Doctoral Dissertation

Secure Multiple Group Data Deduplication in Cloud Data Storage

クラウドストレージにおける多重グループデータの安全な重複除外について

Ei Mon Cho

Graduate School of Science and Engineering,

Saitama University

Supervisor: Professor Takaomi Shigehara

March 2018

Doctoral Dissertation

Secure Multiple Group Data Deduplication in Cloud Data Storage

Ei Mon Cho

Graduate School of Science and Engineering,
Saitama University

Supervisor: Professor Takaomi Shigehara

March 2018

Abstract

With the tremendous growth of available digital data, the use of Cloud Service Providers are gaining much popularity, since these types of services promise to provide convenient and efficient storage services to end-users by taking advantage of a new set of benefits and savings offered by cloud technologies in terms of computational, storage, bandwidth, and transmission costs. In order to achieve savings in storage, cloud storage providers often employ data deduplication techniques to eliminate duplicated data. However, benefits gained through these techniques have to balanced against users' privacy concerns, as these techniques typically require full access to data.

In this thesis, we propose solutions for two secure multiple group setting data deduplication in cloud environments. Firstly, we propose a new framework DDUP-MUG (deduplication for the multiplegroup signature scheme) that allows one or more groups to access a file such that the cloud storage server can avoid duplicates according to the ownership of the file. The main goal of the primitive red is allowing individual management to multiple groups. We propose the group managers mainly manage the new entities and produce revocation lists for clients and the server respectively. We use Message-Locked Encryption (MLE) as an ingredient for deduplication and we provide new three protocols, namely UPL-Dup (for uploading a new message), EDT-Dup (for editing the existing message) and DEL-Dup (for eliminating the existing message) in the DDUP-MUG framework.

Furthermore, we propose a new primitive group signcryption for deduplication called verifiable hash convergent group signcryption (VHCGS) by adding the properties of group signcryption and the verification facilities for the storage server (third party). An interesting technique called signcryption has been proposed, in which both the properties of signature (ownership) and encryption are simultaneously implemented, with better performance than the traditional signature-then-encryption approach. According to the deduplication, we propose a new method for a group of users that can eliminate redundant encrypted data owned by different users.

Acknowledgments

First and foremost, praises and thanks to the *God*, the Almighty, for His showers of blessings throughout my research work to complete the research successfully.

Special thanks are due to my thesis supervisors Professor *Dr.Takeshi Koshiba* and Professor *Dr.Takaomi Shigehara* for their invaluable support throughout the hard moments of graduate school.

First I would like to express my deepest gratitude to my first supervisor *Dr.Takeshi Koshiba.* Thanks for his professional guidance during my PhD study. He inspired me much on the research direction, and gave me valuable ideas and suggestions on both research and future career. Without his great help, my research work would not proceed smoothly.

Special thank also goes to Professor *Dr.Takaomi Shigehara* for giving me an opportunity to become his student for my remaining year of my PhD student life.

I am also grateful to my dissertation committees, *Dr. Yutaka Ohsawa*, *Dr. Atsushi Uchida* and *Dr. Noriaki Yoshiura*, for their support, valuable feedback, and insightful ideas to this research.

My study would not have been accomplished without funds supported of *Nikkei Scholarship Foundation* and *Hirose International Scholarship Foundation.* Special thanks to *Hirose* scholarship committees that encourage me not only by financial but also by warmly kindly supported for my study since April 2015.

My heartfelt thank goes out to my family for their prayers, selfless love and great support to my study. Their love and care help me overcome difficulties and obstacles. Finally, my thanks go to all the people who have supported me to complete the research work directly or indirectly.

Contents

Abstract **i**

Acknowledgments **iii**

List of figures **v**

List of tables **vi**

Publications **vii**

Abbreviations **x**

Notations **xii**

1 Introduction **1**

1.1 Related Works 2

1.2 Motivating applications 7

1.3 Problem Formulation 9

1.4 Main Contributions 11

1.5 Structure of this Thesis 13

2 Cryptographic Mechanism 14

2.1 Public Key Encryption . . . 14

2.2 Hash Function . . . 15

2.3 Digital Signature . . . 16

2.4 Group Signature . . . 17

2.5 Signcryption . . . 18

2.6 Computational Primitives . . . 19

2.6.1 Integer Factorization . . . 19

2.6.2 Discrete-Logarithm . . . 20

2.7 Security Notions . . . 22

2.7.1 Steps to Achieving Provable Security . . . 22

2.7.2 Confidentiality Notion . . . 22

2.7.3 Unforgeability Notion . . . 26

2.7.4 Random Oracle Model . . . 28

3 Cloud Computing 30

3.1 Cryptographic Mechanisms of the Cloud . . . 32

3.1.1 Searchable Encryption . . . 33

3.1.2 Homomorphic Encryption . . . 33

3.1.3 Computing Aggregates over Encrypted Data . . . 34

3.2 Functional Encryption . . . 34

3.2.1 Order Preserving Encryption . . . 35

3.2.2 Identity-based encryption . . . 35

3.2.3 Attributed-based Encryption 36
3.2.4 Predicate Encryption 36
3.3 Verifiable Computing 37
3.3.1 Verifiable Outsourced Computation 37
3.3.2 Verifiable Storage 37
3.4 Other Tools 38
3.4.1 Format Preserving Encryption 38
3.4.2 Proxy Re-encryption 39
3.4.3 Secure Deduplication 39

4 Cloud Data Deduplication 41

4.1 Cloud Storage and Issues 41
4.2 Deduplication 42
4.2.1 Client-Side Deduplication 43
4.2.2 Server-Side Deduplication 43
4.2.3 Level of Data Deduplication 43
4.3 Data Deduplication Techinical Design Issues 44
4.3.1 Types of Data Deduplication 45
4.3.2 Client vs Server Side Deduplication 46
4.3.3 Single User vs Cross User Deduplication 46
4.4 Background 47
4.4.1 Convergent or Message-locked Encryption 47
4.4.2 Proofs of Ownership 50

5 Multiple Group Signature Setting **52**

5.1 Background 52

5.1.1 Group Signature 53

5.1.2 Group Signature from Unlikable Randomizable Signature . . 55

5.2 Management Policy of Proposed Scheme 55

5.3 Proposed Scheme Architecture 56

5.3.1 System Setup Protocol 58

5.3.2 Upload Protocol 59

5.3.3 Edit Protocol 60

5.3.4 Delete Protocol 62

5.3.5 Restore Protocol 63

5.4 Security Analysis 63

5.5 Chapter Summary 69

6 Multiple Group Signcryption **71**

6.1 Background 72

6.1.1 Signcryption 72

6.1.2 Original Signcryption Scheme 72

6.1.3 Public Verifiable Signcryption Scheme 73

6.1.4 Group Signcryption Scheme 74

6.2 Proposed Scheme Architecture 75

6.2.1 KeyGen 75

6.2.2 Join 76

6.2.3 Signcryption 76

6.2.4 Partial Unsigncryption 77

6.2.5 Unsigncryption 77

6.3 Application in Cloud Computing 78

6.3.1 System Setup Protocol 79

6.3.2 Upload Protocol 80

6.3.3 Download Protocol 81

6.4 Application in Mobile Data Transmission 81

6.4.1 System Setup 82

6.4.2 Signcryption 83

6.4.3 Partial-Unsigncryption 83

6.4.4 Unsigncryption 83

6.5 Security Analysis 84

6.6 Chapter Summary 88

7 Related Research and Future Improvements 90

7.1 Proxy Re-encryption 90

7.1.1 Motivation of Proxy Re-encryption 93

7.1.2 Syntax of our PRE scheme 94

7.2 Future Improvements 96

8 Conclusion 97

8.1 Discussion 97

References 100

List of Figures

1.1 Deduplication of the Cloud 2

1.2 Application of Amazon 7

1.3 Application of Apache SVN 8

2.1 Public Key Encryption 15

2.2 Digital Signature 16

2.3 Group Signature 18

2.4 Generic Signcryption 19

2.5 Chosen Plaintext Attack 23

2.6 Chosen Ciphertext Attack 26

4.1 Deduplication 43

4.2 Client Side Deduplication 44

4.3 Server Side Deduplication 45

4.4 Syntax of Traditional Encryption 48

4.5 Syntax of Message-Locked Encryption 49

5.1 Proposed DDUP-MUG Scheme 57

5.2 Illustration of Upload Protocol 59

5.3 Illustration of Edit Protocol 60

5.4 Illustration of Delete Protocol 62

6.1 Cloud Deuplication by using VHCGS 79

6.2 VHCGS Setup Protocol for Cloud Deduplication 80

6.3 VHCGS Upoad Protocol 80

6.4 VHCGS Download Protocol 80

6.5 Multiple Group SMS transmission by using VHCGS 82

7.1 Proxy Re-encryption Scheme 94

List of Tables

1.1 Comparison between deduplication techniques carried over encrypted data 6

5.1 Data Structure of a file at Server 56

8.1 Comparison of Our Proposed Schemes 98

Publications

Peer-Reviewed Papers

- Ei Mon Cho, Takeshi Koshiba, “Cloud Deduplication based on Multiple Group Signature Scheme”, (The 15th International Conference on Computer Applications/ ICCA 2017, Yangon, Myanmar, February 16, 2017), pages 34–41.
- Lwin San, Ei Mon Cho, Takeshi Koshiba,”Non- Transferable Proxy Re-encryption for Group Membership/ Non-Group Membership (full version)”, (The 15th International Conference on Computer Applications/ ICCA 2017, Yangon, Myanmar, February 16, 2017), page 321–327.
- Ei Mon Cho, Takeshi Koshiba, “Secure Deduplication in a Multiple Group Signature Setting” (The 31st IEEE International Conference on Advanced Information Networking and Applications 2017/ AINA2017, Taipei, Taiwan, March 27), page 811–818.
- Ei Mon Cho, Takeshi Koshiba, “Big Data Cloud Deduplication based on Verifiable Hash Convergent Group Signcryption” (IEEE International Workshop on Big Data Security and Services/ BigDataService2017, San Francisco, USA, April 2), page 265–270.
- Ei Mon Cho, Takeshi Koshiba, “Secure SMS Transmission based on Verifiable Hash Convergent Group Signcryption (Poster) (18th IEEE International Conference on Mobile Data Management/ MDM 2017, Daejeon, South Korea, May 30), page 332– 335.
- Ei Mon Cho, Lwin San, Takeshi Koshiba, “ Secure Non-Transferable Proxy Re-Encryption for Group Membership and Non-Membership (The 8th International Workshop on Trustworthy Computing and Security/ TwCSec-2017, Toronto, Canada), Lecture Notes on Data Engineering and Communications Technologies 7, Springer 2018), page 876–887.

- Ei Mon Cho, Lwin San, Takeshi Koshiba, "Non-Transferable Proxy Re-Encryption for Multiple Groups", to appear on The International Journal of Space-Based and Situated Computing.

Non Peer-Reviewed Papers

- Ei Mon Cho, Takeshi Koshiba, “Secure Deduplication for Multiple Group setting”, (2016 Symposium on Cryptography and Information Security, Kumamoto/ SCIS2016, Japan, January 27, 2016)
- Ei Mon Cho, Takeshi Koshiba, “Deduplication based on Verifiable Hash Convergent Group Signcryption”, (2017 Symposium on Cryptography and Information Security/ SCIS2017, Okinawa, Japan, January 27,2017)
- Lwin San, Ei Mon Cho, Takeshi Koshiba, “Non- Transferable Proxy Re-encryption for Group Membership/ Non-Group Membership”, (2017 Symposium on Cryptography and Information Security/ SCIS2017, Okinawa, Japan, January 27, 2017)

Abbreviations

ABE	Attribute-based Encryption
CDA	Chosen Distribution Attack
CDH	Computational Diffie-Hellman
CSP	Cloud Service Provider
DDH	Decisional Diffie-Hellman
DEDUP-MUG	Deduplication for Multiple Group Setting
DL	Discrete-Logarithm
DS	Digital Signature
FPE	Format Preserving Encryption
GM	Group Manager
GS	Group Signature
GSM	Global System for Mobile Communications
HCE	Hash Convergent Encryption
HE	Homomorphic Encryption
IBE	Identity-based encryption
IF	Integer Factorization
IND-CCA	Indistinguishability of Encryptions under Chosen Ciphertext Attack
IND-CCA2	Indistinguishability of Encryptions under Adaptive Chosen Ciphertext Attack
IND-CPA	Indistinguishability of Encryptions under Chosen Plaintext Attack
iMLE	Interactive Message-Locked Encryption
MLE	Message-Locked Encryption
OAEP	Optimal Asymmetric Encryption Padding
PDP	Proof of Data Possession

PE	Predicate Encryption
PoR	Proof of Retrievability
PoW	Proof of Ownership
PRE	Proxy Re-encryption
RSA	Rivest, Shamir, and Adelman
SE	Searchable Encryption
SMS	Short Message Service
URS	Unlinkable Randomizable Group Signature
VC	Verifiable Computation
VHCGS	Verifiable Hash Convergent Group Signcrytion

Notations

M^*	set of words $m_1, m_2, ...m_l, l \geq 0$, over M
$\{0,1\}^*$	set od bit strings of arbitrary length
1^k / 1^λ	constant bit string 1 of length k/λ
$a \oplus b$	bitwise XOR of bit strings a and b
$a \parallel b$	concatenation of strings a and b
$\mathbb{N}$	set of natural numbers: $\{1, 2, ...\}$
$\mathbb{Z}$	set of integers
$\mathbb{R}$	set of real numbers
$\ln(x)$	natural lograrithm of a real $x > 0$
$a \mid b$	$a \in \mathbb{Z}$ divides $b \in \mathbb{Z}$
$\mathbb{Z}_n$	residue class ring modulo n
$\mathbb{Z}_n^*$	units in $\mathbb{Z}_n$
$a \bmod n$	remainder of a modulo n
$a \equiv b \bmod n$	a congruent b modulo n
$\gcd(a, b)$	greatest common divisor of integers
Primes_k	set of primes of binary length k
P or $P(X)$	positive polynomial
$\text{prob}(\varepsilon)$	probability of an event ε
$\text{prob}(x)$	probability of an element $x \in X$
$\text{prob}(y \mid x)$	conditional probability of y aassuming x
$x \leftarrow_{\$} X$	x randomly selected from X
$x \overset{u}{\leftarrow} X$	x uniformly selected from X
$x \leftarrow X, y \leftarrow Y_x$	first x, then y randomly selected
$\text{prob}\ (... : x \leftarrow X)$	probability of ... for randomly chosen x
$H(\cdot)$	Hash Function
$H(X)$	uncertainty (or entropy) of X

Chapter 1

Introduction

With various advantages of cloud storage such as cost savings, accessibility, scalability, etc., users around the world tend to shift their invaluable data to cloud storage. As the data operations charges are increasing, it is an issue for cloud storage providers to provide efficient storage. Cloud storage providers use various techniques to improve storage efficiency, and one of the dominant technique occupied by them is deduplication, which claims to be saving 90% to 95% of storage [56]. Data Deduplication technique is developing as a simple storage optimization technique in distributed storage then widely adopted in primary storage as well as larger storage areas like cloud storage area as shown in Figure.1.1. Now, data deduplication is widely used by various cloud storage providers like Dropbox [57], Amazon S3 [8], Google Drive [71], etc.

Data once conveyed to the cloud servers, it is beyond the security bounds of the data owner. Therefore, most of data owners prefer to outsource theirs in an encrypted format. Data encryption by data owners eliminates cloud service providers chance of deduplicating it because encryption and deduplication techniques have clashing strategies, i.e., data encryption with a key converts data into an unidentifiable format called ciphertext thus encrypting, even the same data, with different keys may output in different ciphertexts, happening deduplication less achievable. However, performing encryption is essential to make data secure, on the other hand, performing deduplication is essential for achieving optimized outsourced storage. Therefore, deduplication and encryption need to work in hand to hand to ensure secure and optimized storage [5]. Various approached and methods used for secure deduplication over encrypted data are studied in this thesis.

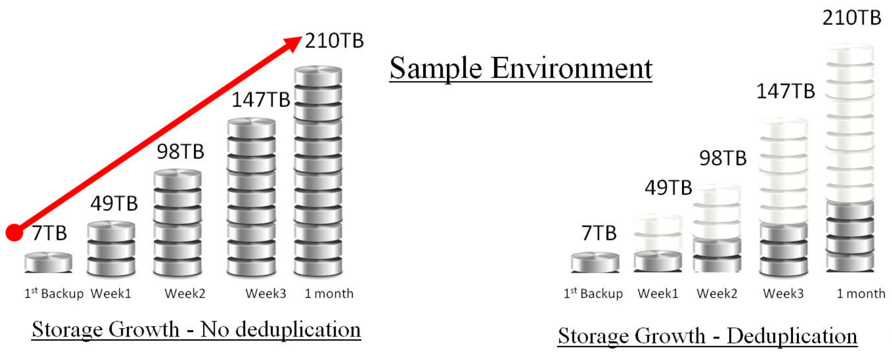


Figure 1.1: Deduplication of the Cloud

Deduplication is basically a compression technique for removing redundant data. Chapter 4 explains the deduplication process before storing data onto memory. Deduplication is widely is used various applications like backup, metadata management, primary storage, global system for mobile coomunication (GSM) etc. to control the system in order to achieve the desired properties.

In this chapter, we address the main problems considered in this thesis by some motivating applications and our main contributions, before giving a mathematical formulation.

1.1 Related Works

This section overviews some research works in the literature on secure data deduplication that are relevant to the scope of this thesis [5].

Bellare et al. [20] propose an encryption scheme where key for encryption and decryption are derived from the message. MLE key generation algorithm matches the message $\boldsymbol{M}$ to a key $\boldsymbol{K}$, and more, the encryption algorithm create ciphertext $\boldsymbol{C}$ of the message using key $\boldsymbol{K}$. Ciphertext $\boldsymbol{C}$ is then mapped to a tag $\boldsymbol{T}$; and the tag is for the duplication checking by the server. Keys used in MLE scheme are of fixed and shorter length, therefore, result will give in much storage overhead.

Chen et al. [45] set forward a method to accomplish dual level source-based deduplication of large encrypted files with block key management and “Proof of Ownership” [87, 140]. Chen claims that MLE proposes for target based file-level deduplication and extending it to dual-level deduplication needs more metadata management. In BL-MLE scheme for the given input file, a master key and the block keys for each message block in the file are generated. With tag generation algorithms, both file tags and block tags are generated and these tags are used validating the equality of blocks and files ensuring the security of the file. Ownership of files or blocks proved and verified by using “PowPrf” and “PowVrf” algorithms in this approach.

In [63], encryption and decryption data are performed by the client and the key for encryption and decryption are provided by a key server located at cloud storage provider. Homomorphic encryption is used as the key management scheme in this approach. Firsly, data encryption key is generated by the initial file uploader and another distributed consequent verified uploader by the key server. Data encryption key used for encryption are encrypted with the hash of file content. Both encrypted data and encryption keys are sent to the storage server. HEDup supports the privacy while enabling deduplication. Key server may become a bottleneck discussed in this approach when the quantity of clients increase for the large-scale deployment, and a decentralized deployment of key server is supposed as a solution.

In [20] Bellare et al. guarantee that Message-Locked Encryption [46] is subject to brute force attack and proposes another architecture called DupLess for brute force attack. The client gets message-based keys, for encryption, from key server via an Oblivious Pseudorandom function (OPRF) protocol. With OPRF, the public key for encryption is shared among the clients where the secret key resides with key server. With this method attackers’ cost of attack increased and a chance is eliminated.

Puzio et al. in [112] propose ClouDedup, a secure storage service which provides block-level deduplication and data confidentiality at the same time using convergent key encryption [56] added with block-level key management. Design of ClouDedup intends to prevent well-known attacks against convergent encryption by embedding a user authentication methods and access control mechanisms. Therefore, a server encryption is used on top of convergent encryption done by the user. For each data segment, a signature is linked to it, and need to be verified for retrieving it. To deal with block-level key management a metadata manager(MM) has been added to architecture. MM uses file table, pointer table and signature table. File table is to store meta data about file; pointer table is to manage storage and a signature table are to store meta data about signature for meta data management.

Bugiel et al. in [38] propose an approach that mainly involves two components: a trusted cloud and a commodity cloud. The trusted cloud is responsible for data encryption and the commodity cloud performs the verifying operations. Security operations are performed by trusted cloud and put the queries to outsourced data are processed by commodity cloud. This approach guarantees the various security issues like leakage of the data, computation manipulations, etc.

In [88] Li et al. propose a hybrid cloud to ensure the security of the deduplication which involves private cloud for providing tokens to access encrypted data in the cloud. Here, data encryption technique is employed by convergent encryption [56] and PoW [75, 140] is used to ensure ownership eligibility to deduplicate the file. In [128] Storer et al. propose to provide a single server and distributed storage systems with data security and space savings. With this method, encryption key is done out of data chunk. Even a full compromise of the system cannot reveal which data chunk is owned which user when the decryption information is encrypted with client's private keys. There are two models for secure deduplication in this method which are the authenticated model and the anonymous model. The authenticated model is similar convergent key structure [56]. The anonymous model hides identities of both authors and readers.

Li et al., in [88] aim at solving the problem of exposing and deduplicating for the sensitive data. With this approach, data chunks are distributed among multiple cloud servers. Moreover, to ensure tag consistency and data confidentiality, a deterministic secret sharing scheme is proposed in this distributed storage system. Opposite side of the conventional deduplication-encryption method, a secret sharing scheme is used instead of the encryption method. Furthermore, a ramp secret sharing scheme [33, 120] is used for the key management.

The main goal of [147], is to address the problem of huge key space overhead and to against brute force attack. For this objective, User Aware Convergent Encryption (UACE) and Multi-Level Key Management (MLK) are used. With UACE, cross-user file-level and single user block-level deduplication is achieved in this approach. File level keys and convergent encryption keys are generated by using a server aided method whereas chunk level keys are operated via a user-aided method. For reducing large key space, chunk keys are encrypted using file-level keys, therefore, a number of sharing users are increased but the key space is not increased. Furthermore, to eliminate the chance of single point of failure, this method uses multiple key servers, equipped with share-level keys that are generated out of file level keys and Shamir's secret sharing scheme [124] is used to communicate with these distributed servers.

In [127], data are differentiated based on popularity. A popular data denotes which is shared among multiple user and assumed to less sensitive and actively included in for deduplication with weaker security. On the other hand, unpopular data provided with security with semantically secure encryption. In [138] Xu et al. propose an approach that does with a weak leakage-resilient [75] than for cross-user client-side deduplication and supporting the security from outside adversaries and "honest-but-curious" cloud storage service providers.

Li et al. [87] address the issue of efficiently and reliably managing a large number of convergent keys for secure deduplication. With this De-key approach, both file level and block level deduplication is supported. Li et al. propose a base line approach where user maintains a master key to encrypt the convergent keys and develops De-key approach. With baseline approach, the user need to protect and manage a large set of master keys, which a tedious task. In De-key approach, user does not need manage any keys, however, needs to keep the distribute convergent keys among multiple servers. Ramp secret sharing scheme is used by De-key approach for securely sharing convergent keys.

Table.1.1 does comparison between various methods, to make deduplication work with encrypted data [6].

Approach	Encryption Scheme	Deduplication Strategy used
MLE [20]	Message locked encryption	File level
BL-MLE [46]	Block Level Message locked encryption	File level & Block level
DupLESS[27]	Enhanced Message level encryption to support security against Brute force attack	File level
ClouDedup[112]	Convergent encryption with added access control mechanisms	File level
Reliable Deduplication[86]	Convergent encryption	Block level
Twin clouds [38]	Convergent encryption	File level
Hybrid clouds [87]	Convergent encryption	File level
Secure Deduplication [128]	Convergent encryption	File level
Popularity Deduplication[127]	Symmetric encryption based on popularity	File level
SecDep [147]	User aware convergent encryption	File level & chunk level

Table 1.1: Comparison between deduplication techniques carried over encrypted data

1.2 Motivating applications

A few illustrative examples will be presented here to demonstrate the ubiquitous multi-group setting in cloud computing applications, and to motivate the problems considered in this thesis. The examples will highlight some of the shortcomings of the state of the art controllers for multi-group systems, which will be addressed in this thesis.

Example 1.1 (Multiple Group in Online Shopping Scenario) There are many reasons why users would choose to have more than one account in the online shopping. Users may have multiple accounts on the Amazon [8], ebay and so on. For example, all sellers are expected to adhere to the policies when listing products on Amazon.com. Buyers and sellers may communicate with one another via the Buyer-Seller Messaging Service, which assigns unique Amazon-generated email addresses to both parties. Duplicate product detail catalog is created by the Amazon.

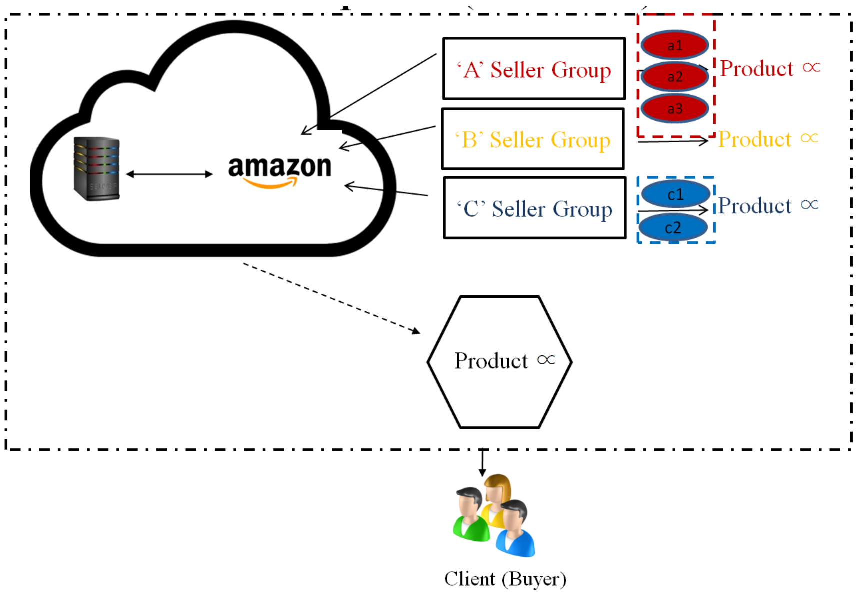


Figure 1.2: Application of Amazon

Operating and maintaining multiple seller central accounts is prohibited. If you have a legitimate business need for a second account, seller can apply for an exception to this policy as shown in Figure. 1.2.

Example 1.2 (Multiple Group in Subversion) For years, Apache Subversion or SVN was the most popular version control system [129]. SVN is a centralized version control system. Basically, software developers use version control for storing and tracking changes in different types of files, such as source code and documentation. This enables multiple developers to work effectively on the same codebase without messing up each others' work. If somebody makes a mistake or something unexpected happens, version control ensures that the latest working version of the code can be restored. Version control systems can be roughly divided into two categories: distributed version control systems (DVCS) and centralized version control systems (CVC). SVN falls into the latter category. Centralized version control system means that the version history is stored in a central server. When a developer wants to make changes to certain files, they check out the files from the central repository to their own computer. After the developer has made changes, they commit the changes back to the central repository. The complete revision history in the repository is not copied to the developer's computer, as it would be in a distributed version control system. Subversion deduplication is enabled by default, although it can be toggled the setting in the repository's file.

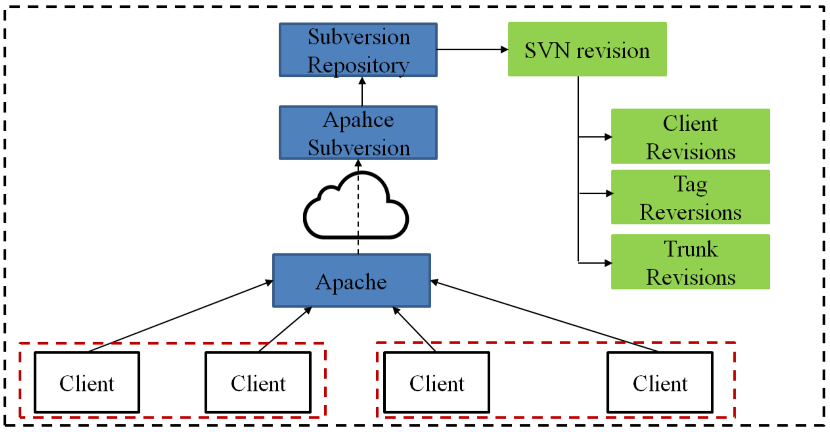


Figure 1.3: Application of Apache SVN

The most common problems that developers have towards SVN is its subgroup controlling users. Branches allow to work on multiple versions of the code simultaneously. In SVN, branches are created as directories inside the repository and this directory structure is the reason why developers are less than the fond of sub-version's branching model Figure.1.3.

The issue becomes clear when a single user of a group interacts to the others and share their data. Frequently, tasks can benefit from collaboration, which would not only make productivity gains, but create a more engaging experience for the entire group. Thus, it begs the question if combining these two or more groups would inherit the advantages of both systems.

Supporting data sharing and simultaneous interaction among multiple independent groups is challenging and presents a problem that needs a very general solution. Such a scheme would need to support one individual and many individuals; one group and many groups; or, a mix of both. Distinct group may wish to collaborate among themselves, or subsets may branch off and choose to work independently. Effectively, groups are interacting on the cloud asynchronously, working on different, even loosely coupled,tasks. Designing a scheme that can support such flexibility requires some research into how groups share securely in same cloud storage.

This thesis explores the idea of supporting collaboration between mutiple individual groups in parallel on shared cloud storage. To this end, this work details accounts of secure duplication at cloud using two user studies to gain a better understanding of what group signature support and group signcryption. Based on these detailed observations, this thesis aspires to establish fundamental design specifications for cross group deduplication systems that will create a more engaging client experience for cloud user. More specifically, to inform the design of cloud environment for multiple groups, this thesis presents accounts about how these groups interact with themselves and others securely.

1.3 Problem Formulation

Message-locked encryption [20] is a new relatively theoretical analysis as a new cryptographic primitive to capture convergent encryption (CE). MLE describes the models of all existing convergent encryption schemes, and it gives the first security definitions of a convergent encryption with cryptographic treatment. It basically identifies two attacks on the schemes: tag consistency (TC) and strong tag consistency (STC). Tag consistency means that cannot compute the tag for $file_1$ and use that encrypt $file_2$ (duplicate faking attack). And strong tag consistency (STC) means that an attacker cannot create an empty file and when a user tries to store a file that has the same tag with the empty file it keeps storing the empty file. So, STC protects against erasure attacks. After modifying the weaknesses of existing schemes, [20] produces a new one that is one pass (key generation, tag, encryption in one time) by randomizing the encryption. Each user derives the same tag for the same user but they use different keys to encrypt. The XOR of the randomize keys with the tag is appended to the ciphertext. This is called randomized convergent encryption (RCE) scheme which is a divergent form of MLE.

Furthermore, one security issue of original convergent encryption is, it is vulnerable to off-line brute force attack. If the adversary knows the entire message space, it can sample each message, computes the hash, encrypts with the computed hash (key) and compares the ciphertext of sampled message with the target ciphertext. If both are the same, the adversary can deduce the sampled message equals the message underlying the target ciphertext. This type of attack has been recognized by Bellare et al., [27] and they formalized a semantic security definition under unpredictability assumption (i.e., not allow the adversary to predict and sample message). The following work DupLESS [27] prevents the off-line brute force attack by introducing a third party entity for co-generating the encryption key, i.e., the encryption key depends on both the message content and a system-wide secret key, which is kept by the third party, in case, the outside adversary (not accessing the third-party entity) cannot launch off-line brute force attack.

One possible weakness of convergent encryption or message-locked encryption is vulnerable to statistical attack. Although the RCE scheme can generate random ciphertext, the tag for message must be deterministic. The tag consistency defined in Bellare et al. [20] requires, if two messages are the same, they must have identical tag. The tags essentially reflect the probabilistic distribution of plain messages. If the message space is with limited min-entropy, and the adversary has some pre-knowledge about the distribution of the message space, it might successfully guess some messages with significant probability.

Another existing issue of convergent encryption or message-locked encryption is that the encryption key is fixed. In case the encryption key need be changed, such as revoking access right. e.g. $Alice$ grant access right to someone in her group, the digital envelop containing encryption key has been sent to her, but she wants to remove grant access right to one person, then she wants to revoke the access right. If the scheme can modify revoking access right among the nominated persons, revoking access right can be apply by group signature, but CE or MLE is not applicable for this purpose. MLE is only on the theoretical side and describe standard model solutions, and it make connections with deterministic encryption, hash functions secure on correlated inputs.

1.4 Main Contributions

In this thesis, we aim to address to the multiple group setting of secure deduplication for cloud environment. Two main contributions of this thesis are twofold.

First contribution of this thesis is the DDUP-MUG (Deduplication for multiple group signature) protocols for secure deuplication with group signature and message-locked encryption properties. By using unlinkable randomizable signature functions, we prove the security of every group control where each group of member has individual group manager connected into CSP as a verifer of deduplication and ownership. DDUP-MUG is characterized by server side deduplication. The above results have been published in the following proceedings:

- Ei Mon Cho, Takeshi Koshiba, “Cloud Deduplication based on Multiple Group Signature Scheme”, (The 15th International Conference on Computer Applications/ ICCA 2017, Yangon, Myanmar, February 16, 2017), pages 34-41
- Ei Mon Cho, Takeshi Koshiba, “Secure Deduplication in a Multiple Group Signature Setting” (The 31st IEEE International Conference on Advanced Information Networking and Applications 2017/ AINA2017, Taipei, Taiwan, March 27), page 811-818
- Ei Mon Cho, Takeshi Koshiba, “Secure Deduplication for Multiple Group setting”, (2016 Symposium on Cryptography and Information Security, Kumamoto/ SCIS2016, Japan, January 27, 2016).

We briefly describe DDUP-MUG in Chapter 5.

The second contribution is the VHCGS (Verifiable Hash Convergent Group Signcryption) for deduplication systems. We proposed a new group signcryption scheme for proxy verifiable with deduplicable properties. We analyze the security of the proposed protocols through signcryption and hash convergent encryption, and give necessary and sufficient criteria. The proposed scheme are proven to verifiable signcryption in the multiple group setting. The above results have been published in the following proceedings:

- Ei Mon Cho, Takeshi Koshiba, "Big Data Cloud Deduplication based on Verifiable Hash Convergent Group Signcryption"(IEEE International Workshop on Big Data Security and Services/ BigDataService2017, San Francisco, USA, April 2), page 265-270
- Ei Mon Cho, Takeshi Koshiba, "Secure SMS Transmission based on Verifiable Hash Convergent Group Signcryption (Poster) (18th IEEE International Conference on Mobile Data Management/ MDM 2017, Daejeon, South Korea, May 30), page 332- 335
- Ei Mon Cho, Takeshi Koshiba, "Deduplication based on Verifiable Hash Convergent Group Signcryption", (2017 Symposium on Cryptography and Information Security/ SCIS2017, Okinawa, Japan, January 27,2017).

We briefly describe VHCGS in Chapter 6.

Furthermore, we proposed a proxy re-rencryption scheme (PRE) which is relatively related in cloud deduplication. With future improvement, our new PRE scheme will support the cloud deuplication for multiple group setting. As result, we published in the following proceedings:

- Ei Mon Cho, Lwin San, Takeshi Koshiba, "Non-Transferable Proxy Re-Encryption for Multiple Groups", to appear on The International Journal of Space-Based and Situated Computing
- Ei Mon Cho, Lwin San, Takeshi Koshiba, " Secure Non-Transferable Proxy Re-Encryption for Group Membership and Non-Membership (The 8th International Workshop on Trustworthy Computing and Security/ TwCSec-2017, Toronto, Canada), page 876–887

- Lwin San, Ei Mon Cho, Takeshi Koshiba, “Non- Transferable Proxy Re-encryption for Group Membership/ Non-Group Membership”, (2017 Symposium on Cryptography and Information Security/ SCIS2017, Okinawa, Japan, January 27, 2017)
- Lwin San, Ei Mon Cho, Takeshi Koshiba, ”Non- Transferable Proxy Re-encryption for Group Membership/ Non-Group Membership (full version)”, (The 15th International Conference on Computer Applications/ ICCA 2017, Yangon, Myanmar, February 16, 2017), page 321–327.

We briefly describe PRE in Chapter 7.

1.5 Structure of this Thesis

The remaining chapters of this thesis are organized as follows. Chapter 2 presents some background in public key encryption,hash function, digital signature, group signature, signcryption, other computational primitives and security notions of relevance for this thesis. Cryptographic mechanism for cloud computing are presented in Chapter 3. In Chapter 4, secure deduplication methods for cloud environment are presented. In Chapter 5, deuduplication for multiple group setting (DEDUP-MUG) is presented. In Chapter 6, a multiple group signcryption scheme (VHCGS) which is useful for secure deduplication is presented. We presented a proxy re-encryption (PRE) scheme which related research of the cloud deduplcation in Chapter 7. The thesis is concluded in Chapter 8, which also contains a discussion on possible future research directions.

Chapter 2

Cryptographic Mechanism

The basic idea of secure communication is public key cryptography which are using the public keys. Every individual's key is sperated into two sections: a public key of encryption for everyone and a secrete key for decryption which is kept secrete by the owner. In this chapter, we are ready to describe the very fundamental notions of security. They will be useful throughout the book.

2.1 Public Key Encryption

Public key encryption, in which a message is encrypted with a recipient's public key. The message cannot be decrypted by anyone who does not possess the matching private key, who is thus presumed to be the owner of that key and the person associated with the public key. In public key encryption scheme, the communication partners do not share a secrete key. Each user has a pair of keys: a *secrete key sk* known only himself and a *public key pk* known by everyone. Figure 2.1 illustrates an outline of a public key encryption scheme.

A general public key encryption scheme can be described as follows.

- Key Generation: The receiver *Bob* creates his private and public key pair, which we denote by sk_{Bob} and pk_{Bob} respectively.
- Encryption: Using *Bob*'s public key pk_{Bob}, the sender *Alice* encrypts her message M, which we call a "plaintext", and obtains a ciphertext C.

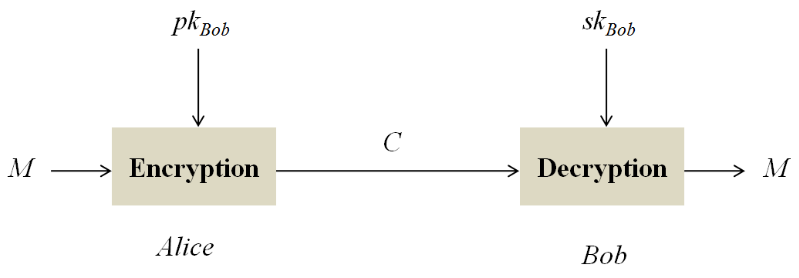


Figure 2.1: Public Key Encryption

- Decryption: Upon receiving the ciphertext C from *Alice*, *Bob* decrypts it using his private key sk_{Bob} to recover the plaintext M.

2.2 Hash Function

A hash function [52] is a compression function H that take an arbitrary length string and outputs a shorter string. Hash value are used to check the integrity of public keys. Mathematically, a hash function is a function

$$h : \{0,1\}^* \longrightarrow \{0,1\}^n, m \longrightarrow h(m)$$

.

For cryptographic applications we must ensure that the mapping H does not produce predictable collisions and that the precise mapping is hard to determine. The most important security properties of a hash function are:

- Collision resistance: it should be computationally infeasible to find two distinct messages x and z such that $H(x) = H(z)$; that is, to find two messages that hash to the same value;
- Pre-image resistance: given the result $y = H(x)$ of H applied to a randomly

chosen message x, it should be computationally infeasible to find a value z such that $H(z) = y$; Second pre-image resistance: given a message x it should be computationally infeasible to find another message z such that $H(x) = H(z)$.

Hash functions find numerous uses in cryptography, from dedicated applications such as password protection through to acting as components in more complex cryptographic tools such as digital signature schemes.

2.3 Digital Signature

Another fundamental public key cryptographic scheme is a digital signature which is firstly proposed by [35]. The ability to construct a digital signature scheme is a great advantage of public key cryptography over symmetric key cryptography. A valid digital signature gives a receiver to trust which the message was created by a known sender, that the sender cannot deny having sent the message (non-repudiation), and that the message can not be changed (integrity). Digital signature is a standard component of most cryptographic protocol, and is commonly used for software distribution, financial transactions and contract management software. Figure 2.2 illustrates an outline of a digital signature scheme.

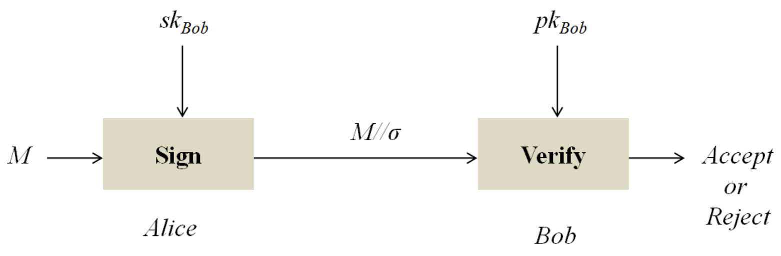


Figure 2.2: Digital Signature

A general digital signature scheme can be described as follows.

- Key Generation: The signer *Alice* creates her private and public key pair, which we denote by sk_{Alice} and pk_{Alice} respectively.
- Signature Generation: Using her private key sk_{Alice}, *Alice* creates a signature σ on her message M.
- Signature Verification: Having obtained the signature σ and the message M from $Alice$, the verifier Bob checks whether σ is a genuine signature on M using *Alice's public key*, pk_{Alice}. If it is, he returns "*Accept*". Otherwise, he returns "*Reject*".

2.4 Group Signature

A group signature scheme allows a member of a group to sign message anonymously on behalf of the group. In the case of later dispute a designated group manager can revoke the anonymity and identify the originator of a signature. Chaum and van Heyst [43] proposed a new type of signature scheme for a group of entities, called group signatures. Such allows a group member to sign a message to sign a message on the group's behalf such that everybody can verify the signature but no one can find out which group member provided it. However, there is a trusted third party, called the group manager, who can reveal the identity of the originator of a signature in the case of later dispute. This act is referred to as "opening" a signature or also as revocation of a signer' s anonymity. The group manager can either be a single entity or a number of coalitions of several entities (e.g., group members). This concept can be generalized to allow designated subsets of all group members to jointly sign a message on behalf of the group.

Group signatures could for instance be used by a company for authenticating price list, press release or digital contracts. The customers need to know only a single company public key to verify signatures. The company can hide any internal organizational structures and responsibilities, but can still find out which employee (i.e., group member) has signed a particular documents.

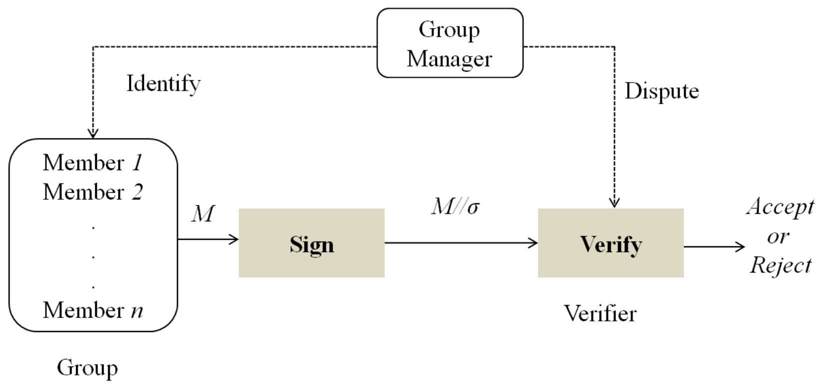


Figure 2.3: Group Signature

2.5 Signcryption

Signcryption is a new paradigm in public key cryptography that simultaneously fulfills both the functions of digital signature and public key encryption in a logically single step, and with a cost significantly lower than that required by the traditional "signature and encryption" approach.

In order to send a confidential letter in a way that it cannot be forged, it has been a common practice for the sender of the letter to sign it, put it in an envelope and then seal it before handing it over to be delivered. Discovering Public key cryptography has made communication between people who have never met before over insecure network. Before sending a message, the sender has to do the following:

1. Sign it using a Digital Signature (DS) scheme
2. Encrypt the message and the signature using a private key encryption algorithm under randomly chosen message encryption key.
3. Encrypt the random message encryption key using the receiver's public key.
4. Send the message.

Signcryption can be defined as a combination of two schemes; one of digital signatures and the other of public key encryption.

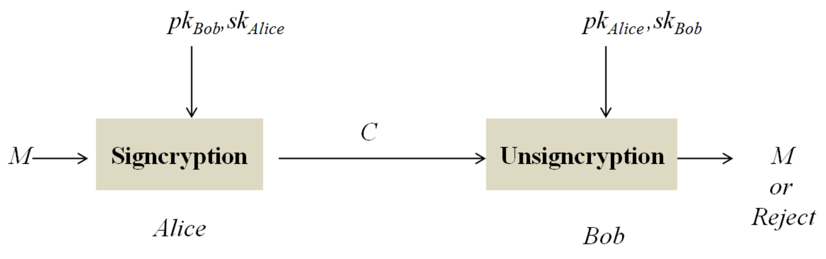


Figure 2.4: Generic Signcryption

2.6 Computational Primitives

The security of a public key cryptographic scheme is depended on the hardness of a certain computational issue. Although several computational problems have been proposed, we overview the *"Integer Factorization"* and *"Discrete-Logarithm"* problems, which are the most widely-used computational problems in the typical cryptographic schemes.

2.6.1 Integer Factorization

Integer Factorization problem, which is commonly call the *"IF problem"*, can be described as follows:

IF: Given $N = pq$ where p and q are large primes, find p and q.

A ciphertext C modulo integer $N = pq$, which is identified to the above *"IF problem"*. Here is that equivalence between computing e^{th} root of an element modulo integer N and solving the *"IF problem"* has not been demonstrated until now. If the *"IF problem"* can be easily (namely, "in polynomial time") solved, then computing $d = e^{-1} \bmod \phi(N)$ can be computed also easily since $\phi(N) = (p-1)(q-1)$ can be efficiently done if p and q are known. Besides, no answer has been given yet for the question whether one have to be factor N to compute $d = e^{-1} \bmod \phi(N)$. For this point, the computational problem of the RSA encryption [118] and signature schemes is categorically called the RSA problem. Certainly, factoring a large integer is one of the fascinating subjects in computational number theory. The quadratic sieve , elliptic curve , and number field sieve methods developed by computational number theorists are now used as factoring algorithms in practice. Among them, the number field sieve method, is specially used by many cryptographers due to its asymptotic running time faster than those of the quadratic sieve and elliptic curve methods, and its effectiveness to parallelization as shown in factoring a 512-bit RSA modulus [42].

2.6.2 Discrete-Logarithm

Addition widely-used computational problem is the Discrete-Logarithm problem, *"DL problem"*, which can be informally defined as follows:

DL: Given a finite cyclic group $G = \{g^0, g^1, g^2, ..., g^{m-1}\}$ where $m = |G|$ is the order of G, and a random element $r \in G$, find the unique integer $i \in \mathbb{Z}_m$ such that $r = g^i$.

The DL problem is figured by Diffie and Hellman [35] like the following intriguing key exchange protocol: $Alice$ and Bob share the common cyclic group G, where $|G| = m$, and its generator g. $Alice$ then chooses a uniformly at random from $\mathbb{Z}_m$, computes g^a, and sends this to Bob. Likewise, Bob selects b uniformly at random from $\mathbb{Z}_m$, computes g^b, and sends this to $Alice$. Since $Alice$ and Bob have their private key a and b respectively, they can calculate the same (secret) key g^{ab}. However, the attacker for the above key exchange protocol has two elements in the group G, that is, g^a and g^b. Since the attacker possess the one more additional information g^b, the security of the Diffie-Hellman key exchange protocol is not same with the hardness of the DL problem. For this reason similar to the case of the RSA problem, the computational problem used in the Diffie-Hellman key exchange protocol is specifically called the *"Computational Diffie-Hellman (CDH) problem"*, which can be shown as follows:

CDH: Given a finite cyclic group $G = \{g^0, g^1, g^2, ..., g^{m-1}\}$ where $m = |G|$ is the order of G, and $g^a \in G$ and $g^b \in G$ for random $a, b \in \mathbb{Z}_m$, computes $g^{ab} \in G$.

After the appearing of the CDH problem, researchers realized that the CDH problem by itself is not enough for many cryptographic schemes to be proven secure in that the attacker may have a chance to get some valuable information about the key g^{ab} even though its entire part cannot be revealed. This is the motivation for defining the following *"Decisional Diffie-Hellman (DDH) problem"*:

DDH: Given a finite cyclic group $G = \{g^0, g^1, g^2, ..., g^{m-1}\}$ where $m = |G|$ is the order of G, and $g^a \in G$, $g^b \in G$ and $g^c \in G$ for random $a, b, c \in \mathbb{Z}_m$, decide whether $c = ab \in \mathbb{Z}_m$.

If one can solve the DL problem easily, he can easily solve the CDH problem too. Similarly, if one can solve the CDH problem easily, he can easily solve the DDH problem. However, the reverse of this reasoning does not generally hold. Detail descriptions are referred to [95] and [35] for more discussions on this problem.

Note that in practice, the group G can be implemented using the subgroup of $\mathbb{Z}_p^* = \{1, 2, ..., p^{-1}\}$ of order q such that $p = aq+1$ for primes p and $q > p^{1/10}$, or the group of points on certain elliptic curves of order q [84, 97] for efficiency. As attack algorithms for the DL problem in a general group G(e.g., regardless of whether G is the subgroup of $\mathbb{Z}_p^*$ or the group of points on elliptic curves), Shank's "baby-step, giant step" and Pollar's "rho" methods are used. As a sub-exponential algorithm for solving the DL problem in $\mathbb{Z}_p^*$, the "index calculus" method is famous.

Finally, we observe that there have been a large number of cryptographic schemes based on the above problems. There are some examples which include the digital signature schemes based on the DL problem such as ElGamal [59], Schnorr [121], and digital signature standard (DSS) [100]; the public key encryption schemes based on the CDH problem such as Pointcheval [111] and Baek-Lee-Kim [15]; the public key encryption scheme based on the DDH problem such as ElGamal [59], Tsiounis-Yung [130], and Cramer-Shoup [50].

2.7 Security Notions

2.7.1 Steps to Achieving Provable Security

Provable security evaluates the security of a given cryptographic scheme by citing a reduction between the properly defined security notion of the scheme and the primary primitive which is secure. Actually, the point of a reduction is originally from the theory of computation. Casually, it is a way of converting one problem to another problem in such a way that a solution to the latter problem can be used to solve the former one [123]. Bellare et al. [22] explains exactly how to achieve provable security, which can be summarized as the following steps:

1. Set up a security goal, such as confidentiality via encryption or authenticity by using signature;
2. Construct a formal attack model and define what it means for a cryptographic scheme to be secure;
3. Show by a reduction that the only way to break the security of cryptographic schemes is to solve computationally hard problems or break other primitives.

Indeed, setting up security goals and constructing relevant attack models, commonly, formulating right definitions for the security of cryptographic schemes is important by itself. Over the last two decades, researchers have been proposing various security notions (definitions of security) for cryptographic schemes either in public key or symmetric key setting. In the next sections, we survey some important confidentiality, unforgeability notions and random oracle model widely used in public key cryptography.

2.7.2 Confidentiality Notion

In case, we reviewed that the $\boldsymbol{RSA}$ encryption scheme [118] is secure in the "$\boldsymbol{one-wayness}$" sense, unless the attacker obtains $\boldsymbol{Bob}$'s private key, he cannot recover the entire of plaintext. But we are not sure whether the $\boldsymbol{RSA}$ encryption scheme is secure in situations where something more than one-wayness is required. Indeed, the $\boldsymbol{RSA}$ encryption scheme is not secure in the following situation; assume that members of a committee use a confidential on-line poll to decide on some course of action and the RSA encryption scheme is employed to encrypt the member' votes. However, there is an additional assumption that the committee members should use one of the predetermined messages, "Yes" and "No", to create their ciphertext. After encrypting one of those messages, each of the members sends over the ciphertext to a chairperson, who is not supposed to know who votes for "Yes" or "No". However, the corrupted chairperson will know who has voted for which by re-encrypting the guessed plaintext ("Yes" or "No") to see if the resulting ciphertext matches what he has. Since there are only two kinds of ciphertexts C_{Yes} and C_{No} which encrypt "Yes" and "No" respectively.

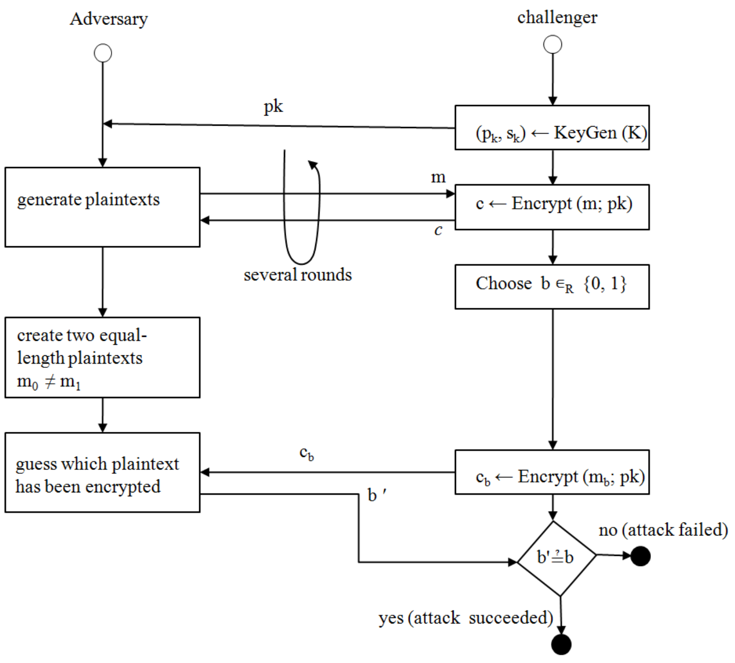


Figure 2.5: Chosen Plaintext Attack

The main motivation for Goldwasser and Micali's [69] is the notion of confidentiality for public key encryption called "semantic security ", same as a "(polynomial) indistinguishability of encryptions under chosen-plaintext attack (IND-CPA) [70]". The association of this notion is that a ciphertext should not disclose any partial information about the plaintext apart from its length to any attacker whose computational power is polynomially bounded. One can still imagine even a harsher situation where the attacker may be supplied with a decryption oracle (algorithm), from which he gets decryptions of some ciphertexts of his choice even if he is not in possession of the decryption key.

This is also called "chosen ciphertext attack", developed and evolved in the series of papers by Naor and Yung [99] , Rackhoff and Simon [116] , and Dolev, Dwork, and Naor [55]. When conducting chosen ciphertext attack's chance to get useful information about a plaintext given its encryption which is called a "target ciphertext" can be very high in some schemes.

Assume that Bob's public and private keys are (N, e) and (p, q, d) such that $N = pq$, where p and q are large primes chosen at random, and $ed = 1 \; mod \; \phi(N)$ where $\phi(N) = (p-1)(q-1)$. Suppose that attacker has obtained a target ciphertext $C = M^e \quad mod \quad N$ which encrypts M. Now, attacker just chooses an arbitrary message $R \in N$, computes $C' = R^e C$, and queries it to the decryption oracle (algorithm) and gets $M' = C'^d$. attacker then computes M'/R. Since

$$M' = C'^d = (R^e C)^d = RC^d = RM,$$

M'/R is the message M which is the plaintext of C.

Practically, RSA-OAEP, a converted version of the RSA encryption scheme using Bellare and Rogaway's [25] "Optimal Asymmetric Encryption Padding (OAEP)", is widely used. In fact, OAEP fixes the above two security problems of the RSA encryption scheme. More precisely, it makes the RSA encryption scheme provably secure in the "indistinguishability of encryptions under chosen ciphertext attack (IND-CCA)" sense under the random oracle assumption which will be discussed in detail in next section. We review that the original proof given in [25] that OAEP applies to any one-way trapdoor permutation was later found to be false by Shoup [122] . Shortly after Shoup's work, Fujisaki, Okamoto, Pointcheval, and Stern [65] confirmed that OAEP in fact applies to the sole RSA function.

Now, we look into the "IND-CCA" notion in the context of modern cryptography. In the current literature, the "IND-CCA" notion is usually described in terms of the following game in which the attacker and the "Challenger" interact each other:

- *Phase 1:* The Challenger generates a private/public key pair and all the necessary common parameters of a given public key encryption scheme in the prescribed manner. The Challenger then gives the public key and the common parameters to the attacker while keeps the private key secret.

- *Phase 2:* The attacker queries a number of ciphertexts to the Challenger to obtain their decryptions. Upon receiving each of the ciphertexts, the Challenger decrypts it using the private key he generated in Phase 1 and returns the resulting decryption to the attacker.

- *Phase 3:* The attacker chooses two equal-length plaintexts (M_0, M_1) and gives them to the Challenger. Upon receiving M_0 and M_1, the Challenger chooses one of them at random and computes its encryption. The Challenger returns the resulting ciphertext (called a "target ciphertext") to the attacker.

- *Phase 4:* The attacker again queries a number of ciphertexts to the Challenger to obtain their decryptions, subject to the restriction that the attacker cannot query the (target) ciphertext obtained in Phase 3. Upon receiving each of the ciphertexts, the Challenger decrypts it using the private key he generated in Phase 1 and returns the result (decryption) to the attacker.

- *Phase 5:* Finally, the attacker outputs his guess on which one of M_0 and M_1 was chosen by the Challenger in Phase 3.

If no attacker can guess correctly with probability significantly greater than 1/2 in Phase 5, the given public key encryption scheme is said to be secure in the "IND-CCA" case. We observe that it can be shown that the "IND-CCA" notion implies the IND-CPA notion, that is, all public key encryption scheme meeting the "IND-CCA" notion also meets the "IND-CPA" notion. Therefore, obtaining a public key encryption scheme secure in the "IND-CCA" sense means that we have already obtained an "IND-CPA" secure one. (Relationships among various notions of confidentiality of public key encryption are formally discussed in [23].)

We also remark that the chosen ciphertext attack discussed in this section is sometimes referred to in the literature as the "adaptive chosen ciphertext attack" to emphasize that the attacker is allowed to query ciphertexts to decryption oracle in an adaptive way before and after he obtains a target ciphertext. For this reason, the "IND-CCA" notion described above is sometimes referred to as "IND-CCA2" [23]. However, throughout this thesis, we adopt "chosen ciphertext attack, IND-CCA".

A final remark is that the above "IND-CCA" notion can be extended to the chosen ciphertext security notion for identity-based encryption, called "IND-ID-

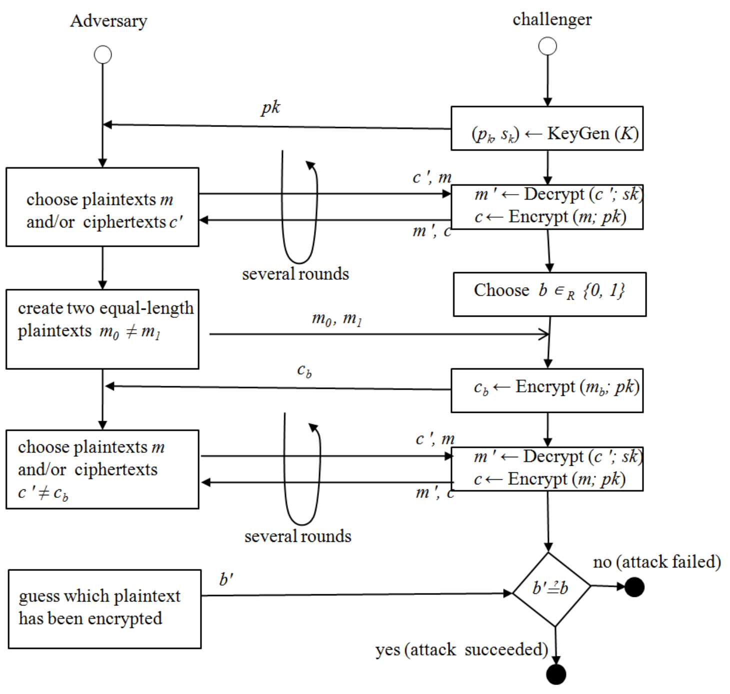

Adversary
challenger
pk
$(p_k, s_k) \leftarrow$ KeyGen (K)
choose plaintexts m and/or ciphertexts c'
c', m
$m' \leftarrow$ Decrypt $(c'; sk)$ $c \leftarrow$ Encrypt $(m; pk)$
m', c
several rounds
Choose $b \in_R \{0, 1\}$
create two equal-length plaintexts $m_0 \neq m_1$
m_0, m_1
c_b
$c_b \leftarrow$ Encrypt $(m_b; pk)$
choose plaintexts m and/or ciphertexts $c' \neq c_b$
c', m
$m' \leftarrow$ Decrypt $(c'; sk)$ $c \leftarrow$ Encrypt $(m; pk)$
m', c
several rounds
no (attack failed)
guess which plaintext has been encrypted
b'
$b' \overset{?}{=} b$
yes (attack succeeded)

Figure 2.6: Chosen Ciphertext Attack

CCA" [35]. In this notion, the attacker is only prohibited to calculate decryption queries, but also to create a number of private key extraction queries to the *PKG* to obtain private keys corresponding to some identities of his choice. An encryption scheme should remain secure under this attack to be IND-ID-CCA secure.

2.7.3 Unforgeability Notion

The evaluation of the security of a digital signature scheme for the attacker can grow "chosen message attack". For attack, the forger attacker has access to *Alice*'s signature generation oracle (algorithm) from which attacker gets signatures of any messages of his selection. At the end of the attack, he returns a new message signature pair as a forgery. The message has not been queried to the signature generation oracle before. This type of forgery is called the "existential forgery". Assume that *Alice*'s public and private keys are (N, e) and (p, q, d) such that $N =$

pq, where p and q are large primes chosen at random, $ed = 1 \; mod \; \phi(N)$ where $\phi(N) = (p-1)(q-1)$. Then, attacker queries chooses two messages $M_1 \in N$ and $M_2 \in N$ and gets signatures $\sigma_1 = M_1^d$ and $\sigma_2 = M_2^d$ from the signature generation oracle. attacker then computes $M_0 = M_1 M_2$ and $\sigma_0 = \sigma_1 \sigma_2$, and outputs (M_0, σ_0) as a forgery. Since

$$\sigma_e' = (\sigma_1 \sigma_2)e = (M_1 d M_2 d)e = M_1 M_2,$$

σ' is a valid signature for the message M'.

However, one may claim that the existential forgery is a too strong security requirement since the output message is likely to be meaningless. Since, in the current computing environment, not only meaningful messages but also arbitrary bit streams such as keys, image files, program code can be signed, so the existential forgery can cause serious damage. We remark that in order to protect the above attack on the RSA signature scheme, we need a hash function to digest the message before it is signed. In [29], it was shown that if the hash function is assumed to be a random oracle, the RSA signature scheme can be "existentially unforgeable under chosen message attack (UEF-CMA)". Now, we look into the UEF-CMA notion in the context of modern cryptography as we previously did for the IND-CCA notion. The UEF-CMA notion was first formalized by Goldwasser, Micali, and Rivest [69] and is usually described in terms of the following attack game in which the attacker and the "Challenger" interact each other:

- *Phase 1:* The Challenger generates a private/public key pair and all the necessary common parameters of a given digital signature scheme in the prescribed manner. The Challenger then gives the public key and the common parameters to the attacker while keeps the private key secret.
- *Phase 2:* The attacker queries a number of messages to the Challenger to obtain signatures on them. Upon receiving each of the messages, the Challenger generates a signature using the private key he generated in Phase 1 and returns the resulting signature to the attacker.
- *Phase 3:* The attacker outputs a new message-signature pair. (Note that the message has not been queried to the Challenge for signature generation in Phase 2.)

The signature scheme is said to be secure in the "UEF-CMA" sense if no attacker can succeed in Phase 3 with great probability. Similarly to the case of "IND-CCA", the above "UEF-CMA" can be extended to the unforgeability notion for identity-based signature, which we call "UF-IDS-CMA" [22, 25]. In this notion, the attacker is not only allowed to make signature generation queries, but also to make a number of private key extraction queries to the PKG to obtain private keys corresponding to some identities of his choice. If a signature scheme remains secure under this attack, it is said to "UF-IDS-CMA" secure.

These notions notions will be used directly or further evolved to analyze the various cryptographic schemes which will be presented in the rest of the chapters. Before moving to the next chapter, we describe about the random oracle model, which is a useful tool for designing efficient and provably secure cryptographic schemes.

2.7.4 Random Oracle Model

After formulating security notions, the next step brings to the provable security approach which shows by a reduction that the only way to break the security of a given cryptographic scheme is to solve a related computationally hard problem. However, this is not always easy unless hash functions used in the architecture of cryptographic scheme are assumed to behave as completely random functions.

The random oracle model, first introduced in [50] and popularized by Bellare and Rogaway [26] , gives a mathematical model of such ideal hash functions. In this model, a hash function $h : X \rightarrow Y$ is chosen at random from $S^{X,Y}$ which denotes the set of all functions from X to Y, and evaluation of h on inputs in X can be done only by querying the random oracle. According to Anderson [9], the random oracle can be compared with a black box in which an elf is sitting with a source of randomness and some means of storage, which are represented as a dice and a scroll respectively. The elf accepts inputs of a certain type from the outside, then looks up the scroll to see whether this query has been answered before. If so, the elf returns the corresponding answer again; otherwise, he randomly generates an answer by throwing the dice. Hence, by the assumptions made in the random oracle model, we obtain the following key property:

Assume that $h \in S^{X,Y}$ is chosen at random. Fix $x \in X$ and $y \in Y$. Then we have $Pr[h(x) = y] = 1/|Y|$.

However, a problem of the random oracle model is that the behavior of the random oracles is so ideal that no realization is possible. One can do is to replace the random oracles by the conventional hash functions such as SHA-1 [101] or MD5 [117] when the cryptographic schemes that use them are implemented. For this reason, the use of the random oracle model is somewhat controversial. Canetti, Goldreich, and Halevi were even able to demonstrate that there exist some special signature and encryption schemes which are secure in the random oracle model but become insecure whenever the random oracles are specified [40]. However, there is also a strong trend that security proofs in the random oracle at least give a certain level of security guarantees although not at the same level as those of the standard provable security approach, and more importantly, the schemes designed in the random oracle are usually very efficient [23]. Actually, it is becoming a consensus that today's standards should include the schemes with proof in the random oracle model rather than those without.

Notes: The chapter 2 is partly based on [Joonsang Baek, 2004; Stefan Rass et.al, 2014] [16, 119].

Chapter 3

Cloud Computing

There are a number of attempts to define cloud computing in numerous ways. Among definitions, the widely accepted one is " The NIST Definition of Cloud Computing" [96] as follows: Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models.

Essential Characteristics:

- *On-demand self-service.* A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service provider.

- *Broad network access.* Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, tablets, laptops, and workstations).

- *Resource pooling.* The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or datacenter). Examples of resources include storage, processing, memory, and network bandwidth.

- *Rapid elasticity.* Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be appropriated in any quantity at any time.
- *Measuring service.* Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer of the utilized service.

Service Models:

- *Software as a Service (SaaS).* The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through either a thin client interface, such as a web browser (e.g., web-based email), or a program interface. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited userspecific application configuration settings.
- *Platform as a Service (PaaS).* The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment.
- *Infrastructure as a Service (IaaS).* The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary

software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications; and possibly limited control of select networking components (e.g., host firewalls).

Deployment Models:

- *Private cloud.* The cloud infrastructure is provisioned for exclusive use by a single organization comprising multiple consumers (e.g., business units). It may be owned, managed, and operated by the organization, a third party, or some combination of them, and it may exist on or off premises.
- *Community cloud.* The cloud infrastructure is provisioned for exclusive use by a specific community of consumers from organizations that have shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be owned, managed, and operated by one or more of the organizations in the community, a third party, or some combination of them, and it may exist on or off premises.
- *Public cloud.* The cloud infrastructure is provisioned for open use by the general public. It may be owned, managed, and operated by a business, academic, or government organization, or some combination of them. It exists on the premises of the cloud provider.
- *Hybrid cloud.* The cloud infrastructure is a composition of two or more distinct cloud infrastructures (private, community, or public) that remain unique entities, but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load balancing between clouds) [96].

3.1 Cryptographic Mechanisms of the Cloud

In term of practical outsourcing of cloud data storage and processing, client's data may be sensitive and it should not be revealed to the untrusted cloud storage providers. Cryptography supports a mathematical tool for the data security of the cloud environments as traditionally has focused on providing *confidentiality* and *data integrity*. However, modern cryptography can achieve significantly more functionality which is effective solutions to specific security issues arising in the cloud environments such as the enabling of processing of encrypted data. In this chapter, we review a number of relatively cryptographic mechanisms that provide interesting functionality, all of which are potentially suitable for a wide range of the cloud environments [7].

3.1.1 Searchable Encryption

One of the basic processing tasks that might be required to be performed on encryption is to allow keyword search over the encrypted files. For cloud storage provider (CSP), the outsourcing files are encrypted and the ciphertext should not leak any information. Therefore, the usual search algorithms are as well no longer applicable. Another solution is to download the encrypted by the client and decrypt, and search at the client side. However, it is impractical for the cloud environment.

Searchable encryption (SE) schemes are encryption methods designed to address this problem which used the clients' *trapdoor information.* SE schemes vary in the expressiveness of queries and the degree of privacy offered. The server thereby neither learns the extract query nor the underlying data. The CSP can then retrieve and respond with all the data

Early SE schemes allow a single data owner to issue queries. Subsequent work [35] enables many clients to write to a database by encrypting data segments with the public key of a single user who may form searches using the corresponding private key. Other solutions allow a single data owner to grant and revoke the ability to search their files [51], or combine both properties to allow multiple readers and multiple writers. In terms of efficiency, some schemes [51] include a search phase in which the workload of the server is not linear in the number of uploaded documents but rather in the number of documents that match the query.

3.1.2 Homomorphic Encryption

By default, traditional encryption schemes do not allow meaningful combinations of ciphertexts. In any case, some encryption schemes are homomorphic in nature and allow some computations to be performed on encrypted data. Certain operations can be applied to two ciphertexts such that the result, when decrypted, outputs a plaintext as if the operation had been applied to the plaintexts themselves. For instance, let C_1 be the encryption of a message m_1 and let C_2 encrypt m_2. Then, in some homomorphic encryption schemes, multiplying C_1 and C_2 together will produce a new ciphertext C_3 which will decrypt to reveal a plaintext equal to m_1 times m_2. It should be noted that, by design, homomorphic encryption schemes are malleable (ciphertexts can be altered and remain valid).

Schemes that show homomorphic properties for a specific operation are known as partially homomorphic encryption schemes. Additionally, if the set of permissible operations enables arbitrary computations to be performed then the schemes are referred to as fully homomorphic [68]. Such schemes are very capable since they permit arbitrary computation on encrypted data, and thus fit the cloud setting particularly well as the untrusted CSP does not require access to the plaintext data. Unfortunately, current schemes tend to be limited in the number of operations that may be applied before decryption will not succeed, or are inefficient in terms of speed and the size of parameters and ciphertexts.

3.1.3 Computing Aggregates over Encrypted Data

There is an example of specific application which is privacy preserving data aggregation, which allows specific types of computation to be performed on encrypted data. The data is assumed to come from multiple independent sources, which are reluctant to share their sensitive information with either other sources or the aggregator. This has led to an active area of research where proposed solutions mainly rely on homomorphic encryption and secret sharing techniques. Some schemes achieve aggregator obliviousness using a trusted dealer that provides the aggregator with the sum of users' secret keys, which in turn allows the decryption of the sum of users' data. Other schemes handle dynamic user populations and arbitrary user failures. Recently, Leontiadis et al. [85] removed the need for trusted key dealers while supporting dynamic group management and user failures.

3.2 Functional Encryption

Functional encryption develops traditional public-key encryption to permit the holder of a private key to get a specific function of an encrypted message. This function could restore the message itself (by the traditional public key encryption), storing the message only if some other criteria is met, or may create the output of some computation specified by the message and private key. In the cloud computing, functional encryption can be set up as a cryptographic enforcement mechanism for access control policies. Functional encryption allows the encryptor (a client) to indicate an access control policy in terms of identities or more general descriptive attributes; decryptors may get the data that they fulfill this arrangement policy. Accordingly, data owners hold control of which entities may learn their data without using explicit prior knowledge of users.

There are a sort of functional encryption scheme that could be useful in cloud environments.

3.2.1 Order Preserving Encryption

Order preserving encryption [3, 34, 132] allows a CSP to perform series of queries on encrypted data in order to get relevant results to a client query. This is a type of deterministic symmetric encryption where numerical comparison operators can be used to encrypted numerical data. It has common applications to querying encrypted databases. The scheme of Boldyreva et al. [34] claims to achieve such numerical range searches in logarithmic time (in the size of the database).

3.2.2 Identity-based encryption

Identity-based Encryption (IBE) [37] permits encryptors to specify an arbitrary identity string (user name, email address, IP address etc.) while preparing a ciphertext, as opposed to use a pre-defined public key. A decryptor can request (either earlier or subsequently) a decryption key related to an identity from a key generation authority. The plaintext is successfully recovered if the identity associated with the ciphertext and the key. Since identity strings can be arbitrary, it is possible to append the current day, for example, in order to specify a lifetime for a decryption key. In addition, one could append access rights or different separations of duty, etc.

3.2.3 Attributed-based Encryption

Attribute-based Encryption (ABE) is suitable when the authorized set of decryptors cannot easily be stated explicitly in terms of identifier strings (for instance, the user population is very large or changes frequently). Rather, authorized decryptors can be defined in terms of attributes. ABE comes in several variants that vary based on the type of the key and ciphertexts. In Key-Policy ABE (KP-ABE) [72], the ciphertext contains a set of attributes that describe the classification and contents of the plaintext, although the decryption key is related to an access structure (which describes the access policy). Decryption succeeds if the set of attributes satisfies the access structure. Hence, a user can be issued a key for a formula specifying their access rights, while ciphertexts can be related a set of attributes describing its contents or required level of protection (e.g. YearlyReport, Accounts, ClearanceLevel2). Moreover, Ciphertext-Policy ABE (CP-ABE) [131] reverses the association of attribute sets and access structures. Ciphertexts are now formed with an associated formula over attributes during decryption keys are issued for an attribute set.

3.2.4 Predicate Encryption

Predicate Encryption (PE) [82] uses the previous notions of functional encryption, particularly KP-ABE. Decryption keys are related a predicate F over attributes and ciphertexts are related to a set of attributes I. Decryption will succeed if $F(I) = 1$. Therefore, if $F(I) = 0$ then no information is got about the encrypted message; this property is referred to as payload hiding. Furthermore, some schemes can accomplish a stronger notion of attribute hiding by which, also hiding the message, no information is learnt about the attribute set I beyond what is normally leaked by the decryption functionality that is the result of $F(I)$. Many PE schemes aim to the specific predicates of inner products, which have been appeared to encompass useful functionality such as boolean formulas in conjunctive normal form and disjunctive normal form, threshold policies and polynomial evaluation.

3.3 Verifiable Computing

It is commonly recommended that CSPs should be "honest-but-curious", by implying that they are trusted to follow the principle of any process, however, be fully trusted with respect to privacy of data that they happen to observe. But it is not necessarily always the case that such a level of trust can be placed in a CSP. Several new cryptographic mechanisms provide benefits that may be suitable in the cloud environments with reduced levels of trust in CSPs.

3.3.1 Verifiable Outsourced Computation

One concern emerges in the cloud environment which CSP is not trusted to get the correct result of a processing computation. In verifiable outsourced computation (VC), a client delegates the execution of computationally demanding operations to the cloud and receives the results alongside a cryptographic proof of their integrity or privacy. These proofs permit the detection of any server misbehavior and, in the meantime, do not allow a client dishonestly accuse a server of misbehaving.

Features of certain verifiable computation schemes include public verifiability, which guarantees that anyone can verify the correct processing of outsourced operations using only public information. Most VC schemes use non-interactive proofs, which restrain the level of interaction between provers and verifiers. There are many different techniques used to build VC schemes, including fully homomorphic encryption [44, 17] and KP-ABE [107]. Pinocchio [109] applies succinct non-interactive arguments of knowledge (SNARKs) to achieve the public verifiable computation.

3.3.2 Verifiable Storage

Another concern arises when CSPs are not trusted to protect the integrity of outsourced data in their charge. A basic solution is for the client to compute and keep a checksum (such as a MAC) of the data. But this kind of verification ascends poorly in the cloud environments where large amounts of data are stored. Verifiable storage schemes aim to make verification more efficient rather than downloading the whole data set, and allow clients to do integrity checks as many times as needed. Solutions commonly fall into two categories:

i deterministic solutions that offer an undeniable guarantee of integrity, and

ii probabilistic solutions in which the verifier is convinced of the integrity of the data with a certain probability only.

Deterministic solutions obtain considerable computation and communication complexity, commonly linear in the size of the entire data. Most schemes introduce probabilistic optimizations based on random sampling; for example checking the entire file, some checks the integrity of a subset of segments included in the file. Probabilistic solutions can generally be classify as either Provable Data Possession (PDP) (verifying that the data is belonged to the server) [13] or Proofs of Retrievability (POR) (verifying that data is recoverable from the server, even if a few modifications have been made by a malicious server) [78]. [13] proposed static data in the context of archival or back-up storage, when others allow for efficient editions of the data (modification, deletion or insertion of blocks) and for efficient integrity verification to ensure that the server stores the latest version of the outsourced data. Moreover, some solutions let to verify to be delegated to a third-party (auditor) [12] and to render this public verification privacy-preserving [133].

3.4 Other Tools

In previous sections, we discussed the three classes of cryptographic mechanism. Here, there still have several other relatively recent cryptographic tools that have the potential for deployment in the cloud environments.

3.4.1 Format Preserving Encryption

Format Preserving Encryption (FPE) [24] enables formatted data to be encrypted to ciphertexts that follow to the same formatting rules (e.g. credit card numbers are encrypted to the random, valid credit card numbers). The encryption induces a pseudo-random generation over all validly formatted strings. This property can be used for storing encrypted databases where data fields must have specific formatting rules. Specifically, it is useful when upgrading heritage outsourced database solutions to be secure. Generally, it is not possible to simply encryption utilizing a non-format preserving encryption scheme without changing the structure of the database itself. One approach to achieve FPE is the “rank-then-encipher” approach where the set of all valid strings are numbered according to some ranking function. After that, using a simple integer FPE scheme (which encrypts integers to integers), one can encrypt the rank of the message. The output ciphertext will be an integer that indexes some random (correctly formatted) message from the message space, which forms the final ciphertext. Such ranking functions exist for all formatted domains where the format can be expressed by a regular language.

3.4.2 Proxy Re-encryption

Proxy Re-encryption is a relatively newly-devised cryptographic primitive which allow a seimi-trusted proxy to transform a ciphertext encrypted under one key into an encryption of the same plaintext under another key, without revealing the underlying the plaintext, proxy re-encryption [13] allows a semi-trusted intermediary (proxy) to convert a ciphertext intended to be read by one entity into one that can be read by another, without the proxy decrypting the ciphertext, or otherwise learning the message itself.

We also proposed non transferable proxy re-encryption [49] which the re-encryption key is generated by a key generator; delegator help to generate the partial decryption key for delegatee using the part of private key. Therefore delegatee and proxy cannot collude to re-delegate decryption rights since they do not have knowledge of PKG key or delegator’s private key.

One example application is to manage access to encrypted data stored on a cloud server, which acts as the proxy. The stored ciphertexts can be transformed such that they can be decrypted by authorized entities, yet the server itself remains unable to read the data.

3.4.3 Secure Deduplication

Secure Deduplication [20] provides a space efficient storage solution for outsourced data of the cloud service. If two users demand to keep the same data file, a cloud server may want to save storage costs by only storing single copy of the file for both users. However, there has difficultly if data is encrypted before submitting to the outsourced cloud, as the security properties of a encryption scheme will output the ciphertexts for both files resulting entirely different . Furthermore, storing just one ciphertext will prevent other users from recovering the data without holding the same decryption key (symmetric encryption). One of the solution for this problem is to use message-locked encryption (MLE) [20] which is relatively new encryption scheme. MLE encrypts the message by using a key derived from the original message. Such as, the key could be defined to be the output of a hash function applied to the message. A tag is generated that the server will use to detect duplicates faking attack. Privacy holds only when the message space has sufficient min-entropy. Another important security property for deduplication is tag consistency, which requires that it is hard to force an honest client to recover a message different from that which it uploaded.

In this thesis, data deduplicaion is main concerned for the secure cloud computing and we describe the detail explanation of secure deduplication from Chapter 4.

Notes: The chapter 3 is based on [James Alderman et.al, 2015] [7].

Chapter 4

Cloud Data Deduplication

We overview the basic concept of cloud deduplication, type of deduplication and some research works on data deduplication which are relevent to the scope of this thesis.

4.1 Cloud Storage and Issues

Cloud storage is a service where data is remotely maintained, managed, and backed up. The service is available to users over a network which allows the user to store files remotely and the user can access them from any location via the Internet. The cloud service provider (CSP) company provides them available to the user online by keeping the uploaded files on an external storage devices. This gives companies using cloud storage services ease and convenience, but can potentially be costly after a long period. Users should also be aware that backing up their data is still required when using cloud storage services because recovering data from cloud storage is much slower than local backup [61].

The use of Internet and other digital services have been grown to a digital data explosion, including the cloud storages. A survey [60] revealed that only 25% of the data in data warehouses are unique in data warehouses. In the presence of this huge problem of big data, it would be beneficial in terms of storage savings if the replicated data could be removed from the data storages. According to the survey in [39], only 25 GB of the total data for each individual user are unique and the remaining ones are similar shared data among various users. On the enterprise level [1], it was reported that businesses hold an average of three to five copies of files, with 15 % to 25 % of these organizations having more than 10 copies of the files. Keeping these facts in view, cloud service providers have massively adopted data deduplication, a technique that allows the cloud service provider to save some storage space by storing only a single copy of previously duplicated data.

A major issue hindering the acceptance of cloud storage services by users is the data privacy issue associated with the cloud paradigm. Indeed, although data is outsourced in its encrypted form, there is no guarantee of data privacy when an honest but curious cloud service provider handles the management of confidential data while these data reside in the cloud. This problem is even more challenging when data deduplication is performed by the cloud service provider as a way to achieve cost savings. Data deduplication is being adopted by many popular cloud storage vendors like Dropbox [57], Spider Oak [126] and Mozy [98]. Recently, Dropbox [58] was reported to have stolen 70 millions of passwords of some users, but they shifted the blame to because a Dropbox employee re-used a password they had used on another site. However, the users cannot compromise the security of their data, and then the attack was perpetrated either by an anonymous external attacker or it was an internal fault attributed to the cloud service provider. According to the above discussion, it is a serious demand of current big data and cloud computing paradigm to handle to issue of data growth and security. Data deduplication is an effective technique exercised to handled the issue. The primary condition is that the data deduplication should be designed in accordance to the security and efficiency requirements of the system in consideration.

4.2 Deduplication

Data deduplication is a data compression technique to eliminate the copies of the identical data, which improves storage utilization cloud as shown in Figure. 4.1. For example, Dropbox [57], Google Drive [71], Mozy [98] and so on have adopted data deduplication to save storage and network data transfer [20, 75, 114, 142, 3]. Normally, the cloud users encrypt data with their own keys to ensure their data privacy and data confidentiality. Therefore, deduplication cannot happen, especially, if the same encrypted data is stored by different owners with different keys, the result will different. It means the typical encryption schemes do not support deduplication [138, 125, 143, 47]. Meanwhile, designing an efficient and secure data sharing scheme for groups in the cloud is not an easy task and pose

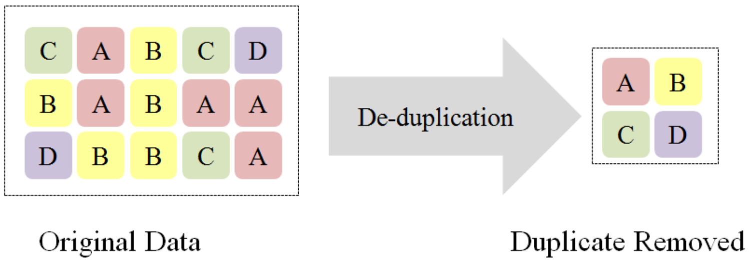


Figure 4.1: Deduplication

many challenging security issues.

4.2.1 Client-Side Deduplication

In a client-side deduplication [61, 82], the client could use a hash function to the block or file and sends the ciphertext to the server side. The server receives and checks, and if it already has a copy of this data, then the data is only sent if the server does not possess a copy of the file as shown in Figure. 4.2.

4.2.2 Server-Side Deduplication

When deduplication can be carried out at the server or directly from the client, it is called the server-side deduplication [61, 82]. The client sends the data to the server executes the deduplication algorithms is execute at the server side as shown in Figure. 4.3.

4.2.3 Level of Data Deduplication

Many kinds of deduplication algorithms are performed at different levels as file-level deduplication and block-level deduplication. Generally, file-level deduplication does three tasks:

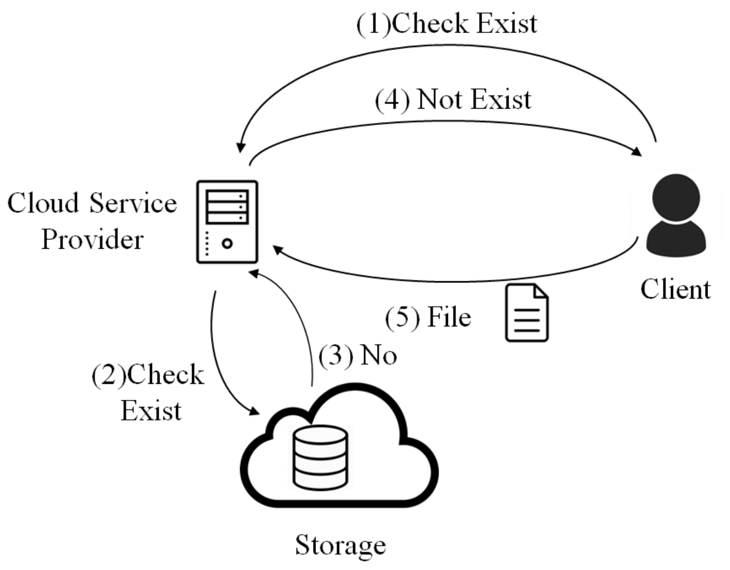


Figure 4.2: Client Side Deduplication

1. checks for redundant copies of the same file,
2. stores the first copy, and then
3. index the other references to the first file. At the other side, when the deduplication is at a block level, the files are separated into data blocks and the same verification for each block formed with the files is carried out.

File-level deduplication will reduce a small amount of space on the storage. Block-level deduplication will save more. Block-level deduplication will execute all of the duplicates, even if named differently, and will store the name variations with pointers to the original blocks [61, 82].

4.3 Data Deduplication Techinical Design Issues

Following is an compact explanation of some of the main technical design considerations of data deduplication.

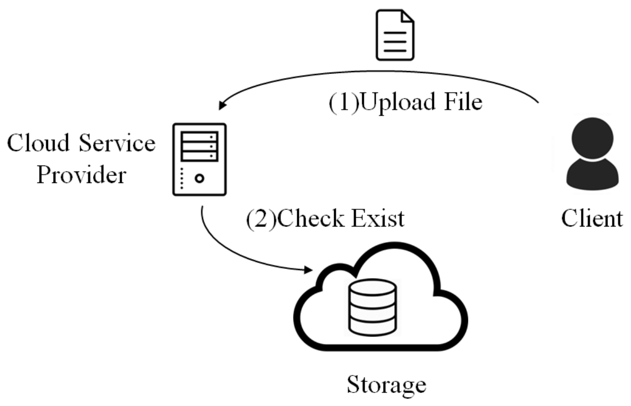


Figure 4.3: Server Side Deduplication

4.3.1 Types of Data Deduplication

Hash Based: Hash based data deduplication methods use a hashing algorithm to identify the chunks of data after the data chunking process.

Content Aware: Content aware data deduplication methods rely on the structure or common patterns of data used by the applications. Content aware technologies are also called byte level deduplication or delta-differencing deduplication. Key element of the content-aware approach is that it uses a higher level of abstraction when analyzing the data [35]. The deduplication server sees the actual objects (files, database objects etc.) and divides data into larger segments of sizes from 8 MB to 100 MB. Since it has the knowledge of the content of the data, it finds segments that are similar and stores only the changed bytes between the objects. That is why it is called a byte level comparison.

Hyper Factor: Hyper factor is a patent pending data deduplication technology that is included in the IBM System Storage Protec TIER Enterprise Edition V2.1 software [142]. According to [141], a new data stream is sent to the ProtecTIER server, where it is first received and analyzed by Hyper factor. For each data element in the new data stream, HyperFactor searches the Memory Resident Index in

ProtecTIER to locate the data in the repository that is most similar to the data element. The similar data from the repository is read. A binary differential between the new data element and the data from the repository is performed, resulting in the delta difference which is stored with corresponding pointers. Another aspect to be considered when dealing any deduplication system is the level of deduplication. Data deduplication can take place at two levels file or block level. In block level deduplication, a file is first broken down into blocks (often called chunks) and then the blocks are compared with other blocks to find duplicates. In some systems, only complete files are compared, which is called Single Instance Storage (SIS). This is not as efficient as block level deduplication as entire files have to be stored again as a result of any minor modification to that file.

4.3.2 Client vs Server Side Deduplication

Client deduplication refers to the comparison of data objects at the source before they are sent to a destination (usually a data backup destination). A benefit of client deduplication is that less data is required to be transmitted and stored at the destination point. A disadvantage is that the deduplication catalog and indexing components are dispersed over the network so that deduplication potentially becomes more difficult to administer. If the main objective is to reduce the amount of network traffic when copying the files, client deduplication is the only feasible option. On the other hand, server side deduplication refers to the comparison of data objects after they arrive at the server/destination point. A benefit of server deduplication is that all the deduplication management components are centralized. A disadvantage is that the whole data object must be transmitted over the network before deduplication occurs. If the goal is to simplify the management of the deduplication process server side deduplication is preferred. Among many popular vendors such as DropBox [57], SpiderOak, Microsoft Sky Drive, Amazon S3 [8], Apple iCloud and Google Drive, only SpiderOak allows to perform server side deduplication [3].

4.3.3 Single User vs Cross User Deduplication

The deduplication can be done on a single user side, where the redundancy among his/her data is identified and removed, but the single user data deduplication is not very practical and does not yield maximum space saving. To maximize the benefits of data deduplication, cross user deduplication is used in practice. This technique consists in identifying the redundant data among different users and then removes the redundancy and save a single copy of the data. According to [2], 60 percent of the data can be deduplicated on average for individual users by using cross user deduplication techniques. In order to save bandwidth, cloud storage provider and users of their services are inclined to apply client-side deduplication where similar data is identified at the client side before being transmitted to the cloud storage. However, the potential benefit of data deduplication in terms of storage space and storage cost can also have some associated drawbacks. In 2012, Drop Box disabled the client side cross user deduplication due to some security concerns [3].

4.4 Background

While providing data confidentiality and security, the traditional encryption is incompatible with data deduplication. Specially, it needs different users to encrypt their data with their own keys as in Figure. 4.4. Thus several copies of identical original plaintext (with their owned keys) will lead to different ciphertexts, and deduplication process will never happen. Public key encryption, by comparison has two different keys, one used to encrypt the string (the public key) and one used to decrypt it (the private key). Each file use common hash function (Sha1, Sha256 and etc.) to store the index of all the hashes already in the system. For example, if there is a new file, then compute hash and look hash up in the index table. If the file is a new one, then this file will add to the index. Else the file known hash, only the hash values is saved as a pointer to existing data.

4.4.1 Convergent or Message-locked Encryption

Encrypting files before uploading them to a storage service allows us to guarantee confidentially with respect to insider attacks of the storage provider (SP) or any outsider that breaks into the system, since neither the SP nor the outsider gets to know the respective decryption keys (they are managed by the clients). We note that today several cloud SPs like Amazon S3 provide means to server side encryption. However this can at most provide confidentiality with respect to external attackers (assuming the decryption keys are appropriately secured at the storage provider), but not against insider attacks, because the SP has access to

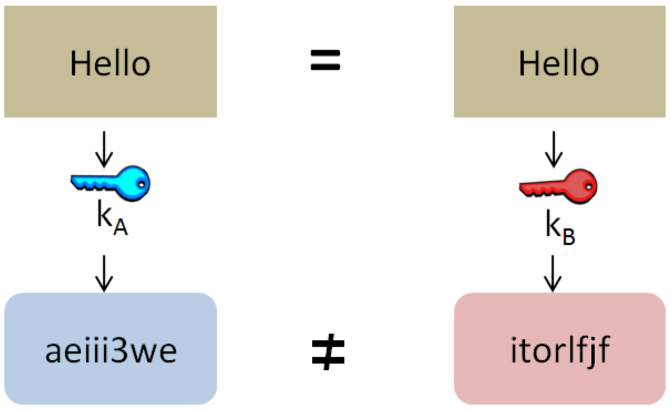


Figure 4.4: Syntax of Traditional Encryption

the outsourced data in plaintext, as all decryption keys are available to the SP.

In [48], the authors discussed whether it is possible to use client-side encryption with the following feature: if the same file is encrypted by two different clients who do not share any internal state nor any secret key, can the resulting ciphertext always be identical. Therefore, they introduced what they call a convergent encryption scheme, which produces identical ciphertext (files) from identical plaintext (files), irrespective of their encryption keys and should provide reasonable confidentiality guarantees with respect to insiders (i.e., the storage server in our scenario).

The basic idea behind their construction is very simple and shares some similarities with turning probabilistic into deterministic public key encryption schemes. Therefore, let *(KeyGen, Encrypt, Decrypt)* be a secure symmetric encryption scheme and let H be a secure hash function. The idea is not use $KeyGen$ to generate a random key, but to compute a key k as $k = H(F)$ where F is the file to be stored. Consequently, the ciphertext F' (the encrypted file) is computed as $F' = Encrypt(F; H(F))$. Note that everybody having the file F can compute $k = H(F)$, and thus produce the deterministic ciphertext F', which allows deduplication at the server without compromising the confidentiality. Note that the server does not know F, and thus the key k required to decrypt F'.

Although convergent encryption has been around for some years and is already used in deployed applications and proposed for the use in deduplication, a formal study for what the authors call message-locked encryption has only been performed very recently. This work, from a practical side, essentially shows that convergent encryption, as introduced above in the context of deduplication, is secure.

This construction works out of the box when using server-side deduplication. However, care must be taken when using convergent encryption with client- side deduplication. Basically, just sending the hash value $H(k) = H(H(F))$ of the F' for checking, then some of the attacks discussed below are (PoW) protocols.

Message-Locked Encryption

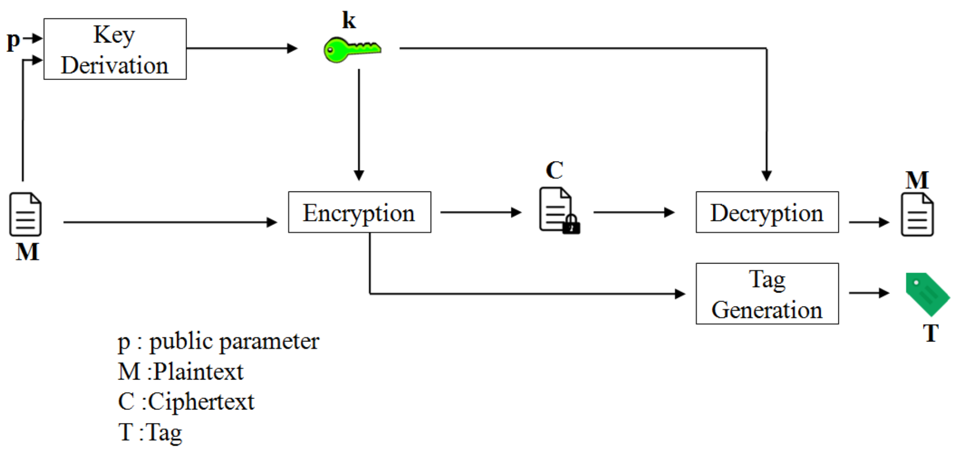


Figure 4.5: Syntax of Message-Locked Encryption

Message-Locked Encryption (MLE) is introduced by Bellare et al. [20] as a theoretical treatment of secure deduplication based on Convergent Encryption (CE) [112, 113]. An MLE scheme (illustrate in Figure. 4.5), encapsulates a standard symmetric-key encryption scheme where encryption algorithm accepts a message M and a key k and outputs a ciphertext C. Decryption algorithm recovers M

from C given k. It also includes a tag generation algorithm that is a way to test whether the plaintexts underlying two ciphertexts are the same or not. See [20, 2] for details.

Interactive Message-Locked Encryption

Interactive message-locked encryption (iMLE) is proposed by Bellare et al. [21] for the interactive properties. iMLE enables incremental updates and is established by an original MLE scheme permitting incremental updates, the interaction being used only for tag consistency. The iMLE consists of an initialization Init algorithm and three protocols for Reg (Register), Put (Upload), and Get (Download). However, iMLE [21] is purely theoretical, we have to adopt in real scenarios.

4.4.2 Proofs of Ownership

When applying client side deduplication with the approach discussed in the introduction; that is by simply sending the hash value of the file for the check, two main classes of potential attacks for which we provide some illustrating examples:

Privacy and Confidentiality: Fraudulent users can check whether someone has stored a specific file. For example, think of some privacy intrusive file or some whistleblowing activity. Furthermore, if a file is highly structured and only a small space of varying content is left, someone can simply brute force all potential files by computing their hash values and check whether the respective file has been stored-think of contract backups. Moreover, if someone just learns the hash values of files (e.g., reports containing sensitive information), then this person can illegally get access to the file.

Misuse of storage service: Fraudulent users may use a storage services as a low bandwidth convert channel. For example, by choosing two files, and assigning them bit values, a bit can be communicated via the information whether or not a file is already present at the SP. Another misuse is by uploading a file to the storage service and then distributing its hash value such that other users can download such files. This would actually let the SP unknowingly act as content distribution network.

Note that in all above examples, access control does not prevent any of these attacks, since deduplication is performed over all users irrespective of access rights to particular files. The problem yeilding to attacks is that short information from the file is sufficient to download the file. Essentially, the solution to the problem is to force users to prove that they know much more information about the file at hand and not only its hash value. Thus, such a solution is called "*Proof of ownership (PoW)*" and more sophisticated proposals than the threshold solution which we briefly sketch below. The security model requires that a client succeeds only with negligible probability in a *PoW*, as long as the client has enough uncertainty about the file (i.e., an attacker is allowed to have partial information about the file). The security guarantees in the subsequent constructions are increasingly relaxed for the sake of efficiency.

- The first construction takes a file, uses an erasure code to produce an encoded file, then computes a Merkle tree over the encoded file. The server stores the root hash of the Merkle tree over the encoded file. The server stores the root hash of the Merkle tree along with the file. A *PoW* then amounts to challenge the client to send the authentication paths. (i.e., siblings of a leaf on the path to the root) for randomly sampled (superlogarithmic) number of leaf indices.

- The second construction avoids such costly and disk access intensive erasure coding. Instead, this solution uses a universal hash function to hash the file into a so-called reduction buffer, which is much smaller than an encoded file, but still too large to share it with other users, say, 64 MB. Then the Merkle tree approach is implemented on this reduction buffer instead of the encoded file.

- The third schemes replaces universal hashing, which costly to compute for such a large output space, with a more efficient mixing and reduction procedure. Essentially, the mixing phase builds the reduction buffer by XORing each block of the original file with four random positions in the buffer (after performing a bit shift). The mixing phase then amplifies confusion and diffusion in the reduction buffer by XORing together random positions of the reduction buffer. Finally, the reduction buffer is used as a basic for the Merkle tree approach.The problem will all the above mentioned contructions is that the client needs to process the entire file for conductiong a PoW.

Chapter 5

Multiple Group Signature Setting

In this chapter, a secure scheme is proposed that achieves multiple group of client for deduplication in cloud storage environment. Its design consists of message-locked encryption along with group signature to control the multiple group of users. The message-locked encryption is used to ensure that the proposed scheme is secured against a semi honest cloud service providers and the group signature is used to enable a classification of group. In such way, the deduplication cross the groups can be performed securely. The proposed scheme is called DDUP-MUG which supports deduplication not only uploading a new file but also updating the existing files and deleting files among the different groups. Security analysis are provided, showing the effectiveness of multiple group deduplication and security of our proposed scheme.

5.1 Background

The motivation behind the scheme is to allow cross-groups which are controlled by each individual group manager. In this scenario, not only multiple users but also multiple groups may own a single message which is located in the same data storage. The ownership of any client (group member) is verified for the UPL-Dup protocol and the EDT-Dup protocol according to what he/she demands whether he/she wants to upload a new file or update an existing file. Finally, a group member can eliminate his own file according to the revocation list by using the DEL-Dup (Delete a duplicated file) protocol. Our three protocols use a deterministic encryption method called Message-Lock Encryption (MLE) or interactive

Message Lock Encryption (iMLE) as an ingredient and also we add unlinkable randomizable group signature (URS) [148] features to manage the individual group. We slightly improve URS group signature with revocable functionality. We combine the above three protocols as a new scheme, Deduplication for Multiple Group Signature (DDUP-MUG). One common fact of three protocols, the client has to send only a pair of tag and ownership. The duplication checking is performed at the server side. If the file is duplicate, the client does not need to upload the ciphertext.

5.1.1 Group Signature

Group Signature schemes, originally defined by Chaum and van Heyst [14], are an important building block for our deduplication method in the multiple-group setting. According to the general group signature (GS) scheme, GS consists of one signer, one verifier and the several members. Group signature schemes allow any member of a group to sign a file on behalf of the group. In general, a group manager controls the group membership and issues group signing keys to group members. The group members sign documents by using the group signature keys. Furthermore, a group signature provides anonymity and unlinkability to the signer, i.e. anybody can verify that the signature is valid which is signed on behalf of the group, but nobody except the group manager can identify the signing member. Additionally, a verifier can check whether two different valid signatures were generated by the same group or not [3, 28, 31, 148].

Role of the Individual Group Manager It is usually accepted that the group manager is a trusted party with respect to joining new entities to the group. Many group signature schemes involve an issuer who is joining new members and many openers who are responsible for opening signature.

Multiple Group Signature According to secure cloud deduplication, the server will store one unique instance of redundant data. Therefore, clients from different groups have to prove that they simultaneously own a stored file. Though multiple group settings are based on the ability to link some designated group signatures via equality proofs, such a fact does not affect the main properties of group signatures such as anonymity and unlinkability. Note that the multiple group setting is very interesting in anonymous authorization scenarios, since in these environments it is quite common for file properties in order to give the authorization to carry out some specific transaction.

Correctness for the Group Signature scheme

A group signature scheme should satisfy the following correctness properties.

Verification Correctness Group signature produced by group members can be verified successfully while manager of the group ensures the signer identification [3, 28, 31, 148].

Definition 1 *A group signature scheme $\mathcal{GS}$ = (KeyGen, Sign, Ver) is correct if all λ which is the security parameter, $N \in \mathbb{N}$ which is the number of entities, all keys (gpk, gmsk, gsk) $\leftarrow$ $\boldsymbol{KeyGen}(1^\lambda, N)$, all identities $i \in [1, N]$ and all messages $\boldsymbol{m} \in \{0,1\}^*$,*

$$Ver(gpk,\ m,\ Sign(gsk[i],m)) = 1.$$

Revocation List Correctness In the fully dynamic group signature scheme, group managers allow prospective members into the group through the Join protocol as well as admit verifier with secret key *gsk*, group public key *gpk* and revocation token *grt* which is used to issue membership credentials [35].

Definition 2 *A group signature scheme with verifier-local revocation $\mathcal{GS}$= (KeyGen, Sign, Ver) is correct if all (gpk, gsk, grt) $\leftarrow$ $\boldsymbol{KeyGen}(1^\lambda, N)$, all identities $i \in [1, N]$ and all messages $\boldsymbol{m} \in \{0,1\}^*$*

$$Ver(gpk, \boldsymbol{RL}_t, m, Sign(gsk[i], m)) = 1 \Leftrightarrow grt[i] \notin \boldsymbol{RL}_t.$$

Actually group signatures with above characteristics have been proposed and adopted explicitly or implicitly by [3, 28, 31, 35, 148].

5.1.2 Group Signature from Unlikable Randomizable Signature

Definition 3 *A group signature consists of the following algorithms for a group manager GM, group members and verifiers [148].*

- *Setup*: an algorithm is initiated by GM to generate group public key *gpk* and a group secrete key *gsk*.
- *Join*: a probabilistic interactive protocol between GM and a group member. If a new group member is added, GM gives a group signing key gpk_i including the member secret key msk_i and the member certificate $cert_i$; and GM adds an entry for i (denoted as reg_i) in its registration table reg storing the protocol transcript, e.g. $cert_i$.
- *GSig*: a probabilistic algorithm by a group member, on input a message m and a group signing key gpk_i= (msk_i, $cert_i$), returns a group signature σ.
- *GVer*: a deterministic algorithm which, on input a message-signature pair (m, σ) and GM's public key *gpk*, returns 1 or 0 indicating the group signature is valid or invalid respectively.
- *Open*: a deterministic algorithm which, on input a message-signature pair (m, σ), the secret key *gsk* of GM, and the registration table *reg*, returns the identity of the group member who signed the signature, and a proof π.

The group signature is σ by using group singing key gpk_i where $s = GSig($ msk_i, $cert_i)$and proof of knowledgefor σ satisfying $v = GVer = 1$. ($GSig$, $GVer$) is the signature generation and verification algorithms of an independent signature scheme.

5.2 Management Policy of Proposed Scheme

Our system aims at secure client-side deduplication for the encrypted messages. Message-Locked Encryption (MLE) is introduced by Bellare et al. [20] as a theoretical treatment of secure deduplication based on Convergent Encryption (CE). An MLE scheme encapsulates a standard symmetric-key encryption scheme where

CipherID	TagID	Ownership		
C_1	T_1	σ_a	σ_b	σ_c
C_2	T_2	σ_a	σ_b	

Table 5.1: Data Structure of a file at Server

encryption algorithm accepts a message *M* and a key *k* and outputs a ciphter-text *C*. Decryption algorithm recovers *M* from *C* given *k*. It also includes a tag generation algorithm that is a way to test whether the plaintexts underlying two ciphertexts are the same or not [2] . We also use interactive message-locked encryption (iMLE) Bellare et al. [21] for the interactive properties. iMLE enables incremental updates and is established by an original MLE scheme permitting incremental updates, the interaction being used only for tag consistency. However, we will not generalize some functions of initialization, registration or download for new users from interactive protocols. Four algorithms *(K, Enc, Dec, GTag)* are contained in the MLE scheme. A message derived key *k* is produced from an algorithm, $k \leftarrow_{\$}$ *K(p, M)*, where *p* is a system-wide public parameter, and we can get the ciphertext by $C \leftarrow_{\$}$ *Enc(k, M)*. Decryption recovers *M* by using $M \leftarrow$ *Dec*(k_0 ,*C)* and the key is also derived from message *M* as $k_0 \leftarrow_{\$}$ *K(p, M)*. Tags which appear from $t \leftarrow$ *GTag(C)* are used to check whether the plaintexts underlying two ciphertexts are the same or not. Every encryption for the same message *M* will have the same tag. There are many MLE and iMLE schemes, and any MLE or iMLE can perform deduplication [20, 21, 2]

5.3 Proposed Scheme Architecture

Our system involves four entities: the agency or group manager, the storage server, the verifier, and the clients (group members) as illustrated in Figure. 5.1. Their responsibilities are described below:

Group Manager(GM) The manager controls the group memberships and generates the membership keys of group members (clients). Furthermore, the GM

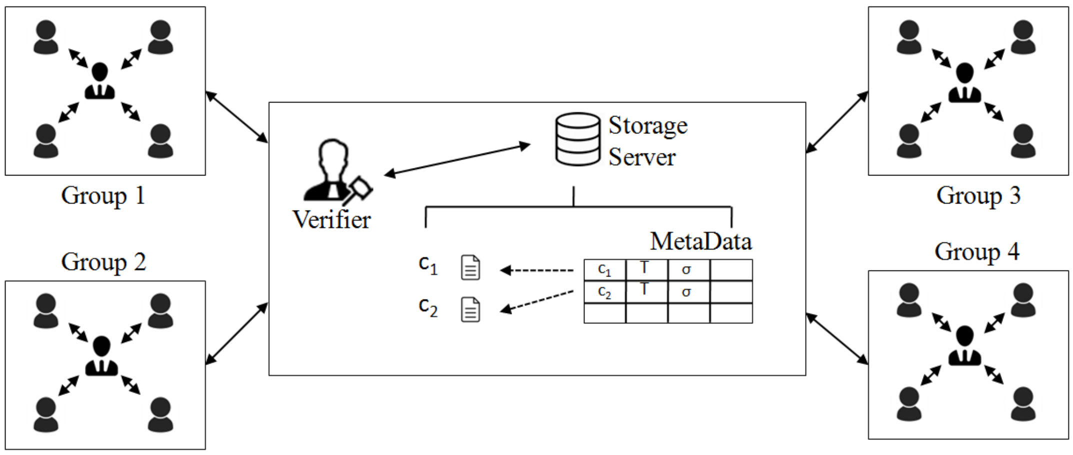


Figure 5.1: Proposed DDUP-MUG Scheme

enables signers to sign on behalf of the group and reveals the identity of the signature's originator when disputes arise. The GM issues the keys and the revocation list for both the client and the storage server. And then the GM also provides the computational services to facilitate key management.

Verifier A verifier can check the validity of the group signature by using the group secret master key and the revocation list.

Storage Server The server provides data outsourcing for all groups. To reduce the cost of storage, the server removes the redundant data of identical files via deduplication and keeps only a unique copy of the data.

Client or Group Member A member is an entity of a group who needs to store/restore/remove data from the server. No client will be allowed to upload data which is a duplicate of data on the server, i.e., the deduplication policy can reduce not only the storage size but also the upload bandwidth.

Now, we present a secure deduplication framework (DDUP-MUG) starting by describing a design concept to address the threats and then elaborate on the detailed construction. The main idea is that for avoiding duplication at storage server. The proposed protocol is composed of four operations: setup, UPL-Dup, EDT-dup and DEL-Dup. Firstly GM and client initiate the setup protocol.

5.3.1 System Setup Protocol

Before the client uploads a message to the storage server, the client, the GM and the verifier must be the members of a group which associates to the storage server. Therefore, Join protocol will interact between client-GM and GM-verifier. We follow the unlinkable randomizable group signature scheme (URS) introduced by Zhou et al. [29]. And then we modify some parts such as removing open and judge and adding a revoke algorithm in our scheme. The detailed setup of the proposed system includes system initialization *(KeyGen)*, user registration *(Join)* and user revocation *(Revoke)* as shown in below.

1. *KeyGen* is a probabilistic key generation algorithm for the group manager, on input 1^{λ} which is the security parameter, and N which is the maximum number of members. Generate the randomness $R \leftarrow_{\$} \{0,1\}^{P(\lambda)}$, $R_1 \leftarrow_{\$} \{0,1\}^{P_1(\lambda)}$ and $R_2 \leftarrow_{\$} \{0,1\}^{P_2(\lambda)}$ where P, P_1, P_2 are polynomials. Compute $(pk_s, sk_s) \leftarrow K_s\left(1^{\lambda}\right)$ and $pk_e \leftarrow K_e\left(1^{\lambda}\right)$ where K_s and K_e are polynomial time algorithms, respectively for signature generation keys and signature encryption keys . Then assign the group public key $gpk = (R, R_1, R_2, pk_e, pk_s)$ and the group manager secret key $gmk = (sk_s)$.

2. *Join* is an interactive protocol between a member and the group manager. $\boldsymbol{Client_i}$ selects member secrete key sk_i and caculate pk_i by a one way function f, $pk_i \leftarrow f(sk_i)$, Generate a non interactive zero knowledge proof π of sk_i as $\pi = P(pk_i, sk_i, R)$ and sends pk_i, π to group manager. If member key verification algorithm $KeyVer(pk_i, \pi, R)$ returns 1, GM generates $cert_i = Sig_f(sk_s, pk_i)$ and sets $reg[i] = pk_i$. Then, $\boldsymbol{Client_i}$ sets $gsk_i = (pk_i, sk_i, cert_i)$. Finally, the secret key gsk_i and *reg[i]* respectively are indexed.

3. *Revoke*, a probabilistic algorithm, $\boldsymbol{RL_t} \leftarrow$ *Revoke (gpk, grt, ru, reg[i])* where *gpk* is the group public key, *grt* is the group revocation token, *gmk* is the revoked user, and *reg[i]*.

These steps are assumed to have been performed uniformly for all following protocols.

5.3.2 Upload Protocol

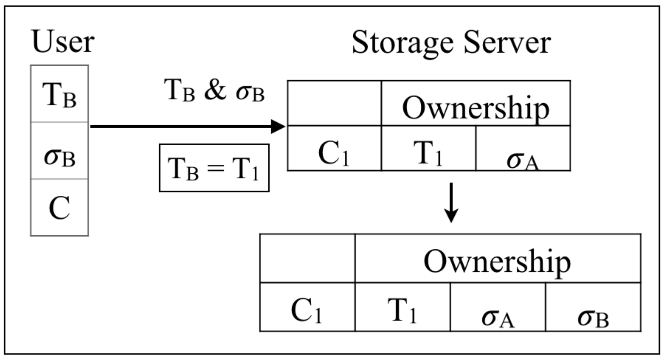


Figure 5.2: Illustration of Upload Protocol

Figure. 5.2 illustrates how to store and share a data file in the storage cloud server. The detail process of encryption and tag generation is based on MLE. The client, GM, and server perform the following operations.

1. The client encrypts the message by taking as input a message M and a message-derived key $k \leftarrow_{\$} K(p, M)$ where p is a public parameter $p \leftarrow_{\$} \{0,1\}^{k(\lambda)}$. A ciphertext C is produced by $C \leftarrow_{\$} Enc(k, M)$.

2. The tagging algorithm T is produced by using $T \leftarrow GTag(p, C)$ in which $GTag$ is tag producing algorithm to detect duplicates and the client constructs the uploading data file and message format as shown in Table. I.

3. The client generates the signature σ by using gpk, gsk_i and the ciphertext C. GM parses $cert_i$ as (γ, Ξ) and gpk as $(R, R_1, R_2, pk_e, pk_s)$. The randomness is generated by $(\gamma', \Xi') \leftarrow Rnd(gpk, sk_i, \gamma, \Xi)$, $\phi \leftarrow SigEnc(pk_e, pk_i, r_i)$ where r_i is random. And then $\pi_1 \leftarrow P_1(\langle pk_e, pk_s, C, \phi, \Xi' \rangle, \langle sk_i, \gamma', r_i \rangle, R_i)$ where π_1 is a non interactive zero knowledge proof and denotes signature by $\sigma \leftarrow GSign(\phi, \Xi', \pi_1)$.

4. The client uploads the ciphertext and signature pair by $Upload$ (C, σ) from the server.

5. The server must check validation and sends (C, σ) pair to the verifier of the group manager.

6. The verifier generate $GVer\ (gpk, (C, \sigma), RL_t)$ and checks validation, and σ parses as (ϕ, Ξ', π_1) and gpk as $(R, R_1, R_2, pk_e, pk_s)$ and assign the output 'valid' or 'invalid'.

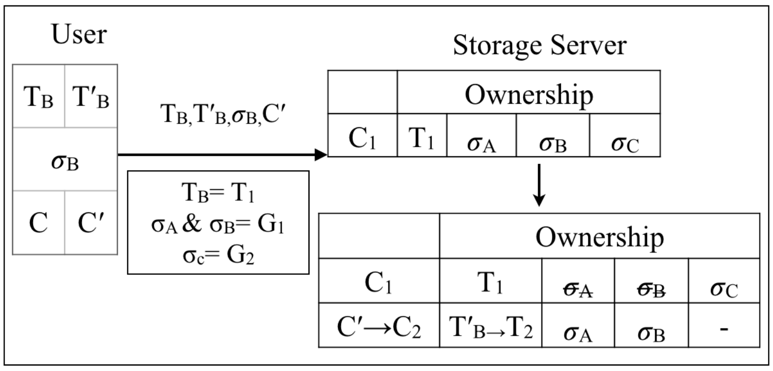
7. After checking the validation of the group, the server checks whether duplicate tag exists on the server or not.

8. If a duplicate file exists, the server keeps σ as this client becomes one of the owners of ciphertext C.

9. The server sends response "duplicate file exist" to the client directly and otherwise, the server replies "no duplicate file exists". See detail in Figure. 5.2.

Finally, the deduplicated encrypted message will be stored in the cloud after the above procedure is performed.

5.3.3 Edit Protocol



User
Storage Server
T_B T'_B
σ_B
C C′
T_B, T'_B, σ_B, C'
$T_B = T_1$
σ_A & $\sigma_B = G_1$
$\sigma_c = G_2$
Ownership
C_1 T_1 σ_A σ_B σ_C
Ownership
C_1 T_1 ~~σ_A~~ ~~σ_B~~ σ_C
$C' \rightarrow C_2$ $T'_B \rightarrow T_2$ σ_A σ_B -

Figure 5.3: Illustration of Edit Protocol

Assume the server already has a ciphertext C of a message M and a client wants to update the existing ciphertext C to C'. Therefore the client must have the original message M_1 and must show that he/she is one of the owners of this ciphertext C. The detail process of encryption and tag generation is based on iMLE. The pre-operation according to the group signature procedures are operated in the way of the setup protocol. After some preparations between the group manager and the member, the following operations are performed as shown in Figure. 5.3.

1. The client encrypts both old and new messages to creates the old ciphertext C and the new ciphertext C'. A message-derived key is $k \leftarrow_{\$} K(p, M)$ where p is a public parameter $p \leftarrow_{\$} \{0,1\}^{k(\lambda)}$ and $k_1 \leftarrow H(p, M)$, $k_2 \leftarrow H(p, M')$ where H is a hash function. Then the new ciphertext C' is produced by $C' \leftarrow Enc(k_2, M)$.

2. The old-new tag pair $(T,\ T')$ is created by $T \leftarrow GTag(p, C)$ and $T' \leftarrow GTag(p, C')$ to check deduplication.

3. The client goes to generate signature σ' by using gpk, gsk and a ciphertext C. [Detail steps of signature generation is operated in the same way as UPL-Dup.]

4. He/she sends a pair of old and new tags, the ciphertext and the signature by $Edit(T,\ T', C',\ \sigma')$ to a server.

5. The server asks for the verifier to check signature validation by sending (C', σ').

6. The verifier checks the signature by the algorithm $GVer(gpk, (C', \sigma'), RL_t)$. The output is 'valid' or 'invalid'. [Detail steps of verification is operated in the same way as UPL-Dup.]

7. After checking the validation of the group, the server maps tag T to meta data table as shown in Table. 5.1.

8. In turn, the server checks the meta data table, whether σ' and all other σ clients originate from the same or different group.

9. If so, then server replaces C_1 with new ciphertext C' , edits the tag value T to T'.

10. Else if C_1 has different σs originating from the different groups, the server will perform the following three steps.

11. Firstly it saves the updated ciphertext C' as a new contents C_2 and adds the data column of ownership as shown in Figure. 5.3.

12. Secondly, the server clusters the two groups according to the ownership of original ciphertext C and the new (updated C') ciphertext C_2. For the task of clustering, we choose the candidate pairs of clusters whose distance are less than a threshold and merge them.

13. Finally, the server performs the clustering to re-organize the tag value and ownerships.

5.3.4 Delete Protocol

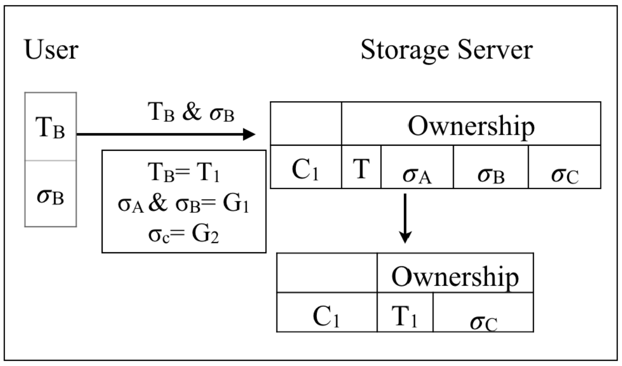


Figure 5.4: Illustration of Delete Protocol

Figure. 5.4 presents a DEL-Dup protocol for an existing message. A client who wants to eliminate a message must have signature σ and tag T for the message.

1. The client sends $\boldsymbol{Delete(name,\sigma)}$ which are a file name and a signature to a server.

2. The server asks for the verifier to check signature validation by sending signature and the associated ciphertext (C, σ).

3. The verifier checks the signature with $\boldsymbol{GVer(gpk,(C,\sigma),RL_t)}$. The output is 'valid' or 'invalid'. [Detail steps of verification is operated in the same way as UPL-Dup]

4. If the server receives the validation of the signature and the revocation list, then it checks the meta data table which is owned by others.

5. After checking the validation of the group, the server maps the tag T to the meta data table as shown in Table. 5.1.

6. If at that time, the index shows that there are no other owners for the ciphertext C, then the server deletes this ciphertext and ownership (C, T and σ).

7. Otherwise, the server has to check the signature of the meta table whose owners are from the same group or the different groups.

8. If all owners are from the same group, the server deletes this ciphertext C and all ownership (tag and signature).

9. If some owners come from different groups, the server must perform two steps. First, the server creates two clusters: cluster1 contains which σ's come from the same group and cluster2 contains the others (which come from different groups). Finally, the server deletes all meta data that exists in the cluster of σ's. See Detail in Figure. 5.4.

5.3.5 Restore Protocol

These restore steps are identical to the any MLE or iMLE [20, 21]. When the user wants to restore a file, the user has to send a request and the file name to the storage server. If the storage server receives a request, the server asks for the verifier to check signature validation by sending ($\boldsymbol{C}$, σ). If it is valid, the storage server sends the ciphertext and tag ($\boldsymbol{C}$, $\boldsymbol{T}$) to the user. The user obtains the ciphertext $\boldsymbol{C}$, the user decrypts it and checks the correctness with tag.

5.4 Security Analysis

DDUP-MUG is designed to ensure data confidentiality and group signature security for cross-group deduplication system. Generally, there are the two types of adversaries; that is external adversary and internal adversary. However, DDUP-MUG resists the external adversary by group authentication. Therefore, we focus on internal adversary by analyzing the data confidentiality and security for group signature. The fundamental security of our approaches is inherited from the original MLE and iMLE. For the sake of self-containment, we briefly review the security of MLE and iMLE. Both MLE and iMLE concern the privacy of unpredictable data and duplicate faking attacks.

The privacy of DDUP-MUG schemes is a pre-requisite for the security of the encryption in these protocols. It concerns plaintexts drawn form a distribution with negligible guessing probability. More concretely, a *CDA* (chosen-distribution attack) consists either of a uniformly distributed string of bits or an encryption of a message drawn randomly form a distribution provided by adversary [20, 2]. The security level is defined by the distinguisher's running time, its advantage over a random guess, and the min-entropy of the distribution that the adversary is allowed to specify. A lower min-entropy requirement corresponds to a stronger security guarantee. Concurrently, the deterministic encryption scheme provides tag consistency quite efficiently. Indeed, if the plaintexts are possible to be recovered, the adversary may compute their tags and test them against that of the challenge ciphertext. Tag consistency provides security against duplicate faking attacks, which means no efficient adversary can find two ciphertexts with matching tags that decrypt to different messages [20, 21].

$KR \quad game : Exp_{SE}^{kr}(\lambda)$
$k \leftarrow K\left(1^{\lambda}\right);$
$k' \leftarrow A\left(\varepsilon_{\lambda}, D_{\lambda}\right);$
$return \quad (k = k')$

We take advantage of MLE and iMLE schemes by Bellare et al. [20, 21] to generate the key, encrypt the message, and produce the tag, all together, in a single process. The scheme is built by using a deterministic symmetric encryption scheme $\mathcal{SE}$ = (*SK, SE, SD*) and random oracle hash function family $\mathcal{H}$ = *(HK, H)*. The encryption scheme uses its message space (typically 0, 1) by a parameter generation *HK*, and shares a common key generation algorithm. $\mathcal{SE}$ consists of a key generation *SK*, a deterministic encryption *SE* and a deterministic decryption *SD*. We need key recovery security and real or random security for the scheme [21]. See KR game in . We denote the KR advantage as $Adv_{SE,A}^{kr}(\lambda) = \Pr[KR_{SE}^{A}(\lambda)]$ and *SE* is *KR*-secure if $Adv_{SE,A}^{kr}(\lambda)$ is negligible for all polynomial time adversary.

```
$ROR \quad game : Exp^{ror}_{SE}(\lambda);$
$b \leftarrow \{0,1\};$
$m \leftarrow \{0,1\}^*;$
$If \quad b = 1, then \quad k \leftarrow K(1^\lambda)$
  $c \leftarrow E(k,m);$
  $return(c);$
$Else \quad return(b);$
```

The formal security definition of real-or-random oracle model is originated from Bellare et al. [1]. If an adversary A is not able to determine the oracles operation of real or random, it is assumed that A is unable to guess encryptions and the encryption scheme be secure with real-or-random security as shown in ROR game. The advantage of an adversary is defined as $Adv^{ror}_{SE,A}(\lambda) = 2 \cdot \Pr[ROR^{A}_{SE}(\lambda)] - 1$ for a negligible function of $Adv^{ror}_{SE,A}(\cdot)$.

```
$P\$ - CDA \quad game : Exp^{p\$-CDA}_{MLE \quad or \quad iMLE,A}(\lambda)$
$p \leftarrow P(1^\lambda);$
$b \leftarrow \{0,1\}; (m, c') \leftarrow D();$
$For \quad i = 1 \quad to \quad n$
  $k_i \leftarrow K(m_i);$
  $c^1_i \leftarrow E(k_i, m_i);$
  $c^0_i \leftarrow \{0,1\}^{|c^1_i|};$
  $b' \leftarrow A(p, c');$
$return \quad (b = b')$
```

Privacy Notation for MLE

MLE is formulated for *CDA* where encryptions of unpredictable messages should be indistinguishable ("*P\$-CDA*" stands for privacy for chosen- distribution attack) which is obtained by the real or random oracle model. In the *P\$-CDA* game definition, M is the message source for MLE. If A is an adversary, we let $Adv^{p\$-CDA}_{MLE,M,A}(\lambda) = 2 \cdot \Pr[P\text{-}CDA^{A}_{MLE,M}(\lambda)] - 1$. That means our encryption is *P\$-CDA* secure for MLE-valid sources if $Adv^{p\$-CDA}_{MLE,M,A}$ is negligible for all messages. *P\$-CDA* maintains the indistinguishability of encryptions for two unpredictable message.

Theorem 1 *Let hash function $\mathcal{H}$ be a random oracle (RO) and $\mathcal{SE}$ = (SK, SE, SD) be a one-time symmetric encryption scheme. Then if $\mathcal{SE}$ is both KR (key recovery)-secure and ROR (real or random)-secure, MLE is P\$-CDA secure.*

Proof Our DEDUP-MUG scheme adopted the MLE one-time symmetric encryption scheme and a hash function family H= $(\mathcal{HK}, \mathcal{H})$. For $\mathcal{SE}$ = *(SK, SE, SD)*: key generation SK, on input 1^λ, output a key K; deterministic encryption $\mathcal{SE}$ maps a key K and plaintext M to a ciphertext C; and deterministic decryption $\mathcal{SD}$ maps a key K and ciphertext C to message M. The probability of $Pr[\mathcal{SD}(\mathcal{K}, \mathcal{SE}) = \mathcal{M}]$= 1 for all $\lambda \in \mathbb{N}$, all $K \in [\mathcal{SK}\ (1^\lambda)]$, and all $M \in \{0, 1\}^*$.

We consider our MLE provide both key recovery and real or random security. In game KR_{SE}, on input $(1^\lambda$, run the adversary A. The query at most once to an encryption oracle with a message M to which the game replies with an encryption of M under key K. Adversary A outputs a bit string K' and wins if $K' = K$. Advantage is defined as $Adv^{kr}_{SE,A}(\lambda) = \Pr[KR^A_{SE}(\lambda)]$ and we say that SE is KR secure if $Adv^{kr}_{SE,A}(\lambda)$ is negligible for all polynomial time adversary.

In game ROR_{SE}, on input (1^λ), firstly choose a random bit b, and then run the adversary A. Adversary A can make multiple queries to an encryption oracle, each query a plaintext $M \in \{0, 1\}^*$. If $b = 1$, then encryption oracle will choose a random key $K \leftarrow_\$ SK(1^\lambda)$ and return $C \leftarrow_\$ (\mathcal{SE}(\mathcal{K}, \mathcal{M}))$. Here, we assume each encryption uses a fresh key. If $b = 0$, encryption oracle will return a random bit string. Adversary needs to guess b to win with advantages of $Adv^{ror}_{SE,A}(\lambda) = 2 \cdot \Pr[ROR^A_{SE}(\lambda)] - 1$ for a negligible function of $Adv^{ror}_{SE,A}(\cdot)$.

We prove *P\$-CDA* is secure when H is a RO. Now, we have *P\$-CDA1* with $b = 1$ and *P\$-CDA0* with $b = 0$. A standard argument be

$$Adv^{P\$\text{-}CDA}_{MLE,A}(\lambda) = 1/2(Pr[P\$\text{-}CDA1^A_{MLE,M}(\lambda)] - Pr[P\$\text{-}CDA0^A_{MLE,M}(\lambda)]).$$

We say that adversary A win with probability of a negligible function of $Adv^{P\$\text{-}CDA}_{MLE,A}(\cdot)$. Our scheme is essentially the same as the scheme of original MLE with encryption, which seems necessary to achieve our notion of security for chosen distribution attack.

Privacy Notation for iMLE

We inherit the feature of interactive iMLE which is introduced by Bellare [21]. The primary security requirement for our encryption schemes is the privacy of unpredictable data as described in P$-CDA game.

Theorem 2 *Let $\mathcal{H}$ be an RO and $\mathcal{SE}$ be both KR-secure and ROR-secure, then iMLE is P$-CDA -secure.*

Discussion

If we use a collision resistant hash function as $\mathcal{H}$ as well different security has been proved. See Theorem 1 proof for the details.

```
$TC \quad game : Exp_{\pi}^{TC}(\lambda)$
$p \leftarrow P(1^{\lambda})$;
$k \leftarrow K(m); (m, c') \leftarrow A(p)$;
$If \quad m = \perp \quad or \quad c' = invalid,$
$then \quad return \quad false :$
$m' \leftarrow D(k, c')$;
$t \leftarrow T(E(k, m)); t' \leftarrow T(c')$;
$If \quad t \neq t' \quad then \quad return \quad false$
$If \quad m = m' \quad then \quad return \quad false$
$If \quad m = \perp \quad then \quad return \quad false$
$return \quad true$
```

Tag Consistency

We turn to security in the sense of tag consistency. The adversary colludes with some users and performs the duplicate fake attacks to the data on the storage server. Specially, the adversary and these colluders may upload their data to the deduplication based system. They upload the correct tags, but replace the data with wrong data. To resist these problem, tag consistency (TC) aims to provide security against the duplicate faking attacks in which a legitimate message is undetectably replaced by a fake one. Both MLE and iMLE have a guarantee for TC security so that an adversary cannot make a client recover a file different from the one he/she uploaded [20, 21].

Theorem 3 *Let $\mathcal{SE}$ = (SK, SE, SD) be a one-time symmetric encryption scheme and tag consistent with any probabilistic polynomial-time adversary A, then no efficient A has non-negligible TC advantage so that $Adv^{exp}_{\pi,A}(\lambda) = \Pr[Exp^{\pi,A}_{TC}(\lambda) \rightarrow true]$ is defined, see TC game.*

Proof We consider the TC game and let A be an adversary. Let $Adv^{exp}_{\pi,A}(\lambda) = \Pr[Exp^{\pi,A}_{TC}(\lambda)]$. We say that MLE is TC secure if $Adv^{TC}_{MLE,A}(\cdot)$ is negligible.

We consider adversary A creates and upload C'. A honest client, having M, A is allowed to pick M,computes $K_\$ \longrightarrow \mathcal{K}_{\mathcal{P}}(\mathcal{M})$, and uploads $C_\$ \longrightarrow \mathcal{E}_{\mathcal{P}}(\mathcal{K}, \mathcal{M})$. The server finds that the tags of C and C' are equal and thus continues to store only C'. Then, the honest client downloads C' and decrypts under K. It expects to recover M, but in a successful deduplicate-faking attack. It recovers instead some $M' \neq M$. TC security protects against this.

Security of Signature

The adversary tries to obtain the signature, and recover the user's data. Specially, the adversary compromise the storage server. However, each group is generated by unlinkable and randomizable group signature [148] . DDUP-MUG ensures the security of group signature.

```
$GS \quad game : Exp^{GS}_{URS,A}(\lambda)$
$b \leftarrow \{0,1\};$
$(gpk, gsk) \leftarrow KeyGen\left(1^\lambda\right);$
$(T_0, \sigma_0, T_1, \sigma_1) \leftarrow A(gpk, gsk);$
$If \quad GVer = (pk, T_0, \sigma_0) = 0 \quad or$
   $GVer = (pk, T_1, \sigma_1) = 0$
$then \quad return \quad false$
$\sigma' \leftarrow Rnd(T_b, \sigma_b);$
$b' \leftarrow A(sk, pk, T');$
$return \quad b'$
```

Theorem 4 *A group signature GS = (keyGen, GSign, GVer, Rand) is unlinkable and randomizable for any probabilistic polynomial time algorithm $Algo$, the probability between $Exp_{GS}^{urs,Algo}(\lambda)$ is negligible (defined GS game) [148].*

$$\Pr[Exp_{GS}^{urs,Algo}(\lambda) = 0] = \Pr[Exp_{GS}^{urs,A}(\lambda) = 1] < \epsilon(\lambda).$$

Proof: For any adversary that is not computationally restricted, a group signature generated by an honest group member is always valid by Definition 1. Ver will always correctly identify the signer the above GS game; the output of GS will always be accepted by the verification algorithm.

We now prove the correctness of the revocation list of our scheme.

Theorem 5 *Revocable Group Signature GS is correct, as defined in Definition 2.*

Proof: We consider public parameter $gpk = (R, R_1, R_2, pk_e, pk_s)$, secrete key gsk where each i, $gsk_i = (pk_i, sk_i, cert_i)$; revocation list $RL_t = (gpk, grt, ru, i)$ as output by key generation algorithm.

An honest signer with secrete key gsk_i for each i generates a signature $\sigma = GSign(\phi, \Xi', \pi_1)$.

In particular, the signer computes the generators by Definition 2. Therefore, the verifier uses the same generators. Now, the first part of signature verification algorithm accepts a signature if $GVer = (gpk, RL_t, \sigma, m) = 1$.

As mentioned above, the adversary cannot obtain other users' data by compromising the storage server or colluding with users. All messages and metadata stored in the storage server is encrypted and the group signature is also randomize. DDUP-MUG can ensure the data security of cloud deduplication.

5.5 Chapter Summary

In this chapter, we have proposed a scheme that supports secure deduplication where several groups are sharing data by using MLE. Multiple group deduplication being present, it is essential to try it out a real system. In doing so, we are taking the utility of an existing source rather than proposing an entirely new one. We first propose three frameworks with cross-group setting which can protect against duplication faking attacks and defend from the unpredictable data attacks. DDUP-MUG fits the original framework of deterministic MLE while satisfying multiple group features by adding the signature scheme. Generally, DDUP-MUG composed of three protocols: UPL-Dup protocol, EDT-Dup protocol, and DEL-Dup protocol. Moreover, DDUP-MUG reduces the bandwidth by sending only tag and signature pair while checking verification and duplication. DDUP-MUG ensures both the message security, tag consistency and the bandwidth efficiency among the cross-group. DDUP-MUG supports extended demands that arise in realistic and secure scenarios.

Chapter 6

Multiple Group Signcryption

In a secure data deduplication scheme, normal public key cryptographic techniques do not support the removal of redundant files. In order to solve this deduplication problem, a message-derived key can be used on encryption and then the same ciphertext of the same plaintext will be identical. Another problem of the deduplication is the ownership of ciphertext; the designated group users need to prove the membership and ownership of the ciphertext. If a tag can consequently generate parallel by the ciphertext, the tag management will be effective in a large amount of encrypted data on the cloud. As both the original message and the signature are encrypted, the filtering process at the storage server cannot authenticate the message independent of the endpoint user. The cloud storage cannot access the signature and the message cannot be decrypted without the receiver key. In addition, we have to think about third parties who should be able to verify the signcrypted message without using any information from the users.

This chapter propose a novel signcryption scheme, “Verifiable Hash Convergent Group Signcryption (VHCGS)”, which possess two properties: group signcryption and proxy signcryption. At the cloud deduplication level, multiple groups of clients can perform signcryption simultaneously and CSP performs verification to further remove the duplicates files, resulting in cost and space reducing. Our proposed scheme is meant to allow to operate such as uploading a signcrypted file through CSP and downloading/ unsigncrypting a file from the CSP. Our VHCGS consists of partial unsigncryption which allows the CSP to check the validity of the true owner of the file and tag correctness which allows the user to check the integrity of its data against the semi-honest CSP.

6.1 Background

6.1.1 Signcryption

Signcryption is a cryptographic primitive which can provide confidentiality and integrity simultaneously [54]. From the viewpoint of data encryption and user ownership, there is a solution called signcryption which has been designed by adding encryption and signature properties [145, 146]. Signcryption is an asymmetric cryptographic method that can simultaneously provide message confidentiality and unforgeability with a lower computational and communicational overhead.

Zheng et al. proposed original signcryption which is "Cost (Signcryption & Encryption) $\ll$ Cost (Signature) + Cost (Encryption)" [145]. It is used to send a message from a sender to a receiver with authenticity and confidentiality. A signcrypted message moves from the sender to the receiver. However, anyone cannot examine the contents of ciphertext except the designated receiver. Furthermore, anyone cannot decide the authenticity of the original message. Gamage et al. proposed a new signcryption method for public verification [67]. Kwak et al. proposed the distributed signcryption using distributed encryption where any party can signcrypt a message and distribute it to a designated group and any participant from the receipent group can unsigncrypt the original message [79, 80]
.

6.1.2 Original Signcryption Scheme

Zheng's signcryption [145, 146] refers to an original cryptographic method that involves the functions of encryption and digital signature simultaneously. The scheme provides a method to save the computational cost of encryption and signature compared to traditional signature-then-encryption. This signcryption is based on the Digital Signature Standard (DSS) with a minor adjustment. Assume $Alice$ (a sender) signcrypts a message and Bob (a receiver) unsigncrypts the message. $(x_a, y_a = g^{x_a})$ and $(x_b, y_b = g^{x_b})$ are the private key and public key pairs for $Alice$ and Bob, respectively. $hash(\cdot)$ denotes a one-way hash function, $hash_k(\cdot)$ a keyed one-way hash function with key k, and $E_k\,(D_k)$ a symmetric encryption and decryption [18].

Signcrypt: Choose $z \in_R Z_R$. Compute $k = y_b^a \bmod p$. Divide k into k_1 and k_2. Compute $r = hash_{k_2}(m)$, $s = z(r + x_a)^{-1} \bmod q$, and $c = E_{k_1}(m)$.

Unsigncrypt: $k = (y_a \cdot g^r)^{s \cdot x_b} \bmod p$. Split k into k_1 and k_2. Compute $m = D_{k_1}(c)$. Verify $r \overset{?}{=} hash_{k_2}(m)$.

6.1.3 Public Verifiable Signcryption Scheme

Generally, signcryption is used to send a message from a sender to a receiver with confidentiality and integrity. One notices that the third party such as proxy or firewall which are widely used today. When a signcrypted message passed through a third trusted party (proxy), it cannot determine the authenticity of the message and it becomes an issue.

Bao et al. [18] proposed a signcryption scheme that allows a third party to publicly verify the message. Nonetheless, it does not offer confidentiality.

Gamage et al. [67] proposed an alternative scheme for publicly verifiable scheme the origin of a ciphertext by anyone (a signature (r, s) on the encrypted message c. This is accomplished in such a way that a storage server can verify that (r, s) is a valid signature c. The Gamage's signcryption departs from the naive "Encryption-then-Sign" composition because c is encrypted by using the random value x.

Signcryption: Choose an integer x randomly from $\{1, ..., q-1\}$ and compute $k = hash\left(y_x^b \bmod p\right)$ and $y =$ third trusted party) without learning anything about the plaintext. In this scheme, a ciphertext (c, r, s) can be thought of as containing $a \bmod p$. The signcrypted message (c, r, s) is computed by *Alice* as $c = E_k(m)$, $r = hash(y, c)$, $s = \frac{s}{r + x_a} \bmod q$.

Unsigncryption: *Bob* can recover the message by full unsigncryption from(c, r, s), $y = (y_a g^r)^s \bmod p$, $k = hash(y^{x_b} \bmod p)$ and $m = D_k(c)$. If $hash(y, c) \overset{?}{=} r$ and the signature is accepted.

Signature Verification: Any verifier will compute from(c, r, s), $y = (y_a g^r)^s \bmod p$ and this step is called *partial unsigncryption* for signature verification. If $hash(y, c) \overset{?}{=}$

r and the signature is accepted. Finally, the verification process is performed without recovering the message.

Unfortunately, the above schemes do not support in the multiple user setting.

6.1.4 Group Signcryption Scheme

In this section, we review the Kwak-Moon's group singature scheme [79, 80]. The sender and the receiver belong to group A (G_A) and group B(G_B) respectively. Additionally, each of communication is a session and the session key does not help an attacker to determine the session key from another session. G_A and G_G share a session key based on the strong RSA assumption. The scheme consists of the following five algorithms: *Setup, Join, Signcryption, Unsigncryption*, and *Open.*

Setup: The *GM* (group manager) initiates the setup protocols to generate signature key, verification key and signature key for each corresponding.

Join: An entity i who wants to join a group generates his own secret key ϵ_i and computes the membership key τ_i. The GM generates each membership certificate and a group public key.

Signcryption: Assume that Alice is a sender from G_A and the receiver belongs to group G_B. $Alice$ wants to signcrypt a message with and then she sends it to the group G_B using group G_B's public key. To signcrypt a message m, $Alice$ uses the one-way hash function and symmetric key encryption and identity of G_A.

Unsigncryption: Bob who belongs to G_B can unsigncrypt the signcrypted message using his membership and group public key. And he discovers the session key and decrypts ciphertext and verify the signature certificate.

Open: In the case of a dispute, Bob forwards the c_1, ω and his public key to the GM of G_A after decrypting signcrypted message and identifying the ID_{G_A}. The GM can then identify the group member, $Alice$, who issued the signcryption by testing whether signature is valid or not.

6.2 Proposed Scheme Architecture

We use Bellare's Hash Convergent Encryption in [20] as the initial scheme to work with because of the following reasons. HCE will encrypt identical data into identical ciphertexts. Another reason why we take advantage of Bellare's HCE is for its tag consistency. In the process of decryption, the tag *Tag* of the ciphertext is checked by re-generating the decrypted tag Tag'. If these two tags are unequal ($Tag \neq Tag'$), then it returns false.

Furthermore, we follow the Kwak-Moon's group signcryption [79, 80] for the properties of the group signature. In particular, the group manager GM is trusted by every group member and he publishes the group public key and corresponding secret key. Verifiable Hash Convergent Signcryption consists of five tuples: (*KeyGen*, *Join*, *Signcryption*, *Partial Unsigncryption*, *Unsigncryption*).

Assume that a sender of a group signcrypts a message and sends it to the server that is associated with the designated group. The server can partially unsigncrypt the ciphertext, that is checking the validity of the ciphertext and the signature. To follow the third party verifiable signcryption, we inherit the properties of Gamage's proxy verifiable signcryption scheme [20]. In this scheme, the proxy(the trusted third party) can verify the ciphertext and the signature without recovering the original message.

Thereafter, we construct our proposed VHCGS according to the following algorithms.

6.2.1 KeyGen

The GM will do the following:

- Choose two random secret l_p bit primes p, q such that $p = 2p + 1, q = 2q + 1$ are prime. Set the modulus $n = pq$.
- Choose random elements $a, a_0, h \in_R QR(n)$ (of order p, q) where QR is defined as the set of quadratic residues.
- Choose his secret element $x \in_R \mathbb{Z}_{pq}$ and sets $y = g^x \bmod n$.

Then the group public key is $Y = (n, a, a_0, y, g, h)$ and the corresponding secret key is $S = (p, q, x)$.

6.2.2 Join

Our scheme is based on the premise that the strong RSA assumption is compatible with the underlying group signature. The detailed steps of Join are as follows:

- User U_i who wants to join a group generates his own secret x_i and sends it to the group manager through an interactive protocol as in the Kwak-Moon scheme. A_i denotes $(a^{x_i} a_0)^{1/e_i} \bmod n$.
- U_i computes $y_i = g^{x_i} \bmod n$ and sends y_i to the GM.
- The signature σ_i for y_i is calculated by the GM and then the GM chooses a random $x \in_R \mathbb{Z}^*_{pq}$ and computes $y = g^x \bmod n$.
- After receiving each member's (y_i, σ_i), the GM checks whether the (y_i, σ_i)'s are valid or not. If they are all correct, the GM then computes the members' $\alpha_i = y_i^x \bmod n$ and generates the group secret shared key $\kappa = H_0 (c||\alpha_1||...\alpha_N)$, where c is the initial vector, $H_0(\cdot)$ is a one-way function, and N is the size of the group.
- The GM sends $(\kappa_i, c, y, \sigma_s)$ to each member, where κ_i is $\kappa \bigoplus H_1 (c||\alpha_i), H_1(\cdot)$ is another one-way hash function different from $H_0(\cdot)$, and σ_s denotes the signature of $\kappa_i||c||y$. Each member checks the signature σ_s, then computes $\sigma_i = y^{x_i} \bmod n$ and recovers the shared secrete value κ with all the group members and publishes $\Omega = g^k \bmod n$.

6.2.3 Signcryption

Consider U_i belongs to the designated group G_a. G_a and G_b are the different group and V_j is the third party (server) who can partially unsigncrypt the message to check the validity of the signature and the ownership of the ciphertext. The G_a's public key is $y_a = (n, a, a_0, y, g, h, \Omega_a)$ and the corresponding GM's secret key is $S_a = (p', q', x)$. Armed with $(x_i, A_i, e_i, y_i, \alpha_i, \Omega_a)$, a group member U_i in group G_a can then generate and send an encrypted group signature of the message (C_1, C_2) to V_j as the server.

- Choose $r_1, r_2 \in_R \{0,1\}^{2l_p}$ and compute the session key $k_s = F(g^{r_2})$, where $F(\cdot)$ is a suitable hash function such that $|F(\cdot)| < |\bar{n}|$.
- Compute the message derived key $k_m \leftarrow (p, m)$ where p is the public parameter.
- Compute $C_1 \leftarrow (b_0, b_1) \leftarrow \left(\bar{n}^{r_1}, \kappa_s \Omega_b^{r_1}\right)$, $\sigma = E_{k_s}(m||C_1)$, $C_2 = HCE(m, k_m)$, $Tag = (C_2, K_m)$, $C_3 = \sigma||C_2$, where $\sigma_{k_s}(\cdot)$ is the Kwak-Moon group signature, $HCE_{k_m}(\cdot)$ is hash convergent encryption.
- Send (C_1, C_3, Tag) to the server.

6.2.4 Partial Unsigncryption

Assume server V_j join the the destinated group and get k_s and y. V_j can partially unsigncrypt by using his group's k_s to do the following with ciphertext (C_1, C_3, Tag).

- Recover the session key $k_s \leftarrow b_1(b_0)^{k_j}$ and message derived key $k_m \leftarrow (p, C_2)^{k_j}$.
- Unsigncrypt $D_{K_s}(C_3) = C_2||\sigma$.
- Verify the signature: if $hash(C_2||\sigma)$, accept the signature.

6.2.5 Unsigncryption

User U_j, one of the receivers in the group G_b, uses his group's k_b to do the following with ciphertext (C_1, C_3, Tag).

- Recover the session key $k_s \leftarrow b_1(b_0)^{k_a}$.
- Unsigncrypt $D_{k_s}(C_3) = C_2||\sigma$.
- Verify the signature using the same verification process as in the Kwak-Moon scheme.
- Decrypt $HDC_{k_m}(C_2) = m$ by using the hash convergent decryption process on input k_m.

- Calculate tag $Tag' = Hash\left(H, H\left(k_m, m\right)\right)$.
- If $Tag = Tag'$, then the message is valid; otherwise invalid.

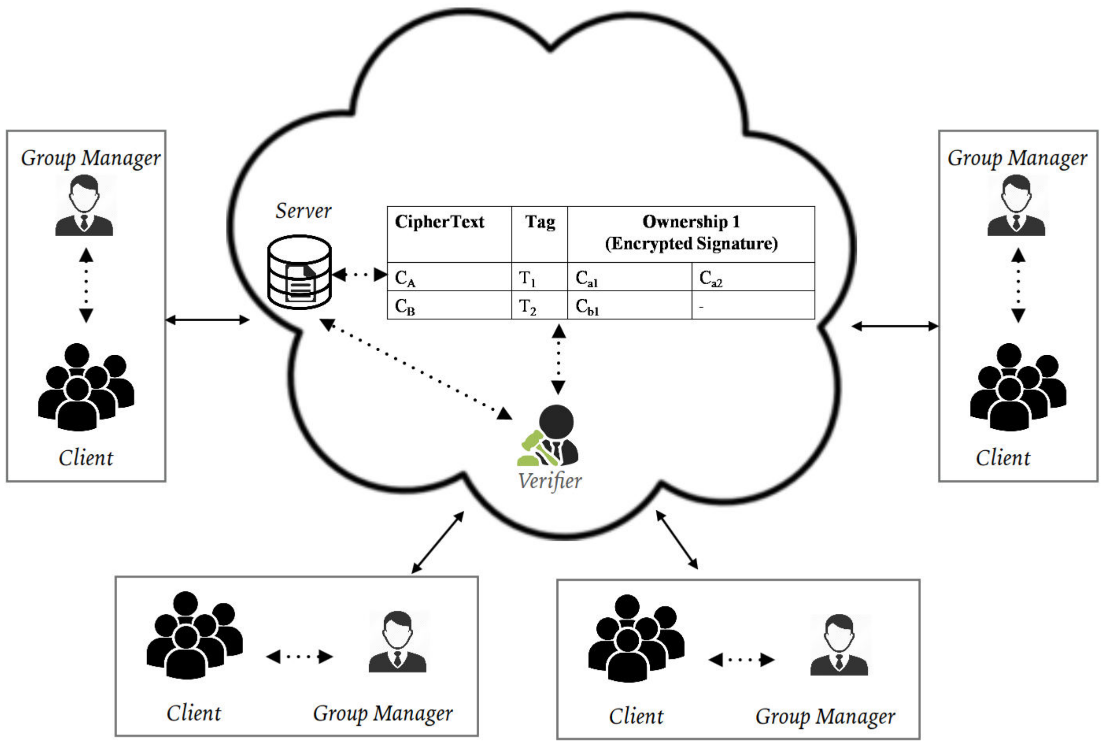
6.3 Application in Cloud Computing

This section focuses on cloud environment deduplication. We propose a framework which consists of embedding VHCGS protocol along with signcryption, partial-unsigncryption and unsigncryption. The design of the proposed framework for secure cloud deduplication involves three participants: group manager, group member/ client and the cloud storage server/ verifier as shown in Figure. 6.1. In our proposed cloud deduplication application aims the following:

- to support Multiple Group Setting which many groups may simultaneously join to the storage server.
- allow third party to publicly verify the signcrypted message (without getting any information of original message)

To consider a more specific application, we built the scenario of the cloud deduplication for VHCGS. First, we design an upload protocol to store a new ciphertext at the storage server. Then we demonstrate a download protocol by which the client can restore a ciphertext by verifying ownership. We assume that:

- a user may register in a group,
- there may be one or more group which connected to the storage server, and the storage server is assumed as a third party for verifier,
- the storage server is allowed to check the ownership of the particular message via the verification process,
- the user (who can proof as a owner) can download the message and unsigncrypt the original message,
- the correctness of the message can be checked.



CipherText	Tag	Ownership 1 (Encrypted Signature)	
C_A	T_1	C_{a1}	C_{a2}
C_B	T_2	C_{b1}	-

Figure 6.1: Cloud Deuplication by using VHCGS

6.3.1 System Setup Protocol

Every client, the GM and the storage server perform the setup operations as shown in Figure. 6.2 . This step is assumed to have been performed uniformly as a setup for the entire framework which consists of *KeyGen* and *Join* from our VHCGS.

The GM initiates the *KeyGen* algorithm of the VHCGS scheme and establishes the group public key gpk and the group manager secret key gsk. When a client wants to join the group as a new group member, the client generates his own secret key usk and sends it to the GM. *Join* is an interactive protocol between a member and the GM. After that, the GM checks the validity and defines *user ID* and shares gpk. Thereafter, the GM establishes the connection (*Join Verifier*) between the verifier of the server by using gpk and *authentication ID* of the verifier.

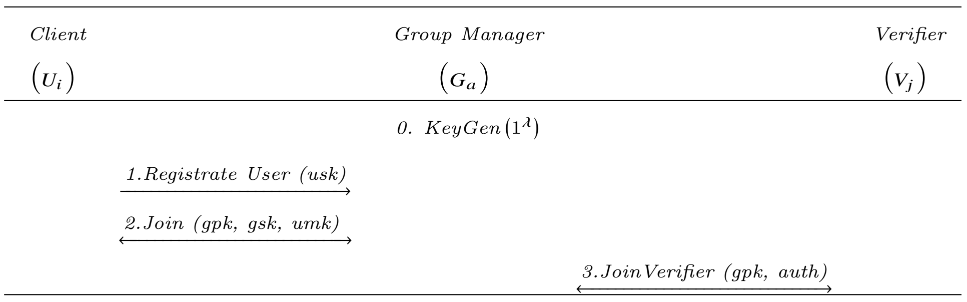


Figure 6.2: VHCGS Setup Protocol for Cloud Deduplication

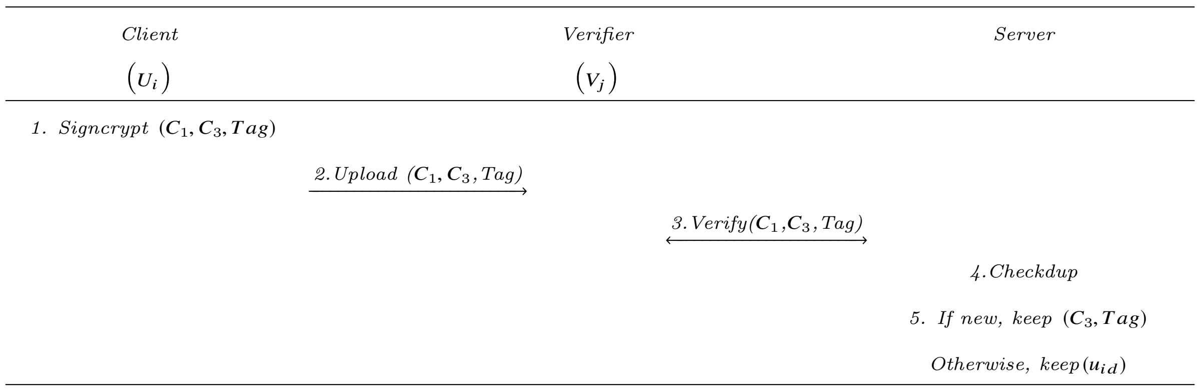

Client
(U_i)
Verifier
(V_j)
Server
1. Signcrypt (C_1, C_3, Tag)
2.Upload (C_1, C_3, Tag)
3.Verify(C_1, C_3, Tag)
4.Checkdup
5. If new, keep (C_3, Tag)
Otherwise, keep(u_{id})

Figure 6.3: VHCGS Upoad Protocol

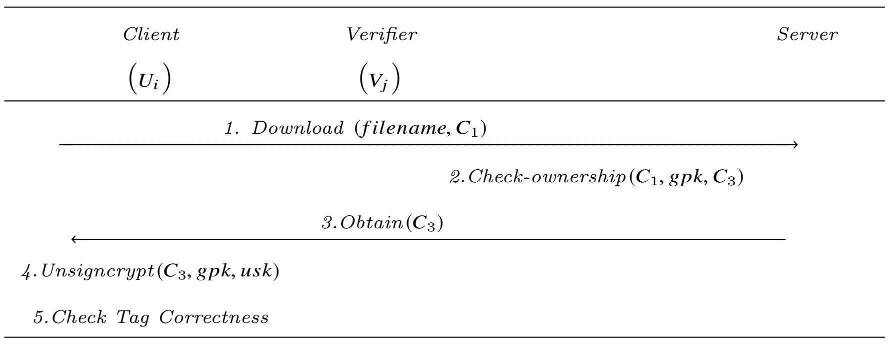

Client
(U_i)
Verifier
(V_j)
Server
1. Download $(filename, C_1)$
2.Check-ownership(C_1, gpk, C_3)
3.Obtain(C_3)
4.Unsigncrypt(C_3, gpk, usk)
5.Check Tag Correctness

Figure 6.4: VHCGS Download Protocol

6.3.2 Upload Protocol

Figure. 6.3 illustrates the deduplicaiton process when a client stores and shares a data file in the storage cloud server. The detailed process of the upload protocol is based on our VHCGS scheme.

Firstly, the client generates a signature of file as C_1. Then he encrypts the file m to C_2 and generates a tag by using the ciphertext C_2. The ciphertext of the file and the ciphertext of the signature is signcrypted into C_3. After that, the client uploads the ciphertexts and the tag pair (C_1, *Tag*) to the server. The server asks to the verifier for verification of the ciphertext. The verifier can partially unsigncrypt to check the validity of the file and return 'valid' or 'invalid'.

After checking the validity of the group, the server checks whether there exists the same ciphertext on the server. If no duplicate file exists on the server, the server keeps $C_1 \rightarrow \sigma$ as the signature (ownership) of the message, C_3 as the ciphertext of the file and *Tag* to check for fake deduplication respectively. Otherwise, the server stores only σ as the new signature (ownership) of the existing ciphertext.

Finally, the deduplicated encrypted message will be stored in the cloud storage server after the above procedure is performed.

6.3.3 Download Protocol

When a client tries to download a file from the storage server, the following steps are performed as in Figure. 6.4. He has to send a download request to the server by using his signature (ownership) of the file. The server asks to the verifier for verification of the ciphertext. As the verifier can partially unsigncrypt to check the validation of the file, return 'valid' or 'invalid'. The server checks the ownership of the ciphertext. Once the server finishes the validation of signature and ownership of message, it gives the ciphertext to the client. Then, the client unsigncrypts the ciphertext. Finally, the client check the tag for the correctness of the ciphertext C_2. In the restore session, the only owner from the designated group can decrypt the ciphertext. The *Tag* is useful for checking deduplication fake.

6.4 Application in Mobile Data Transmission

According to our VHCGS, the signcryption algorithm for each message takes input for real or random and the result of signcryted message is polynomial size. Therefore, we can apply VHCGS in GSM (Global System for Mobile Communications)

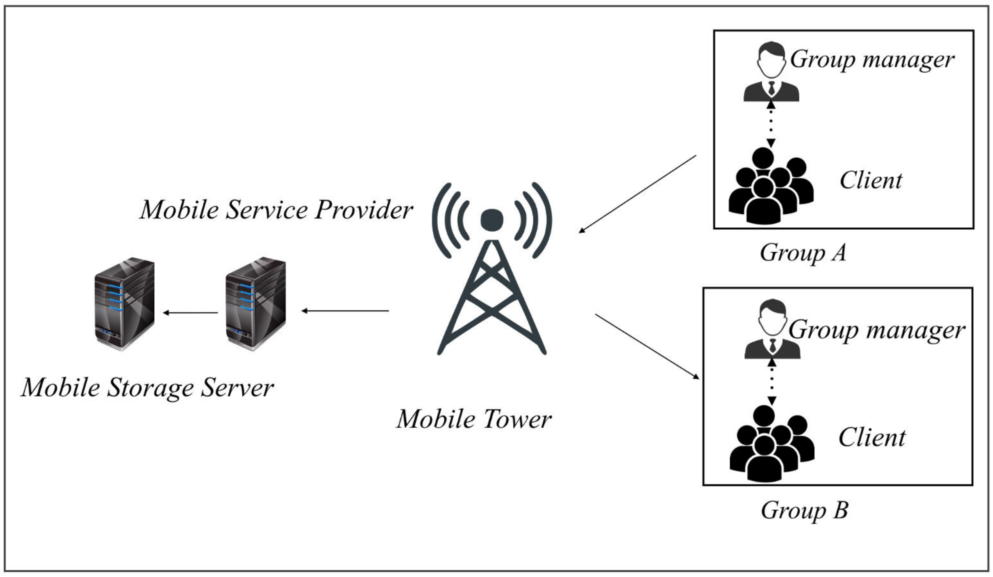


Figure 6.5: Multiple Group SMS transmission by using VHCGS

especially in SMS (Short Message Service) application. Here, we demonstrate how the VHCGS scheme can be used in this framework. An SMS transmission system is composed of a set of protocols in which four participants are involved: the clients, the group manager for each group of the clients, and the service provider as shown in Figure. 6.5.

6.4.1 System Setup

The client, the group manager and the mobile service provider perform the following operations. These steps are assumed to have been performed uniformly as a setup for the entire protocol.

- The group manager initiates the *KeyGen* algorithm of the VHCGS scheme and establishes the group public key and the group manager secret key.
- *Join* is an interactive protocol between a client and the group manager. When a new client wants to join the group, the group member generates his own secret key and sends it to the group manager. After that, the group

manager checks the validity and defines user ID and the share group secret key.

6.4.2 Signcryption

In Figure. 6.5, a message is sent from one group to another group via the service provider. The detailed process is based on our VHCGS scheme.

- The client generates the signature of file C_1.
- And encrypts the file m to C_2.
- Then, the client signcrypts the ciphertext of the file and the ciphertext of the signature into C_3.
- Finally, the ciphertexts (C_1, C_3) is uploaded to the storage provider.

6.4.3 Partial-Unsigncryption

In these steps, the storage server can check the validity of the ciphertext and deduplicate at the service provider side, too.

- The CSP partially unsigncrypts to check the validity of the file and return 'valid' or 'invalid'.
- The CSP checks whether the redundant ciphertext is exist or not. If there is any duplicate file existing at the storage server, the new redundant file is removed and the index goes to the identical message.

6.4.4 Unsigncryption

When the CSP forwards the ciphertext to the destination group, any participants from the destination group can decrypt the original message by the following steps.

- The service provider forwards (C_1, C_3) to the destination group.

- Any clients from the destination group decrypt the ciphertext by discovering the session key and the public key of the group member.
- Finally, the client unsigncrypts the ciphertext.

6.5 Security Analysis

Since it is supposed to provide encrypted group signature, the proposed scheme should have both the group signature's security properties and confidentiality at the same time. In addition, it should also have a membership identification process that confirms whether or not the signer of the signature is indeed the sender to guard against possible misuse by a non-group member with another member's group signature. For the proposed encrypted group signature, most of these properties depend on the Kwak-Moon scheme that is based on the strong $\boldsymbol{RSA}$ assumption.

Correctness: The signcrypted message produced by a valid group member must be accepted by the unsigncryption by checking the Tag , which can be shown by inspection of the protocol.

Definition 4 *(Tag Correctness:) A signcrypted message of $\boldsymbol{VHCGS}(\cdot)$ is tag consistent, if for any a probabilistic polynomial time adversary $\boldsymbol{A}$ for $\boldsymbol{HCE}(\cdot)$ is secure.*

Tag provide security against duplicate faking attack in which a message is replaced by a fake one. In such way, we can prevent from the adversary $\boldsymbol{A}$ creating and uploading message $\boldsymbol{C'}$ [19].

Unforgeability: Only valid group members are able to signcrypt a message on behalf of the group. If we assume that the hash keyed function behaves as a random function $\boldsymbol{hash_k}(\cdot)$, then the group private key will not be revealed to anyone. Therefore this is an unforgeable protocol.

Anonymity: The groups' and the entities' information are encrypted by $E_k(\cdot)$ and $HCE(\cdot)$. No one (including the receiver) can extract the information about the signer except the group manager. The formal security definition of real-or-random oracle model is originated from [19] .

Definition 5 *(ROR): Let SE=(Gen, Enc, Dec) be a public key encryption system with security parameter λ. An adversary of ROR game is define as follows for $\{real, random\}$:*

```
$ROR \quad game : Exp_{SE}^{ror}(\lambda)$
$b \leftarrow \{0,1\};$
$m \leftarrow \{0,1\}^*;$
$If \quad b = 1, then \quad k \leftarrow K\left(1^{\lambda}\right);$
$c \leftarrow E(k,m); return(c);$
$Else \quad return \quad b;$
```

If an adversary A is not able to determine the oracles operation of real or random, it is assumed that A is unable to guess encryptions and the encryption scheme be secure with real-or-random security as shown in above ROR game. The advantage of an adversary is defined as $Adv_{SE,A}^{ror}(\lambda) = 2 \cdot Pr\left[ROR_{SE}^{A}(\lambda)\right] - 1$ for a negligible function of $Adv_{SE,A}^{ror}(\cdot)$.

Theorem 6 *(VHCGS-CDA): Let $HCE(\cdot)$ is ROR secure mode $\in \{real, random\}$ then the VHCGS is in a security model for chosen distribution attack.*

Proof We consider as same as the HCE. In game ROR_{SE}, on input (1^{λ}), firstly choose $\{real, random\}$ mode, and then run the adversary A. Adversary A can make multiple queries to an encryption oracle, each query a plaintext $M \in \{0,1\}^*$. If $b = real$, then encryption oracle will choose a random key $K \leftarrow_{\$} SK(1^{\lambda})$ and return $C \leftarrow_{\$} (\mathcal{SE}(\mathcal{K}, \mathcal{M}))$. Here, we assume each encryption uses a fresh key. If $b = random$, encryption oracle will return a random bit string. Adversary needs to guess b to win with advantages of $Adv_{SE,A}^{ror}(\lambda) = 2 \cdot \Pr[ROR_{SE}^{A}(\lambda)] - 1$ for a negligible function. We say that hash function H is secure in VHCGS-CDA if $Adv_{VHCGS}^{cda}(\cdot)$ is negligible.

```
$VHCGS - CDA \quad game : Exp_{cda}^{HCE}(\lambda)$
$(pk, sk) \overset{R}{\leftarrow} Gen(1^{\lambda})$;
$(m, mode) \overset{R}{\leftarrow} A(1^{\lambda})$;
$if \quad mode = real,$
$C \leftarrow Enc_{pk}(m, r)$;
$if \quad mode = random,$
$m' \overset{R}{\leftarrow} M, C \leftarrow Enc_{pk}(m', r)$;
$b \overset{R}{\leftarrow} A(C, mode)$;
$return \quad b$
```

Unlinkability: It is computationally difficult to figure out whether two signcrypted ciphertexts (C_3, C_3') were generated by the same group member or not. C_1 is generated using a random r_1, while C_2 is encrypted using $HCE(\cdot)$. Even if the receivers can decrypt the ciphertext, they cannot determine it because of the Kwak-Moon-scheme and the Bellare-HCE scheme.

Exculpability: No one can obtain any information about a group private key x_i from $\alpha\,(= y^{x_i})$. Thus the value of x_i is computationally hidden. Moreover, GM cannot signcrypt on behalf of a group member because computing the discrete logarithm is assumed to be infeasible.

Traceability: Any verifier can partially unsigncrypt (C_1, C_3, Tag) to check the validity of membership and can recover the membership certificate via the facilities of the Gamage-scheme and the group signature properties.

Coalition-resistance: The underlying the Kwak-Moon group signcryption is a provably secure coalition-resistant scheme and the secret shared key k computed by the GM cannot be computed by any coalition attack except valid group members who know his $\alpha_i = y^{x_i}$, where x_i is the entity's private key.

Membership identification: A signcrypted message C_3 is generated by a non-group member. C_1 cannot be computed from $\sigma(m||C_1)$ without a verifiable session key. Hence, only valid group members can generate valid encrypted group signatures.

Confidentiality: According to Bellare's HCE scheme, the adversary does not know about the secret information of the user. Furthermore, the adversary can recover neither signature nor data information without finding out the session key k_s.

The signed message C_3 is encrypted by $HCE(\cdot)$. Even if an adversary got the ciphertext C_3, the confidentiality of the data is still ensured.

Definition 6 *(Session Key Freshness) A protocol excecution that entity B believes it has undertaken with entity A and using session identity session key is fresh if*

1. *entity B has successfully terminated the protocol execution,*
2. *the attacker has not required the Reveal oracle on the input (ID_{G_A}, ID_{G_B}, session key) unless the attacker has called the Expire (ID_{G_A}, ID_{G_B}, session key) oracle,*
3. *the attacker has not required the Reveal oracle on the input (ID_{G_B}, ID_{G_A}, session key) unless the attacker has called the Expire (ID_{G_B}, ID_{G_A}, session key) oracle,*
4. *the attacker has not required the corruption oracle on either ID_{G_A} or ID_{G_B}.*

Theorem 7 *(IND-VHCGS-CCA): If session key establishment is secure for outsider then the VHCGS is in a security model in which the attacker make no Expire queries and in which session identity is defined to be ciphertext C.*

Proof We say that our VHCGS signcryption scheme, which is proved to be secure against distinguisher under adaptive chosen-message attacks if $\Pr[Exp^{ind-vhcgs-cca}_{ghcgs,A}(\lambda) = 1] \leq \epsilon(\lambda)$ for all adversaries A running on polynomial time algorithm and sending at most q_s queries to sincryption oracle on each sender and q_u queries to unsigncryptin oracle for each receiver in the following experiment: $Exp^{ind-vhcgs-cca}_{ghcgs,A}(\lambda)$.

- The challenger C runs algorithm $K \rightarrow Setup(1^k)$ of signcryption scheme and give the adversary A K. And C runs $KeyGen(K, ID_i)$ to generate all key pairs (pk_{IDi}, sk_{IDi}), secrete parameter sk_{IDi} for all i. The challenger implements the oracles for A.

- The adversary A receives all public keys of honest parties and may start to perform at most q_s to signcryption oracle on each sender and q_u queries to unsigncryption oracle on each receiver and makes a polynomial umber of $Corrupt$corruption queries and $EstablishParty$ quries where the number of $EstablishParty$ is determined by the session time t by Definition of "Session key freshness".

- At the end of the game, A may terminate with the output of a signcrypted ciphertext C_A and identities of sender ID and receiver ID. Finally, 0 is returned if one of the following conditions holds:

  - the sender ID is corrupted,
  - Unsigncyption $=\perp$;

  otherwise, return 1.

The proof of security is made possible by a technical detail in the security model: the definition of C as the session identity and the inability of the attacker to expire session. These combine to mean that the attacker cannot re-submit a ciphertext C to entity B as this would involve sending to completed session.

6.6 Chapter Summary

We have proposed a signcryption scheme that supports secure deduplication where several groups are sharing data by using VHCGS. This is an attempt to try out cross-group user deduplication in a real cloud environment. We use VHCGS scheme not only in the cloud deduplicaiton but also in SMS transmission which can protect against duplication for the service providers and defend against unpredictable data attacks. VHCGS fits the original framework of deterministic hash convergent encryption while satisfying a group feature by adding the cloud server verifiable group signcryption. VHCGS is composed of three protocols: a setup protocol, an upload protocol, and a download protocol. VHCGS ensures both message security and tag consistency as well as the bandwidth efficiency of the group user and cloud storage server. VHCGS supports the extended demands that arise in realistic and secure scenarios.

Chapter 7

Related Research and Future Improvements

The one of the most important of cloud service is information security and privacy to give the owners. And information on the cloud became frequently large amount of increasing in a scrambled frame. As we mention in previous chapters, encryption information presents the difficulties for cloud deduplication, which gets to be distinctly vital for handling in cloud. Even conventional deduplication plans are good ways to keep the storage size of the cloud, we would like to try the other approaches to control the storage size and ownership of the information. Hence, we will present the new proxy re-encryption scheme to improve the cloud deduplciation. In this paper, we propose a plan which can be used in deduplicate encoded information put away in cloud of ownership properties of message and intermediary re-encryption.

This chapter presents features in related studies that could be applied to improve the deduplication framework but were outside the scope of deduplicaple state.

7.1 Proxy Re-encryption

Proxy Re-Encryption (PRE) is a public key cryptosystem which allows a semi-trusted proxy to transform a ciphertext encrypted under one key into another ciphertext of the same plaintext under another key, without revealling any information of the plaintext. The concept of the proxy cryptosystem which is also called proxy decryption, is introduced by Mambo et al. [91]. A proxy decryptor decrypts the encrypted ciphertext by Alice's public key on behalf of Alice. In 1998, Blaze et al. [30] proposed the notion of proxy re-encryption which is similar to the proxy cryptosystem. In proxy re-encryption, the proxy converts Alice's ciphertext to a new ciphertext for Bob without exposing the plaintext. Thus, the proxy should be key independent to avoid compromising the private keys of the sender (Alice) and the receiver (Bob). The scheme is only useful when the mutual trust relationship exists between Alice and Bob. In the PRE scheme, a proxy with re-encryption keys can change a ciphertext for Alice (a delegator) into another ciphertext of the same plaintext for Bob (a delegatee). The proxy cannot get any information about the plaintext or the private key [30, 41, 74, 110].

The PRE scheme can be calssified into two types: *unidirectional* PRE schemes and *bidirectional* PRE schemes. In the unidirectional proxy re-encryption [77], Alice can delegate to Bob without having to delegate to Alice without any secret key of Bob. It only requires the secret key of Alice and the public key of Bob to discover a re-encryption key from Alice to Bob. It does not allow Bob to Alice (the reverse direction).

In 2010, Matsuda, Nishimaki, and Tanaka [93] proposed a bidirectional proxy re-encryption scheme without bilinear maps. The re-encryption key to converting ciphertext from Alice to Bob can be also used to translate from Bob to Alice. Canetti and Hohenberger [41] proposed the CCA (chosen ciphertext attack) secure bidirectional multi-hop PRE scheme in the standard model by using the bilinear maps. A bidirectional single-hop PRE scheme without bilinear maps is considered by Deng et al [53] in 2008 and then it is also CCA secure in random oracle model.

Bidirectional PRE schemes have attracted much attention from the cryptography community [14, 30, 73, 77, 89, 94, 103, 134] because they have many interesting and useful applications, such as the email forwarding, the encrypted files distribution, the digital rights management [104] and the cloud data sharing [62].

There is another type of PRE scheme which is called *group-based* proxy re-encryption. Group communication becomes popular in many applications. Generally speaking, two groups are supposed: a sender group and a receiver group. Any member from the sender group can encrypt a message and send to the designated receiver group. Any member from the designated receiver group can decrypt the ciphertext. However, there have some problems such as proxy re-encryption or forwarding message. For example, by dividing tasks among a group A and another group B, an encrypted message from the group A should be allowed to decrypt by the group B. Such kind of scenarios is supported by the group-based PRE scheme, which was first proposed by Ma et al.[90]. Their scheme is bidirectional proxy re-encryption scheme. In their scheme, a message is sent from the group A to the group B, any member from the group B can decrypt the ciphertext. And the proxy allows the reverse direction. Because re-encryption key is generated by using the private key of the group A and the group B.

In 2006, Ateniese [14] proposed a proxy re-encryption scheme which supports unidirectional proxy re-encryption and uses the delegator's private key for protecting the collusion of a proxy and a delegatee. However, this scheme is lacking the *non-transferable* property. This problem was first addressed by Libert in 2008 [89]. The proxy and delegatee are quite difficult to protect from colluding. Libert et al. solved the problem instead of preventing the collision of proxy and delegatee, by using *traceable* proxy re-encryption. However, it cannot prevent the re-delegation of the proxy.

Wang et al. [134] proposed "identity-based proxy re-encryption scheme" to solve the problem of proxy colluding, in 2010. Their major advantages are that the proxy and the delegates cannot delegate the decryption right to the others without the permission of the public key generator (PKG). However, PKG in their scheme can decrypt both of the original ciphertext and the re-encrypted ciphertext. This means that the transferable problem still cannot be solved.

In 2015, Wang at al. [136] proposed a new scheme for protecting critucal information systems which is based Cramer-Shoup encryption scheme. But, it can not achieve the delegator's $IND - CCA$ security for the proxy and the delegatee.

All above schemes do not provide non-transferable property or collusion property for the proxy re-encryption scheme. For example, the proxy or Bob can collude to get the decryption key to anyone. In addition, the re-encryption key for the proxy is generated by the trusted private key generator (PKG). However, this kind of schemes have the key escrow problem which PKG is a malicious and PKG can decrypt the original ciphertext or re-encrypted ciphertext. Furthermore, PKG can generate many re-encryption keys for adversary without accessing any right from the Alice. This problem is called PKG despotism problem. In 2012, He et al.[76] proposed the non-transferable proxy re-encryption scheme which is suitable for the key escrow problem and the PKG despotism problem. Their scheme is based on the certificateless cryptography.

7.1.1 Motivation of Proxy Re-encryption

Proxy re-encryption has many practical applications such as email forwarding, health care cloud system and so on. For example, health care cloud system is a cloud computing service using for storing, maintaining and backing up personal health information of the patient. This system acts as a third party between physicians and patients. Therefore, this system needs to secure for patient's health records and their biometric data.By using proxy re-encryption techniques, the health care cloud system can be secured as in [137].

In particular, our proxy re-encryption can be used in secure file sharing system. In distributed file sharing system, the third party is difficult to be trusted from the confidential point of view. Therefore, the distributed file users are desired to apply the encryption methods for the confidentiality. The proxy can distribute the encrypted file without using the information of original data. With group or non-group members, our scheme supports to decrypt by using the authority of the sender. When a receiver accesses to the proxy to request forwarding the ciphertext, the proxy re-encrypt the message without learning any information from the original ciphertext or key.

Our new PRE scheme takes the background idea of non-transferable proxy re-encryption scheme [76] to support the non-transferable property. We suppose that there are three kinds of participants: a delegator i (Alice), a delegatee j (Bob) who is the same group with Alice and another delegatee j (Charlie) who is from the outside of the group. Therefore, we need to think about the non-transferable property for both the same group or the outsider. For the communication between Alice and Bob, we borrow the idea of [90] for group signature properties. Alice does not need to send her certificate to the proxy or Bob. By using delegatee ID from Alice's and Bob's public keys, the proxy can generate the re-encryption key. At the point of the communication of Alice to Charlie, Alice has to send her signature (certificate) of the designated ciphertext to Charlie. When Charlie sends his public key and the proof of Alice signature, the proxy can generate the re-encryption key.

7.1.2 Syntax of our PRE scheme

Our proposed scheme belongs to thirteen algorithms. "Delegator *i*" (Alice) owns a message, "Delegatee *j*" (Bob) is the same group with "Delegator *i*" (Alice) and "Delegatee *k*" (Charlie) is the receiver of the message without same group of "Delegator *i*" (Alice) as shown in Figure.7.1.

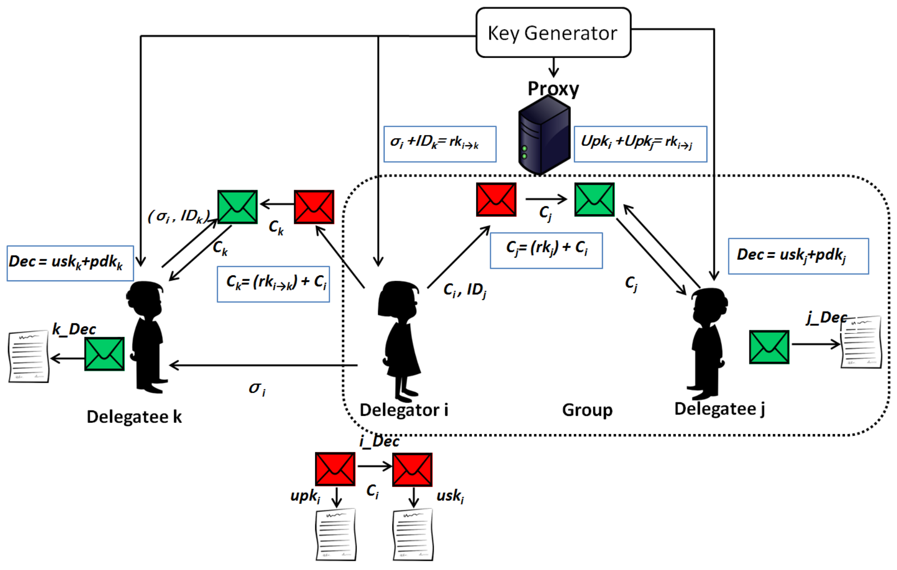


Figure 7.1: Proxy Re-encryption Scheme

- Setup: From the input of the security parameter 1^k, the public parameters mpk and the master secret key msk are generated.
- Key Generation:
  - Set-Secret-Value. The algorithm generates a secret value which is only known to the user himself.
  - Partial-Private-Key. On input a user's identity ID and msk, the algorithm generates a partial private key for the user.
  - Set-Private-Key. On input the partial private key and the secret value, the algorithm outputs the whole private key for the user.
  - Set-Public-Key. On input a user's identity ID and a secret value, the algorithm generates a public key.

- Private Key Correctness Check: The algorithm checks the correctness of the private key.
- Encryption: The encryption algorithm takes a public key upk_A of the delegator $Alice$ and a message m as input, outputs a ciphertext C_A encrypted under upk_A.
- $Alice$-Decryption (delegator i): The decryption algorithm takes a private key usk_i of the delegator i and a ciphertext C_i as input, outputs the message m.
- Bob-Re-Encryption Key Generation: The algorithm verifies the delegator $j's$ signature and the public key. The re-encryption key generation algorithm outputs a re-encryption key rk_j and other relational values.
- Bob-Partial-Decryption-Key Generation: The algorithm checks the correctness of the re-encryption key, and generates a partial decryption key.
- Bob-Re-Encryption: The re-encryption algorithm takes re-encryption key rk_j and ciphertext C_i as input, outputs a re-encrypted ciphertext C_j under upk_j.
- Bob-Decryption (delegatee j): The decryption algorithm takes the private key usk_j of delegatee j, the partial decryption key and the ciphertext C_i as input, outputs message m.
- $Charlie$-Re-Encryption Key Generation: The algorithm verifies the delegator $k's$ ID and the public key. The re-encryption key generation algorithm outputs a re-encryption key rk_k and other relational values.
- $Charlie$-Partial-Decryption-Key Generation: The algorithm checks the correctness of the re-encryption key, and generates a partial decryption key.
- $Charlie$-Re-Encryption: The re-encryption algorithm takes re-encryption key rk_k and ciphertext C_i as input, outputs a re-encrypted ciphertext C_k under upk_k.
- $Charlie$-Decryption (delegatee k): The decryption algorithm takes private key usk_k of delegatee k, the partial decryption key and ciphertext C_i as input, outputs the message m.

7.2 Future Improvements

Cloud computing offers another method for deduplication benefits by improving different assets (e.g., capacity, figuring) and giving them to clients in view of their requests. Cloud service provider provides a major sharing by connecting multiple group together. It must have the properties such as adaptability, flexibility, adaptation to internal failure, and pay-per-utilize. In this way, it has turned into a promising administration stage. The most essential cloud administration is information management among the different users or different groups.

The clients transfer individual or classified information to the server farm of a Cloud Service Provider (CSP) and permit it to keep up these information. Since interruptions and assaults towards touchy information at CSP are not avoidable, it is reasonable to expect that CSP cannot be completely trusted by cloud clients. Also, the loss of control over their very own information prompts to big data security dangers, particularly information protection spillages. Because of the quick improvement of information mining and different examination innovations, the security issue gets to be distinctly honestly. Subsequently, a great practice is to encrypt the information on the cloud with a specific end goal to guarantee information security and client protection. Be that as it may, the same or diverse clients may transfer copied information in scrambled frame to CSP, particularly for situations where information are shared among numerous clients.

Our proposed PRE scheme can be adopted to data ownership and management of encrypted data storage with deduplication. Solutions were given to add cross' group file sharing in the future.

Chapter 8

Conclusion

As digital data is growing tremendously, cloud storage services are gaining popularity since they promise to provide convenient and efficient storage services that can be accessed any time, from anywhere. At the same time, with the advent of cloud computing and its digital storage services, the growth of digital content has become irrepressible at both the group and individual levels. These increasing volumes of data among different group of cloud users need some methods for the storage, processing and availability and cloud technology offers all the potentials to fulfill these requirements. A major issue is handle by the deduplication technology of the cloud storage service. Although data is encrypted, it is difficult to guarantee of data privacy without management of CSP which are sharing among the different group of client. In this thesis, we have addressed the issue of the security of the data in the cloud storage by two approaches when these data are shared among multiple group.

8.1 Discussion

Encryption, verification, and access control are three important domains in security field. We explored these two aspects in mutiple group and cloud environment as shown in Table. 8.1. For both schemes of this thesis, our results show that our strategy gives secure and efficient storage space savings for cross group setting of cloud environment.

The first one (DDUP-MUG) is an cross group data deduplication scheme em-

	DDUP-MUG	VHCGS
1. Deduplication	Yes	Yes
2. Side	Server	Server
3. Type	Cross Group	Cross Group
4. Encryption	MLE/iMLE	HCE
5. Ownership	Group Signature	Encrypted Signature
6. Upload	Yes	Yes
7. Update	Yes	No
8. Delete	Yes	No
9. Download	No	Yes
10.Secrity Proof	Encryption & Signature	Signcryption

Table 8.1: Comparison of Our Proposed Schemes

bedding the unlinkable randomizable group signature (URS) group signature while ensuring that the CSP can perform data deuplication with taking the duties of verifier. We improve the work of MLE by proposing a mutilple multiple group setting for cloud service provider as well as cloud users. In particular, we solve the problem of

1. enhancing the convenience and security of MLE
2. accessing revoke user in each group and
3. protecting data on updating from other groups of users or lost of data during the deduplication process.

DDUP-MUG consists of five protocols: setup protocol, join protocol, upload protocol, edit protocol and delete protocol. At the point of security, DDUP-MUG against Chosen Distribution Attack (CDA) by the encryption of message drawn by randomly. Concurrently, DDUP-MUG provides the tag consistency which is against the duplicate faking attack which means adversary cannot find two ciphetexts by matching the tags of two different files. Furthermore, when every group are controlled by the group manager and each group is generated by URS, DDUP-MUG ensure the security of every signature. Therefore, entire cloud deduplcation environment is protected by the group signature and encryption.

The second one focuses on verifiable hash convergent group signcryption (VHCGS) which can be used in cloud computing. In our scheme, both signature and original message is encrypted. For authentication from the credential, the client generates signature for proving the possession of data for cloud service providers, without asking any information of message or client. VHCGS involves the functions of encryption and signature simultaneously which provides to save the computational cost of encryption and signature compared to traditional signature-then-encryption. In our proposed VHCGS scheme, there are five algorithms which are ***KeyGen***, ***Join***, ***Signcryption***, ***Partial Unsigncryption*** and ***Unsigncryption***. In order to address the deduplication issue, the CSP needs to check the duplication of the files and to verify the ownership the files. Therefore, we think about the " Partial Unsigncryption" step which the CSP can verify the signcrypted message without any knowledge of the original message. VHCGS also supports multiple group user deduplication in a real cloud environment. We use VHCGS scheme not only in the cloud deduplicaiton but also in SMS transmission which is against duplication for the cloud providers (or mobile service providers) and defend against unpredictable data attacks.

The novelty of our scheme stems from combining and exploiting mutiple group signatures as well as signcryption so that we can randomize the signature to make the signature look different for multiple groups of users and hide information of the messages which are not the concerns of the CSP. In this thesis, the duduplication is processed by using the message-locked encryption (MLE) and hash convergent encryption (HCE). Here, both encryption methods are relatively new cryptographic methods of convergent encryption. MLE is improved to applicable at cross group data deduplication. Furthermore, HCE is also adding the properties of signcryption by using session key and message derived key for cross group deduplication process.

References

[1] A guide to data de duplication, "http://www.computerweekly.com/feature/A-guide-to-data-de-duplication", 2017.

[2] M. Abadi, D. Boneh, I. Mironov, A. Raghunthan, G. Sev, "Message-locked encryption for lock-dependent messages", *CRYPTO'13*, Lecture Notes in Computer Science, vol. 8042, pp. 374–39, 2013.

[3] A. Agarwal, R. Saraswat, "A survey of group signature technique, its applications and attacks", *International Journal of Engineering and Innovative Technology' 13*, vol. 2, issue 10, pp. 28–35, 2013

[4] R. Agrawal, J. Kiernan, R. Srikant, Y. Xu, "Order preserving encryption for numeric data", *Proceedings of the 2004 ACM SIGMOD international conference on Management of data. SIGMOD' 04*, pp. 563–574, 2004.

[5] K. Akhila, A. Ganesh, C. Sunitha, "Ensuring security for Fixed Block Level Deduplication in Cloud Backup", *IOSR Journal of Computer Engineering*, pp. 06–12.

[6] K. Akhila, A. Ganesh, C. Sunitha, "A Study on Deduplication Techniques over Encrypted Data", *Procedia Computer Science*, vol. 87, pp. 38–43, 2016.

[7] J. Alderman, J. Crampton , K. M. Martin, " Cryptographic Tools for Cloud Environments", *Guide to Security Assurance for Cloud Computing, Computer Communications and Networks' 15*, pp. 15–30, 2015.

[8] AmazonS3, " http://aws/amazon.com/s3s", 2017.

[9] R. Anderson, "Security Engineering: A Guide to Building Dependable Distributed Systems", John Wiley and Sons, Inc., 2001.

[10] E. Artin, "Theory of Braids, Annals of Math.", no. 48, pp. 101–126, 1947.

[11] F. Armknecht, J. Bohli, G. O. Karame, F. Youssef, “Transparent Data Deduplication in the Cloud”, *ACM Conference on Computer and Communications Security' 15*, pp. 886–900, 2015.

[12] F. Armknecht, J. Bohli, G. O. Karame, Z. Liu, C. A Reuter, C.A, “ Outsourced proofs of retrievability”, *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security' 14*, pp. 831–843, 2014.

[13] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, D. Song, “ Provable data possession at untrusted stores”, *Proceedings of the 14th ACM Conference on Computer and Communications Security' 07*, pp. 598–609, 2007.

[14] G. Ateniese, K. Fu, M. Green, S. Hohenberger, “ Improved proxy re-encryption schemes with applications to secure distributed storage”, *ACM Transactions on Information and System Security' 06* vol. 9, pp. 1–30, 2006.

[15] J. Baek, B. Lee, and K. Kim, “Provably Secure Length-Saving Public-Key Encryption Scheme under the Computational Diffie-Hellman assumption”, *Electronics and Telecommunications Research Institute (ETRI) Journal' 00*, vol. 22, No. 4, pp. 25–31, 2000.

[16] J. Baek, “Construction and Formal Security Analysis of Cryptographic Schemes in the Public Key Setting”, *Monash University*, pp. 1–184, 2004.

[17] M. Barbosa, P. Farshim, “Delegatable homomorphic encryption with applications to secure outsourcing of computation”, *Topics in Cryptology CT-RSA' 12*, Lecture Notes in Computer Science, vol. 7178, pp. 296–312, 2012.

[18] F. Bao, R. H. Deng, “A Signcryption Scheme with Signature Directly Verifiable by Public Key”, *Public Key Cryptography' 98*, Lecture Notes in Computer Science, vol. 1431, pp. 55–59, 1988.

[19] M. Bellare, A. Boldyreva, A. Neill, “Deterministic and efficiently searchable encryption”, *Advances in Cryptology , CRYPTO' 07*, Lecture Notes in Computer Science, vol. 4622, pp. 535–552, 2007.

[20] M. Bellare, S. Keelveedhi, T. Ristenpart, “Message-Locked Encryption and Secure Deduplication”, *EUROCRYPT' 13*, Lecture Notes in Computer Science, vol. 7881, pp. 296–312, 2013.

[21] M. Bellare, S. Keelveedhi, "Interactive message-locked encryption and secure deduplication", *Public-Key Cryptography' 15*, Lecture Notes in Computer Science, vol. 9020 , pp. 516–538, 2015.

[22] M. Bellare, "Practice-Oriented Provable-Security, Lectures on Data Security", *Modern Cryptology in Theory and Practice' 88*, Lecture Notes in Computer Science, vol. 1561, pp. 1–15, 1988.

[23] M. Bellare, A. Desai, D. Pointcheval and P. Rogaway, "Relations Among Notions of Security for Public-Key Encryption Schemes", *Advances in Cryptology Proceedings of CRYPTO' 98*, Lecture Notes in Computer Science, vol. 1462, pp. 26–45, 1998

[24] M. Bellare,T. Ristenpart, P. Rogaway, T. Stegers, " Format-preserving encryption", *Selected Areas in Cryptography' 09*, Lecture Notes in Computer Science, vol. 5867, pp. 295–312, 2009.

[25] M. Bellare, P. Rogaway, "Optimal Asymmetric Encryption, Advances in Cryptology", *Proceedings of EUROCRYPT' 94*, Lecture Notes in Computer Science, vol. 950, pp. 92–111, 1994.

[26] M. Bellare, P. Rogaway, " Random Oracles are Practical: A Paradigm for Designing Efficient Protocols", *Proceedings of the 1st ACM Conference on Computer and Communications Security, CCS' 93*, pp. 62–73, 1993.

[27] M .Bellare, S. Keelveedhi, T. Ristenpart, "DupLESS: server-aided encryption for deduplicated storage", *USENIX Security Symposium' 13*, pp. 1–16, 2013.

[28] M .Bellare, H. Shi, C. Zhang,"Foundations of group signatures: The case of dynamic groups ", *CT-RSA'05, The Cryptographers' Track at the RSA Conference' 05*, Lecture Notes in Computer Science , vol. 3376, pp. 136–153, 2005.

[29] M. Bellare, P. Rogaway, "Exact Security of Digital Signatures- How to Sign with RSA and Rabin", *Advances in Cryptology ? Proceedings of EUROCRYPT' 96*, Lecture Notes in Computer Science, vol. 1070, pp. 399–416, 1996.

[30] M. Blaze, G. Bleumer, M. Strauss, "Divertible protocols and atomic proxy cryptography", *In EUROCRYPT' 98*, LNCS 1403, pages 127–144, 1998.

[31] V. Benjumea, S. Choi, J. Lopez, M. Yung, "Fair traceable multi-group signature", *12th International Conference Financial Cryptography and Data Security' 08*, Lecture Notes in Computer Science, vol. 5143 pp. 231–246, 2008.

[32] I. Bhattacharya, L. Getoor, “Deduplication and Group Detection using Links”, *ACM SIGKDD Workshop on Link Analysis and Group Detection, 2004.*

[33] G. R. Blakley, C. Meadows, “Security of ramp schemes”, *CRYPTO' 84*, pp. 242–268, 1984.

[34] A. Boldyreva, N. Chenette, Y. Lee, A. O'Neill, “Order-preserving symmetric encryption”, *Advances in Cryptology ? EUROCRYPT' 09*, Lecture Notes in Computer Science, vol. 5479, 224–241, 2009.

[35] D. Boneh, “The Decision Diffie-Hellman Problem, Algorithmic Number Theory Symposium”, *Proceedings of ANTS' 98*, Lecture Notes in Computer Science vol. 1423, pp. 48–63, 1998.

[36] D. Boneh, H. Shacham, “Group Signatures with verifier-local revocation”, *ACM Conference on Computer and Communications Security' 04*, pp. 168–177, 2004.

[37] D. Boneh, M. Franklin, “ Identity-based encryption from the Weil pairing”, *Advances in Cryptology - CRYPTO' 01*, Lecture Notes in Computer Science vol. 2139, 213–229, 2001.

[38] S. Bugiel, S. Nurnberger, A. Sadeghi, and T. Schneider, “Twin clouds: An architecture for secure cloud computing”, *Workshop Cryptography Security Clouds' 11*, pp. 32–44, 2011.

[39] Can you compress and dedupe it depends., “http://storagesavvy.richardanderson.net/2010/12/17/can-you-compress-and-dedupe-it-depends/”, 2017.

[40] R. Canetti, O. Goldreich, S. Halevi, “The Random Oracle Methodology, Revisited”, *The Computing Research Repository, 2000.*

[41] R. Canetti, and S. Hohenberger, “Chosen-ciphertext secure proxy re-encryption”, *In ACM conference on Computer and Communication Security*, pages 185–1946, 2007.

[42] S. Cavallar, B. Dodson, A. K. Lenstra, W. Lioen, P. L. Montgomery, B. Murphy, H. te Riele, K. Aardal, J. Gilchrist, G. Guillerm, P. Leyland, J. Marchand, F. Morain, A. Muffett, C. Putnam, C. Putnam, and P. Zimmermann, “Factorization of a 512-bit RSA Modulus, Advances in Cryptology”, *Proceedings of*

EUROCRYPT' 00, Lecture Notes in Computer Science, vol. 1807, pp. 1–18, 2000.

[43] D. Chaum, E. van Heyst, "Group Signatures", *EUROCRYPT' 91*, Lecture Notes in Computer Science, vol. 547, pp. 257–265, 1991.

[44] K. Chung, Y. Kalai, S. Vadhan, "Improved delegation of computation using fully homomorphic encryption", *Advances in Cryptology, CRYPTO' 10*, Lecture Notes in Computer Science, vol. 6223, pp. 483–501, 2010.

[45] R. Chen, Y. Mu, G. Yang, F. Guo, "OBC: A block-ciphertext mode of operation for efficient authenticated encryption", *Proceedings of the 8th ACM Conference on Computer and Communications Security, ACM CCS' 01*, pp. 196–205, 2001.

[46] R. Chen, Y. Mu, G. Yang, F. Guo, "BL-MLE: Block-Level Message-Locked Encryption for secure large file deduplication", *IEEE Transactions on Information Forensics and Security' 15*, vol. 10, pp. 2643–2652, 2015.

[47] E. M. Cho, T. Koshiba, "Secure Deduplication for Multiple Group setting", *SCIS' 16*, pp. 34–41, 2016.

[48] E. M. Cho, T. Koshiba, "Secure Deduplication in a Multiple Group Signature Setting", *The 31st IEEE International Conference on Advanced Information Networking and Applications, AINA' 17*, pp. 811–818, 2017.

[49] E. M. Cho, L. San, T. Koshiba, " Secure Non-Transferable Proxy Re-Encryption for Group Membership and Non-Membership", *The 8th International Workshop on Trustworthy Computing and Security/ TwCSec' 17*, 2017.

[50] R. Cramer and V. Shoup, "A Practical Public Key Cryptosystem Provably Secure against Adaptive Chosen Ciphertext Attack, Advances in Cryptology", *Proceedings of CRYPTO' 98*, Lecture Notes in Computer Science, vol. 1462, pp. 13–25, 1998.

[51] R. Curtmola, J. Garay, S. Kamara, R. Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient constructions", *Journal of Computer Security' 11*, vol. 19, pp. 895–934, 2011.

[52] Q. Dang, "Recommendation for applications using approved hash algorithms", *National Institute of Standards and Technology Special Publication' 08*, vol. 107, 2008.

[53] R. H. Deng, J. Weng, S. Liu and K. Chen, “Chosen-ciphertext secure proxy re-encryption without pairings”, *In Matthew K. Franklin, Lucas Chi Kwong Hui, and Duncan S. Wong, editors*, *CANS*, volume 5339 of *Lecture Notes in Computer Science*, Springer, pages 1–7, 2008.

[54] A. W. Dent, Y. Zheng, “Practical Signcryption. Information Security and Cryptography”, ISBN: 978-3-540-89409-4, *Springer* , 2010.

[55] D. Dolev, C. Dwork, M. Naor, “Non-malleable Cryptography”, *Annual ACM Symposium on Theory of Computing ? Proceedings of STOC' 91*, pp. 542–552, 1991.

[56] J. Douceur, A. Adya, W. Bolosky, D. Simon, and M. Theimer, “Reclaiming space from duplicate files in a serverless distributed file system”, *IEEE' 02*, pp. 617–624, 2002.

[57] Dropbox , “Dropbox http://www.dropbox.com”, 2017.

[58] Dropbox stolen data, “http://uk.businessinsider.com/dropbox-hack-68-million-accounts-passwords-stolen-employee-reused-2016-8”, 2016.

[59] T. ElGamal, “A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms”, *IEEE Transactions on Information Theory' 85*, vol. 31, pp. 469–472, 1985.

[60] EMC Corporation, “The digital universe decade- are you ready?”, 2010.

[61] EMC2 Cloud backup System, “https://www.emc.com/corporate/glossary/datd-deduplication.htm”.

[62] X. Fan, F. Liu, “Various Proxy Re-Encryption Schemes from Lattices”, *IACR Cryptology ePrint Archive 2016/278.*

[63] M. R. Felipe, K. M. M. Aung, Mediana, “HEDup: Secure Deduplication with Homomorphic Encryption”, *10th IEEE International Conference on Networking, Architecture and Storage, NAS' 15*, pp. 215–223, 2015.

[64] A. Fiat and A. Shamir, “How to Prove Yourself: Practical Solutions of Identification and Signature Problems”, *Proceedings of CRYPTO' 86*, Lecture Notes in Computer Science vol. 263, pp. 186–184, 1986.

[65] E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern, “RSA-OAEP is Secure under the RSA Assumption”, *Journal of Cryptology' 04*, vol.17, no. 2, pp. 81–104, 2004.

[66] C. Gamage, J. Leiwo, Y. Zheng, "An Efficient Scheme for Secure Message Transmission using Proxy-Signcryption", *Australasian Computer Science Conference, Auckland, New Zealand' 99*, pp. 420–431, 1999.

[67] C. Gamage, J. Leiwo, Y. Zheng. "Encrypted Message Authentication by Firewalls", *Public Key Cryptography PKC' 99*, Lecture Notes in Computer Science, vol. 1560, pp. 69–81, 1999.

[68] C. Gentry,' "A fully homomorphic encryption scheme", 2009.

[69] S. Goldwasser, S. Micali, R. Rivest, "A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks", *Society for Industrial and Applied Mathematics, SIAM Journal on Computing' 88*, vol. 17, no. 2, pp. 281–308, 1988.

[70] S. Goldwasser, S. Micali, " Probabilistic Encryption", *Journal of Computer and System Sciences' 84*, vol. 28, pp. 270–299, 1984.

[71] GoogleDrive," http://www.drive.google.com", 2017

[72] V. Goyal, O. Pandey, A. Sahai, B. Waters, " Attribute-based encryption for fine-grained access control of encrypted data", *Proceedings of the 13th ACM Conference on Computer and Communications Security' 06*, pp. 89–98, 2006.

[73] H. Guo, Z. Zhang, J. Xu, "Non-Transferable Proxy Re-Encryption", *In IACR Cryptology ePrint Archive 2015/1216.*

[74] L. Guo, L. Hu, "Efficient bidirectional proxy re-encryption with direct chosen-ciphertext security", *Computers and Mathematics with Applications 63(1)*, pages 151–157, 2012.

[75] S. Halevi, D. Harnik, B. Pinkas, A. Shulman-Peleg, "Proofs of ownership in remote storage systems", *ACM Conference on Computer and Communications Security' 11*, pp. 491–500, 2011.

[76] Y. He, T. W. Chim, L. C. K. Hui, and S. Yiu, "Non-Transferable Proxy Re-Encryption Scheme", *In NTMS 2012*, pages 1–4, 2012.

[77] A. Ivan and Y. Dodis, "Proxy cryptography revisited", *In proceeding of NDSS'03*, The Internet Society (2003).

[78] A. Juels, B. S. Kaliski Jr, "PORs: Proofs of retrievability for large files", *Proceedings of the 14th ACM Conference on Computer and Communications Security' 07*, pp. 584–597, 2007.

[79] D. Kwak, S. Moon, "Efficient Distributed Signcryption Scheme as Group Signcryption", *ACNS' 03*, Lecture Notes in Computer Science, vol. 2846, pp. 403–417, 2003.

[80] D. Kwak, S. Moon, G. Wang, R. Deng. "A secure extension of the Kwak-Moon group signcryption scheme", *ACNS' 03*, Lecture Notes in Computer Science, vol. 2846, pp. 435–444, 2003.

[81] J. Katz, A. Sahai, B. Waters, "Predicate encryption supporting disjunctions, polynomial equations, and inner products", *In: Advances in Cryptology EUROCRYPT' 08*, pp. 146–162, 2008.

[82] M. S .Kiraz, "Solving the Secure Storage Dilemma: An Efficient Scheme for Secure Deduplication with Privacy-Preserving Public Auditing ", *IACR Cryptology ePrint Archive' 16*, no. 696, 2016.

[83] K. Ko, S. Lee, J. Cheon, J. Han, J. Kang, and C. Park, "New Public-Key Cryptosystem Using Braid Groups, Advances in Cryptology", *Proceedings of CRYPTO 2000* , Lecture Notes in Computer Science, vol. 1880, pp 166–183.

[84] N. Koblitz, "Elliptic Curve Cryptosystems", *Mathematics of Computation' 87*, vol. 48, no. 177, pp 203–209, 1987.

[85] I. Leontiadis, K. Elkhyaoui, R. Molva, "Private and dynamic time-series data aggregation with trust relaxation", *Cryptology and Network Security' 14*, Lecture Notes in Computer Science vol. 8813, pp. 305–320, 2014.

[86] J. Li, X. Chen, M. Li, J. Li, P.P. Lee, W. Lou, "Secure deduplication with efficient and reliable convergent key management", *Parallel and Distributed Systems' 14*, no. 6, pp. 1615–1625, 2014.

[87] J. Li, K. L. Yan, X. Chen, P. P. Lee, and W. Lou. "A hybrid cloud approach for secure authorized deduplication." *Parallel and Distributed Systems' 15* , no. 5, pp. 1206–1216, 2015.

[88] J. Li, X. Chen, X. Huang, S. Tang, Y. Xiang, M. M. Hassan, A. Alelaiwi, "Secure Distributed Deduplication Systems with Improved Reliability", *IEEE Transactions on Computers' 15* vol. 64(12), pp. 3569–3579, 2015.

[89] B. Libert and D. Vergnaud, "Tracing malicious proxies in proxy re-encryption", *In Pairing-Based Cryptography-Pairing 2008*, pages 332–353, 2008.

M

[90] C. Ma, and J. Ao, “Group-Based Proxy Re-encryption Scheme”, *ICIC (1) 2009*, pages 1025–1034, 2009.

[91] M. Mambo and E. Okamoto, “Proxy cryptosystems: Delegation of the power to decrypt ciphertexts”, *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol.E80A, no.1, pages 54–63, 1997.

[92] W. Mao, “Modern Cryptography: Theory and Practice”, Prentice Hall, 2004.

[93] T. Matsuda, R. Nishimaki and K. Tanaka, “CCA proxy re-encryption without bilinear maps in the standard model”, *PKC 2010*, volume 6056 of *Lecture Notes in Computer Science*, Springer, pages 261–278, 2010.

[94] T. Matsuo, “Proxy re-encryption systems for identity-based encryption”, *In Pairing-Based Cryptography-Pairing 2007*, pages 247–267, 2007.

[95] U. Maurer, “Towards Proving the Equivalence of Breaking the Diffie-Hellman Protocol and Computing Discrete Logarithms”, *Advances in Cryptology ?Proceedings of CRYPTO’ 94*, Lecture Notes in Computer Science, vol. 839, pp. 271–281, 1994.

[96] P. Mell, T. Grance’, “The NIST definition of cloud computing. Technical report, National Institute of Standards and Technology, Information Technology Laboratory, 2009”

[97] V. Miller, “Use of Elliptic Curves in Cryptography, Advances in Cryptology”, *Proceedings of CRYPTO’ 85*, Lecture Notes in Computer Science, vol. 218, pp. 417–429, 1985.

[98] Mozy, “https://mozy.com/”, 2017

[99] M. Naor, M. Yung, “Public-Key Cryptosystems Provably Secure against chosen ciphertext Attacks”, *Annual ACM Symposium on Theory of Computing ? Proceedings of STOC’ 90*, pp. 427–437, 1990.

[100] National Institute of Standards and Technology, “Digital Signature Standards”, *U.S. Department of Commerce*, NIST FIPS PUB 186, May 1994.

[101] National Institute of Standards and Technology, “Secure Hash Standard”, *U.S. Department of Commerce*, NIST FIPS PUB 180-1, April 1995.

[102] T .Nakanishi, H. Fujii, Y. Hira, N. Funabiki, “Revocable Group Signature Schemes with Constant Costs for Signing an Verifying”, *IEICE Transactions’ 10*, vol. 93-A(1), pp. 50–62, 2010.

[103] K. Niu, X. A. Wang, and M. Q. Zhang, "How to solve key escrow problem in proxy re-encryption from cbe to ibe", *In DBTA*, pages 95–98, 2009.

[104] D.Nuez, I. Agudo, J. Lopez, "A Parametric Family of Attack Models for Proxy Re-Encryption", *IEEE 28th Computer Security Foundations Symposium, CSF 2015*, pages 290–301, 2015.

[105] T. Okamoto and S. Uchiyama, "A New Public-Key Cryptosystemas as Secure as Factoring, Advances in Cryptology", *Proceedings of EUROCRYPT' 98*, Lecture Notes in Computer Science, vol. 1403, pp. 308–318, 1998.

[106] P. Paillier, "Public Key Cryptosystems Based on Composite Degree Rediduosity Classes, Advances in Cryptology", *Proceedings of EUROCRYPT' 99*, Lecture Notes in Computer Science, vol. 1592, pp. 223–238, 1999.

[107] B. Parno, M. Raykova, V. Vaikuntanathan, " How to delegate and verify in public: verifiable computation from attribute-based encryption", *Theory of Cryptography' 12*, Lecture Notes in Computer Science, vol. 7194, pp. 422–439, 2012.

[108] B. Parno, J. Howell, C. Gentry, M. Raykova, "Pinocchio: nearly practical verifiable computation", *In: IEEE Symposium on Security and Privacy' 13*, pp. 238–252, 2013.

[109] B. Pinkas, T. Reinman, "Oblivious RAM revisited" *Advances in Cryptology CRYPTO' 10*, Lecture Notes in Computer Science, vol. 6223, pp. 502–519, 2010.

[110] L. Phong, L Wang, Y. Aono, M. Nguyen, X. Boyen, "Proxy Re-Encryption Schemes with Key Privacy from LWE", *IACR Cryptology ePrint Archive 2016/327*.

[111] D. Pointcheval, "New Public Key Cryptosystems Based on the DependentRSA Problems, Advances in Cryptology", *Proceedings of EUROCRYPT' 99*, Lecture Notes in Computer Science vol. 1592, pp. 239–254, 1999.

[112] P. Puzio, R. Molva, M. Onen, S. Loureiro, "Cloudedup: secure deduplication with encrypted data for cloud storage", *5th International Conference on Cloud Computing Technology and Science' 13*, pp. 363–370, 2013.

[113] P. Puzio, R. Molva, M. Onen, S. Loureiro, "PerfectDedup: secure data deduplication", *Data Privacy Management, and Security Assurance,*

DPM/QASA@ESORICS' 15, Lecture Notes in Computer Science, vol. 9481, pp. 150–166, 2015.

[114] R. D. Pietro, A. Sorniotti, "Boosting efficiency and security in proof of ownership for deduplication", *ASIACCS' 12*, pp. 81–82, 2012.

[115] M. O. Rabin, "Digitalized Signatures and Public-Key Functions as Intractable as Factorization," MIT/LCS/TR-212, MIT Laboratory for Computer Science, 1979.

[116] C. Rackoff, D. Simon, "Non interactive Zero-Knowledge Proof of Knowledge and chosen ciphertext Attack", *Advances in Cryptology - Proceedings of CRYPTO' 91*, Lecture Notes in Computer Science, vol. 576, pp.434–444, 1991.

[117] R. L. Rivest. "The MD5 Message-Digest Algorithm, Internet Repost", *RFC' 92* vol. 1321, 1992.

[118] R. L. Rivest, Adi Shamir, and Leonard M. Adleman. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", *Communications of the ACM' 78*, vol. 21 (2), pp. 120–126, 1978.

[119] S. Rass, D. Slamanig, *Cryptography for Security and Privacy in Cloud Computing' 14*, Artech House, 2014.

[120] A. D. Santis, B. Masucci, "Multiple Ramp Schemes", *IEEE Trans. Information Theory' 99*, vol. 45(5), pp. 1720–1728, 1999.

[121] C. P. Schnorr, "Efficient Identification and Signatures for Smart Cards, Advances in Cryptology", *Proceedings of CRYPTO' 89*, Lecture Notes in Computer Science, vol. 435, pp. 235–251, 1989.

[122] V. Shoup, "OAEP Reconsidered, Advances in Cryptology", *Proceedings of CRYPTO' 01*, Lecture Notes in Computer Science vol. 2139, pp. 239–259, 2001.

[123] M. Sipser, "Introduction to the Theory of Computation", *PWS Publishing Company, 1997.*

[124] A. Shamir, "How to share a secret", *Commun. ACM' 97* , vol. 22, no. 11, pp. 612–613, 1997.

[125] Y .Shin, J. Hur, K. Kim, "Security weakness in the Proof of Storage with Deduplication", *IACR Cryptology ePrint Archive' 12*, vol. 554, pp. 886–900, 2012.

[126] Spideroak, “https://spideroak.com/”, 2017

[127] J. Stanek, A. Sorniotti, E. Androulaki, and L. Kencl, “A secure data deduplication scheme for cloud storage”, *Financial Cryptography' 14*, pp. 99–118, 2014.

[128] M. W .Storer, K. Greenan, D. D. Long, and E. L. Miller, “Secure data deduplication”, *4th ACM international workshop on Storage Security and Survivability*, pp. 1–10, October 2008.

[129] SVN, “https://subversion.apache.org/”, 2017

[130] Y. Tsiounis and M. Yung, “On the Security of ElGamal-Based Encryption, Public Key Cryptography”, *Proceedings of PKC' 98*, Lecture Notes in Computer Science vol. 1431, pp 117–134, 1998.

[131] B. Waters, “Ciphertext-Policy Attribute-Based Encryption: An Expressive, Efficient, and Provably Secure Realization”, *Public Key Cryptography PKC' 11*, Lecture Notes in Computer Science, vol. 6571, pp. 53–70, 2011.

[132] C. Wang, N. Cao, J. Li, K. Ren, W. Lou, “Secure ranked keyword search over encrypted cloud data”, *Proceedings - International Conference on Distributed Computing Systems' 10*, pp. 253–262, 2010.

[133] C. Wang, Q. Wang, K. Ren, W. Lou, “Privacy-preserving public auditing for data storage security in cloud computing”, *In: INFOCOM ,2010.*

[134] L. Wang, L. Wang, M. Mambo, and E. Okamoto, “New identity- based proxy re-encryption schemes to prevent collusion attacks”, *In Pairing 2010*, pages 327–346. Springer, 2010.

[135] X. A. Wang and X. Y. Yang, “Identity based broadcast encryption based on one to many identity based proxy re-encryption”, *In IEEE*, pages 47–50, 2009.

[136] X. A. Wang, J. Ma and X. Y. Yang, “A new proxy re-encryption scheme for protecting critical information systems”, *In Journal of Ambient Intelligence and Humanized Computing 6(6)* , pages 699–711, 2015.

[137] X. A.Wang, J. Ma, F. Xhafa, M. Zhang and X. Luo, “Cost-effective secure E-health cloud system using identity based cryptographic techniques”, *In Journal of Future Generation Computer Systems*, vol 67, pages 242–254, 2017.

[138] J .Xu, E. Chang, J. Zhou, “Leakage-Resilient Client - side Deduplication of Encrypted Data in Cloud Storage”, *ASIACCS' 13*, pp. 195–206, 2013.

[139] L. Xu, X. Wu, and X. Zhang, "A certificateless proxy re-encryption scheme for secure data sharing with public cloud", *ASIACCS 2012*, pages 87–88, 2012.

[140] C. Yang, J. Ma, J. Ren, "Provable Ownership of Encrypted Files in De-Duplication Cloud Storage", *Ad Hoc & Sensor Wireless Networks' 15*, vol. 26, no. 1–4, pp. 43–72, 2015.

[141] J .Yuan, S. Yu, "Secure and constant cost public cloud storage auditing with deduplication", *IEEE-CNS' 13*, pp. 145–153, 2013.

[142] Y. Zheng, X. Yuan, X. Wang, J. Jiang, C. Wang, X. Gui, "Enabling encrypted cloud media center with secure deduplication", *ASIACCS' 15*, pp. 63–72, April 2015.

[143] Q. Zheng, S. Xu, "Secure and Efficient Proof of Storage with Deduplication", *CODASPY' 12*, 2012.

[144] Y. Zheng, H. Imai, "How to construct efficient signcryption schemes on elliptic curves", *Information Processing Letters*, vol.68, pp. 227–233, 1998.

[145] Y. Zheng. "Digital Signcryption or how to achieve cost (signature & encryption) $\ll$ cost(signature) + cost (encryption)", *CRYPTO' 97*, Lecture Notes in Computer Science, vol.1294, pp. 165–179, 1997.

[146] Y. Zheng. "Signcryption and its application in efficient public key solutions", *Information Security Workshop ISW' 97*, Lecture Notes in Computer Science, vol.1396, pp. 291–312, 1997.

[147] Y. Zhou, D. Feng, W. Xia, M. Fu, F. Huang, Y. Zhang, C. Li, "SecDep: a user-aware efficient fine-grained secure deduplication scheme with multi-level key management", *31st Symposium on Mass Storage Systems and Technologies, MSST' 15*, pp. 1–14, 2015.

[148] S. Zhou, D. Lin, "Unlinkable Randomizable Signature and Its Application in Group Signature", *Inscrypt' 07*, pp. 328–342, 2007.