and Informatics, Universiti Malaysia Sabah (UMS). He is my advisor since my Bachelor degree. Without his patience, motivation and continuous supports and advices, I am afraid that I would not able to make this far gratefully.

I would like to thank my co-advisor Associate Professor Yuto Lim from Japan Advanced Institute of Science and Technology (JAIST). I am grateful for his precious comments and advices which have always broaden my horizons on scientific research. His dedication in supervising always inspire me towards a new and feasible direction throughout my research.

Last but not least, I would like to express my deepest gratitude to my beloved family for mental support throughout my Master degree, although we were always a sea apart. Sincerely, I devote a thousand thanks to my caring family.

Tan Choon Beng
22 December 2017

ABSTRACT

Cloud storage is an online storage service offered by Cloud Service Provider (CSP), where client's data is hosted on cloud servers' side. However, as client does not have physical access to outsourced data, cloud storage sometime labelled as untrustworthy or semi-trustworthy. To ensure cloud data integrity and availability, which are the precondition for the existence of a cloud storage system, a protocol known as Proof of Retrievability (PoR) is introduced. PoR allows cloud storage to proof to the client that the stored data is intact and fully retrievable. Recently, Slepian-Wolf Based Proof of Retrievability (SW-PoR) was introduced to provide cost-efficient and time consistent exact repair mechanism for erroneous outsourced data. However, to achieve maximum resiliency for data correctness, encoding process of SW-PoR requires considerably long computational time compared to the conventional storage method involving replication. Hence, this research proposed two viable solutions as extension to SW-PoR to address this limitation. The solutions are named as Partial Binary Encoding for SW-PoR (PBE-SW-PoR) and Optimized SW-PoR (Opti-SW-PoR). PBE-SW-PoR allows part of the data, *A*, to be encoded by SW-PoR while the other part of the data, *B*, is secured by adapting Cyclic Redundancy Check (CRC) and replication. Opti-SW-PoR adapted the concept of partitioning to reduce computation time of SW-PoR. Simulation was conducted to evaluate the performance of the proposed solutions by means of comparison to the original SW-PoR scheme in term of computation time. In the simulation, PBE-SW-PoR and Opti-SW-PoR showed significant reduction with respect to total computation time compared to the original SW-PoR. PBE-SW-PoR took 85871.4 seconds while Opti-SW-PoR took 59.8 seconds respectively at data size of 1,000 file blocks. Using the same size of data, the original SW-PoR recorded 835,205.4 seconds of total computation time.

ABSTRAK

(PENAMBAHBAIKAN BUKTI KEBOLEHULANGAN SEMULA BERASASKAN SLEPIAN-WOLF DARI SEGI PRESTASI KOMPUTASI UNTUK DATA AWAN)

Penyimpanan awan merupakan satu perkhidmatan penyimpanan data atas talian yang disediakan oleh Pembekal Perkhidmatan Awan (CSP), di mana data pelanggan dihoskan di server awan, manakala pelanggan biasanya perlu membayar suatu jumlah bayaran penggunaan kepada CSP berdasarkan kadar penggunaan. Namun, disebabkan pelanggan tiada akses fizikal ke data penyumberan luar, penyimpanan awan kerap dikatakan kurang dipercayai atau semi-dipercayai. Demi memastikan integriti dan ketersediaan data awan yang menjadi prasyarat kewujudan sesuatu sistem penyimpanan awan, protokol yang dikenali sebagai Bukti Kebolehulangan Semula (PoR) telah diperkenalkan. PoR membenarkan penyimpanan awan untuk membuktikan kepada pelanggan bahawa data yang disimpan adalah utuh dan dapat dikembalikan dengan sepenuhnya. Sejak kebelakangan ini, Bukti Kebolehulangan Semula Berasaskan Slepian-Wolf (SW-PoR) telah diperkenalkan demi menyediakan kos efektif dan masa yang konsisten dalam pembaikan tepat mekanisme untuk data penyumberan luar yang rosak. Namun, untuk mencapai daya tahan lasak yang maksima untuk ketepatan data, proses pengekodan dalam SW-PoR memerlukan masa komputasi yang agak panjang berbanding dengan kaedah penyimpanan data secara tradisional seperti replikasi. Oleh itu, penyelidikan ini bercadangkan dua kaedah penyelesaian yang berpotensi sebagai kerja lanjutan SW-PoR untuk menangani isu ini. Kaedah-kaedah penyelesaian tersebut dinamakan Pengekodan Binari Separa untuk SW-PoR (PBE-SW-PoR) dan Optimum SW-PoR (Opti-SW-PoR). PBE_SW-PoR membolehkan sebahagian data sahaja, A, dikodkan, manakala bahagian yang selebihya, B, dijamin dengan mengadaptasi cek redundansi kitaran (CRC) dan replikasi. Opti-SW-PoR mengadaptasi konsep pembahagian untuk mengurangkan masa komputasi SW-PoR. Simulasi telah dijalankan untuk menilai prestasi kaedah-kaedah peneyelesaian yang dicadangkan tersebut dengan cara perbandingan dengan SW-PoR yang asal dari segi masa komputasi. Dalam simulasi itu, PBE-SW-PoR dan Opti-SW-PoR menunjukkan penggurangan jumlah masa komputasi yang menonjol. PBE-SW-PoR dan Opti-SW-PoR masing-masing menggunakan 85871.4 saat dan 59.8 saat sahaja dalam pengekodan data sebanyak 1,000 blok fail, manakala SW-PoR asal memakan jumlah masa sebanyak 835,205.4 saat.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AES	-	Advanced Encryption Standard
bit	-	binary digit
BLS signature	-	Boneh-Lynn-Shacham signature
CRC	-	Cyclic Redundancy Check
CSP	-	Cloud Service Provider
DOJ	-	Department of Justice
EC	-	Erasure Coding
ECC	-	Error Correcting Codes
GB	-	Giga Bytes
GHz	-	Giga Hertz
kb	-	kilobits
MAC	-	Message Authentication Code
Mb	-	Megabits
NC	-	Network Coding
NSA	-	National Security Agency
Opti-SW-PoR	-	Optimized SW-PoR
OS	-	Operating System
PBE-SW-PoR	-	Partial Binary Encoding for SW-PoR
PDP	-	Provable Data Possession
PoR	-	Proof of Retrievability
PoW	-	Proof of Ownership
PRF	-	Pseudorandom Functions
RAM	-	Random Access Memory
SSL	-	Secure Sockets Layer
SW-PoR	-	Slepian-Wolf based Proof of Retrievability
TLS	-	Transport Layer Security
TPA	-	Third Party Auditor
XOR	-	exclusive OR

LIST OF SYMBOLS

$\|F\|$	-	file size
$\|P\|$	-	number of elements in the list of permutation
$\oplus$	-	XOR
A	-	Part A of data in PBE-SW-PoR
B	-	Part B of data in PBE-SW-PoR
c	-	coded block
$\hat{c}$	-	metadata
C (m, 3) or $\binom{m}{3}$	-	Combination of m choose 3 (Binomial Coefficient)
crc	-	size of CRC bits
F	-	file
m	-	number of file blocks
m'	-	number of file blocks per partition
n	-	number of XORs
n'	-	number of XORs per partition
p	-	number of partitions
q	-	number of file blocks in A
r	-	redundancy per file block
rep	-	total size of replicates of CRC added B
s	-	size of a pair of coded block and metadata
S	-	storage cost
$serv$	-	number of servers
$\overline{w}$	-	file block
$\|\overline{w}\|$	-	block size

CHAPTER 1

INTRODUCTION

1.1 Overview

Cloud computing is a distributed, online based service provided by Cloud Service Provider (CSP), where online resources such as storage spaces and computing power are shared among cloud users (Mell and Grance, 2011). Due to the ubiquitousness and accessibility, the demand on cloud computing has increased over the years (Statista, 2017) while the highest global spending is on the Infrastructure as a Service (IaaS) (Srinivas, Reddy, and Qyser, 2014). Although the storage trend has shifted to cloud storage due to its advantages such as efficient telecommute, ubiquitous data storage and backup, as well as disaster recovery services (Baciu, 2015), but some organisations still using traditional data storage method via local servers due to some factors like vendor lock-in, reliability, privacy, pricing, interoperability, and security concern (Quest, 2015) (Nedelcu, Stefanet, Tamasescu, Tintoiu, and Vezeanu, 2015). Nonetheless, the ubiquitousness of cloud computing leads to high probability of cloud incidents like unauthorised access, confidential disclosure and financial loss. For instances, several well-known gigantic CSPs such as Amazon, Google, Microsoft and Sony have suffered from cloud incidents (Ko, Lee, and Rajan, 2013). Hence, cloud information security is one of the key factors that should be addressed to reduce and solve most of the cloud incidents worldwide, while promoting global cloud adoption securely.

Information security is identified as the key factor as well as the main concern towards the adoption of cloud in many corporations. It is composed of three main

components, namely confidentially, integrity and availability (ISACA, 2015). Confidentiality is defined as preserving authorized restrictions on access and disclosure, including means for protecting privacy and proprietary information; integrity is defined as the guarding against improper information modification or destruction, and includes ensuring information non-repudiation and authenticity; availability is defined as ensuring timely and reliable access to and use of information (ISACA, 2015).

With respect to the cloud data security concern, three cloud storage protocols were introduced:

i. Proof of Ownership (PoW),
ii. Provable Data Possession (PDP), and
iii. Proof of Retrievability (PoR).

Figure 1.1 illustrates the difference between PoW, PDP and PoR in a general view. PoW is a protocol used by cloud storage to ensure the stored data is only made available and accessible by legitimate users or true data owner only. Examples of PoW schemes are (Halevi, Harnik, Pinkas, and Shulman-Peleg, 2011), (Yu, Chen, and Chao, 2015), (Hur, Koo, Shin, and Kang, 2016), and (Lorena and Agustin, 2015). On the other hand, PDP schemes such as (Ateniese, Burns, and Herring, 2007), (Mukundan, Madria, and Linderman, 2014), (Lin, Shen, Chen, and Sheldon, 2017), and (Wang, Wu, Qin, Tang, and Susilo, 2017) were introduced to allow the storage server to proof to its data client that the stored data is correctly stored by the servers. As PDP does not provide recovery to erroneous data, PoR was introduced by (Juels and Kaliski Jr., 2007) to address this issue. Hence, in term of data retrievability, PoR is better than PDP. Nonetheless, PoR might have a higher computation time in data encoding and decoding for error recovery mechanism. The early work of PoR (Juels and Kaliski Jr., 2007) have a limitation in term of number of times challenge-proof (data audit) are allowed to be conducted. To overcome this limitation, an improved PoR scheme was introduced by (Shacham and Waters, 2008) that apply erasure coding (EC) to allow recovery in case of data corruption and no constraint on the number of times

challenge-proof (data audit) allowed to be conducted. However, it provides unbounded number of PoR challenges to ensure data integrity.

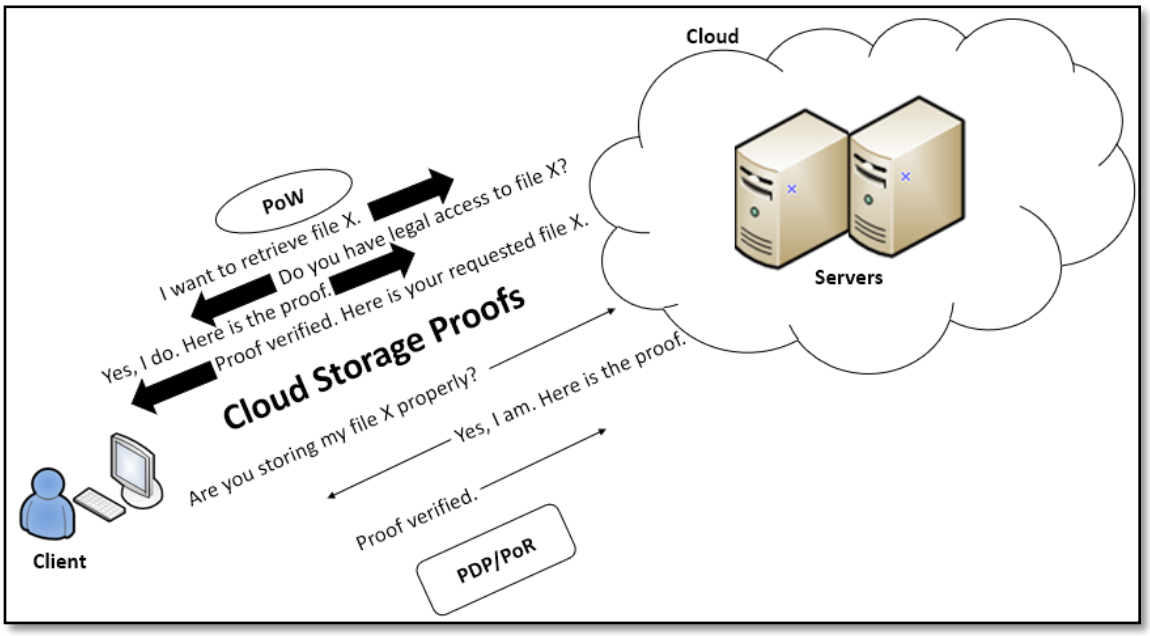

Figure 1.1: Difference between PoW versus PDP and PoR

Conventional data storage that involves replications requires abundant of storage spaces for data storage whereas cloud storage protocols like PDP and PoR schemes consumed less. However, PDP and PoR schemes requires data preprocessing, usually known as data encoding, before data is stored in distributed servers on cloud. Correspondingly, the conventional data storage method such as replication need only a minimal or even does not need any data preprocessing before the storing process take place. Eventually, PDP and PoR will perform slower than conventional data storage method like replication. Hence, performance in term of computation time of cloud storage protocols like PDP and PoR schemes is always the main concern in the construction of the scheme itself. It leaves a research gap on cloud storage schemes for achieving more efficient and secure PDP and PoR schemes.

1.2 Problem Statements

Recently, a variation of POR known as Slepian-Wolf based Proof of Retrievability (SW-PoR), designed for cloud storage to ensure the distributed cloud data is intact (Thao, Kho, and Lim, 2014). Intact of data means the stored data is in a condition where it is not damaged or not impaired in any way to ensure data completeness. There are three functions in SW-PoR scheme namely Encode, Retrieve and Repair. Details of these three functions will be described in Chapter 2.

SW-PoR scheme has been shown to be able to fully recover data even if data erroneous or lost occurs with its Repair function (Thao *et al.*, 2014) . Furthermore, SW-PoR also proven to be secure and efficient compared to linear network coding scheme for encode, retrieve and repair functions when simulated on different sizes of data up to 150 kilobits (kb) (Thao *et al.*, 2014). With respect to the real-world cloud system, the size of the data used in the reported simulation (Thao *et al.*, 2014) which is only 150 kb, is very difficult to represent the real cloud environment as the size data might be as small as few hundred kb of word document, up to few hundred megabits (Mb) of video file. The SW-PoR computation time is still an issue that need to be addressed. The time consumed by SW-PoR increases linearly when the file size increases for retrieve function, but the encode function rather experienced an exponential increment. When larger data is involved, the encoding time of SW-PoR scheme would be long, causing upload-to-storage time increases greatly. As a result, data integrity might be lost due to the un-trust of client to CSP, as trustworthiness of data is an important component in data integrity (Bertino and Lim, 2010).

1.3 Research Motivation

Storing a file in cloud does not end at uploading the file successfully, but it must also consider security measures such as encryption, error correction and recovery measures.

Hence, the process to store data in cloud requires synchronization time for data preprocessing before actual storage take place (Google, 2017a). Although it is common for cloud servers to consume some time for data preprocessing, but the length of the time used is always an issue. In most cases, a more secure and complex process of storing a file in cloud servers requires a longer synchronizing time (Google, 2017b).

Ironically, the longer the time for a file to stay in transit for preprocessing, the longer the time the file to be stored exposed to security threats like eavesdropping and other malicious activities. An excellent cloud storage in term of security and trustworthy requires efficient algorithms for the computation, particularly when the growth rate of data size is exponential. Hence, the motivation of the work presented in this thesis is to have a viable approach that allow shorter computation time of PoR scheme, specifically SW-PoR scheme, for cloud storage.

1.4 Research Questions

To achieve the motivation stated in the foregoing section, two research questions were identified:

1. What improvement can be conducted in order to reduce the computation time of SW-PoR?
2. Will the proposed improved SW-PoR compute efficiently on larger size data that uses an accepted amount of time to complete?

1.5 Research Objectives

The objectives of this research, derived from the identified research questions, are listed as follows:

1. To reduce the computation time of SW-PoR encoding using two proposed schemes, namely Partial Binary Encoding for SW-PoR (PBE-SW-PoR) and Optimized SW-PoR (Opti-SW-PoR).
2. To apply the two proposed schemes in (1) on larger data in simulation similar to actual cloud environment.
3. To compare and analyse the performance of the proposed scheme(s) in term of computation time against the original SW-POR.

1.6 Research Scope

This research is an extension work to SW-PoR scheme (Thao *et al.*, 2014) to improve its computational performance. Performance evaluation for the proposed schemes is based on simulation. Data sizes simulated are ranged from 200 file blocks to 1000 file blocks (1Mb), where each file block composed of 1024 bits. Larger data sizes are not simulated as original SW-PoR encoding would require extremely long time to complete, which is explained in Chapter 3. Simulation is conducted using personal computer for the mentioned data sizes to imitate real cloud environment. Machine specification used for simulation are as follow: Intel i5-3210M processor, 2.50 GHz, 8GB of RAM, Windows 10 (64-bit) OS.

1.7 Research Methodology

In order to achieve the research objectives, a methodology consists of four main phases is designed. Figure 1.2 illustrates the methodology employed in this research.

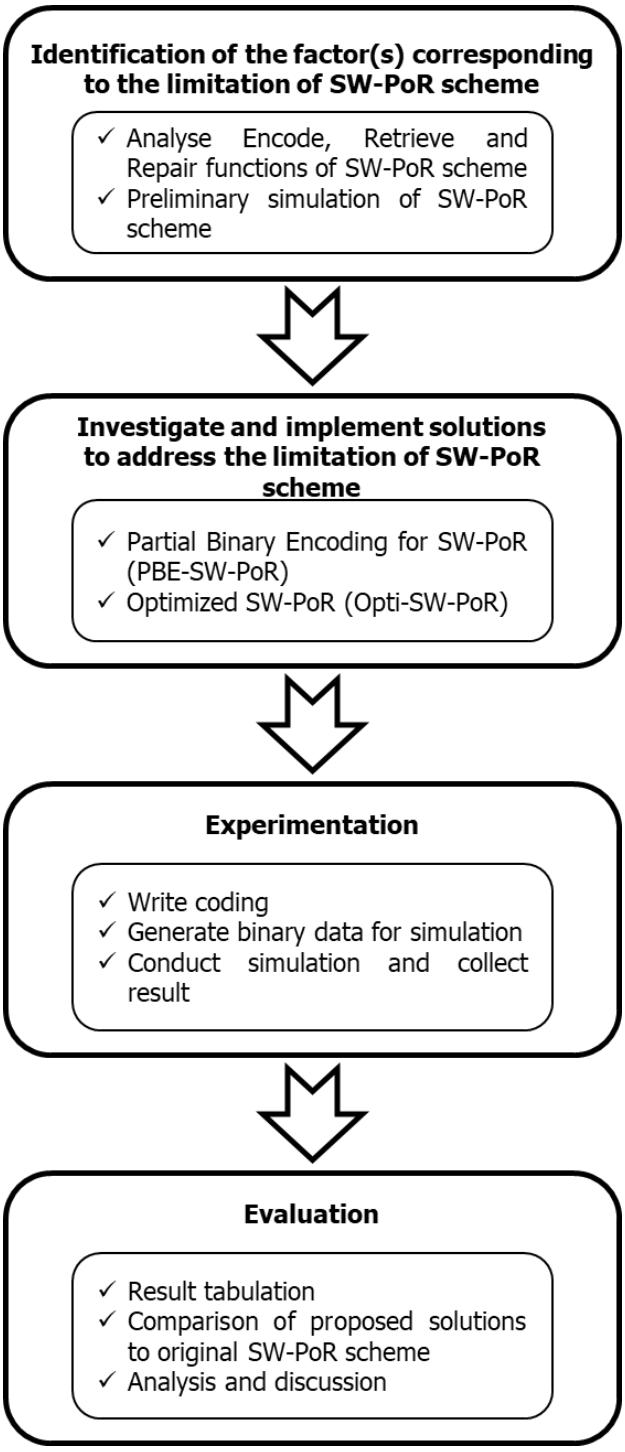

Figure 1.2: Research methodology

The main issue of SW-PoR is the encoding time increase exponentially when data size increases. To mitigate this issue, the factor(s) which cause the exponential increment in SW-PoR encoding time across different data sizes has to be identified. In this phase, Encode, Retrieve, and Repair functions of SW-PoR scheme have to be analyzed. Meanwhile, to ensure a thorough understanding of SW-PoR scheme, a preliminary simulation which mimic the simulation done by (Thao *et al.*, 2014) has to be conducted. Once the factor to the issue is identified, two solutions are investigated and implemented, which are Partial Binary Encoding for SW-PoR (PBE-SW-PoR) and Optimized SW-PoR (Opti-SW-PoR). The next phase is experimentation, which include codes writing, binary data generation for simulation, and conduct simulation and collect results. Once results are collected, they are organized and presented in tables and graphs. Then, from the results collected, the performances of the two proposed solutions will be compared with the original SW-PoR.

1.8 Evaluation Criteria

The main criteria to be evaluated in this thesis is computational performance of the two proposed solutions, PBE-SW-PoR and Opti-SW-PoR, in term of computation time. Unit of measurement used is seconds. Performances of Encode, Retrieve, and Repair functions of original SW-PoR, PBE-SW-PoR, and Opti-SW-PoR will be compared and evaluated. Besides, computational overhead for the two proposed solutions will be identified and calculated in term of seconds (please see Chapter 3 and 4 for details on computational overhead considered in this thesis).

1.9 Research Contributions

The main contributions of this research are listed as follows:

1. Two viable schemes of SW-PoR that consumed lower computation time compared to the original SW-PoR:
    a. PBE-SW-PoR

b. Opti-SW-PoR

2. Simulation to show and compare performance of the two proposed schemes with original SW-PoR in term of computation time.
3. Analysis and discussion of the simulation results.

1.10 Thesis Outline

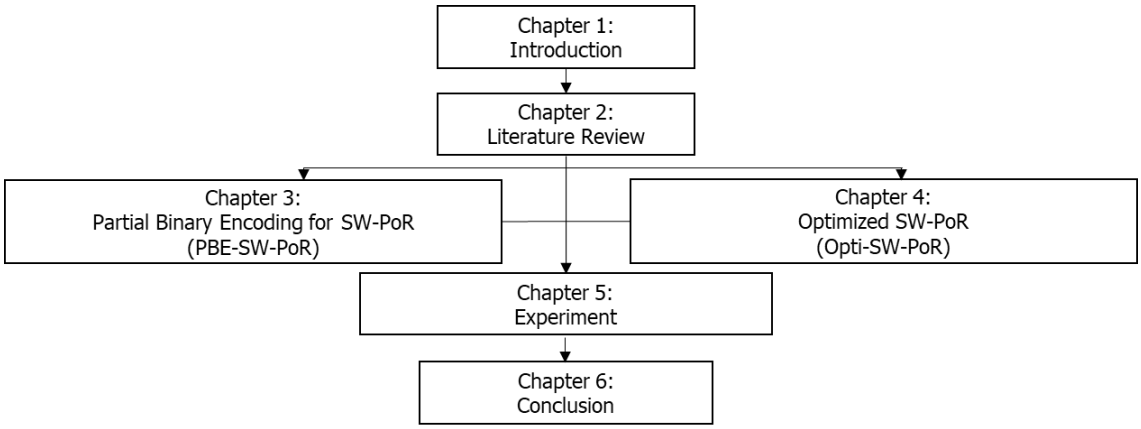

Figure 1.3: Thesis outline

This thesis consists of eight chapters as illustrated in Figure 1.3. Chapter 1 presents the introduction and overview of the research work. Related works, including the original SW-POR, are reviewed in Chapter 2. Chapter 3 and Chapter 4 present the proposed approaches, which are PBE-SW-PoR and Opti-SW-PoR respectively. The experimental setup, simulation and results are discussions in Chapter 5. Lastly, Chapter 6 concludes this research with summary and future research directions.

1.11 List of Publication

The work described in this thesis has been published in a number of refereed publication as itemized below:

1. Conference

a. Tan Choon Beng, Mohd Hanafi Ahmad Hijazi, and Yuto Lim. (2017). *Partial Binary Encoding for Slepian-Wolf Based Proof of Retrievability*. Student Conference on Research and Development 2017 (SCOReD2017). (Presented). This paper won the "Top 10 Paper Award" in SCOReD2017 conference.
This paper describes the Partial Binary Encoding for Slepian-Wolf Based Proof of Retrievability (PBE-SW-PoR), which is one of the proposed solution to the original SW-PoR presented in Chapter 3 of this thesis.

b. Tan Choon Beng, Mohd Hanafi Ahmad Hijazi, and Yuto Lim. (2017). *An Optimization Approach Towards a Proof of Retrievability Scheme for Cloud Storage*. International Conference on Computer and Network Applications 2017 (ICCNA2017). (Presented).
This paper describes the Optimized Slepian-Wolf Based Proof of Retrievability (Opti-SW-PoR), which is one of the proposed solution to the original SW-PoR presented in Chapter 4 of this thesis.

2. Journal

a. Tan Choon Beng, Mohd Hanafi Ahmad Hijazi, Yuto Lim and Abdullah Gani. (2018). *A Survey on Proof of Retrievability for Cloud Data Integrity and Availability: Cloud Storage State-of-the-Art, Issues, Solutions and Future Trends*. Journal of Network and Computer Applications. (Under Review)
This paper describes the work done in reviewing recent PoR schemes which is presented in Chapter 2 of this thesis.

3. Research Competition

a. Mohd Hanafi Ahmad Hijazi, Tan Choon Beng, and Yuto Lim. (2017). *Improving Slepian-Wolf based Proof of Retrievability (SW-PoR) Computation Performance on Cloud Data*. "Pertandingan Penyelidikan dan Rekacipta UMS (PEREKA 2017).

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Cloud storage is an online data storage system provided by CSP to data clients, where data clients are charged based on a usage rate. Although the emergence of cloud storage provides an alternative to data storage method, ubiquitous data accessing and storing provided there is Internet connection, and the saving local storage space, cloud data integrity and availability are always restricting factors to refrain the adoption of cloud storage in some organisations. As a solution to this, cloud storage protocols like PDP and PoR have been introduced to cloud storage. With respect to the work presented in this thesis, PoR is employed due to its ability to ensure data integrity, availability and recover stored data from corruptions. Recently, a variation of PoR scheme, Slepian-Wolf PoR (SW-PoR) was introduced to provides direct repair (repair directly at servers, clients need no action) and exact repair (corrupted data can be repaired to exactly the same as previous one) with a constant repair time. However, SW-PoR has a considerably high encoding computation time that is exponential. Hence, the work presented in this thesis proposed approaches to increase the efficiency of SW-PoR by means of reducing the computational time of SW-PoR. This chapter presents a review of relevant previous work on cloud storage, its remaining issues and PoR schemes. The SW-PoR is also described. Section 2.2 presents related work on cloud storage. PoR scheme and its variation SW-PoR are described in Sections 2.3 and 2.4 respectively. Section 2.5 concludes this chapter.

2.2 State-of-the-Art Cloud Storage

In order to keep up the pace of the development of cloud storage schemes, this section discusses on current cloud storage issues, vulnerabilities and challenges as well as countermeasures are presented. Based on the works reviewed in the literature, the possible cloud storage issues, vulnerabilities and challenges, together with some suggestions about the countermeasures are identified and presented in this section as well.

2.2.1 Conventional Cloud Storage

Conventional cloud storage using error correcting codes and data replication across distributed servers consumes a lot of storage spaces although it is fast and guaranteed to be error free. Usually, there is a replicate copy in each server to prevent outages and data corruption, while allowing fast retrieval with respect to data geolocation.

As the consumption of data storage spaces is directly reflected to the amount of storage hardware like storage disks needed to store data, the cost for data storage using conventional method like error correcting coded replicates across every server is extremely high.

All in all, either the conventional cloud storage or the latest cloud storage technologies do have their pros and cons respectively. In the next subsection, risks associated with cloud storage are reviewed and summarized, whereas the following subsection highlights about the issues of the cloud storage protocol, PoR with respect to cloud storage.

2.2.2 Risks Associated with Cloud Storage

With the emergence of the Internet, communication among people became borderless, this applies to information sharing as well. Despite countless advantages the Internet brings to society, it comes with a package of demerits to its applications. Regarding to this, cloud storage is an example of application of Internet, helps in reducing client's local storage spaces and managing the outsourced data on behalf of the client, but at the same time associating several risks to client's data.

There are several risks associated with cloud storage as described by (Fergus, 2017). The first risk is untrustworthy CSP where data integrity is affected when CSP is not trustworthy (Bertino and Lim, 2010). As client data is outsourced to cloud storage, the data is at possession of CSP who has full control over it. In conventional cloud storage, there is no data integrity schemes applied to secure the data, exposing the entrusted data to exposed to several risks like data tampering, un-authorized data modification and re-outsourced, even worse if deleted without notice. Besides, when the outsourced data is only physically accessible by CSP who have full control over the stored data, the security assurance of stored data lies solely under CSP. Second, there is an incomplete stored data deletion. Regarding to this, CSP may still possessing a duplicate copy of the stored data even if the data has been permanently deleted by the client, since CSP always has the full control and physical accessibility to the stored data. Common reason given by CSP for incomplete stored data deletion is for deletion rollback purpose in case of accidental deletion. Hence, client is unable to confirm whether their data is still possessed by CSP after deletion. Consequently, the data may be used by the CSP for other means. To avoid this, the best solution is by choosing a reliable and trustworthy CSP for outsourcing their data whereas privacy measures such as encryption can be applied to secure the outsourced data.

Besides, cloud storage outage is one of the cloud storage risk requires serious concern because this issue is closely related to the availability of stored data. When

cloud storage is on outage, all outsourced data is not available for retrieval or access. Although cloud storage outage is very unlikely to happens, but occasionally an annual downtime for servers' maintenance is possible. This is the reason why CSP usually guaranteed for 99.99% guaranteed availability but not 100% (Amazon, 2017). Therefore, it does not guarantee zero cloud storage outage. In fact, some titans among CSPs like Microsoft (Victoria, 2014), Amazon (Mohseen, 2014b) and Dropbox (Mohseen, 2014a) had their storage service outage in the past. In spite of there won't be anything significant lost during a wide scale during these outages, but the outage might cause important stored data irretrievable at times of need. Hence, this should raise the awareness of both data client and CSP about the risk of cloud storage outage.

Lastly, government intrusion is also a risk associated to cloud storage mentioned by (Fergus, 2017) which disrupting the confidentiality of stored data. Storing confidential information in cloud servers has put the authorities at ease to access it without permission from the data client. For example, some authorities may claim that the data is owned by CSP, thus forcing CSP to legally obligate to hands-out needed or targeted data stored in their respective cloud servers. Normally, some CSP will not easily turn-out data demanded by the authorities without a court order, however there is no guarantee of zero data leakage and disclosure of outsourced confidential information. For instance, public concern on data privacy was significantly raised in 2013 after the incident where one of the contractor, Edward Snowden from U.S. Department of Justice (DOJ), the National Security Agency (NSA), accidentally exposed information hinting NSA was using USA Patriot Act (BBC News, 2014) to justify the bulk collection of data about millions of phone calls (TechTarget, 2015). With respect to this, possible countermeasure is to encrypt information especially those sensitive one, on client side (Steffen, 2013) before outsourcing the encrypted data to the cloud servers.

Encryption is a process of converting data into non-human readable codes using cryptography keys to ensure data confidentiality. With respect to this, many

large CSP not limited to Dropbox, Microsoft OneDrive and Google Drive are offering their storage services with encryption on the outsourced data. In detail, (Davey, 2016) and (Virtru, 2016) have done some reviews for comparing Dropbox, OneDrive and Google Drive. Dropbox employed 128-bit Secure Sockets Layer or Transport Layer Security (SSL/TLS) for data-in-transit encryption and 256-bit Advanced Encryption Standard (AES) encryption to data-at-rest (Dropbox, 2017). Meanwhile, Google Drive employed 256-bit SSL/TLS encryption for data-in-transit and 128-bit AES encryption for data-at-rest (Google, 2017). In term of encryption, a higher number of bit key for encryption is considerably secure in term of possibility of brute-force break through by high end machines like quantum computer. On the other hand, OneDrive employed 128-bit SSL/TSL encryption for data-in-transit, but the encryption for data-at-rest is only available in OneDrive for Business using 256-bit AES (Microsoft, 2017), which means data stored in personal OneDrive accounts are vulnerable to security threats since no encryption is employed on data at rest (Sookasa, 2015). In comparison, it is clear that which CSP is more secure and otherwise. In addition, encryption can be done on either client-side or server-side, depends on where the encryption keys are kept. For a stronger security means in term of privacy, it is better to allow client-side encryption rather than server-side encryption so that the encryption keys are hold by client not CSP even though resources needed for data encryption like computation and processing power are more than a burden on client device.

In summary, cloud storage has three important risks including untrustworthy CSP, cloud storage outages, and government intrusion. These cloud storage risks are closely associated with data integrity, data confidentiality and data availability. Among these three main criteria, data integrity and availability are given more priority than confidentiality as they are the pre-conditions of the existence of a cloud storage system (Thao *et al.*, 2014). In fact, not all CSPs provide cryptography protection for stored data, and this is clearly shown in our real life where Microsoft OneDrive do not provide any encryption services on data-at-rest of personal accounts (Microsoft, 2017) for data confidentiality assurance. Hence, this is the reason why PoR schemes are needed to ensure both integrity and availability of data stored in cloud storage.

2.3 Proof of Retrievability (POR)

As time passes, the cloud storage has evolved to the next level by implementing lower storage cost method such as PDP and PoR schemes, for examples (Wang, Wu, Qin, Tang, and Susilo, 2017) (Juels and Kaliski Jr., 2007) (Shacham and Waters, 2008). Literature on PoR schemes introduced in recent years is presented in Section 2.3.1. Issues related to PoR schemes are described in Section 2.3.2.

2.3.1 PoR Schemes and the Taxonomy

Up to this year, 2017, about 10 to 11 years have passed since the introduction of cloud storage protocol, PoR in 2007 by (Juels and Kaliski Jr., 2007). Since 2007, there have been lots of PoR schemes proposed by researchers in the recent years for a variety of scope and domain. To have a clear and thorough understanding of recent works done on PoR, there is a need to review and summarize the latest trend of PoR schemes. In this section, review and summary of PoR schemes published and indexed by Scopus from 2013 to 2016 are conducted. Nevertheless, two papers (Juels and Kaliski Jr., 2007) and (Shacham and Waters, 2008), which have been widely referred in the construction of PoR schemes, are also considered which then added up to 43 PoR schemes being reviewed in this section. Figure 2.1 shows the time line of PoR schemes.

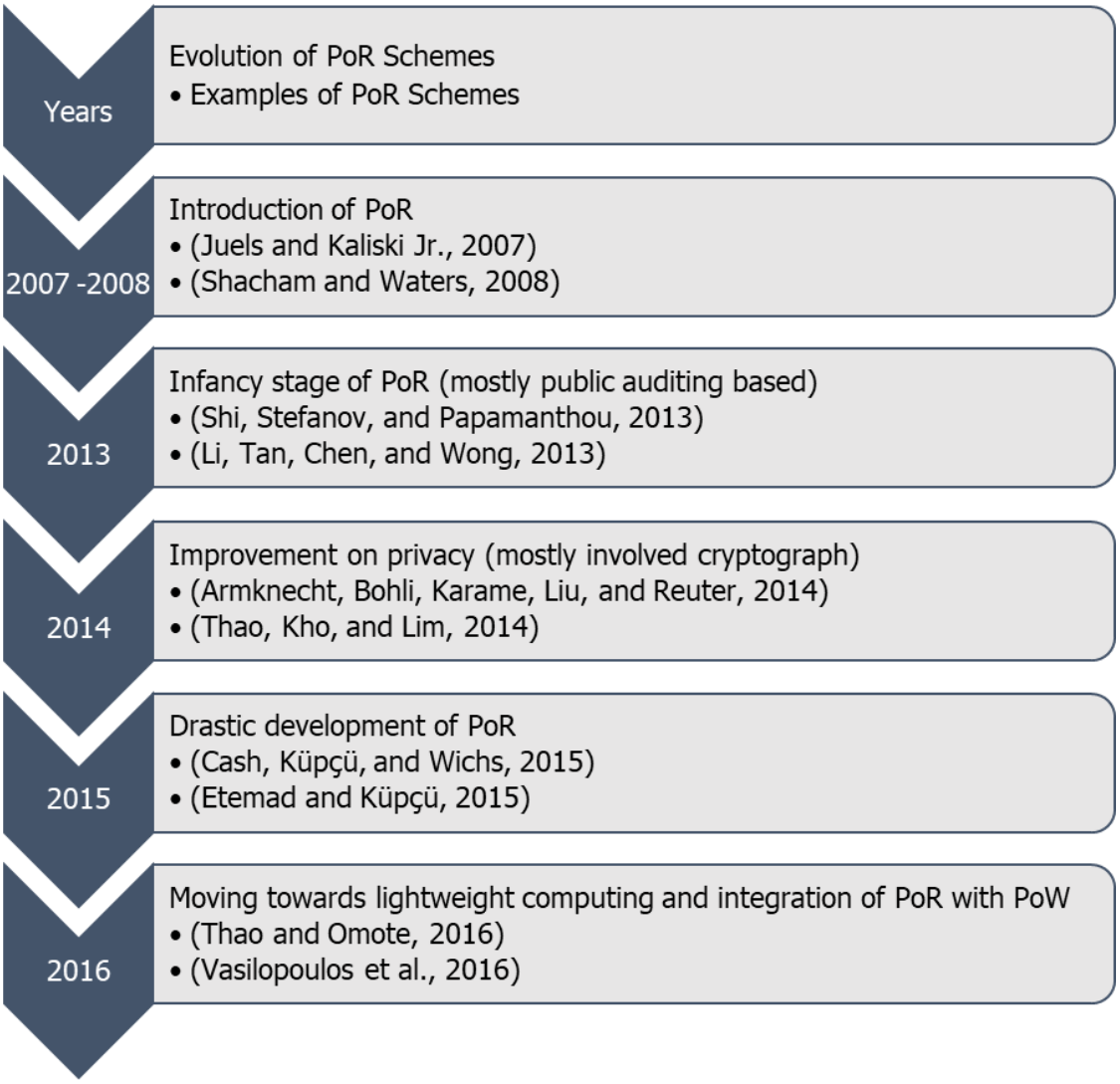

Figure 2.1: Time line of PoR schemes

The process to ensure availability of outsourced data via the method of replication across distributed servers in cloud is very resource extensive as data growth rate is exponential. Generally, the work of (Juels and Kaliski Jr., 2007) is a sentinel based PoR scheme proposed to ensure availability of cloud data similar to PDP. It also has error correcting codes component that enable cloud data to recover from corruption. The sentinel is a randomly-valued check block embedded in encrypted file for storage verification. To identify whether data corruption is correctly recovered, and verify whether the stored file is subjected to tampering or otherwise, Message

Authentication Code (MAC) is employed by (Juels and Kaliski Jr., 2007), Meanwhile, the sentinel embedded in data is functioned for storage auditing purpose (storage verification) whether the data is entirely stored or otherwise. Encryption is used for file security purpose. In addition to the way sentinels are embedded randomly into the encrypted file, preventing archive from distinguish between sentinels and actual data, leaving the storage servers no choice but have to store the entrusted file properly (Juels and Kaliski Jr., 2007).

Nevertheless, the PoR scheme proposed by (Juels and Kaliski Jr., 2007) has the limitation of bounded number of PoR 'challenge', which is the number of times a client or auditor can challenge the storage servers to provide proof where the entrusted file is stored properly via PoR. To address this limitation, (Shacham and Waters, 2008) proposed two audit schemes in the paper: (i) private audit scheme by using pseudorandom functions (PRF) and (ii) public audit scheme by using Boneh-Lynn-Shacham signature (BLS signature). The proposed auditing schemes reduced the challenge-proof response length by combining blocks and a number of authenticators into a single aggregated block and authenticator. In comparison, the private audit scheme has shown a shorter server response time compared to the public audit scheme. To ensure recovery and data corruption resiliency, erasure coding was used in PoR scheme proposed by (Shacham and Waters, 2008).

In the time line of PoR schemes shown in Figure 2.1, up to this moment, the available PoR schemes used different approaches and techniques of implementation. To adopt the PoR into real cloud environment, CSPs have to select the one that best fits their business objectives. By this mean, a taxonomy of PoR schemes is constructed to describe the common attributes of PoR schemes including the nature of data, cloud storage server setup, form of stored data, recovery, storage auditing, cryptography, as well as experimentation and analysis. The constructed taxonomy has adopted and modified the structure and several relevant attributes presented by (Zafar *et al.*, 2017), however the presented taxonomy as shown in Figure 2.2, hence not all attributes

presented by (Zafar *et al.*, 2017) are relevant to be assimilated in this thesis. The taxonomy is then elaborated with examples of recent PoR schemes in Table 2.1

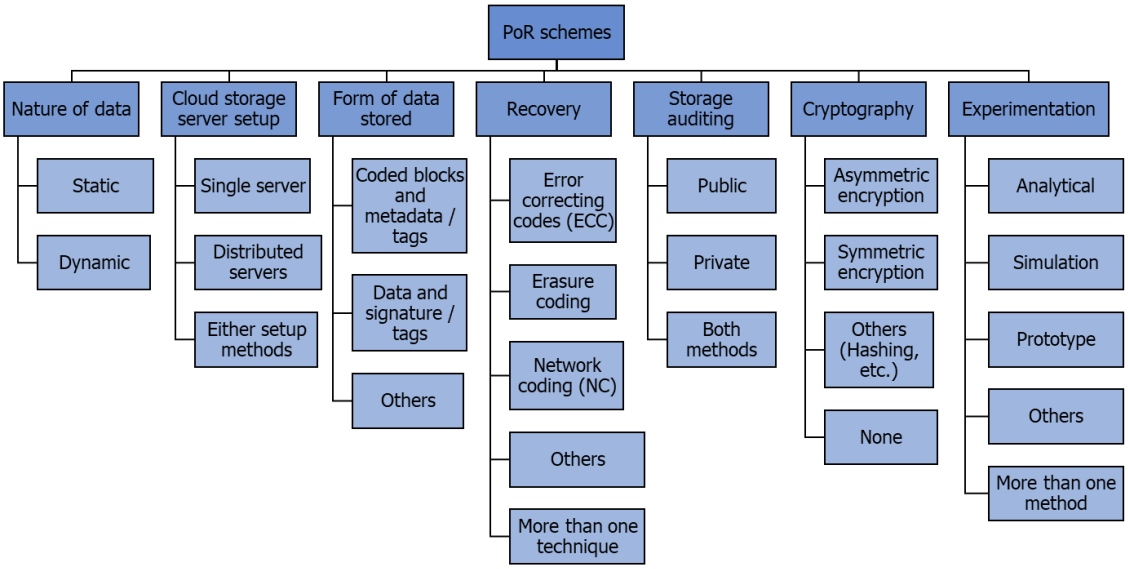

Figure 2.2: Hierarchical structure of the taxonomy of PoR schemes

Table 2.1: Elaborated taxonomy of recent PoR schemes with examples

	Attributes	**Sub-Attributes**	**References**
PoR Schemes	Nature of data	Static	(Juels and Kaliski Jr., 2007), (Shacham and Waters, 2008), (Yuan and Yu, 2013a), (Song and Deng, 2013), (Sarkar and Safavi-Naini, 2013), (Yan, Zhu, Gu, Zheng, and Fei, 2013), (Yuan and Yu, 2013b), (Armknecht, Bohli, Karame, Liu, and Reuter, 2014), (Thao *et al.*, 2014), (Chauhan and Saxena, 2014), (Zhang, Tang, and Mao, 2014), (Omote and Tran, 2014), (Juels, Kelley, Tamassia, and Triandopoulos, 2015), (Liu and Zic, 2015), (Shin, Koo, Hur, and Yun, 2015), (Jianchao, Huixia, Shoushan, Yaxing, and Wei, 2015), (Rashid, Miri, and Woungang,

			2015), (Omote and Thao, 2015), (Au, Mu, and Cui, 2015), (Omote and Tran, 2015), (Du, Deng, Chen, He, and Zheng, 2015), (Thao and Omote, 2016), (Vasilopoulos *et al.*, 2016), (Lavauzelle and Levy-Dit-Vehel, 2016), (Jingwei Li, Li, Xie, and Cai, 2016), (Sengupta, Bag, Ruj, and Sakurai, 2016)
		Dynamic	(Shi, Stefanov, and Papamanthou, 2013), (Jin Li, Tan, Chen, and Wong, 2013), (Rass, 2013), (Husain, Ko, Uurtamo, Rudra, and Sridhar, 2014), (Huang, Liu, Xian, Wang, and Fu, 2014), (Miller, Juels, Shi, Parno, and Katz, 2014), (Cash, Küpçü, and Wichs, 2015), (Etemad and Küpçü, 2015), (Kiraz, Sertkaya, and Uzunkol, 2015), (Jin Li, Tan, Chen, Wong, and Xhafa, 2015), (Tiwari and Gangadharan, 2015), (Ren, Wang, Wang, and Xu, 2015), (Mishra, Bhardwaj, and Kumar, 2015), (Wang *et al.*, 2015), (Xu, Zhou, Jiang, and Xue, 2016), (Saxena and Dey, 2016), (Omote and Tran, 2016)
	Cloud storage server setup	Single server	(Rass, 2013), (Chauhan and Saxena, 2014), (Cash *et al.*, 2015), (Liu and Zic, 2015), (Shin *et al.*, 2015), (Kiraz *et al.*, 2015), (Rashid *et al.*, 2015), (Xu *et al.*, 2016), (Lavauzelle and Levy-Dit-Vehel, 2016)
		Distributed servers	(Juels and Kaliski Jr., 2007), (Shacham and Waters, 2008), (Shi *et al.*, 2013), (Jin Li *et al.*, 2013), (Yuan and Yu, 2013a), (Song and Deng, 2013), (Sarkar and Safavi-Naini, 2013), (Yan *et al.*, 2013), (Yuan and Yu, 2013b), (Armknecht *et al.*, 2014), (Thao *et al.*, 2014), (Huang *et al.*, 2014), (Miller *et al.*, 2014), (Zhang *et al.*, 2014), (Omote and Tran, 2014), (Etemad and Küpçü, 2015), (Juels *et al.*, 2015), (Jianchao *et al.*, 2015), (Omote and Thao, 2015), (Jin Li *et al.*, 2015), (Au *et al.*, 2015), (Omote and Tran,

			2015), (Tiwari and Gangadharan, 2015), (Ren *et al.*, 2015), (Mishra *et al.*, 2015), (Du *et al.*, 2015), (Wang *et al.*, 2015), (Thao and Omote, 2016), (Vasilopoulos *et al.*, 2016), (Saxena and Dey, 2016), (Jingwei Li *et al.*, 2016), (Sengupta *et al.*, 2016), (Omote and Tran, 2016)
		Either setup methods	(Husain *et al.*, 2014)
	Form of data stored	Coded blocks and metadata / tags	(Juels and Kaliski Jr., 2007), (Shi *et al.*, 2013), (Jin Li *et al.*, 2013), (Yuan and Yu, 2013a), (Thao *et al.*, 2014), (Cash *et al.*, 2015), (Jianchao *et al.*, 2015), (Omote and Thao, 2015), (Jin Li *et al.*, 2015), (Omote and Tran, 2015), (Ren *et al.*, 2015), (Du *et al.*, 2015), (Xu *et al.*, 2016), (Lavauzelle and Levy-Dit-Vehel, 2016), (Omote and Tran, 2016)
		Data and signature / tags	(Shacham and Waters, 2008), (Song and Deng, 2013), (Sarkar and Safavi-Naini, 2013), (Yan *et al.*, 2013), (Yuan and Yu, 2013b), (Armknecht *et al.*, 2014), (Huang *et al.*, 2014), (Miller *et al.*, 2014), (Chauhan and Saxena, 2014), (Zhang *et al.*, 2014), (Omote and Tran, 2014), (Juels *et al.*, 2015), (Liu and Zic, 2015), (Shin *et al.*, 2015), (Kiraz *et al.*, 2015), (Rashid *et al.*, 2015), (Au *et al.*, 2015), (Tiwari and Gangadharan, 2015), (Mishra *et al.*, 2015), (Wang *et al.*, 2015), (Thao and Omote, 2016), (Saxena and Dey, 2016), (Jingwei Li *et al.*, 2016)
		Others	(Rass, 2013), (Husain *et al.*, 2014), (Etemad and Küpçü, 2015), (Vasilopoulos *et al.*, 2016), (Sengupta *et al.*, 2016)
	Recovery	Error correcting codes (ECC)	(Juels and Kaliski Jr., 2007), (Jin Li *et al.*, 2013), (Rass, 2013), (Husain *et al.*, 2014), (Chauhan and Saxena, 2014), (Juels *et al.*, 2015), (Rashid *et al.*, 2015), (Tiwari and Gangadharan, 2015), (Vasilopoulos *et al.*, 2016), (Jingwei Li *et al.*, 2016)

		Erasure coding	(Shacham and Waters, 2008), (Shi *et al.*, 2013), (Yuan and Yu, 2013a), (Yan *et al.*, 2013), (Yuan and Yu, 2013b), (Armknecht *et al.*, 2014), (Miller *et al.*, 2014), (Zhang *et al.*, 2014), (Cash *et al.*, 2015), (Etemad and Küpçü, 2015), (Shin *et al.*, 2015), (Jin Li *et al.*, 2015), (Au *et al.*, 2015), (Du *et al.*, 2015), (Xu *et al.*, 2016), (Sengupta *et al.*, 2016)
		Network coding (NC)	(Omote and Thao, 2015), (Thao and Omote, 2016), (Omote and Tran, 2016)
		Others	(Sarkar and Safavi-Naini, 2013), (Thao *et al.*, 2014), (Huang *et al.*, 2014), (Liu and Zic, 2015), (Jianchao *et al.*, 2015), (Kiraz *et al.*, 2015), (Mishra *et al.*, 2015), (Wang *et al.*, 2015), (Lavauzelle and Levy-Dit-Vehel, 2016), (Saxena and Dey, 2016)
		More than one technique	(Song and Deng, 2013), (Omote and Tran, 2014), (Omote and Tran, 2015), (Ren *et al.*, 2015)
	Storage auditing	Public	(Shi *et al.*, 2013), (Jin Li *et al.*, 2013), (Yuan and Yu, 2013a), (Song and Deng, 2013), (Sarkar and Safavi-Naini, 2013), (Yan *et al.*, 2013), (Yuan and Yu, 2013b), (Armknecht *et al.*, 2014), (Husain *et al.*, 2014), (Shin *et al.*, 2015), (Kiraz *et al.*, 2015), (Omote and Thao, 2015), (Jin Li *et al.*, 2015), (Au *et al.*, 2015), (Tiwari and Gangadharan, 2015), (Ren *et al.*, 2015), (Mishra *et al.*, 2015), (Wang *et al.*, 2015), (Thao and Omote, 2016), (Saxena and Dey, 2016)
		Private	(Juels and Kaliski Jr., 2007), (Rass, 2013), (Thao *et al.*, 2014), (Miller *et al.*, 2014), (Chauhan and Saxena, 2014), (Omote and Tran, 2014), (Cash *et al.*, 2015), (Etemad and Küpçü, 2015), (Juels *et al.*, 2015), (Liu and Zic, 2015), (Jianchao *et al.*, 2015), (Rashid *et al.*, 2015), (Omote and Tran, 2015), (Du *et al.*, 2015),

			(Xu *et al.*, 2016), (Vasilopoulos *et al.*, 2016), (Lavauzelle and Levy-Dit-Vehel, 2016), (Jingwei Li *et al.*, 2016), (Sengupta *et al.*, 2016), (Omote and Tran, 2016)
		Both methods	(Shacham and Waters, 2008), (Huang *et al.*, 2014), (Zhang *et al.*, 2014)
	Cryptography	Asymmetric encryption	(Shacham and Waters, 2008), (Armknecht *et al.*, 2014), (Miller *et al.*, 2014), (Kiraz *et al.*, 2015), (Au *et al.*, 2015), (Ren *et al.*, 2015), (Omote and Tran, 2016)
		Symmetric encryption	(Juels and Kaliski Jr., 2007), (Shi *et al.*, 2013), (Yuan and Yu, 2013a), (Sarkar and Safavi-Naini, 2013), (Yan *et al.*, 2013), (Yuan and Yu, 2013b), (Chauhan and Saxena, 2014), (Omote and Tran, 2014), (Cash *et al.*, 2015), (Juels *et al.*, 2015), (Liu and Zic, 2015), (Jianchao *et al.*, 2015), (Rashid *et al.*, 2015), (Omote and Thao, 2015), (Omote and Tran, 2015), (Mishra *et al.*, 2015), (Wang *et al.*, 2015), (Xu *et al.*, 2016), (Vasilopoulos *et al.*, 2016), (Lavauzelle and Levy-Dit-Vehel, 2016), (Saxena and Dey, 2016), (Jingwei Li *et al.*, 2016), (Sengupta *et al.*, 2016)
		Others (Hashing, etc.)	(Jin Li *et al.*, 2013), (Song and Deng, 2013), (Rass, 2013), (Husain *et al.*, 2014), (Huang *et al.*, 2014), (Jin Li *et al.*, 2015), (Tiwari and Gangadharan, 2015)
		None	(Thao *et al.*, 2014), (Zhang *et al.*, 2014), (Etemad and Küpçü, 2015), (Shin *et al.*, 2015), (Du *et al.*, 2015), (Thao and Omote, 2016)
	Experimentation and analysis	Analytical	(Juels and Kaliski Jr., 2007), (Shacham and Waters, 2008), (Jin Li *et al.*, 2013), (Yuan and Yu, 2013a), (Sarkar and Safavi-Naini, 2013), (Rass, 2013), (Chauhan and Saxena, 2014), (Omote and Tran, 2014), (Cash *et al.*, 2015), (Etemad and Küpçü, 2015), (Shin *et al.*, 2015),

			(Jianchao *et al.*, 2015), (Kiraz *et al.*, 2015), (Au *et al.*, 2015), (Omote and Tran, 2015), (Wang *et al.*, 2015), (Xu *et al.*, 2016), (Vasilopoulos *et al.*, 2016), (Omote and Tran, 2016)
		Simulation	(Shi *et al.*, 2013), (Yuan and Yu, 2013b), (Thao *et al.*, 2014), (Husain *et al.*, 2014), (Huang *et al.*, 2014), (Miller *et al.*, 2014), (Zhang *et al.*, 2014), (Juels *et al.*, 2015), (Rashid *et al.*, 2015), (Omote and Thao, 2015), (Tiwari and Gangadharan, 2015), (Ren *et al.*, 2015), (Du *et al.*, 2015), (Thao and Omote, 2016), (Lavauzelle and Levy-Dit-Vehel, 2016), (Saxena and Dey, 2016), (Jingwei Li *et al.*, 2016), (Sengupta *et al.*, 2016)
		Prototype	(Armknecht *et al.*, 2014), (Liu and Zic, 2015), (Mishra *et al.*, 2015)
		Others	(Jin Li *et al.*, 2015)
		More than one method	(Song and Deng, 2013), (Yan *et al.*, 2013)

The first attribute to include in the taxonomy of PoR schemes is nature of data. Data can be categorized into two main forms, static data and dynamic data. Static data is the data that stay unchanged most of the time after been created, whereas dynamic data is the data that always changed or updated. Nature of data is an important attribute in PoR scheme as some CSPs provide storage of static data, while some provide storage of dynamic data. Hence, adoption of which PoR scheme in cloud depends on the needs and the compatibility of PoR scheme in term of dynamic data operations support such as update, delete and insert operations on stored data. The PoR scheme introduced by (Juels and Kaliski Jr., 2007) as well as widely referenced model of PoR scheme by (Shacham and Waters, 2008) are both exhibit static data nature in their schemes, where they do not support dynamic operations. Similarly, PoR schemes proposed by (Yuan and Yu, 2013b), (Armknecht, Bohli, Karame, Liu, and Reuter, 2014), (Thao *et al.*, 2014), (Chauhan and Saxena, 2014) and (Omote and Tran, 2014) are designed to deal with static data. On the other hand, dynamic data

operations are supported in PoR schemes proposed by (Shi, Stefanov, and Papamanthou, 2013), (Jin Li, Tan, Chen, and Wong, 2013), (Rass, 2013), (Husain, Ko, Uurtamo, Rudra, and Sridhar, 2014), (Huang, Liu, Xian, Wang, and Fu, 2014) and (Miller, Juels, Shi, Parno, and Katz, 2014).

Next, the second attribute included in the taxonomy is cloud server setup. Regarding to this, there are two main ways cloud storage server to setup for PoR schemes, firstly, single server setup and secondly, multi-servers' setup or distributed servers' setup. PoR schemes which store the full data in a single server would apply single server's setup, some examples of this include PoR schemes proposed by (Cash, Küpçü, and Wichs, 2015), (Liu and Zic, 2015), (Shin *et al.*, 2015), (Kiraz, Sertkaya, and Uzunkol, 2015) and (Rashid *et al.*, 2015) in recent years. On the contrary, PoR schemes which partition or split the full data into parts or chunks, and then distributed to store in multiple servers before storing, suits the setup of distributed servers. For instances, PoR schemes proposed by (Etemad and Küpçü, 2015), (Juels, Kelley, Tamassia, and Triandopoulos, 2015), (Jianchao, Huixia, Shoushan, Yaxing, and Wei, 2015), (Omote and Thao, 2015) and (Jin Li *et al.*, 2015) apply distributed servers' setup. Obviously, apply distributed servers' setup to store a single file is more resilient and secure compared to store full data in a single server in term of data availability and error recoverability. In term of pros and cons of single server setup for PoR scheme, it yields considerably lower communication cost as this requires no communication between servers, but this setup requires the PoR scheme to enable recovery of full data each time server corruption happens. Besides, this type of server's setup often exposed to higher risk of server downtime problem. Frankly, distributed servers' setup method is better in term of data availability and corruption resiliency than that of single server setup for PoR schemes. In fact, there are more PoR schemes proposed in recent years that employ distributed servers' setup have outnumber the single server's setup PoR schemes. Nevertheless, there are PoR schemes which can be implemented in either server's setup method, for example (Husain *et al.*, 2014).

Thirdly, the form of stored data in cloud storage servers is an important attribute of PoR which included in the taxonomy. Data can be stored in cloud servers in a variety of forms, for examples, raw data which in its original form (not encrypted or coded) and distributed erasure coded data chunks. Nevertheless, there is no data form which is superior than the others, rather which form of data is used depends on the techniques applied in PoR schemes such as erasure coding and replication which work on these data. Depends on different techniques and requirements, data can be stored as chunks across distributed servers, or even as forward error-correcting code data stored in just a single server. For PoR schemes reviewed in this section of the thesis, we can categorize forms which data can takes as follows:

i. coded blocks with metadata or tags,
ii. data with signature or tags, and
iii. the others.

The first form, coded blocks and metadata or tags, can be seen as data that is broken into pieces of chunks or parts, then these data blocks changed their form into coded blocks after undergo some operations such as XOR, for example, $1100 \oplus 0011 = 1111$. Meanwhile, metadata or tags in this case served as a key or information for specific purposes not limited to decoding, for example the number of bit '1' in the coded data is stored as metadata. PoR schemes with data stored as the first form (coded blocks with metadata or tags) proposed by (Shi *et al.*, 2013), (Jin Li *et al.*, 2013) and (Yuan and Yu, 2013a) are mostly applied in distributed server's setup, nevertheless, there are some exception cases like PoR schemes proposed by (Cash *et al.*, 2015), (Xu, Zhou, Jiang, and Xue, 2016), and (Lavauzelle and Levy-Dit-Vehel, 2016). The second data storage form, which is data with signature or tags, can be seen as raw data, added with some codes such as parity bits to preserve data correctness (no corruption) known as metadata or tags. For PoR schemes which have the data to be stored in this form, it seems to more favorable with single server's setup based PoR schemes for examples PoR schemes proposed by (Chauhan and Saxena,

2014), (Liu and Zic, 2015), (Shin *et al.*, 2015), (Kiraz *et al.*, 2015) and (Rashid *et al.*, 2015), compared to the others. Lastly, some PoR schemes may store client's data in the forms other than the two data forms mentioned previously, not limited to PoR schemes proposed by (Rass, 2013), (Husain *et al.*, 2014), and (Etemad and Küpçü, 2015), but the two forms mentioned in previous outnumber than the others. However, it seems to have no problem whether in which data form is more favorable to be stored in cloud storage servers, for those PoR schemes applied distributed servers' setup.

On the other hand, recovery is an important attribute of PoR worth concern. In term of data error recovery, one of the most popular technique is error correcting codes (ECC) as it consumes less computation time and resources such as storage space and memory compared to other data error recovery techniques. Moreover, due to its simplicity and lower computation cost, it is widely employed in PoR schemes not limited to those proposed by (Juels and Kaliski Jr., 2007), (Jin Li *et al.*, 2013), (Rass, 2013), (Juels *et al.*, 2015), and (Tiwari and Gangadharan, 2015). On the contrary, ECC generally greatly increase the size of data. For example, parity check codes which required each data bit to be assigned a parity bit for error checking, causing data size to increase greatly. Another widely used recovery technique is erasure coding. It is a type of coding by which split data into pieces, encoded with redundant data, and then stored across distributed storage servers. By comparison, it contributes to lesser increment in data size, usually about 50% increase in data size, compared to ECC and replication. As a result, in present there are many PoR schemes being designed using erasure coding, for example PoR schemes proposed by (Shacham and Waters, 2008), (Shi *et al.*, 2013), (Armknecht *et al.*, 2014), (Cash *et al.*, 2015), (Du, Deng, Chen, He, and Zheng, 2015), and (Sengupta, Bag, Ruj, and Sakurai, 2016). In addition, network coding (NC) an efficient coding technique widely used in data transmission, is also assimilated in PoR schemes proposed by (Omote and Thao, 2015), (Thao and Omote, 2016), and (Omote and Tran, 2016) for error recovery. With respect to this, the main idea behind NC is by conducting exclusive OR (XOR) operation among data blocks to form a coded block. From the aspect of increment in data size, it is similar to erasure

coding, causing data to increase its size by around 50%. However, in term of efficiency, network coding is better than erasure coding. In case of data corruption, erasure coded data required an action of retrieval of full data before recovery can be applied, whereas NC coded data required only other healthy coded blocks which are constructed from the data blocks used to construct the corrupted coded blocks for error recovery. In addition, other recovery techniques including dispersal coding and Slepian-Wolf coding, etc. also have been adopted in PoR schemes proposed by (Mishra, Bhardwaj, and Kumar, 2015), (Wang *et al.*, 2015), (Lavauzelle and Levy-Dit-Vehel, 2016), and (Saxena and Dey, 2016). As there is no limitation for PoR schemes to apply only one recovery technique, hence there are some which have adopted more than one recovery techniques, not limited to PoR schemes proposed by (Song and Deng, 2013), (Omote and Tran, 2014), (Omote and Tran, 2015), and (Ren *et al.*, 2015).

The fifth attribute is the storage auditing. Generally, storage auditing is a method of verification in PoR scheme to check either the cloud storage servers are properly storing clients' data by asking the storage servers to provide proofs via PoR challenges. There are two types of storage auditing as follows:

i. private auditing conducted by data owners or shared data users, and
ii. public auditing conducted by authorized third party auditor (TPA).

For privacy concern, it is more preferable to employ private auditing where data is not exposed to any public personal. Meanwhile, as public auditing which require data auditing to be done by TPA, hence storing non-confidential data or encrypted sensitive data in cloud using PoR scheme that employed public auditing would also considered safe. Hence, it is quite objective to say auditing technique is better by looking in privacy aspect only, rather differ requirement would determine this. In fact, there are almost similar in number of PoR schemes adopting public auditing including those proposed by (Shi *et al.*, 2013), (Armknecht *et al.*, 2014), (Shin *et al.*, 2015), and

(Omote and Thao, 2015), whereas examples of private auditing such as those proposed by (Juels and Kaliski Jr., 2007), (Cash *et al.*, 2015), (Jianchao *et al.*, 2015), and (Vasilopoulos *et al.*, 2016). Meanwhile, only a few PoR schemes adopting both private and public auditing such as those proposed by (Shacham and Waters, 2008), (Huang *et al.*, 2014), and (Zhang, Tang, and Mao, 2014).

Next, the sixth attribute of PoR to include in the taxonomy is cryptography. Cryptography is a choice of securing privacy and ensuring confidentiality of stored data. Cryptographic techniques being reviewed here including encryption, hashing and others. Generally, encryption is one of the common cryptography approach used widely for preserving privacy of stored data, where data is translated into secret codes before storage take place, while secret keys are needed to read the encrypted data via decryption. There are two main encryption techniques employed:

i. symmetric encryption that uses the same key for both encryption and decryption process, and
ii. asymmetric encryption that uses different key for encryption and decryption.

In term of level of security, asymmetric encryption is stronger than symmetric encryption, but as a drawback, it needs more time to compute compared to symmetric encryption. Corresponding to application of encryption in PoR schemes, most of the work done have employed symmetric encryption not limited to those proposed by (Juels and Kaliski Jr., 2007), (Shi *et al.*, 2013), (Yuan and Yu, 2013a), (Omote and Tran, 2014), (Cash *et al.*, 2015), (Wang *et al.*, 2015), and (Sengupta *et al.*, 2016). On the other hand, although the frequency of adoption is not as large as encryption, hashing is another choice of cryptography approach to apply in PoR schemes. Generally, hashing is a cryptographic function which transform data into a shorter fixed-length value or key, for instance checksum and digital fingerprint. Some

examples of PoR schemes which have adopted hashing for the stored data are those proposed by (Jin Li *et al.*, 2013), (Rass, 2013), (Husain *et al.*, 2014), (Huang *et al.*, 2014), and (Tiwari and Gangadharan, 2015). However, as confidentiality falls behind availability and integrity which served as the precondition for the existence of a cloud storage system, hence it is not necessary to include a cryptography protection to stored data in cloud. Some PoR schemes are designed without involvement of any cryptography technique, for their performance and efficiency concern. In fact, there are several PoR schemes proposed in recent that have adopted none of cryptography approach, for examples PoR schemes proposed by (Thao *et al.*, 2014), (Zhang *et al.*, 2014), (Etemad and Küpçü, 2015), and (Du *et al.*, 2015).

Lastly, the seventh attribute of PoR to illustrate in the taxonomy is experimentation and analysis. To keep up with the evolution of technology and requirements in our lives, data storage method has been changed over time, similarly PoR schemes as well. Due to the needs of data clients in various aspects, PoR schemes have been proposed by researchers up to now spring up like mushrooms. Consequently, it turns into a situation where CSP has to choose one of so many PoR schemes available which perform the best, and best suited to their business objectives and clients' needs. Hence, in order to choose the most suited PoR scheme to adopt into cloud storage system, performance comparison is the best and common way. For this purpose, performance analysis is a norm in research field and papers for any new proposal of technique.

There are many methods can be used for showing, proving and comparing the performance of proposed PoR schemes. Some widely used experimentation and analysis methods for PoR schemes to show and compare their performances including analytical solution, simulation, and prototype. On the first hand, analytical solution is a method of showing the performance of proposed and compared schemes, by giving a general description about the performance of the schemes for any value of parameters (Bernie, 1999). On the second hand, simulation is also a performance showing method

commonly used, but it different with analytical solution by which it is a process of imitation of the proposed scheme in a real-world process over time with specified parameters as described by (Sahin, 2006) and (Maria, 1997). On the other hand, prototype is a preliminary product of a scheme designed to collect more experimental or testing data before a better version of the schemes could be implemented (Christie, Jensen, Buckley, Menefee, Ziegler, Wood and Crawford, 2012). Depends on few factors, such as precision and accuracy of complexity analysis, compatibility and viability of simulation in real cloud environment, feasibility of prototype, determining the most trustworthy performance proving and comparing method for PoR schemes is very difficult. Indeed, it is a very subjective question or topic to discuss. Nevertheless, the trend of experimentation and analysis used in recent PoR schemes is obvious. As for PoR schemes' papers being reviewed in this section, obviously analytical and simulation approach are more or less similar in their use frequency, whereas prototype and other methods are less favorable. In addition, it is very infrequent for researchers to show and compare the performance of their proposed PoR schemes via more than one method.

In summary, PoR schemes can be defined and categorized based on the seven attributes as described in the taxonomy. From the taxonomy, it is found that the trend of PoR schemes construction is moving towards to dynamic data nature because dynamic PoR suits not only dynamic data, but it is compatible to static data which requires no update as well. Furthermore, distributed servers' setup is more prominent compared to single server's setup for the reasons of data corruption resiliency and backup.

In addition, any data form to store in cloud servers seems to have no issue in PoR schemes which employed distributed servers' setting. However, data stored in the form of coded blocks and metadata or tags is considered more secure because data is not stored exactly in the same form. For example, data '1100' is coded and stored as '1111', requires malicious adversary to work harder to retrieve the data. In term of

recovery, although erasure coding is leading the trend at the moment, but network coding might be a good choice for PoR construction in the future, due to resource and computation efficiency of network coding in data recovery process compared to erasure coding.

For storage auditing, although it is very difficult to tell which one is more prominent between public and private storage auditing. Nevertheless, it is favorable if both public and private auditing are made selectable in a PoR scheme to fulfill the wide variety needs of different users worldwide, for instances some users concern about privacy, whereas some busy users may need TPA to help in data auditing. Meanwhile, whether to include cryptography in PoR schemes, it depends to researchers to choose either efficiency and security, as it is a give and take. Nevertheless, the literature had reveal that most PoR schemes do provide a minimum level of security using symmetric encryption. Lastly, it is easier for researchers to compare their works with the others using analytical method for experimentation and analysis towards efficiency of PoR schemes as this can clearly shows the differences from readers view.

2.3.2 Issues of PoR with Respect to Cloud Storage

In the previous sub-section, the reasons behind why PoR schemes are needed in cloud storage were discussed. In this sub-section, issues related to PoR schemes with respect to cloud storage are pointed out and discussed.

Although PoR schemes could be implemented to ensure data availability and data integrity of stored data in cloud servers, several issues arise such as efficiency, supportability of devices, malicious threats, and data deduplication issues.

First of all, the efficiency is the first issue to discuss, specifically computational, storage and communication efficiency of PoR schemes with respect to the stored data in cloud storage servers (Zafar , Khan, Malik, Ahmed, Anjum, Khan, Javed, Alam and Jamil, 2017). Generally, data integrity schemes especially PoR schemes, require data preprocessing before the data is outsourced to cloud storage servers, in order to ensure integrity and availability of stored data. The main drawback of data preprocessing is time and resource consuming. Therefore, a cloud storage system which has implemented a PoR scheme would suffers from slower data storing process than the other which has not, due to additional data preprocessing such as employing erasure codes that is time consuming before storing the data into servers. Therefore, it is crucial for a PoR scheme to have a computational and communicational efficient construction. Meanwhile, the storage efficiency is also an important factor to consider, as the digital data growth rate is always exponential (Zafar *et al.*, 2017). This is also a reason why replication of full data across distributed cloud servers (Amazon, 2017) is no longer a suitable and applicable data storage method, thus further emphasizing the importance and reason of PoR schemes should be adopted.

With the emergence of Internet, electronic devices like laptops, smart phones and tablets may needed to integrate and connect to the Internet, and so to cloud storage services. For instance, we may need our portable electronic smart devices such as smart phones or tablets to have access to our cloud storage account, working on outsourced storing documents using these devices which are always resource constrained. Thus, it is a challenge to design a lightweight data auditing scheme, specifically PoR scheme for mobile devices which are resource limited (Lin, Shen, Chen, and Sheldon, 2017) (Sookhak, Talebian, Ahmed, Gani, and Khan, 2014). Even though there have been people working on this, for example in (Jin Li, Tan, Chen, Wong, and Xhafa, 2015), the researchers have proposed a lightweight data auditing scheme for resource constraint devices such as smart phones and tablets. However, the scheme (Jin Li *et al.*, 2015) itself requires a more efficient construction for less storage requirement with lower communication cost, according to researchers.

On the other hand, PoR schemes are vulnerable to cyber malicious threats and attacks which cause data losses. Examples of PoR related cyber malicious threats may include, but not limited to tag forgery attack (Zafar *et al.*, 2017) (Jeevitha, Chandrasekar, and Karthik, 2015) where malicious cloud storage servers attempt to hide stored data damage and bypass auditing process; data deletion attack (Zafar *et al.*, 2017) in situation where only tags are needed for proof generation rather than data itself; and replace attack (Zafar *et al.*, 2017) where corrupted or deleted stored data block and tag pairs are replaced with other valid pairs so to pass data auditing. In addition, malicious storage servers may attempt to cache responses of precious passed auditing challenge to be replayed in future auditing (Ren, Wang, Wang, and Xu, 2015) (Thao and Omote, 2016). Not limited to these attacks, there is possibility where a malicious storage server may try to act dishonest via pollution attack (Ren *et al.*, 2015) (Zafar *et al.*, 2017), where it passes the auditing process with valid data, but providing corrupted stored data blocks during repair phase to construct a faulty new data blocks instead of recovery. Besides, another malicious threat to mention is data leakage attack (Zafar *et al.*, 2017), where malicious cloud storage servers attempt to extract stored data when verification is using wiretapping. Nevertheless, a suggested provided by (Zafar *et al.*, 2017) such that data block and metadata pairs should be constructed in a way that they have strong binding with each other, while proof generation for data auditing should involves both data block and metadata pairs as well as randomness factor in PoR challenge-response mechanism.

The last issue of PoR worth mentioned here is data deduplication. Data deduplication is a process of eliminating redundant data copies stored in the cloud in order to save cloud storage spaces (Dell EMC, 2017). Data deduplication is commonly adopted in PoW. Meanwhile, PoR is making redundant copies at data blocks level to provide recovery and retrievability. Hence, in general, PoR schemes are contradicting with the nature of cloud data deduplication as PoR tends to create redundant data while PoW tends to eliminates redundant data via data deduplication. Worth to mention, there have been a few PoR schemes proposed in recent years to integrate

PoR schemes with data deduplication where PoR can coexist with PoW, for example (Yuan and Yu, 2013b), (Shin, Koo, Hur, and Yun, 2015), (Rashid, Miri, and Woungang, 2015), and (Vasilopoulos *et al.*, 2016). However, the work done on integrating PoR with deduplication for PoW is very limited. To the best of of the author's knowledge, there is still a left future work for PoR schemes which has integrated with deduplication for PoW, to be enabled dynamic operations.

2.4 Slepian-Wolf Based Proof of Retrievability (SW-PoR)

SW-PoR is a PoR scheme proposed by (Thao *et al.*, 2014) to address several limitations which most of network coding based PoR schemes have not covered. These limitations are listed in the following:

i. Newly constructed coded block is not exactly the same form as the corrupted one,
ii. the size of coded block is larger or equal to the size of each file block, and
iii. although efficiency is always a focus, but most of network coding based PoR schemes as reviewed and mentioned by (Thao *et al.*, 2014) still suffers from high storage, computation, and communication costs.

With respect to the limitations aforementioned, the SW-PoR scheme has contributed to smaller size of data to store in cloud servers compared to most of network coding based PoR schemes via storing data in the form of coded block and metadata pairs. Besides, in SW-PoR scheme, the newly constructed coded block and metadata pair is the exact repair to the corrupted one, which make it outstanding compared to most of the network coding PoR schemes as additional overheads become unnecessary when exact repair take place. Moreover, SW-PoR has also contributed to allow lower storage, communication, and computation costs than most

of the network coding PoR schemes, which is very beneficial in the real cloud system when the original data size is huge.

A notable different of SW-PoR scheme from network coding based PoR scheme is that the form of a data taking, specifically coded block, to be stored in cloud servers. The different is that network coding based PoR schemes usually take the XOR of file blocks directly as the coded blocks to be stored in cloud servers, whereas in SW-PoR scheme, bin index which is smaller in size compared to the XOR constructed blocks, its previous form. Bin index is the index where the constructed XOR block lies in a list of permutation, given the metadata as number of bit '1' in the XOR constructed block served to construct the list of permutation.

In brief, there are mainly three functions in SW-PoR scheme, namely Encode, Retrieve and Repair functions. These functions work as storing, retrieving and repairing client's data respectively. Each is described in Sections 2.4.1, 2.4.2 and 2.4.3 respectively. The performance discussion is presented in Section 2.4.4.

2.4.1 Encode Function

In SW-PoR scheme, to store a file to cloud server, it has to undergo encoding process, changes the form it takes, and lastly being stored in cloud servers. The whole process is handled by the Encode function of SW-PoR.

When a file is uploaded by data client to be stored in cloud servers, the file is firstly being split into a number of file blocks, m, of equal size. Then, the encoding process takes place in which three file blocks out of all the file blocks, $\overline{w}_i$, are chosen, encoded into a single block while the size is equal to a file block, via XOR operation among the three selected file blocks. The encoding process of SW-PoR continues until

all file blocks are involved and has reach a sufficient number of XOR constructed blocks, such that using this number of XOR constructed blocks, the stored data can be fully retrievable.

Then, these XOR constructed blocks are converted into coded block and metadata pairs before they are distributed to be stored across cloud servers. To convert a XOR constructed block into pair of coded block and metadata, firstly the number of bit '1' in the XOR constructed block is calculated to be stored as metadata. Then, coded block, specifically the index or frankly the position of the XOR constructed block lies in the list of permutation constructed using two pieces of information, namely the block length or block size, and the metadata.

One thing to take note here is that, the number of bits used to represent coded block and metadata pairs in binary form are not the same to represent the file blocks. Number of bits needed to represent a coded block is calculated using the formula $\log_2 /P_i/$ where the logarithm of base 2 is needed for binary representation, and binomial coefficient formula $/P_i/ = \binom{block\ length}{\hat{c}_0}$ to calculate the number of elements in the list of permutation.

After all the XOR constructed blocks have been converted to pairs of coded block and metadata, they are then stored across distributed servers, and the Encode function ends here. For a better understanding to the Encode function as described, Figure 2.3 is presented to illustrate the conceptual view of the Encode function in SW-PoR scheme.

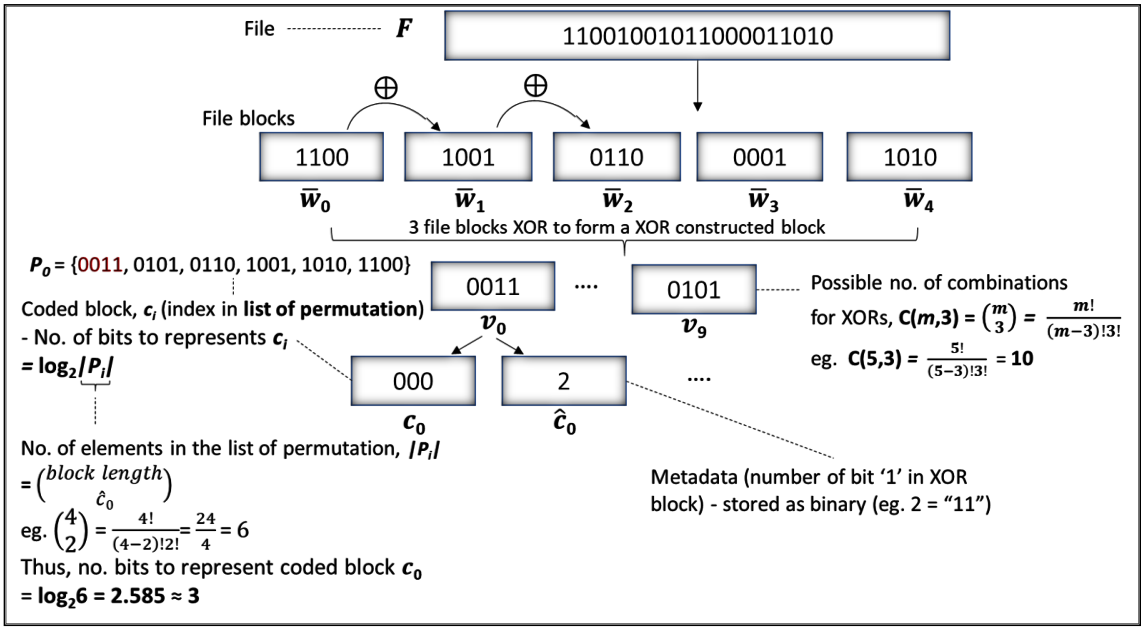

Figure 2.3: Conceptual view of SW-PoR Encode Function

2.4.2 Retrieve Function

In SW-PoR scheme, to retrieve a file from cloud server, it has to undergo decoding process by convert back from the form of coded block and metadata pairs to XOR constructed blocks, and then go through Gaussian Elimination to revert the constructed XOR blocks into their respective file blocks, joined up to get back the original file to be downloaded by data client. The whole process is handled by the Retrieve function of SW-PoR.

When a stored file is requested by data client to download, cloud servers have to provide a number of coded block and metadata pairs equal to the total number of file blocks, *m*, split from the original file. The pairs of coded block and metadata are chosen in such a way that the coefficient vectors of their form in XOR constructed blocks can construct a binary square matrix which has full rank. Here, the coefficient vectors mean the binaries, where indices of file blocks being used to construct that XOR constructed block as bit '1' and otherwise as bit '0'. After a number of coded block

and metadata pairs has been chosen, they are converted back to XOR constructed blocks. Then, a full rank square matrix is constructed using the coefficient vectors of these XOR constructed blocks.

Next, a column consists of the XOR constructed blocks translated from their coded block and metadata pairs is added to the matrix as the last column. Lastly, the matrix is then undergoing Gaussian elimination, being reduced to identity matrix with additional column at the end of it served as file blocks in ascending order. As additional information to this, Gaussian elimination is an algorithm for solving systems of linear equations. Then, the resolved matrix is extracted its last column, having each binary element of the column joining up as file blocks ascendingly to form back the original file.

To achieve a clearer and easier to understand the Retrieve function as described, Figure 2.4 is presented to illustrate the conceptual view of the Retrieve function in SW-PoR scheme.

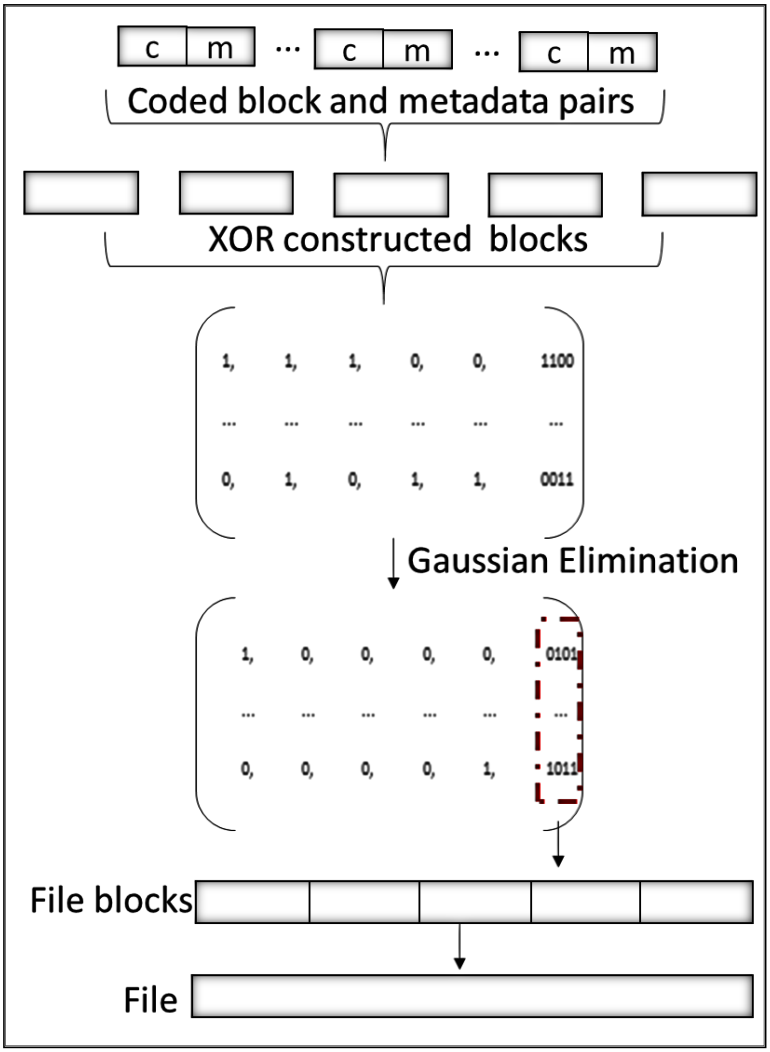

Figure 2.4: Conceptual view of SW-PoR Retrieve Function

2.4.3 Repair Function

In SW-PoR scheme, to repair corrupted pair of coded block and metadata, it is repaired by SW-PoR using three healthy pairs of coded block and metadata stored in cloud servers. The whole process is handled by the Retrieve function of SW-PoR.

To repair a corrupted pair of coded block and metadata, indices of the three operands which make up the XOR of the corrupted pair of coded block and metadata

are identified, where the operands mean file blocks. For example, if the coded block and metadata pair is converted from a XOR constructed block which is constructed using file block indexed zero, file block indexed one, and file block indexed two, then the indices are zero, one, and two. Then, from a list of number ranged from {0, m-1}, where m represents the number of file blocks, two numbers have to be chosen in such a way that they are not any of the three operands (file blocks) of the coded block and metadata pair. At this point, five numbers required for corruption recovery have identified.

Next, these five numbers are used in the corruption repair phase, where servers storing XORs which made from these five numbers as operands are identified. Servers which are storing the selected three pairs of coded block and metadata are required to provide them to repair the corruption. Then, these three pairs of coded block and metadata are converted back to their respective XOR constructed blocks. After that, the three XOR constructed blocks undergo XOR operation with each other to produce a new XOR constructed block. It is then converted to coded block and metadata pair to replace the corrupted one. The newly constructed pair of coded block and metadata is exactly the same as the corrupted one when it is healthy. The conceptual view and technical view of the Repair function are shown in Figures 2.4 and 2.5 respectively.

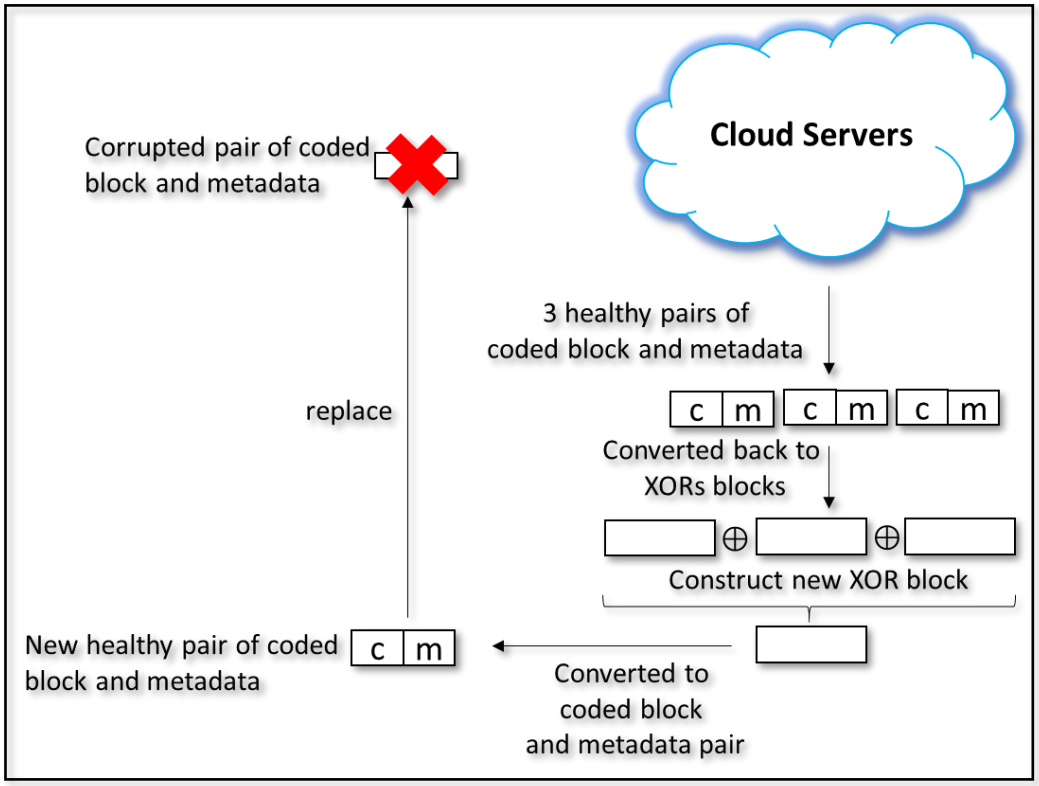

Figure 2.5: Conceptual view of SW-PoR Repair Function

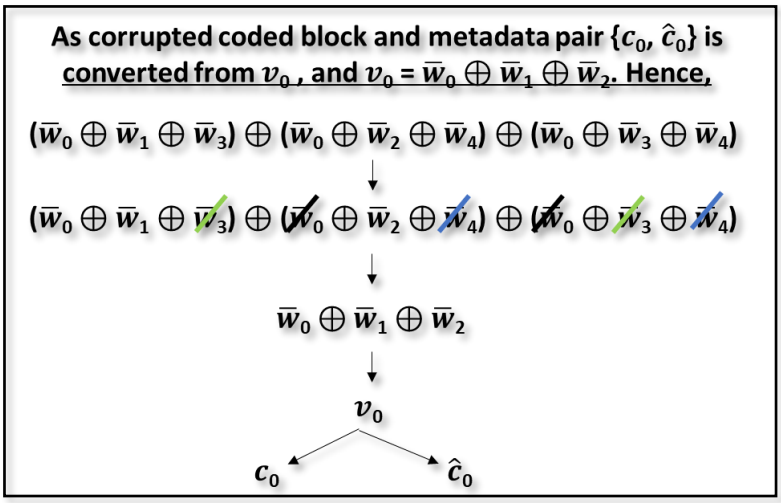

Figure 2.6: Technical view of SW-PoR Repair Function

2.4.4 Preliminary Simulations, Computational Performances and Discussions

This section describes about simulation done by (Thao *et al.*, 2014) for SW-PoR scheme using predefined settings, which will be described in this section as well. In this thesis, a preliminary simulation is conducted as a test run to mimic the simulation done by (Thao *et al.*, 2014), in order to ensure a thorough understanding about Encode, Retrieve, and Repair functions in SW-PoR scheme. This preliminary simulation in this thesis is conducted using the similar range of data size, except with an additional iteration of simulation using a larger size of data.

Then, settings used by (Thao *et al.*, 2014) in their simulation for SW-PoR scheme are explained, followed by description on relevant setting to be used in SW-PoR scheme for fair redundancy, and lastly clarification on expected performance and drawback of the relevant setting to be used in SW-PoR scheme for fair redundancy.

(A) Simulation of SW-PoR Scheme in Previous Work

Simulation of SW-PoR scheme is conducted (Thao *et al.*, 2014) using a machine with specification as follows: Intel Core i5 processor, 2.4 GHz, 4 GB of RAM, and Windows 7 (64-bit OS). Each file block in the simulation is set to 2^{10} bits or equivalent to 1024 bits. The simulation is conducted across different data sizes (number of file blocks, m) ranged from 50 file blocks to 150 file blocks. The number of XOR constructed blocks, similarly the number of pairs of coded block and metadata, n, to be constructed are set to m+1. Meanwhile, the required number of XORs, $n = m+1$ are chosen ascendingly from the list of all possible XOR combinations, $\{\overline{w}_0 \oplus \overline{w}_1 \oplus \overline{w}_2, \ldots, \overline{w}_0 \oplus \overline{w}_1 \oplus \overline{w}_{m-1}\}$, $\{\overline{w}_0 \oplus \overline{w}_2 \oplus \overline{w}_3, \ldots, \overline{w}_0 \oplus \overline{w}_2 \oplus \overline{w}_{m-1}\}$, …, $\{\overline{w}_1 \oplus \overline{w}_2 \oplus \overline{w}_3, \ldots, \overline{w}_1 \oplus \overline{w}_2 \oplus \overline{w}_{m-1}\}$, …, $\{\ldots, \overline{w}_{m-3} \oplus \overline{w}_{m-2} \oplus \overline{w}_{m-1}\}$. All the three functions of SW-PoR scheme, namely Encode, Retrieve, and Repair functions are simulated with the defined setting. The result of simulation conducted by (Thao *et al.*, 2014) is shown in Table 2.2.

Table 2.2: Simulation result of SW-PoR scheme (Thao *et al.*, 2014)

No. of file blocks (*m*)	**Computation time of SW-PoR scheme (seconds, s)**		
	Encode	**Retrieve**	**Repair**
50	0.25	0.53	0.0314
75	0.37	0.83	0.0310
100	0.43	1.11	0.0312
125	0.59	1.51	0.0315
150	0.67	1.76	0.0310

As shown in Table 2.2, both Encode and Retrieve functions have shown a pattern of linear increment of computation time when data size is increasing. Meanwhile, the Repair function has shown a pattern of constant computation time across data sizes. Nonetheless, the Encode and Retrieve functions seems to take only a small amount of time to complete the computations using the setting of $n = m+1$ in choosing XORs for encoding.

(B) Preliminary Simulation for SW-PoR Scheme

For a better understanding about SW-PoR scheme and the setting used by (Thao *et al.*, 2014) in their simulation, a similar simulation is conducted and presented in this thesis as well. The only different is that simulation is conducted until number of file blocks, m reaches 175 for all three functions of SW-PoR. Same setting is used for this simulation, including the number of bits per file block, and the setting of $n = m+1$ in choosing XORs for encoding as well. In this simulation, a machine with specification as follows is used: Intel Core i5 processor, 2.50 GHz, 8 GB of RAM, and Windows 10 (64-bit OS). The result of preliminary simulation conducted for this thesis is shown in Table 2.3. For the ease of comparison and referring, simulation conducted by (Thao *et al.*, 2014) will

be indicated as Simulation A, whereas the simulation conducted in this thesis will be indicated as Simulation B.

Table 2.3: Comparison of preliminary simulation of SW-PoR scheme with simulation conducted by (Thao *et al.*, 2014) using similar setting

No. of file blocks (*m*)	**Computation time of SW-PoR scheme (seconds, s)**					
	Encode		**Retrieve**		**Repair**	
	A	**B**	**A**	**B**	**A**	**B**
50	0.25	0.27	0.53	0.57	0.0314	0.032
75	0.37	0.40	0.83	0.88	0.0310	0.035
100	0.43	0.51	1.11	1.25	0.0312	0.035
125	0.59	0.62	1.51	1.60	0.0315	0.035
150	0.67	0.75	1.76	1.96	0.0310	0.036
175	0.25	0.84	0.53	2.55	0.0314	0.035

As shown in Table 2.3, the preliminary simulation of SW-PoR, Simulation B conducted for this thesis shows similar result with the Simulation A conducted by (Thao *et al.*, 2014). In this simulation, both the Encode and Retrieve functions have shown linear increment of computation time when data size is increasing, whereas Repair function consumed about constant computation time regardless of different data sizes.

As the major part of process SW-PoR Encode function are the constructions of XOR constructed blocks and coded block and metadata pairs, hence the major portion of computation time used in simulating Encode function comes from these two processes. Both the simulation conducted by (Thao *et al.*, 2014) and the preliminary simulation conducted for this thesis were using the setting of $n = m+1$ (a linear variable) in choosing XORs for encoding which has resulted a linear increment in computation time when data size increases. Meanwhile, as the Retrieve function requires the same number of coded block and metadata pairs as the total number of

file blocks split in Encode function earlier, hence the number of pairs of coded block and metadata needed, $n = m$, which is also a linear variable. Due to this factor, computation time for retrieving a file from cloud servers with SW-PoR would be also linear across data sizes. Lastly, as repairing a pair of corrupted coded block and metadata, three pairs of healthy coded block and metadata are needed, $n = 3$, which is a constant, hence the computation time of Repair function across data sizes is also a constant as proven in both simulations.

(C) Drawback of the Setting of SW-PoR Used by the Authors

The current SW-PoR scheme (Thao *et al.*, 2014) is using the setting where the number of XORs required, $n = m+1$ for encoding while choosing them from the list of all possible XOR combinations, C (m, 3), in ascending order. However, this setting creates imbalanced redundancy of file blocks across the output of Encode function, coded block and metadata pairs. For a better understanding about the drawback of the setting used in current SW-PoR scheme, *Example 1* is provided as follow:

Example 1:

A file of 20 bits in size is split into 5 file blocks, $m = 5$, of 4 bits each during encoding in SW-PoR scheme, $\overline{w}_0$, $\overline{w}_1$, $\overline{w}_2$, $\overline{w}_3$, and $\overline{w}_4$. Hence, the possible number of combination to choose XORs is n = C (m, 3) = C (5, 3) = 10, where C (n, r) or ${}^{n}\mathrm{C}_{r}$ (given n, takes r) is the binomial coefficient or commonly known as combination in mathematics context. However, the setting used in current SW-PoR, will only takes $n = m+1 = 5+1 = 6$ XORs out of 10. Table 2.4 compares n = C (m, 3) and $n = m+1$ settings in term of combination of XORs.

Table 2.4: Combination of XORs taken for settings n = C (m, 3) versus n = m+1

	Settings	
Chosen XORs for encoding	**n = C (m, 3)**	**n = m+1**
	$\overline{w}_0 \oplus \overline{w}_1 \oplus \overline{w}_2$	$\overline{w}_0 \oplus \overline{w}_1 \oplus \overline{w}_2$
	$\overline{w}_0 \oplus \overline{w}_1 \oplus \overline{w}_3$	$\overline{w}_0 \oplus \overline{w}_1 \oplus \overline{w}_3$
	$\overline{w}_0 \oplus \overline{w}_1 \oplus \overline{w}_4$	$\overline{w}_0 \oplus \overline{w}_1 \oplus \overline{w}_4$
	$\overline{w}_0 \oplus \overline{w}_2 \oplus \overline{w}_3$	$\overline{w}_0 \oplus \overline{w}_2 \oplus \overline{w}_3$
	$\overline{w}_0 \oplus \overline{w}_2 \oplus \overline{w}_4$	$\overline{w}_0 \oplus \overline{w}_2 \oplus \overline{w}_4$
	$\overline{w}_0 \oplus \overline{w}_3 \oplus \overline{w}_4$	$\overline{w}_0 \oplus \overline{w}_3 \oplus \overline{w}_4$
	$\overline{w}_1 \oplus \overline{w}_2 \oplus \overline{w}_3$	-
	$\overline{w}_1 \oplus \overline{w}_2 \oplus \overline{w}_4$	-
	$\overline{w}_1 \oplus \overline{w}_3 \oplus \overline{w}_4$	-
	$\overline{w}_2 \oplus \overline{w}_3 \oplus \overline{w}_4$	-

Using the setting of full redundancy, n = C (m, 3), each of the five blocks ($\overline{w}_0$, $\overline{w}_1$, $\overline{w}_2$, $\overline{w}_3$, and $\overline{w}_4$) can be found in six XOR constructed blocks respectively, where redundancy of file blocks is fair and balance. Meanwhile, using the setting of n = m+1, $\overline{w}_0$ can be found in six XOR constructed blocks, but each of the remaining file blocks, $\overline{w}_1$, $\overline{w}_2$, $\overline{w}_3$, and $\overline{w}_4$ can be found in only three XOR constructed blocks. Although the imbalanced redundancy seems to be less significant in this case as the number of file blocks of a file is small, but for a large file this imbalanced redundancy can be seen very clearly and the impact can be very significant. This issue is presented in *Example 2*.

Example 2:

In this example, a larger file size is used, where number of file blocks, m = 10, from $\overline{w}_0$ to $\overline{w}_9$. For the setting of n = m+1, only 11 XORs out of all possible combinations (120) are constructed which are listed as follows:

i. $\overline{w}_0 \oplus \overline{w}_1 \oplus \overline{w}_2$

ii. $\overline{w}_0 \oplus \overline{w}_1 \oplus \overline{w}_3$

iii. $\overline{w}_0 \oplus \overline{w}_1 \oplus \overline{w}_4$

iv. $\overline{w}_0 \oplus \overline{w}_1 \oplus \overline{w}_5$

v. $\overline{w}_0 \oplus \overline{w}_1 \oplus \overline{w}_6$

vi. $\overline{w}_0 \oplus \overline{w}_1 \oplus \overline{w}_7$

vii. $\overline{w}_0 \oplus \overline{w}_1 \oplus \overline{w}_8$

viii. $\overline{w}_0 \oplus \overline{w}_1 \oplus \overline{w}_9$

ix. $\overline{w}_0 \oplus \overline{w}_2 \oplus \overline{w}_3$

x. $\overline{w}_0 \oplus \overline{w}_2 \oplus \overline{w}_4$

xi. $\overline{w}_0 \oplus \overline{w}_2 \oplus \overline{w}_5$

From the list shown in *Example 2*, we can see that $\overline{w}_0$ can be found in 11 XORs; $\overline{w}_1$ can be found in eight XORs; $\overline{w}_2$ can be found in four XORs; each of $\overline{w}_3$, $\overline{w}_4$ and $\overline{w}_5$, can be found in two XORs; however, each of $\overline{w}_6$, $\overline{w}_7$, $\overline{w}_8$ and $\overline{w}_9$ can only be found in one XOR. Since each of $\overline{w}_6$, $\overline{w}_7$, $\overline{w}_8$ and $\overline{w}_9$ can only be found in only one XOR, hence they do not have any resiliency towards data corruption as they cannot be recovered from data corruption. Therefore, if data corruption occurs on any of these XOR constructed blocks, the whole file cannot be recovered, and will become irretrievable. Hence, the way of choosing XORs with the setting $n = m+1$ used in current SW-PoR contradicts with the definition of PoR which ensure 100% retrievability. Rather, the setting of n = C (m, 3) which provides maximum redundancy and resiliency should be used in SW-PoR to prevent data irretrievability issue as shown in the example above.

(D) Fair Redundancy with Respect to Data Retrievability in SW-PoR Using the Setting of Binomial Coefficient, C (m, 3) and Its Drawback

Using the setting of binomial coefficient, C (*m*, 3) to encode a file in SW-PoR creates fair redundancy, unlike the setting of $n = m+1$ used by (Thao *et al.*, 2014) which will creates unfair and unbalance redundancy that has a problem of data irretrievability. However, using this setting, the possible number of combination of XORs increases exponentially when data size increases. This is because C (*m*, 3) is an exponential variable. Hence, even with a slight increase in data size, the possible number of combination of XORs does not increase linearly, rather it increases exponentially, as shown in Table 2.5.

Table 2.5: Relation of *m* and *n* when *n* = C (*m*, 3)

m	***n* = C (*m*, 3)**
10	120
20	1140
30	4060
40	9880
50	19600

As the SW-PoR encoding composes majorly the constructions of XOR constructed blocks and coded block and metadata pairs as mentioned, the computation time of encoding a file with SW-PoR using the setting of C (*m*, 3) is greatly influenced by this factor. As data size increases, number of file blocks increases as well, which shows a directly proportional relation. However, as file size increases, the number of XORs to construct, and also the number of pairs of coded block and metadata to be converted from these XOR constructed blocks will increase exponentially due to the exponential variable, C (*m*, 3). Therefore, an exponential increment in Encode function computation time would be expected when data size increases.

2.5 Conclusion

Cloud storage data client to has been introduced as an alternative or even a replacement to local storage. Through cloud storage, management procedure and maintenance cost from data client side can be reduced. Nevertheless, cloud storage requires serious concern about the integrity of outsourced data. As solutions, many data integrity schemes especially PoR schemes have been proposed by researchers to ensure data availability and data integrity. This chapter has presented the literature on state-of-the-art of PoR schemes, published in 2013 to 2016. In addition, issues and vulnerabilities of cloud storage and PoR schemes have been identified as well, together with the countermeasures respectively. Besides, a taxonomy for recent PoR schemes is presented and summarized in previous section. Lastly, the variation of PoR, SW-PoR scheme has been described in detail.

In summary, the current SW-PoR which is using the setting of $n = m+1$ in choosing the XORs for encoding has a problem of unbalanced redundancy across file blocks. Moreover, this problem will cause file irretrievability issue which is contradicts with the definition of PoR itself for ensuring 100% file retrievability. This problem can be solved by taking all possible combination of XORs with the setting of C (m, 3). Unfortunately, the setting of C (m, 3) will cause exponential increment in encoding time. Therefore, in this thesis, two viable solutions to the identified issues are introduced and described in Chapter 3 and 4.

CHAPTER 3

Partial Binary Encoding for SW-PoR (PBE-SW-PoR)

3.1 Chapter Organization

This chapter presents the first approach to an efficient SW-PoR named Partial Binary Encoding for SW-PoR (PBE-SW-PoR). PBE-SW-PoR addresses the limitation of SW-PoR encoding by means of reducing the number of file blocks involved in encoding. Section 3.2 describes the details of PBE-SW-PoR. Section 3.3 concludes this chapter.

3.2 Partial Binary Encoding for SW-PoR (PBE-SW-PoR)

As described in Section 2.4.4, the setting of $n = m+1$ used by (Thao *et al.*, 2014) is less appropriate due to imbalanced redundancy. Balanced and maximum redundancy can be achieved using the setting of C (m, 3), but it will cause exponential increment in encoding time that is considerably high when data size increases gradually. Similarly, when data size decreases gradually, SW-PoR encoding time decreases exponentially. Hence, the straight forward solution to reduce overall data storing computation time is by reducing the size of the data to be encoded in SW-PoR. Sections 3.2.1 to 3.2.4 describe the conceptual model, computational overhead, storage cost, and resiliency of PBE-SW-PoR respectively.

3.2.1 Conceptual Model

The idea of PBE-SW-PoR was found from the assumption that computational time of SW-PoR encoding could be reduced by decreasing the size of data to be encoded. One method to achieve this is by using data splitting. In PBE-SW-PoR, data is first split into two parts, *A* and *B.* Only *A* will be encoded while *B* will be stored in cloud servers without passing through SW-PoR encoding function. Encoded data by SW-PoR is ensured to be fully protected against corruption, hence it has no issue on data corruption and recovery. However, *B* that is not encoded needs another recovery mechanism against corruption. In this case, replication (Kalkal and Malhotra, 2015) is selected to provide backup of *B*. This is because replication is one of the simplest and fastest data backup method for disaster recovery, as described in Section 2.2.1.

Another issue raised, with respect to *B*, is error checking to detect error on the data. To address this, error-detecting code in the form of Cyclic Redundancy Check (CRC) is used. CRC is a common technique used in data transmission for error checking. During data transmission, CRC bits are added to the transmitted packets and act as error-detecting codes. These codes are used to check whether the data is having any error when reaching destination (Sang, 2015). Generally, CRC is implemented in such a way that a pre-agreed polynomial is used to construct a divisor for CRC operation. Then, raw data is XOR divided by the divisor. At the end of the XOR division, the remainder obtained is then added to the ending of raw data as CRC bits before transmission takes place. An example of CRC operation and error checking is shown in Figure 3.1 with a data "11000110". The degree of divisor "111" is two, hence the CRC bits is two bits, having initial CRC bits of "00" added to the end of data bits, resulted into a dividend of "11000110**00**". Then, a XOR division is conducted to the dividend "11000110**00**" by the XOR divisor "111", resulted in getting a remainder "1". The remainder "1" is then added to dividend "11000110**00**", completing the CRC operation with an output of CRC bits added data "11000110**01**".

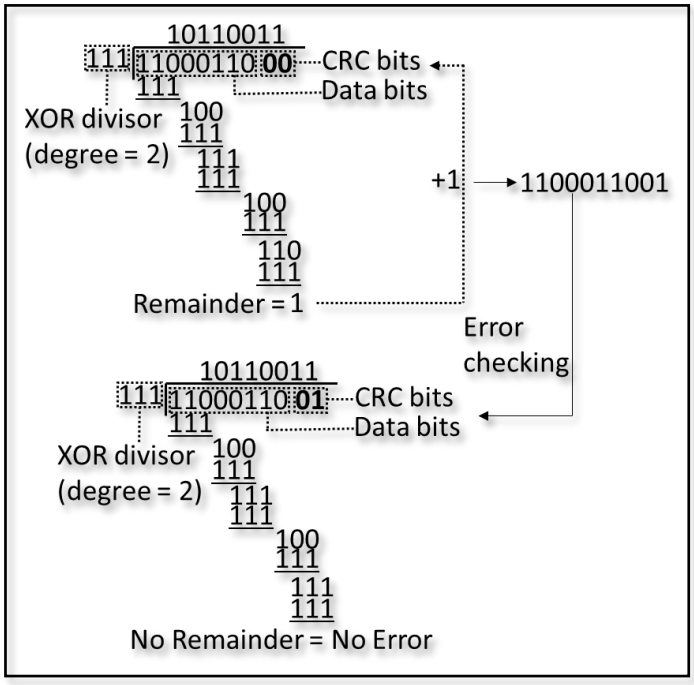

Figure 3.1: Example of CRC operation and error checking

To summarize the overall process, PBE-SW-PoR firstly split the data into two parts *A* and *B*; *A* will be encoded using SW-PoR and *B* is added with CRC bits for error checking before being replicated and stored in the cloud servers. The number of replications of *B* (with CRC-bits) is equal to the number of servers available; a minimum of two copies are required. To retrieve the stored data, *A* is decoded first and then it joined up with *B* which has its CRC-bits removed to form back the original raw data. The concept of the proposed PBE-SW-PoR is illustrated in Figure 3.2.

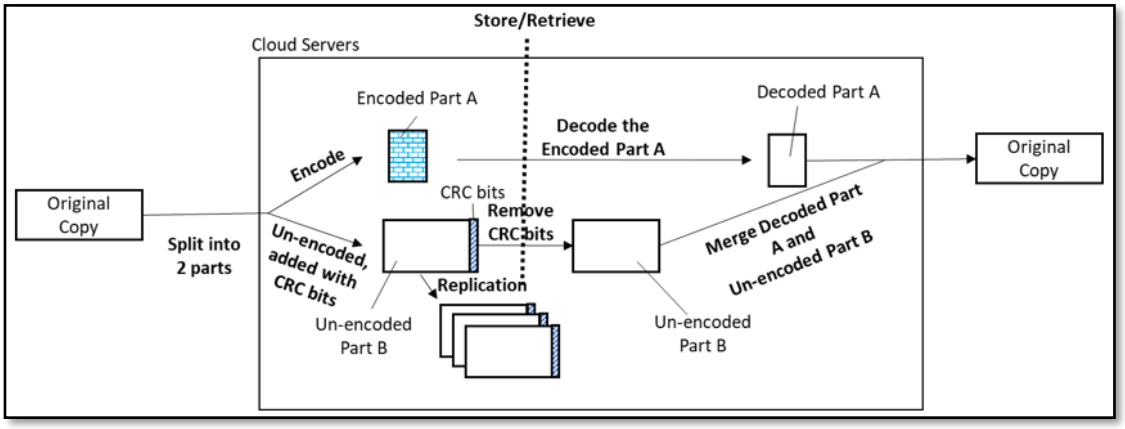

Figure 3.2: Conceptual model of PBE-SW-PoR

By using PBE-SW-PoR, the number of XORs, *n,* can be reduced to C (*q*, 3) where $q = \frac{m}{2}$, from the original SW-POR of C (*m*, 3). Although the value of *n* remains exponential, it is however significantly lower than the original SW-PoR. The significant reduction of number of XORs conducted in encoding leads to a shorter computation time and subsequently reduce the overall computation time for data storing process of SW-PoR.

Here is an example to illustrate the different between the original SW-PoR and PBE-SW-PoR in term of computational performance. Assume the same example as in Chapter 3, where a client has a file, *F,* of size 1,024,000 bits (approximately one Megabits) to store in the cloud, with the same setting including file block size of 2^{10} bits (1024 bits), the number of file blocks, *m* = 1,000 and the number of XORs required in original SW-PoR is *n* = C (*m*, 3) = C (1000, 3) = 166,167,000.

PBE-SW-PoR uses the same setting as in original SW-PoR for file size, block size, and number of file blocks. To ensure SW-PoR does not lost its role while having the aim of reducing size of data to be encoded in SW-PoR, simple split ratio of 1:1 would be a good choice (fair test case) to avoid bias test cases like 1:9. By using the simple

split ratio of 1:1, in which the data is split into *A* and *B* equally so that the time consumed by processing *A* (through SW-PoR) and *B* (through CRC operation and replication) can be compared fairly. This is because only using the split ratio of 1:1, where sizes of *A* and *B* are equal, a fair comparison can be conducted to see the difference in efficiency of processing *A* and *B*. For example, if the time used to process *B* is less than *A* when the size of *A* and *B* is equal, then increasing the size of *B* (and reducing the size of *A*) will further reduce the computation time. However, there is a tradeoff between computation time and redundancy per file block at *A*. This is because reducing the size of *A* may results in a lower computation time, but it also caused less number of redundancy per file block at *A*. Hence, it is important to ensure sufficient number of redundancy is preserved in *A*. It is however out of the scope of the work presented in this chapter to find the optimum split ratio in PBE-SW-PoR.

For presentation purpose, a file splitting ratio in PBE-SW-PoR of 1:1 is used in the remainder of this chapter. Using this splitting ratio, *B* only has half number of the total file blocks. With respect to this, the number of XORs required for the whole file is n = C (q, 3) = C ($\frac{m}{2}$, 3) = C ($\frac{1{,}000}{2}$, 3) = C (500, 3) = 20,708,500.

Compared to the number of XORs required in original SW-PoR and PBE-SW-PoR, 166,167,000 versus 20,708,500 respectively, a significant number of reductions of XORs can be seen. Similarly, it can be said that the number of XORs required in PBE-SW-PoR is reduced to about one-eighth compared to that of original SW-PoR. Hence, the computation cost can be reduced greatly in PBE-SW-PoR close to one order of magnitude.

For numerical estimation and comparison, the simulation result of (Thao *et al.*, 2014) is used. From the simulation, using the setting of $n = m+1$ as the number of XORs, a file consists of 50 file blocks requires 51 XORs, and the computation time

taken is 0.25 seconds. Excluding external factors like memory capacity etc. which is out of the scope of this thesis, by average each XOR requires a computation time of approximately 0.005 seconds. Take this number to fit in this example, the file *F* of size 1,024,000 bits which requires 166,167,000 XORs in original SW-PoR with the setting of C (*m*, 3) requires approximately about 830,835 seconds (230.8 hours) to complete the computation. Contrary, as PBE-SW-PoR computes about 20,708,500 XORs, hence it needs approximately 103,542.5 seconds (28.76 hours) to complete the computation. As a result, using PBE-SW-PoR theoretically can reduced greatly overall computation time. Table 3.1 compares the original SW-PoR with PBE-SW-PoR using analytical method.

Table 3.1: Analytic comparison of original SW-PoR vs PBE- SW-PoR

Schemes	**Original SW-PoR**	**PBE-SW-PoR**
File size	$\lvert F \rvert$	
Block size	$\lvert \overline{w} \rvert$	
No. of file blocks, *m*	$\frac{\lvert F \rvert}{\lvert \overline{w} \rvert}$	
No. of XORs required for whole file, *n*	C (*m*, 3)	C (*q*, 3), where q = $\frac{m}{2}$

3.2.2 Computational Overhead

In PBE-SW-PoR, there are a number of processes contributed to additional computation time overhead, including data splitting, CRC operation, and replication which are not required in SW-PoR. Data splitting is a one-shot-complete process of splitting a data into two parts for PBE- SW-PoR. As CRC operation is also conducted once only for *B*, hence the time consumed would be short, similar to that of data transmission. Meanwhile, CRC added *B* is replicated and stored simultaneously across cloud servers. Hence, replication process is counted only once regardless the number of servers available where each server stores a copy. The actual time consumed by

data splitting, CRC operation, and replication is described in the simulation presented in Chapter 5.

3.2.3 Storage Cost

The storage cost of data using SW-PoR is C (m, 3) x s, where $s = \log_2 \binom{|\overline{w}|}{\hat{c}} + \log_2 |\overline{w}|$ in bits. With respect to PBE-SW-PoR, using a 1:1 splitting ratio, A requires C ($\frac{m}{2}$, 3) x s, as only half of the data is encoded using SW-PoR. The storage cost for B, which is not encoded by SW-PoR, can be calculated by adding the number of CRC bits multiply by the number of copies of B replicated to be stored across the cloud servers. The Formula to calculate the storage cost for PBE-SW-PoR is shown in Table 3.2.

Comparison between the original SW-PoR and PBE-SW-PoR, with respect to storage cost, shows that PBE-SW-PoR requires lower storage cost when the number of servers is small. Similarly, if the number of servers is large, the storage cost of PBE-SW-PoR might be higher than the original SW-PoR, but this can be addressed by limiting the number of copies of B. Nonetheless, a minimum of two copies of B are necessary as at minimum one healthy copy of B is needed to repair the corrupted one. The summary is shown in Table 3.2.

Table 3.2: Storage cost of original SW-PoR vs PBE- SW-PoR

Schemes	**Original SW-PoR**	**PBE-SW-PoR**
Size of a pair of coded block and metadata, *s*	$\log_2 \binom{\lvert\overline{w}\rvert}{\hat{c}} + \log_2 \lvert\overline{w}\rvert$	
No. of pairs of coded block and metadata, *n*	C (*m*, 3)	C ($\frac{m}{2}$, 3)
Size of CRC bits	NA	*crc*
Number of servers	*serv*	
Total size of replicates of CRC added *B, rep*	NA	*crc* + $\frac{\lvert F\rvert}{2}$ x *serv*
Storage cost, *S*	C (*m*, 3) x *s*	C ($\frac{m}{2}$, 3) x *s* + *rep*

3.2.4 Resiliency

Resiliency of SW-PoR scheme can be found at the redundancy per file block which can be calculated by taking C (*m*, 3) multiply by $\frac{3}{m}$. By comparison, the part of data encoded in PBE-SW-PoR has the same resiliency as original SW-PoR due to the setting of C ($\frac{m}{2}$, 3), for 1:1 splitting. *B* which is stored using PBE-SW-PoR depends on the number of healthy replicates available and the time used to recover from corruption.

$$Redundancy\ per\ file\ block\ (r) = \binom{m}{3}\ x\frac{3}{m}\ or\ \text{C}\ (m, 3)\ \text{x}\ \frac{3}{m} \qquad \text{(Formula 3.1)}$$

As only half of data is encoded via SW-PoR, hence redundancy per file block of *A* in PBE-SW-PoR can be calculated using the Formula 3.1. Note that C (*m*, 3) is the maximum number of possible combinations of XOR in encoding function. Excessive redundancy can be reduced using the setting of C ($\frac{m}{2}$, 3) in PBE-SW-PoR instead of C (*m*, 3) in the construction of XORs during encoding. This is because the higher the value of *m*, the higher the value of C (*m*, 3) as shown in Table 2.5. Besides, by limiting the number of copies of replicates for *B* in PBE-SW-PoR can avoid the problem of

excessive redundancy. As corruption can happen any time, even at times of repairing a corrupted stored data, hence long repair time might cause the repair to fail. In term of resiliency of PBE-SW-PoR, corrupted *B* can be repaired by simply replacing the corrupted one with a healthy copy. Hence, PBE-SW-PoR is resilient against corruption.

3.3 Conclusion

In summary, the proposed PBE-SW-PoR has been described in detail that includes its conceptual model, computational overhead, storage cost, and resiliency. As the repair process in SW-PoR takes only about 0.03 seconds (Thao *et al.*, 2014) for each pair of corrupted metadata and coded block, hence *A* in PBE-SW-PoR should has similar resiliency that is comparable with the original SW-PoR. On the other hand, in PBE-SW-PoR, the corrupted *B* can be replaced with a healthy copy stored in other servers. Lastly, excessive redundancy can be reduced by both C (*q*, 3), where $q = \frac{m}{2}$, of *A* which is much lesser than *m*, and limited or fixed number of copies of *B*. The simulation of the proposed PBE-SW-PoR, to measure its performance, is presented in Chapter 5.

CHAPTER 4

Optimized SW-PoR (Opti-SW-PoR)

4.1 Chapter Organization

This chapter presents the second approach proposed to reduce the computational time of SW-PoR, named the Optimized SW-PoR (Opti-SW-PoR). Opti-SW-PoR addresses the limitation of SW-PoR encoding through reducing the total number of XORs to be constructed with the concept of partitioning. Section 4.2 describes the details of the Opti-SW-PoR while Section 4.3 concludes this chapter.

4.2 Optimized SW-PoR (Opti-SW-PoR)

The computation time of SW-PoR encoding depends on the number of XORs. Hence, the approach proposed in this chapter was found based on the assumption that by reducing the number of XORs, the computation time of SW-PoR encoding can also be reduced. To avoid the problem of imbalanced redundancy as described in Section 3.2, the setting of C (m, 3) as number of XORs in encoding is necessary. However, this setting incurred exponential increment in computation time as data size increases linearly, as explained in Chapter 2 (Section 2.4.4). In this section, the proposed Opti-SW-PoR conceptual model, storage cost and resiliency are described in detail in Section 4.2.1, 4.2.2 and 4.2.3 respectively.

4.2.1 Conceptual Model

The main idea behind the proposed Opti-SW-PoR was found from the concept of partitioning (Chen, Zhou, and Cao, 2015) (Zheng, Chong, Myers, and Zdancewic, 2003). Partitioning a file means to divide a file into a number of counterparts. By adapting the concept of partitioning, the number of XORs for encoding can be reduced with the setting of $m > m'$, where m' is the number of file blocks per partition. Formula 4.1 define the number of partitions, p, of a file.

$$p = \frac{m}{m'} \qquad \text{(Formula 4.1)}$$

In the idea of partitioning, a file is viewed as a number of smaller files of equal size, without actual splitting of the file. The encoding is conducted on each partition separately instead on the whole file as how SW-PoR works. XOR operations are conducted among the file blocks within their respective partition only, without involving file blocks on other partitions. Figure 4.1 illustrates the difference between the original SW-PoR with the proposed Opti-SW-PoR with respect to the number of file partitions. In the original SW-PoR, the file encoding and retrieving for the whole file is only executed once. On the other hand, the Opti-SW-PoR requires the file encoding and retrieving to be conducted for every partition of the file (partition level).

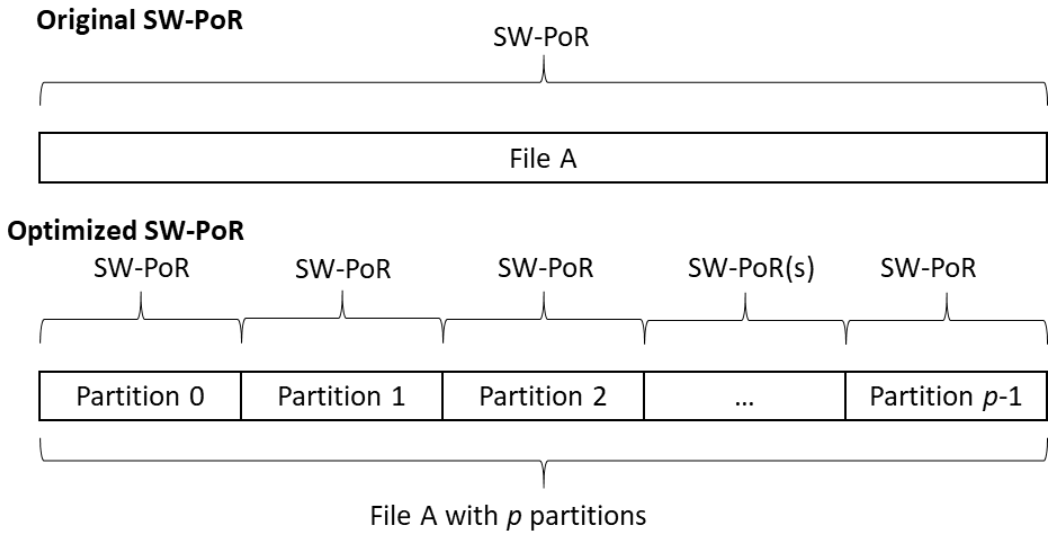

Figure 4.1: Overview of the difference between original SW-PoR and Opti-SW-PoR with respect to number of file partitions

In Opti-SW-PoR, each partition requires a number of XORs, $n' = C(m', 3)$, thus the whole file requires a number of $n' \times p$ XORs. Notice that $n' = C(m', 3)$ remains a constant regardless of change in file size as the number of file blocks per partition m' is set as a constant. The increase of file size leads to an increment of directly proportional in the number of partitions, p, hence the increase in number of XORs is linear. Figure 4.2 illustrates how Opti-SW-PoR performs encoding, retrieving, and repairing at partition level compared to the original SW-PoR.

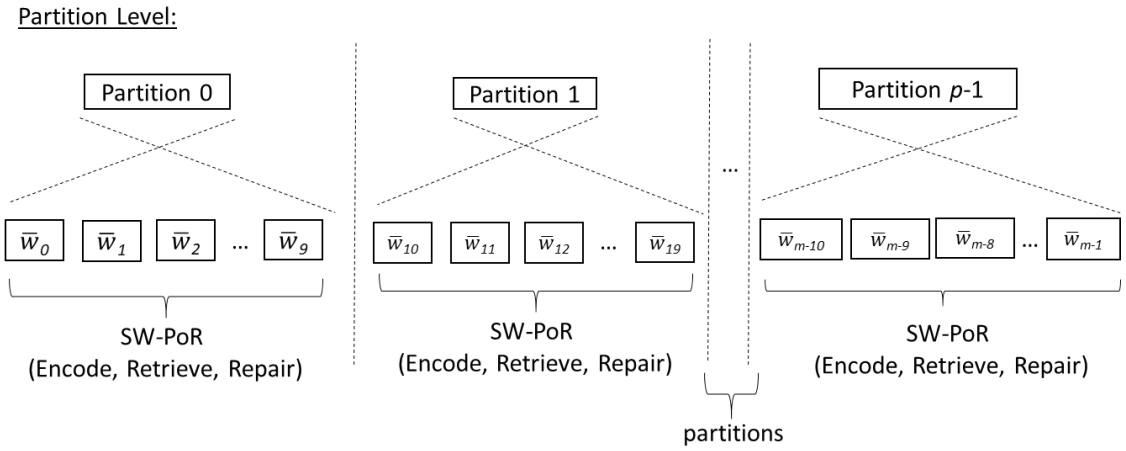

Figure 4.2: Conceptual model of Opti-SW-PoR at partition level

The following is an example to illustrates the different between the original SW-PoR and Opti-SW-PoR. Assume a client has a file, *F*, of size 1,024,000 bits (approximately one Megabits) to be stored in the cloud. The setting in original SW-PoR is as follows. Given that each file block constitutes of 2^{10} bits (1024 bits), the number of file blocks, *m* is calculated by dividing the file size by block size as shown in Formula 4.2. Therefore $m = \frac{1{,}024{,}000}{1024}$ = 1,000 and the number of XORs required in the original SW-PoR is n = C (m, 3) = C (1000,3) = 166,167,000.

$$No.of\ file\ blocks, m = \frac{File\ size}{Block\ size} \qquad \text{(Formula 4.2)}$$

Similarly, Opti-SW-PoR uses the same setting and formula as in the original SW-PoR for file size, block size, and number of file blocks. In addition, each partition constitutes of 10 file blocks, m' = 10, thus there are 100 partitions, p = 100. Meanwhile, each partition required only a constant number of XORs, n' = C (m', 3) = C (10, 3) = 120. With respect to this, the number of XORs required for the whole file is n'x p = 120 x 100 = 12,000.

Based on the number of XORs required by the original SW-PoR (166,167,000) and Opti-SW-PoR (12,000), the computation cost will reduce greatly in Opti-SW-PoR. For numerical estimation and comparison, the simulation result of (Thao *et al.*, 2014) using the setting of $n = m+1$ (this setting does not provide resiliency as only a few file blocks has redundancy) as the number of XORs involved as mentioned in Section 2.4.4 is referred. For a file consists of 50 file blocks requires 51 XORs, and the computation time taken is 0.25 seconds. By average, each XOR requires approximately 0.005 seconds computation time. For a file, *F*, of size 1,024,000 bits which requires 166,167,000 XORs in original SW-PoR with the setting of C (m, 3), it requires about 830,835 seconds (approximately 230.8 hours) to complete the computation. Contrary, as Opti-SW-PoR generates only 12,000 XORs, hence it needs only about 60 seconds to complete the computation. A huge different with respect to computation time between the original SW-PoR and Opti-SW-PoR with setting of C (m, 3) can be seen. Table 4.1 shows the comparison between the original SW-PoR and the proposed Opti-SW-PoR using analytical method.

Table 4.1: Analytic comparison of original SW-PoR vs Opti- SW-PoR

Schemes	**Original SW-PoR**	**Opti-SW-PoR**
File size	$\|F\|$	
Block size	$\|\overline{w}\|$	
No. of file blocks, *m*	$\frac{\|F\|}{\|\overline{w}\|}$	
No. of file blocks in each partition	NA	m'
No. of partitions, *p*	NA	$\frac{m}{m'}$
No. of XORs required per partition, *n'*	NA	C (m', 3)
No. of XORs required for whole file, *n*	C (m, 3)	p x n'

Note that as shown in Table 4.1, C (m, 3) is an exponential variable, whereas C (m', 3) is a constant. Meanwhile, p is a linear variable, whereas n' is a constant. The

number of file block per partition, m' is the same in every partition, thus C (m', 3) turns into a constant, n', for every partition throughout the whole encoding process in Opti-SW-PoR. As n' turns into a constant for every partition, the total number of XORs required is p x n'(linear).

By adapting the concept of partitioning, both balanced redundancy and reduction in number of XORs can be achieved. It also reduces the computation time from exponential to linear increment. In addition, there is no computation time overhead produced from Opti-SW-PoR as it involves only multiple iterations of SW-PoR. Although it involves multiple iteration of SW-PoR, Opti-SW-PoR considers only parts of XORs that original SW-PoR produced, (p x n) < C (m, 3) respectively. This is the reason why Opti-SW-PoR has no overhead even if it conducts multiple iterations of SW-PoR. For an example, says *Example A*, if a file consists of eight file blocks, original SW-PoR would construct 56 XORs, while Opti-SW-PoR construct only eight XORs if there are two partitions of four file blocks each. Table 4.2 shows the XOR computed by the original SW-PoR for *Example A*. The XORs presented in highlighted text are the XORs that will be computed when Opti-SW-PoR is employed. Table 4.3 shows these XORs according to partitions.

Table 4.2: XORs constructed by the original SW-PoR

$\overline{w}_0 \oplus \overline{w}_1 \oplus \overline{w}_2$	$\overline{w}_0 \oplus \overline{w}_1 \oplus \overline{w}_3$	$\overline{w}_0 \oplus \overline{w}_1 \oplus \overline{w}_4$	$\overline{w}_0 \oplus \overline{w}_1 \oplus \overline{w}_5$	$\overline{w}_0 \oplus \overline{w}_1 \oplus \overline{w}_6$	$\overline{w}_0 \oplus \overline{w}_1 \oplus \overline{w}_7$
$\overline{w}_0 \oplus \overline{w}_2 \oplus \overline{w}_3$	$\overline{w}_0 \oplus \overline{w}_2 \oplus \overline{w}_4$	$\overline{w}_0 \oplus \overline{w}_2 \oplus \overline{w}_5$	$\overline{w}_0 \oplus \overline{w}_2 \oplus \overline{w}_6$	$\overline{w}_0 \oplus \overline{w}_2 \oplus \overline{w}_7$	
$\overline{w}_0 \oplus \overline{w}_3 \oplus \overline{w}_4$	$\overline{w}_0 \oplus \overline{w}_3 \oplus \overline{w}_5$	$\overline{w}_0 \oplus \overline{w}_3 \oplus \overline{w}_6$	$\overline{w}_0 \oplus \overline{w}_3 \oplus \overline{w}_7$		
$\overline{w}_0 \oplus \overline{w}_4 \oplus \overline{w}_5$	$\overline{w}_0 \oplus \overline{w}_4 \oplus \overline{w}_6$	$\overline{w}_0 \oplus \overline{w}_4 \oplus \overline{w}_7$			
$\overline{w}_0 \oplus \overline{w}_5 \oplus \overline{w}_6$	$\overline{w}_0 \oplus \overline{w}_5 \oplus \overline{w}_7$				
$\overline{w}_0 \oplus \overline{w}_6 \oplus \overline{w}_7$					
$\overline{w}_1 \oplus \overline{w}_2 \oplus \overline{w}_3$	$\overline{w}_1 \oplus \overline{w}_2 \oplus \overline{w}_4$	$\overline{w}_1 \oplus \overline{w}_2 \oplus \overline{w}_5$	$\overline{w}_1 \oplus \overline{w}_2 \oplus \overline{w}_6$	$\overline{w}_1 \oplus \overline{w}_2 \oplus \overline{w}_7$	
$\overline{w}_1 \oplus \overline{w}_3 \oplus \overline{w}_4$	$\overline{w}_1 \oplus \overline{w}_3 \oplus \overline{w}_5$	$\overline{w}_1 \oplus \overline{w}_3 \oplus \overline{w}_6$	$\overline{w}_1 \oplus \overline{w}_3 \oplus \overline{w}_7$		
$\overline{w}_1 \oplus \overline{w}_4 \oplus \overline{w}_5$	$\overline{w}_1 \oplus \overline{w}_4 \oplus \overline{w}_6$	$\overline{w}_1 \oplus \overline{w}_4 \oplus \overline{w}_7$			
$\overline{w}_1 \oplus \overline{w}_5 \oplus \overline{w}_6$	$\overline{w}_1 \oplus \overline{w}_5 \oplus \overline{w}_7$				
$\overline{w}_1 \oplus \overline{w}_6 \oplus \overline{w}_7$					
$\overline{w}_2 \oplus \overline{w}_3 \oplus \overline{w}_4$	$\overline{w}_2 \oplus \overline{w}_3 \oplus \overline{w}_5$	$\overline{w}_2 \oplus \overline{w}_3 \oplus \overline{w}_6$	$\overline{w}_2 \oplus \overline{w}_3 \oplus \overline{w}_7$		
$\overline{w}_2 \oplus \overline{w}_4 \oplus \overline{w}_5$	$\overline{w}_2 \oplus \overline{w}_4 \oplus \overline{w}_6$	$\overline{w}_2 \oplus \overline{w}_4 \oplus \overline{w}_7$			
$\overline{w}_2 \oplus \overline{w}_5 \oplus \overline{w}_6$	$\overline{w}_2 \oplus \overline{w}_5 \oplus \overline{w}_7$				
$\overline{w}_2 \oplus \overline{w}_6 \oplus \overline{w}_7$					

$\overline{w}_3 \oplus \overline{w}_4 \oplus \overline{w}_5$	$\overline{w}_3 \oplus \overline{w}_4 \oplus \overline{w}_6$	$\overline{w}_3 \oplus \overline{w}_4 \oplus \overline{w}_7$			
$\overline{w}_3 \oplus \overline{w}_5 \oplus \overline{w}_6$	$\overline{w}_3 \oplus \overline{w}_5 \oplus \overline{w}_7$				
$\overline{w}_3 \oplus \overline{w}_6 \oplus \overline{w}_7$					
$\overline{w}_4 \oplus \overline{w}_5 \oplus \overline{w}_6$	$\overline{w}_4 \oplus \overline{w}_5 \oplus \overline{w}_7$				
$\overline{w}_4 \oplus \overline{w}_6 \oplus \overline{w}_7$					
$\overline{w}_5 \oplus \overline{w}_6 \oplus \overline{w}_7$					

Table 4.3: XORs constructed by Opti-SW-PoR

Partition 0	Partition 1
$\overline{w}_0 \oplus \overline{w}_1 \oplus \overline{w}_2$	$\overline{w}_4 \oplus \overline{w}_5 \oplus \overline{w}_6$
$\overline{w}_0 \oplus \overline{w}_1 \oplus \overline{w}_3$	$\overline{w}_4 \oplus \overline{w}_5 \oplus \overline{w}_7$
$\overline{w}_0 \oplus \overline{w}_2 \oplus \overline{w}_3$	$\overline{w}_4 \oplus \overline{w}_6 \oplus \overline{w}_7$
$\overline{w}_1 \oplus \overline{w}_2 \oplus \overline{w}_3$	$\overline{w}_5 \oplus \overline{w}_6 \oplus \overline{w}_7$

4.2.2 Storage Cost

In SW-PoR, data is stored as coded block and metadata pairs, $\{c_i, \hat{c}_i\}$ across distributed cloud servers. Hence, the storage cost of data in SW-PoR can be estimated by multiplying the storage cost per pair of coded block and metadata by the number of pairs of coded block and metadata constructed. Though, exact storage cost is very difficult to be identified as the size of each pair of coded block and metadata is vary.

As stated in (Thao *et al.*, 2014) the coded block has a size of $\log_2 \binom{|\overline{w}|}{\hat{c}}$, whereas metadata has a size of $\log_2 |\overline{w}|$, given the block length, $|\overline{w}|$. Hence, a pair of coded block and metadata has a storage cost, $s = \log_2 \binom{|\overline{w}|}{\hat{c}} + \log_2 |\overline{w}|$, and thus the overall storage cost of a data stored using SW-PoR is $S = n \times s$, given n is the number of coded block and metadata pairs converted from XOR constructed blocks.

By comparison, Opti-SW-PoR requires lower storage cost than the original SW-PoR due to the lower number of coded block and metadata pairs produced in encoding.

The storage cost of Opti-SW-PoR is $S = p \times n' \times s$. which is far lower than that of original SW-PoR, $S = n \times s = C(m, 3) \times s$, as $p \times n'$ is a linear variable while $C(m, 3)$ is an exponential variable. Using the previous example, *Example A* as described in Section 4.2.1, where a file consists of eight file blocks is encoded using the original SW-PoR and Opti-SW-PoR. Using the original SW-PoR, the number of combinations of XOR is C (8, 3) = 56. In Opti-SW-PoR, there are two partitions with four file blocks each, thus the number of combinations of XOR per partition is C (4, 3) = 4. In total, Opti-SW-PoR will construct eight XORs (4 x 2 = 8). The original SW-PoR has to construct 56 pairs of coded block and metadata, while Opti-SW-PoR has to construct eight pairs only. If the size of each pair of coded block and metadata cost is 100 bits, then the original SW-PoR needs 56 x 100 bits = 5,600 bits of storage size, while Opti-SW-PoR needs only 8 x 100 bits = 800 bits of storage size which is about half of a magnitude less than the original SW-PoR. Table 4.4 summarizes the comparison of storage cost for both the original SW-PoR and the Opti-SW-PoR.

Table 4.4: Storage cost of the original SW-PoR vs Opti- SW-PoR

Schemes	**Original SW-PoR**	**Opti-SW-PoR**
Size of a pair of coded block and metadata, *s*	$\log_2 \binom{\lvert\overline{w}\rvert}{\hat{c}} + \log_2 \lvert\overline{w}\rvert$	
No. of pairs of coded block and metadata, *n*	$C(m, 3)$	$p \times n'$
Storage cost, *S*	$C(m, 3) \times s$	$p \times n' \times s$

4.2.3 Resiliency

Resiliency of a PoR scheme defined as the ability of the scheme to recover corrupted data. With respect to SW-PoR, data repair can be performed by using redundant data (redundant file block). Hence, the resiliency of SW-PoR is measured in which whether the number of redundancy per file block is enough for corruption repair or otherwise.

Using the setting of C (*m*, 3) in SW-PoR encoding provides balanced redundancy at the file block level, which means that each file block has the same number of redundancy. The number of redundancy for each file block is calculated by multiplying the number of XORs produced, C (*m*, 3), by three (since each of the XOR constructed block stores the information of three file blocks), then divided by the number of file blocks, *m*, as shown in Formula 3.1.

Redundancy per file block in the original SW-PoR can be calculated using the formula as shown in Formula 3.1. Similarly, redundancy per file block in Opti-SW-PoR can also be calculated using the Formula 3.1 as well, with minor distinction where the variable *m* has to be replaced with number of file block per partition, *m'.* By comparison, Opti-SW-PoR has a lesser number of redundancy per file block than that of the original SW-PoR as the number of XORs produced is limited in Opti-SW-PoR due to partitioning. However, Opti-SW-PoR share similar resiliency with original SW-PoR with C (*m*, 3) setting in encoding. This is because the setting of C (*m'*, 3) is used in Opti-SW-PoR which takes all possible combination of file blocks within partition for XORs formation. Thus, within partition, redundancy of file blocks is maximum. Hence, the redundancy per file block in Opti-SW-PoR will not be affected by the file size or the total number of file blocks. Contrary, redundancy per file block in the original SW-PoR with the setting of C (*m*, 3) will grows exponentially in number when the file size increases.

Both original SW-PoR with C (*m*, 3) setting at file level and Opti-SW-PoR with C (*m'*, 3) at partition level have sufficient redundancy to repair a corrupted pair of coded block and metadata. This is because both C (*m*, 3) and C (*m'*, 3) are maximum possible number of combinations of file blocks for XORs formation in original SW-PoR and Opti-SW-PoR respectively, as stated above. The repair time consumed a very short time, approximately 0.03 seconds as shown in the simulation of (Thao *et al.*, 2014). Hence, both original SW-PoR and Opti-SW-PoR are similar in term of resiliency. Nonetheless, original SW-PoR will always has excessive redundancy as the number of

XOR constructed blocks produced is always exponential in number. Even though excessive redundancy also guarantees resiliency, but it causes the storage size required to increase exponentially. The adverse effect of excessive redundancy can be seen in Table 4.2. The XOR blocks produced using Opti-SW-PoR shown in Table 4.3 provide similar resiliency to those in Table 4.2.

4.3 Conclusion

In summary, the proposed Opti-SW-PoR has been described in detail with conceptual model, storage cost, and resiliency. Repair of error data in SW-PoR takes a very short time to compute, approximately 0.03 seconds as shown in simulation of (Thao *et al.*, 2014) for each pair of corrupted metadata and coded block. Although Opti-SW-PoR has a lower number of redundancy than the original SW-PoR, the redundancy per file block at partition level of Opti-SW-PoR is maximized with C (*m'*, 3) setting. Hence, it is sufficient to repair any data corruption. Lastly, excessive redundancy will only increase the storage cost as the number of redundancy per file block increases exponentially as the file size increase.

CHAPTER 5

Simulation and Performance Analysis

5.1 Chapter Organization

This chapter presents the simulations of the proposed PBE-SW-PoR and Opti-SW-PoR for performance evaluation. Section 5.2 describes the setup of the simulation work. Section 5.3 evaluates and analyses the performances of Opti-SW-PoR and PBE-SW-PoR based on the results of the simulations. The conclusion is presented in Section 5.4.

5.2 Experimental Setup

In this thesis, simulation experiments are performed to test and evaluate the performance of Opti-SW-PoR and PBE-SW-PoR. Simulation is conducted using file block size of 1,024 bits, simulation data sizes ranged from 200 to 1,000 file blocks, and a computer with specification shown in Table 5.1. This setting has been selected as described in Chapter 1 (Section 1.6) of this thesis.

Table 5.1: Machine specification for simulation

Processor	Intel i5-3210M
Clock Speed	2.50 GHz
RAM	8 GB
Operating System	Windows 10 (64-bit)

Random binary data is used in the simulation to imitate the real-world cases in which data might have no specific trend or sequence (redundant bits) due to encryption and compression. Random binary data are generated using the class Random of Java language. Generated binary data are kept in text files to be used in the simulation. This setting is similar to the work presented in (Thao *et al.*, 2014). The Encode, Retrieve, and Repair functions of the original SW-PoR, Opti-SW-PoR, and PBE-SW-PoR are simulated using the same binary data generated. For Opti-SW-PoR, file block per partition, m' can be any values, where $m' < m$ as described in Section 4.2.1. In the simulation, m' is set to 10 for values of m ranged from 200 to 1000. The search for optimum value of m' is left as future work as it is out of the scope of this research. Thus number of C (m', 3) = C (10, 3) = 120 combinations of XORs per partition which is a constant throughout every partition. In original SW-PoR, number of XORs is set to C (m, 3), whereas in PBE-SW-PoR, number of XORs is set to C (q, 3) where q = $\frac{m}{2}$ as A composed of half of the file. For PBE-SW-PoR, file split ratio is set to 1:1 in the simulation as the file is split into two equal halves for fair comparison between time used to encode A and B as explained in Section 3.2.1. Computation time of each simulation runs are recorded for performance evaluation as presented in Section 5.3.

5.3 Performance Evaluation

This section presents the results of the simulation in the form of overall computation time including overhead recorded by each approach to complete the Encode, Retrieve and Repair functions. In the simulation conducted, the amount of computation time used by an approach indicates its performance. The lower the computation time, the faster the approach completed the task and thus the better the performance of the approach.

Table 5.2 and Figure 5.1 show the computation time of each approach used to complete the encode function. The computation time of encoding has a direct correlation with the data size; more time is needed to complete the encode function for the original

SW-PoR, Opti-SW-PoR, and PBE-SW-PoR. This can be explained in which the increase of data size has contributed to the increase of the number of file blocks, thus increasing the number of combinations for XORs in Encode function. Hence, more time is needed to complete the encoding when data size increases.

Table 5.2: Performance of Encode function in original SW-PoR, Opti-SW-PoR, and PBE-SW-PoR

No. of file blocks	Computation time (seconds)		
	Original SW-PoR	Opti-SW-PoR	PBE-SW-PoR
200	8,314.8	**9.8**	708.1
400	67,700.8	**19.7**	5,660.7
600	230,355.7	**29.6**	19,383.4
800	419,598.9	**39.1**	44,784.3
1,000	835,179.7	**49.3**	85,863.0

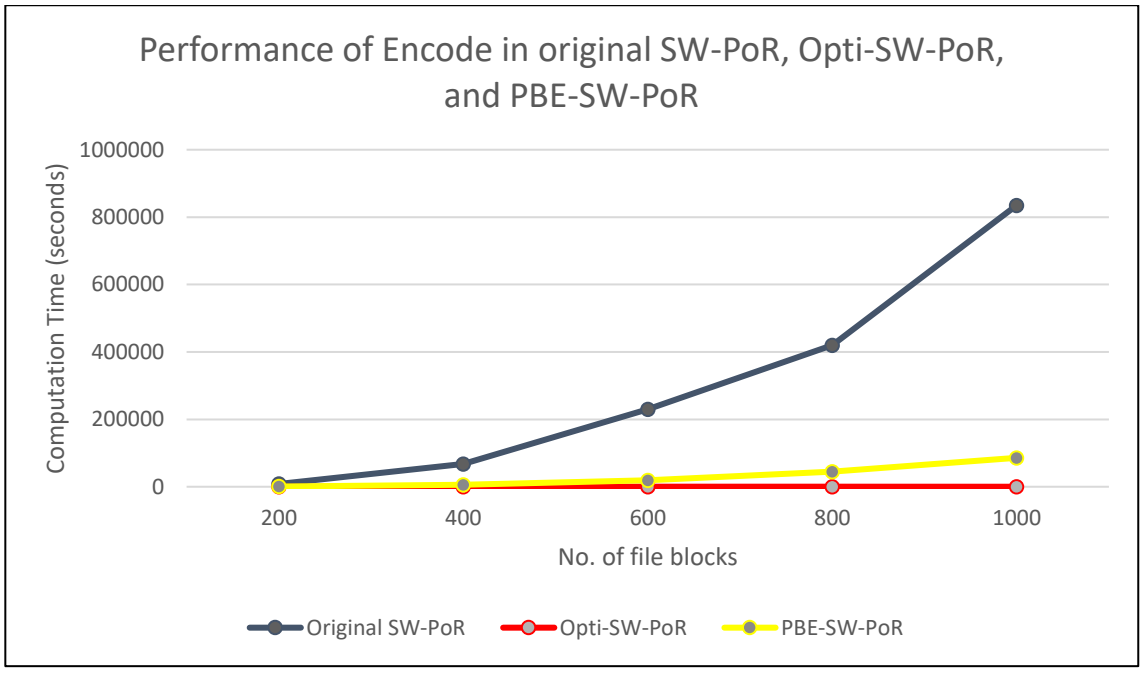

Figure 5.1: Performance of Encode function in original SW-PoR, Opti-SW-PoR, and PBE-SW-PoR

Both the original SW-PoR and PBE-SW-PoR have shown the trend of exponential increment of computation time to complete the encoding. However, PBE-SW-PoR encode faster than the original SW-PoR. As the data size increases from 200 file blocks to 1,000 file blocks, original SW-PoR took 8,314.8 to 835,179.7 seconds respectively. Meanwhile, PBE-SW-PoR used much lesser time, from 708.1 to 85,863.0 seconds respectively. From the simulation results, PBE-SW-PoR only required approximately one-tenth of computation time needed by original SW-PoR to complete the encoding on the specified data sizes. On the other hand, Opti-SW-PoR has shown the trend of linear increment of computation time to complete the encoding and consequently outperformed both the original SW-PoR and PBE-SW-PoR in term of computation time used to complete the encoding. Using a data size of 200 file blocks, Opti-SW-PoR took only 9.8 seconds to complete the encoding. The encoding computation time of Opti-SW-PoR increases gradually about 9 to 10 seconds for each increment of 200 file blocks in data size. The encoding computation time of Opti-SW-PoR increased to 49.3 when data size reached 1,000 file blocks.

The obtained results are as expected and can be justified. Both the original SW-PoR and PBE-SW-PoR have exponential computation time in encoding because the amount of XORs to be conducted is exponential, C (m, 3) and C (q, 3) respectively, where q = $\frac{m}{2}$. On the other hand, Opti-SW-PoR has linear encoding time due to the amount of XORs is linear, p x n'. As p x n' < C (q, 3) < C (m, 3), hence the amount of computation time required to complete the encoding is Opti-SW-PoR is less than PBE-SW-PoR is less than the original SW-PoR.

Meanwhile, the encoding computation time used to process *B* is less than *A* in PBE-SW-PoR. When the number of file blocks, m = 200, the process of CRC plus replication for *B* took 0.01 seconds, whereas process of SW-PoR for *A* took about 708.0 seconds. The computation time used for *B* increases slightly from 0.01 seconds to 0.06 seconds when data size increases from 200 to 1,000 file blocks. This shows that the time

used to process *B* is lesser than *A*, hence higher portion of *B* will further reduce the total computation time as described in Section 3.2.1 of this thesis. Table 5.3 shows about the comparison between time used to compute *A* and *B* across different data sizes in PBE-SW-PoR.

Table 5.3: Computation time used to compute *A* and *B* in PBE-SW-PoR

No. of file blocks	**Computation time (seconds)**	
	A	***B***
200	708.0	**0.01**
400	5,660.5	**0.02**
600	19,383.3	**0.04**
800	44,784.1	**0.05**
1,000	85,862.8	**0.06**

From Table 5.4 and Figure 5.2, a clear trend can be seen for data retrieval time of the original SW-PoR, Opti-SW-PoR, and PBE-SW-PoR recorded in the simulations. All three approaches experienced linear increment of computation time in data retrieval as data size grows. When data size grows from 200 to 1,000 file blocks, the retrieval time of original SW-PoR increases from 3.4 to 25.7 seconds respectively. Opti-SW-PoR took only 2.1 to 10.5 seconds using the same data size. Generally, Opti-SW-PoR only used about half of retrieval time used by the original SW-PoR. Meanwhile, PBE-SW-PoR used only 1.2 to 8.4 seconds to retrieve data of size 200 to 1,000 file blocks respectively.

Table 5.4: Performance of Retrieve function in original SW-PoR, Opti-SW-PoR, and PBE-SW-PoR

No. of file blocks	**Computation time (seconds)**		
	Original SW-PoR	**Opti-SW-PoR**	**PBE-SW-PoR**
200	3.4	2.1	**1.2**
400	8.7	4.1	**2.8**
600	12.5	6.2	**3.9**
800	16.8	8.2	**5.6**
1,000	25.7	10.5	**8.4**

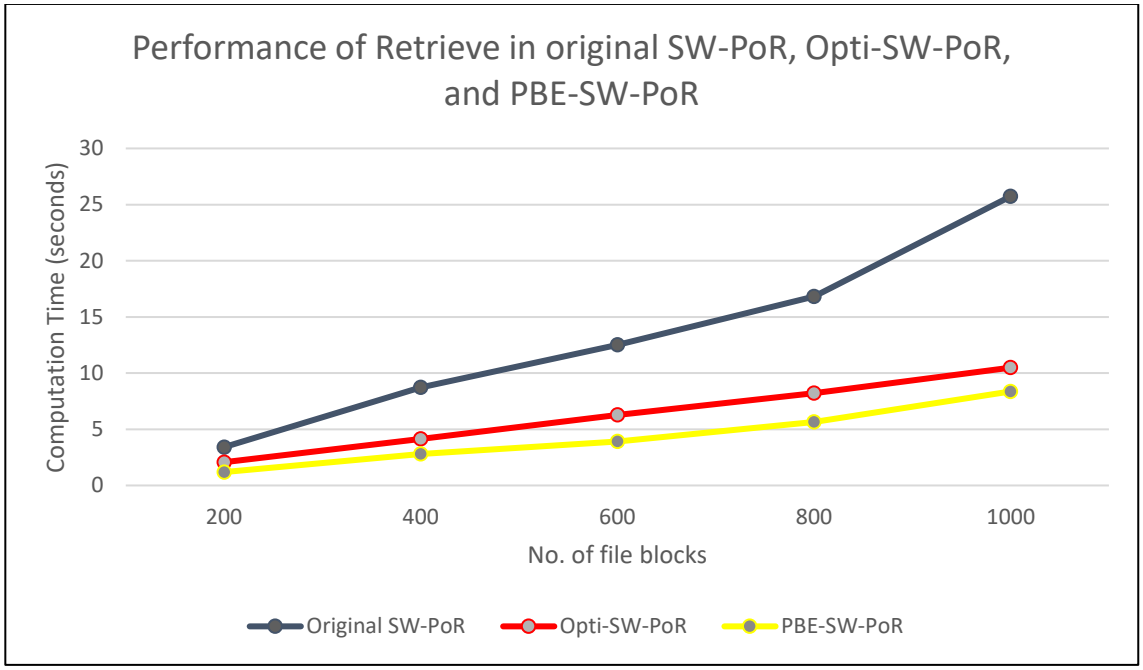

Figure 5.2: Performance of Retrieve function in original SW-PoR, Opti-SW-PoR, and PBE-SW-PoR

The obtained linear retrieval time for all three approaches is due to the number of file blocks to retrieve increases linearly as data size increases linearly. Retrieval of stored data using SW-PoR requires the solving of a square matrix (m x m) via Gaussian Elimination, as described in detailed in Chapter 2 (Section 2.4.2) of this thesis. When data size increases, the number of file blocks (m) increases, and the retrieval time increases. This also means that, when the data size and the number of file blocks (m) increases, the square matrix (m x m) required to solve becomes larger, causing the retrieval time to increase. PBE-SW-PoR only required to solve a square matrix of size (q x q) = ($\frac{m}{2}$ x $\frac{m}{2}$) where q = $\frac{m}{2}$, since the data file is divided into two parts equally. Hence, the retrieval time of the whole data in PBE-SW-PoR (including data counterpart stored with CRC plus replication) is much smaller. Theoretically, when the matrix to be solved is reduced to one-quarter from (m x m) to ($\frac{m}{2}$ x $\frac{m}{2}$), the retrieval time should be reduced

to one-quarter as well. However, this is not the case for PBE-SW-PoR where the retrieval time recorded is about one-third of the original SW-PoR. This is because overhead occurs in: (i) the removal of CRC bits in B and (ii) joining of the two separated data parts. For Opti-SW-PoR, the retrieval time needed recorded is less than the original SW-PoR or about half of the original SW-PoR. This is because solving smaller matrix or matrices of constant size (m' x m') = (10 x 10) is faster than solving a larger matrix (Pan, 1992), which is also indicated in the simulation results of PBE-SW-PoR. On the other hand, Table 5.5 and Figure 5.3 show about the computation time of Repair function in original SW-PoR, Opti-SW-PoR, and PBE-SW-PoR.

Table 5.5: Performance of Repair function in original SW-PoR, Opti-SW-PoR, and PBE-SW-PoR

No. of file blocks	**Computation time (seconds)**		
	Original SW-PoR	**Opti-SW-PoR**	**PBE-SW-PoR**
200	0.034	0.031	0.032
400	0.032	0.032	0.031
600	0.031	0.031	0.031
800	0.033	0.032	0.031
1,000	0.031	0.032	0.031

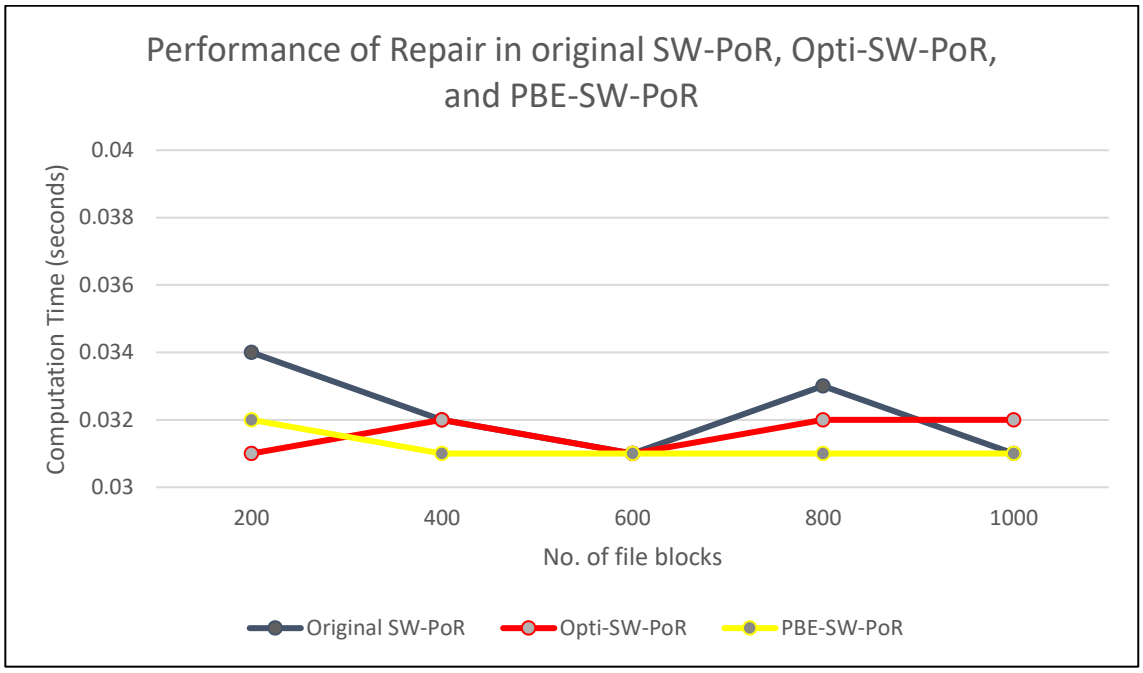

Figure 5.3: Performance of Repair function in original SW-PoR, Opti-SW-PoR, and PBE-SW-PoR

As the process to repair a corrupted pair of coded block and metadata in all the three approaches require three pairs of healthy coded block and metadata, the repair time is expected to be similar and remains constant across different data sizes. In the simulation, the repair time across different data sizes for the original SW-PoR, PBE-SW-PoR, and Opti-SW-PoR is between 0.03 to 0.04 seconds. Repair of the corrupted B for PBE-SW-PoR does not need any computation, except the transmission of a healthy replicate which is error-free to replace the corrupted ones. Thus, the repair of the corrupted B using PBE-SW-PoR is excluded from the repair time calculation as time used for transmission of data is not included in computation time calculation.

To assess the performance of the work presented in this thesis, other criteria are also evaluated that include actual storage cost, redundancy per file block, and overhead (additional computation time). In the simulation, block size ($|\overline{w}|$) is set to a constant, 1,024 bits as mentioned in Section 5.2. Hence, a metadata, $\hat{c}$, has a constant size of $\log_2$

$|\bar{w}|$ = $\log_2$ 1,024 = 10. However, coded blocks, c, are variable, which has a range depends on the value of the corresponding metadata. Size of a coded block can be calculated given the value of the metadata and the block size using formula $\log_2 \binom{|\bar{w}|}{\hat{c}}$ as stated in Section 4.2.2. Table 5.6 shows the relation of $|\bar{w}|$, $\hat{c}$, and C ($|\bar{w}|$, $\hat{c}$) in order to identify the possible minimum and maximum number of bits of a coded block. With a constant value of $|\bar{w}|$ and variable values of metadata, $\hat{c}$, the values of C ($|\bar{w}|$, $\hat{c}$) are ranged from a minimum value of 1 to a maximum value of C ($|\bar{w}|$, $\frac{|\bar{w}|}{2}$). Hence, the minimum and maximum sizes of a coded block, $\log_2$ C ($|\bar{w}|$, $\hat{c}$) are $\log_2$ 1 = 0 bits and $\log_2 \binom{1,024}{512}$ = 1018.674 ≈ 1019 bits respectively. However, the size of coded block cannot be 0 bits as the lowest value of coded block which is zero also needed to be represented in a minimum of one bit that is '0'. Hence, the size of a coded block is ranged from a minimum of one bit to maximum of 1019 bits. Figure 5.4 shows the range of size of a coded block used in the simulation.

Table 5.6: Relation of $|\bar{w}|$, $\hat{c}$, and C ($|\bar{w}|$, $\hat{c}$)

$|\bar{w}|$ (bits)	**Value of $\hat{c}$**	**C ($|\bar{w}|$, $\hat{c}$)**
10	0	1
	1	10
	2	45
	3	120
	4	210
	5	252
	6	210
	7	120
	8	45
	9	10
	10	1

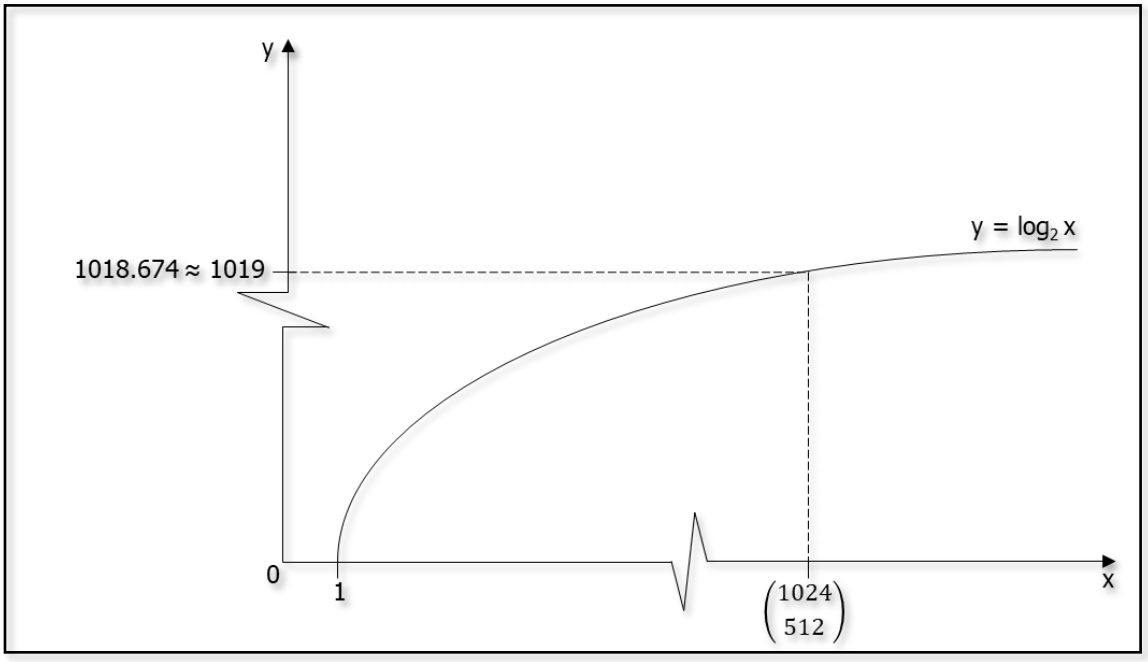

Figure 5.4: Graph of $y = \log_2 x$ to show the range of size of a coded block

The size of a pair of coded block and metadata can be calculated by adding the size of the metadata (10 bits) and the size of the coded block (1 to 1,019 bits). Hence, the size of a pair of coded block and metadata is ranged from a minimum of 11 bits to a maximum of 1,029 bits. Actual storage cost (minimum and maximum) of data encoded by SW-PoR can be calculated by multiplying the size of a pair of coded block and metadata with the number of file blocks. Meanwhile, the replicated CRC-added *B* stored through PBE-SW-PoR can be calculated by using the formula in Table 3.2. In the simulation of PBE-SW-PoR, three copies of CRC-added data counterpart are stored, *serv* = 3 to allow at least one out of two copies of healthy (error-free) replicates *B* when a corruption occur at *B*, whereas number of CRC bits is two, *crc* = 2 as the XOR divisor used in CRC operation is three bits (degree of divisor of three bits is two) as described in Section 3.2.1. The actual storage cost of each simulated number of file blocks are listed in Table 5.7 and 5.8 for the original SW-PoR, PBE-SW-PoR, and Opti-SW-PoR based on the formula described in Table 3.2 (in Chapter 3) and 4.4 (in Chapter 4).

Table 5.7: Minimum actual storage cost of original SW-PoR, Opti-SW-PoR, and PBE-SW-PoR

Min actual storage cost (bits)				
Scheme		**Original SW-PoR**	**Opti-SW-PoR**	**PBE-SW-PoR**
No. of file blocks	**200**	14,447,400	**26,400**	2,085,906
	400	116,454,800	**52,800**	15,061,806
	600	394,022,200	**79,200**	49,927,706
	800	935,149,600	**105,600**	117,683,606
	1,000	1,827,837,000	**132,000**	229,329,506

Table 5.8: Maximum actual storage cost of original SW-PoR, Opti-SW-PoR, and PBE-SW-PoR

Max actual storage cost (bits)				
Scheme		**Original SW-PoR**	**Opti-SW-PoR**	**PBE-SW-PoR**
No. of file blocks	**200**	1,351,488,600	**2,469,600**	166,696,506
	400	10,893,817,200	**4,939,200**	1,352,103,006
	600	36,858,985,800	**7,408,800**	4,585,219,506
	800	87,478,994,400	**9,878,400**	10,895,046,006
	1,000	170,985,843,000	**12,348,000**	21,310,582,506

Both Table 5.7 and 5.8 clearly shows that Opti-SW-PoR consumed the least storage spaces. Generally, Opti-SW-PoR needs a linearly larger storage spaces for data size which increase linearly. On the other hand, storage spaces needed to store data increases exponentially for both original SW-PoR and PBE-SW-PoR when data size increases linearly. However, the storage spaces needed by PBE-SW-PoR is about one-seventh to one-eighth lesser than original SW-PoR. At data size of 1,000 file block, original SW-PoR needs a storage space of 1,827,837,000 to 170,985,843,000 bits, whereas PBE-SW-PoR and Opti-SW-PoR needs about 229,329,506 to 21,310,582,506 bits and 132,000 to 12,348,000 bits respectively only of the storage spaces to store the data.

The actual resiliency of each approach (original SW-PoR, Opti-SW-PoR, and PBE-SW-PoR) can be calculated based on the amount of redundancy produced by each approach as described in both Chapter 3 (Section 3.2.4) and Chapter 4 (Section 4.2.3).

Actual redundancy per file block for the original SW-PoR can be calculated using Formula 3.1. Actual redundancy per file block in Opti-SW-PoR is partition based, hence it can be calculated by taking the number of file blocks per partition instead of total number of file blocks using Formula 3.1. Similarly, number of redundancy per file block for PBE-SW-PoR can be calculated for the SW-PoR encoded *A* using Formula 3.1, whereas the number of redundancy of replicated CRC-added *B* is three copies. A summary of the actual data redundancy per file block for each scheme is shown in Table 5.9.

Table 5.9: Actual data redundancy per file block of each scheme

No. of file blocks	**Original SW-PoR**	**Opti-SW-PoR**	**PBE-SW-PoR**
200	19,701	**36**	4,851
400	79,401	**36**	19,701
600	179,101	**36**	44,551
800	318,801	**36**	79,401
1,000	498,501	**36**	124,251

As shown in Table 5.9, the actual redundancy per file block is the least and constant in Opti-SW-PoR, while the highest in original SW-PoR. When the number of file blocks increases, the number of redundancy per file block increases in both original SW-PoR and PBE-SW-PoR. This indicates that redundancy for every single file block grows unlimitedly, which is excessive when the number of file blocks increases. To curb this issue as described in Section 3.2.4, PBE-SW-PoR works in such a way that the number of file block to encode is limited., whereas Opti-SW-PoR solved the issue by making the number of file block per partition a constant which leads to a constant number of redundancy per file block, 36 as shown in Table 5.9.

With respect to the overhead incurred by the proposed approaches over the original SW-PoR, there are no additional processes need to be performed for Opti-SW-

PoR as described in Section 4.2.1. Hence, its overhead is zero. On the other hand, PBE-SW-PoR requires a number of processes on top of SW-PoR, including data storing (data splitting, CRC operation and replication) and data retrieval (CRC removal and data joining). The computation time for these processes are identified as overhead. Nonetheless, these overheads imposed only very little effect on the overall computation time as shown in Table 5.10. Repair of corrupted CRC-added B is done by sending the healthy replicate to replace the corrupted one without additional task, hence repair overhead for PBE-SW-PoR is zero.

Table 5.10: Overhead of PBE-SW-PoR

No. of file blocks	**Overhead (seconds)**	
	Data storing	**Data retrieval**
200	0.104	0.013
400	0.133	0.014
600	0.169	0.016
800	0.204	0.017
1,000	0.234	0.019

To evaluate and summarize the overall computation performance of original SW-PoR, PBE-SW-PoR and Opti-SW-PoR, the computation time of Encode, Retrieve, and Repair functions of each approach are summed. Table 5.11 shows about the total computation time of each approach.

Table 5.11: Total computation time of each approach

No. of file blocks	Computation time (seconds)		
	Original SW-PoR	**Opti-SW-PoR**	**PBE-SW-PoR**
200	8,318.2	**11.9**	709.3
400	67,709.5	**23.8**	5663.5
600	230,368.2	**35.8**	19387.3
800	419,615.7	**47.3**	44789.9
1,000	835,205.4	**59.8**	85871.4

As shown in Table 5.11, Opti-SW-PoR used the least total computation time, whereas original SW-PoR consumed the most total computation time. Both original SW-PoR and PBE-SW-PoR have shown the trend of exponential increment in total computation time which is affected by the exponential encoding time in both approaches. However, PBE-SW-PoR has shown lesser total computation time compared to original SW-PoR, close to one order of magnitude across data sizes, where the difference is about 10 times. On the other hand, Opti-SW-PoR has the trend of linear increment in total computation time, which is more than four orders of magnitude lesser than that of original SW-PoR at data size of 1,000 file blocks. All in all, Opti-SW-PoR is the best one among the compared approaches in term of total computation time.

5.4 Conclusion

In summary, both Opti-SW-PoR and PBE-SW-PoR have shown better performances than the original SW-PoR in term of data storing time and data retrieval time. Repair time of each scheme remains constant, between 0.03 to 0.04 seconds regardless of the data sizes. Meanwhile, it is shown in Table 5.7 and 5.8 that both Opti-SW-PoR and PBE-SW-PoR consumes less storage cost than the original SW-PoR. Lastly, Opti-SW-PoR does not have any overhead over the original SW-PoR, whereas PBE-SW-PoR has only a minimal overhead for data storing and retrieval, which is far less than half second across data sizes.

CHAPTER 6

Conclusion and Future Works

6.1 Chapter Organization

This chapter consists of four main sections. Section 6.1 presents the organization of this chapter; Section 6.2 summarizes this research; Section 6.3 summarizes research main findings and contributions; whereas Section 6.4 lists out some possible future works.

6.2 Research Summary

To ensure cloud data integrity and availability, Proof of Retrievability (PoR) was introduced to cloud storage. A recent PoR scheme named Slepian-Wolf Based Proof of Retrievability (SW-PoR) was introduced to provide cost-efficient exact repair mechanism for erroneous data. However, to achieve maximum resiliency for data correctness, encoding process of SW-PoR requires considerably long computational time.

The motivation of the research presented in this thesis is to design an approach that allow shorter encoding computation time in implementation of SW-POR scheme for cloud storage. To achieve this, two viable solutions were proposed, namely Opti-SW-PoR and PBE-SW-PoR.

As the computation time of encoding process in SW-PoR depends on the number of coded block and metadata pairs to be constructed, the decrement in the number of XORs to be constructed has a direct relationship with the encoding time of SW-PoR. Meanwhile, the exponential factor of C (m, 3) causes the exponential increment in the encoding computation time of SW-PoR as data size increases. Hence, by limiting or decreasing the number of file blocks (m) to be encoded can be an effective solution to the problem. Both Opti-SW-PoR and PBE-SW-PoR proposed and implemented to limit and decrease the encoding time controlling factor, C (m, 3), in XORs construction. Opti-SW-PoR limits the number of XORs via partitioning the file before encoding. In Opti-SW-PoR, XOR operations are conducted in such a way of mutually exclusive with other partitions, which means each partition is treated as an individual file. By doing this, the total number of XORs constructed is greatly reduced. On the other hand, PBE-SW-PoR reduced the effect of C (m, 3) in XORs construction by adapting file splitting, CRC, and replication before SW-PoR encoding is performed. By halving the size of the file, the number of XORs to be constructed in encoding is greatly reduced in PBE-SW-PoR thus reducing the overall computation time of encoding.

6.3 Main Findings and Contributions

The objectives of this research are listed as follows:

1. To reduce the computation time of SW-PoR encoding using two proposed schemes, namely Partial Binary Encoding for SW-PoR (PBE-SW-PoR) and Optimized SW-PoR (Opti-SW-PoR).
2. To apply the two proposed schemes in (1) on larger data in simulation similar to actual cloud environment.
3. To compare and analyse the performance of the proposed scheme(s) in term of computation time against the original SW-POR.

To achieve the objective (1) of the research which is to reduce the computation time of SW-PoR encoding, two approaches have been introduced, namely PBE-SW-PoR and Opti-SW-PoR were introduced in Chapter 3 and 4 respectively. PBE-SW-PoR comes from the idea of reduction in size of data to be encoded by SW-PoR. In PBE-SW-PoR, data is firstly split into two parts equally, namely *A* and *B*. *A* is encoded by SW-PoR, whereas *B* is secured using CRC and replication. On the other hand, Opti-SW-PoR comes from the idea of partitioning. In Opti-SW-PoR, data is viewed as a number of smaller files of equal size, without actual splitting of the data. In Opti-SW-PoR, encoding is conducted on each partition separately instead on the whole file as how SW-PoR works. XOR operations are conducted among the file blocks within their respective partition only, without involving file blocks on other partitions.

To achieve objective (2) of this research which is to apply the proposed schemes, PBE-SW-PoR and Opti-SW-PoR on larger data in simulation, simulation is conducted to compare original SW-POR, PBE-SW-PoR and Opti-SW-PoR for Encode, Retrieve, and Repair functions in term of computation time. The simulation is conducted using data sizes ranged from 200 to 1000 file blocks which is far greater than 150 file blocks simulated by (Thao et al., 2014). Details of the experimental setup of the simulation can be found in Chapter 5 of this thesis.

To achieve the objective (3) of this research which is to compare and analyse the performance of PBE-SW-PoR and Opti-SW-PoR, the result of simulation conducted is recorded and described in Chapter 5. Three schemes (original SW-PoR, Opti-SW-PoR, and PBE-SW-PoR) were implemented in simulation for performances evaluation. With data size of 1,000 file blocks, original SW-PoR used 835,179.698 seconds, whereas PBE-SW-PoR and Opti-SW-PoR used 85,863.045 seconds and 49.300 seconds respectively to complete the encoding. This shows that the encoding performances of both PBE-SW-PoR and Opti-SW-PoR are better than original SW-PoR in term of computation time, while Opti-SW-PoR performed the best. On the other hand, with data size of 1,000 file blocks, original SW-PoR took 25.732 seconds to retrieve the file,

whereas PBE-SW-PoR and Opti-SW-PoR took 8.367 seconds and 10.484 seconds to retrieve the data. This shows that both PBE-SW-PoR and Opti-SW-PoR performed better than original SW-PoR in retrieving the data, while PBE-SW-PoR performed the best. Meanwhile, PBE-SW-PoR and Opti-SW-PoR retain the similar repair time as the original SW-PoR which is constant about 0.03 to 0.04 seconds.

6.4 Future Works

Based on the output of this research, possible future works are listed as follows:

1. For Opti-SW-PoR, the optimum number of file blocks per partition has yet to be found to allow lowest possible computation time of encoding while allowing sufficient number of redundancy per file block.
2. For PBE-SW-PoR, the optimum ratio for splitting before encoding has yet to be investigated.

REFERENCES

Amazon. (2017). Amazon Simple Storage Service (S3) FAQs. Retrieved September 27, 2017, from https://aws.amazon.com/s3/faqs/

Armknecht, F., Bohli, J.-M., Karame, G. O., Liu, Z., & Reuter, C. A. (2014). Outsourced Proofs of Retrievability. *CCS '14 Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, 831–843. https://doi.org/10.1145/2660267.2660310

Ateniese, G., Burns, R., & Herring, J. (2007). Provable Data Possession at Untrusted Stores. *Proceedings of the 14th …*, (1), 598–610. https://doi.org/10.1145/1315245.1315318

Au, M. H., Mu, Y., & Cui, H. (2015). Proof of retrievability with public verifiability resilient against related-key attacks. *IET Information Security*, *9*(1), 43–49. https://doi.org/10.1049/iet-ifs.2013.0322

Baciu, I. (2015). Advantages and disadvantages of cloud computing services, from the employee's point of view, (13), 95–101. Retrieved from https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2787612

BBC News. (2014). Edward Snowden: Leaks that exposed US spy programme. Retrieved from http://www.bbc.com/news/world-us-canada-23123964

Bernie, D. (1999). *Simulation versus Analytic Modeling in Large Computing Environments*.

Bertino, E., & Lim, H. S. (2010). Assuring data trustworthiness - Concepts and research challenges. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *6358 LNCS*, 1–12. https://doi.org/10.1007/978-3-642-15546-8_1

Cash, D., Küpçü, A., & Wichs, D. (2015). *Dynamic Proofs of Retrievability via Oblivious RAM*. Journal of Cryptology. https://doi.org/10.1007/978-3-642-38348-9_17

Chauhan, N. S., & Saxena, A. (2014). A robust scheme on proof of data retrievability in cloud. *Proceedings of the 2014 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2014*, 665–671. https://doi.org/10.1109/ICACCI.2014.6968574

Chen, K., Zhou, Y., & Cao, Y. (2015). Online Data Partitioning in Distributed Database Systems. *18th International Conference on Extending Database Technology (EDBT)*, 1–12. https://doi.org/10.5441/002/edbt.2015.02

Christie, E. J., Jensen, D. D., Buckley, R. T., Menefee, D. A., Ziegler, K. K., Wood, K. L., & Crawford, R. H. (2012). Prototyping Strategies: Literature Review and Identification of Critical Variables. *American Society for Engineering Education. Pp. 01154-22. 2012.*, 1154–1122.

Davey, W. (2016). How secure are Dropbox, Microsoft OneDrive, Google Drive and Apple iCloud cloud storage services? Retrieved September 27, 2017, from http://www.alphr.com/apple/1000326/how-secure-are-dropbox-microsoft-onedrive-google-drive-and-apple-icloud-cloud-storage

Dell EMC. (2017). Data deduplication – Dell EMC Glossary. Retrieved September 27, 2017, from https://www.emc.com/corporate/glossary/data-deduplication.htm

Dropbox. (2017). Dropbox Business - Under the hood: Architecture overview. Retrieved September 27, 2017, from https://www.dropbox.com/business/trust/security/architecture

Du, R., Deng, L., Chen, J., He, K., & Zheng, M. (2015). Proofs of ownership and retrievability in cloud storage. *Proceedings - 2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2014*, 328–335. https://doi.org/10.1109/TrustCom.2014.44

Etemad, M., & Küpçü, A. (2015). Generic Efficient Dynamic Proofs of Retrievability. *Cryptology ePrint Archive*, 85–96. Retrieved from http://eprint.iacr.org/2015/880

Fergus, O. (2017). Top Ten Major Risks Associated With Cloud Storage. Retrieved September 27, 2017, from https://www.cloudwards.net/top-ten-major-risks-associated-with-cloud-storage/

Google. (2017). Data Lifecycle on Google Cloud Platform. Retrieved October 19, 2017, from https://cloud.google.com/solutions/data-lifecycle-cloud-platform

Google. (2017). Security - Google Cloud Help: Privacy and Security. Retrieved September 27, 2017, from https://support.google.com/googlecloud/answer/6056693?hl=en

Halevi, S., Harnik, D., Pinkas, B., & Shulman-Peleg, A. (2011). Proofs of Ownership in Remote Storage Systems. *Proceedings of the 18th ACM Conference on Computer and Communications Security*, 491–500. https://doi.org/10.1145/2046707.2046765

Huang, K., Liu, J., Xian, M., Wang, H., & Fu, S. (2014). Enabling dynamic proof of retrievability in regenerating-coding-based cloud storage. *2014 IEEE International Conference on Communications Workshops, ICC 2014*, 712–717. https://doi.org/10.1109/ICCW.2014.6881283

Hur, J., Koo, D., Shin, Y., & Kang, K. (2016). Secure Data Deduplication with Dynamic Ownership Management in Cloud Storage. *IEEE Transactions on Knowledge and Data Engineering*, *28*(11), 3113–3125. https://doi.org/10.1109/TKDE.2016.2580139

Husain, M. I., Ko, S. Y., Uurtamo, S., Rudra, A., & Sridhar, R. (2014). Bidirectional data verification for cloud storage. *Journal of Network and Computer Applications*, *45*, 96–107. https://doi.org/10.1016/j.jnca.2014.07.003

ISACA. (2015). Isaca. *Glossary*, 1–103.

Jeevitha, M., Chandrasekar, A., & Karthik, S. (2015). Survey On Verification Of Storage Correctness In Cloud Computing. *International Journal Of Engineering And Computer Science*, *4*(4). Retrieved from http://www.ijecs.in/issue/v4-i9/46 ijecs.pdf

Jianchao, B., Huixia, L., Shoushan, L., Yaxing, Z., & Wei, L. (2015). Proof of retrievability based on LDPC codes. *Journal of China Universities of Posts and Telecommunications*, *22*(4), 17–25. https://doi.org/10.1016/S1005-8885(15)60663-X

Juels, A. ., & Kaliski Jr., B. S. . (2007). Pors: Proofs of retrievability for large files. *Proceedings of the ACM Conference on Computer and Communications Security*, 584–597. https://doi.org/10.1145/1315245.1315317

Juels, A., Kelley, J., Tamassia, R., & Triandopoulos, N. (2015). Falcon Codes : Fast , Authenticated LT Codes (Or : Making Rapid Tornadoes Unstoppable). *Ccs '15*, 1032–1047. https://doi.org/10.1145/2810103.2813728

Kalkal, M., & Malhotra, S. (2015). Replication for Improving Availability & Balancing Load in Cloud Data Centres. *International Journal of Advanced Research in Computer Science and Software Engineering*, *5*(4), 108–110.

Kiraz, M. S., Sertkaya, I., & Uzunkol, O. (2015). An efficient ID-based message recoverable privacy-preserving auditing scheme. *2015 13th Annual Conference on Privacy, Security and Trust, PST 2015*, 117–124. https://doi.org/10.1109/PST.2015.7232962

Ko, R., Lee, S., & Rajan, V. (2013). Cloud Computing Vulnerability Incidents: A Statistical Overview. *Cloud Security Alliance*, 21.

Lavauzelle, J., & Levy-Dit-Vehel, F. (2016). New proofs of retrievability using locally decodable codes. *IEEE International Symposium on Information Theory - Proceedings*, *2016–Augus*, 1809–1813. https://doi.org/10.1109/ISIT.2016.7541611

Li, J., Tan, X., Chen, X., & Wong, D. S. (2013). An efficient proof of retrievability with public auditing in cloud computing. *Proceedings - 5th International Conference on Intelligent Networking and Collaborative Systems, INCoS 2013*, 93–98. https://doi.org/10.1109/INCoS.2013.185

Li, J., Tan, X., Chen, X., Wong, D. S., & Xhafa, F. (2015). OPoR: Enabling proof of retrievability in cloud computing with resource-constrained devices. *IEEE Transactions on Cloud Computing*, *3*(2), 195–205. https://doi.org/10.1109/TCC.2014.2366148

Li, J., Li, J., Xie, D., & Cai, Z. (2016). Secure Auditing and Deduplicating Data in Cloud. *IEEE Transactions on Computers*, *65*(8), 2386–2396. https://doi.org/10.1109/TC.2015.2389960

Lin, C., Shen, Z., Chen, Q., & Sheldon, F. T. (2017). A Data Integrity Verification Scheme in Mobile Cloud Computing. *Journal of Network and Computer Applications*, *77*, 146–151. https://doi.org/http://dx.doi.org/10.1016/j.jnca.2016.08.017

Liu, D., & Zic, J. (2015). Proofs of encrypted data retrievability with probabilistic and homomorphic message authenticators. *Proceedings - 14th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2015*, *1*, 897–904. https://doi.org/10.1109/Trustcom.2015.462

Lorena, G.-M., & Agustin, O. (2015). An efficient confidentiality-preserving Proof of Ownership for deduplication. *Journal of Network and Computer Applications*, *50*, 49–59. https://doi.org/10.1016/j.jnca.2014.12.004

Maria, A. (1997). Introduction to modelling and simulation. *Winter Simulation Conference*, 7–13. https://doi.org/10.1109/WSC.1997.640371

Mell, P., & Grance, T. (2011). The NIST definition of cloud computing. *NIST Special Publication*, *145*, 7. https://doi.org/10.1136/emj.2010.096966

Microsoft. (2017). Microsoft Trust Center | Encryption. Retrieved September 27, 2017, from https://www.microsoft.com/en-us/trustcenter/security/encryption

Miller, A., Juels, A., Shi, E., Parno, B., & Katz, J. (2014). Permacoin: Repurposing bitcoin work for data preservation. *Proceedings - IEEE Symposium on Security and Privacy*, 475–490. https://doi.org/10.1109/SP.2014.37

Mishra, N., Bhardwaj, R., & Kumar, R. (2015). Data traceability in cloud environment. *International Conference on Computing, Communication and Automation, ICCCA 2015*, 674–677. https://doi.org/10.1109/CCAA.2015.7148459

Mohseen, L. (2014). Dropbox Explains Reason Behind 2014 Outage. Retrieved September 27, 2017, from https://www.cloudwards.net/news/dropbox-explains-reason-behind-2014-outage-2534/

Mukundan, R., Madria, S., & Linderman, M. (2014). Efficient integrity verification of replicated data in cloud using homomorphic encryption. *Distributed and Parallel Databases*, *32*(4), 507–534. https://doi.org/10.1007/s10619-014-7151-0

Nedelcu, B., Stefanet, M.-E., Tamasescu, I.-F., Tintoiu, S.-E., & Vezeanu, A. (2015). Cloud Computing and its Challenges and Benefits in the Bank System. *Database Systems Journal*, *VI*(1), 44–58.

Omote, K., & Thao, T. P. (2015). MD-POR: Multisource and Direct Repair for Network Coding-Based Proof of Retrievability. *International Journal of Distributed Sensor Networks*, *2015*, 1–14. https://doi.org/10.1155/2015/586720

Omote, K., & Tran, P. T. (2014). A New Efficient and Secure POR Scheme Based on Network Coding. *2014 IEEE 28th International Conference on Advanced Information Networking and Applications*. https://doi.org/10.1109/AINA.2014.17

Omote, K., & Tran, P. T. (2015). ND-POR: A POR based on network coding and dispersal coding. *IEICE Transactions on Information and Systems*, *E98D*(8), 1465–1476. https://doi.org/10.1587/transinf.2015EDP7011

Omote, K., & Tran, P. (2016). D2-POR : Direct Repair and Dynamic Operations in Network Coding-Based Proof of Retrievability. *IEICE Transactions on Information and Systems*, (4), 816–829.

Pan, V. (1992). Complexity of Computations with Matrices and Polynomials. *SIAM Review*, *34*(2), 225–262. https://doi.org/10.1137/1034049

Quest. (2015). The Benefits and Challenges of Cloud Computing, *32*(7), 2015.

Rashid, F., Miri, A., & Woungang, I. (2015). Proof of Storage for Video Deduplication in the Cloud. *Proceedings - 2015 IEEE International Congress on Big Data, BigData Congress 2015*, 499–505. https://doi.org/10.1109/BigDataCongress.2015.79

Rass, S. (2013). Dynamic Proofs of Retrievability from Chameleon-Hashes. *Security and Cryptography (SECRYPT), 2013 International Conference*.

Ren, Z., Wang, L., Wang, Q., & Xu, M. (2015). Dynamic proofs of retrievability for coded cloud storage systems. *IEEE Transactions on Services Computing*, *PP*(99), 1–13. https://doi.org/10.1109/TSC.2015.2481880

Sahin, S. (2006). Computer simulations in science education: Implications for distance education. *Turkish Online Journal of Distance Education*, *7*(4), 132–146.

Sang, S. (2015). Implementation of Cyclic Redundancy Check in Data Communication. In *International Conference on Computational Intelligence and Communication Networks* (pp. 529–531). https://doi.org/10.1109/CICN.2015.108

Sarkar, S., & Safavi-Naini, R. (2013). Proofs of retrievability via fountain code. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *7743 LNCS*, 18–32. https://doi.org/10.1007/978-3-642-37119-6_2

Saxena, R., & Dey, S. (2016). Cloud Audit: A Data Integrity Verification Approach for Cloud Computing. *Procedia Computer Science*, *89*, 142–151. https://doi.org/10.1016/j.procs.2016.06.024

Sengupta, B., Bag, S., Ruj, S., & Sakurai, K. (2016). Retricoin: Bitcoin Based on Compact Proofs of Retrievability. *Proceedings of the 17th International Conference on Distributed Computing and Networking*, 14:1--14:10. https://doi.org/10.1145/2833312.2833317

Shacham, H., & Waters, B. (2008). Compact proofs of retrievability. *Journal of Cryptology*, *26*(3), 442–483. https://doi.org/10.1007/s00145-012-9129-2

Shi, E., Stefanov, E., & Papamanthou, C. (2013). Practical Dynamic Proofs of Retrievability. *CCS '13 Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, 325–336. https://doi.org/10.1145/2508859.2516669

Shin, Y., Koo, D., Hur, J., & Yun, J. (2015). Secure proof of storage with deduplication for cloud storage systems. *Multimedia Tools and Applications*. https://doi.org/10.1007/s11042-015-2956-z

Song, X., & Deng, H. (2013). Lightweight proofs of retrievability for electronic evidence in cloud. *Information (Switzerland)*, *4*(3), 262–282. https://doi.org/10.3390/info4030262

Sookasa. (2015). OneDrive Security: An Overview. Retrieved September 27, 2017, from https://www.sookasa.com/resources/onedrive-security/

Sookhak, M., Talebian, H., Ahmed, E., Gani, A., & Khan, M. K. (2014). A review on remote data auditing in single cloud server: Taxonomy and open issues. *Journal of Network and Computer Applications*, *43*, 121–141. https://doi.org/10.1016/j.jnca.2014.04.011

Srinivas, J., Reddy, K., & Qyser, A. (2014). Cloud Computing Basics. *Building the Infrastructure for Cloud Security*, *1*, 3–22. https://doi.org/10.1007/978-1-4614-7699-3

Statista. (2017). Public cloud infrastructure spending worldwide 2015-2026 | Statistic. Retrieved November 15, 2016, from https://www.statista.com/statistics/507952/worldwide-public-cloud-infrastructure-hardware-and-software-spending-by-segment/

TechTarget. (2015). What is USA Patriot Act? Retrieved September 27, 2017, from http://searchdatamanagement.techtarget.com/definition/Patriot-Act

Thao, T. P., Kho, L. C., & Lim, A. O. (2014). SW-POR: A Novel POR Scheme Using Slepian-Wolf Coding for Cloud Storage. *2014 IEEE 11th Intl Conf on Ubiquitous Intelligence and Computing and 2014 IEEE 11th Intl Conf on Autonomic and Trusted Computing and 2014 IEEE 14th Intl Conf on Scalable Computing and Communications and Its Associated Workshops*, 464–472. https://doi.org/10.1109/UIC-ATC-ScalCom.2014.11

Thao, T. P., & Omote, K. (2016). ELAR : Extremely Lightweight Auditing and Repairing for Cloud Security. *ACM International Conference Proceeding Series*, *5*, 40–51.

Tiwari, D., & Gangadharan, G. R. (2015). A novel secure cloud storage architecture combining proof of retrievability and revocation. *2015 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2015*, 438–445. https://doi.org/10.1109/ICACCI.2015.7275648

Vasilopoulos, D., Antipolis, S., Önen, M., Antipolis, S., Antipolis, S., & Antipolis, S. (2016). Message-Locked Proofs of Retrievability with Secure Deduplication. *CCSW 2016 - Proceedings of the 2016 ACM Cloud Computing Security Workshop*, 73–83.

Virtru. (2016). Dropbox Encryption vs. Google Drive Encryption: Which is More Secure? Retrieved September 27, 2017, from https://www.virtru.com/blog/dropbox-encryption/

Wang, Y., Wu, Q., Qin, B., Chen, X., Huang, X., & Zhou, Y. (2015). Group-oriented Proofs of Storage. *Asiaccs*, (1), 73–84. https://doi.org/10.1145/2714576.2714630

Wang, Y., Wu, Q., Qin, B., Tang, S., & Susilo, W. (2017). Online / Offline Provable Data Possession. *IEEE Transactions on Information Forensics and Security*, *12*(5), 1182–1194.

Xu, J., Zhou, F., Jiang, Z., & Xue, R. (2016). Dynamic proofs of retrievability with square-root oblivious RAM. *Journal of Ambient Intelligence and Humanized Computing*, *7*(5), 611–621. https://doi.org/10.1007/s12652-015-0258-y

Yan, G., Zhu, Y. F., Gu, C. X., Zheng, Y. H., & Fei, J. L. (2013). An efficient proof of retrievability scheme for fully homomorphic encrypted data. *Journal of Networks*, *8*(2), 339–344. https://doi.org/10.4304/jnw.8.2.339-344

Yu, C. M., Chen, C. Y., & Chao, H. C. (2015). Proof of ownership in deduplicated cloud storage with mobile device efficiency. *IEEE Network*, *29*(2), 51–55. https://doi.org/10.1109/MNET.2015.7064903

Yuan, J., & Yu, S. (2013). Secure and constant cost public cloud storage auditing with deduplication. *2013 IEEE Conference on Communications and Network Security, CNS 2013*, 145–153. https://doi.org/10.1109/CNS.2013.6682702

Yuan, J., & Yu, S. (2013). Proofs of retrievability with public verifiability and constant communication cost in cloud. *Cloud Computing '13 Proceedings of the 2013 International Workshop on Security in Cloud Computing*, 19–26. https://doi.org/10.1145/2484402.2484408

Zafar, F., Khan, A., Malik, S. U. R., Ahmed, M., Anjum, A., Khan, M. I., … Jamil, F. (2017). A survey of cloud computing data integrity schemes: Design challenges, taxonomy and future trends. *Computers and Security*, *65*. https://doi.org/10.1016/j.cose.2016.10.006

Zhang, J., Tang, W., & Mao, J. (2014). Efficient public verification proof of retrievability scheme in cloud. *Cluster Computing*, *17*(4), 1401–1411. https://doi.org/10.1007/s10586-014-0394-8

Zheng, L., Chong, S., Myers, A. C., & Zdancewic, S. (2003). Using replication and partitioning to build secure distributed systems. *Proceedings - IEEE Symposium on Security and Privacy*, *2003–Janua*, 236–250. https://doi.org/10.1109/SECPRI.2003.1199340