Temporal Access Control on Cloud Data

Ayan Das

Temporal Access Control on Cloud Data

Dissertation submitted in partial fulfillment of the requirements for the degree of

Master of Technology
in
Computer Science

by

Ayan Das
[Roll No: CS-1414]

under the guidance of

Dr. Sushmita Ruj
Assistant Professor
Cryptology and Security Research Unit

Indian Statistical Institute
Kolkata-700108, India

July 2016

To my family and my supervisor

CERTIFICATE

This is to certify that the dissertation entitled **"Temporal Access Control on Cloud Data"** submitted by **Ayan Das** to Indian Statistical Institute, Kolkata, in partial fulfillment for the award of the degree of **Master of Technology in Computer Science** is a bonafide record of work carried out by him under my supervision and guidance. The dissertation has fulfilled all the requirements as per the regulations of this institute and, in my opinion, has reached the standard needed for submission.

Dr. Sushmita Ruj
Assistant Professor,
Cryptology and Security Research Unit,
Indian Statistical Institute,
Kolkata-700108, INDIA.

Acknowledgements

I take this opportunity to express my gratitude towards my project supervisor and mentor, Dr. Sushmita Ruj , for her guidance, encouragement and useful critiques of this research work. There were times when I had to change the concept of the project completely. It was her support that led me through to complete the anticipated work. I really appreciate Dr. Sushmita Ruj's willingness to generously devote her time for clariying my doubts, editorial comments and her efforts to ensure that I successfully complete the work as per schedule.

My honest appraisals to our institute which played the role of a great platform on which I could carry out my work with almost no difficulty, as the relevant research papers needed for my work were readily available.

Ayan Das
M.Tech. II Year
Discipline of Computer Science and Engineering
ISI Kolkata

Abstract

Cloud storage is a model of data storage in which the digital data is stored in remote server. The physical storage spans multiple servers (and often locations), and the physical environment is typically owned and managed by a hosting company. These cloud storage providers are responsible for keeping the data available and accessible. Generally data is used to store in cloud without any encryption. But The two big concerns about cloud storage are reliability and security. Clients are not likely to entrust their data to another company without a guarantee that they will be able to access their information whenever they want and no one else will be able to get at it. One solution of these problem is temporal access control which is based on attribute based encryption.

Attribute-based access control is one of the most important security mechanisms for data storage, especially in cloud computing. Attribute-based encryption is an attribute-based access control mechanism which requires data to be kept encrypted at servers. This data is open to access to all, but can be decrypted only by those users whose attributes satisfy a given access policy. Also the data owner can revoke any authorized user from decrypting the data in any point of time. An issue in attribute-based encryption is huge time complexities for generating the keys and encrypting the content. The problem becomes critical when the number of attributes is large, however little work has been done to develop schemes that support efficient and reliable storage of data in cloud with temporal access control. Here, we review some of the important work that has been done in this field. We then present a protocol which improves the efficiency, scalibilty and makes it feasible for real time implementation. We also analyze it's time and communication complexity to demonstrate the efficiency of our methodology. Another merit of our scheme is that the computationally intense task is outsourced to the cloud without compromising on the security of the scheme. In this way, we propose a comprehensive scheme which promises encompass all the major issues in attribute-based encryption.

Contents

List of Tables

Chapter 1

Introduction

1.1 Introduction

Data is usually stored in cloud server without any encryption. But The two big concerns about cloud storage are reliability and security. Clients aren not likely to entrust their data to another company without a guarantee that they will be able to access their information whenever they want and no one else will be able to get at it. There is always the possibility that a hacker will find an electronic back door and access data from cloud server. Hackers could also attempt to steal the physical machines on which data are stored. Here our temporal access control scheme comes into the picture.
Access control refers to a method of selectively granting access to a resource. Access control has numerous applications in cloud computing as access to data has to be restricted to a confined set of users. In a typical access control scheme the data is stored in cloud server in encrypted format. Only the authorized users who have a specific set of attributes can decrypt the data. For example, in social networks like Facebook, statuses, photos, videos, and posts have to visible only to a fixed set of users. It is essential to realize that privacy constitutes an inseparable necessity in online social networks. As another example, file sharing using Dropbox or Google Drive must ensure that access to shared files is granted only to the authorized users. To realize these forms of access control, the three most widely recognized models are as follows:

Identity-Based Access Control
Identity-Based Access Control is access control based on the identity of the user where access authorizations to specific objects are assigned based on user identity. In this method, a list of users is maintained and the identities of users requesting access are checked against the list to mediate access control. This approach is simplistic and works well for small groups but is highly inefficient for large and dynamic groups like social networks.

Role-Based Access Control
Role-Based Access Control is access control in which permissions are assigned to roles rather than to individual users and users are assigned to roles rather than directly to permissions. For example, in a company, certain resources may be accessible only by the role of a manager. As a result, all individuals who have been assigned the role of a manager will be able to access the resource. It is easy to see, that this method is especially useful in a company where it is natural and intuitive to have roles assigned to individuals. Although, this method is more expressive than identity-based access control, it is possible to improve upon this model to support more fine-grained access control.

Attribute-Based Access Control
Attribute-Based Access Control is access control in which every user is assigned a set of attributes and access is granted based on an access policy on attributes. We formally define an access policy as follows:

Definition 1 *Let $\mathcal{A}$ denote a set of attributes, $\mathcal{A} = \{A_1, \cdots, A_m\}$. An access policy $\mathcal{P}$ is a Boolean function on AND/OR logical operations, generated by the grammar: $\mathcal{P} ::= A_i | \mathcal{P}$ AND $\mathcal{P} | \mathcal{P}$ OR $\mathcal{P}$ where $A_i \in \mathcal{A}$.*

For example, suppose that a social network user has grouped his friends into lists like close friends, batch mates, acquaintances, and family. The user may want to share a sensitive status so that only friends who have certain credentials or attributes can access it. For instance, the user may specify the following access structure for accessing this information: ('close friends' **AND** 'batch mates') **OR** ('family').

As illustrated by this example, it can be crucial that the person in possession of the secret data be able to choose an access policy based on specific knowledge of the underlying data. Furthermore, this person may not know the exact identities of all other people who should be able to access the data, but rather she may only have a way to describe them in terms of descriptive attributes or credentials. This provides a very expressive and fine-grained way of controlling access to data.

Traditionally, this type of expressive access control is enforced by employing a trusted server to store data locally. The server is entrusted as a monitor that checks that a user presents proper credentials before allowing him to access records or files. XACML (eXtensible Access Control Markup Language) defines a declarative access control policy language implemented in XML and a processing model describing how to evaluate authorization requests according to the rules defined in policies.

The drawback of this trend is that it is increasingly difficult to guarantee the security of data using traditional methods. If the trusted server is compromised, the data will

be lost forever. For these reasons, we would like to require that sensitive data is stored in an encrypted form so that it will remain private even if a server is compromised.

Attribute-Based Encryption
In attribute-based encryption, data is kept encrypted at the server and is open to all users. However, only those users who have sufficient attributes to satisfy the access policy will be able to successfully decrypt the ciphertext. There are majorly two ways of realizing attribute-based encryption - key-policy attribute-based encryption (KP-ABE) and ciphertext-policy attribute-based encryption (CP-ABE).

In key-policy attribute-based encryption, ciphertexts are associated with sets of descriptive attributes, and users' keys are associated with policies. In key-policy attribute based encryption, the encryptor exerts no control over who has access to the data it encrypts, except by its choice of descriptive attributes for the data.

In ciphertext-policy attribute-based encryption, the access policy is embedded in the ciphertext and not in the users' keys. This allows the encryptor to have complete control over who has access over the data it encrypts.

User Revocation
User revocation refers to the problem of managing attribute-based encryption given that some users or their attributes have been revoked. Data access must then be denied for these revoked users.
Traditional revocation approaches for attribute-based encryption use frequent rekeying, and cannot block access to previously stored data without re-encrypting it. Efficient schemes like EASiER [7] have been proposed to deal with user revocation for online social networks.Our scheme have adopted the idea of revocation in our temporal access scheme
A typical example of where revocation can be used is suppose in hospital some data related to specific patient can be accessed by that patient starting from the time of admission. Now if the patient check out the hospital then hospital need to revoke that patient from accessing the data. But data is supposed to be still visible to doctors and hospital authorities. So we cannot delete the data at that time. So the alternative is to revoke the patient whenever he check out from hospital. This can be done by our temporal access scheme with user revocation.
Temporal Constraints
Another issue in attribute-based encryption is the treatment of attributes which are time-dependent. Every user is assigned an access privilege which contains a time component i.e. a license $A_t[t_a, t_b]$ where $[t_a, t_b]$ is assigned time interval for the attribute A_t. Mediating access control with temporal constraints is not handled by the conventional ciphertext-policy attribute-based encryption proposed by Bethencourt et. al. [1]. For example, we cannot generate a user's private key with the

temporal constraint $4 \leq Month \leq 10$, which is particularly useful for expressing fine-grained access policies. Comparison-based encryption [6], which is built on top of ciphertext-policy attribute-based encryption provides an effective way of dealing with such constraints.

Our Work
The schemes mentioned so far have handled attribute-based encryption without two-phase encryption or distributed key generation. Two-phase encryption are done only on other ABE scheme and distributed key generation is done on simple CPABE scheme. To the best of our knowledge, no significant work has been done to design a scheme which encompasses both these issues in a unified manner in any temporal access scheme. In this thesis, we propose a novel scheme which efficiently addresses both the issues along with the outsourced decryption for mobile cloud. So our protol will work much more efficiently in mobile cloud. in a single protocol. This scheme offers to be a complete package to realize fine-grained access control and has applications in numerous domains like cloud computing and social networks. Also, we provide a complexity analysis to prove the effectiveness of our methodology.

1.2 Our Contributions

In this thesis, we proposed an efficient temporal access control scheme for mobile cloud. Our scheme enjoy the following features:-

- Our first temporal access scheme support user revocation where the authorized user can be revoked from accessing the data stored in cloud. at any point of time.
- Key generation for each user is divided into multiple authorities where each authority will generate key for specific subset of attributes and then those keys will be combined into a single key by a central authority.
- Encryption phase is divided into two phases, Offline encryption(can be done when message and access structure of that message is unknown) and online encryption.
- A decryptor can outsource the decryption task to a proxy (which can be cloud service provider) in such a way that proxy don't get any information about plaintext.
- We have shown analytically that key generation, encryption and decryption cost reduced compared to other existing CPABE scheme.
- we have shown step by step development of our scheme by describing the following scheme:-

1. Scheme without decryption Outsourcing and revocation
2. Scheme without decryption Outsourcing with revocation
3. Scheme with decryption Outsourcing and revocation
4. Scheme with decryption Outsourcing, revocation and online-offline encryption
5. Scheme with decryption Outsourcing, revocation, online-offline encryption and multi-authority key generation

- We have proved the security of our scheme from differnt aspects

1.3 Thesis Outline

This thesis is organized as follows. We define some of the essential preliminaries in chapter 2. In chapter 3, we review some of the existing work in the fields of attribute-based encryption, user revocation and temporal access control. Chapter 4 explain first three basic schemes mentioned above. In chapter 5 we explain last two temporal access scheme with two phase encryption and multi-authority key generation,In chapter 6 we evaluate the complexity of our proposed scheme and compare it with comparison-based encryption. We show that the additional overhead associated with user revocation is not significant and thereby, our scheme is efficient. Finally, in chapter 6, we conclude the thesis and also mention our future goals with respect to our novel scheme.

Chapter 2

Preliminaries

This section introduces some basic concepts and preliminaries used in attribute-based encryption. First, we present the concept of secret sharing and describe Shamir's secret sharing scheme in detail. Then, we define a very general type of structure known as an access structure, which is used to label different sets of encrypted data. Finally, we go on to present a few facts related to groups with efficiently computable bilinear maps.

2.1 Secret Sharing

Secret sharing refers to a method of distributing a secret amongst a group of participants, each of whom is allocated a share of the secret. The secret can be reconstructed only when a sufficient number, of possibly different types, of shares are combined together. Individual shares are of no use on their own. We formally define threshold secret sharing and describe Shamir's secret sharing scheme, which is the most widely used secret sharing scheme.

2.1.1 Threshold Secret Sharing

A (t, n) *threshold secret sharing scheme* distributes a secret among n participants in such a way that any t of them can recreate the secret, but any $t-1$ or fewer members gain no information about it. The piece held by a single participant is called a *share* of the secret. Secret sharing schemes are normally set up by a trusted authority who computes all shares and distributes them to participants via secure channels. The trusted authority who sets up the scheme is called a *dealer*. The recovery of the secret is done by the so-called *combiner* who on behalf of the cooperating group computes the secret. The combiner can be collective, i.e., all active participants show to each other their shares so that any active participant can calculate the secret.

Assume that secrets belong the set K and shares are from the set S. Let S_i be the set from which the dealer draws shares for the participants $P_i, i = 1, \cdots, n$. The set of all participants $P = \{P_1, \cdots, P_n\}$.

Definition 2 *A (t, n) threshold scheme is a collection of two algorithms. The first algorithm is called the dealer*

$D : K \rightarrow S_1 \times S_2 \times \cdots \times S_n$

assigns shares to the participants for a secret $k \in K$. The share $s_i \in S_i$ is communicated via a secure channel to the participant P_i. If all share sets S_i are equal we simply say that $s_i \in S$.

The second algorithm (the combiner)

$C : S_{i_1} \times S_{i_2} \times \cdots \times S_{i_j} \rightarrow K$

takes an arbitrary collection of shares and attempts to compute the secret. The combiner recovers the secret successfully only if the number j of different shares is greater than or equal to t $(j \geq t)$. It fails if the number j of shares is smaller than t $(j < t)$.

2.1.2 Shamir's Secret Sharing Scheme

Shamir's secret sharing scheme [8] uses Lagrange polynomial interpolation to design a (t, n) threshold secret sharing scheme. All calculations are done in a Galois Field $GF(p)$ where the prime p is a large enough integer (so that the secret is always smaller than p).

A (t, n) Shamir scheme is constructed starting with the dealer. The dealer chooses n different points $x_i \in GF(p)$ for $i = 1, \cdots, n$. These points are public. Next, the dealer randomly selects coefficients $a_0, \cdots, a_{t-1}$ from $GF(p)$. The polynomial $f(x) = a_0 + a_1 x + \cdots + a_{t-1} x^{t-1}$ is of degree at most $(t-1)$. The shares are $s_i = f(x_i)$ for $i = 1, \cdots, n$ and the secret $k = f(0)$. The share s_i is distributed to the participant $P_i \in P$ via a secure channel and is kept secret.

When t participants agree to cooperate, the combiner takes their shares and tries to recover the secret polynomial $f(x)$. The combiner knows t points on the curve $f(x)$

$(x_{i_j}, f(x_{i_j})) = (x_{i_j}, s_{i_j})$ for $j = 1, \cdots, t$.

The Lagrange interpolation formula allows us to determine the polynomial $f(x)$ of degree $(t-1)$ from t different points (x_{i_j}, s_{i_j}), thus,

$$f(x) = \sum_{j=1}^{t} s_{i_j} \prod_{1 \le l \le t, l \ne j} \frac{x - x_{i_l}}{x_{i_j} - x_{i_l}}.$$

The secret $k = f(0)$, therefore, we obtain,

$$k = a_0 = \sum_{j=1}^{t} s_{i_j} b_j,$$

where,

$$b_j = \prod_{1 \le l \le t, l \ne j} \frac{x_{i_l}}{x_{i_l} - x_{i_j}}$$

If the combiner knows $(t-1)$ or fewer shares, it cannot find the unique solution for $k = a_0$ as the system contains t unknowns but fewer than t equations.

2.2 Access Structures

2.2.1 Access Structures

Definition 3 *The collection of all subsets of participants who are able to access the secret is called the access structure* Γ.

The access structure of (t, n) threshold schemes is $\Gamma = \{\mathcal{A} \in 2^{\mathcal{P}} : |\mathcal{A}| \geq t\}$, where $2^{\mathcal{P}}$ is the class of all subsets of $\mathcal{P}$. The access structure, in this case, consists of all groups whose cardinality is at least t. These groups are called *authorized subsets*. On the other hand, an *unauthorized subset* is a group that does not belong to Γ or whose cardinality is smaller than t.
Consider the scenario where it has been decided by the Uni- versity that the marksheets of students in the Computer Science department prior to publishing will be made available only to the Computer Science faculty and the Examination Department and to dean of studies. So, the relevant policy for encrypting marksheets would be (COM- PUTER SCIENCE AND FACULTY) OR EXAMINATION DEPARTMENT OR "Dean". So the access structure will look like :-

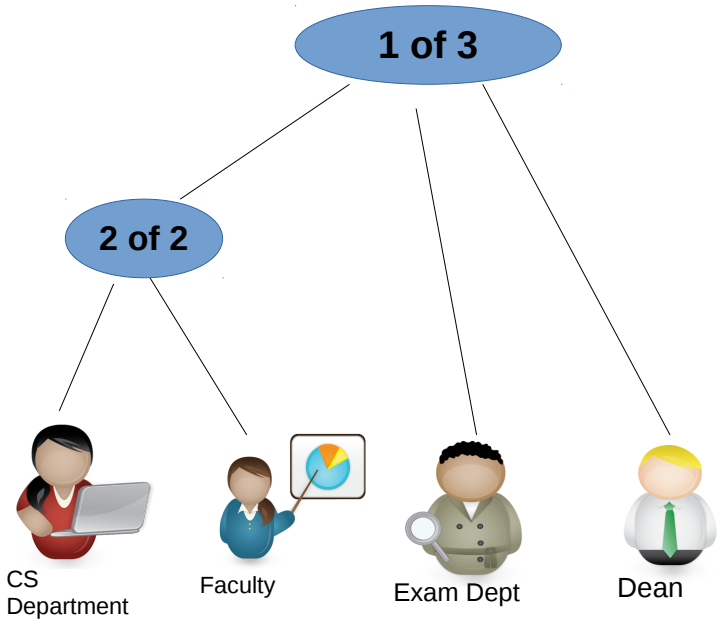


Figure 2.1: Access structure for result before publishing

2.2.2 Monotone Access Structures

Definition 4 *An access structure Γ is monotone if for any subset $\mathcal{A} \in \Gamma$, all its supersets $\mathcal{B}$ are contained in Γ, that is, if $\mathcal{A} \in \Gamma$ and $\mathcal{A} \subseteq \mathcal{B}$, then $\mathcal{B} \in \Gamma$.*

2.3 Bilinear Maps

2.3.1 Bilinear Maps

Let $\mathbb{G}_0$ and $\mathbb{G}_1$ be two multiplicative cyclic groups of prime order p. Let g be a generator of $\mathbb{G}_0$ and e be a bilinear map, $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$. The bilinear map e has the following properties:

- Bilinearity: $\forall u, v \in \mathbb{G}_0$ and $a, b \in \mathbb{Z}_p$, we have $e(u^a, v^b) = e(u, v)^{ab}$;
- Non-degeneracy: $e(g, g) \neq 1$.

We say that $\mathbb{G}_0$ is a bilinear group if the group operation in $\mathbb{G}_0$ and the bilinear map $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$ are both efficiently computable.

Proposition 1 *The bilinear map e is symmetric i.e. $\forall u, v \in \mathbb{G}_0$, $e(u, v) = e(v, u)$.*

2.3.2 Composite Order Bilinear Maps

A bilinear map group system [4] $\mathbb{S} = (N = pq, \mathbb{G}, \mathbb{G}_T, e)$ where $N = pq$ is the RSA-modulus, p, q are two large primes, $\mathbb{G}$ and $\mathbb{G}_T$ are two cyclic groups with order $n = s'p'q'$, and e is a computable bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ with the following properties:

- Bilinearity: for any $g, h \in \mathbb{G}$ and all $a, b \in Z$, $e(g^a, h^b) = e(g, h)^{ab}$;
- Non-degeneracy: $e(g, h) \neq 1$ whenever g and h are the generators of the group $\mathbb{G}$;
- Computability: $e(g, h)$ is efficiently computable.

In this system, N is made public and n, s', p', q' are kept secret.

Proposition 2 *Let $\mathbb{G}_{s'}$ and $\mathbb{G}_{n'}$ denote subgroups of order s' and $n' = p'q'$ in $\mathbb{G}$, respectively. For $g \in \mathbb{G}_{s'}$ and $h \in \mathbb{G}_{n'}$, $e(g, h)$ is the identity element in $\mathbb{G}_T$.*

Proof:

Let w denote a generator of $\mathbb{G}$, then, $w^{n'}$ generates $\mathbb{G}_{s'}$ and $w^{s'}$ generates $\mathbb{G}_{n'}$.

Hence, for some k_1, k_2, $g = (w^{n'})^{k_1}$ and $h = (w^{s'})^{k_2}$, we have

$e(g, h) = e((w^{n'})^{k_1}, (w^{s'})^{k_2}) = e(w^{k_1}, w^{k_2})^{s'n'} = 1$

□

2.4 Forward/Backward Derivation Functions

This scheme utilizes the one-way property to represent a total order on integers. This means that given $t_i \leq t_j$ and two corresponding values v_{t_i} and v_{t_j}, there exists an efficient algorithm to obtain v_{t_j} from v_{t_i}; however it is hard to compute v_{t_i} from v_{t_j}. Based on this idea, the forward and backward derivation functions are formally defined.

Let comparable variables be denoted as a countable set $U = \{t_1, t_2, ..., t_T\}$ constituted from the discrete consecutive integers with a total order $0 \leq t_1 \leq t_2 \leq .... \leq t_T \leq Z$, where Z is the largest integer. In order to construct a cryptographic algorithm for integer comparison, a map $\psi : U \to V$, where $V = \{v_{t_1}, ..., v_{t_T}\}$ is a set of values. It is essential for ψ to be an order-preserving map, that is, a map such that $t_i \leq t_j$ in U implies there exists a partial-order $\preceq$ in V to ensure $v_{t_i} \preceq v_{t_j}$ in V where $v_{t_i} = \psi(t_i)$ and $v_{t_j} = \psi(t_j)$. In order to setup this kind of relation over V, we consider the partial-order in V as the one-way property in cryptography, which is defined as a forward derivation function:

Definition 5 *Given a function $f : V \to V$ based on a set $(U, \leq)$, it is called a forward derivation function if it satisfies the conditions:*

- ***Easy to compute:*** *the function f can be computed in polynomial-time, if $t_i \leq t_j$, i.e., $v_{t_j} \leftarrow f_{t_i \leq t_j}(v_{t_i})$;*
- ***Hard to invert:*** *it is infeasible for any probabilistic polynomial (PPT) algorithm to compute v_{t_i} from v_{t_j} if $t_i < t_j$.*

Similarly, we also define a function $\bar{f}$ for the derivation in opposite direction, which is called Backward Derivation Function (BDF). In order to avoid interference between f and $\bar{f}$, we use a different sign $\bar{\psi} : U \to \bar{V}$, and then the BDF $\bar{f}$ is defined as follows:

Definition 6 *Given a function $\bar{f} : \bar{V} \to \bar{V}$ based on a set $(U, \leq)$, it is called a backward derivation function if it satisfies the conditions:*

- ***Easy to compute:*** *the function $\bar{f}$ can be computed in polynomial-time, if $t_i \geq t_j$, i.e., $\bar{v}_{t_j} \leftarrow \bar{f}_{t_i \geq t_j}(\bar{v}_{t_i})$;*
- ***Hard to invert:*** *it is infeasible for any probabilistic polynomial (PPT) algorithm to compute $\bar{v}_{t_i}$ from $\bar{v}_{t_j}$ if $t_i > t_j$.*

2.5 Cryptographic Construction of FDF/BDF

The cryptographic construction for integer comparisons is constructed based on forward/backward derivation functions. This construction is built on a special multiplicative group $\mathbb{G}_{n'}$ of RSA-type composite order $n' = p'q'$, where p', q' are two large primes. First, we choose two different random generators $\varphi, \bar{\varphi}$ in a group $\mathbb{G}_{n'}$, where $\varphi^{n'} = \bar{\varphi}^{n'} = 1$. Next, we choose two different random λ and μ in $\mathbb{Z}^*_{n'}$, where the order of λ, μ is sufficiently large in $\mathbb{Z}^*_{n'}$.

Based on RSA cryptography system, we define two mapping functions $\psi(\cdot), \bar{\psi}(\cdot)$ from an integer set $U = \{t_1, t_2, \cdots, t_T\}$ into $V = \{v_{t_1}, ..., v_{t_T}\}$ and $\bar{V} = \{\bar{v}_{t_1}, ..., \bar{v}_{t_T}\}$ as follows:

$v_{t_i} \leftarrow \psi(t_i) = \varphi^{\lambda^{t_i}} \in \mathbb{G}_{n'}$

$\bar{v}_{t_i} \leftarrow \bar{\psi}(t_i) = \bar{\varphi}^{\mu^{Z - t_i}} \in \mathbb{G}_{n'}$

Next, according to the definition of $\psi(\cdot)$ and $\bar{\psi}(\cdot)$, it is easy to define the FDF $f(\cdot)$ and BDF $\bar{f}(\cdot)$ as

$$v_{t_j} \leftarrow f_{t_i \leq t_j}(v_{t_i}) = (v_{t_i})^{\lambda^{t_j - t_i}} \in \mathbb{G}_{n'}$$

$$\bar{v}_{t_j} \leftarrow \bar{f}_{t_i \geq t_j}(\bar{v}_{t_i}) = (\bar{v}_{t_i})^{\mu^{t_i - t_j}} \in \mathbb{G}_{n'}$$

It is intractable to obtain v_{t_i} from v_{t_j} for $t_i < t_j$ under the RSA assumption that λ^{-1} and μ^{-1} cannot be efficiently computed based on the secrecy of n'.

Chapter 3

Related Work

In this chapter, we review some of the related work done in the field of attribute-based encryption, user revocation and temporal access control. First, we describe ciphertext-policy attribute-based encryption, which allows attribute information to be contained in the user's private key and access policies on data to be defined independently. Next, we describe the revocation scheme of Naor and Pinkas, which is a widely used scheme and proved to be secure. Finally, we describe comparison-based encryption which is used to efficiently represent ranged attributes as well as temporal constraints.

3.1 Ciphertext-Policy Attribute-Based Encryption

In ciphertext-policy attribute-based encryption, the access policy is embedded in the ciphertext and not in the users' keys. This allows the encryptor to have complete control over who has access over the data it encrypts. In this section, we review the algorithms, model, and construction of ciphertext-policy attribute-based encryption scheme.

3.1.1 Algorithms

An ciphertext-policy attribute based encryption scheme [1] consists of four fundamental algorithms: Setup, Encrypt, KeyGen, and Decrypt. In addition, the option of a fifth algorithm Delegate is allowed.

Setup. The setup algorithm takes no input other than the implicit security parameter. It outputs the public parameters PK and a master key MK.

Encrypt$(PK, M, \mathcal{A})$**.** The encryption algorithm takes as input the public parameters PK, a message M, and an access structure $\mathcal{A}$ over the universe of attributes. The algorithm will encrypt M and produce a ciphertext CT such that only a user

that possesses a set of attributes that satisfies the access structure will be able to decrypt the message. It is assumed that the ciphertext implicitly contains $\mathcal{A}$.

Key Generation(MK, S)**.** The key generation algorithm takes as input the master key MK and a set of attributes S that describe the key. It outputs a private key SK.

Decrypt(PK, CT, SK)**.** The decryption algorithm takes as input the public parameters PK, a ciphertext CT, which contains an access policy $\mathcal{A}$, and a private key SK, which is a private key for a set S of attributes. If the set S of attributes satisfies the access structure $\mathcal{A}$ then the algorithm will decrypt the ciphertext and return a message M.

Delegate$(SK, \widetilde{S})$**.** The delegate algorithm takes as input a secret key SK for some set of attributes S and a set $\widetilde{S} \subseteq S$. It output a secret key $\widetilde{SK}$ for the set of attributes $\widetilde{S}$.

3.1.2 Model

In the construction, private keys will be identified with a set S of descriptive attributes. A party that wishes to encrypt a message will specify through an access tree structure, a policy that private keys must satisfy in order to decrypt.

Each interior node of the tree is a threshold gate and the leaves are associated with attributes. (It is to be noted that this setting is very expressive. For example, a tree can be represented with 'AND' and 'OR' gates by using, respectively, 2 of 2 and 1 of 2 threshold gates.) A user will be able to decrypt a ciphertext with a given key if and only if there is an assignment of attributes from the private key to nodes of the tree such that the tree is satisfied.

Access tree $\mathcal{T}$

Let $\mathcal{T}$ be a tree representing an access structure. Each non-leaf node of the tree represents a threshold gate, described by its children and a threshold value. If num_x is the number of children of a node x and k_x is its threshold value, then $0 < k_x \leq num_x$. When $k_x = 1$, the threshold gate is an OR gate and when $k_x = num_x$, it is an AND gate. Each leaf node x of the tree is described by an attribute and a threshold value $k_x = 1$.

To facilitate working with the access trees, a few functions have been defined. The parent of the node x in the tree is denoted by $parent(x)$. The function $att(x)$ is defined only if x is a leaf node and denotes the attribute associated with the leaf node x in the tree. The access tree $\mathcal{T}$ also defines an ordering between the children of every

node, that is, the children of a node are numbered from 1 to *num*. The function $index(x)$ returns such a number associated with the node x, where the index values are uniquely assigned to nodes in the access structure for a given key in an arbitrary manner.

Satisfying an access tree

Let $\mathcal{T}$ be an access tree with root r. $\mathcal{T}_x$ denotes the subtree of $\mathcal{T}$ rooted at the node x. Hence $\mathcal{T}$ is the same as $\mathcal{T}_r$. If a set of attributes γ satisfies the access tree $\mathcal{T}_x$, it is denoted as $\mathcal{T}_x(\gamma) = 1$. $\mathcal{T}_x(\gamma)$ is computed recursively as follows. If x is a non-leaf node, evaluate $\mathcal{T}_{x'}(\gamma)$ for all children x' of node x. $T_x(\gamma)$ returns 1 if and only if at least k_x children return 1. If x is a leaf node, then $T_x(\gamma)$ returns 1 if and only if $att(x) \in \gamma$.

3.1.3 Construction

Let $\mathbb{G}_0$ be a bilinear group of prime order p, and let g be a generator of $\mathbb{G}_0$. In addition, let $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$ denote the bilinear map. A security parameter, κ, will determine the size of the groups. The Lagrange coefficient $\triangle_{i,S}$ for $i \in \mathbb{Z}_p$ and a set, S, of elements in $\mathbb{Z}_p$ is also defined: $\triangle_{i,S} = \prod_{j \in S, j \neq i} \frac{x-j}{i-j}$. Additionally, a hash function $H : \{0,1\}^* \to \mathbb{G}_0$ is employed that will be modeled as a random oracle. The function will map any attribute described as a binary string to a random group element.

Setup. The setup algorithm will choose a bilinear group $\mathbb{G}_0$ of prime order p with generator g. Next, it will choose two random exponents $\alpha, \beta \in \mathbb{Z}_p$. The public key is published as:

$$PK = \mathbb{G}_0, g, h = g^{\beta}, f = g^{\frac{1}{\beta}}, e(g,g)^{\alpha}$$

and the master key $MK = (\beta, g\alpha)$. (Note that f is used only for delegation.)

Encrypt($PK, M, \mathcal{T}$). The encryption algorithm encrypts a message M under the tree access structure $\mathcal{T}$. The algorithm first chooses a polynomial q_x for each node x (including the leaves) in the tree $\mathcal{T}$. These polynomials are chosen in the following way in a topdown manner, starting from the root node R. For each node x in the tree, set the degree d_x of the polynomial q_x to be one less than the threshold value k_x of that node, that is, $d_x = k_x - 1$.

Starting with the root node R the algorithm chooses a random $s \in \mathbb{Z}_p$ and sets $q_R(0) = s$. Then, it chooses d_R other points of the polynomial q_R randomly to define it completely. For any other node x, it sets $q_x(0) = q_{parent(x)}(index(x))$ and chooses

d_x other points randomly to completely define q_x.

Let Y be the set of leaf nodes in $\mathcal{T}$. The ciphertext is then constructed by giving the tree access structure $\mathcal{T}$ and computing

$$CT = (\mathcal{T}, \widetilde{C} = Me(g,g)^s, C = h^s, \forall y \in Y : C_y = g^{q_y(0)}, C'_y = H(att(y))^{q_y(0)}).$$

KeyGen(MK,S). The key generation algorithm will take as input a set of attributes S and output a key that identifies with that set. The algorithm first chooses a random $r \in \mathbb{Z}_p$, and then random $r_j \in \mathbb{Z}_p$ for each attribute $j \in S$. Then it computes the key as

$$SK = (D = g^{\frac{(\alpha+r)}{\beta}}, \forall j \in S : D_j = g^r \cdot H(j)^{r_j}, D'_j = g^{r_j}).$$

Delegate($SK, \widetilde{D}$). The delegation algorithm takes in a secret key SK, which is for a set S of attributes, and another set $\widetilde{S}$ such that $\widetilde{S} \subseteq S$. The secret key is of the form $SK = (D, \forall j \in S : D_j, D'_j)$. The algorithm chooses random $\widetilde{r}$ and $\widetilde{r}_k \forall k \in \widetilde{S}$. Then it creates a new secret key as

$$\widetilde{SK} = (\widetilde{D} = Df^{\widetilde{r}}, \forall k \in \widetilde{S} : \widetilde{D}_k = D_k g^{\widetilde{r}} H(k)^{\widetilde{r}_k}, \widetilde{D}'_k = D'_k g^{\widetilde{r}_k}).$$

The resulting secret key $\widetilde{SK}$ is a secret key for the set $\widetilde{S}$. Since the algorithm re-randomizes the key, a delegated key is equivalent to one received directly from the authority.

Decrypt(CT, SK). The decryption procedure is defined as a recursive algorithm. A recursive algorithm DecryptNode(CT, SK, x) is defined that takes as input a ciphertext $CT = (\mathcal{T}, \widetilde{C}, C, \forall y \in Y : C_y, C'_y)$, a private key SK, which is associated with a set S of attributes, and a node x from $\mathcal{T}$. If the node x is a leaf node, then let $i = att(x)$ and define as follows: If $i \in S$, then

$$Decrypt(CT, SK, x)$$

$$= \frac{e(D_i, C_x)}{e(D'_i, C'_x)}$$

$$= \frac{e(g^r \cdot H(i)^{r_i}, h^{q_x(0)})}{e(g^{r_i}, H(i)^{q_x(0)})}$$

$$= e(g,g)^{rq_x(0)}$$

If $i \notin S$, the we define $Decrypt(CT, SK, x) = \perp$.

Consider the recursive case when x is a non-leaf node. The algorithm DecryptNode(CT, SK, x) then proceeds as follows: For all nodes z that are children of x, it calls DecryptNode(CT, SK, z) and stores the output as F_z. Let S_x be an arbitrary k_x-sized set of child nodes z such that $F_z = \perp$. If no such set exists then the node was not satisfied and the function returns $\perp$.

Otherwise, we compute

$$F_x = \prod_{z \in S_x} F_z^{\triangle_{i,S'_x}(0)}$$
$$= \prod_{z \in S_x} (e(g,g)^{r \cdot q_z(0)})^{\triangle_{i,S'_x}(0)}$$
$$= \prod_{z \in S_x} (e(g,g)^{r \cdot q_{parent(z)}(index(z))})^{\triangle_{i,S'_x}(0)} \text{ (by construction)}$$
$$= \prod_{z \in S_x} e(g,g)^{r \cdot q_x(i) \cdot \triangle_{i,S'_x}(0)}$$
$$= e(g,g)^{r \cdot q_x(0)} \text{ (using polynomial interpolation)}$$

and return the result.

Now that the function DecryptNode has been defined, the decryption algorithm can be defined using DecryptNode. The algorithm begins by simply calling the function on the root node R of the tree $\mathcal{T}$. If the tree is satisfied by S we set $A = DecryptNode(CT, SK, r) = e(g,g)^{rq_R(0)} = e(g,g)^{rs}$. The algorithm now decrypts by computing

$$\widetilde{C}/(e(C,D)/A) = \widetilde{C}/(e(h^s, g^{\frac{\alpha+r}{\beta}})/e(g,g)^{rs}) = M.$$

3.2 Revocation Schemes

User revocation is of paramount importance for dynamic groups, where the membership status of users to groups is constantly changing. Mediating access control for dynamic groups can be quite a challenge. In this section, we review the widely used revocation scheme of Naor and Pinkas [3].

3.2.1 Revocation scheme of Naor and Pinkas

This scheme consists of two phases:

Initialization: The group controller generates a random polynomial P of degree t over $\mathbb{Z}_p$. It sends a personal key $< I_u, P(I_u) >$ to each user u with identity I_u. This

process is performed once for all future revocations.

Revocation: The group controller learns the identities of t users $I_{u_1}, \cdots, I_{u_t}$ to revoke. It then chooses a random r, and sets the new key to be $g^{rP(0)}$, which would be unknown to revoked users. It broadcasts the message $g^r, < I_{u_1}, g^{rP(I_{u_1})} >, \cdots, < I_{u_t}, g^{rP(I_{u_t})} >$ encrypted with current group key. Each non revoked user can compute $g^{rP(I_u)}$ and combine it with the broadcast values to obtain $g^{rP(0)}$ using Lagrange's interpolation.

Lagrange's interpolation formula for a polynomial P of degree t from its $t+1$ points $x_0, \cdots, x_t$, is

$$P(0) = \sum_{i=0}^{t} \lambda_i P(x_i)$$

where λ_i's are Lagrange coefficients which depend on the x_i's, i.e.,

$\lambda_i = \prod_{j \neq i} \frac{x_i}{x_i - x_j}$. Therefore,

$$g^{rP(0)} = g^{r \sum_{i=0}^{t} \lambda_i P(x_i)} = \prod_{i=0}^{t} g^{r \lambda_i P(x_i)}$$

and knowing $t+1$ pairs $< I_u, g^{rP(I_u)} >$ allows computing $g^{rP(0)}$.

3.3 Comparison Criterion

The constraint on an integer attribute A_t can be represented with the interval $[t_i, t_j]$, where $[t_i, t_j]$ is a range (or interval) denoting the lower (e.g. beginning time) and upper (e.g. ending time) bounds for the instants in A_t. On the other hand, in order to realize comparison-based access control, a user is also assigned a digital certificate (called access privilege) which includes an integer attribute A_t. Given a range constraint $[t_i, t_j]$ and an access privilege $[t_a, t_b]$ on the same attribute A_t, the following criterion must be satisfied:

Definition 7 *Comparison Criterion: Given an access constraint $t_i \leq A_t \leq t_j$ for the protected resources and a privilege $t_a \leq A_t \leq t_b$ in the users certificate, secure data access control must guarantee that the user can be permitted to access the resources if and only if $[t_i, t_j] \cap [t_a, t_b] \neq \emptyset$.*

beginfigure[hbt]

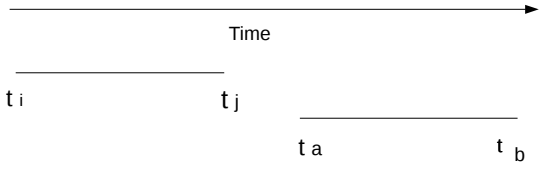


Fig 1: Non overlapping temporal attribute (Decryption not possible with these attribute)

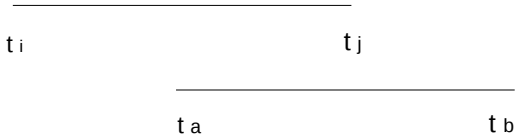


Fig 2: Overlapping temporal attribute (Decryption possible with these attribute)

3.4 Fine-grained Access Control with Comparison

In mathematics, the ordering imposed on a set of elements U is said to be a total order if and only if every two elements are comparable in U. The set of integers, ordered usually by $\leq, \geq$ (or $<, >$) relations, is totally ordered and so are the subsets of natural numbers and rational numbers. As a result, attributes such as level, time, and, position, also satisfy the total order, which can be mapped into consecutive integers. Therefore, we consider the values of these attributes as a countable set constituted in the range $[0, Z]$, $U = \{t_1, \cdots, t_T\} \subseteq [0, Z]$. Based on this ordering relation on U, an attribute-based access control with comparison operations is defined as follows:

- $\mathcal{A}$: the set of attributes $\mathcal{A} = \{A_1, \cdots, A_m\}$;
- $A_k(t_i, t_j)$: the range constraint of attribute A_k on $[t_i, t_j]$, i.e., $t_i \leq A_k \leq t_j$;
- $\mathcal{P}$: the access control policy expressed as a Boolean function on AND/OR logical operations, generated by the grammar: $\mathcal{P} ::= A_k(t_i, t_j) | \mathcal{P}$ AND $\mathcal{P} | \mathcal{P}$ OR $\mathcal{P}$; and
- $\mathcal{L}$: the access privilege assigned into the users certificate, generated by $\mathcal{L} ::= \{A_k(t_a, t_b)\}_{A_k \in \mathcal{A}}$.

3.5 Temporal Access Control Scheme

Temporal access control with user revocation [24] encrypts and stores data in clouds in such a way that only authorized users are able to decrypt it within a specified time period(i.e if the comparision criteria fulfill for all the temporal attributes).

3.5.1 Entities involved

We consider a cloud storage system with five types of entities: System Manager, Data Owners, Data Users, the Cloud Server, and the Proxy Server. The system manager defines the system parameters. The data owners define the access policy and encrypt data under these policies before hosting it on the cloud server. The cloud server stores the encrypted data. The proxy server performs partial decryption. It cannot calculate the plaintext from the partially encrypted data and is used to outsource the heavy decryption task from users. The revocation is also delegated to the proxy server. The data users refer to users who want to access the encrypted data. Thus, only those users who possess attributes that satisfy the access policy will be able to successfully decrypt the data but will also require help from the proxy server. Each user is assigned a user ID along with an access privilege, which contains attributes along with their time range of validity. We assume that the cloud service provider (CSP) and proxy server are honest but curious, that is, they will discharge their duty honestly but will try to deduce as much information as possible based on inputs.

The detail of the scheme will be in the section Scheme-3 of Three Basic Temporal Access Scheme chapter.

3.5.2 drawback of the Scheme

This naive approach has the following shortcomings:

- It cannot perform the key generation in distributed way. There will be only one key generation authority who is responsible for generation of key for all the attribute. So this will become inefficient when number of attribute set is large
- As encryption is directly proportional to the number of attributes by which the message is going to get encrypted. So there will be a large computation overhead when the number of attribute is huge. which is really a big problem for low powered device like mobile phone.
- This scheme does not have any formal security proof

  Our proposed scheme overcome all of these three shortcomings.

Chapter 4

Three Basic Temporal Access Scheme

4.1 Introduction

In this chapter, we present three schemes which are basically step by step improvement starting from the basic temporal access scheme. First, We define basic temporal access scheme then we will improve that twice by adding extra functionalities.

4.2 Scheme-1:Basic Temporal Access Control Scheme

Here We propose a basic temporal access control scheme to protect and selectively access data in clouds. This is our first scheme on temporal access control. Our this scheme encrypts and stores data in clouds in such a way that only authorized users are able to decrypt it.
There will be two types of attribute.First one is non-temporal attribute for which a user can decrypt the content as long as the data exist in cloud. And another is temporal attribute for which data can be accessed within some specific time period. This scheme is implemented Without any proxy for decryption and without the facility of revocation.The entities described in 3.5.1 are all involved here except proxy server. Later on we will see more complicated schemes with extra facilities and extra randomization on this basic scheme.

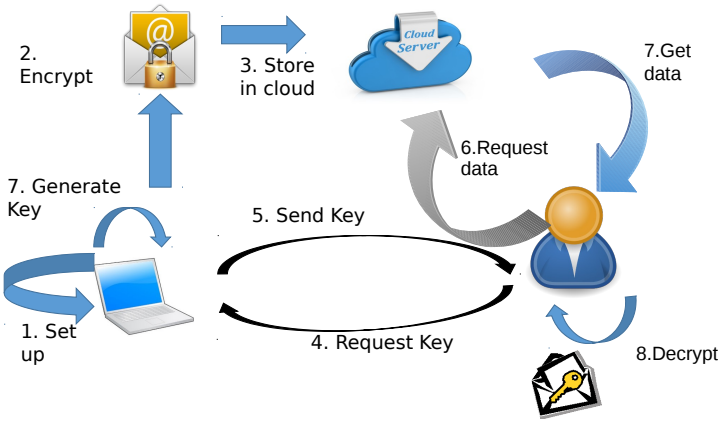


Figure 4.1: A Basic Temporal Access Control Scheme

4.2.1 Framework

A brief description of each of these algorithms is as follows:

- Setup$(1^\kappa) \rightarrow (MK, PK)$: Takes a security parameter κ as input, outputs the master secret key MK and the public key PK.
- KeyGen$(MK, u_k, \mathcal{L}) \rightarrow SK_\mathcal{L}$: Takes the user ID u_k, the access privilege $\mathcal{L}$, and MK as input, outputs the users private key $SK_\mathcal{L}$.
- Encrypt$(PK, M, \mathcal{P}) \rightarrow (\mathcal{H}_\mathcal{P}, C')$: Takes a comparable access policy $\mathcal{P}$, public key PK and message M as input, outputs the ciphertext header $\mathcal{H}_\mathcal{P}$ and ciphertext C'.
- Decrypt$(SK_\mathcal{L}, \mathcal{H}_\mathcal{P}, C') \rightarrow M$: Takes the users private key $SK_\mathcal{L}$, Ciphertext header $\mathcal{H}_\mathcal{P}$, and the ciphertext C' as input, outputs the message M.

4.2.2 Construction

We provide detailed implementations of the algorithms mentioned in Section 5.3.1 and present our novel construction to enforce user revocation.

- *Setup*$(1^\kappa) \rightarrow (MK, PK)$
  This is run by the system manager. Let κ be the security parameter. The setup

algorithm takes κ (which depends on the application) and outputs the master secret key MK and public key PK. We use bilinear map $\mathbb{S} = (\mathbb{G}, \mathbb{G}_T, e(\cdot,\cdot))$ of composite order $n = s'n'$ with two subgroups $\mathbb{G}_{s'}$ and $\mathbb{G}_{n'}$ of $\mathbb{G}$. Random generators $w \in \mathbb{G}$, $g \in \mathbb{G}_{s'}$, and $\varphi, \bar{\varphi} \in \mathbb{G}_{n'}$ are chosen. Two random numbers $\lambda, \mu \in \mathbb{Z}_n^*$ are selected, to implement forward and backward derivation functions. By Proposition 1, we have $e(g,\varphi) = e(g,\bar{\varphi}) = 1$, but $e(g,w) \neq 1$. A hash function $H : \{0,1\}^* \to \mathbb{G}$ is used to map an attribute to a binary string which represents a random group element. A randomly generated polynomial P of degree c (the maximum number of revoked users) over $\mathbb{Z}_n$ will be employed to enforce revocation. Next, this algorithm chooses two random exponents $\alpha, \beta \in \mathbb{Z}_n^*$ and outputs the public key

$$PK = (\mathbb{S}, n, g, h, \zeta, \eta, w, \varphi, \bar{\varphi}, \lambda, \mu, H(\cdot)),$$

where,

$$h = w^{\beta}, \zeta = e(g,w)^{\alpha}, \eta = g^{\frac{1}{\beta}},$$

and the master key

$$MK = (g^{\alpha}, \beta, n', P).$$

- $KeyGen(MK, u_k, \mathcal{L}) \to SK_{\mathcal{L}}$

Given a user with ID u_k[1] and access structure $\mathcal{L}$ on a set of attributes $S = \{A_t\} \subseteq \mathcal{A}$, this algorithm chooses a unique τ_k for each user u_k. Assume that the user u_k is assigned a temporal attribute $A_t \in \mathcal{L}$ with the constraint $A_t[t_a, t_b]$ and non-temporal attribute $B_t \in \mathcal{L}$. This algorithm chooses a random $r \in \mathbb{Z}$ and sets the user's attribute key as

$$(D_t, D_{t_a}^{'}, \bar{D}_{t_b}^{'}, D_t^{''},)_{A_t[t_a,t_b]\in\mathcal{L}} \text{and}$$

$$(DB_t)_{B_t\in\mathcal{L}}$$

where,

$$\begin{aligned} D_t &= g^{\tau_k} H(A_t)^r, \\ D_{t_a}^{'} &= (v_{t_a})^r, \\ \bar{D}_{t_b}^{'} &= (\bar{v}_{t_b})^r, \\ D_t^{''} &= w^r, \\ DB_t &= g^{\tau_k} H(B_t)^r, \end{aligned}$$

and,

$$v_{t_a} = \varphi^{\lambda^{t_a}}, \bar{v}_{t_b} = \bar{\varphi}^{\mu^{Z-t_b}} \in \mathbb{G}_{n'},$$

[1]We assume that each user ID $u_k \in \mathbb{Z}_l$ where $l = min(s, p', q')$, so that, for every pair $(i,j), i \neq j, u_i - u_j \in \mathbb{Z}_n^*$. As a result, its inverse $(u_i - u_j)^{-1}$, which is needed in the *ProxyRekey* phase will exist and the secret can be successfully recovered.

Finally, this algorithm outputs the user's private key

$$SK_{\mathcal{L}} = (D = g^{\frac{(\alpha+\tau_k)}{\beta}}, \{(D_t, D_{t_a}^{'}, \bar{D}_{t_b}^{'}, D_t^{''})\}_{A_t[t_a,t_b]\in\mathcal{L}}, \{(DB_t)\}_{B_t\in\mathcal{L}}).$$

(4.1)

- *Encrypt*$(PK, \mathcal{P}, M) \rightarrow (\mathcal{H}_{\mathcal{P}}, C')$
  Let $\mathcal{T}$ be the access tree for the access policy $\mathcal{P}$. Let s be a secret in $\mathbb{Z}_N$ for the access tree $\mathcal{T}$, and x, y are chosen such that $x + y = q_{A_t}(0)$, which is the secret share at leaf node corresponding to attribute A_t. Please refer to Section 4, [12] for a detailed meaning of $q_{A_t}(0)$. Given an access tree $\mathcal{T}$, the ciphertext is composed of a ciphertext header,

  $$\mathcal{H}_{\mathcal{P}} = (\mathcal{T}, C = h^s, \{(\bar{E}_{t_i}, E_{t_i}^{'}, E_{t_j}, E_{t_j}^{'})\}_{A_t[t_i,t_j]\in\mathcal{P}}, \{(EB_t^{\prime})\}_{B_t\in\mathcal{P}}). \tag{4.2}$$

  and a ciphertext $C' = Me(g, w)^{s\alpha}$, where,

  $(\bar{E}_{t_i}, E_{t_i}^{'}, E_{t_j}, E_{t_j}^{'}) = ((\bar{v}_{t_i}w)^x, H(A_t)^x, (v_{t_j}w)^y, H(A_t)^y)$
  and $(EB_t^{\prime}, \bar{EB}_t) = (H(B_t)^{q_{B_t}(0)}, w^{q_{B_t}(0)})$.

- *Decrypt*$(SK_{\mathcal{L}}, \mathcal{L}') \rightarrow M$

  Given the private key $SK_{\mathcal{L}}$ and a specified $\mathcal{L}'$, this algorithm checks whether, for every $A_t[t_a, t_b] \in \mathcal{L}$ and $A_t[t_i, t_j] \in \mathcal{L}'$, $t_a \leq t_j$ and $t_b \geq t_i$ is true for each attribute $A_t \in \mathcal{L}'$.
  If true, the user computes,

  $$D_{t_j}^{'} \leftarrow f_{t_a\leq t_j}(D_{t_a}^{'}) \cdot D_t^{''} = (v_{t_j}w)^r$$
  $$\bar{D}_{t_i}^{'} \leftarrow \bar{f}_{t_b\geq t_i}(\bar{D}_{t_b}^{'}) \cdot D_t^{''} = (\bar{v}_{t_i}w)^r$$

  Finally, it outputs $\widetilde{SK}_{\mathcal{L}'}$ as the derivation key for $\mathcal{L}'$.
  On receiving the private key $\widetilde{SK}_{\mathcal{L}'}$, and the ciphertext header $\mathcal{H}_{\mathcal{P}}$, this algorithm checks whether each range attribute $A_t[t_i, t_j] \in \mathcal{L}'$ is consistent with $A_t[t_i, t_j] \in \mathcal{P}$. If true, the secret share $q_{A_t}(0)$ over $\mathbb{G}_T$ is reconstructed as follows:-

$$
\begin{aligned}
F_1 \longleftarrow & \frac{e(D_t, E_{t_j})}{e(D'_{t_j}, E'_{t_j})} \\
= & \frac{e(g^{\tau_k} H(A_t)^r, (v_{t_j} w)^y)}{e((v_{t_j} w)^r, H(A_t)^y)} \\
= & \frac{e(g^{\tau_k}, (v_{t_j} w)^y) \cdot e(H(A_t)^r, (v_{t_j} w)^y)}{e((v_{t_j} w)^r, H(A_t)^y)} \\
= & e(g^{\tau_k}, (v_{t_j} w)^y) \\
= & e(g^{\tau_k}, v_{t_j}^y) \cdot e(g^{\tau_k}, w^y) \\
= & e(g^{\tau_k}, w)^y
\end{aligned}
$$

As $e(g, v_{t_j}) = 1$ for composite group property.
Similarly,

$$
\begin{aligned}
F_2 \longleftarrow & \frac{e(D_t, \bar{E}_{t_i})}{e(\bar{D'}_{t_i}, E'_{t_i})} \\
= & \frac{e(g^{\tau_k} H(A_t)^r, (\bar{v}_{t_i} w)^x)}{e((\bar{v}_{t_i} w)^r, H(A_t)^x)} \\
= & \frac{e(g^{\tau_k}, (\bar{v}_{t_i} w)^x) \cdot e(H(A_t)^r, (\bar{v}_{t_i} w)^x)}{e((\bar{v}_{t_i} w)^r, H(A_t)^x)} \\
= & e(g^{\tau_k}, (\bar{v}_{t_i} w)^x) \\
= & e(g^{\tau_k}, \bar{v}_{t_i}^x) \cdot e(g^{\tau_k}, w^x) \\
= & e(g^{\tau_k}, w)^x
\end{aligned}
$$

For $B_t \in \mathcal{L}'$ is consistent with $B_t \in \mathcal{P}$, then the secret share $q_{B_t}(0)$ of s over $\mathbb{G}_T$ is reconstructed as,

$$
\begin{aligned}
F_3 \longleftarrow & \frac{e(DB_t, \bar{EB}_t)}{e(D''_t, EB'_t)} \\
= & \frac{e(g^{\tau_k} H(B_t)^r, (w^{q_{B_t}(0)})}{e(w^r, H(B_t)^{q_{B_t}(0)})} \\
= & \frac{e(g^{\tau_k}, w^{q_{B_t}(0)}) \cdot e(H(B_t)^r, w^{q_{B_t}(0)})}{e(w^r, H(B_t)^{q_{B_t}(0)})} \\
= & e(g^{\tau_k}, w)^{q_{B_t}(0)}
\end{aligned}
$$

$F_t = F_1 \cdot F_2 = e(g^{\tau_k}, w)^{q_{A_t}(0)}$

where, $e(g^{\tau_k}, v_{t_j}^y) = e(g^{\tau_k}, \bar{v}_{t_i}^x) = 1$ because $g^{\tau_k} \in \mathbb{G}_{s'}$ and $v_{t_j}^y, \bar{v}_{t_i}^x \in \mathbb{G}_{n'}$. Next, the value of $T = e(g^{\tau_k}, w)^s$ is computed from $\{e(g^{\tau_k}, w)^{q_{A_i}(0)}\}_{A_i \in \mathcal{P}}$ and $\{e(g^{\tau_k}, w)^{q_{B_i}(0)}\}_{B_i \in \mathcal{P}}$ by using the recursive DecryptNode algorithm described in Bethencourt et al. [12]. Finally, the new ciphertext header $\widetilde{\mathcal{H}}_{\mathcal{P}} = (C, T)$ is returned.

Once the ciphertext header $\widetilde{\mathcal{H}}_{\mathcal{P}} = (C, T) = (w^{\beta s}, e(g^{\tau_k}, w)^s)$ is formed, the secret δ is used to compute $D' = D \cdot \eta = g^{\frac{(\alpha+\tau_k)}{\beta}} g^{\frac{1}{\beta}} = g^{\frac{(\alpha+\tau_k)}{\beta}}$.

Let $ek = \dfrac{e(C, D')}{T} = \dfrac{e(g^{\frac{(\alpha+\tau_k)}{\beta}}, w^{\beta s})}{e(g^{\tau_k}, w)^s} = e(g^{\alpha}, w)^s$.
Finally, the plaintext message is computed by
$M = C'/ek$.

4.3 Scheme-2:Temporal Access Control with Revocation Added

This is the extension of the previous scheme where we have added revocation of authorized user. In this scheme the data owner can revocake any authorized user from accessing the encrypted content at any point of time. The entities described in 3.5.1 are all involved here except proxy server.

4.3.1 Framework

A brief description of each of these algorithms is as follows:

- Setup$(1^\kappa) \rightarrow (MK, PK)$: Takes a security parameter κ as input, outputs the master secret key MK and the public key PK.
- KeyGen$(MK, u_k, \mathcal{L}) \rightarrow SK_{\mathcal{L}}$: Takes the user ID u_k, the access privilege $\mathcal{L}$, and MK as input, outputs the users private key $SK_{\mathcal{L}}$.
- Encrypt phase-1$(PK, M, \mathcal{P}) \rightarrow (\mathcal{H}_{\mathcal{P}}, C')$: Takes a comparable access policy $\mathcal{P}$, public key PK and message M as input, outputs the ciphertext header $\mathcal{H}_{\mathcal{P}}$ and ciphertext C'.
- Encrypt phase-2$(PK, MK, RL) \rightarrow (\widetilde{\mathcal{H}_{\mathcal{P}}}, RIV)$: Takes the public key PK, master key MK, and the revocation list RL as inputs, outputs modified ciphertext $\widetilde{\mathcal{H}_{\mathcal{P}}}$, which contains the information necessary to enable the data owner to partially decrypt the ciphertext.

- Delegate$(SK_{\mathcal{L}}, \mathcal{L}') \rightarrow \widetilde{SK}_{\mathcal{L}'}$: Takes a private key $SK_{\mathcal{L}}$ and a specified privilege requirement $\mathcal{L}'$ as input, outputs a derivation key $\widetilde{SK}_{\mathcal{L}'}$, if each attribute in $\mathcal{L}$ and $\mathcal{L}'$ match.
- Decrypt:-1$(\widetilde{SK}_{\mathcal{L}'}, \mathcal{H}_{\mathcal{P}}, RIV) \rightarrow \widetilde{\mathcal{H}_{\mathcal{P}}}$: Takes a users private key $\widetilde{SK}_{\mathcal{L}'}$, RIV and the ciphertext header $\mathcal{H}_{\mathcal{P}}$ as input, outputs a new ciphertext header $\widetilde{\mathcal{H}_{\mathcal{P}}}$ if $\mathcal{L}'$ satisfies the range constraints of $\mathcal{P}$.
- Decrypt:-2r$(SK_{\mathcal{L}}, \widetilde{\mathcal{H}_{\mathcal{P}}}, C') \rightarrow M$: Takes the users private key $SK_{\mathcal{L}}$, Ciphertext header $\widetilde{\mathcal{H}}_{\mathcal{P}}$, and the ciphertext C' as input, outputs the message M.

4.3.2 Construction

We provide detailed implementations of the algorithms mentioned in Section 5.3.1 and present our novel construction to enforce user revocation.

- *Setup*$(1^{\kappa}) \rightarrow (MK, PK)$
  This is run by the system manager. Let κ be the security parameter. The setup algorithm takes κ (which depends on the application) and outputs the master secret key MK and public key PK. We use bilinear map $\mathbb{S} = (\mathbb{G}, \mathbb{G}_T, e(\cdot,\cdot))$ of composite order $n = s'n'$ with two subgroups $\mathbb{G}_{s'}$ and $\mathbb{G}_{n'}$ of $\mathbb{G}$. Random generators $w \in \mathbb{G}$, $g \in \mathbb{G}_{s'}$, and $\varphi, \bar{\varphi} \in \mathbb{G}_{n'}$ are chosen. Two random numbers $\lambda, \mu \in \mathbb{Z}_n^*$ are selected, to implement forward and backward derivation functions. By Proposition 1, we have $e(g, \varphi) = e(g, \bar{\varphi}) = 1$, but $e(g, w) \neq 1$. A hash function $H : \{0,1\}^* \rightarrow \mathbb{G}$ is used to map an attribute to a binary string which represents a random group element. A randomly generated polynomial P of degree c (the maximum number of revoked users) over $\mathbb{Z}_n$ will be employed to enforce revocation. Next, this algorithm chooses two random exponents $\alpha, \beta \in \mathbb{Z}_n^*$ and outputs the public key

  $$PK = (\mathbb{S}, n, g, h, \zeta, \eta, w, \varphi, \bar{\varphi}, \lambda, \mu, H(\cdot)),$$

  where,

  $$h = w^{\beta}, \zeta = e(g, w)^{\alpha}, \eta = g^{\frac{1}{\beta}},$$

  and the master key

  $$MK = (g^{\alpha}, \beta, n', P).$$

- *KeyGen*$(MK, u_k, \mathcal{L}) \rightarrow SK_{\mathcal{L}}$

  Given a user with ID u_k[1] and access structure $\mathcal{L}$ on a set of attributes $S = \{A_t\} \subseteq \mathcal{A}$, this algorithm chooses a unique τ_k for each user u_k. Assume that

[1]We assume that each user ID $u_k \in \mathbb{Z}_l$ where $l = min(s, p', q')$, so that, for every pair $(i, j), i \neq j, u_i - u_j \in \mathbb{Z}_n^*$. As a result, its inverse $(u_i - u_j)^{-1}$, which is needed in the *ProxyRekey* phase will exist and the secret can be successfully recovered.

the user u_k is assigned a temporal attribute $A_t \in \mathcal{L}$ with the constraint $A_t[t_a, t_b]$ and non-temporal attribute $B_t \in \mathcal{L}$. This algorithm chooses a random $r \in \mathbb{Z}$ and sets the user's attribute key as

$$(D_t, D_{t_a}^{'}, \bar{D}_{t_b}^{'}, D_t^{''}, R_{t_a}^{'}, \bar{R}_{t_b}^{'}, R_t^{''})_{A_t[t_a,t_b]\in\mathcal{L}} \text{and}$$

$$(DB_t, RB_t^{''})_{B_t\in\mathcal{L}}$$

where,

$$\begin{aligned}
&D_t = g^{\tau_k} H(A_t)^{r\widetilde{P}(0)},\\
&D_{t_a}^{'} = (v_{t_a})^r,\\
&\bar{D}_{t_b}^{'} = (\bar{v}_{t_b})^r,\\
&D_t^{''} = w^r,\\
&R_{t_a}^{'} = (D_{t_a}^{'})^{P(u_k)} = (v_{t_a})^{rP(u_k)},\\
&\bar{R}_{t_b}^{'} = (\bar{D}_{t_b}^{'})^{P(u_k)} = (\bar{v}_{t_b})^{rP(u_k)},\\
&R_t^{''} = (D_t^{''})^{P(u_k)} = w^{rP(u_k)},\\
&DB_t = g^{\tau_k} H(B_t)^{r\widetilde{P}(0)},\\
&RB_t^{''} = (D_t^{''})^{P(u_k)} = w^{rP(u_k)},
\end{aligned}$$

and,

$$\begin{aligned}
&v_{t_a} = \varphi^{\lambda^{t_a}}, \bar{v}_{t_b} = \bar{\varphi}^{\mu^{Z-t_b}} \in \mathbb{G}_{n'},\\
&\widetilde{P}(0) = P(0) + 1.
\end{aligned}$$

Finally, this algorithm outputs the user's private key

$$\begin{aligned}
SK_{\mathcal{L}} = (D = g^{\frac{(\alpha+\tau_k)}{\beta}},\ &\{(D_t, D_{t_a}^{'}, \bar{D}_{t_b}^{'}, D_t^{''}, R_{t_a}^{'}, \bar{R}_{t_b}^{'}, R_t^{''})\}_{A_t[t_a,t_b]\in\mathcal{L}},\\
&\{(DB_t, RB_t^{''})\}_{B_t\in\mathcal{L}}).
\end{aligned}$$

- *Encrypt Phase 1* $(PK, \mathcal{P}, M) \rightarrow (\mathcal{H}_{\mathcal{P}}, C')$
  Let $\mathcal{T}$ be the access tree for the access policy $\mathcal{P}$. Let s be a secret in $\mathbb{Z}_N$ for the access tree $\mathcal{T}$, and x, y are chosen such that $x + y = q_{A_t}(0)$, which is the secret share at leaf node corresponding to attribute A_t. Please refer to Section 4, [12] for a detailed meaning of $q_{A_t}(0)$. Given an access tree $\mathcal{T}$, the ciphertext is composed of a ciphertext header,

$$\mathcal{H}_{\mathcal{P}} = (\mathcal{T}, C = h^s, \{(\bar{E}_{t_i}, E_{t_i}^{'}, E_{t_j}, E_{t_j}^{'})\}_{A_t[t_i,t_j]\in\mathcal{P}}, \{(EB_t, EB_t')\}_{B_t\in\mathcal{P}}). \quad (4.3)$$

  and a ciphertext $C' = Me(g, w)^{s\alpha}$, where,

$$\begin{aligned}
&(\bar{E}_{t_i}, E_{t_i}^{'}, E_{t_j}, E_{t_j}^{'}) = ((\bar{v}_{t_i} w)^x, H(A_t)^x, (v_{t_j} w)^y, H(A_t)^y)\\
&\text{and } (EB_t, EB_t') = (\omega^{q_{B_t}(0)}, H(B_t)^{q_{B_t}(0)}).
\end{aligned}$$

- *Encrypt Phase 2*$(PK, MK, RL) \rightarrow RIV$
  The data owner chooses a polynomial P of degree c, with coefficients in Z_n^*. Let RL be the revocation list and $u_i, i \in \{1, 2, \cdots, c\}$, be the identities of the revoked users. The owner evaluates the polynomial $P(u_i)$ at these points, using the master secret key MK. If there are less than c revoked users, then the owner generates random points x and evaluates $P(x)$ such that x does not correspond to any user's identity. This ensures that the proxy key RIV is of fixed length.

$$RIV = \forall u_i \in RL :< u_i, P(u_i) >$$

  Now the data owner will calculates

$$\lambda_i = \frac{u_k}{u_k - u_i} \prod_{j \neq i} \frac{u_j}{u_j - u_i}, \forall i, j \in \{1, \cdots, c\},$$
$$k \notin \{1, \cdots, c\}$$

  For every attribute $A_t[t_i, t_j] \in \mathcal{P}$, it calculates

$$\begin{aligned} R_{t_i}^{''} &= (E_{t_i}^{'})^{\sum_{i=1}^{c} \lambda_i P(u_i)} = H(A_t)^{x \sum_{i=1}^{c} \lambda_i P(u_i)}, \\ R_{t_j}^{''} &= (E_{t_j}^{'})^{\sum_{i=1}^{c} \lambda_i P(u_i)} = H(A_t)^{y \sum_{i=1}^{c} \lambda_i P(u_i)}, \\ RB_t^{'''} &= (EB_t^{'})^{\sum_{i=1}^{c} \lambda_i P(u_i)} = H(B_t)^{q_{B_t}(0) \sum_{i=1}^{c} \lambda_i P(u_i)}. \end{aligned}$$

  Now the new ciphertext will be

$$\widetilde{\mathcal{H}_{\mathcal{P}}} = (\mathcal{T}, C = h^s, \{(\bar{E}_{t_i}, E_{t_i}^{'}, E_{t_j}, E_{t_j}^{'}, R_{t_i}^{''}, R_{t_j}^{''})\}_{A_t[t_i,t_j] \in \mathcal{P}}, \{(EB_t, EB_t^{'}, , RB_t^{'''})\}_{B_t \in \mathcal{P}}).$$

  Since the user's private key $SK_{\mathcal{L}}$ is blinded by $\widetilde{P}(0)$, it additionally needs $R_{t_i}^{''}$, $R_{t_j}^{''}$ for decryption. The data owner also computes λ_k and gives it to the user with ID u_k.

- *Delegate*$(SK_{\mathcal{L}}, \mathcal{L}') \rightarrow \widetilde{SK}_{\mathcal{L}'}$

  Given the private key $SK_{\mathcal{L}}$ and a specified $\mathcal{L}'$, this algorithm checks whether, for every $A_t[t_a, t_b] \in \mathcal{L}$ and $A_t[t_i, t_j] \in \mathcal{L}'$, $t_a \leq t_j$ and $t_b \geq t_i$ is true for each

attribute $A_t \in \mathcal{L}'$.
If true, the user computes,

$$
\begin{aligned}
D'_{t_j} &\leftarrow f_{t_a \le t_j}(D'_{t_a}) \cdot D''_t = (v_{t_j} w)^r \\
\bar{D}'_{t_i} &\leftarrow \bar{f}_{t_b \ge t_i}(\bar{D}'_{t_b}) \cdot D''_t = (\bar{v}_{t_i} w)^r \\
R'_{t_j} &\leftarrow f_{t_a \le t_j}(R'_{t_a}) = (v_{t_j})^{rP(u_k)} \\
\bar{R}'_{t_i} &\leftarrow \bar{f}_{t_b \ge t_i}(\bar{R}'_{t_b}) = (\bar{v}_{t_i})^{rP(u_k)}
\end{aligned}
$$

$$
\begin{aligned}
R_{t_j} &= e(R''_t \cdot R'_{t_j}, E'_{t_j})^{\lambda_k} \cdot e(D'_{t_j}, R''_{t_j}) \\
&= e(v_{t_j} w, H(A_t))^{yr(\lambda_k P(u_k) + \sum_{i=1}^{t} \lambda_i P(u_i))} \\
&= e(v_{t_j} w, H(A_t))^{yrP(0)} \\
\bar{R}_{t_i} &= e(R''_t \cdot \bar{R}'_{t_i}, E'_{t_i})^{\lambda_k} \cdot e(\bar{D}'_{t_i}, \bar{R}''_{t_i}) \\
&= e(\bar{v}_{t_i} w, H(A_t))^{xr(\lambda_k P(u_k) + \sum_{i=1}^{t} \lambda_i P(u_i))} \\
&= e(\bar{v}_{t_i} w, H(A_t))^{xrP(0)} \\
RB_t &= e(RB''_t, EB'_t)^{\lambda_k} \cdot e(DB'_t, RB'''_t) \\
&= e(w, H(B_t))^{q_{B_t}(0) rP(0)}
\end{aligned}
$$

Finally, it outputs $\widetilde{SK}_{\mathcal{L}'}$ as the derivation key for $\mathcal{L}'$.

- *Decrypt Phase 1* $(\widetilde{SK}_{\mathcal{L}'}, \mathcal{H}_{\mathcal{P}}) \rightarrow \widetilde{\mathcal{H}}_{\mathcal{P}}$

On receiving the private key $\widetilde{SK}_{\mathcal{L}'}$, and the ciphertext header $\mathcal{H}_{\mathcal{P}}$, this algorithm checks whether each range attribute $A_t[t_i, t_j] \in \mathcal{L}'$ is consistent with $A_t[t_i, t_j] \in \mathcal{P}$. If true, the secret share $q_{A_t}(0)$ over $\mathbb{G}_T$ is reconstructed as follows:-

$$
\begin{aligned}
F_1 \longleftarrow & \frac{e(D_t, E_{t_j})}{R_{t_j} \cdot e(D'_{t_j}, E'_{t_j})} \\
= & \frac{e(g^{\tau_k} H(A_t)^{r\widetilde{P}(0)}, (v_{t_j} w)^y)}{R_{t_j} \cdot e((v_{t_j} w)^r, H(A_t)^y)} \\
= & \frac{e(g^{\tau_k}, (v_{t_j} w)^y) \cdot e(H(A_t)^{r\widetilde{P}(0)}, (v_{t_j} w)^y)}{R_{t_j} \cdot e((v_{t_j} w)^r, H(A_t)^y)} \\
= & \frac{e(g^{\tau_k}, (v_{t_j} w)^y) \cdot e(v_{t_j} w, H(A_t))^{yrP(0)}}{R_{t_j}} \\
= & e(g^{\tau_k}, v_{t_j}^y) \cdot e(g^{\tau_k}, w^y) \\
= & e(g^{\tau_k}, w)^y
\end{aligned}
$$

Similarly,

$$
\begin{aligned}
F_2 \longleftarrow & \frac{e(D_t, \bar{E}_{t_i})}{\bar{R}_{t_i} \cdot e(\bar{D'}_{t_i}, E'_{t_i})} \\
= & \frac{e(g^{\tau_k} H(A_t)^{rP'(0)}, (\bar{v}_{t_i} w)^x)}{\bar{R}_{t_i} \cdot e((\bar{v}_{t_i} w)^r, H(A_t)^x)} \\
= & \frac{e(g^{\tau_k}, (\bar{v}_{t_i} w)^x) \cdot e(H(A_t)^{rP'(0)}, (\bar{v}_{t_i} w)^x)}{\bar{R}_{t_i} \cdot e((\bar{v}_{t_i} w)^r, H(A_t)^x)} \\
= & \frac{e(g^{\tau_k}, (\bar{v}_{t_i} w)^x) \cdot e(\bar{v}_{t_i} w, H(A_t))^{xrP(0)}}{\bar{R}_{t_i}} \\
= & e(g^{\tau_k}, \bar{v}_{t_i}^x) \cdot e(g^{\tau_k}, w^x) \\
= & e(g^{\tau_k}, w)^x
\end{aligned}
$$

For $B_t \in \mathcal{L}'$ is consistent with $B_t \in \mathcal{P}$, then the secret share $q_{B_t}(0)$ of s over $\mathbb{G}_T$ is reconstructed as,

$$
\begin{aligned}
F_3 \longleftarrow & \frac{e(D_t, EB_t)}{RB_t \cdot e(D_t'', EB_t')} \\
= & \frac{e(g^{\tau_k} H(B_t)^{rP'(0)}, w^{q_{B_t}(0)})}{RB_t \cdot e(w^r, H(B_t)^{q_{B_t}(0)})} \\
= & \frac{e(g^{\tau_k}, w^{q_{B_t}(0)}) \cdot e(H(B_t)^{r\widetilde{P}(0)}, w^{q_{B_t}(0)})}{RB_t \cdot e(w^r, H(B_t)^{q_{B_t}(0)})} \\
= & \frac{e(g^{\tau_k}, w^{q_{B_t}(0)}) \cdot e(w, H(B_t))^{q_{B_t}(0) rP(0)}}{RB_t} \\
= & e(g^{\tau_k}, w^{q_{B_t}(0)}) \\
= & e(g^{\tau_k}, w)^{q_{B_t}(0)}
\end{aligned}
$$

$F_t = F_1 \cdot F_2 = e(g^{\tau_k}, w)^{q_{A_t}(0)}$

where, $e(g^{\tau_k}, v_{t_j}^y) = e(g^{\tau_k}, \bar{v}_{t_i}^x) = 1$ because $g^{\tau_k} \in \mathbb{G}_{s'}$ and $v_{t_j}^y, \bar{v}_{t_i}^x \in \mathbb{G}_{n'}$. Next, the value of $T = e(g^{\tau_k}, w)^s$ is computed from $\{e(g^{\tau_k}, w)^{q_{A_i}(0)}\}_{A_i \in \mathcal{P}}$ and $\{e(g^{\tau_k}, w)^{q_{B_i}(0)}\}_{B_i \in \mathcal{P}}$ by using the recursive DecryptNode algorithm described in Bethencourt et al. [12]. Finally, the new ciphertext header $\widetilde{\mathcal{H}}_{\mathcal{P}} = (C, T)$ is returned.

- *Decrypt Phase 2*$(SK_{\mathcal{L}}, \widetilde{\mathcal{H}}_{\mathcal{P}}, C') \rightarrow M$

  On receiving the new ciphertext header $\widetilde{\mathcal{H}}_{\mathcal{P}} = (C, T) = (w^{\beta s}, e(g^{\tau_k}, w)^s)$, Now it will compute $D' = D \cdot \eta = g^{\frac{(\alpha+\tau_k)}{\beta}} g^{\frac{1}{\beta}} = g^{\frac{(\alpha+\tau_k)}{\beta}}$.

  Let $ek = \dfrac{e(C, D')}{T} = \dfrac{e(g^{\frac{(\alpha+\tau_k)}{\beta}}, w^{\beta s})}{e(g^{\tau_k}, w)^s} = e(g^{\alpha}, w)^s$.
  Finally, the plaintext message is computed by
  $M = C'/ek$.

4.4 Scheme-3:Temporal Access Control Scheme with added decryption outsourcing

Here in this scheme I have added decryption outsourcing where a part of decryption process is outsourced to proxy server or any third party. The proxy will partially decrypt the content but it won't be able to decrypt the content fully. So decryptor can rely on untrusted proxy as well for partial encryption. The entities described in 3.5.1 are all involved here.

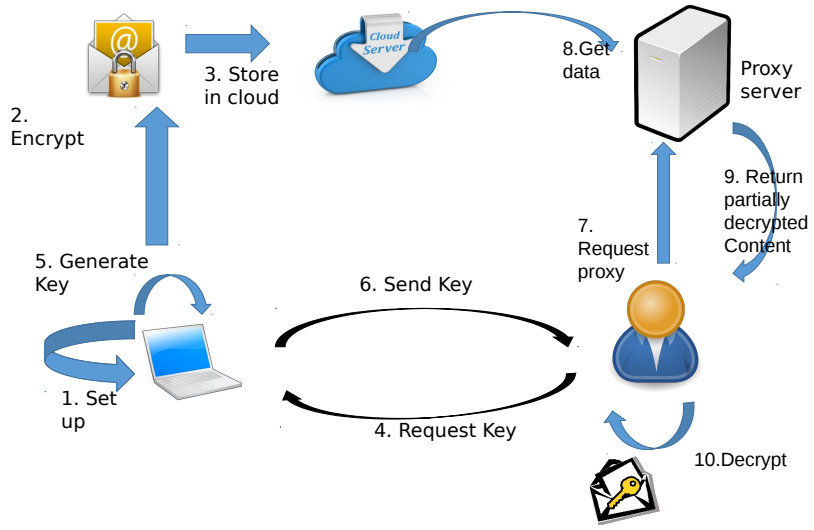


Figure 4.2: With user revocation and decryption outsourcing

4.4.1 Framework

A brief description of each of these algorithms is as follows:

- Setup$(1^\kappa) \rightarrow (MK, PK)$: Takes a security parameter κ as input, outputs the master secret key MK and the public key PK.
- KeyGen$(MK, u_k, \mathcal{L}) \rightarrow SK_\mathcal{L}$: Takes the user ID u_k, the access privilege $\mathcal{L}$, and MK as input, outputs the users private key $SK_\mathcal{L}$.
- Encrypt$(PK, M, \mathcal{P}) \rightarrow (\mathcal{H}_\mathcal{P}, C')$: Takes a comparable access policy $\mathcal{P}$, public key PK and message M as input, outputs the ciphertext header $\mathcal{H}_\mathcal{P}$ and ciphertext C'.
- ProxyRekey$(PK, MK, RL) \rightarrow PXK$: Takes the public key PK, master key MK, and the revocation list RL as inputs, outputs the proxy key PXK, which contains the information necessary to enable the proxy server to partially decrypt the ciphertext.
- Convert$(PXK, \mathcal{H}_\mathcal{P}, u_k) \rightarrow (\lambda_k, \{(R''_{t_i}, R''_{t_j})\}_{A_t[t_i,t_j]\in\mathcal{P}}, RB''_{t\ B_t\in\mathcal{P}})$: Takes the proxy key PXK, ciphertext header $\mathcal{H}_\mathcal{P}$, and user ID u_k as input, outputs components needed to unblind the user's private key $SK_\mathcal{L}$.
- Delegate$(SK_\mathcal{L}, \mathcal{L}') \rightarrow \widetilde{SK}_{\mathcal{L}'}$: Takes a private key $SK_\mathcal{L}$ and a specified privilege requirement $\mathcal{L}'$ as input, outputs a derivation key $\widetilde{SK}_{\mathcal{L}'}$, if each attribute in $\mathcal{L}$ and $\mathcal{L}'$ match.

- DecryptProxy($\widetilde{SK}_{\mathcal{L}'}, \mathcal{H}_{\mathcal{P}}) \rightarrow \widetilde{\mathcal{H}_{\mathcal{P}}}$: Takes a users private key $\widetilde{SK}_{\mathcal{L}'}$ and the ciphertext header $\mathcal{H}_{\mathcal{P}}$ as input, outputs a new ciphertext header $\widetilde{\mathcal{H}_{\mathcal{P}}}$ if $\mathcal{L}'$ satisfies the range constraints of $\mathcal{P}$.

- DecryptUser($SK_{\mathcal{L}}, \widetilde{\mathcal{H}_{\mathcal{P}}}, C') \rightarrow M$. Takes the users private key $SK_{\mathcal{L}}$, Ciphertext header $\widetilde{\mathcal{H}}_{\mathcal{P}}$, and the ciphertext C' as input, outputs the message M.

4.4.2 Construction

We provide detailed implementations of the algorithms mentioned in Section 5.3.1 and present our novel construction to enforce user revocation.

- *Setup*$(1^{\kappa}) \rightarrow (MK, PK)$
  This is run by the system manager. Let κ be the security parameter. The setup algorithm takes κ (which depends on the application) and outputs the master secret key MK and public key PK. We use bilinear map $\mathbb{S} = (\mathbb{G}, \mathbb{G}_T, e(\cdot,\cdot))$ of composite order $n = s'n'$ with two subgroups $\mathbb{G}_{s'}$ and $\mathbb{G}_{n'}$ of $\mathbb{G}$. Random generators $w \in \mathbb{G}$, $g \in \mathbb{G}_{s'}$, and $\varphi, \bar{\varphi} \in \mathbb{G}_{n'}$ are chosen. Two random numbers $\lambda, \mu \in \mathbb{Z}_n^*$ are selected, to implement forward and backward derivation functions. By Proposition 1, we have $e(g, \varphi) = e(g, \bar{\varphi}) = 1$, but $e(g, w) \neq 1$. A hash function $H : \{0,1\}^* \rightarrow \mathbb{G}$ is used to map an attribute to a binary string which represents a random group element. A randomly generated polynomial P of degree c (the maximum number of revoked users) over $\mathbb{Z}_n$ will be employed to enforce revocation. Next, this algorithm chooses two random exponents $\alpha, \beta \in \mathbb{Z}_n^*$ and outputs the public key

$$PK = (\mathbb{S}, n, g, h, \zeta, \eta, w, \varphi, \bar{\varphi}, \lambda, \mu, H(\cdot)),$$

  where,

$$h = w^{\beta}, \zeta = e(g, w)^{\alpha}, \eta = g^{\frac{1}{\beta}},$$

  and the master key

$$MK = (g^{\alpha}, \beta, n', P).$$

- *KeyGen*$(MK, u_k, \mathcal{L}) \rightarrow SK_{\mathcal{L}}$

  Given a user with ID u_k and access structure $\mathcal{L}$ on a set of attributes $S = \{A_t\} \subseteq \mathcal{A}$, this algorithm chooses a unique τ_k for each user u_k. Assume that the user u_k is assigned a temporal attribute $A_t \in \mathcal{L}$ with the constraint $A_t[t_a, t_b]$

[1]We assume that each user ID $u_k \in \mathbb{Z}_l$ where $l = min(s, p', q')$, so that, for every pair $(i, j), i \neq j, u_i - u_j \in \mathbb{Z}_n^*$. As a result, its inverse $(u_i - u_j)^{-1}$, which is needed in the *ProxyRekey* phase will exist and the secret can be successfully recovered.

and non-temporal attribute $B_t \in \mathcal{L}$. This algorithm chooses a random $r \in \mathbb{Z}$ and sets the user's attribute key as

$$(D_t, D_{t_a}^{'}, \bar{D}_{t_b}^{'}, D_t^{''}, R_{t_a}^{'}, \bar{R}_{t_b}^{'}, R_t^{''})_{A_t[t_a,t_b]\in\mathcal{L}} \text{and}$$

$$(DB_t, DB_t^{'}, RB_t^{'}, RB_t^{''})_{B_t\in\mathcal{L}}$$

where,

$$\begin{aligned}
&D_t = g^{\tau_k} H(A_t)^{r\widetilde{P}(0)},\\
&D_{t_a}^{'} = (v_{t_a})^r,\\
&\bar{D}_{t_b}^{'} = (\bar{v}_{t_b})^r,\\
&D_t^{''} = w^r,\\
&R_{t_a}^{'} = (D_{t_a}^{'})^{P(u_k)} = (v_{t_a})^{rP(u_k)},\\
&\bar{R}_{t_b}^{'} = (\bar{D}_{t_b}^{'})^{P(u_k)} = (\bar{v}_{t_b})^{rP(u_k)},\\
&R_t^{''} = (D_t^{''})^{P(u_k)} = w^{rP(u_k)},\\
&DB_t = g^{\tau_k} H(B_t)^{r\widetilde{P}(0)},\\
&RB_t^{''} = (D_t^{''})^{P(u_k)} = w^{rP(u_k)},
\end{aligned}$$

and,

$$\begin{aligned}
&v_{t_a} = \varphi^{\lambda^{t_a}}, \bar{v}_{t_b} = \bar{\varphi}^{\mu^{Z-t_b}} \in \mathbb{G}_{n'},\\
&\widetilde{P}(0) = P(0) + 1.
\end{aligned}$$

Finally, this algorithm outputs the user's private key

$$\begin{aligned}
SK_{\mathcal{L}} = (D = g^{\frac{(\alpha+\tau_k)}{\beta}},\ &\{(D_t, D_{t_a}^{'}, \bar{D}_{t_b}^{'}, D_t^{''}, R_{t_a}^{'}, \bar{R}_{t_b}^{'}, R_t^{''})\}_{A_t[t_a,t_b]\in\mathcal{L}},\\
&\{(DB_t, RB_t^{''})\}_{B_t\in\mathcal{L}}).
\end{aligned}$$

- *Encrypt*$(PK, \mathcal{P}, M) \rightarrow (\mathcal{H}_{\mathcal{P}}, C')$
  Let $\mathcal{T}$ be the access tree for the access policy $\mathcal{P}$. Data owner will do the encryption of data.
  Let s be a secret in $\mathbb{Z}_N$ for the access tree $\mathcal{T}$, and x, y are chosen such that $x + y = q_{A_t}(0)$, which is the secret share at leaf node corresponding to attribute A_t. Please refer to Section 4, [12] for a detailed meaning of $q_{A_t}(0)$. Given an access tree $\mathcal{T}$, the ciphertext is composed of a ciphertext header,

$$\mathcal{H}_{\mathcal{P}} = (\mathcal{T}, C = h^s, \{(\bar{E}_{t_i}, E_{t_i}^{'}, E_{t_j}, E_{t_j}^{'})\}_{A_t[t_i,t_j]\in\mathcal{P}}, \{(EB_t^{'})\}_{B_t\in\mathcal{P}}).$$

(4.4)

and a ciphertext $C' = Me(g,w)^{s\alpha}$, where,

$(\bar{E}_{t_i}, E'_{t_i}, E_{t_j}, E'_{t_j}) = ((\bar{v}_{t_i}w)^x, H(A_t)^x, (v_{t_j}w)^y, H(A_t)^y)$
and $(EB'_t, , D\bar{B}') = (H(B_t)^{q_{B_t}(0)}, w^{q_{B_t}(0)})$.
In case the message size($|M|$) is huge we will apply hybrid encryption where the data owner choose a random symmetric key $syk \in \mathbb{Z}$ and compute $C' = syk * e(g,w)^{s\alpha}$ and the ciphertext of the message will be $\bar{C} = M^{syk}$ and $\bar{C}$ will be added to $\mathcal{H}_{\mathcal{P}}$.

- *ProxyRekey*$(PK, MK, RL) \rightarrow PXK$
The data owner chooses a polynomial P of degree c, with coefficients in Z_n^*. Let RL be the revocation list and $u_i, i \in \{1, 2, \cdots, c\}$, be the identities of the revoked users. The owner evaluates the polynomial $P(u_i)$ at these points, using the master secret key MK. If there are less than c revoked users, then the owner generates random points x and evaluates $P(x)$ such that x does not correspond to any user's identity. This ensures that the proxy key PXK is of fixed length.

$$PXK = \forall u_i \in RL :< u_i, P(u_i) >$$

- *Convert*$(PXK, H_{\mathcal{P}}, u_k) \rightarrow$
$(\lambda_k, \{(R''_{t_i}, R''_{t_j})\}_{A_t[t_i,t_j]\in\mathcal{P}}, \{RB''_t\}_{B_t\in\mathcal{P}})$
RL will be stored in a hash table of proxy server. Every time revocation happen the data owner inform proxy and it will update the RL accordingly. Given the proxy key PXK, ciphertext header $H_{\mathcal{P}}$, and the user ID u_k, the proxy calculates

$$\lambda_i = \frac{u_k}{u_k - u_i} \prod_{j \neq i} \frac{u_j}{u_j - u_i}, \forall i, j \in \{1, \cdots, c\},$$
$$k \notin \{1, \cdots, c\}$$

For every attribute $A_t[t_i, t_j] \in \mathcal{P}$, it calculates

$$R''_{t_i} = (E'_{t_i})^{\sum_{i=1}^{c} \lambda_i P(u_i)} = H(A_t)^{x \sum_{i=1}^{c} \lambda_i P(u_i)},$$
$$R''_{t_j} = (E'_{t_j})^{\sum_{i=1}^{c} \lambda_i P(u_i)} = H(A_t)^{y \sum_{i=1}^{c} \lambda_i P(u_i)},$$
$$RB'''_t = (EB'_t)^{\sum_{i=1}^{c} \lambda_i P(u_i)} = H(B_t)^{q_{B_t}(0) \sum_{i=1}^{c} \lambda_i P(u_i)}.$$

Since the user's private key $SK_{\mathcal{L}}$ is blinded by $\widetilde{P}(0)$, it additionally needs R''_{t_i}, R''_{t_j} for decryption. The proxy also computes λ_k and gives it to the user with ID u_k.

- $Delegate(SK_{\mathcal{L}}, \mathcal{L}') \rightarrow \widetilde{SK}_{\mathcal{L}'}$

  Given the private key $SK_{\mathcal{L}}$ and a specified $\mathcal{L}'$, this algorithm checks whether, for every $A_t[t_a, t_b] \in \mathcal{L}$ and $A_t[t_i, t_j] \in \mathcal{L}'$, $t_a \leq t_j$ and $t_b \geq t_i$ is true for each attribute $A_t \in \mathcal{L}'$.
  If true, the user computes,

$$\begin{aligned}
D'_{t_j} &\leftarrow f_{t_a \leq t_j}(D'_{t_a}) \cdot D''_t = (v_{t_j} w)^r \\
\bar{D}'_{t_i} &\leftarrow \bar{f}_{t_b \geq t_i}(\bar{D}'_{t_b}) \cdot D''_t = (\bar{v}_{t_i} w)^r \\
R'_{t_j} &\leftarrow f_{t_a \leq t_j}(R'_{t_a}) = (v_{t_j})^{rP(u_k)} \\
\bar{R}'_{t_i} &\leftarrow \bar{f}_{t_b \geq t_i}(\bar{R}'_{t_b}) = (\bar{v}_{t_i})^{rP(u_k)}
\end{aligned}$$

  Next, this algorithm chooses a random $\delta \in \mathbb{Z}$ and computes
  $\widetilde{SK}_{\mathcal{L}'} = \{\widetilde{D}_t, \widetilde{D}'_{t_j}, \widetilde{\bar{D}'}_{t_i}, R_{t_j}, \bar{R}_{t_i}\}_{A_t \in \mathcal{L}'}, \{\widetilde{DB}_t, \widetilde{DB}'_t, RB_t,\}_{B_t \in \mathcal{L}'}$, where,

$$\begin{aligned}
\widetilde{D}_t &= D_t \cdot (gH(A_t))^{\delta} = g^{\tau_k + \delta} H(A_t)^{r\widetilde{P}(0) + \delta} \\
\widetilde{D}'_{t_j} &= D'_{t_j} \cdot (v_{t_j} w)^{\delta} = (v_{t_j} w)^{r+\delta} \\
\widetilde{\bar{D}'}_{t_i} &= \bar{D}'_{t_i} \cdot (\bar{v}_{t_i} w)^{\delta} = (\bar{v}_{t_i} w)^{r+\delta} \\
\widetilde{DB}_t &= DB_t \cdot (gH(A_t))^{\delta} = g^{\tau_k + \delta} H(B_t)^{r\widetilde{P}(0) + \delta} \\
\widetilde{DB}'_t &= D'_{t_j} \cdot w^{\delta} = w^{r+\delta}
\end{aligned}$$

$$\begin{aligned}
R_{t_j} &= e(R''_t \cdot R'_{t_j}, E'_{t_j})^{\lambda_k} \cdot e(D'_{t_j}, R''_{t_j}) \\
&= e(v_{t_j} w, H(A_t))^{yr(\lambda_k P(u_k) + \sum_{i=1}^{t} \lambda_i P(u_i))} \\
&= e(v_{t_j} w, H(A_t))^{yrP(0)} \\
\bar{R}_{t_i} &= e(R''_t \cdot \bar{R}'_{t_i}, E'_{t_i})^{\lambda_k} \cdot e(\bar{D}'_{t_i}, \bar{R}''_{t_i}) \\
&= e(\bar{v}_{t_i} w, H(A_t))^{xr(\lambda_k P(u_k) + \sum_{i=1}^{t} \lambda_i P(u_i))} \\
&= e(\bar{v}_{t_i} w, H(A_t))^{xrP(0)} \\
RB_t &= e(RB''_t, EB'_t)^{\lambda_k} \cdot e(DB'_t, RB'''_t) \\
&= e(w, H(B_t))^{q_{B_t}(0) rP(0)}
\end{aligned}$$

  Finally, it outputs $\widetilde{SK}_{\mathcal{L}'}$ as the derivation key for $\mathcal{L}'$.

- $DecryptProxy(\widetilde{SK}_{\mathcal{L}'}, \mathcal{H}_{\mathcal{P}}, u_k) \rightarrow \widetilde{\mathcal{H}}_{\mathcal{P}}$

On receiving the private key $\widetilde{SK}_{\mathcal{L}'}$, and the ciphertext header $\mathcal{H}_{\mathcal{P}}$, this algorithm checks whether the user u_k is present in the hash table RL. If not then it will check each range attribute $A_t[t_i, t_j] \in \mathcal{L}'$ is consistent with $A_t[t_i, t_j] \in \mathcal{P}$. If true, the secret share $q_{A_t}(0)$ over $\mathbb{G}_T$ is reconstructed as follows:-

$$
\begin{aligned}
F_1 \longleftarrow\ & \frac{e(\widetilde{D}_t, E_{t_j})}{R_{t_j} \cdot e(\widetilde{D}'_{t_j}, E'_{t_j})} \\
=\ & \frac{e(g^{\tau_k+\delta} H(A_t)^{r\widetilde{P}(0)+\delta}, (v_{t_j} w)^y)}{R_{t_j} \cdot e((v_{t_j} w)^{r+\delta}, H(A_t)^y)} \\
=\ & \frac{e(g^{\tau_k+\delta}, (v_{t_j} w)^y) \cdot e(H(A_t)^{r\widetilde{P}(0)+\delta}, (v_{t_j} w)^y)}{R_{t_j} \cdot e((v_{t_j} w)^{r+\delta}, H(A_t)^y)} \\
=\ & \frac{e(g^{\tau_k+\delta}, (v_{t_j} w)^y) \cdot e(v_{t_j} w, H(A_t))^{yrP(0)}}{R_{t_j}} \\
=\ & e(g^{\tau_k+\delta}, v_{t_j}^y) \cdot e(g^{\tau_k+\delta}, w^y) \\
=\ & e(g^{\tau_k+\delta}, w)^y
\end{aligned}
$$

Similarly,

$$
\begin{aligned}
F_2 \longleftarrow\ & \frac{e(\widetilde{D}_t, \bar{E}_{t_i})}{\bar{R}_{t_i} \cdot e(\widetilde{\widetilde{D}'}_{t_i}, E'_{t_i})} \\
=\ & \frac{e(g^{\tau_k+\delta} H(A_t)^{rP'(0)+\delta}, (\bar{v}_{t_i} w)^x)}{\bar{R}_{t_i} \cdot e((\bar{v}_{t_i} w)^{r+\delta}, H(A_t)^x)} \\
=\ & \frac{e(g^{\tau_k+\delta}, (\bar{v}_{t_i} w)^x) \cdot e(H(A_t)^{rP'(0)+\delta}, (\bar{v}_{t_i} w)^x)}{\bar{R}_{t_i} \cdot e((\bar{v}_{t_i} w)^{r+\delta}, H(A_t)^x)} \\
=\ & \frac{e(g^{\tau_k+\delta}, (\bar{v}_{t_i} w)^x) \cdot e(\bar{v}_{t_i} w, H(A_t))^{xrP(0)}}{\bar{R}_{t_i}} \\
=\ & e(g^{\tau_k+\delta}, \bar{v}_{t_i}^x) \cdot e(g^{\tau_k+\delta}, w^x) \\
=\ & e(g^{\tau_k+\delta}, w)^x
\end{aligned}
$$

For $B_t \in \mathcal{L}'$ is consistent with $B_t \in \mathcal{P}$, then the secret share $q_{B_t}(0)$ of s over $\mathbb{G}_T$ is reconstructed as,

$$
\begin{aligned}
F_3 \quad \longleftarrow \quad & \frac{e(\widetilde{D}_t, \bar{E}B_t)}{\bar{R}B_t \cdot e(\widetilde{\bar{D}B'}_t, EB'_t)} \\
= \quad & \frac{e(g^{\tau_k+\delta} H(B_t)^{rP'(0)+\delta}, (w^{q_{B_t}(0)})}{RB_t \cdot e(w^{r+\delta}, H(B_t)^{q_{B_t}(0)})} \\
= \quad & \frac{e(g^{\tau_k+\delta}, w^x) \cdot e(H(B_t)^{r\widetilde{P}(0)+\delta}, w^{q_{B_t}(0)})}{RB_t \cdot e(w^{r+\delta}, H(B_t)^x)} \\
= \quad & \frac{e(g^{\tau_k+\delta}, w^x) \cdot e(w, H(B_t))^{q_{B_t}(0) rP(0)}}{\bar{R}B_t} \\
= \quad & e(g^{\tau_k+\delta}, w^{q_{B_t}(0)}) \\
= \quad & e(g^{\tau_k+\delta}, w)^{q_{B_t}(0)}
\end{aligned}
$$

$F_t = F_1 \cdot F_2 = e(g^{\tau_k+\delta}, w)^{q_{A_t}(0)}$

where, $e(g^{\tau_k+\delta}, v_{t_j}^y) = e(g^{\tau_k+\delta}, \bar{v}_{t_i}^x) = 1$ because $g^{\tau_k+\delta} \in \mathbb{G}_{s'}$ and $v_{t_j}^y, \bar{v}_{t_i}^x \in \mathbb{G}_{n'}$. Next, the value of $T = e(g^{\tau_k+\delta}, w)^s$ is computed from $\{e(g^{\tau_k+\delta}, w)^{q_{A_i}(0)}\}_{A_i \in \mathcal{P}}$ and $\{e(g^{\tau_k+\delta}, w)^{q_{B_i}(0)}\}_{B_i \in \mathcal{P}}$ by using the recursive DecryptNode algorithm described in Bethencourt et al. [12]. Finally, the new ciphertext header $\widetilde{\mathcal{H}}_{\mathcal{P}} = (C, T)$ is returned.

- *DecryptUser*$(SK_{\mathcal{L}}, \widetilde{\mathcal{H}}_{\mathcal{P}}, C') \rightarrow M$

  On receiving the new ciphertext header $\widetilde{\mathcal{H}}_{\mathcal{P}} = (C, T) = (w^{\beta s}, e(g^{\tau_k+\delta}, w)^s)$, the secret δ is used to compute $D' = D \cdot \eta^\delta = g^{\frac{(\alpha+\tau_k)}{\beta}} g^{\frac{\delta}{\beta}} = g^{\frac{(\alpha+\tau_k+\delta)}{\beta}}$.

  Let $ek = \dfrac{e(C, D')}{T} = \dfrac{e(g^{\frac{(\alpha+\tau_k+\delta)}{\beta}}, w^{\beta s})}{e(g^{\tau_k+\delta}, w)^s} = e(g^\alpha, w)^s$.
  Finally, the plaintext message is computed by
  $M = C'/ek$.

  In case of hybrid encryption the user will first get the symmetric key $syk = C'/ek$ and then it will calculate messgage $M = \bar{C}^{syk^{-1}}$

Chapter 5

Our implementations

5.1 Introduction

In this chapter we have presented two novel schemes which are the improvements of last scheme described above. In the scheme-4 we have added two phase encryption for encryption cost improvement and in the scheme-5 we have added the feature of distributed key generation above scheme-4.

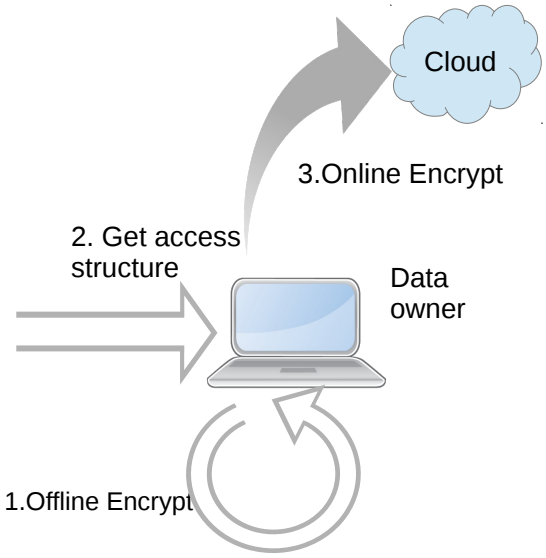
5.2 Scheme-4:Temporal Access Control with added two phase encryption

Here we propose a modified version of temporal access control scheme to protect and selectively access data in clouds with two phases of encryption called online-offline encryption which is introduced here for the first time in CPABE with shamir secret sharing scheme. In this new variant of attribute-based encryption to reduce computational load during encryption we did some preprocessing of ciphertext before the access structure and the message is known to us.

5.2.1 Framework

A brief description of each of these algorithms is as follows:

- Setup$(1^\kappa) \rightarrow (MK, PK)$: Takes a security parameter κ as input, outputs the master secret key MK and the public key PK.
- KeyGen$(MK, u_k, \mathcal{L}) \rightarrow SK_{\mathcal{L}}$: Takes the user ID u_k, the access privilege $\mathcal{L}$, and MK as input, outputs the users private key $SK_{\mathcal{L}}$.
- OfflineEncrypt$(PK, \mathcal{L}) \rightarrow C_\rho$: for each attribute (if any) i.e. $\mathcal{L} = \{A_k(t_a, t_b)\}_{A_k \in \mathcal{A}_T} \bigcup \{B_k\}_{B_k \in \mathcal{A}_N}$ data owner will generate intermediate ciphertext C_a and C'_a where $a \in \mathcal{L}$



- OnlineEncrypt$(PK, M, \mathcal{P}) \rightarrow (\mathcal{H}_{\mathcal{P}}, C')$: Takes a comparable access policy $\mathcal{P}$, public key PK and message M as input, outputs the ciphertext header $\mathcal{H}_{\mathcal{P}}$ and final ciphertext C'.
- ProxyRekey$(PK, MK, RL) \rightarrow PXK$: Takes the public key PK, master key MK, and the revocation list RL as inputs, outputs the proxy key PXK, which contains the information necessary to enable the proxy server to partially decrypt the ciphertext.
- Convert$(PXK, \mathcal{H}_{\mathcal{P}}, u_k) \rightarrow (\lambda_k, \{(R''_{t_i}, R''_{t_j})\}_{A_t[t_i,t_j]\in\mathcal{P}}, RB''_{t\ B_t\in\mathcal{P}})$: Takes the proxy key PXK, ciphertext header $\mathcal{H}_{\mathcal{P}}$, and user ID u_k as input, outputs components needed to unblind the user's private key $SK_{\mathcal{L}}$.
- Delegate$(SK_{\mathcal{L}}, \mathcal{L}') \rightarrow \widetilde{SK}_{\mathcal{L}'}$: Takes a private key $SK_{\mathcal{L}}$ and a specified privilege requirement $\mathcal{L}'$ as input, outputs a derivation key $\widetilde{SK}_{\mathcal{L}'}$, if each attribute in $\mathcal{L}$ and $\mathcal{L}'$ match.
- DecryptProxy$(\widetilde{SK}_{\mathcal{L}'}, \mathcal{H}_{\mathcal{P}}) \rightarrow \widetilde{\mathcal{H}_{\mathcal{P}}}$: Takes a users private key $\widetilde{SK}_{\mathcal{L}'}$ and the ciphertext header $\mathcal{H}_{\mathcal{P}}$ as input, outputs a new ciphertext header $\widetilde{\mathcal{H}_{\mathcal{P}}}$ if $\mathcal{L}'$ satisfies the range constraints of $\mathcal{P}$.
- DecryptUser$(SK_{\mathcal{L}}, \widetilde{\mathcal{H}_{\mathcal{P}}}, C') \rightarrow M$. Takes the users private key $SK_{\mathcal{L}}$, Ciphertext header $\widetilde{\mathcal{H}}_{\mathcal{P}}$, and the ciphertext C' as input, outputs the message M.

5.2.2 Construction

We provide detailed implementations of the algorithms mentioned in Section 5.3.1 and present our novel construction to enforce user revocation.

- $Setup(1^\kappa) \rightarrow (MK, PK)$
  This is run by the system manager. Let κ be the security parameter. The setup algorithm takes κ (which depends on the application) and outputs the master secret key MK and public key PK. We use bilinear map $\mathbb{S} = (\mathbb{G}, \mathbb{G}_T, e(\cdot,\cdot))$ of composite order $n = s'n'$ with two subgroups $\mathbb{G}_{s'}$ and $\mathbb{G}_{n'}$ of $\mathbb{G}$. Random generators $w \in \mathbb{G}$, $g \in \mathbb{G}_{s'}$, and $\varphi, \bar{\varphi} \in \mathbb{G}_{n'}$ are chosen. Two random numbers $\lambda, \mu \in \mathbb{Z}_n^*$ are selected, to implement forward and backward derivation functions. By Proposition 1, we have $e(g, \varphi) = e(g, \bar{\varphi}) = 1$, but $e(g, w) \neq 1$. A hash function $H : \{0,1\}^* \rightarrow \mathbb{G}$ is used to map an attribute to a binary string which represents a random group element. A randomly generated polynomial P of degree c (the maximum number of revoked users) over $\mathbb{Z}_n$ will be employed to enforce revocation. Next, this algorithm chooses two random exponents $\alpha, \beta \in \mathbb{Z}_n^*$ and outputs the public key

  $$PK = (\mathbb{S}, n, g, h, \zeta, \eta, w, \varphi, \bar{\varphi}, \lambda, \mu, H(\cdot)),$$

  where,

  $$h = w^\beta, \zeta = e(g, w)^\alpha, \eta = g^{\frac{1}{\beta}},$$

  and the master key

  $$MK = (g^\alpha, \beta, n', P).$$

- $KeyGen(MK, u_k, \mathcal{L}) \rightarrow SK_{\mathcal{L}}$

  Given a user with ID u_k[1] and access structure $\mathcal{L}$ on a set of attributes $S = \{A_t\} \subseteq \mathcal{A}$, this algorithm chooses a unique τ_k for each user u_k. Assume that the user u_k is assigned a temporal attribute $A_t \in \mathcal{L}$ with the constraint $A_t[t_a, t_b]$ and non-temporal attribute $B_t \in \mathcal{L}$. This algorithm chooses a random $r \in \mathbb{Z}$ and sets the user's attribute key as

  $$(D_t, D'_{t_a}, \bar{D}'_{t_b}, D''_t, R'_{t_a}, \bar{R}'_{t_b}, R''_t)_{A_t[t_a,t_b] \in \mathcal{L}} \text{and}$$

  $$(DB_t, DB'_t, RB'_t, RB''_t)_{B_t \in \mathcal{L}}$$

[1]We assume that each user ID $u_k \in \mathbb{Z}_l$ where $l = min(s, p', q')$, so that, for every pair $(i, j), i \neq j, u_i - u_j \in \mathbb{Z}_n^*$. As a result, its inverse $(u_i - u_j)^{-1}$, which is needed in the *ProxyRekey* phase will exist and the secret can be successfully recovered.

where,

$$
\begin{aligned}
&D_t = g^{\tau_k} H(A_t)^{r\widetilde{P}(0)},\\
&D_{t_a}^{'} = (v_{t_a})^r,\\
&\bar{D}_{t_b}^{'} = (\bar{v}_{t_b})^r,\\
&D_t^{''} = w^r,\\
&R_{t_a}^{'} = (D_{t_a}^{'})^{P(u_k)} = (v_{t_a})^{rP(u_k)},\\
&\bar{R}_{t_b}^{'} = (\bar{D}_{t_b}^{'})^{P(u_k)} = (\bar{v}_{t_b})^{rP(u_k)},\\
&R_t^{''} = (D_t^{''})^{P(u_k)} = w^{rP(u_k)},\\
&DB_t = g^{\tau_k} H(B_t)^{r\widetilde{P}(0)},\\
&RB_t^{''} = (D_t^{''})^{P(u_k)} = w^{rP(u_k)},
\end{aligned}
$$

and,

$$
\begin{aligned}
&v_{t_a} = \varphi^{\lambda^{t_a}}, \bar{v}_{t_b} = \bar{\varphi}^{\mu^{Z-t_b}} \in \mathbb{G}_{n'},\\
&\widetilde{P}(0) = P(0) + 1.
\end{aligned}
$$

Finally, this algorithm outputs the user's private key

$$
\begin{aligned}
SK_{\mathcal{L}} = (D = g^{\frac{(\alpha+\tau_k)}{\beta}},\\
&\{(D_t, D_{t_a}^{'}, \bar{D}_{t_b}^{'}, D_t^{''}, R_{t_a}^{'}, \bar{R}_{t_b}^{'}, R_t^{''})\}_{A_t[t_a,t_b]\in\mathcal{L}},\\
&\{(DB_t, RB_t^{''})\}_{B_t\in\mathcal{L}}).
\end{aligned}
$$

- *OfflineEncrypt*$(PK) \rightarrow (C_\rho)$

  For each temporal attribute $A_t \in \mathcal{L}$ generate two random number $\rho_{A_{t_x}} \in \mathbb{Z}$ and $\rho_{A_{t_y}} \in \mathbb{Z}$ then calculate:-
  $\bar{E}_{t_i} = (\bar{v}_{t_i} w)^{\rho_{A_{t_x}}}$
  $E_{t_i}^{'} = H(A_t)^{\rho_{A_{t_x}}}$
  $E_{t_j} = (v_{t_j} w)^{\rho_{A_{t_y}}}$
  $E_{t_j}^{'} = H(A_t)^{\rho_{A_{t_y}}}$
  And for each non temporal attribute $B_t \in \mathcal{L}$ choose random number $\rho_{B_t} \in \mathbb{Z}$
  $EB_t' = w^{\rho_{B_t}}$
  $D\bar{B}' = H(B_t)^{\rho_{B_t}}$
  So the initial ciphertext generated in Offline will be :-

$$
C_\rho = ((\bar{E}_{t_i}, E_{t_i}^{'}, E_{t_j}, E_{t_j}^{'})_{A_t[t_i,t_j]\in\mathcal{P}}, \{(EB_t', D\bar{B}')\}_{B_t\in\mathcal{P}}). \tag{5.1}
$$

- *OnlineEncrypt*$(PK, \mathcal{P}, M) \rightarrow (\mathcal{H}_\mathcal{P}, C')$

  Data owner will do the encryption of data.Let $\mathcal{T}$ be the access tree for the access policy $\mathcal{P}$. The algorithm first chooses a polynomial q_x for each node x

(including the leaves) in the tree T . These polynomials are chosen in the following way in a topdown manner, starting from the root node R. For each node x in the tree, set the degree d_x of the polynomial q_x to be one less than the threshold value k_x of that node, that is, $d_x = k_x - 1$. Starting with the root node R the algorithm chooses a random $s \in Z_p$ and sets $q_R(0) = s$. Then, it chooses d_R other points of the polynomial q_R randomly to define it completely. For any other node x, it sets $q_x(0) = q_{parent(x)}(index(x))$ and chooses d_x other points randomly to completely define q_x.Please refer to Section 4, [12] for a detailed meaning of access structure and threshold.

For every temporal attribute calculate ρ_{A_t} such that $\rho_{A_t} = q_{A_t}(0) - (\rho_{A_{t_x}} + \rho_{A_{t_y}})$, where $q_{A_t}(0)$ is the secret share at leaf node corresponding to attribute A_t .
And for every non-temporal attribute B_t choose ρ'_B such that $\rho_{B_t} + \rho'_B = q_{B_t}(0)$
Given an access tree $\mathcal{T}$, the ciphertext is composed of a ciphertext header,

$$\mathcal{H}_{\mathcal{P}} = (\mathcal{T}, C = h^s, C_\rho, \{\rho_{A_t}\}_{A_t[t_i,t_j]\in\mathcal{P}} \{\rho'_B\}_{B_t\in\mathcal{P}}). \tag{5.2}$$

and a ciphertext $C' = Me(g,w)^{s\alpha}$,

In case the message size is huge we will apply hybrid encryption where the data owner choose a random symmetric key $syk \in \mathbb{Z}$ and compute $C' = syk * e(g,w)^{s\alpha}$ and the ciphertext of the message will be $\bar{C} = Enc_{syk}(M)$ where Enc is any symmetric key encryption according to user's choice.
Then $\bar{C}$ will be added to $\mathcal{H}_{\mathcal{P}}$.

- *ProxyRekey*$(PK, MK, RL) \rightarrow PXK$
  The data owner chooses a polynomial P of degree c, with coefficients in Z_n^*. Let RL be the revocation list and $u_i, i \in \{1, 2, \cdots, c\}$, be the identities of the revoked users. The owner evaluates the polynomial $P(u_i)$ at these points, using the master secret key MK. If there are less than c revoked users, then the owner generates random points x and evaluates $P(x)$ such that x does not correspond to any user's identity. This ensures that the proxy key PXK is of fixed length.

$$PXK = \forall u_i \in RL :< u_i, P(u_i) >$$

- *Convert*$(PXK, H_{\mathcal{P}}, u_k) \rightarrow$
  $(\lambda_k, \{(R''_{t_i}, R''_{t_j})\}_{A_t[t_i,t_j]\in\mathcal{P}}, \{RB''_t\}_{B_t\in\mathcal{P}})$
  RL will be stored in a hash table of proxy server. Every time revocation happen the data owner inform proxy and it will update the RL accordingly. Given the proxy key PXK, ciphertext header $H_{\mathcal{P}}$, and the user ID u_k, the proxy calculates

$$\lambda_i = \frac{u_k}{u_k - u_i} \prod_{j\neq i} \frac{u_j}{u_j - u_i}, \forall i, j \in \{1, \cdots, c\},$$

$$k \notin \{1, \cdots, c\}$$

For every attribute $A_t[t_i, t_j] \in \mathcal{P}$, it calculates

$$R''_{t_i} = (E'_{t_i})^{\sum_{i=1}^{c} \lambda_i P(u_i)} = H(A_t)^{\rho_{A_{t_x}} \sum_{i=1}^{c} \lambda_i P(u_i)},$$
$$R''_{t_j} = (E'_{t_j})^{\sum_{i=1}^{c} \lambda_i P(u_i)} = H(A_t)^{\rho_{A_{t_y}} \sum_{i=1}^{c} \lambda_i P(u_i)},$$
$$RB'''_t = (EB'_t)^{\sum_{i=1}^{c} \lambda_i P(u_i)} = H(B_t)^{\rho_{B_t} \sum_{i=1}^{c} \lambda_i P(u_i)}.$$

Since the user's private key $SK_{\mathcal{L}}$ is blinded by $\widetilde{P}(0)$, it additionally needs R''_{t_i}, R''_{t_j} for decryption. The proxy also computes λ_k and gives it to the user with ID u_k.

- *Delegate*$(SK_{\mathcal{L}}, \mathcal{L}') \rightarrow \widetilde{SK}_{\mathcal{L}'}$

Given the private key $SK_{\mathcal{L}}$ and a specified $\mathcal{L}'$, this algorithm checks whether, for every $A_t[t_a, t_b] \in \mathcal{L}$ and $A_t[t_i, t_j] \in \mathcal{L}'$, $t_a \leq t_j$ and $t_b \geq t_i$ is true for each attribute $A_t \in \mathcal{L}'$.
If true, the user computes,

$$D'_{t_j} \leftarrow f_{t_a \leq t_j}(D'_{t_a}) \cdot D''_t = (v_{t_j} w)^r$$
$$\bar{D}'_{t_i} \leftarrow \bar{f}_{t_b \geq t_i}(\bar{D}'_{t_b}) \cdot D''_t = (\bar{v}_{t_i} w)^r$$
$$R'_{t_j} \leftarrow f_{t_a \leq t_j}(R'_{t_a}) = (v_{t_j})^{rP(u_k)}$$
$$\bar{R}'_{t_i} \leftarrow \bar{f}_{t_b \geq t_i}(\bar{R}'_{t_b}) = (\bar{v}_{t_i})^{rP(u_k)}$$

Next, this algorithm chooses a random $\delta \in \mathbb{Z}$ and computes
$\widetilde{SK}_{\mathcal{L}'} = D_{del}, \{\widetilde{D}_t, \widetilde{D}'_{t_j}, \widetilde{\bar{D}'}_{t_i}, R_{t_j}, \bar{R}_{t_i}\}_{A_t \in \mathcal{L}'}, \{\widetilde{DB}_t, \widetilde{DB}'_t, RB_t,\}_{B_t \in \mathcal{L}'}$, where,

$$\widetilde{D}_t = D_t \cdot (gH(A_t))^{\delta} = g^{\tau_k + \delta} H(A_t)^{r\widetilde{P}(0) + \delta}$$
$$\widetilde{D}'_{t_j} = D'_{t_j} \cdot (v_{t_j} w)^{\delta} = (v_{t_j} w)^{r+\delta}$$
$$\widetilde{\bar{D}'}_{t_i} = \bar{D}'_{t_i} \cdot (\bar{v}_{t_i} w)^{\delta} = (\bar{v}_{t_i} w)^{r+\delta}$$
$$\widetilde{DB}_t = DB_t \cdot (gH(A_t))^{\delta} = g^{\tau_k + \delta} H(B_t)^{r\widetilde{P}(0) + \delta}$$
$$\widetilde{DB}'_t = D'_{t_j} \cdot w^{\delta} = w^{r+\delta}$$
$$D_{del} = g^{\delta}$$

$$
\begin{aligned}
R_{t_j} &= e(R_t^{''} \cdot R_{t_j}^{'}, E_{t_j}^{'})^{\lambda_k} \cdot e(D_{t_j}^{'}, R_{t_j}^{''}) \\
&= e(v_{t_j} w, H(A_t))^{\rho_{A_{t_y}} r(\lambda_k P(u_k) + \sum_{i=1}^{t} \lambda_i P(u_i))} \\
&= e(v_{t_j} w, H(A_t))^{\rho_{A_{t_y}} r P(0)} \\
\bar{R}_{t_i} &= e(R_t^{''} \cdot \bar{R}_{t_i}^{'}, E_{t_i}^{'})^{\lambda_k} \cdot e(\bar{D}_{t_i}^{'}, \bar{R}_{t_i}^{''}) \\
&= e(\bar{v}_{t_i} w, H(A_t))^{\rho_{A_{t_x}} r(\lambda_k P(u_k) + \sum_{i=1}^{t} \lambda_i P(u_i))} \\
&= e(\bar{v}_{t_i} w, H(A_t))^{\rho_{A_{t_x}} r P(0)} \\
RB_t &= e(RB_t^{''}, EB_t^{'})^{\lambda_k} \cdot e(DB_t^{'}, RB_t^{'''}) \\
&= e(w, H(B_t))^{\rho_{B_t} P(0)}
\end{aligned}
$$

Finally, it outputs $\widetilde{SK}_{\mathcal{L}'}$ as the derivation key for $\mathcal{L}'$.

- *DecryptProxy*$(\widetilde{SK}_{\mathcal{L}'}, \mathcal{H}_{\mathcal{P}}, u_k) \rightarrow \widetilde{\mathcal{H}}_{\mathcal{P}}$

On receiving the private key $\widetilde{SK}_{\mathcal{L}'}$, and the ciphertext header $\mathcal{H}_{\mathcal{P}}$, this algorithm checks whether the user u_k is present in the hash table RL. If not then it will check each range attribute $A_t[t_i, t_j] \in \mathcal{L}'$ is consistent with $A_t[t_i, t_j] \in \mathcal{P}$. If true, the secret share $q_{A_t}(0)$ over $\mathbb{G}_T$ is reconstructed as follows:-

$$
\begin{aligned}
F_1 \longleftarrow\ & \frac{e(\widetilde{D}_t, E_{t_j})}{R_{t_j} \cdot e(\widetilde{D}_{t_j}^{'}, E_{t_j}^{'})} \\
=\ & \frac{e(g^{\tau_k + \delta} H(A_t)^{r\widetilde{P}(0) + \delta}, (v_{t_j} w)^{\rho_{A_{t_y}}})}{R_{t_j} \cdot e((v_{t_j} w)^{r+\delta}, H(A_t)^{\rho_{A_{t_y}}})} \\
=\ & \frac{e(g^{\tau_k + \delta}, (v_{t_j} w)^{\rho_{A_{t_y}}}) \cdot e(H(A_t)^{r\widetilde{P}(0) + \delta}, (v_{t_j} w)^{\rho_{A_{t_y}}})}{R_{t_j} \cdot e((v_{t_j} w)^{r+\delta}, H(A_t)^{\rho_{A_{t_y}}})} \\
=\ & \frac{e(g^{\tau_k + \delta}, (v_{t_j} w)^{\rho_{A_{t_y}}}) \cdot e(v_{t_j} w, H(A_t))^{\rho_{A_{t_y}} r P(0)}}{R_{t_j}} \\
=\ & e(g^{\tau_k + \delta}, v_{t_j}^{\rho_{A_{t_y}}}) \cdot e(g^{\tau_k + \delta}, w^{\rho_{A_{t_y}}}) \\
=\ & e(g^{\tau_k + \delta}, w)^{\rho_{A_{t_y}}}
\end{aligned}
$$

Similarly,

$$
\begin{aligned}
F_2 \longleftarrow & \frac{e(\widetilde{D}_t, \bar{E}_{t_i})}{\bar{R}_{t_i} \cdot e(\widetilde{\bar{D}'}_{t_i}, E'_{t_i})} \\
= & \frac{e(g^{\tau_k+\delta} H(A_t)^{rP'(0)+\delta}, (\bar{v}_{t_i} w)^{\rho_{A_{t_x}}})}{\bar{R}_{t_i} \cdot e((\bar{v}_{t_i} w)^{r+\delta}, H(A_t)^{\rho_{A_{t_x}}})} \\
= & \frac{e(g^{\tau_k+\delta}, (\bar{v}_{t_i} w)^{\rho_{A_{t_x}}}) \cdot e(H(A_t)^{rP'(0)+\delta}, (\bar{v}_{t_i} w)^{\rho_{A_{t_x}}})}{\bar{R}_{t_i} \cdot e((\bar{v}_{t_i} w)^{r+\delta}, H(A_t)^{\rho_{A_{t_x}}})} \\
= & \frac{e(g^{\tau_k+\delta}, (\bar{v}_{t_i} w)^{\rho_{A_{t_x}}}) \cdot e(\bar{v}_{t_i} w, H(A_t))^{\rho_{A_{t_x}} rP(0)}}{\bar{R}_{t_i}} \\
= & e(g^{\tau_k+\delta}, \bar{v}_{t_i}^{\rho_{A_{t_x}}}) \cdot e(g^{\tau_k+\delta}, w^{\rho_{A_{t_x}}}) \\
= & e(g^{\tau_k+\delta}, w)^{\rho_{A_{t_x}}}
\end{aligned}
$$

For $B_t \in \mathcal{L}'$ is consistent with $B_t \in \mathcal{P}$, then the secret share $q_{B_t}(0)$ of s over $\mathbb{G}_T$ is reconstructed as,

$$
\begin{aligned}
F_3 \longleftarrow & \frac{e(\widetilde{D}_t, \bar{EB}_t)}{\bar{RB}_t \cdot e(\widetilde{D\bar{B}'}_t, EB'_t)} \\
= & \frac{e(g^{\tau_k+\delta} H(B_t)^{rP'(0)+\delta}, w^{\rho_{B_t}})}{RB_t \cdot e(w^{r+\delta}, H(B_t)^{\rho_{B_t}})} \\
= & \frac{e(g^{\tau_k+\delta}, w^{\rho_{B_t}}) \cdot e(H(B_t)^{r\widetilde{P}(0)+\delta}, w^{\rho_{B_t}})}{RB_t \cdot e(w^{r+\delta}, H(B_t)^{\rho_{B_t}})} \\
= & \frac{e(g^{\tau_k+\delta}, w^{\rho_{B_t}}) \cdot e(w, H(B_t))^{\rho_{B_t} rP(0)}}{\bar{RB}_t} \\
= & e(g^{\tau_k+\delta}, w^{\rho_{B_t}}) \\
= & e(g^{\tau_k+\delta}, w)^{\rho_{B_t}}
\end{aligned}
$$

$F_t = F_1 \cdot F_2 = e(g^{\tau_k+\delta}, w)^{(\rho_{A_{t_x}} + \rho_{A_{t_y}})}$

where, $e(g^{\tau_k+\delta}, v_{t_j}^y) = e(g^{\tau_k+\delta}, \bar{v}_{t_i}^x) = 1$ because $g^{\tau_k+\delta} \in \mathbb{G}_{s'}$ and $v_{t_j}^y, \bar{v}_{t_i}^x \in \mathbb{G}_{n'}$. Next, the value of $T = e(g^{\tau_k+\delta}, w)^s$ is computed from $\{e(g^{\tau_k+\delta}, w)^{q_{A_i}(0)}\}_{A_i \in \mathcal{P}}$ and $\{e(g^{\tau_k+\delta}, w)^{q_{B_i}(0)}\}_{B_i \in \mathcal{P}}$ by using the recursive DecryptNode algorithm below:-

We now consider the recursive case when x is a non-leaf node. The algorithm DecryptNode(CT, SK,x) then proceeds as follows:
For all nodes z that are children and all z are leaf node of x, it calls DecryptNode(CT, SK,z) and stores the output(F_t for leaf node with temporal attribute and F_3 for leaf node with non-temporal attribute) as F_z. Let S_x be an arbitrary k_x-sized set of child nodes z such that $F_z \neq \perp$. If no such set exists then the node was not satisfied and the function returns $\perp$. Otherwise, we first compute
$(e(D,h)/\zeta) \cdot e(D_{del}, w)$
$= (e(g, w^{\beta})^{\alpha+\tau_k/\beta}/e(g,w)^{\alpha}) \cdot e(g,w)^{\delta}$
$= e(g,w)^{\tau_k} \cdot e(g,w)^{\delta}$
$= e(g,w)^{\tau_k+\delta}$

Then it will calculate recursively:-
$F_x = e(g^{\tau_k+\delta}, w)^{\sum_{z\in S_x} \rho_{A_t}} \prod_{z\in S_x} F_z^{\triangle_{i,S'_x}(0)}$, (where i=index(z) $S'_x = index(z) : z \in S_x$)
Note that in the R.H.S in the first term ρ_{A_t} is mentioned assuming it is temporal attribute. But in case of non-temporal it will be ρ'_{B_t}. For simplicity we are using ρ_{A_t} for the rest of the equations. So
$F_x =$
$e(g^{\tau_k+\delta}, w)^{\sum_{z\in S_x} \rho_{A_t}} \prod_{z\in S_x} e(g^{\tau_k+\delta}, w)^{(\rho_{A_{t_x}}+\rho_{A_{t_y}})\triangle_{i,S'_x}(0)}$
$= e(g^{\tau_k+\delta}, w)^{\sum_{z\in S_x} \rho_{A_t}} \cdot e(g^{\tau_k+\delta}, w)^{\sum_{z\in S_x} (\rho_{A_{t_x}}+\rho_{A_{t_y}})\triangle_{i,S'_x}(0)}$
$= e(g^{\tau_k+\delta}, w)^{\sum_{z\in S_x} (\rho_{A_t}+\rho_{A_{t_x}}+\rho_{A_{t_y}})\triangle_{i,S'_x}(0)}$
$= e(g^{\tau_k+\delta}, w)^{\sum_{z\in S_x} (q_{A_t}(0))\triangle_{i,S'_x}(0)}$
now $q_{A_t}(0) = q_{parent(z)(index(z))} = q_x(i)$
So the above equation will be:-

$$F_x = e(g^{\tau_k+\delta}, w)^{\sum_{z\in S_x} q_x(i)\triangle_{i,S'_x}(0)} = e(g^{\tau_k+\delta}, w)^{q_x(0)} \tag{5.3}$$

(using polynomial interpolation).
Now suppose for all non-leaf node x whose children are also non-leaf node suppose z the value of F_x will be :-
$\prod_{z\in S_x} F_z^{\triangle_{i,S'_x}(0)}$
$=\prod_{z\in S_x} e(g^{\tau_k+\delta}, w)^{q_z(0).\triangle_{i,S'_x}(0)}$
$= e(g^{\tau_k+\delta}, w)^{\sum_{z\in S_x} q_{parent(z)}(index(z)).\triangle_{i,S'_x}(0)}$
$= e(g^{\tau_k+\delta}, w)^{\sum_{z\in S_x} q_x(i).\triangle_{i,S'_x}(0)}$
$= e(g^{\tau_k+\delta}, w)^{q_x(0)}$,
And as mentioned before in online-encrypt phase $q_R(0) = s$ where R is the root node so for root node the F_x will be $e(g^{\tau_k+\delta}, w)^s$,
. Finally, the new ciphertext header $\widetilde{\mathcal{H}}_{\mathcal{P}} = (C, T)$ is returned.

- $DecryptUser(SK_{\mathcal{L}}, \widetilde{\mathcal{H}}_{\mathcal{P}}, C') \rightarrow M$

On receiving the new ciphertext header $\widetilde{\mathcal{H}}_{\mathcal{P}} = (C, T) = (w^{\beta s}, e(g^{\tau_k+\delta}, w)^s)$, the secret δ is used to compute $D' = D \cdot \eta^{\delta} = g^{\frac{(\alpha+\tau_k)}{\beta}} g^{\frac{\delta}{\beta}} = g^{\frac{(\alpha+\tau_k+\delta)}{\beta}}$.

Let $ek = \dfrac{e(C, D')}{T} = \dfrac{e(g^{\frac{(\alpha+\tau_k+\delta)}{\beta}}, w^{\beta s})}{e(g^{\tau_k+\delta}, w)^s} = e(g^{\alpha}, w)^s$.
Finally, the plaintext message is computed by
$M = C'/ek$.

In case of hybrid encryption the user will first get the symmetric key $syk = C'/ek$ and then we will calculate messgage $M = Dec_{syk}(\bar{C})$

5.2.3 drawback of this scheme

One major limitation to the this scheme is that the user must go to a trusted party and prove his identity in order to obtain a secret key which will allow him to decrypt messages. In this case, each user must go to the trusted server, prove that he has a certain set of attributes, and then receive secret keys corresponding to each of those attributes. However, this means we must have one trusted server who monitors all attributes who keeps records of drivers licenses, voter registration, and college enrollment. In reality, we have 3 different entities responsible for maintaining this information (the RI DMV, the RI Board of Elections, and the University office), so we would want to be able to entrust each of these to a different (and perhaps not entirely trusted) server. Also it will reduce the overall time complexity of key generation as parallaly key will be generated for different attribute subset by each of the keyGen authority. Sahai and waters first propose the scheme of multi-authority attribute based encryption in [20] to deal with this problem. In our next scheme we fit the idea of [20] in our online-offline temporal access scheme.

5.3 Scheme-5: Distributed Access Control Scheme

We propose a new variant of previous temporal access control scheme where key generation can be done in distributed manner by different key generation authorities. Here each authority will generate key for a specific subset of attributes and then a central authority will consolidate all the subset's keys and make a central key. So basically in this scheme the key generation will be done in a distributed fashion making the scheme more scalable, time efficient and secure.

5.3.1 Framework

A brief description of each of these algorithms is as follows:

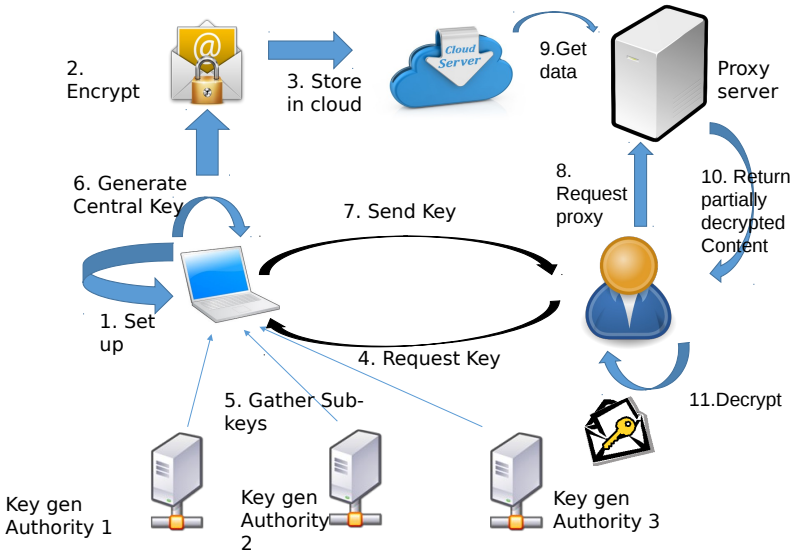


Figure 5.1: Temporal access control with Multi-Authority key generation

- Setup$(1^\kappa) \rightarrow (MK, PK)$: Takes a security parameter κ as input, outputs the master secret key MK and the public key PK.
- $KeyGen_m(MK, u_k, \mathcal{L}_m) \rightarrow SK_{\mathcal{L}_m}$
  Key generation authority m take the master secret key user id and the subset of attributes for which it need to generate the key. Then it will generate the key for the set of attributes which are the intersection of $\mathcal{L}_m$ and user u_k's attribute.
- $KeygenCentral(PK, \tau_{k_0}) \rightarrow SK_{\mathcal{L}})$
  Central authority will first get all the $\tau_{k_m} = F_{s_m}(u_k)$ and then calculate secret key g_{τ_k} and add g_{τ_k} to the central secret key for user k which is $SK_{\mathcal{L}}$.
- $OfflineEncrypt(PK, \mathcal{L}) \rightarrow (C_\rho)$: Here data owner will take the public key and the set of attributes by which message will be encrypted. Then it will generate intermediate ciphertext from that and wait till it get the access structure and plaintext.
- OnlineEncrypt$(PK, M, \mathcal{P}) \rightarrow (\mathcal{H}_\mathcal{P}, C')$: Once the access structure and plaintext is become available, a comparable access policy $\mathcal{P}$, public key PK and message M as input, outputs the ciphertext header $\mathcal{H}_\mathcal{P}$ and final ciphertext C'.
- ProxyRekey$(PK, MK, RL) \rightarrow PXK$: Takes the public key PK, master key MK, and the revocation list RL as inputs, outputs the proxy key PXK, which contains the information necessary to enable the proxy server to partially decrypt the ciphertext.

- Convert$(PXK, \mathcal{H}_{\mathcal{P}}, u_k) \rightarrow (\lambda_k, \{(R''_{t_i}, R''_{t_j})\}_{A_t[t_i,t_j]\in\mathcal{P}}, RB''_{t\ B_t\in\mathcal{P}})$: Takes the proxy key PXK, ciphertext header $\mathcal{H}_{\mathcal{P}}$, and user ID u_k as input, outputs components needed to unblind the user's private key $SK_{\mathcal{L}}$.
- Delegate$(SK_{\mathcal{L}}, \mathcal{L}') \rightarrow \widetilde{SK}_{\mathcal{L}'}$: Takes a private key $SK_{\mathcal{L}}$ and a specified privilege requirement $\mathcal{L}'$ as input, outputs a derivation key $\widetilde{SK}_{\mathcal{L}'}$, if each attribute in $\mathcal{L}$ and $\mathcal{L}'$ match.
- DecryptProxy$(\widetilde{SK}_{\mathcal{L}'}, \mathcal{H}_{\mathcal{P}}) \rightarrow \widetilde{\mathcal{H}_{\mathcal{P}}}$: Takes a users private key $\widetilde{SK}_{\mathcal{L}'}$ and the ciphertext header $\mathcal{H}_{\mathcal{P}}$ as input, outputs a new ciphertext header $\widetilde{\mathcal{H}_{\mathcal{P}}}$ if $\mathcal{L}'$ satisfies the range constraints of $\mathcal{P}$.
- DecryptUser$(SK_{\mathcal{L}}, \widetilde{\mathcal{H}_{\mathcal{P}}}, C') \rightarrow M$. Takes the users private key $SK_{\mathcal{L}}$, Ciphertext header $\widetilde{\mathcal{H}}_{\mathcal{P}}$, and the ciphertext C' as input, outputs the message M.

5.3.2 Construction

We provide detailed implementations of the algorithms mentioned in Section 5.3.1 and present our novel construction to enforce user revocation.

- *Setup*$(1^{\kappa}) \rightarrow (MK, PK)$
  This is run by the system manager. Let κ be the security parameter. The setup algorithm takes κ (which depends on the application) and outputs the master secret key MK and public key PK. We use bilinear map $\mathbb{S} = (\mathbb{G}, \mathbb{G}_T, e(\cdot,\cdot))$ of composite order $n = s'n'$ with two subgroups $\mathbb{G}_{s'}$ and $\mathbb{G}_{n'}$ of $\mathbb{G}$. Random generators $w \in \mathbb{G}$, $g \in \mathbb{G}_{s'}$, and $\varphi, \bar{\varphi} \in \mathbb{G}_{n'}$ are chosen. Two random numbers $\lambda, \mu \in \mathbb{Z}_n^*$ are selected, to implement forward and backward derivation functions. By Proposition 1, we have $e(g, \varphi) = e(g, \bar{\varphi}) = 1$, but $e(g, w) \neq 1$. A hash function $H : \{0,1\}^* \rightarrow \mathbb{G}$ is used to map an attribute to a binary string which represents a random group element. A randomly generated polynomial P of degree c (the maximum number of revoked users) over $\mathbb{Z}_n$ will be employed to enforce revocation. Next, this algorithm chooses two random exponents $\alpha, \beta \in \mathbb{Z}_n^*$ and outputs the public key

$$PK = (\mathbb{S}, n, g, h, \zeta, \eta, w, \varphi, \bar{\varphi}, \lambda, \mu, H(\cdot)),$$

  where,

$$h = w^{\beta}, \zeta = e(g, w)^{\alpha}, \eta = g^{\frac{1}{\beta}},$$

  and the master key

$$MK = (g^{\alpha}, \beta, n', P).$$

  Define a psudo-random function $F : \mathbb{Z}_n^* * \mathbb{Z}_n^* \rightarrow \mathbb{Z}_n^* *$ Also Choose seeds $s_1, \cdots, s_M$ for for all the M authorities.Also choose $\tau_0 \in \mathbb{Z}_n^*$ as the master key for central authority which will be used to combined all the authority's decrypted content.

- $KeyGen_m(MK, u_k, \mathcal{L}_m) \rightarrow SK_{\mathcal{L}_m}$
  Authority m will execute this function for the attribute set $\mathcal{L}_m$. Note that user u_k's whole attribute set is divided into some disjoint sets $\mathcal{L}_m$ for all the M authorities. Given a global user with ID u_k and access structure $\mathcal{L}_m \subseteq \mathcal{L}$ on a set of attributes $S = \{A_t\} \subseteq \mathcal{A}$, this algorithm generate a unique $\tau_{k_m} = F_{s_m}(u_k)$ for user u_k. For simplicity we are denoting τ_{k_m} simply as τ_k till the end of DecryptProxy phase.
  Assume that the user u_k is assigned a temporal attribute $A_t \in \mathcal{L}$ with the constraint $A_t[t_a, t_b]$ and non-temporal attribute $B_t \in \mathcal{L}$. This algorithm chooses a random $r \in \mathbb{Z}$ and sets the user's attribute key as

$$(D_t, D_{t_a}^{'}, \bar{D}_{t_b}^{'}, D_t^{''}, R_{t_a}^{'}, \bar{R}_{t_b}^{'}, R_t^{''})_{A_t[t_a,t_b]\in\mathcal{L}} \text{and}$$

$$(DB_t, DB_t^{'}, RB_t^{'}, RB_t^{''})_{B_t\in\mathcal{L}}$$

  where,

$$\begin{aligned}
&D_t = g^{\tau_k} H(A_t)^{r\widetilde{P}(0)},\\
&D_{t_a}^{'} = (v_{t_a})^r,\\
&\bar{D}_{t_b}^{'} = (\bar{v}_{t_b})^r,\\
&D_t^{''} = w^r,\\
&R_{t_a}^{'} = (D_{t_a}^{'})^{P(u_k)} = (v_{t_a})^{rP(u_k)},\\
&\bar{R}_{t_b}^{'} = (\bar{D}_{t_b}^{'})^{P(u_k)} = (\bar{v}_{t_b})^{rP(u_k)},\\
&R_t^{''} = (D_t^{''})^{P(u_k)} = w^{rP(u_k)},\\
&DB_t = g^{\tau_k} H(B_t)^{r\widetilde{P}(0)},\\
&RB_t^{''} = (D_t^{''})^{P(u_k)} = w^{rP(u_k)},
\end{aligned}$$

  and,

$$\begin{aligned}
&v_{t_a} = \varphi^{\lambda^{t_a}}, \bar{v}_{t_b} = \bar{\varphi}^{\mu^{Z-t_b}} \in \mathbb{G}_{n'},\\
&\widetilde{P}(0) = P(0) + 1.
\end{aligned}$$

  Finally, this algorithm outputs the user's private key

$$\begin{aligned}
SK_{\mathcal{L}_m} = (D = g^{\frac{(\alpha+\tau_{k_0})}{\beta}},\ &\{(D_t, D_{t_a}^{'}, \bar{D}_{t_b}^{'}, D_t^{''}, R_{t_a}^{'}, \bar{R}_{t_b}^{'}, R_t^{''})\}_{A_t[t_a,t_b]\in\mathcal{L}},\\
&\{(DB_t, RB_t^{''})\}_{B_t\in\mathcal{L}}).
\end{aligned}$$

- $KeygenCentral(PK, \tau_{k_0}) \rightarrow (SK_{\mathcal{L}}))$
  Central authority will first get all the $\tau_{k_m} = F_{s_m}(u_k)$ and then calculate secret key $g_{\tau_k} = g^{\tau_{k_0} - \sum_{m=1}^{M} \tau_{k_m}}$ and add g_{τ_k} to the central secret key for user k. So the central secret key will be :-

$$SK_{\mathcal{L}} = (\bigcup_{m=1}^{M} SK_{\mathcal{L}_m}, g_{\tau_k}) \tag{5.4}$$

- *OfflineEncrypt*$(PK) \rightarrow (C_\rho)$

  For each temporal attribute $A_t \in \mathcal{L}$ generate two random number $\rho_{A_{t_x}} \in \mathbb{Z}$ and $\rho_{A_{t_y}} \in \mathbb{Z}$ then calculate:-
  $\bar{E}_{t_i} = (\bar{v}_{t_i} w)^{\rho_{A_{t_x}}}$
  $E'_{t_i} = H(A_t)^{\rho_{A_{t_x}}}$
  $E_{t_j} = (v_{t_j} w)^{\rho_{A_{t_y}}}$
  $E'_{t_j} = H(A_t)^{\rho_{A_{t_y}}}$
  And for each non temporal attribute $B_t \in \mathcal{L}$ choose random number $\rho_{B_t} \in \mathbb{Z}$
  $EB'_t = w^{\rho_{B_t}}$
  $\bar{DB'} = H(B_t)^{\rho_{B_t}}$
  So the initial ciphertext generated in Offline will be :-

$$C_\rho = ((\bar{E}_{t_i}, E'_{t_i}, E_{t_j}, E'_{t_j})_{A_t[t_i,t_j] \in \mathcal{P}}, \{(EB'_t, \bar{DB'})\}_{B_t \in \mathcal{P}}). \tag{5.5}$$

- *OnlineEncrypt*$(PK, {\mathcal{P}_m}_{m=1}^{M}, M) \rightarrow (\mathcal{H}_\mathcal{P}, C')$

  Data owner will do the encryption of data.Let $\mathcal{T}_m$ be the access tree for the access policy $\mathcal{P}_m$. The algorithm first chooses a polynomial q_x for each node x (including the leaves) in the tree T . These polynomials are chosen in the following way in a topdown manner, starting from the root node R. For each node x in the tree, set the degree d_x of the polynomial q_x to be one less than the threshold value k_x of that node, that is, $d_x = k_x - 1$.
  The algorithm first chooses a random $s \in Z_p$ Starting with the root node R of each access tree $\mathcal{T}_m$ and sets $q_R(0) = s$. Then, it chooses d_R other points of the polynomial q_R randomly to define it completely. For any other node x, it sets $q_x(0) = q_{parent(x)}(index(x))$ and chooses d_x other points randomly to completely define q_x.Please refer to Section 4, [12] for a detailed meaning of access structure and threshold.

  For every temporal attribute calculate ρ_{A_t} such that $\rho_{A_t} = q_{A_t}(0) - (\rho_{A_{t_x}} + \rho_{A_{t_y}})$, where $q_{A_t}(0)$ is the secret share at leaf node corresponding to attribute A_t .
  And for every non-temporal attribute B_t choose ρ'_B such that $\rho_{B_t} + \rho'_B = q_{B_t}(0)$
  Given an access tree $\mathcal{T}$, the ciphertext is composed of a ciphertext header,

$$\mathcal{H}_\mathcal{P} = (\mathcal{T}, E_{CA} = w^s, C = h^s, C_\rho, \{\rho_{A_t}\}_{A_t[t_i,t_j] \in \mathcal{P}} \{\rho'_B\}_{B_t \in \mathcal{P}}). \tag{5.6}$$

  and a ciphertext $C' = Me(g, w)^{s\alpha}$.

  In case the message size is huge we will apply hybrid encryption where the data owner choose a random symmetric key $syk \in \mathbb{Z}$ and compute $C' = syk * e(g, w)^{s\alpha}$

and the ciphertext of the message will be $\bar{C} = Enc_{syk}(M)$ where Enc is any symmetric key encryption according to user's choice.
Then $\bar{C}$ will be added to $\mathcal{H}_{\mathcal{P}}$.

- *ProxyRekey*$(PK, MK, RL) \rightarrow PXK$
  The data owner chooses a polynomial P of degree c, with coefficients in Z_n^*. Let RL be the revocation list and $u_i, i \in \{1, 2, \cdots, c\}$, be the identities of the revoked users. The owner evaluates the polynomial $P(u_i)$ at these points, using the master secret key MK. If there are less than c revoked users, then the owner generates random points x and evaluates $P(x)$ such that x does not correspond to any user's identity. This ensures that the proxy key PXK is of fixed length.

$$PXK = \forall u_i \in RL :< u_i, P(u_i) >$$

- *Convert*$(PXK, H_{\mathcal{P}}, u_k) \rightarrow$
  $(\lambda_k, \{(R_{t_i}^{''}, R_{t_j}^{''})\}_{A_t[t_i,t_j]\in\mathcal{P}}, \{RB_t^{''}\}_{B_t\in\mathcal{P}})$
  RL will be stored in a hash table of proxy server. Every time revocation happen the data owner inform proxy and it will update the RL accordingly. Given the proxy key PXK, ciphertext header $H_{\mathcal{P}}$, and the user ID u_k, the proxy calculates

$$\lambda_i = \frac{u_k}{u_k - u_i} \prod_{j \neq i} \frac{u_j}{u_j - u_i}, \forall i, j \in \{1, \cdots, c\},$$
$$k \notin \{1, \cdots, c\}$$

  For every attribute $A_t[t_i, t_j] \in \mathcal{P}$, it calculates

$$\begin{aligned}
R_{t_i}^{''} &= (E_{t_i}^{'})^{\sum_{i=1}^{c} \lambda_i P(u_i)} = H(A_t)^{\rho_{A_{t_x}} \sum_{i=1}^{c} \lambda_i P(u_i)}, \\
R_{t_j}^{''} &= (E_{t_j}^{'})^{\sum_{i=1}^{c} \lambda_i P(u_i)} = H(A_t)^{\rho_{A_{t_y}} \sum_{i=1}^{c} \lambda_i P(u_i)}, \\
RB_t^{'''} &= (EB_t^{'})^{\sum_{i=1}^{c} \lambda_i P(u_i)} = H(B_t)^{\rho_{B_t} \sum_{i=1}^{c} \lambda_i P(u_i)}.
\end{aligned}$$

  Since the user's private key $SK_{\mathcal{L}}$ is blinded by $\widetilde{P}(0)$, it additionally needs $R_{t_i}^{''}$, $R_{t_j}^{''}$ for decryption. The proxy also computes λ_k and gives it to the user with ID u_k.

- *Delegate*$(SK_{\mathcal{L}}, \mathcal{L}') \rightarrow \widetilde{SK}_{\mathcal{L}'}$

  Given the private key $SK_{\mathcal{L}}$ and a specified $\mathcal{L}'$, this algorithm checks whether, for every $A_t[t_a, t_b] \in \mathcal{L}$ and $A_t[t_i, t_j] \in \mathcal{L}'$, $t_a \leq t_j$ and $t_b \geq t_i$ is true for each

attribute $A_t \in \mathcal{L}'$.
If true, the user computes,

$$\begin{aligned}
D'_{t_j} &\leftarrow f_{t_a \le t_j}(D'_{t_a}) \cdot D''_t = (v_{t_j} w)^r \\
\bar{D}'_{t_i} &\leftarrow \bar{f}_{t_b \ge t_i}(\bar{D}'_{t_b}) \cdot D''_t = (\bar{v}_{t_i} w)^r \\
R'_{t_j} &\leftarrow f_{t_a \le t_j}(R'_{t_a}) = (v_{t_j})^{rP(u_k)} \\
\bar{R}'_{t_i} &\leftarrow \bar{f}_{t_b \ge t_i}(\bar{R}'_{t_b}) = (\bar{v}_{t_i})^{rP(u_k)}
\end{aligned}$$

Next, this algorithm chooses a random $\delta \in \mathbb{Z}$ and computes
$\widetilde{SK}_{\mathcal{L}'} = \{\widetilde{D}_t, \widetilde{D}'_{t_j}, \widetilde{\bar{D}'}_{t_i}, R_{t_j}, \bar{R}_{t_i}\}_{A_t \in \mathcal{L}'}, \{\widetilde{DB}_t, \widetilde{DB}'_t, RB_t,\}_{B_t \in \mathcal{L}'}$, where,

$$\begin{aligned}
\widetilde{D}_t &= D_t \cdot (gH(A_t))^{\delta} = g^{\tau_k + \delta} H(A_t)^{r\widetilde{P}(0) + \delta} \\
\widetilde{D}'_{t_j} &= D'_{t_j} \cdot (v_{t_j} w)^{\delta} = (v_{t_j} w)^{r+\delta} \\
\widetilde{\bar{D}'}_{t_i} &= \bar{D}'_{t_i} \cdot (\bar{v}_{t_i} w)^{\delta} = (\bar{v}_{t_i} w)^{r+\delta} \\
\widetilde{DB}_t &= DB_t \cdot (gH(A_t))^{\delta} = g^{\tau_k + \delta} H(B_t)^{r\widetilde{P}(0) + \delta} \\
\widetilde{DB}'_t &= D'_{t_j} \cdot w^{\delta} = w^{r+\delta}
\end{aligned}$$

$$\begin{aligned}
R_{t_j} &= e(R''_t \cdot R'_{t_j}, E'_{t_j})^{\lambda_k} \cdot e(D'_{t_j}, R''_{t_j}) \\
&= e(v_{t_j} w, H(A_t))^{\rho_{A_{t_y}} r(\lambda_k P(u_k) + \sum_{i=1}^{t} \lambda_i P(u_i))} \\
&= e(v_{t_j} w, H(A_t))^{\rho_{A_{t_y}} rP(0)} \\
\bar{R}_{t_i} &= e(R''_t \cdot \bar{R}'_{t_i}, E'_{t_i})^{\lambda_k} \cdot e(\bar{D}'_{t_i}, \bar{R}''_{t_i}) \\
&= e(\bar{v}_{t_i} w, H(A_t))^{\rho_{A_{t_x}} r(\lambda_k P(u_k) + \sum_{i=1}^{t} \lambda_i P(u_i))} \\
&= e(\bar{v}_{t_i} w, H(A_t))^{\rho_{A_{t_x}} rP(0)} \\
RB_t &= e(RB''_t, EB'_t)^{\lambda_k} \cdot e(DB'_t, RB'''_t) \\
&= e(w, H(B_t))^{\rho_{B_t} P(0)}
\end{aligned}$$

Finally, it outputs $\widetilde{SK}_{\mathcal{L}'}$ as the derivation key for $\mathcal{L}'$.

- *DecryptProxy*$(\widetilde{SK}_{\mathcal{L}'}, \mathcal{H}_{\mathcal{P}}, u_k) \rightarrow \widetilde{\mathcal{H}}_{\mathcal{P}}$

On receiving the private key $\widetilde{SK}_{\mathcal{L}'}$, and the ciphertext header $\mathcal{H}_{\mathcal{P}}$, this algorithm checks whether the user u_k is present in the hash table RL. If not then it will check each range attribute $A_t[t_i, t_j] \in \mathcal{L}'$ is consistent with $A_t[t_i, t_j] \in \mathcal{P}$. If

true, the secret share $q_{A_t}(0)$ over $\mathbb{G}_T$ is reconstructed as follows:-

$$\begin{aligned}
F_1 &\longleftarrow \frac{e(\widetilde{D}_t, E_{t_j})}{R_{t_j} \cdot e(\widetilde{D}'_{t_j}, E'_{t_j})} \\
&= \frac{e(g^{\tau_k+\delta} H(A_t)^{r\widetilde{P}(0)+\delta}, (v_{t_j} w)^{\rho_{A_{t_y}}})}{R_{t_j} \cdot e((v_{t_j} w)^{r+\delta}, H(A_t)^{\rho_{A_{t_y}}})} \\
&= \frac{e(g^{\tau_k+\delta}, (v_{t_j} w)^{\rho_{A_{t_y}}}) \cdot e(H(A_t)^{r\widetilde{P}(0)+\delta}, (v_{t_j} w)^{\rho_{A_{t_y}}})}{R_{t_j} \cdot e((v_{t_j} w)^{r+\delta}, H(A_t)^{\rho_{A_{t_y}}})} \\
&= \frac{e(g^{\tau_k+\delta}, (v_{t_j} w)^{\rho_{A_{t_y}}}) \cdot e(v_{t_j} w, H(A_t))^{\rho_{A_{t_y}} rP(0)}}{R_{t_j}} \\
&= e(g^{\tau_k+\delta}, v_{t_j}^{\rho_{A_{t_y}}}) \cdot e(g^{\tau_k+\delta}, w^{\rho_{A_{t_y}}}) \\
&= e(g^{\tau_k+\delta}, w)^{\rho_{A_{t_y}}}
\end{aligned}$$

Similarly,

$$\begin{aligned}
F_2 &\longleftarrow \frac{e(\widetilde{D}_t, \bar{E}_{t_i})}{\bar{R}_{t_i} \cdot e(\widetilde{\bar{D}'}_{t_i}, E'_{t_i})} \\
&= \frac{e(g^{\tau_k+\delta} H(A_t)^{rP'(0)+\delta}, (\bar{v}_{t_i} w)^{\rho_{A_{t_x}}})}{\bar{R}_{t_i} \cdot e((\bar{v}_{t_i} w)^{r+\delta}, H(A_t)^{\rho_{A_{t_x}}})} \\
&= \frac{e(g^{\tau_k+\delta}, (\bar{v}_{t_i} w)^{\rho_{A_{t_x}}}) \cdot e(H(A_t)^{rP'(0)+\delta}, (\bar{v}_{t_i} w)^{\rho_{A_{t_x}}})}{\bar{R}_{t_i} \cdot e((\bar{v}_{t_i} w)^{r+\delta}, H(A_t)^{\rho_{A_{t_x}}})} \\
&= \frac{e(g^{\tau_k+\delta}, (\bar{v}_{t_i} w)^{\rho_{A_{t_x}}}) \cdot e(\bar{v}_{t_i} w, H(A_t))^{\rho_{A_{t_x}} rP(0)}}{\bar{R}_{t_i}} \\
&= e(g^{\tau_k+\delta}, \bar{v}_{t_i}^{\rho_{A_{t_x}}}) \cdot e(g^{\tau_k+\delta}, w^{\rho_{A_{t_x}}}) \\
&= e(g^{\tau_k+\delta}, w)^{\rho_{A_{t_x}}}
\end{aligned}$$

For $B_t \in \mathcal{L}'$ is consistent with $B_t \in \mathcal{P}$, then the secret share $q_{B_t}(0)$ of s over $\mathbb{G}_T$ is reconstructed as,

$$
\begin{aligned}
F_3 \quad \longleftarrow \quad & \frac{e(\widetilde{D}_t, \bar{EB}_t)}{\bar{RB}_t \cdot e(\widetilde{\bar{DB}'}_t, EB'_t)} \\
= \quad & \frac{e(g^{\tau_k+\delta} H(B_t)^{rP'(0)+\delta}, w^{\rho_{B_t}})}{RB_t \cdot e(w^{r+\delta}, H(B_t)^{\rho_{B_t}})} \\
= \quad & \frac{e(g^{\tau_k+\delta}, w^{\rho_{B_t}}) \cdot e(H(B_t)^{r\widetilde{P}(0)+\delta}, w^{\rho_{B_t}})}{RB_t \cdot e(w^{r+\delta}, H(B_t)^{\rho_{B_t}})} \\
= \quad & \frac{e(g^{\tau_k+\delta}, w^{\rho_{B_t}}) \cdot e(w, H(B_t))^{\rho_{B_t} rP(0)}}{\bar{RB}_t} \\
= \quad & e(g^{\tau_k+\delta}, w^{\rho_{B_t}}) \\
= \quad & e(g^{\tau_k+\delta}, w)^{\rho_{B_t}}
\end{aligned}
$$

$F_t = F_1 \cdot F_2 = e(g^{\tau_k+\delta}, w)^{(\rho_{A_{t_x}} + \rho_{A_{t_y}})}$

where, $e(g^{\tau_k+\delta}, v^y_{t_j}) = e(g^{\tau_k+\delta}, \bar{v}^x_{t_i}) = 1$ because $g^{\tau_k+\delta} \in \mathbb{G}_{s'}$ and $v^y_{t_j}, \bar{v}^x_{t_i} \in \mathbb{G}_{n'}$. Next, the value of $T_m = e(g^{\tau_k+\delta}, w)^s$ is computed from $\{e(g^{\tau_k+\delta}, w)^{q_{A_i}(0)}\}_{A_i \in \mathcal{P}}$ and $\{e(g^{\tau_k+\delta}, w)^{q_{B_i}(0)}\}_{B_i \in \mathcal{P}}$ by using the recursive DecryptNode algorithm below:-

We now consider the recursive case when x is a non-leaf node. The algorithm DecryptNode(CT, SK,x) then proceeds as follows:
For all nodes z that are children and all z are leaf node of x, it calls DecryptNode(CT, SK,z) and stores the output(F_t for leaf node with temporal attribute and F_3 for leaf node with non-temporal attribute) as F_z. Let S_x be an arbitrary k_x-sized set of child nodes z such that $F_z \neq \perp$. If no such set exists then the node was not satisfied and the function returns $\perp$. Otherwise, we first compute
$(e(D,h)/\zeta) * e(e,w)^{\delta}$
$= (e(g,w^{\beta})^{\alpha+\tau_k/\beta}/e(g,w)^{\alpha}) * e(e,w)^{\delta}$
$= e(g,w)^{\tau_k} * e(e,w)^{\delta}$
$= e(g,w)^{\tau_k+\delta}$

Then it will calculate recursively:-
$F_x = e(g^{\tau_k+\delta}, w)^{\sum_{z \in S_x} \rho_{A_t}} \prod_{z \in S_x} F_z^{\triangle_{i,S'_x}(0)}$, (where i=index(z) $S'_x = index(z) : z \in S_x$)
Note that in the R.H.S in the first term ρ_{A_t} is mentioned assuming it is temporal attribute. But in case of non-temporal it will be ρ'_{B_t}. For simplicity we are using ρ_{A_t} for the rest of the equations. So
$F_x =$
$e(g^{\tau_k+\delta}, w)^{\sum_{z \in S_x} \rho_{A_t}} \prod_{z \in S_x} e(g^{\tau_k+\delta}, w)^{(\rho_{A_{t_x}} + \rho_{A_{t_y}}) \triangle_{i,S'_x}(0)}$

$= e(g^{\tau_k+\delta}, w)^{\sum_{z\in S_x} \rho_{A_t}} . e(g^{\tau_k+\delta}, w)^{\sum_{z\in S_x} (\rho_{A_{t_x}}+\rho_{A_{t_y}})^{\triangle_{i,S'_x}(0)}}$
$= e(g^{\tau_k+\delta}, w)^{\sum_{z\in S_x} (\rho_{A_t}+\rho_{A_{t_x}}+\rho_{A_{t_y}})^{\triangle_{i,S'_x}(0)}}$
$= e(g^{\tau_k+\delta}, w)^{\sum_{z\in S_x} (q_{A_t}(0))^{\triangle_{i,S'_x}(0)}}$

now $q_{A_t}(0) = q_{parent(z)(index(z))} = q_x(i)$

So the above equation will be:-
$F_x = e(g^{\tau_k+\delta}, w)^{\sum_{z\in S_x} q_x(i)^{\triangle_{i,S'_x}(0)}}$,
$F_x = e(g^{\tau_k+\delta}, w)^{q_x(0)}$,

(using polynomial interpolation).
Now suppose for all non-leaf node x whose children are also non-leaf node suppose z the value of F_x will be :-

$\prod_{z\in S_x} F_z^{\triangle_{i,S'_x}(0)}$
$=\prod_{z\in S_x} e(g^{\tau_k+\delta}, w)^{q_z(0).\triangle_{i,S'_x}(0)}$
$= e(g^{\tau_k+\delta}, w)^{\sum_{z\in S_x} q_{parent(z)}(index(z)).\triangle_{i,S'_x}(0)}$
$= e(g^{\tau_k+\delta}, w)^{\sum_{z\in S_x} q_x(i).\triangle_{i,S'_x}(0)}$
$= e(g^{\tau_k+\delta}, w)^{q_x(0)}$,

And as mentioned before in online-encrypt phase $q_R(0) = s$ where R is the root node so for root node the F_x will be $e(g^{\tau_k+\delta}, w)^s$,
Now for M different authority as mentioned earlier τ_k will be different. For authority m τ_k is τ_{k_m}
Now calculate

$$T = e(E_{CA}, g_{\tau_k}) \prod_{m=1}^{M} T_m \tag{5.7}$$

$$= e(E_{CA}, g_{\tau_k}) \prod_{m=1}^{M} e(g^{\delta+\tau_{k_m}}, w)^s \tag{5.8}$$

$$= e(w^s, g^{\tau_{k_0}-\sum_{m=1}^{M}\tau_{k_m}}) e(w^s, g)^{\sum_{m=1}^{M}(\delta+\tau_{k_m})} \tag{5.9}$$

$$= e(w^s, g)^{\tau_{k_0}-\sum_{m=1}^{M}\tau_{k_m}+\sum_{m=1}^{M}\tau_{k_m}+M*\delta} \tag{5.10}$$

$$= e(w^s, g)^{\tau_{k_0}+M*\delta} \tag{5.11}$$

$$\tag{5.12}$$

. Finally, the new ciphertext header $\widetilde{\mathcal{H}}_{\mathcal{P}} = (C, T)$ is returned.

- $DecryptUser(SK_{\mathcal{L}}, \widetilde{\mathcal{H}}_{\mathcal{P}}, C') \rightarrow M$

  On receiving the new ciphertext header $\widetilde{\mathcal{H}}_{\mathcal{P}} = (C, T) = (w^{\beta s}, e(g^{\tau_{k_0}+M\delta}, w)^s)$, the secret δ is used to compute $D' = D \cdot \eta^{M\delta} = g^{\frac{(\alpha+\tau_{k_0})}{\beta}} g^{\frac{M\delta}{\beta}} = g^{\frac{(\alpha+\tau_{k_0}+M\delta)}{\beta}}$.

  Let $ek = \dfrac{e(C, D')}{T} = \dfrac{e(g^{\frac{(\alpha+\tau_{k_0}+M\delta)}{\beta}}, w^{\beta s})}{e(g^{\tau_{k_0}+M\delta}, w)^s} = e(g^{\alpha}, w)^s$.
  Finally, the plaintext message is computed by
  $M = C'/ek$.

  In case of hybrid encryption the user will first get the symmetric key $syk = C'/ek$ and then we will calculate messgage $M = Dec_{syk}(\bar{C})$

Chapter 6

Complexity Analysis

In this chapter, we analyze the performance of our proposed scheme. First, we describe the notations used in our complexity analysis. We go on to consider each algorithm and analyze both its computational as well as communication complexity. We show that the complexity of our scheme is consistent with comparison-based encryption and is thereby efficient.

6.1 Notations

We use a number of notations to denote the time taken for various operations in order to concisely represent the complexity of our scheme. Those are present in table 6.1:

6.2 Complexity comparision in form of table

We have used abbriviated terms for each of the five scheme in the diagrams and table. They are the following:-

- KPABE : The scheme of Key Policy Attribute Based Encryption ([2])
- CPABE : The scheme of Cipher Policy Attribute Based Encryption ([1])
- CBABE : The scheme of comparision Based access control ([6])
- TAABE : The scheme of Temporal Access Control ([24])
- OOABE : Our scheme Scheme-4: Temporal Access Control with added two phase encryption
- MAABE : Our scheme Scheme-5: distributed Access Control Scheme (added Multi-Authority key generation)

Table 6.1: Notations

Symbol	Description
$(\mathbb{E}_T/\mathbb{E}_1)$	time taken for exponentiation in the group $\mathbb{G}_T/\mathbb{G}_1$
$(\mathbb{M}_T/\mathbb{M}_1)$	time taken for exponentiation in $\mathbb{G}_T/\mathbb{G}_1$
$\mathbb{D}_T$	time taken Division in $\mathbb{G}_\mathbb{T}$
$\mathbb{S}_{\mathbb{Z}_ı}$	time taken for sum in $\mathbb{Z}_ı$
$\mathbb{M}_{\mathbb{Z}_ı}$	time taken for multiplication in $\mathbb{Z}_ı$
S_A	number of leaf nodes in the access tree
S_{A_t}	number of temporal attributes among the leaf nodes in the access tree
S_{A_n}	number of nontemporal attributes among the leaf nodes in the access tree
S_D	set of all attributes by which decryptor is going to decrypt
S_{D_t}	set of temporal attributes by which decryptor is going to decrypt
S_{D_n}	set of non-temporal attributes by which decryptor is going to decrypt
$\mathbb{P}$	Time required for pairing operation
$\mathbb{H}$	Time required for hash using H
$\mathbb{P}_{enc}$	complexity of internal node required for encryption
$\|m\|$	size of the message
$\|\mathbb{G}_T\|/\|\mathbb{G}_1\|/\|\mathbb{Z}_p\|$	Size of the group $\mathbb{G}_T/\mathbb{G}_1/\mathbb{Z}_p$
$\|\tau\|$	size of the access structure
$\|\gamma\|$	size of the attribute set
au	Number of authorities in case of Multi-authority attribute based encryption

Table 6.2: Comparing computational complexity of key generation with other scheme

Scheme	Key generation cost
CPABE [12]	$S_{A_t}(2\mathbb{E}_1 + \mathbb{M}_1 + H) + 2\mathbb{E}_1 + \mathbb{S}_{z_1} + \mathbb{D}_{z_1}$
CBABE [6]	$3\mathbb{E}_1 + \mathbb{S}_{z_1} + \mathbb{D}_{z_1} + S_{A_t}(3\mathbb{E}_1 + \mathbb{M}_1 + H + 2\mathbb{M}_{z_1})$
TAABE [24]	$3\mathbb{E}_1 + \mathbb{M}_t + S_{A_t}(6\mathbb{E}_1 + \mathbb{M}_1 + H) + S_{A_n}(2\mathbb{E}_1 + \mathbb{M}_1 + H)$
OOABE	Same as temporal access control
MAABE	$(OOABE)/au + S_{z_1}(au + 1) + \mathbb{E}_1$

Table 6.3: Comparing computational complexity of encryption with other scheme

Scheme	Offline Computational Complexity of encryption	Online Computational Complexity of encryption
KPABE [2]		$\mathbb{S}_A\mathbb{E}_1 + \mathbb{E}_T + \mathbb{M}_T$
CPABE [12]		$\mathbb{E}_T + \mathbb{M}_T + \mathbb{E}_1 + S_A(2\mathbb{E}_1 + H) + \mathbb{P}_e nc + S_A$
CBABE [6]		$\mathbb{E}_1 + S_{A_t}(4\mathbb{E}_1 + H + 2\mathbb{M}_1) + \mathbb{P}_e nc + S_A + S_{A_t}\mathbb{S}_z$
TAABE [24]		$\mathbb{E}_T + \mathbb{M}_T + S_{A_t}(4\mathbb{E}_1 + H + 2\mathbb{M}_1 + \mathbb{S}_z) + \mathbb{P}_e nc + S_{A_n}(H + 2\mathbb{E}_1)$
OOABE	$S_{A_t}(2R + 4\mathbb{E}_1 + H + 2\mathbb{M}_1 + \mathbb{S}_z) + S_{A_n}(H + 2\mathbb{E}_1)$	$\mathbb{P}_{enc} + \mathbb{S}_z S_A + \mathbb{E}_1 + \mathbb{M}_T + \mathbb{E}_T$
MAABE	complexity of OOABE	complexity of OOABE $+\mathbb{E}_T$

Table 6.4: Comparing decryption complexity with other scheme

Scheme	Delegete	DecryptProxy	DecryptUser
KPABE [2]			$\mathbb{PS}_D + \mathbb{E}_T\mathbb{P}_{enc} + \mathbb{D}_T$
CPABE [12]			$\mathbb{PS}_D + \mathbb{E}_T\mathbb{P}_{enc} + 2\mathbb{D}_T + \mathbb{P}$
CBABE [6]		$\mathbb{E}_T\mathbb{P}_{enc} + 2\mathbb{PS}_D$	$\mathbb{D}_T + \mathbb{P} + \mathbb{E}_1 + \mathbb{M}_1$
TAABE [24]	$\mathbb{S}_{D_t}(7\mathbb{E}_1 + 10\mathbb{M}_1 + H + 4P + 2\mathbb{E}_T) +$ $\mathbb{S}_{D_n}(2\mathbb{M}_1 + \mathbb{E}_1 + H + \mathbb{E}_T + 2P) + \mathbb{E}_1 + \mathbb{M}_1$	$\mathbb{S}_{D_t}(3\mathbb{M}_T + 4P + 2\mathbb{D}_T) +$ $\mathbb{S}_{D_n}(\mathbb{M}_T + \mathbb{D}_T + 2P) + \mathbb{P}_{enc}$	$2\mathbb{D}_T + \mathbb{P} + \mathbb{E}_1 + \mathbb{M}_1$
OOABE	Time comlexity of Temporal access scheme	Time comlexity of Temporal access scheme + $2P + \mathbb{D}_T + \mathbb{M}_T + \mathbb{S}_z\mathbb{S}_{D_t} + O(\mathbb{S}_{D_T})\mathbb{M}_T$	Time comlexity of Temporal access scheme
MAABE	Time comlexity of Temporal access scheme	Time comlexity of OOABE + $P + \mathbb{M}_T(au + 1)$	Time comlexity of OOABE $+\mathbb{M}_{z_1}$

Table 6.5: Comparing Ciphertext size or communication cost with other scheme

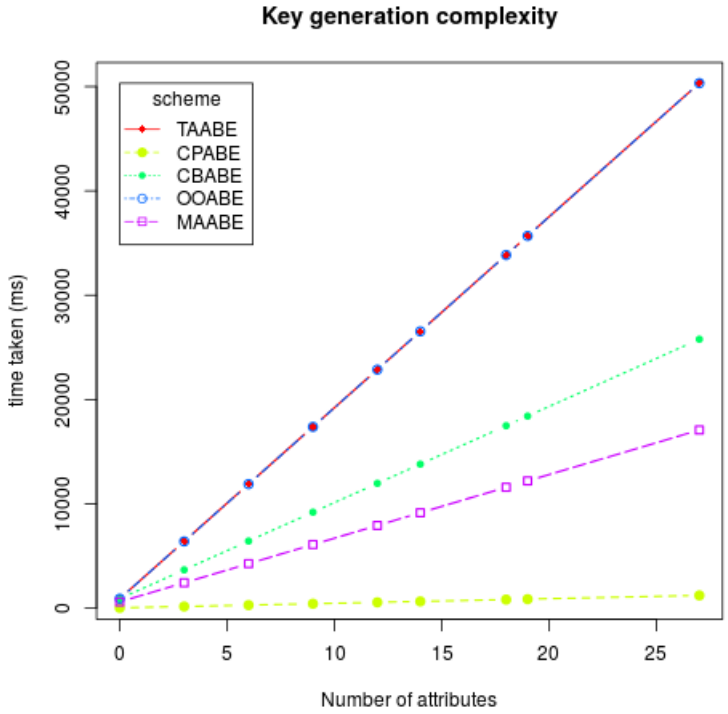
Scheme	Ciphertext size
KPABE [2]	$\lvert\mathbb{G}_T\rvert + \lvert\mathbb{G}_1\rvert\mathbb{S}_A + \lvert\gamma\rvert$
CPABE [12]	$\lvert\mathbb{G}_T\rvert + \lvert 2\mathbb{G}_1\rvert\mathbb{S}_A + \lvert\tau\rvert$
CBABE [6]	$\lvert\mathbb{G}_T\rvert + \lvert\mathbb{G}_1\rvert + 4\lvert\mathbb{G}_1\rvert\mathbb{S}_A + \lvert\tau\rvert$
TAABE[24]	$\lvert\mathbb{G}_T\rvert + \lvert\mathbb{G}_1\rvert + 4\lvert\mathbb{G}_1\rvert\mathbb{S}_{A_t} + 2\lvert\mathbb{G}_1\rvert\mathbb{S}_{A_n} + \lvert\tau\rvert$
OOABE	size of temporal access scheme ciphertext $+2\lvert\mathbb{Z}_p\rvert\mathbb{S}_A$
MAABE	size of OOABE ciphertext $+\lvert\mathbb{G}_1\rvert$

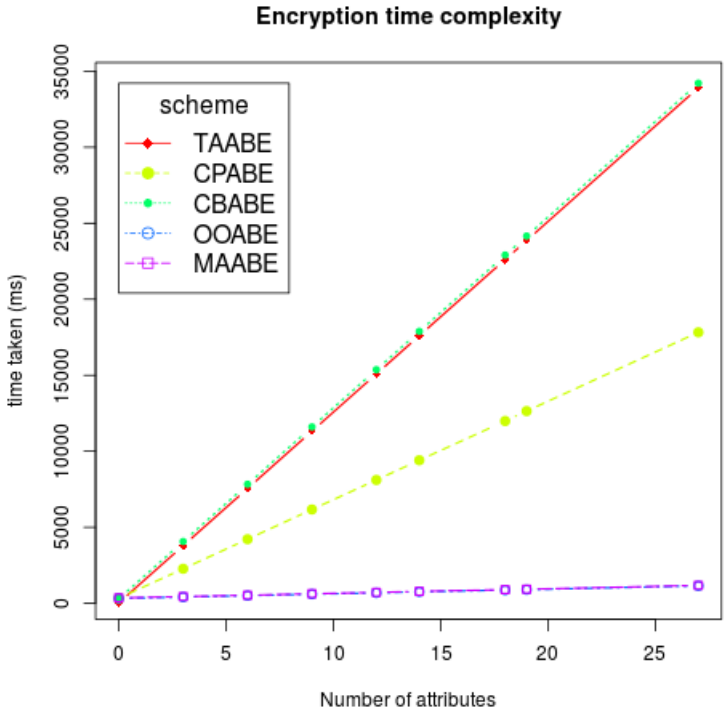
6.3 Comparing Complexity comparision in the form of graph

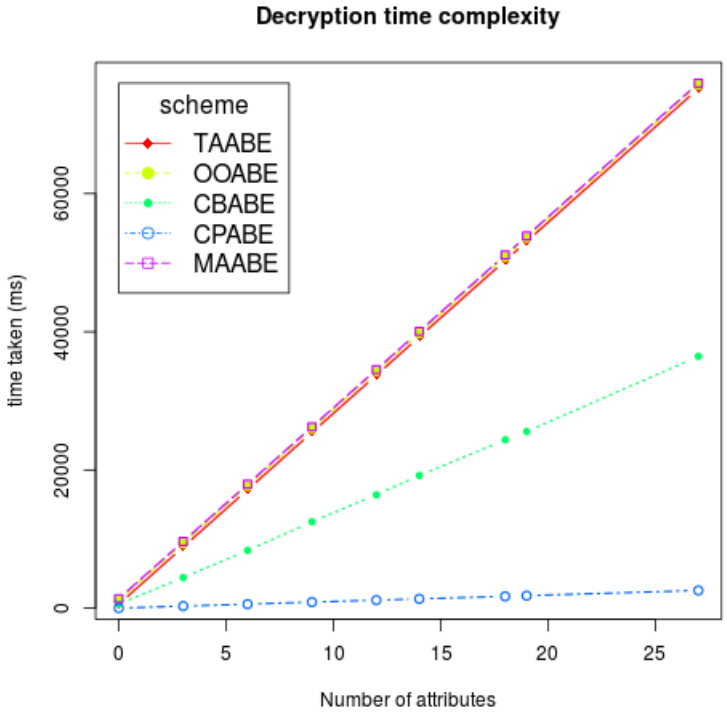
We did complexity analysis on computer with Intel Core i3-5010U CPU @ 2.10GHz*4 with 4 GB of RAM and Ubuntu 14.04 using JPBC library We used two type of bilinear map. 1st one is type-A(prime order with $q = 512$ and $r = 160$ bits) for CPABE and KPABE and other is type-A1 (composite order pairing with 2 prime and size of each prime is 512 bits) for comparison based encryption, Temporal access based encryption, Online-offline temporal access encryption and Multi-Authority temporal access encryption.

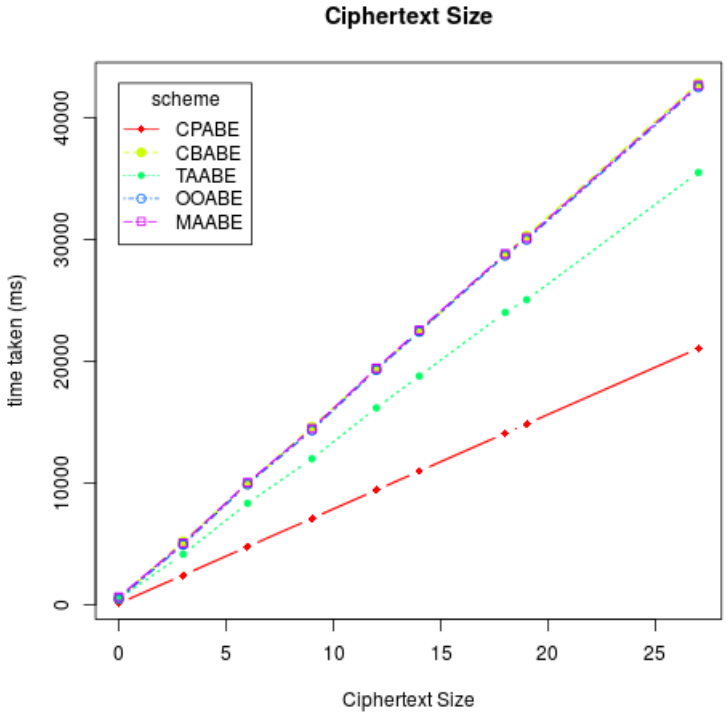
The assumption here are:-

- as there is no non-temporal attribute in comparision based ABE so we are assuming that there is no non-temporal attribute in temporal access control ABE, online-offline ABE and multi-authority ABE as well for better understanding.
- In case of multi-authority ABE the number of authority we are assuming is 3.
- The set of access trees by which we have done the experiment is put on `https://github.com/ayanDas-isi/TemporalAccess/blob/master/StructureVsTimeTaken.txt`
- Also the code for generating these graphs from the attribute set is put on `https://github.com/ayanDas-isi/TemporalAccess/blob/master/cpabeTimeTest.zip` . Anyone interested can have a look at this link.




Encryption time complexity
scheme
TAABE
CPABE
CBABE
OOABE
MAABE
time taken (ms)
0
5000
10000
15000
20000
25000
30000
35000
0
5
10
15
20
25
Number of attributes


Decryption time complexity
scheme
TAABE
OOABE
CBABE
CPABE
MAABE
time taken (ms)
0
20000
40000
60000
0
5
10
15
20
25
Number of attributes


Ciphertext Size
scheme
CPABE
CBABE
TAABE
OOABE
MAABE
time taken (ms)
0
10000
20000
30000
40000
0
5
10
15
20
25
Ciphertext Size

Chapter 7

Security proof

7.1 Security for collusion privilage attack

We depend on the confidentiality of r to guarantee the security of scheme against collusion privilege attacks. For sake of clarity, we only consider the collusion attacks by two adversaries to analyze all possible cases. For example, two users, u_a and u_b , intend to transfer the u_as a range attribute

$$
\begin{aligned}
&D_t = g^{\tau_a} H(A_t)^{r\widetilde{P}(0)},\\
&D_{t_{a_1}}^{'} = (v_{t_{a_1}})^r,\\
&\bar{D}_{t_{b_1}}^{'} = (\bar{v}_{t_{b_1}})^r,\\
&D_t^{''} = w^r,\\
&R_{t_{a_1}}^{'} = (D_{t_{a_1}}^{'})^{P(u_a)} = (v_{t_{a_1}})^{rP(u_a)},\\
&\bar{R}_{t_{b_1}}^{'} = (\bar{D}_{t_{b_1}}^{'})^{P(u_a)} = (\bar{v}_{t_{b_1}})^{rP(u_a)},\\
&R_t^{''} = (D_t^{''})^{P(u_a)} = w^{rP(u_a)},
\end{aligned}
$$

to u_b's range attribute

$$
\begin{aligned}
&D_t = g^{\tau_b} H(A_t)^{r\widetilde{P}(0)},\\
&D_{t_{a_2}}^{'} = (v_{t_{a_2}})^r,\\
&\bar{D}_{t_{b_2}}^{'} = (\bar{v}_{t_{b_2}})^r,\\
&D_t^{''} = w^r,\\
&R_{t_{a_2}}^{'} = (D_{t_{a_2}}^{'})^{P(u_b)} = (v_{t_{a_2}})^{rP(u_b)},\\
&\bar{R}_{t_{b_2}}^{'} = (\bar{D}_{t_{b_2}}^{'})^{P(u_b)} = (\bar{v}_{t_{b_2}})^{rP(u_b)},\\
&R_t^{''} = (D_t^{''})^{P(u_b)} = w^{rP(u_b)},
\end{aligned}
$$

7.1.1 Case 1.1

One possibility is to extract g^{τ_a} and $H(A_t)^{r\widetilde{P}(0)}$ from D_t . And to form a private key like :-

$$
\begin{aligned}
&D_t = g^{\tau_b} H(A_t)^{r\widetilde{P}(0)},\\
&D_{t_{a_1}}^{'} = (v_{t_{a_1}})^r,\\
&\bar{D}_{t_{b_1}}^{'} = (\bar{v}_{t_{b_1}})^r,\\
&D_t^{''} = w^r,\\
&R_{t_{a_1}}^{'} = (D_{t_{a_1}}^{'})^{P(u_a)} = (v_{t_{a_1}})^{rP(u_a)},\\
&\bar{R}_{t_{b_1}}^{'} = (\bar{D}_{t_{b_1}}^{'})^{P(u_a)} = (\bar{v}_{t_{b_1}})^{rP(u_a)},\\
&R_t^{''} = (D_t^{''})^{P(u_a)} = w^{rP(u_a)},
\end{aligned}
$$

We call it CPA-1(chosen plaintext attack). For CPA-1 we will prove the following theorem :-

Theorem 1 *Given a TA(temporal access) cryptosystem over the RSA type elliptic curve system S_N, It is impossible to extract the values g_{τ_a} or $H(A_t)^{rP(0)}$ from the user's key SK_L if computational Co-Diffie-Hellman assumption holds.*
This theorem shows that the colluders cannot forge a new key by exchanging g^{τ_a} and $H(A_t)^{rP(0)}$ from some known private keys.Hence our scheme can resist the CPA-1 type attack.

Proof of the theorem is given at the end.

7.1.2 Case 1.2

For CPA-II attacks, the attackers try to replace all the t_{a_1}, t_{b_1} by t_{a_2}, t_{b_2} , where $t_{a_2} < t_{a_1} < t_{b_1} < t_{b_2}$. However, the confidentiality of r and r' can guarantee the security of scheme against this attack in terms of the following theorem (see the proof in Appendix B)

Theorem 2 *Given a multi-tuple $(N, \varphi, \lambda, t_i, (\varphi^r)^{\lambda^{t_i}})$ over the RSA-type elliptic curve system S_N , where $r \in R_z$. It is intractable to compute $(t_j, (\varphi^r)^{\lambda^{t_j}})$ with $t_j < t_i$ for all PPT algorithms under the RSA assumption.*

Proof of the theorem is given at the end.

7.2 Security for KS-CDA attack

Security for KS-CDA Attacks In addition to collusion attack, chosen derivation-key attack (CDA) is a more easy-to-implement approach to break our TA scheme, in which the adversary only needs to eavesdrop the channel via the proxy server. In this way, the adversary gain some knowledge from the stolen derivation keys, and attempt to forge a new private-key with the help of a known private-key which it get from the proxy server. Our scheme can prevent the CDA attack from two aspects:
1) the derivation key is formed from the users unique identity τ_k, so that other users cannot use this key according to Theorem 1, and 2)More randomization is added by adding a new random variant σ into the derivation key to wrap the original private key under the Diffie-Hellmen assumption. Hence, we prove that our scheme is KS-CHA secure under the Bilinear co-CDH assumption as follows:

Theorem 3 *Given a RSA-type elliptic curve system $S_N = (N = pq, G, GT, e(\cdot,\cdot))$ with order $n = sn'$, CBE cryptosystem over S_N is key secure against chosen derivation-key attacks (KS-CDA) under the Bilinear co-CDH assumption on G even if the secret s and n' is known.*

Proof of KS-CDA resistance

Assume that there exists a PPT algorithm A that can breaks this problem over S_N with the known s, n' . Given a Co-CDH problem $(G_1, G_1^x, G_2, G_2^y) \rightarrow G_2^{xy}$ in G, we can construct an efficient algorithm B to solve this Co-CDH problem according to the algorithm A as follows: (1) Setup: B follows the Setup algorithm to get the elements $(g, h, \xi, \lambda, \mu)$ and then sets $w = G_2, \varphi = G_2^{k_1} \in G_{n'}, \bar{\varphi} = G_2^{k_2} \in G_{n'}$, where α, β, k_1, k_2 are known to B, $s \mid k_1$, and $s \mid k_2$ and $z = log_{G_1} G_2$ is unknown to B. Therefore,
B sends $PK = (S_N, g, h, \zeta, w, \varphi, \bar{\varphi}, \lambda, \mu)$ to the adversary A and $H(\cdot)$ can be obtained by the random Oracle query of B.
(2) Learning: A chooses a range attribute A_t and query Delegate algorithm with the polynomial number of users $u_{k_1}, ..., u_{k_s}$ with any time interval $A_{t_i}[t_{k_i}, t_{k_i'}] \in L$.
For each query, B chooses two random τ_i, δ_i and $H_i \in G$ and sets $r_i = y$, and then computes
$\widetilde{D_{t_i}} = (g^{\tau_i}, H(A_t)^{r_i})^{\delta_i} = (g^{\tau_i} G_2^{xy})^{\delta_i} = H^{\delta_i}$,
$\widetilde{D'_{t_{k_i}}} = (v_{t_{k_i}} w)^{r_i \delta_i} = (G_2^{y k_1 \lambda^{t_{k_i}}} G_2^y)^{\delta_i} = (G_2^y)^{\delta_i (k_1 \lambda^{t_{k_i}} + 1)}$
$\widetilde{D'_{t_{k_i'}}} = (v_{t_{k_i'}} w)^{r_i \delta_i} = (G_2^{y k_2 \lambda^{Z - t_{k_i'}}} G_2^y)^{\delta_i} = (G_2^y)^{\delta_i (k_2 \lambda^{Z - t_{k_i}} + 1)}$
and sends these derivation keys $\widetilde{SK_{L_i}} = (\widetilde{D_{t_i}}, \widetilde{D'_{t_{k_i}}}, \widetilde{D'_{t_{k_i'}}})$ to the adversary A. Note that, $H(A_t) = G_2^x$ and SK_{L_i} is anonymous for B because τ_i is unknown.
(3) Challenge: B chooses two random τ , r and defines $r_i = r^*/z$ which is unknown by B. And then it computes $D^* = g^{(\alpha + \tau^*)/\beta}$,

Table 7.1: Comparison with bsw Scheme

Scheme	Ciphertext		
bsw's Scheme	$g^{(\alpha+\tau_k)/\beta}$	$g^{\tau_k}H(A_t)^r$	w^r
our scheme	$g^{(\alpha+\tau_k)/\beta}$	$g^{\tau_k}H(A_t)^r$	$(v_{t_a}.w)^r$

$D_t^* = g^{\tau *}H(A_t)^{r*} = g^{\tau *}(G_1^x)^{r*}$,
${D'_{t_i}}^* = {v_{t_i}}^{r^*} = G_2^{k_1\lambda^{t_i}r^*/z} = G_1^{k_1 r*\lambda^{t_i}}$,
$D'^*_{t_j} = {v_{t_j}}^{r^*} = G_2^{k_2\lambda^{Z-t_j}r^*/Z} = {G_1}^{k_2 r^*\lambda^{Z-t_j}}$, ${D_t}''^* = wr^* = {G_2}^{r*/z} = {G_1}^{r^*}$.
Hence, B sends $SK_L = (D^*, ({D_t}^*, D'^*_{t_a}, D'^*_{t_b}, {D_t}''^*)$ as a challenge private key to A, where $L = A_t[t_i, t_j]$.
(3) Response: If the output of algorithm A is (L_i, SK_{L_i}),
where $SK_{L_i} = (D^*, (D^*_{t_i}, D'^*_{t_{k_i}}, D'^*_{t'_{k_i}}, {D_t}''^*))$ and $A_{t_i}[t_{k_i}, t_{k'_i}] \in L_i$, B checks whether the equations $D'^*_{t_{k_i}} = {G_2}^{yk_1\lambda t_{k_i}}$,
$D'^*_{t_{k'_i}} = {G_2}^{yk_1\lambda t_{Z-k'_i}}$,
${D_t}''^* = G_2^y$ and $e(G_1, D^*_{t_i}/g^{r^*}) = e(G_1^x, G_2^y)$ hold. If not, B repeats step (1), Else, B computes $G_2^x y = D^*_{t_i}/g^{^*}$ and returns it as output. The output of algorithm B is valid because the input of A satisfies $D^*_{t_i} = g^{r^*}G_2^{xy}$.
This means that the algorithm B is a PPT algorithm to solve Co-CDH problem only if A is also a PPT algorithm. But we know that the Co-CDH problem is hard for any PPT algorithms, hence this contradicts the hypothesis.

7.3 Security against SS-CDA attack

When a service provider tries to reveal the encrypted contents, it can explore potential security issues of our scheme. First, we consider the ciphertext-only attack. We will present our TA scheme is as strong as the bsws scheme. In order to demonstrate that the cloud service providers cannot compromise the ciphertext without private keys, we compare the difference between the ciphertext of our scheme and that of bsw scheme in Table 1.

It is easy to find that the different between them is merely the value $v_{t_a}^r$ which is introduced into ciphertexts. In fact, our scheme is compatible with bsws CP-ABE scheme for string-based matching. Hence, our scheme can be considered as an extension of bsws scheme in this point. Thus, our scheme remains the same security properties as of bsws scheme, i.e., semantically secure against chosen plaintext attack (IND-CPA).This means that the cloud service providers cannot obtains the contents of ciphertexts without the knowledge of private keys. Next, we analyze whether the derivation keys $\widetilde{SK}_{\mathcal{L}'}$ observed by the adversary (or proxy) increase the adversarys advantage against our scheme. Although $\widetilde{SK}_{\mathcal{L}'}$ are delegated from the private key $\widetilde{SK}_{\mathcal{L}}$, it cannot be used to decrypted the ciphertexts because

1) they contain only part of information of the private-keys, and
2) the random number δ is used to avoid revealing the decryption information to the adversary. In order to verify the validity of this method, we prove that any (polynomial) number of derivation keys observed by the adversary cannot increased the advantage of attacks under the Bilinear co-CDH assumption. This theorem is described as follows :

Theorem 4 *Given a RSA-type elliptic curve system $S_N = (N = pq, G, GT, e(.,.))$ with order $n = sn'$, TA cryptosystem over S_N is semantically secure against chosen derivationkey attacks (SS-CDA) under the Bilinear co-CDH assumption on G even if the secret s and n' is known.*

7.3.1 Proof of SS-CDA resistance

Proof. Assume that there exists a PPT algorithm A that can breaks this problem over S_N with the known s,n'. Given a Bilinear Co-CDH problem $(G_1, G_1^x, G_2, G_2^y) \to G_2^{xy}$, we can construct an efficient algorithm B to solve this Co-CDH problem according to the algorithm A as follows:
(1) Setup: B chooses a random integer θ and defines $\alpha = xy, \beta = \theta/z$, where$z = log_{G_1} G_2$ is unknown. B chooses the random integers $\lambda, \mu, k_1, k_2$to computes $g = G_1^{n'}$, $h = w^y = G_1^{\theta}, w = G_2, \zeta = e(G_1^x, G_2^y) = e(G_1, G_2)^{xy}, \varphi = G_2^{k_1}$, and $\bar{\varphi} = G_2^{k_2}$, where $s|k_1$ and $s|k_2$. So that B generates $PK = (S_N, g, h, \zeta, w, \varphi, \bar{\varphi}, \lambda, \mu)$ and sends it to A. $H(\cdot)$ can be obtained by the random Oracle query of B.
(2) Learning: A can send the polynomial number of delegate queries with any time interval $L_i = A_{t_i}[t_{k_i}, t_{k_i'}]$. For each query, B chooses the random τ_i, δ, r_i and computes
$\widetilde{D_{t_i}} = (g^{\tau_i}, H(A_{t_i})^{r_i})^{\delta} = {G_1}^{\delta\tau_i} {G_2}^{\delta k_i r_i}$,
$\widetilde{D'_{t_{k_i}}} = ((v_{t_{k_j}} w)^{r_i}.w^{r_i})^{\delta} = {G_2}^{\delta r_i (k_1 \lambda^{k_i} + 1)}$,
$\widetilde{D'_{t_{k_i'}}} = ((\bar{v}_{t_{k_i}} w)^{r_i}.w^{r_i})^{\delta} = {G_2}^{\delta r_i (k_1 \lambda^{(Z-k_i)} + 1)}$,
where $H(A_{t_i}) = G_2^{k_i}$ and k_i is random integer. Finally, B returns
$SK''_{L_i} = \widetilde{D_{t_i}}, \widetilde{D'_{t_{k_i}}}, \widetilde{D'_{t_{k_i'}}} A_{t_i}[t_{k_i}, t_{k_i'}] \in L_i$ to A.

(3) Challenge: B sets s=y and chooses a random a and ${G_2}^b = {G_2}^y / {G_2}^a$, where $ws = {G_2}^y$ and s=a+b. Such that, B computes $h^s = ({G_1}^y)^{\theta}$, and $\bar{E_{t_i}} = (\bar{v_{t_i}}.w)^a = ({G_2}^a)^{(} k_2 \mu^{Z - t_i} + 1)$,
$E'_{t_i} = H(A_t)^a = (G_2^a)^{k_i}$,
$E_{t_j} = (v_{t_j}.w)^b = ({G_2}^b)^{(} {k_1}^{\lambda_{t_j}} + 1)$
${E'}_{t_j} = H(A_t)^b = (G_2^b)^{k_i}$.
B outputs $\mathbb{H}_P^* = (\tau, h^s((\bar{E}_{t_i}, {E'}_{t_i}), E_{t_j}, {E'}_{t_j})_{A_t[t_i, t_j] \in \tau})$ as the challenge ciphertext to A.
(4) Response: A outputs a session key ek to B, and B also output it as result. If the output of algorithm A is valid, B is also valid because $ek = e(g^{\alpha}, w^s) = e({G_1}^{xy}, G_2) = e(G_1, G_2)$. This means that the algorithm B is a PPT algorithm to solve Co-CDH

problem only if A is also a PPT algorithm. But it is well-known that the Co-CDH problem is hard for any PPT algorithms, hence this contradicts the hypothesis

7.4 Security against revoked user

Suppose one user u_k has a range attribute $\mathrm{A}[t_i, t_j]$ but it has been revoked in t_p where $t_i < t_p < t_j$. Now suppose he want to get access to the encrypted content and hence he need the derivation key. Meanwhile the revoked list PXK will be updated and he will be in revoked list.
Now when proxy want to form

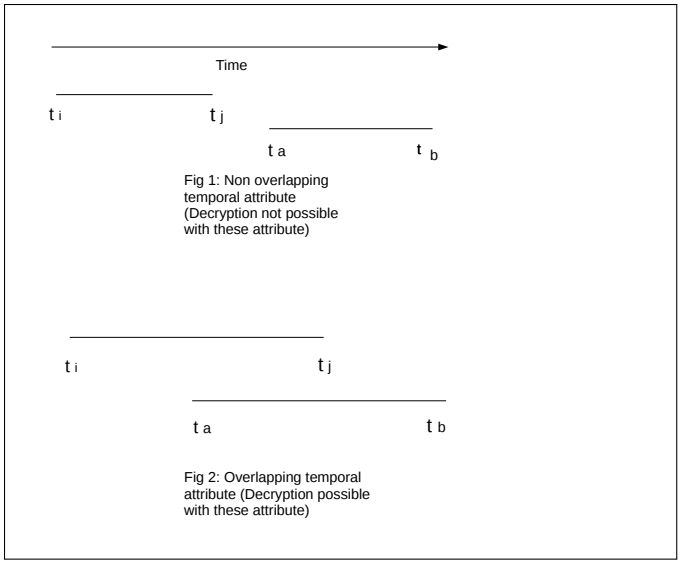
$$\lambda_i = \frac{u_k}{u_k - u_i} \prod_{j \neq i} \frac{u_j}{u_j - u_i}, \forall i, j \in \{1, \cdots, c\},$$

$$k \notin \{1, \cdots, c\}$$

for revoked user u_k he won't be able to form any of the λ_i because of the missing part $\frac{u_k}{u_k - u_i}$ and so he cannot calculate $\mathrm{P}(0)$. So derivation key formation will not be possible.

7.5 Security against attack by the user with non overlapping time duration with ciphertext

Suppose a user is having the attribute $A_t[t_a, t_b]$ and the ciphertext policy is having the range attribute $A_t[t_i, t_j]$ and the range of these two is non-overlapping . That means either $t_a > t_j$ or $t_b < t_i$ then it won't be able to calculate the some of these four values:-
$(D'_{t_j}, \bar{D}'_{t_i}, R'_{t_j}, \bar{R}'_{t_i})$ according to RSA assumption (Theorem 2).



7.6 Security against derivation key attack by collusion

We depend on the confidentiality of $\tau_k + \delta$ and $r_k\widetilde{P}(0) + \delta$ to guarantee the security of scheme against collusion privilege attacks. For sake of clarity, we only consider the collusion attacks by two adversaries to analyze all possible cases. For example, two users, u_a and u_b , intend to transfer the u_as a range attribute

$\widetilde{D}_t = D_t \cdot (gH(A_t))^{\delta_1} = g^{\tau_a+\delta_1} H(A_t)^{r_1\widetilde{P}(0)+\delta_1}$

$\widetilde{D}'_{t_{j_1}} = D'_{t_{j_1}} \cdot (v_{t_{j_1}} w)^{\delta_1} = (v_{t_{j_1}} w)^{r_1+\delta_1}$

$\widetilde{\bar{D}'}_{t_{i_1}} = \bar{D}'_{t_{i_1}} \cdot (\bar{v}_{t_{i_1}} w)^{\delta_1} = (\bar{v}_{t_{i_1}} w)^{r_1+\delta_1}$

$\widetilde{DB}'_t = D'_{t_{j_1}} \cdot w^{\delta_1} = w^{r_1+\delta_1}$

into u_b's range attribute :-

$\widetilde{D}_t = D_t \cdot (gH(A_t))^{\delta_2} = g^{\tau_b+\delta_2} H(A_t)^{r_2\widetilde{P}(0)+\delta_2}$

$\widetilde{D}'_{t_{j_2}} = D'_{t_{j_2}} \cdot (v_{t_{j_2}} w)^{\delta_2} = (v_{t_{j_2}} w)^{r_2+\delta_2}$

$\widetilde{\bar{D}'}_{t_{i_2}} = \bar{D}'_{t_{i_2}} \cdot (\bar{v}_{t_{i_2}} w)^{\delta_2} = (\bar{v}_{t_{i_2}} w)^{r_2+\delta_2}$

$\widetilde{DB}'_t = D'_{t_{j_2}} \cdot w^{\delta_2} = w^{r_2+\delta_2}$

7.6.1 Exchanging τ_a

If τ_a and τ_b can be exchanged then attackers will be able to get the key for extended temporal attribute ranges.but both τ_a and $\delta_1 \in Z$ so $\tau_a + \delta_1 \in Z$ is a random number and also $r_k\widetilde{P}(0) + \delta$ is a random number. So the problem reduced to the problem of

CPA-1 attack the proof against which is shown in theorem-1

7.6.2 Exchanging range attribute

Another possibility is user try to replace range attribute of $(\widetilde{D}'_{t_{j_1}}, \widetilde{\bar{D}'}_{t_{i_1}})$ by $(\widetilde{D}'_{t_{j_2}}, \widetilde{\bar{D}'}_{t_{i_2}})$ where $t_{i_2} < t_{i_1} < t_{j_1} < t_{j_2}$.
Note that as δ_1 and r_1 are random so $\delta_1 + r_1$ is random also $\delta_1 + r_1 \in Z$ is also random and similerly for Another user . So the problem reduced to Theorem 2 of CPA-2 attack.
Note that even if both user able to collude derivation key it still need to collude the private key as well to access the encrypted content fully.

7.6.3 Security against derivation key attack by 2 revoked user's collusion

Suppose 2 revoked user u_a and u_b having the range attribute (t_{a_i}, t_{a_j}) and (t_{b_i}, t_{b_j}) try to collude at time t_c where the revocation happens for both user before t_c where $t_c < t_{a_j}$ and $t_c < t_{b_j}$
The parts of the secret key they need to collude $(R_{t_{a_j}}, \bar{R}_{t_{a_i}}, RB_{t_a})$ and $(R_{t_{b_j}}, \bar{R}_{t_{b_i}}, RB_{t_b})$. But in order to get these attribute values both need to form P(0). And in order to get P(0) user need to form λ_i where $i \in Revokedlist$.
where

$$\lambda_i = \frac{u_k}{u_k - u_i} \prod_{j \neq i} \frac{u_j}{u_j - u_i}, \forall i, j \in \{1, \cdots, c\},$$

$$k \notin \{1, \cdots, c\}$$

But for 2 of the users u_k will be in the revoked list so both won't be able to form any of the λ_i for the missing part $\frac{u_k}{u_k - u_i}$. So none will be able to form $(R_{t_{a_j}}, \bar{R}_{t_{a_i}}, RB_{t_a})$ and $(R_{t_{b_j}}, \bar{R}_{t_{b_i}}, RB_{t_b})$ so collusion is not possibile.

7.6.4 Security against derivation key attack by one revoked user and one non-revoked user collusion

Suppose one revoked user u_a and one non revoked user u_b having the range attribute (t_{a_i}, t_{a_j}) and (t_{b_i}, t_{b_j}) try to collude at time t_c where revocation happens at time t_r where $t_{b_i} < t_{a_i} < t_r < t_{b_j} < t_c < t_{a_j}$.
Note that the CPA-2 attack is possibile because of the above inequality.
Now collusion needed for $(R_{t_{a_j}}, \bar{R}_{t_{a_i}}, RB_{t_a})$ and $(R_{t_{b_j}}, \bar{R}_{t_{b_i}}, RB_{t_b})$ as well .

But user u_b won't be able to calculate $(R_{t_{b_j}}, \bar{R}_{t_{b_i}}, RB_{t_b})$ as $t_{b_j} < t_c$ so collusion is not possibile.

7.7 Security against derivation key attack by the previous derivation key

Suppose a user u_k is having a valid derivation key d_k and at some point of time revocation happen for u_k. Now after revocation he can't be able to form derivation key.
Now if he send the previous derivation key d_k to the DecryptProxy phase the proxy will first check if that user is currently in the RL by searching in hash table RL. And it will find the u_k in RL so it will stop decrypting.

7.8 Proofs of CPA attack

Proof of CPA-1 attack resistance Proof. First, let $g^{\tau_k} = w\xi, H(A_t) = w^k, v^{t_a} = w^{k_1}$ and $\bar{v_{t_b}} = w^{k_2}$ in G, so we use the same generator w to denote SK_L as $D_t = g_k^{\tau} H(A_t)^r = w^{\xi+kr}$, $D_{t_a} = v_{t_a}^r = w^{k_1 r}, \bar{D_{t_b}} = \bar{v_{t_b}}^r = w^{k_2 r}$, and $D_t = w^r \in G$. Such that, we convert the theorem into the problem: it is intractable to extract the values (W^{ξ}, w^{kr}) from $(w, w^r, w^k, w^{k_1}, w^{k_2}, w^{k_1 r}, w^{k_2 r}, w^{\xi+kr})$. It is obvious that two unknown k_1, k_2 have no concern with this problem, such that the above problem is reduced into $(w, w^r, w^k, w^{\xi+kr}) \rightarrow (w^{\xi}, w^k r)$.
Assume that there exists a PPT algorithm A that can breaks this problem. Given a Co-CDH problem $(G_1, G_1^x, G_2, G_2^y) \rightarrow G_2^{xy}$, we can construct an efficient algorithm B to solve this CoCDH problem according to the algorithm A as follows:
(1) B invokes the algorithm A on input $(w = G_1, w^r = G_1^x, w^k = G_2^y, w^{\xi+kr} = G_2^z)$, where z is a random integer;
(2) If the output of algorithm A is (R_1, R_2), B checks whether two equations $R_1.R_2 = G_2^z$ and $e(G_1, R_2) = e(G_1^x, G_2^y)$ hold. If not, B repeats step (1);
(3) B computes $G_2^{xy} = R^2$ and returns it as output. The output of algorithm B is valid because the input of A satisfy- fies $r = x, w^{kr} = (G_2^y)^r = G_2^{xy} = R_2, e(G_1, R_2) = e(G_1, G_2^{xy}) = e(G_1^x, G_2^y)$ and $G_2^z = w^{\xi+kr} = R_1.R_2$
This means that the algorithm B is a PPT algorithm to solve Co-CDH problem only if A is also a PPT algorithm. But it is well- known that the Co-CDH problem is hard for any PPT algorithms, hence this contradicts the hypothesis.

Proof of CPA-2 attack resistance Seeking a contradiction, we assume that there exists a PPT algorithm A that can get a $(t_j, (\varphi^r)^{\lambda^{t_j}})$ under above input $(N, \varphi, \lambda, t_i, (\varphi^r)^{\lambda^{t_i}})$, where $t_j < t_i$. We can use the algorithm A to construct a PPT algorithm B that

can break the RSA problem over elliptic curve: given the public-key (G, N, e) and a ciphertext C to compute the plaintext $M = C^{e^{-1}}$. The algorithm B is described as follows:
(1) Given a RSA problem (G, N, e), B invokes the algorithm A on input $(N, \varphi, \lambda = e, t_i, C)$, where t_i is randomly chosen in integer set and $\varphi = C^{r'}$ is a random element in G.
(2)If the algorithm A returns a solution (t_j, R), B first checks if $R^{\lambda^{t_i - t_j}} = C$ and $t_i - t_j - 1 >= 0$. If not B repeats step (1);
(3) B computes $M = R^{e^{t_i - t_j - 1}} \in G$ in terms of $R^{\lambda^{t_i - t_j}} = C = M^{\lambda}$, and return the ciphertext M.
In the algorithm B, we cannot know the secret $r = 1/\lambda^{t_i} r'(\text{mod n'})$ for unknown n' (because of the actual difficulty of factoring large number N = pq), even though $\varphi = C^{r'}$ and r' is known. This means that the algorithm B is a PPT algorithm to solve RSA problem only if A is also a PPT algorithm. But it is well-known that the RSA problem is hard for any PPT algorithms, hence this contradicts the hypothesis.

7.9 Security against the online offline encryption scheme using shamir's secret sharing

Theorem 4 The online/offline CP-ABE scheme is selectively CPA-secure with respect to Definition 2.2 in [21] under the assumption that the security scheme CPABE of [6] is a selectively CPA-secure CP-ABE system.

Proof. To prove the theorem, we will show that any PPT attacker A with a non-negligible advantage in the OO-ABKEM-Exp experiment against the above scheme, which we will denote $\prod_{OO}$ = (Setup, Extract, OfflineEncrypt, OnlineEncrypt, Decrypt), can be used to break the selective CPA-security of the CP-ABE security scheme of [21], which we will denote $\prod_{CP} = (Setup_{CP}, Extract_{CP}, Encrypt_{CP}, Decrypt_{CP})$, with a PPT simulator B. The simulator plays the role of the challenger and interacts with A in OO-ABKEM-Exp with security parameter k and the universe of attributes set to $U = Z_p$.

Initialization Initially, B receives an access structure τ from A and gives it to the CP challenger.

Setup Next, B receives the public parameters $PK = (S_N, g, h, \zeta, w, \varphi, \bar{\varphi}, \lambda, \mu)$ from the CP challenger and passes them to A unchanged.

Phase 1 The secret keys are the same in both schemes, so any key generation request from A is passed to the CP challenger to obtain the key. The condition here

is that the requesting key should not satisfy the access structure τ.

Challenge B chooses two distinct, random messages m_0, m_1 in the CP message space and sends them to its CP challenger, and receives back a challenge ciphertext $CT_{RW} = (\tau, C = m * e(g^{\alpha}, w)^{s}, (EB'_t = w^{\rho_{B_t}} D\bar{B}' = H(B_t)^{\rho_{B_t}})$ for all $A_t \in U$). where m is either m_0 or m_1.
Here we will show for non-temporal attribute B_t and the game will be same for temporal attribute A_{t_y} and A_{t_x} then selects random blinding values $\rho'_B \in Zp$ such that $\rho'_B + \rho_B = q_{B_t}(0)$ and computes the ciphertext as (τ, CT_{CP}) followed by :-
$CT_{OO} = (CT_{CP}, \rho'_B$ for all attribute $B)$
To see why this is a correctly formed ciphertext, one must plug these values back into the decryption equation, worked out in steps for the correctness section, and see that the blinding values all cancel out. Next, B guess which message was encrypted $\tau_b \in 0, 1$ and computes $key_{guess} := C/m_{\tau_b}$. Finally, B then sends to A the tuple (key_{guess}, CT_{OO}).

Phase 2 B proceeds as in Phase 1. One added restriction is that it cannot issue a decryption query on the challenge ciphertext CT_{OO}

Guess Eventually, A outputs a bit τ_a .
If $\tau_a = 0$ (meaning that A guesses that key guess is the key encapsulated by CT_{OO}), then B outputs τ_b .
If $\tau_a = 1$(meaning that A guesses that key guess is a random key), then B outputs1τ_b . The distribution for A is perfect. Thus, if A has advantage in the OO-ABKEM-Exp experiment, then B breaks the CP CP-ABE system with the same probability.

Chapter 8

Future Work and Conclusion

Although our proposed scheme provides an efficient way to manage two-phase encryption , multi-authority key generation with user revocation and temporal constraints, we believe there is still scope for improvement in terms of the number of computations performed in the steps of each algorithm. Also we are using here composite ordered pairing which is costly compared to prime order pairing. So in future we will try to replace composite order pairing with prime order pairing. . We also intend to provide a single formal security proof for our scheme to demonstrate that it is secure.

In conclusion, it has been a challenging project and we are proud of the outcome of this project. Our objective was to design an all-encompassing novel scheme and we are happy to say that we have succeeded through our endeavors. As the idea and construction is novel, this thesis is likely to become a publication. As my first original work, this thesis will always be cherished by me throughout my life.

Bibliography

[1] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In IEEE Symposium on Security and Privacy, pages 321-334, 2007.

[2] V goyal, O Panday, A Sahai, B Waters.Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data,Theory of Cryptography Conference, page 515-534, 2006.

[3] M. Naor and B. Pinkas. Efficient trace and revoke schemes. International Conference on Financial Cryptography,1–20, 2001.

[4] D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. In Advances in Cryptology (CRYPTO'2001), volume 2139 of LNCS, pages 213-229, 2001.

[5] S.D. Galbraith and J.F. McKee. Pairings on elliptic curves over finite commutative rings. In 10th IMA International Conference of Cryptography and Coding, Cirencester, UK, December 19-21, 2005, Proceedings, pages 392-409, 2005.

[6] Y. Zhu, H. Hu, G. Ahn, M. Yu, H. Zhao. Comparison-Based Encryption for Fine-grained Access Control in Clouds. Proceedings of the second ACM conference on Data and Application Security and Privacy,page 105–116, 2012.

[7] S. Jahid, P. Mittal, N. Borisov. EASiER: Encryption-based Access Control in Social Networks with Efficient Revocation. In Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security (ASIACCS'11), ACM,pages 411-415, 2011. .

[8] A. Shamir. How to share a secret. ommunications of the ACM, 22(11), pages 612-613, 1979.

[9] K. Yang, Z. Liu, Z. Cao, X. Jia, D. Wong, K. Ren. TAAC: Temporal Attribute-based Access Control for Multi-Authority Cloud Storage Systems, IACR Cryptology ePrint Archive,volume 2012, pages 651, year 2012.

[10] S. Yu, C. Wang, K. Ren, W. Lou. Attribute Based Data Sharing with Attribute Revocation. In Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security (ASIACCS10), ACM, 2010, pages 261-270.

[11] A. Sahai and B.Waters. Fuzzy identity-based encryption. In EUROCRYPT, pages 457-473, 2005.

[12] B. Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In Proceedings of the 4th International Conference on Practice and Theory in Public Key Cryptography (PKC11), Springer, pages 53-70,year 2011.

[13] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS06), ACM, pages 89-98, 2006.

[14] A. Beimel. Secure schemes for secret sharing and key distribution. DSc dissertation, 1996.

[15] Y. Zhu, H. Hu, G. Ahn, D. Huang, S. Wang. Towards Temporal Access Control in Cloud Computing. In Proceedings of annual IEEE international conference on computer communications (INFOCOM 2012), pages 2576-2580, year 2012.

[16] S. Yu, C. Wang, K. Ren, and W. Lou. Achieving secure, scalable, and fine-grained data access control in cloud computing. In INFOCOM, 2010 Proceedings IEEE, pages 19. Ieee, year 2010.

[17] J. Li, X. Chen, J. Li, C. Jia, J. Ma, and W. Lou.Fine-grained access control system based on outsourced attribute based encryption. In Computer Security - ESORICS 2013 - 18th European Symposium on Research in Computer Security, Egham, UK, September 9-13, 2013. Proceedings, pages 592609, year 2013.

[18] S. Hohenberger and B. Waters, Attribute-based encryption with fast decryption. In Public-Key Cryptography - PKC 2013 - 16th International Conference on Practice and Theory in Public-Key Cryptography, Nara, Japan, February 26 - March 1, 2013. Proceedings, pages 162179, 2013.

[19] T. Jung, X. Li, Z. Wan, and M. Wan. Privacy preserv- ing cloud data access with multi-authorities. In Proceedings of the IEEE INFOCOM 2013, Turin, Italy, April 14-19, 2013, pages 26252633, 2013.

[20] M. Chase. Multi-authority attribute based encryption,Theory of Cryptography Conference, page 515-534, year 2007.

[21] M. Green, S. Hohenberger, B. Waters. Outsourcing the decryption of ABE ciphertexts. In Proceedings of the 20th USENIX Security Symposium. USENIX Association, volume 2011. number 3, year 2011.

[22] K. Yang, X. Jia, K. Ren. DAC-MACS. Effective Data Access Control for Multi-Authority Cloud Storage Systems. In Cryptology ePrint Archive, Report 2012/419, year 2012.

[23] S. Zhang and P. Chen and J. Wang. Online/Offline Attribute Based Signature. In Broadband and Wireless Computing, Communication and Applications (BWCCA), pages 566–571, year 2014 Ninth International Conference.

[24] N. Balani and S. Ruj. Temporal access control with user revocation for cloud data, IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications.pages 336–343, year 2014.