VIETNAM NATIONAL UNIVERSITY OF HOCHIMINH CITY
THE INTERNATIONAL UNIVERSITY
SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

DATA PRIVACY IN CLOUD COMPUTING

By
PHẠM HÀ MINH THY – ITITIU19056

A thesis submitted to the School of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
Bachelor of Information Technology

Ho Chi Minh City, Vietnam
2023 – 2024

DATA PRIVACY IN CLOUD COMPUTING

APPROVED BY:

_____ ,

Le Hai Duong, Dr

_____

_____

_____

_____

THESIS COMMITTEE

(Whichever applies)

ACKNOWLEDGMENTS

I would like to express my deep appreciation and gratitude to Dr. Le Hai Duong for his professional guidance. His constant encouragement and support were instrumental in helping me achieve my goal.

I am also grateful to all the members at the School of Computer Science and Engineering, whose extensive knowledge and support have been invaluable to me. Their technical assistance and positive attitude have made my years of study a truly remarkable experience. I am thankful to the faculty of the School of Computer Science of the International University for providing me with the opportunity to finish my research. I would like to give thanks to the lecturers whom I had the opportunity to interact with and acquired the necessary knowledge for this thesis. Additionally, gratitude is also given to the members of my reading and examination committee. Lastly, I would like to extend my heartfelt thanks to my family and friends. Without their love and encouragement, this thesis would not be completed.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Since its inception until now, cloud computing has brought convenience and flexibility to us in terms of sharing data. However, there is resistance for users to share their data over the cloud, especially data containing sensitive information. This paper presents a cloud computing encryption system that aims to enhance data security and privacy in cloud environments. The recommended system is a strategy to add a layer of security to the cloud platform. With the increasing reliance on cloud storage and services, it is crucial to ensure that sensitive data remains protected from unauthorized access. The proposed encryption system provides a robust solution to this challenge. This system utilizes the methods of Revocable-Storage Identity-Based Encryption (RS-IBE) and Advanced Encryption Standard (AES) for encryption and decryption. Firstly, a random message is hashed to generate an encryption key. The encryption key will then be applied to the AES to encrypt a file. To protect the key from exposure, a key encapsulation method is carried out. The approach is to apply RS-IBE scheme into encrypting and decrypting the cryptography key. The system uses a tree-based structure to manage the user's identity and time. The depth of the tree and the total of nodes can be adjusted according to specific requirements, providing flexibility in user and data management. This allows the system to be optimized based on their storage needs and security preferences. Besides, with the ability to revoke users at time periods and continuously update the ciphertext, this scheme appears to meet the security requirements of a data-sharing system.

CHAPTER 1

INTRODUCTION

1.1. Background

As society becomes more modernized, the amount of digital data has significantly increased, which yields a growing demand for efficient storage and data sharing. Traditional on-premises storage solutions have become increasingly inadequate and will not be able to handle such a huge amount of data. People are facing the challenge of managing and accessing their data effectively, especially when it comes to scalability, accessibility, and cost-efficiency. This has led to the appearance of cloud computing. Several major cloud providers provide cloud storage services can be named as Google Cloud Provider, Microsoft Azure, Apple iCloud, Amazon S3, Cloudflare R2, etc.

Storage is one of the essential components of cloud computing. Data stored in the cloud can be managed and accessed remotely, providing scalability, flexibility, and cost-efficiency. Another advantage of cloud storage is the high availability of data. As data is replicated across multiple data centers, even in the case of disasters, it will remain protected and accessible. However, security is a critical consideration in cloud storage that prevents many users from approaching this advanced technology. Cloud service providers have been implementing robust security measures, such as encryption, access controls, and data isolation, to protect data from unauthorized access and ensure data privacy. However, as vulnerability always exists in a system, more security measures need to be proposed to protect users from security threats.

1.2. Problem Statement

Data privacy is a critical concern in cloud computing. As cloud storage services are being used more and more, the need to keep sensitive information secured and protected is crucial. Hence, this thesis focuses on three main problems that users may encounter while experiencing cloud computing including (1) protecting data when outsourcing to the cloud, (2) controlling access, and (3) encrypting process efficiency.

Protecting data when outsourcing to the cloud is the first common problem for users. This is also one of the main challenges in cloud computing. Users need to have confidence that their sensitive information is protected from unauthorized access, data breaches, and potential loss. Therefore, finding effective methods and techniques to safeguard data in the cloud is crucial for maintaining trust and confidence in cloud computing services.

Controlling access to cloud resources is another significant concern in cloud computing. Users need to have granular control over who can access their data. Ensuring proper authentication and authorization mechanisms, as well as implementing robust access control policies, are key factors in maintaining the confidentiality and integrity of shared data.

Another problem is the *efficiency of the encrypting process*. While encryption is an essential security measure for protecting sensitive data, it can also introduce challenges in terms of process efficiency. The computational overhead of encryption algorithms and the potential impact on system performance need to be carefully considered. Finding the right balance between effective encryption methods and efficient processing is crucial to ensure data security without compromising system speed and overall performance.

1.3. Scope and Objectives

Having a clear view of objectives is a necessary step to determine the scope of the design and implementation of the system. The objectives are outlined in the below table:

Table 1.1. Thesis Objectives

Objective	Content
1	Building a secure data sharing system
2	Implementing the RS-IBE scheme for key encryption
3	Encrypting files using AES
4	Suggesting a modification in the ciphertext update definition of Wei et al.

[Obj. 1] *Building a secure data sharing system*: The objective is to develop a system that ensures the secure sharing of data among authorized users. This includes implementing access controls, authentication mechanisms, and encryption to protect the confidentiality and integrity of the shared data.

[Obj. 2] *Implementing the RS-IBE scheme for key encryption*: The objective is to incorporate the RS-IBE (Revocable Identity-Based Encryption) scheme into the data sharing system. This encryption scheme enables the use of user identities as public keys, allowing for efficient and flexible key management, as well as the revocation of access if necessary.

[Obj. 3] *Encrypting files using AES*: The objective is to utilize the Advanced Encryption Standard (AES) to protect the files stored and shared within the system. AES is a widely adopted symmetric encryption algorithm known for its strong security and efficiency. Therefore, it is a good choice to ensure the integrity and authenticity of the files, making it difficult for unauthorized modifications to go undetected.

[Obj. 4] *Suggesting a modification in the ciphertext update definition of Wei et al.*: The objective is to prove that the formula to determine the updated ciphertext of Wei et al. is wrong, as well as to propose a modification to the definition.

These objectives are aimed at achieving a secure and efficient data sharing system by implementing various encryption techniques such as the RS-IBE scheme and AES encryption for file protection.

1.4. Assumption and Solution

In the scope of this thesis, a file sharing system with identity-based cryptography will be developed to overcome the problems stated in Section 1.2 and achieve the objectives in Section 1.3. The mechanism of this system is to enable users to perform encryption of the shared data and upload the ciphertext to the cloud. The data provider takes the identity of the receiver to encrypt the file. The data receiver provides his/her identity to generate a secret key. The secret key will then be used to generate a decryption key. Any users who want to access the data can use their identities to authenticate, download, and decrypt the ciphertext. By using the data receiver identity to encrypt files, the data providers have the ability to define who can access his/her data. Unauthorized users will not be able to gain access to or decrypt the shared data. The system also has a revocation list indicating users that are revoked at different time periods. Once a user is revoked, he/she can still download the file, but he/she will no longer be able to decrypt the ciphertext since the decryption cannot be computed.

1.5. Structure of thesis

The thesis is broken down into six chapters, each of which describes the topic and offers several points of view.

Chapter 1: Introduction

Chapter 2: Literature Review and Related Work

Chapter 3: Methodology

Chapter 4: Implementation and Results

Chapter 5: Discussion and Evaluation

Chapter 6: Conclusion and Future Work

CHAPTER 2

LITERATURE REVIEW/RELATED WORK

This chapter provides a brief synopsis of the development and history of the data encryption system, which is the subject of this thesis. A summary of related projects will also be given so that a fuller understanding of the issue can be obtained.

2.1. Background of Security

Designing a data sharing system requires basic knowledge about cryptography and how the encrypt-decrypt process works.

2.1.1. Cybersecurity and Information Security

Cyber Security and Information Security are both related to the action of defending computer systems and data against malicious users and threats. The two terms are often used interchangeably; however, their definition and concepts are quite different. Cyber security is the practice of protecting stored, transmitted, and processed information in network systems, digital devices, network devices, and transmission lines. On the other hand, information security is about ensuring the confidentiality, integrity, and availability of data by protecting it from unauthorized access and data modification. This thesis focuses more on information security.

2.1.2. The CIA Triad

The CIA triad, which is widely recognized as the foundation of an effective information security plan, encompasses three fundamental principles:

- **Confidentiality**: This principle revolves around the preservation of authorized restrictions, ensuring that only authorized users are granted access to sensitive data. By maintaining confidentiality, organizations can prevent unauthorized individuals from gaining access to valuable information.
- **Integrity**: The principle of integrity aims to safeguard data from improper modifications and destruction. It emphasizes the importance of maintaining data accuracy and preventing unauthorized alterations. By ensuring the integrity of data, organizations can trust the reliability and trustworthiness of their information.

- **Availability**: The principle of availability focuses on ensuring that data is readily accessible and usable when needed. It encompasses timely and reliable access to data, allowing authorized users to retrieve and utilize information efficiently.

In the context of this thesis, the primary focus is on achieving the goal of confidentiality. However, it is important to acknowledge that other critical security aspects, such as integrity and availability, are also significant components of a comprehensive information security framework. While not explicitly covered within the scope of this thesis, these aspects should be considered and addressed to ensure a holistic approach to information security.

2.1.3. Cryptography and Encryption

Cryptography is the technique of hiding or securing information, ensuring only the person for whom the message is intended can read it. The system uses a set of rule-based calculations and algorithms to transform plain text into a ciphertext.

Encryption is a type of cryptography that is used to protect data from being read or stolen. Encryption works by using a key to encrypt the readable data into an alternative form known as ciphertext, which is incomprehensible. The intended receiver can decipher the ciphertext back to plain text using the proper decryption key. The security level of encryption depends significantly on the complexity of the cryptography key.

The proposed system uses the SHA-256 encryption algorithm and Advanced Encryption Standard (AES) for file encrypting.

SHA-256 is one of the strongest hash functions developed by the United States National Security Agency (NSA) and published as the U.S. federal standard by the National Institute of Standards and Technology (NIST). Due to the high probability of difference, SHA is used to convert a piece of data into an almost-unique constant length signature. This is a one-way process as when a text is hashed, it cannot be reverted to its original form.

AES is a symmetric block cipher developed by the United States National Institute of Standards and Technology (NIST) to effectively defend sensitive data against with Brute force attack. Symmetric block cipher uses the same key for encryption and decryption. An initial vector (IV) is added to the key to expand the key space making it more challenging to be cracked by brute force.

2.1.4. Bilinear Pairing

A bilinear pairing is a mapping between two cyclic groups, G1 and G2, with a prime order q. It is denoted as e: $G1 \times G1 \rightarrow G2$ and possesses the following properties:

- *Bilinearity*: The pairing is bilinear, meaning that for any u, h in G_1 and any a, b in Z_q^*, the equation $e(u^a, h^b) = e(u, h)^{ab}$ holds.
- *Non-degeneracy*: The pairing is non-degenerate, implying that $e(g, g) \neq 1$.
- *Computability*: There exists an efficient algorithm to compute the pairing $e(u, h)$ for any u, h in G1.

2.2. Related Works

2.2.1. Identity-Based Encryption

When using public-key cryptography, it is required to have certificates for every public key. This can be followed by many problems and difficulties in managing certificates. Shamir, therefore, introduced the method of identity-based cryptography in 1985 as a solution to those issues [1]. With identity-based cryptography, the recipient's identity will be utilized for encryption in place of a public key. A private key used for decryption would be generated by the key authority from the recipient's identity.

In the field of cryptography, the issue of key revocation in Identity-Based Encryption (IBE) has been a long-standing challenge. Initially, Boneh and Franklin proposed the first practical IBE scheme with key revocation [2]. This method involved associating a time period with the ciphertext and regularly sending private keys for each time period to non-revoked users. However, as the use of IBE expanded and more users were added to the system, it became evident that this revocation method was not scalable. Furthermore, the need for a secure channel for the key authority and non-revoked users to exchange new keys added another layer of complexity to the process.

2.2.2. Revocable-Storage Identity-Based Encryption

To overcome these limitations, a new concept called Revocable-Storage Identity-Based Encryption (RS-IBE) was introduced by Jianghong Wei, Wenfen Liu, and Xuexian Hu in 2018 [3]. The fundamental concept behind this RS-IBE scheme is to extend the functionality of an IBE scheme by enabling ciphertext updates and key revocation.

By introducing the concept of revocable-storage, the RS-IBE scheme addresses the scalability issues faced by the traditional revocation method. Instead of relying on a large number of pre-generated private keys, the RS-IBE scheme dynamically manages key revocation and updates, optimizing resource utilization and reducing the storage requirements. This scalability makes the RS-IBE scheme suitable for large-scale deployments and enables efficient management of access control in IBE systems.

Furthermore, the RS-IBE scheme also aims to achieve the goals of data confidentiality, backward secrecy, and forward secrecy. Data confidentiality ensures that the encrypted data remains secure and inaccessible to unauthorized users. Backward secrecy ensures that even if a user's key is compromised or revoked, the previously encrypted data remains confidential. Forward secrecy ensures that even if a user's key is compromised in the future, the previously encrypted data remains secure. These goals collectively enhance the overall security and privacy of the system, making the RS-IBE scheme a valuable tool in cryptographic applications.

The RS-IBE scheme proposed by Jianghong Wei, Wenfen Liu, and Xuexian Hu offers a more scalable and efficient solution for key revocation in Identity-Based Encryption. This scheme also represents a significant advancement in the field of cryptography, offering a valuable tool for secure data encryption and access control in various applications.

Security model of RS-IBE scheme

The RS-IBE scheme uses a tree-based structure for user and time management. To ensure secure and efficient data encryption and decryption, a series of well-defined steps are operated.

Firstly, during the system setup phase, the Setup and FuncDef functions are executed to initialize the necessary public parameters and functions. This step establishes the foundation for the subsequent processes.

Next, secret keys, which are unique for each user, are generated using the PKGen function. These secret keys play a crucial role in the encryption and decryption processes, ensuring that only authorized users can access and decrypt the encrypted data. The generated secret keys are then distributed to the respective users, securely enabling them to perform decryption operations.

To encrypt the file, the system utilizes the Encrypt function, which takes the ID of the intended data receiver as input. This ID ensures that the encrypted file can only be decrypted by the authorized recipient. The encryption process guarantees that the sensitive data remains confidential and protected from unauthorized access.

When a user requests the file at a specific time, the KeyUpdate and CTUpdate functions generate and distribute updated ciphertext and keys to the users, ensuring that they have access to the most recent versions of the encrypted data.

In the event of a user revocation at a particular time, the system employs the Revoke function to update the Revocation List (RL). This function ensures that the revoked user's access is terminated, preventing any unauthorized decryption attempts.

On the user's side, the DKGen function generates a decryption key from the provided values of the secret key and key updates. This decryption key serves as the means to decrypt the ciphertext and access the original file. The Decrypt function is then employed, utilizing the decryption key to reverse the encryption process and reveal the plaintext data.

The main functions of Wei's RS-IBE scheme are defined as follows:

- $\boldsymbol{Setup}(1^{\lambda}, T, N)$: The setup function takes in the total amount of time and the maximum number of users to compute the public parameters, the master secret key, and the F_u and F_h functions for users and time. It also initializes the revocation list $RL = \emptyset$.
- $\boldsymbol{PKGen}(PP, MSK, ID)$: The private key generation function takes in the previously defined public parameters, master secret key, and user's ID to generate the secret key for each user.
- $\boldsymbol{KeyUpdate}(PP, MSK, RL, t, st)$: The key update function takes in the previously defined public parameters, master secret key, the current revocation list and time to compute the key update at a new time.
- $\boldsymbol{DKGen}(PP, SK_{ID}, KU_t)$: The decryption key generation takes in the previously defined public parameters, secret key, and key update to generate the decryption key. If a user is revoked, the function returns $\perp$.
- $\boldsymbol{Encrypt}(PP, ID, t, M)$: The encryption function takes in the previously defined public parameters, a user's ID, time, and the to-be-encrypted message to compute the ciphertext.
- $\boldsymbol{CTUpdate}(PP, CT_{ID,t}, t')$: The ciphertext update function takes in the previously defined public parameters, the ciphertext, and a new time to update the ciphertext corresponding to the new time.
- $\boldsymbol{Decrypt}(PP, CT_{ID,t}, DK_{ID,t'})$: The decryption function takes in the previously defined public parameters, the updated ciphertext and the updated decryption key to retrieve the message. If the ciphertext is invalid, the function returns $\perp$.
- $\boldsymbol{Revoke}(PP, ID, RL, t, st)$: The revoke function takes in the previously defined public parameters, a user's ID to be revoked, the revocation list, and time to add the user's ID to the revocation list and update the list to the new one.

Besides the main functions, Wei, Liu, and Hu also introduced a function called KUNodes to find the non-revoked children of revoked nodes.

- $\boldsymbol{KUNodes}(BT, RL, t)$: The KUNodes function takes in the binary tree, a revocation list, and the time when the user is revoked. First, all nodes in the path to the revoked nodes are marked as revoked. Then, the algorithm outputs a set Y containing the non-revoked children of the revoked nodes.

2.2.3. A Critical Analysis of Wei's Paper

In 2020, Kwangsu Lee published a paper titled "Comments on "Secure Data Sharing in Cloud Computing Using Revocable-Storage Identity-Based Encryption"" in which he provided a comprehensive review of Wei's work [4]. Lee's analysis focused on identifying certain errors and inconsistencies in Wei's paper, particularly with regards to the definition and implementation of the CTNodes, an algorithm to find nodes in T. According to Lee, Wei's formula allows for access to a time period in the past due to the inclusion of left nodes. Lee expounded on this issue and provided a detailed explanation of the implications and potential drawbacks of such a definition.

Wei et al. has defined the CTNodes function as for each leaf node v_t of binary tree T,

$$\boldsymbol{CTNodes}(T_t) = \{v \mid \boldsymbol{Parent}(v) \in \boldsymbol{Path}(v_t), v \notin \boldsymbol{Path}(v_t)\} \cup \{v_t\}$$

This means that T_t contains all the nodes that do not belong to the path to leaf node v_t, but whose parent does.

For instance, let the leaf node be v_{t_2}. The set defined by Wei's CTNodes would be:

$$Path(v_{011}) = \{0,01,011\}$$

$$CTNodes(v_{011}) = \{010,011,00,1\}$$

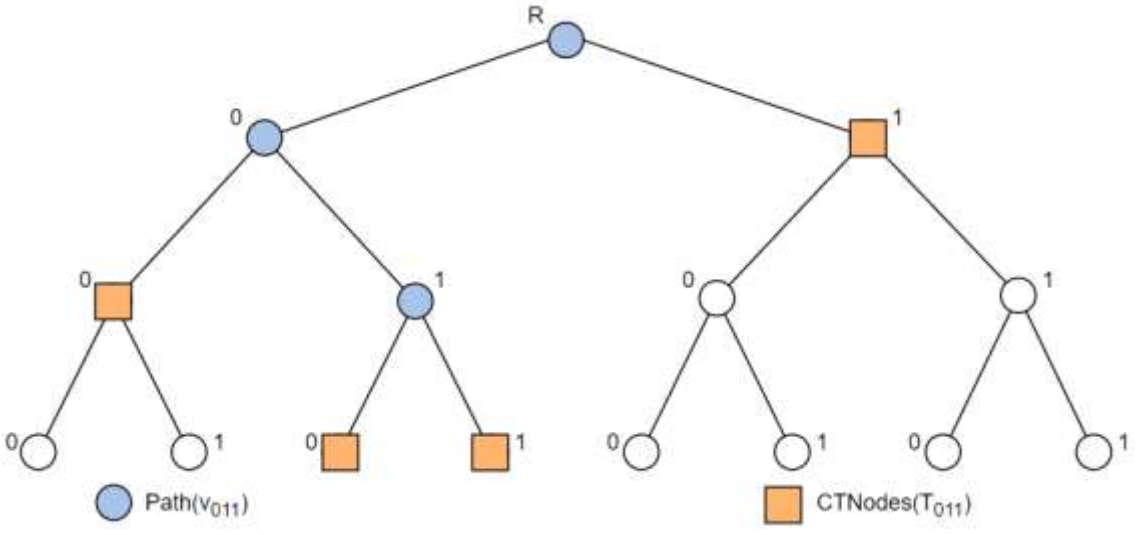

Figure 2.1. Example of Wei's CTNodes

The CTNodes contains nodes 00 and 010, which are on the left side of node 011. This set is not correct since these are the time periods in the past.

Furthermore, in his paper, Lee not only identified the flaws in Wei's work but also proposed an improved definition for CTNodes as below.

$$\boldsymbol{CTNodes} = \boldsymbol{RightSibling}(\boldsymbol{Path}(v_t)) \setminus \boldsymbol{Path}(\boldsymbol{Parent}(v_t)) \cup \{v_t\}$$

where $RightSibling(S)$ is a set of $RightChildren(Parent(v))$.

Continue with the previous example, let the leaf node be v_{t_2}, the set defined by Lee's CTNodes would be:

$$Path(v_{011}) = \{0,01,011\}$$

$$CTNodes(v_{011}) = \{011,1\}$$

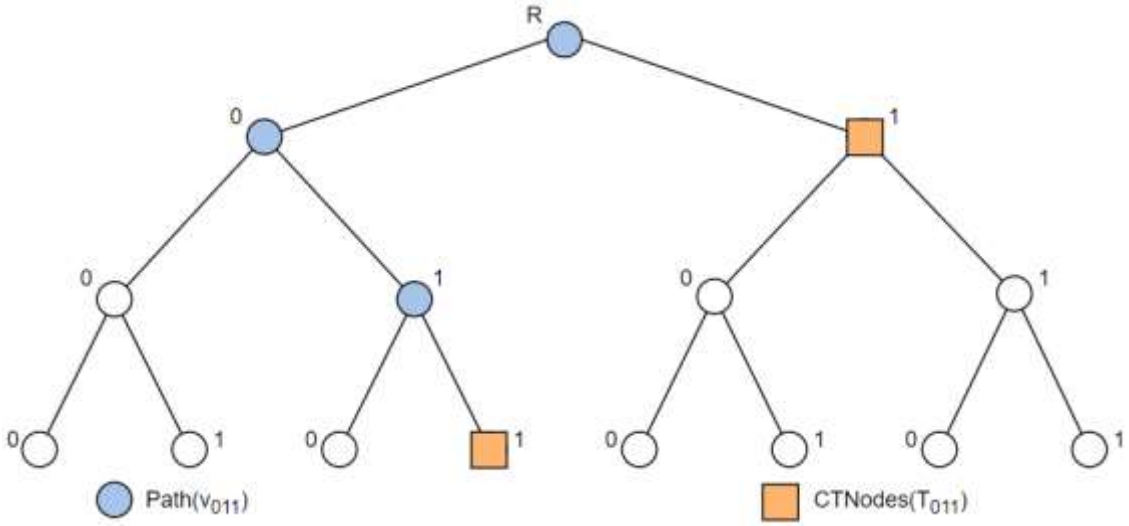

Figure 2.2. Example of Lee's CTNodes

The CTNodes contains node 1, which is on the right side of node 011. This set is correct since the time period is in the future.

By revising the existing RS-IBE scheme, Lee aimed to address the shortcomings and enhance the overall functionality and security of the system. His proposed definition sought to mitigate the potential risks associated with accessing past time periods while maintaining the integrity and effectiveness of the RS-IBE scheme. Kwangsu Lee's paper has served as a valuable contribution to the field, shedding light on the mistakes made in Wei's work and offering a new perspective and solution to enhance the RS-IBE scheme.

CHAPTER 3

METHODOLOGY

3.1. Overview

As mentioned before, the proposed system is based on the RS-IBE method of Wei et al. However, modifications have been made to achieve the goals of efficiency and accuracy. The previous articles only show theoretical calculations and do not have specific implementation. Therefore, in this thesis, I aim to implement this system in the most complete way as possible.

In the proposed system, the RS-IBE scheme is only applied for key encryption and management, so performance is much improved. If RS-IBE was directly applied to encrypt files, the data provider would regularly need to perform the tasks of downloading, decrypting, and uploading the ciphertext to the server. This could result in a considerable increase in computing costs for large data, which would reduce efficiency.

Another specialty of the proposed system is the innovative approach to binary tree implementation. Unlike the method of Wei et al., which requires building and storing all nodes in the binary tree, the proposed system eliminates the need for such extensive calculations and storage. Instead, it employs an on-the-fly calculation approach, where only the necessary nodes for computation are calculated. This means that the proposed system is able to dynamically generate and calculate the required nodes as needed, without the need for pre-built binary trees or storing unnecessary data.

3.2. System Design

The system design is similar to other data sharing systems. As indicated in Figure 3.1, the data provider encrypts data using the identity of the data receiver and uploads the ciphertext of the data to the cloud server. At the time when the data receiver requests for download, the system updates the cyphertext, and the data receiver calculates the decryption key at that specific time. Then, he can download the updated ciphertext from the cloud server and use his decryption key to decrypt it. Keys are managed by the key manager. When the data receiver is revoked, meaning his authorization has expired, he still can download the ciphertext; however, he cannot generate a decryption key to decrypt that ciphertext.

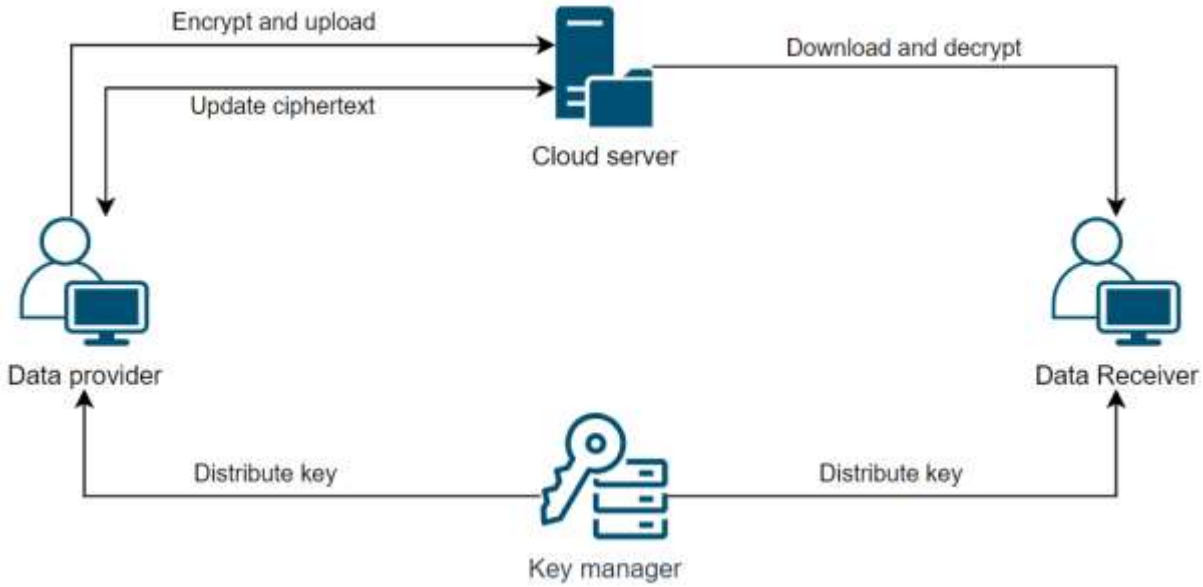

Figure 3.1. System design

3.3. System Workflow

The system design proposed for this thesis combines RS-IBE (Revocable-Storage Identity-Based Encryption) with SHA-256 and AES to create a comprehensive and robust data encryption solution.

As shown in Figure 3.2, the encryption process begins with the plaintext file, which represents the actual data that needs to be encrypted. As an initial step, a key is generated internally by hashing a random message using the SHA-256 hashing algorithm. This key serves as the encryption key for the AES encryption algorithm. To ensure secure key encapsulation, the encryption key is then encrypted under the user identity using RS-IBE. Finally, the plaintext file is encrypted using the AES encryption algorithm, with the generated encryption key. The encryption process transforms the plaintext data into ciphertext, ensuring that the sensitive information remains confidential and protected.

To decrypt the decrypted file, the corresponding user identity is required. The user needs to decrypt the encrypted encryption key using RS-IBE under their identity. Once the encrypted encryption key is decrypted using the RS-IBE private key, the original encryption key is obtained. This decrypted encryption key is then used in the AES decryption algorithm to decrypt the encrypted file. The decryption process transforms the ciphertext back into the original plaintext data, making it readable and usable.

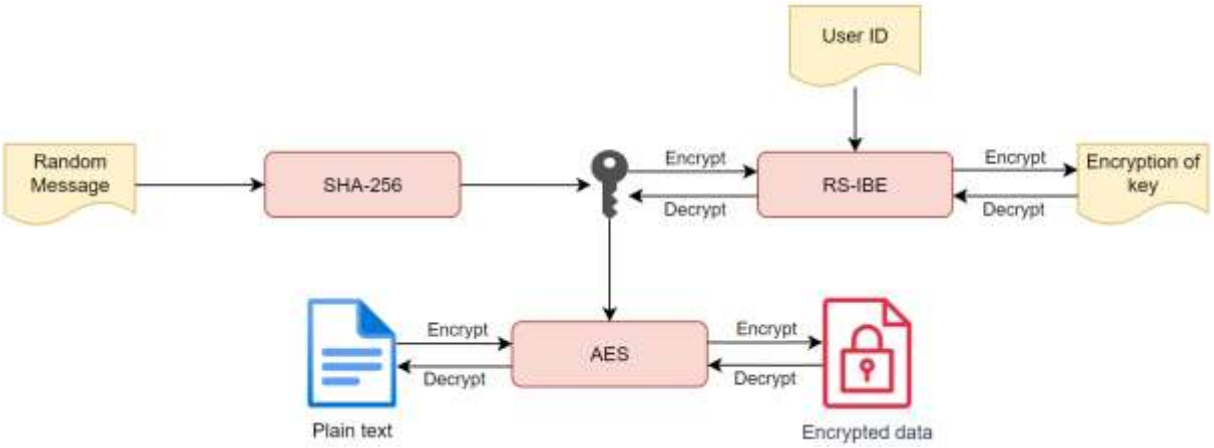

Figure 3.2. System workflow

CHAPTER 4

IMPLEMENTATION AND RESULTS

4.1. Construction

This section presents the construction of the proposed system.

4.1.1. findTo

Let T be a list of nodes in a ciphertext.

$\boldsymbol{findTo}(ID)$: The findTo (CTNodes) algorithm is defined to get the right siblings of the nodes on the path to node v_T, and put it to T.

$$T = RightSibling\big(Path(v_t)\big)\backslash Path\big(Parent(v_t)\big) \cup \{v_t\}$$

For example, let $v_t = v_{010}$

- $Path(v_{010}) = \{0, 01, 010\}$
- $Path(Parent(v_{010})) = \{0, 01\}$
- $RightSibling\big(Path(v_{010})\big) = \{1, 011\}$

$$\mathrm{T} = RightSibling\big(Path(v_t)\big)\backslash Path\big(Parent(v_t)\big) \cup \{v_t\}$$
$$= \{1, 011\}\backslash\{0, 01\} \cup \{010\} = \{1, 011, 010\}$$

The result of this example is demonstrated in Figure 4.1.

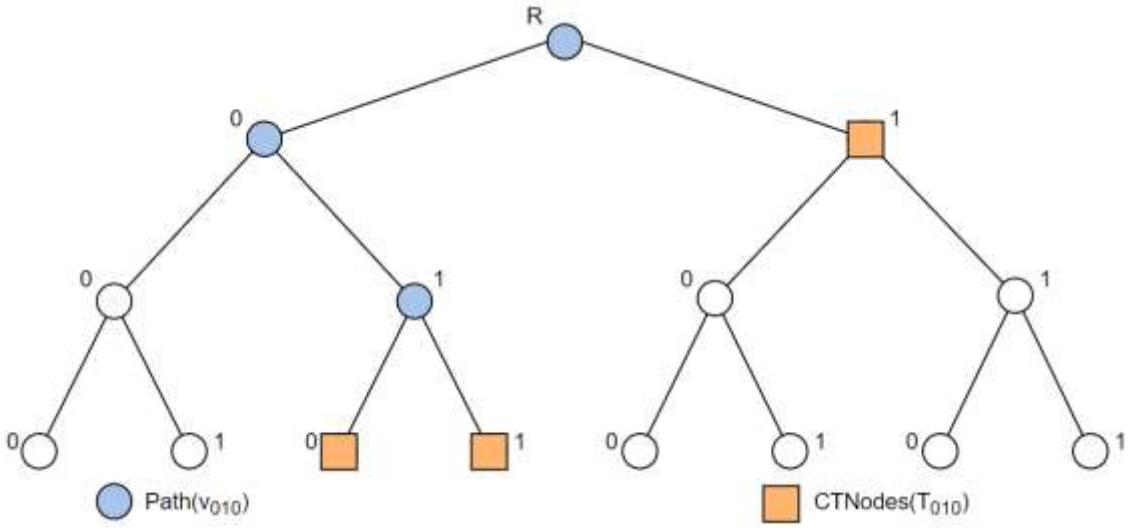

Figure 4.1. Example of findTo

4.1.2. KUNodes

KUNodes(Y, t, RL): The KUNodes algorithm considers the time t and revocation list RL to find out if node n is revoked at time t, then outputs the children of the revoked nodes.

1) Initialize 2 linked lists X and Y.
2) For each node in RL, compare t_i of that node with the current time t.
   If $t_i \leq t$, mark that node as revoked and add its path to X.
3) For each node θ in X, find its left child θ_l and right child θ_r.
   If θ_l is not in X, add θ_l to Y.
   If θ_r is not in X, add θ_r to Y.
4) If $Y = \emptyset$, add root node to Y.
5) Return the linked list Y.

For instance, the users with IDs 011 and 101 is revoked. Then, KUNodes would release a set $Y = \{00,010,100,11\}$. The result of this example is demonstrated in Figure 4.2.

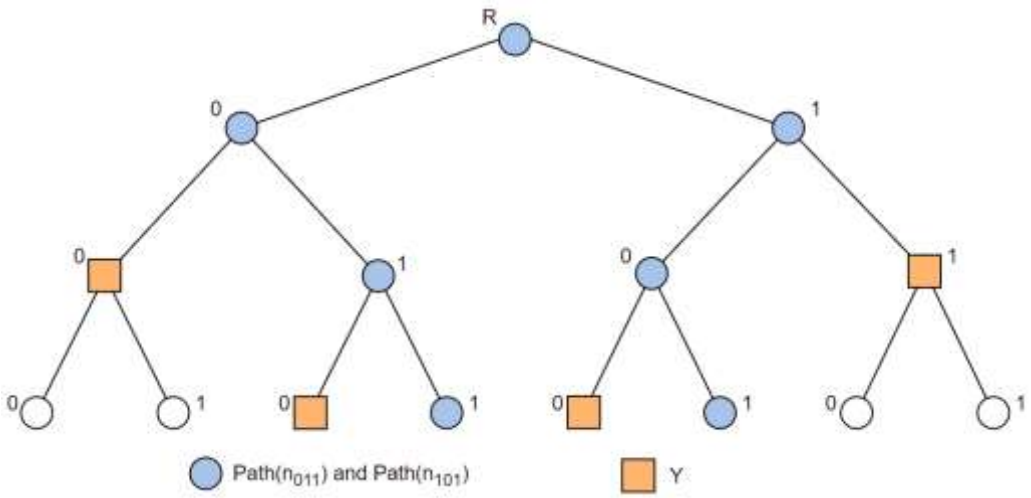

Figure 4.2. Example of KUNodes

4.1.3. Setup

Setup(PP): The setup function computes the public parameters PP.

The setup algorithm works as follows:

1) Choose two bilinear groups G_1 and G_2.
2) Random two group elements g, g_2 from group G_1 and an integer α from $\mathbb{Z}_r$. Compute $g_1 = g^{\alpha}$.
3) Choose two vectors $u = (u_1, u_2, u_3, \dots, u_n) \in G_1^{n+1}$ and $h = (h_1, h_2, h_3, \dots, h_l) \in G_1^{l+1}$.
4) Initialize the revocation list $RL = \emptyset$.
5) Return the public parameters $PP = \{G_1, G_2, g, g_1, g_2, u, h\}$.

4.1.4. FuncDef

$\boldsymbol{FuncDef}(ID, t, PP, PR)$: This function defines F_u and F_h based on ID and time t.

1) For each ID and time t, compute $Fu(ID)$ and $Fh(t)$:

$$F_u(ID) = u_0 \prod_{i=1}^{n} u_i^{ID[i-1]}$$

$$F_h(t) = h_0 \prod_{j=1}^{l} h_j^{t[j-1]}$$

2) Return the functions $F_u(ID)$ and $F_h(t)$.

4.1.5. SKGen

$\boldsymbol{SKGen}(PP, PR, Path)$: This private key generation requires an ID and the path to a node to generate the private key SK_{ID} following the steps below:

1) Choose a node n and get its ID.
2) For each node $\theta \in Path(n)$:
   a. Verify if $g_{\theta,0}$ is already defined. Otherwise, randomly pick $g_{\theta,0}$ from G_1. Compute $g_{\theta,1} = g_2/g_{\theta,0}$ and store the pair $(g_{\theta,0}, g_{\theta,1})$ in the node θ.
   b. Randomly choose an integer $r_{\theta,0}$ from group $\mathbb{Z}_r$ and compute the secret key.

$$SK_{ID,\theta} = \left(SK_{\theta,0}, SK_{\theta,1}\right)$$
$$= (g_{\theta,0}^{\alpha}\, F_u(ID)^{r_{\theta,0}},$$
$$g^{r_{\theta,0}})$$

3) Return the secret key SK_{ID}.

4.1.6. KeyUpdate

$\boldsymbol{KeyUpdate}(PP, PR, t, KUNodes)$: The key update function takes in the public parameters PP, time period t, and the list of nodes provided by the KUNodes algorithm to produce the updated key KU_t for each node in KUNodes list.

For each node $\theta \in KUNodes$:

1) Extract the value $g_{\theta,1}$, which has been computed in the SKGen function.
2) Randomly choose an integer $r_{\theta,1}$ from $\mathbb{Z}_r$.
3) Update the secret key:

$$KU_{t,\theta} = \left(KU_{\theta,0}, KU_{\theta,1}\right)$$
$$= (g_{\theta,1}^{\alpha} \cdot F_h^{r_{\theta,1}},$$
$$g^{r_{\theta,1}})$$

4) Return the updated key KU_t.

4.1.7. DKGen

$\boldsymbol{DKGen}(PP, PR, DK, ID, Path, KUNodes)$: The decryption key generation algorithm uses the public parameter PP, the nodes in the path to node n, and the KUNodes list to generate the decryption key DK_{ID} for a user with ID with time t.

For each node $\theta \in Path(n) \cap KUNodes$:

1) Randomly choose r_0 and r_1 from $\mathbb{Z}_r$.
2) Generate the decryption key:

$$DK_{ID,t} = \left(DK_{t,1}, DK_{t,2}, DK_{t,3}\right)$$
$$= (SK_{\theta,0} \cdot KU_{\theta,0} \cdot F_u(ID)^{r_0} \cdot F_h(t)^{r1},$$
$$SK_{\theta,1} \cdot g^{r_0},$$
$$KU_{\theta,1} \cdot g^{r_1})$$

3) Return the decryption key $DK_{ID,t}$.

4.1.8. Encrypt

$\boldsymbol{Encrypt}(PP, PR, CT, t, key)$: The encryption algorithm takes as input the public parameter PP, time t, and key to encrypt and output the ciphertext CT.

1) Find the elements of set T using $findTo()$.
2) Randomly choose s_t from $\mathbb{Z}_r$ and set $s_{v_t} = s_t$.
3) Compute the ciphertext:

$b_v \in \{0,1\}^{\leq l}$: binary sequence of the path from root to node v.

$$C_0 = key \cdot e(g_1, g_2)^{s_t}$$
$$C_1 = g^{-s_t}$$
$$C_2 = F_u(ID)^{s_t}$$
$$C_v = \left(C_{v,0}, C_{v,|b_v|+1}, C_{v,|b_v|+2}, \dots, C_{v,l}\right)$$
$$= \left(\left(h_0 \prod_{j=1}^{|b_v|} h_j^{b_v[j-1]}\right)^{s_v}, h_{|b_v|+1}^{s_v}, h_{|b_v|+2}^{s_v}, \dots, h_l^{s_v}\right)$$

4) Return the ciphertext $CT_{ID,t}$.

4.1.9. CTUpdate

$\boldsymbol{CTUpdate}(PP, PR1, PR2, CT, CT2, t, t2)$: The CTUpdate algorithm updates the ciphertext CT from time t to new time t' $(t' \geq t)$.

1) For each node v' in T′, find a node v in T such that v is a prefix of v'.
2) Randomly choose $s_{t\prime}$ from $\mathbb{Z}_r$.
3) Compute the new set of CT values.

$$C_0' = C_0 \cdot e(g_1, g_2)^{s_{t\prime}}$$

$$C_1' = C_1 \cdot g^{-s_{t\prime}}$$

$$C_2' = C_2 \cdot F_u(ID)^{s_{t\prime}}$$

$$C_{v'} = \left(C_{v',0}, C_{v',|b_{v'}|+1}, C_{v',|b_{v'}|+2}, \dots, C_{v',l}\right)$$

$$= \left(C_{v,0} \cdot \prod_{j=|b_v|+1}^{|b_{v'}|} C_{v,j}^{b_{v'}[j-1]} \cdot \left(\mathrm{h}_0 \prod_{j=1}^{|b_{v'}|} h_j^{b_{v'}[j-1]} \right)^{s_{v'}} \right)$$

The old definition of Wei et al.:

$$C_{v'} = \left(C_{v',0}, C_{v',|b_{v'}|+1}, C_{v',|b_{v'}|+2}, \dots, C_{v',l}\right)$$

$$= \left(C_{v,0} \cdot \prod_{j=|b_v|+1}^{|b_{v'}|} C_{v,j} \cdot \left(h_0 \prod_{j=1}^{|b_{v'}|} h_j^{b_{v'}[j-1]} \right)^{s_{v'}} \right)$$

This formula is not correct. Proof is provided in Section 4.2.

4) Return the updated ciphertext $CT_{ID,t'}$.

4.1.10. Decrypt

$\boldsymbol{Decrypt}(CT, DK, t\ key)$: The decrypt algorithm decrypts the ciphertext CT using the decryption key DK. The return of the correct key key means the ciphertext CT and the decryption key DK are valid.

Decrypt the ciphertext to find the value key key with the formula:

$$key = C_0' \cdot e\left(C_1', DK_{t',1}\right) \cdot e\left(C_2', DK_{t',2}\right) \cdot e(C_{v_{t'},0}, DK_{t',3})$$

4.1.11. Revoke

$\boldsymbol{Revoke}(RL, ID)$: The revoke algorithm's inputs are the revocation list RL and the ID of the to-be-revoked node. This function is used to update the RL by adding a new node to it.

4.2. Modification to the CTUpdate definition

While studying and implementing the RS-IBE scheme, I discovered an issue with the CTUpdate function of Wei et al. In the example that follows, I will explain the reason why

Wei's CTUpdate formula is incorrect and suggest an improved and more accurate formula for this function.

Let $t = 100$ and $t' = 110$.

$T_{100} = \{100,101,11\}$

$T_{110} = \{111,110\}$

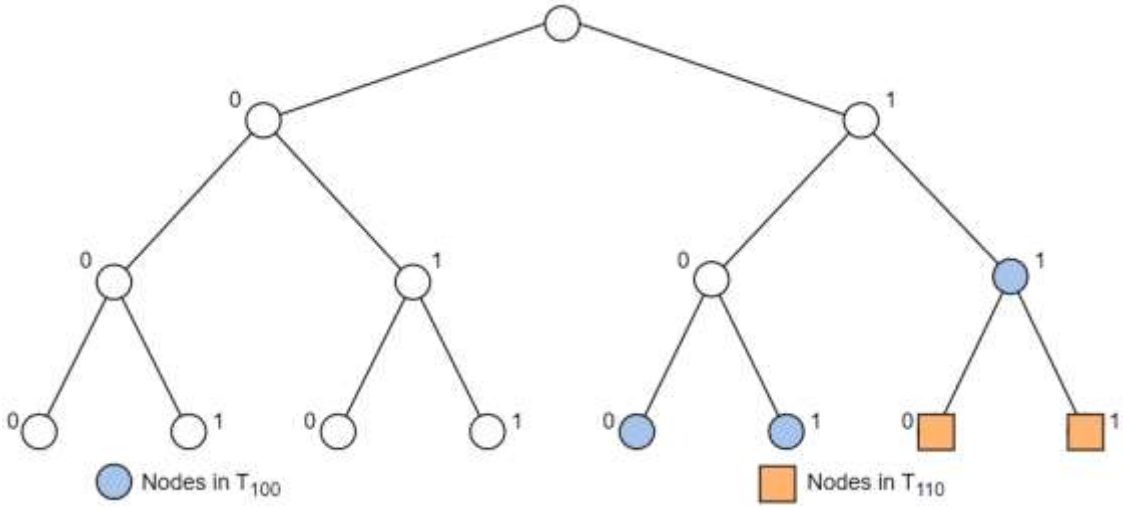

Figure 4.3. Set T of node 100 and 110

When user encrypts at time $t = 100$:

$C_0 = M \cdot e(g_1, g_2)^{s_t}$

$C_1 = g^{-s_t}$

$C_2 = F_u(ID)^{s_t}$

$C_{110,0} = (h_0 h_1^1 h_2^0 h_3^0)^{s_t} = (h_0 h_1)^{s_t}$

Ciphertext is updated at time $t' = 110$:

For node 110, node 11 in T_{100} is a prefix of nodes 110 and 111.

Update the ciphertext:

$C_0' = C_0 \cdot e(g_1, g_2)^{s_{t'}} = key \cdot e(g_1, g_2)^{s_t} \cdot e(g_1, g_2)^{s_{t'}} = key \cdot e(g_1, g_2)^{s_t + s_{t'}}$

$C_1' = C_1 \cdot g^{-s_{t'}} = g^{-s_t} \cdot g^{-s_{t'}} = g^{-(s_t + s_{t'})}$

$C_2' = C_2 \cdot F_u(ID)^{s_{t'}} = F_u(ID)^{s_t} \cdot F_u(ID)^{s_{t'}} = F_u(ID)^{s_t + s_{t'}}$

$C_{110,0} = C_{11,0} \cdot C_{11,3}^0 \cdot (h_0 h_1^1 h_2^1 h_3^0)^{s_{t'}}$

Where:

$C_{11,0} = (h_0 h_1 h_2)^{s_t}$

$C_{11,3} = (h_3)^{s_t}$

According to Wei et al., the $C_{110,0}$would be:

$$C_{110,0} = (h_0 h_1 h_2)^{s_t} \cdot h_3^{s_t} \cdot (h_0 h_1 h_2)^{s_{t'}} = (h_0 h_1 h_2)^{s_t + s_{t'}} \cdot h_3^{s_t}$$

However, when applying this value of $C_{110,0}$ to the formula of Decrypt function, $h_3^{s_t}$ will be redundant. This results in an incorrect returned value of key.

For other elements to be eliminated in Decrypt function and the algorithm returns the correct value of key, the target should be:

$$C_{110,0} = (h_0 h_1 h_2)^{s_t + s_{t'}}$$

Thus, the formula for updating the ciphertext should be:

$$C_{v'} = \left(C_{v',0}, C_{v',|b_{v'}|+1}, C_{v',|b_{v'}|+2}, \dots, C_{v',l}\right)$$

$$= \left(C_{v,0} \cdot \prod_{j=|b_v|+1}^{|b_{v'}|} C_{v,j}^{b_{v'}[j-1]} \cdot \left(\mathrm{h}_0 \prod_{j=1}^{|b_{v'}|} h_j^{b_{v'}[j-1]} \right)^{s_{v'}} \right)$$

4.3. Demonstration

Let the depth of the system be 3. Thus, the maximum number of users and time periods are both $2^3 = 8$.

$Path(\eta)$ is the set of nodes from path to node η.

RL is the revocation list containing the list of users to be revoked.

$KUNodes(RL, t)$ is a list of non-revoked nodes whose parents are revoked.

4.3.1. Setup

We have two random vectors $u = (u_1, u_2, u_3, \dots, u_n) \in G_1^4$ and $h = (h_1, h_2, h_3, \dots, h_l) \in G_1^4$.

4.3.2. FuncDef

Choose η_7 and v_{t_4}, ID and t will be 111 and 100, respectively.

Two functions $F_u(ID)$ and $F_h(t)$ are calculated as follows:

$$F_u(111) = u_0 \prod_{i=1}^{3} u_i^{ID[i-1]} = u_0 u_1^1 u_2^1 u_3^1 = u_0 u_1 u_2 u_3$$

$$F_h(100) = h_0 \prod_{j=1}^{3} h_j^{t[j-1]} = h_0 h_1^1 h_2^0 h_3^0 = h_0 h_1$$

4.3.3. SKGen

$$Path(111) = \{1, 11, 111\}$$

Choose $g_{\theta,0} \overset{R}{\leftarrow} G_1$

We have some value pairs:

$$(g_{1,0}, g_{1,1} = g_2/g_{1,0})$$
$$(g_{11,0}, g_{11,1} = g_2/g_{11,0})$$
$$(g_{111,0}, g_{111,1} = g_2/g_{111,0})$$

Choose $r_{\theta,0} \overset{R}{\leftarrow} \mathbb{Z}_r$

The values for SK are:

$$SK_{111,1} = (SK_{1,0}, SK_{1,1}) = (g_{1,0}^{\alpha}\, F_u(111)^{r_{1,0}}, g^{r_{1,0}})$$
$$SK_{111,11} = (SK_{11,0}, SK_{11,1}) = (g_{11,0}^{\alpha}\, F_u(111)^{r_{11,0}}, g^{r_{11,0}})$$
$$SK_{111,111} = (SK_{111,0}, SK_{111,1}) = (g_{111,0}^{\alpha}\, F_u(111)^{r_{111,0}}, g^{r_{111,0}})$$

4.3.4. KeyUpdate

Assume the revocation list includes two nodes 000 and 001. $RL = \{000, 001\}$.

$$\text{KUNodes} = \{1, 01\}$$

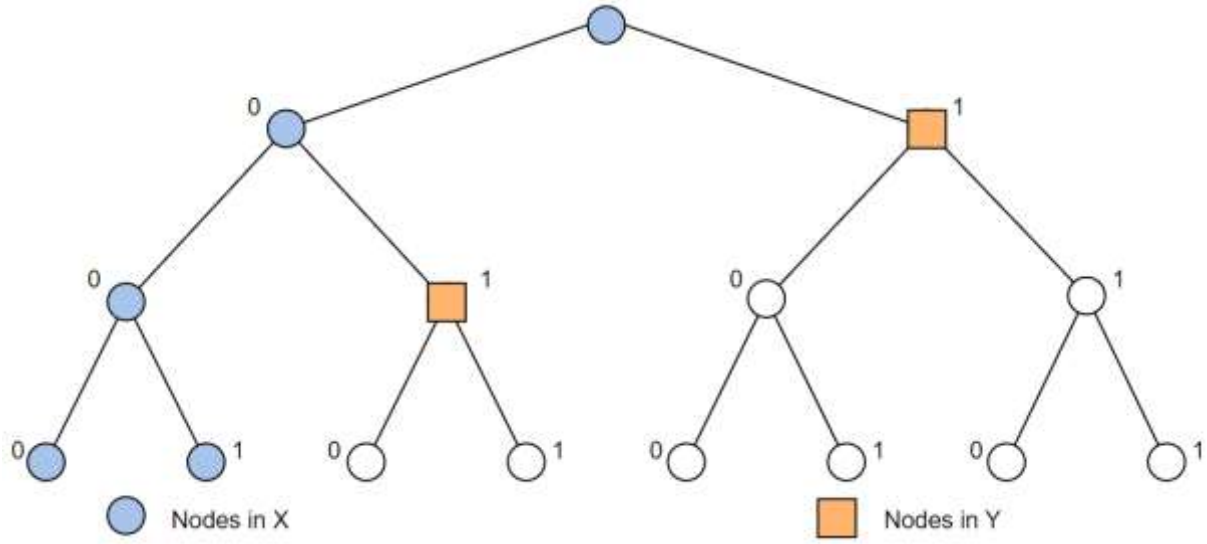

Figure 4.4. An example of KUNodes for nodes 000 and 001

Choose v_{t_6}, t will be 110.

Extract $g_{\theta,1}$

Choose $r_{\theta,1} \overset{R}{\leftarrow} \mathbb{Z}_r$

The values for KU are:

$$KU_{t,1} = (KU_{1,0}, KU_{1,1}) = (g_{1,1}^{\alpha} \cdot \mathrm{F_h}(110)^{r_{1,1}}, g^{r_{1,1}})$$
$$KU_{t,01} = (KU_{01,0}, KU_{01,1}) = (g_{01,1}^{\alpha} \cdot \mathrm{F_h}(110)^{r_{01,1}}, g^{r_{01,1}})$$

4.3.5. DKGen

There is a node $\theta \in Path(n) \cap KUNodes$, which is $\theta = 1$.

The values for $\theta = 1$ are:

$$SK_{111,1} = \left(SK_{1,0}, SK_{1,1}\right) = (g_{1,0}^{\alpha}\, F_u(ID)^{r_{1,0}}, g^{r_{1,0}})$$

$$KU_{t,1} = \left(KU_{1,0}, KU_{1,1}\right) = (g_{1,1}^{\alpha} \cdot \mathrm{F_h(t)}^{r_{1,1}}, g^{r_{1,1}})$$

Choose $r_0, r_1 \overset{R}{\leftarrow} \mathbb{Z}_r$

The values DK are:

$$DK_{ID,t} = \left(DK_{t,1}, DK_{t,2}, DK_{t,3}\right)$$

$$= (SK_{1,0} \cdot KU_{1,0} \cdot F_u(ID)^{r_0} \cdot F_h(t)^{r1},$$

$$SK_{1,1} \cdot g^{r_0},$$

$$KU_{1,1} \cdot g^{r_1})$$

4.3.6. Encrypt

Encrypt at time v_{t_4}, $t = 100$

Choose $s_t \overset{R}{\leftarrow} \mathbb{Z}_r$

$v_{\mathrm{T}} = v_{t_4}$, then

- $Path(v_{t_4}) = \{1, 10, 100\}$
- $Path(Parent(v_{t_4})) = \{1, 10\}$
- $RightSibling\left(Path\left(v_{t_4}\right)\right) = \{11, 101\}$

$$\mathrm{T} = RightSibling\left(Path(v_t)\right) \backslash Path\left(Parent(v_t)\right) \cup \{v_t\}$$

$$= \{11, 101\} \backslash \{1, 10\} \cup \{100\} = \{11, 100, 101\}$$

$$\mathrm{T}_{100} = \{11, 100, 101\}$$

Calculate the ciphertext:

$C_0 = key \cdot e(g_1, g_2)^{s_t}$

$C_1 = g^{-s_t}$

$C_2 = F_u(ID)^{s_t}$

$C_{11,0} = (h_0 h_1^1 h_2^1)^{s_t} = (h_0 h_1 h_2)^{s_t}$

$C_{11,3} = (h_3)^{s_t}$

4.3.7. CTUpdate

Update ciphertext at v_{t_6}, $t' = 110$

Choose $s_{t'} \overset{R}{\leftarrow} \mathbb{Z}_r$

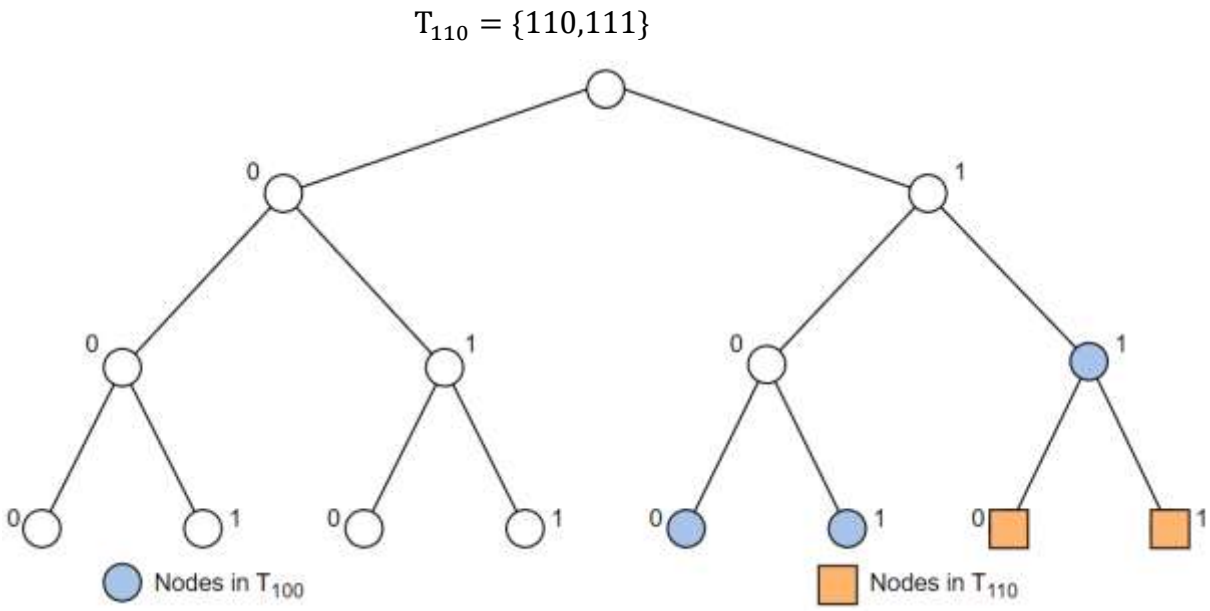

Figure 4.5. An example of T for nodes 100 and 110

For node 110, node 11 in T_{100} is a prefix of nodes 110 and 111.

Update the ciphertext:

$C_0' = C_0 \cdot e(g_1, g_2)^{s_{t'}} = M \cdot e(g_1, g_2)^{s_t} \cdot e(g_1, g_2)^{s_{t'}} = M \cdot e(g_1, g_2)^{s_t + s_{t'}}$

$C_1' = C_1 \cdot g^{-s_{t'}} = g^{-s_t} \cdot g^{-s_{t'}}$

$C_2' = C_2 \cdot F_u(ID)^{s_{t'}} = F_u(ID)^{s_t} \cdot F_u(ID)^{s_{t'}} = F_u(ID)^{s_t + s_{t'}}$

$C_{110,0} = C_{11,0} \cdot C_{11,3}^0 \cdot (h_0 h_1^1 h_2^1 h_3^0)^{s_{t'}}$

Where:

$C_{11,0} = (h_0 h_1 h_2)^{s_t}$

$C_{11,3} = (h_3)^{s_t}$s

Thus,

$$C_{110,0} = (h_0 h_1 h_2)^{s_t} \cdot ((h_3)^{s_t})^0 \cdot (h_0 h_1 h_2)^{s_{t'}} = (h_0 h_1 h_2)^{s_t + s_{t'}}$$

4.3.8. Decrypt

Decrypt the ciphertext to decrypt M:

$$C_0' \cdot e\left(C_1', DK_{t',1}\right) \cdot e\left(C_2', DK_{t',2}\right) \cdot e(C_{v_{t'},0}, DK_{t',3})$$

$$= key \cdot e(g_1, g_2)^{s_t + s_{t'}} \cdot e\left(g^{-(s_t + s_{t'})}, g_2^{\alpha} \cdot F_u(ID)^{r_{\theta,0} + r_0} \cdot \mathrm{F_h(t)}^{r_{\theta,1} + r_1}\right)$$

$$\cdot e(F_u(ID)^{s_t + s_{t'}}, g^{r_{\theta,0} + r_0}) \cdot e(F_h(t)^{s_t + s_{t'}}, g^{r_{\theta,1} + r_1})$$

$$= key \cdot e(g_1, g_2)^{s_t + s_{t'}} \cdot e(g^{\alpha}, g_2)^{-(s_t + s_{t'})}$$

$$= key \cdot e(g_1, g_2)^{s_t + s_{t'}} \cdot e(g_1, g_2)^{-(s_t + s_{t'})} = key$$

Where $g_1 = g^{\alpha}$

4.4. Implementation

I implement the system using mainly the Pairing-Based Cryptography library version 0.5.14 [5]. The implementation is performed on a Linux Ubuntu20.04 (64-bit) machine with an 11th Gen Intel® Core™ i7 CPU (1185G7 @ 3.00GHz × 2) and 8.00 GB RAM.

In the implementation, I set the level of the binary tree to be DEPTH = 3. Therefore, the number of users and time periods are N = 8 and T = 8. The revocation list is set to contain two nodes 000 and 001.

The system runs in the following order:

At the initial time period:

1. Run the initial setup to generate the public parameters needed for the calculation.
2. Generate two public functions, one for the user and one for the time period.
3. Generate the secret key for each user.
4. Encrypt the message at the first time period to receive a ciphertext.

At the new time period:

1. Generate another two public functions for the new user and the new time period.
2. Update the key and the ciphertext to the new time period.
3. From the provided values of the secret key and the key update, users generate themselves a decryption key DK.
4. Decrypt the ciphertext to receive the message.
5. If a user is revoked at the new time period, he/she will not be able to update the secret key, and hence, generating the decryption key at the new time period will be impossible.
6. The message is used as a hash to encrypt and decrypt the shared file.

4.5. Results

First, the system randomizes a message.

Figure 4.6. Random message and Hash message

```
Random message =
[502784566633496936416777501316573829553417303689008130427939186
1460483948908821396081875636983118539901177583166441474242710403
231152402302838516225068063,
706906686531017812642394927944643037807227618286242698582251589O
```

```
111195690676129894313682830444036126278964346988541394430524579892598080430973150676285915]
```

Hash the message.

```
SHA-256 Hash:
386aa1d55dafdebe8075a5e2a17227cb19c1804974a64ef8083c1e3c6775f82c
0000000000000000000000000000000000000000000000000000000000000000
```

The hash is then converted to an element of bilinear group GT. Then, the file encryption process starts with the decryption key generated from the hashed message.

Figure 4.7. File encryption key

```
key before =
[295477352951539622633095320970617170395921605696163913334127927
408065963291463004472493034022103755289136016757834810939917930146462413978708830265645465
6,
502784566633496936416777501316573829553417303689008130427939186146048394890882139600187563
698311853990117758316644147424271040323115240230283851622506806
3]
```

After encrypting the file, the system runs through the RS-IBE process to encrypt the encryption key.

At the time period t', the RS-IBE process continues to be run to update the ciphertext and the decryption key. The Decrypt function is also run to decrypt the encrypted encryption key.

The results of the key after being encrypted and decrypted are compared with each other. If they are identical, the signature verifies. The file decryption process occurs successfully.

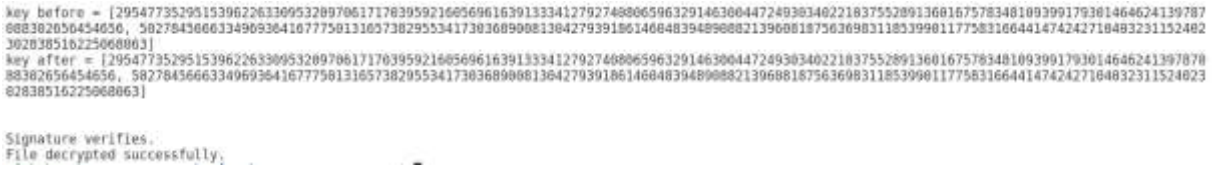

Figure 4.8. Keys comparision

```
key before =
[295477352951539622633095320970617170395921605696163913334127927
408065963291463004472493034022103755289136016757834810939917930146462413978708830265645465
6,
502784566633496936416777501316573829553417303689008130427939186146048394890882139600187563
698311853990117758316644147424271040323115240230283851622506806
3]
```

```
4646241397870883026564546 56,
5027845666334969364167775013165738295534173036890081304279391861
4604839489088213960818756369831185399011775831664414742427104032
3115240230283851622506806 3]
key after =
[2954773529515396226330953209706171703959216056961639133341279 27
4080659632914630044724930340221037552891360167578348109399179301
4646241397870883026564546 56,
5027845666334969364167775013165738295534173036890081304279391861
4604839489088213960818756369831185399011775831664414742427104032
3115240230283851622506806 3]
```

For example, we have a plaintext file:

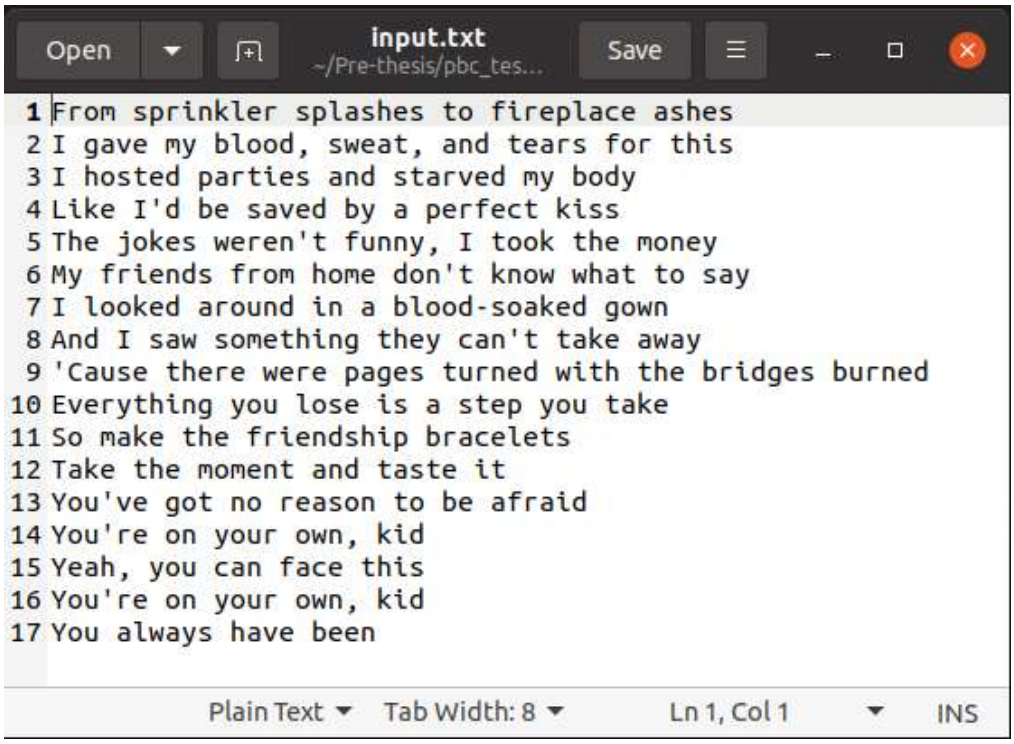

Figure 4.9. Plaintext file

Encrypt the plaintext file with AES to get the ciphertext file:

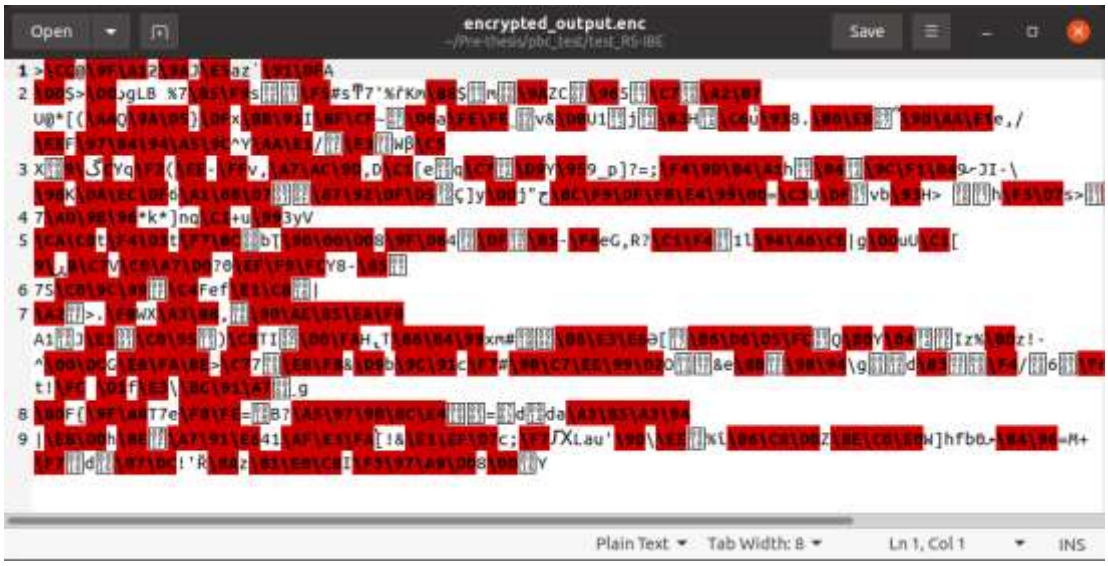

Figure 4.10. Ciphertext file

Decrypt the ciphertext file to reclaim the plaintext:

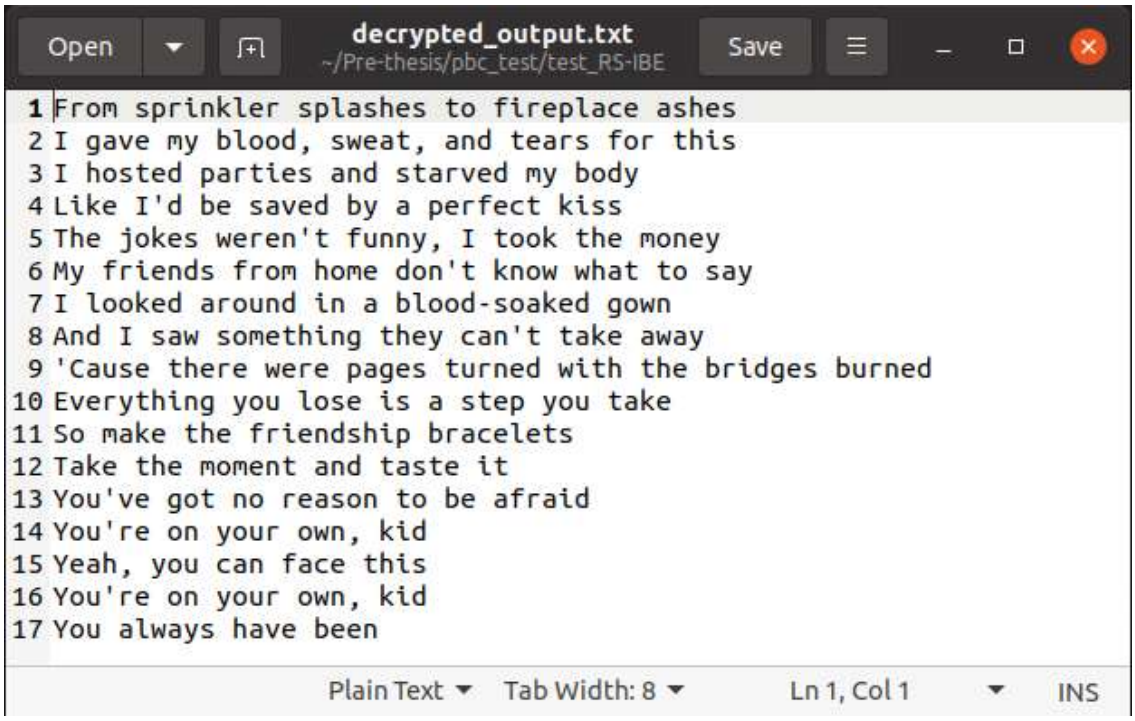

Figure 4.11. Decrypted file

As indicated in the result, the system successfully performs hashing the random message to produce a key using SHA-256, encrypting at a time period t and decrypting at another time period t' of the key using RS-IBE. The key is successfully used to encrypt a plaintext file and decrypt a ciphertext file with AES.

CHAPTER 5

DISCUSSION AND EVALUATION

5.1. Discussion

The outlined objectives have been met with the demo version such as a system providing a secure approach for file sharing, a RS-IBE model for key encapsulation, encrypting files with AES, and proposing a modification to Wei et al.'s formula. Integrating the RS-IBE strategy into AES and SHA has added an extra layer of security to the system. It applies a complex yet effective method to the key encryption phase, ensuring the keys are well-protected.

The algorithm and security models used in the system are not the optimal method but can well perform the basic functions with fairly good performance. Nevertheless, the system still has many limitations that need to be overcome such as supporting only text files or only encrypting files for each user individually, not being able to encrypt a group of users. The content will be further developed when new technologies are applied because the current system is still in the stage of developing fundamental library functions and verifying if the key and data are properly encrypted and correctly decrypted.

5.2. Evaluation

The proposed system design offers several significant advantages that contribute to its effectiveness and suitability for various applications. First, it ensures a high level of security for the encrypted data by combining RS-IBE, SHA-256, and AES. This multi-layered security approach provides secure key encapsulation, access control (RS-IBE), and strong encryption algorithms (SHA-256 and AES), ensuring the confidentiality and integrity of the data. Secondly, the system design incorporates a revocation function, allowing for user access removal. This enhances the access management feature of the system. Lastly, the new approach for calculating nodes of the binary tree has greatly improved the efficiency and performance of the system. By only calculating the nodes that are needed for a particular computation, unnecessary calculations and computational overhead are eliminated, resulting in faster processing. Since the binary tree is not pre-built entirely, the system has the ability to scale with varying input sizes. Furthermore, different data types and data structures such as structs, linked lists, and symbol tables are implemented to safely store and extract values, maintaining the correctness of the results.

CHAPTER 6

CONCLUSION AND FUTURE WORK

5.1. Conclusion

In conclusion, when it comes to sharing data, cloud computing offers great value for users in terms of convenience and flexibility. Following the definition of Wei et al., a detailed implementation with some modifications to the RS-IBE scheme including the combination of SHA algorithm to enhance the security for file encryption has been proposed. Besides using user identities to encrypt and decrypt, this system also enables identity revocation and continuous ciphertext update, which prevents revoked users from accessing the previously shared data. First, RS-IBE is applied to encrypt and store the encryption key. The key is used to encrypt and decrypt a specific file following the AES cipher. A detailed demonstration of the algorithm for each function is also carried out. Moreover, I proved that the formula to compute ciphertext update is incomplete. Therefore, a modification to the ciphertext update formula has been suggested and applied to the proposed system.

5.2. Future work

Although the current system offers some basic functions needed for an encrypt-decrypt process, many shortcomings still need to be worked on so that the system can be better improved.

Firstly, the algorithm should be updated in the future to enable sharing with a group of users or multiple users. Currently, the sender can only encrypt a file for each recipient individually at a time.

Secondly, improving the system to run with real time. The time in the current system is managed by a binary tree structure, which does not reflect the realistic time. A method to track and measure time such as epoch time or Unix time should be implemented.

Finally, in order to expand the system and make it accessible to more users, a comprehensive interface needs to be developed. The current system uses a terminal and command line interface for interaction, which may be challenging or even impossible for users to communicate with the system. Users would find it simpler to interact and carry out operations with a graphical interface.

REFERENCES

[1] A. Shamir, "Identity-Based Cryptosystems and Signature Schemes," *Advances in Cryptology*, pp. 47-53, 1985.

[2] M. F. D. Boneh, "Identity-Based Encryption from the Weil Pairing," *Advances in Cryptology --- CRYPTO 2001*, pp. 213-229, 2001.

[3] W. L. a. X. H. J. Wei, "Secure Data Sharing in Cloud Computing Using Revocable-Storage Identity-Based Encryption," *IEEE Transactions on Cloud Computing*, vol. 6, no. 4, pp. 1136-1148, 2018.

[4] K. Lee, "Comments on “Secure Data Sharing in Cloud Computing Using Revocable-Storage Identity-Based Encryption”," *IEEE Transactions on Cloud Computing*, vol. 8, no. 4, pp. 1299-1300, 2020.

[5] B.Lynn, "PBC Library - Pairing-Based Cryptography," [Online]. Available: https://crypto.stanford.edu/pbc/.

APPENDIX