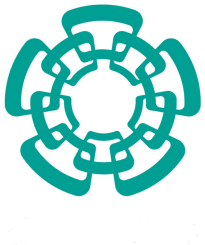


Center for Research and Advanced Studies

Cinvestav Tamaulipas

An Attribute-Based Encryption Scheme for Storage, Sharing and Retrieval of Digital Documents in the Cloud

Thesis by:

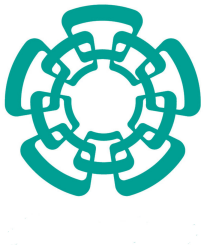
Melissa Brigitthe Hinojosa Cabello

as the fulfillment of the
requirement for the degree of:

Master of Science in Engineering and Computer Technologies

Thesis Director:
Miguel Morales Sandoval, Ph.D.

Cd. Victoria, Tamaulipas, Mexico. October, 2020



Centro de Investigación y de Estudios Avanzados del IPN

Unidad Tamaulipas

Esquema de Almacenamiento, Compartición y Recuperación de Documentos en la Nube Mediante Cifrado Basado en Atributos

Tesis que presenta:

Melissa Brigitthe Hinojosa Cabello

Para obtener el grado de:

Maestro en Ciencias en Ingeniería y Tecnologías Computacionales

Director de la Tesis:
Dr. Miguel Morales Sandoval

Cd. Victoria, Tamaulipas, México. Octubre, 2020

This research thesis was developed within the scope of the project 281565 from the "*SEP-CONACYT Research Fund for Education*", directed by Miguel Morales Sandoval.

The thesis of Melissa Brigitthe Hinojosa Cabello is approved by:

Jose Luis Gonzalez Compean, Ph.D.

Jose Juan Garcia Hernandez, Ph.D.

Miguel Morales Sandoval, Ph.D., Committee Chair

Cd. Victoria, Tamaulipas, Mexico, October 1st., 2020

This work is dedicated to M. Guadalupe Cabello Espinosa, my mom, who always gives me her unconditional support, love and confidence.

Acknowledgements

- Thanks mom for always being there with me and for your valuable advice that has allowed me to become better, not only as a person, but also as a professional. Thank you very much for guiding and supporting me through the complex journey the completion of this master's degree meant. For your patience and comprehension at all times, and for showing me lovingly even my mistakes, my infinite gratitude.

- Thanks to my thesis director, Dr. Miguel Morales, for his guidance on the development of this project and for his dedication and patience in sharing his knowledge with me. I also thank to Dr. Heidy Marin for her continuous tracking and support on the development of each stage of this project since its beginning as a bachelor's degree thesis project. Thank you both for your trust and for giving me the opportunity to collaborate with you. Finally, thanks for always being aware of my learning progress and for encourage me to continue my academic training.

- Thanks to my thesis reviewers, Dr. Jose Luis Gonzalez and Dr. Jose Juan Garcia, for their valuable and enriching feedback that contributed to improve the presentation and explanation of the concepts related to this project as well as the results accomplished by its development.

- Thanks to my friends from college Evelyn Limon and Victor Vazquez, for those countless pleasant and funny moments; Maribel Marin, for her advice and for encouraging me to overcome any difficulty and to move on when I felt overwhelmed or frustrated; and Homero Huerta, for continuously demonstrating his confidence by encouraging me to successfully complete this challenge.

- Thanks to all the staff of Cinvestav Tamaulipas; the professors, who contributed with their valuable lessons to expand my skills and expertise, thus improving my academic and professional background; and the administration personnel who became my friends since the IETAM project and helped me whenever I needed it. I have to specially thank to Dr. Miguel Morales and Dr. Javier Rubio for suggesting me to keep two options to continue my graduate studies, maybe at this moment I would not be here.

- Thanks to all my friends and classmates at Cinvestav for all the adventures lived, for sharing moments of happiness, sleepless nights and difficult moments. For all your support, feedback and advice, thank you all very much.

Contents

List of Figures

List of Tables

List of Algorithms

Publications

Melissa Brigitthe Hinojosa-Cabello and Miguel Morales-Sandoval. Esquema de Cifrado de Datos con Capacidades de Busqueda. Proceedings of the *Encuentro Estatal de Estudiantes Destacados en Tecnologias de Informacion (TopTamaulipas 2019)*. Cd. Victoria, Tamps., Mexico. Pages 11-14. ISBN: 78-607-9023-62-1.

Miguel Morales-Sandoval, Melissa B. Hinojosa-Cabello, Heidy M. Marin-Castro and Jose Luis Gonzalez-Compean. Attribute-based encryption approach for storage, sharing and retrieval of encrypted data in the cloud. *IEEE Access Journal*. DOI: 10.1109/ACCESS.2020.3023893. (Accepted)

Resumen

Esquema de Almacenamiento, Compartición y Recuperación de Documentos en la Nube Mediante Cifrado Basado en Atributos

por

Melissa Brigitthe Hinojosa Cabello
Unidad Tamaulipas
Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, 2020
Dr. Miguel Morales Sandoval, Director

En la actualidad, el cómputo en la nube facilita el acceso a una gran variedad de recursos bajo demanda. Uno de los servicios de tecnologías de la información con mayor demanda es el almacenamiento en la nube. Su auge se debe principalmente a que día a día los usuarios de cualquier tipo de servicio generan una gran cantidad de datos, los cuales requieren almacenarse y/o respaldarse. Si bien el uso de servicios de almacenamiento en la nube trae consigo grandes ventajas, también existen importantes retos, tales como la confidencialidad de los datos y el control de acceso hacia éstos. Aunque la confidencialidad se puede alcanzar mediante el cifrado, dicho proceso dificulta la búsqueda de información, ya que el proveedor de servicio no puede aplicar algoritmos de búsqueda tradicionales sobre colecciones de datos cifrados. Asimismo, al cifrar los datos es necesario que el propietario imponga y maneje las restricciones de acceso a su información. Para resolver el problema de las búsquedas sobre datos cifrados ha surgido la técnica criptográfica Searchable Encryption (SE). Aun cuando ésta cuenta con tres enfoques de solución, no todos son completamente viables para ser implementados en entornos reales de almacenamiento y compartición de datos en la nube. Es por ello que en este trabajo de investigación se propone definir y construir un esquema de cifrado de datos, bajo los enfoques ABE y ABSE, que permita preservar las capacidades de búsqueda y recuperación de información, al mismo tiempo que se garantiza la confidencialidad de los datos y se provee un mecanismo eficaz para la gestión del control de acceso hacia los mismos.

Abstract

An Attribute-Based Encryption Scheme for Storage, Sharing and Retrieval of Digital Documents in the Cloud

by

Melissa Brigitthe Hinojosa Cabello
Cinvestav Tamaulipas
Center for Research and Advanced Studies, 2020
Miguel Morales Sandoval, Ph.D., Advisor

Nowadays, cloud computing eases the access to a wide variety of resources on demand. One of the IT services in greatest demand is cloud storage. Its importance is mainly the result of the vast amount of data that users of any type of service produce every day, which needs to be stored or backed up. Even though cloud storage services offer great advantages, there are also important challenges, such as information confidentiality and data access control. Although confidentiality can be achieved by means of encryption, it makes more difficult for the service provider to perform searching operations because traditional search algorithms cannot be applied over encrypted data collections. Moreover, when data is encrypted, it is necessary for the owners to impose and manage restrictions for the access to their information. To solve the problem of searching on encrypted data, it has emerged the cryptographic technique Searchable Encryption (SE). While there are three SE solution approaches, not all of them are completely suitable for its deployment in real-world cloud storage and data sharing scenarios. Therefore, in this research project we propose the definition and construction of a data encryption scheme, through ABE and ABSE approaches, which preserves searching and information retrieval capabilities, while ensuring data confidentiality and providing an effective mechanism for its access control management.

Nomenclature

ABE	Attribute-Based Encryption
ABSE	Attribute-Based Searchable Encryption
AES	Advanced Encryption Standard
BDHP	Bilinear Diffie-Hellman Problem
CP-ABE	Ciphertext-Policy Attribute-Based Encryption
CSP	Cloud Service Provider
DET	Digital Envelope Technique
DHP	Diffie-Hellman Problem
DLP	Discrete Logarithm Problem
DO	Data Owner
DU	Data User
ECC	Elliptic Curve Cryptography
ECDLP	Elliptic Curve Discrete Logarithm Problem
FABECS	Fully Attribute-Based Encryption Scheme for Secure Cloud Storage, Sharing and Retrieval
HBC	Honest but Curious
IBE	Identity-Based Encryption
PBC	Pairing-Based Cryptography
PKC	Public-Key Cryptography
SE	Searchable Encryption
SHBC	Semi-Honest but Curious

1 Introduction

This chapter describes the motivation and preliminary background of the problem addressed in this thesis. It also presents the research questions to be answered and the hypothesis to be proven in this research project, as well as the thesis objectives. Finally, the proposed methodology for the project development and its main contributions are described.

1.1 Preliminaries

Cloud computing is the term referred to the provision of any service through the Internet. In this sense, the distribution of computer resources and services is done on demand, adjusted to the customers' needs and made available in a wide variety of settings [16, 38]. Cloud computing involves a lot of devices that can be geographically distributed in different locations but are connected to each other through one or more Internet service providers. Nowadays, one of the IT services in high demand is cloud storage [11].

Cloud storage is a service model in which a cloud service provider stores and manages data on

remote servers. Since the access to such data is carried out over the Internet, it is possible to manage the information from anywhere, at any time, and practically through any digital device. In addition to these benefits, cloud storage provides reliability and availability, because of the multilocated service providers' resources in order to guarantee redundant access, as well as infrastructure and management costs savings [11]. The simple cloud storage model includes three actors as shown in Figure 1.1:

1. **Data owner**, who outsources a document collection to a cloud service provider.
2. **Cloud storage service provider**, which is responsible for storing owners' data and make it available to authorized users.
3. **Data user**, who performs query operations over owners' data using the provider's searching capabilities.

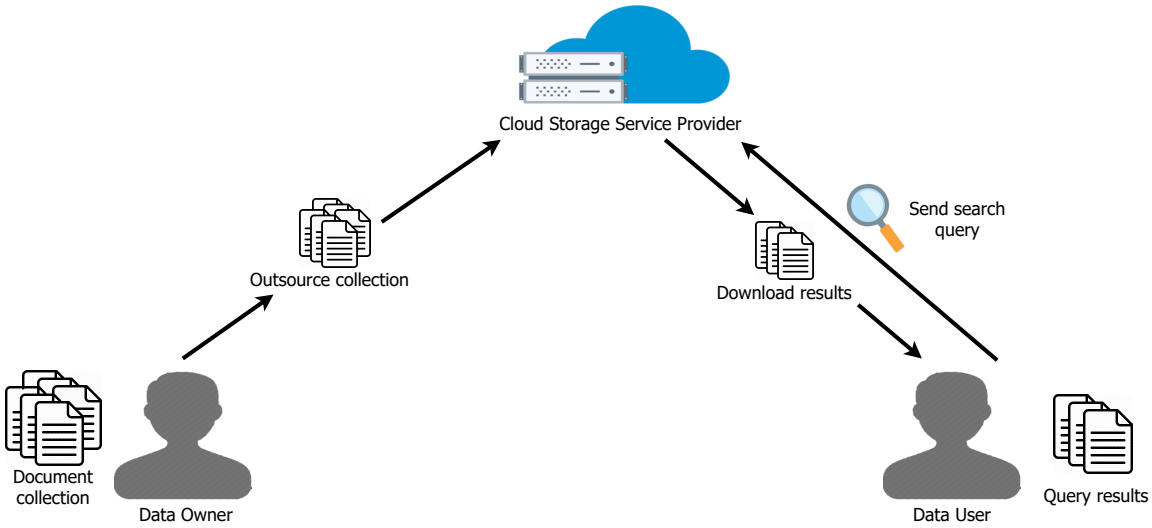
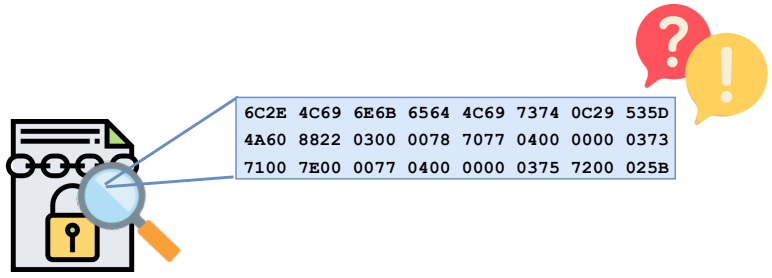


Figure 1.1: Simple cloud storage system model.

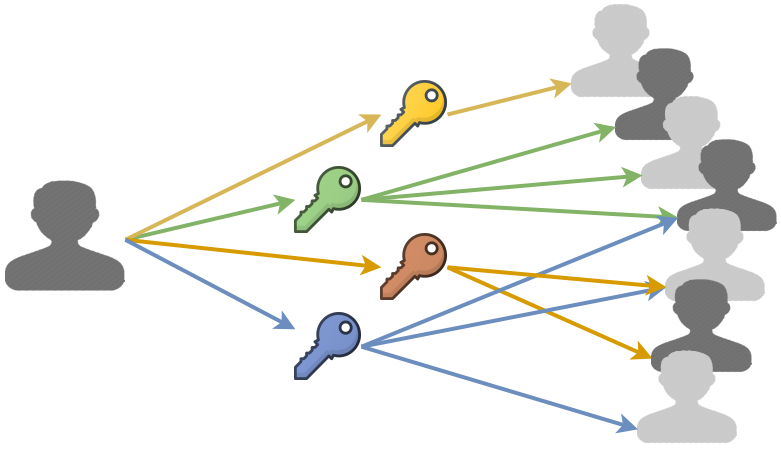
Both data owners and data consumers can be either corporate or domestic users. An entity could have both roles of data owner and data user at a time [23]. Nonetheless, in case of sensitive information, one of the main concerns of data owners in the context of cloud storage is information confidentiality and access control. Since the service provider has full access to the owners' data, it could use such data without their consent. An alternative solution for a data owner is to encrypt all

his information prior to uploading it to the cloud storage provider and to keep safe the cryptographic decryption keys. However, this alternative involves two main drawbacks as depicted in Figure 1.2:

- Searching and retrieval capabilities of the storage service provider cannot be exploited because data is encrypted and saved in an unintelligible form. The owner would have to download all his data, locally decrypt it and then to apply a search and retrieval algorithm.
- Data sharing with other users become a complex task, since it would be data owner's responsibility to implement a mechanism to efficiently distribute the cryptographic decryption keys.



(a) Once data is encrypted, there is no way to use traditional retrieval mechanisms to locate and download data of interest.



(b) Once data is encrypted, there should exist an efficient mechanism to securely share the decryption keys to end users.

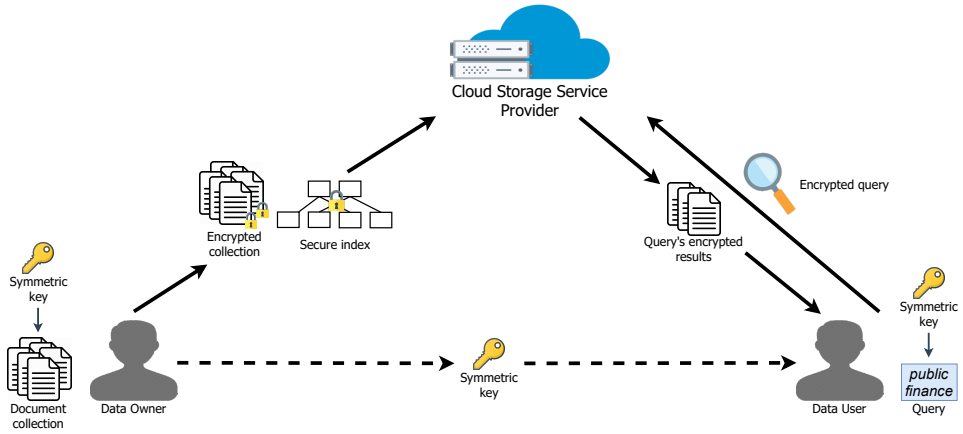
Figure 1.2: Main disadvantages of cloud data encryption.

Searchable Encryption (SE) is a suitable cryptographic technique to overcome the problem of searching over encrypted data. SE guarantees information confidentiality by allowing data owners to store encrypted data on the cloud while preserving the authorized data users' capability of searching and retrieving encrypted data [9]. So, an untrusted server can still performing searching operations over encrypted data in order to satisfy the users' information needs, but cannot learn about the intelligible form of encrypted data and the search criteria, as well as of the search and access patterns. In the literature, SE has been implemented from three main approaches: Searchable Symmetric Encryption (SSE), Public-Key Encryption with Keyword Search (PEKS) and Attribute-Based Searchable Encryption (ABSE).

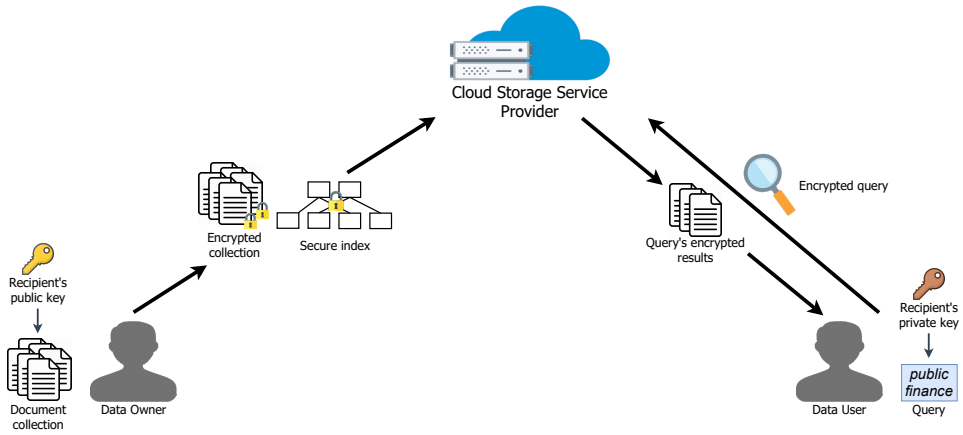
In such approaches the keywords extracted from the owners' document collection become the data to be encrypted and later queried. Although the document collection must be also encrypted before its outsourcing, none of these approaches considers the files encryption as an essential part of its solution. In fact, most of the existing proposals just assume that a symmetric cipher must be used to encrypt owners' files due to efficiency considerations. However, any of them provides a solution to the underlying problem of key distribution and management.

SSE follows the basic concept of private key cryptography in which a single key is used for both data encryption, decryption and, in the context of SE, the encrypted queries (trapdoors) generation [41]. In PEKS each data owner and data user possess a key pair $\{sk, PK\}$, where the recipient's PK is used to encrypt the keywords and the associated sk allows the trapdoors generation [6]. With ABSE, on the one hand, keywords are encrypted once using an access policy over a set of attributes and, on the other hand, the search trapdoors are created from data users attributes. In this way only those users who possess the right set of attributes can search and retrieve data of interest [1]. Figure 1.3 highlights the main differences among SE approaches by showing its respective operation models. Even though the concept of SE is quite simple and it has three solution approaches, not all of them are completely feasible to be implemented in real cloud storage and data sharing scenarios. Because of that, the construction of efficient schemes of this type still represents a complex task and

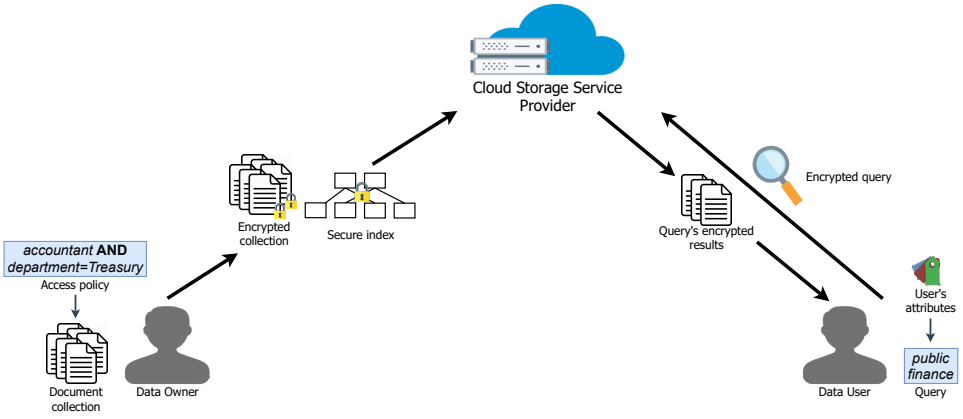
remains so far as an open problem.



(a) Searchable Symmetric Encryption



(b) Public-Key Encryption with Keyword Search



(c) Attribute-Based Searchable Encryption

Figure 1.3: Existing Searchable Encryption approaches.

1.2 Motivation

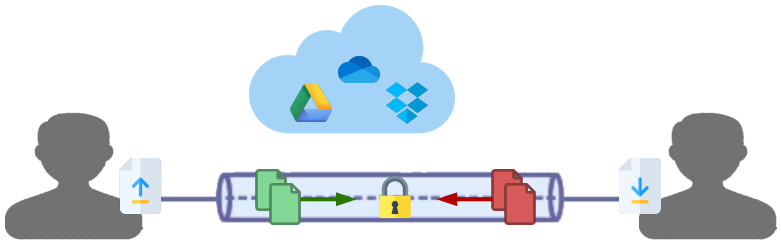
Technological advances and the constant increase in the amount of data created every day have made cloud storage services vital for users who need to store or back up their information. More often, both people and organizations choose to upload their information to external servers through cloud storage services. This is because cloud computing facilitates the access to several assets on demand with broad access to the provider network through standard mechanisms. Also, cloud storage services allow users to reduce the volume of data manipulated locally, and prevent the acquisition of expensive storage infrastructure.

Despite there are several storage models, the public cloud model has the greatest number of users. However, using public clouds can lead to certain security risks, especially when the information of data owners is sensitive [10], for example, clinical records, corporate financial documents, and personal data, just to name a few. As mentioned above, the use of cloud storage services provides important benefits, since the access to the resources is generally achieved through the Internet. However, it is also true that there are still pending challenges to be solved, especially those concerned with information security [9]. Among these challenges are data confidentiality and its secure transmission through insecure communication channels, as well as authentication and fine-grained access control for data sharing.

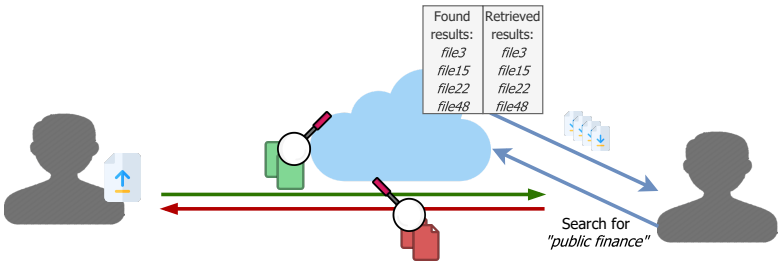
1.3 Research Problem

Information security is one of the primary concerns of users of any service provided through the cloud, as in the case of storage. The concern focuses on how to verify whether the information of an owner keeps confidential when placed within the service provider infrastructure. Although cloud service providers could be honest and do not disclosure users' information, it is also true that they can be curious, and eventually, could learn or derive information from the data they possess. In this

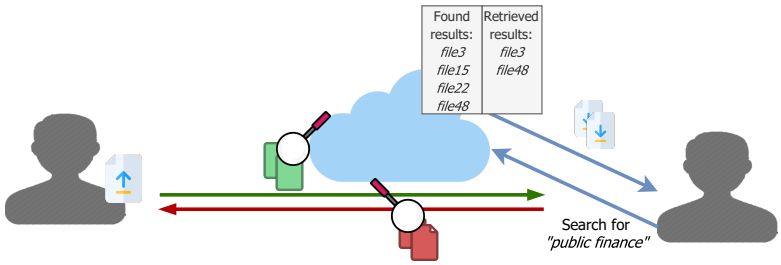
context, it could be said that the main adversary in the cloud storage service is the provider itself [25, 37].



(a) Trusted third party



(b) Honest but curious



(c) Semi-honest but curious

Figure 1.4: Storage service providers categorization.

According to [47], storage service providers could be categorized as *trusted third party* (TTP), *honest but curious* (HBC), and *semi-honest but curious* (SHBC); such classification is shown in Figure 1.4. In the former case, service users trust completely in the service provider based on the service level agreements (SLAs) and its brand reputation. In addition to storing and managing its customer files, the provider affords easy interaction between users and guarantees a secure communication among them. In the HBC model, the provider is considered honest because it stores the files of its customers and performs the expected functions according to the established protocol in the SLAs.

However, it tries to infer as much information as possible from the actions and interactions of users.

In the SHBC model, a semi-honest but curious provider behaves as in the HBC model but, additionally, it can execute only a fragment or all of the search operations and return to the user all or just a portion of the results it found in an honest way. In other words, an SHBC provider may or not follow the agreed protocol, but it may try to learn from the available information and its customers' interactions [37, 47]. While there are some proposed SE schemes in the literature as a solution to cloud storage under the HBC and SHBC scenarios, like the ones presented in [41], [6], and [44], those schemes have not been completely studied and evaluated in practical and real scenarios. Derived from the review of the state of the art and the challenges that still need to be faced, ABSE seems to be the most appropriate SE approach for cloud storage scenarios due to the following features:

1. It allows the implementation of cryptographic access control mechanisms, solving the problem of information distribution to data users.
2. It allows data encryption, which guarantees the confidentiality of owners' data.
3. It provides information retrieval mechanisms on encrypted collections, which makes possible for the server performing search tasks without compromising data confidentiality or the efficiency of the operations in the users' side.

1.3.1 Problem Statement

Although ABSE theoretically meets the basic security requirements and guarantees the searching capabilities for the cloud storage service provider, in the literature there are no ABSE constructions that have been evaluated in real scenarios. That is, there are no proposals in the state of the art that have been evaluated to determine their feasibility in real applications in such a way that the functional and non-functional requirements are satisfied in the context of HBC and SHBC scenarios. In other words, a practical scheme for secure cloud storage must be able to provide an appropriate

solution to all the possible problems involved in its efficient construction and deployment in cloud scenarios. For example, among these problems are 1) an efficient distribution and management of the symmetric encryption keys, and 2) the null reliability on the service provider, since it could be either HBC or SHBC, while at the same time it must be preserved 3) the data confidentiality, and 4) the retrieval and sharing capabilities of the storage service. ABSE relies in Attribute-Based Encryption (ABE), which has many variants and settings to be deployed. There are several ABSE schemes in the literature, each one suitable for different applications or use cases, and considering different properties and added functionalities. So, a carefully selection of ABE-related algorithms is required.

Table 1.1: NIST SP800-57 Standard for the existing security levels.

Security Level	Validity Period	Key Size		
		AES	ECC	Extension Field
80-bit	Outdated	128-bit	160-bit	1024-bit
112-bit	2016 – 2030	128-bit	224-bit	2048-bit
128-bit	2030 – 2040	128-bit	256-bit	3072-bit
192-bit	>2030	192-bit	384-bit	7680-bit
256-bit	>2030	256-bit	512-bit	15360-bit

Moreover, a very important functional requirement for an ABSE scheme is the security level that it could offer. According to different international standards, such as the NIST SP800-57 report and the ECRYPT-CSA D5.4 project, the expected security levels are currently at least of 128-bits. Table 1.1 shows the key sizes recommended by the National Institute of Standards and Technology (NIST) regarding the security level that is expected to be guaranteed, particularly in the case of the AES symmetric cipher and the elliptic curve cryptography (ECC). It is also included the minimum size of an extension field and the approximate validity period for each security level. As it can be seen, AES covers three possible security levels using a 128-bit key, since it represents its minimum key size. On the other hand, the size an ECC key must have is twice the security level n that is intended to offer.

This is due because the best algorithm to solve the discrete logarithm problem on which ECC bases its security has a complexity $O(\sqrt{n})$ [15, 37].

However, according to the current state of art there are no ABSE schemes constructions that provides security levels greater than 80-bit, which represents an outdated security level. The construction of an ABSE scheme for security levels of 128-bit or more implies changing the algorithms definition to operate with asymmetric bilinear pairings, which are mathematical objects based on elliptic curve cryptography.

Another relevant non-functional requirement for an ABSE scheme is the efficiency, which is critical for its use in real applications. However, schemes based on bilinear pairings and elliptic curves are computationally so expensive. There is not an ABSE scheme that evaluates its efficiency in cloud storage scenarios. According to the current state of art of ABSE schemes, it is stated that:

i. There is no proposal using both ABE and ABSE to overcome the security challenges of cloud storage in both HBC and SHBC adversarial models.

ii. There are no ABSE schemes constructions that provide security levels greater than 128-bits.

iii. There are no ABSE constructions experimentally evaluated in cloud storage and data sharing scenarios.

1.3.2 Research Questions

In this project are addressed the following research questions.

Question 1:

Which are the requirements for cloud storage, data sharing, and information retrieval applications in an ABSE scheme?

Question 2:

Which is the algorithmic construction of an ABSE scheme to support security levels equal to 128-bit or higher?

Question 3:

Which are the more adequate algorithmic or deployment strategies that make feasible the ABSE scheme on real scenarios?

1.3.3 Hypothesis

It is feasible the construction of an ABSE scheme that works in the SHBC model for practical cloud storage and data sharing scenarios, in such a way that the minimal functional requirements are accomplished on this environments at the same time that non-functional requirements, like security levels and efficiency, are met.

1.4 Thesis Objectives

The general and specific objectives of this research project are defined hereafter.

1.4.1 Main Objective

To provide a security scheme fully constructed over Attribute-Based Encryption, well suited for secure cloud storage and data sharing scenarios.

1.4.2 Specific Objectives

- To identify the basic requirements of an attribute-based encryption security scheme for its use in cloud storage and data sharing environments.
- To define the main modules and architecture of an attribute-based encryption security scheme for cloud storage and data sharing applications, considering functional, cryptographic, information retrieval and other requirements.

- To provide an attribute-based encryption security scheme construction that allows its deployment on real cloud storage applications with security levels equal to 128-bit or higher.

1.5 Methodology

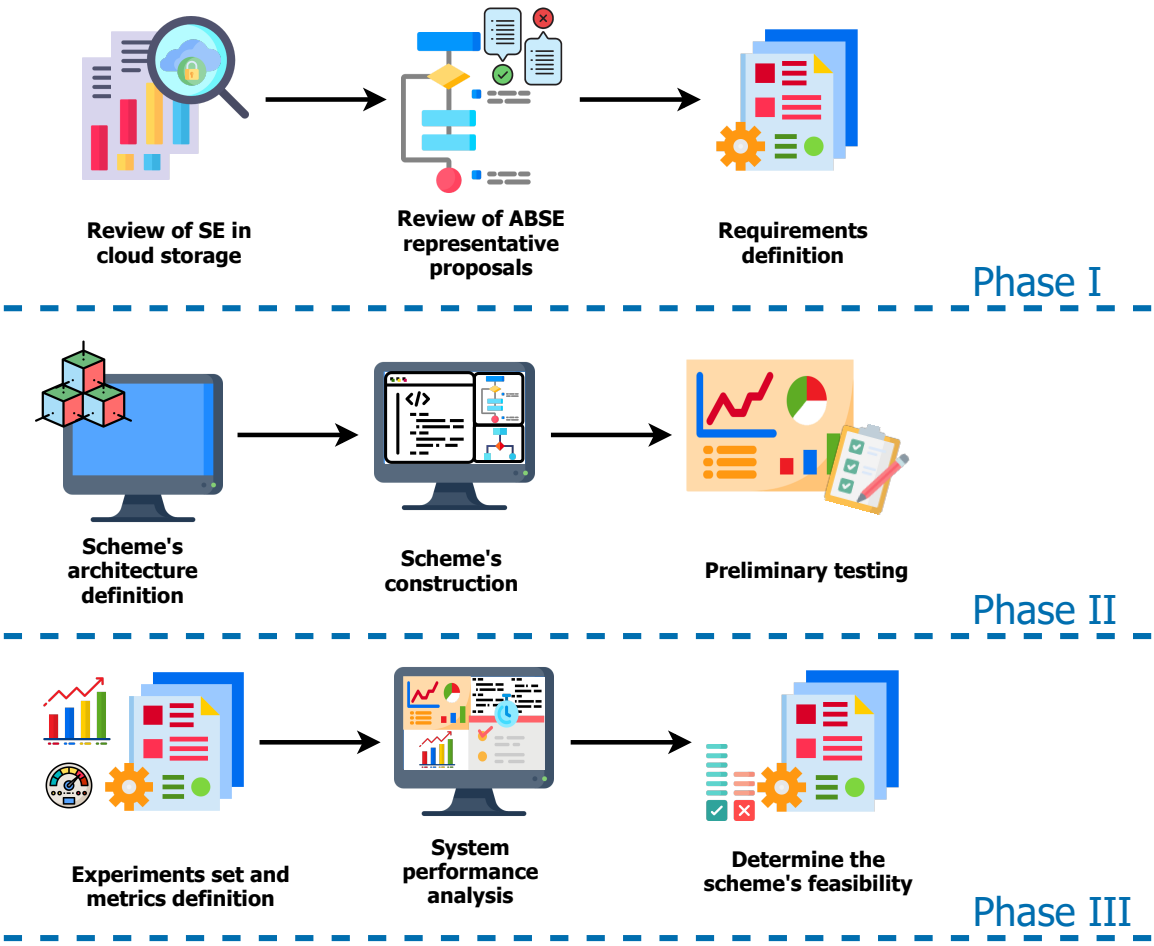


Figure 1.5: Proposed methodology for the development of the research project.

As depicted in Figure 1.5, the research project is divided into three phases, from which the objectives are progressively fulfilled. The phase 1 involves the literature review in order to identify the desirable requirements that must be met for secure cloud storage and data sharing scenarios. Phase 2 consists in the definition of the scheme's components and architecture, as well as its construction

and a preliminary evaluation of the components. Finally, phase 3 involves both the experiments definition and the performance evaluation of the designed scheme. This section specifies the tasks needed for the construction of the proposed ABSE scheme that lead to the achievement of the project's objectives.

1.5.1 Phase 1: ABSE scheme essential requirements definition

1. Perform a systematic review of Searchable Encryption literature and its applications in the context of cloud storage under the SHBC adversarial model.

2. Perform a systematic review of the literature on the most representative ABSE techniques, identify their scope, limitations, as well as major components and algorithms in the basic operation model and under the SHBC adversarial model.

3. Based on 1 and 2, determine the desirable functional requirements of an ABSE scheme for cloud storage and data sharing applications.

1.5.2 Phase 2: Proposed scheme's construction

1. Define the ABSE scheme's architecture based on the functional requirements identified in Phase 1.

2. Define the components of the scheme's construction, for example: indexing algorithms and structures, data encryption algorithm, cryptographic keys establishment, user attributes management, access control policy generator, and so on.

3. Perform the construction of cryptographic schemes in such a way that can be provided security levels greater than 128-bits, which is generally achieved through the use of asymmetric pairings, not yet available in the state of the art.

4. Perform white box and black box tests on the implemented components, as well as at integration level.

5. Evaluate the relevance of using algorithm acceleration strategies for a scheme's efficient deployment at components level, for example: parallel programming patterns, multithreading, and multitasking.

1.5.3 Phase 3: Design of scheme's proofs and performance evaluation

1. Deploy the ABSE scheme's system model defined in Phase 2 by building an experimental prototype.

2. Evaluate the relevance of using acceleration and prototype deployment strategies at the system level.

3. Define the experiments set, the corpus and metrics to be used for the evaluation of the performed ABSE construction.

4. Execute the scheme's performance evaluation in the different test cases established in 3.

5. Analyze the system performance obtained results during the experimental process.

6. Determine the scheme's feasibility in the context of cloud storage and data sharing scenarios by focusing on a particular case study, for example, organizational environments.

1.6 Contributions

In this research project we propose the definition of the basic requirements and the architecture of a searchable data encryption scheme and its construction with the aim of preserving the searching and information retrieval capabilities for cloud storage and data sharing scenarios. We call this scheme ***Fully Attribute-Based Encryption Scheme for Secure Cloud Storage, Sharing and***

Retrieval (FABECS) which is intended to provide confidentiality and access control over encrypted data outsourced in the cloud, as well as privacy and access control for searching over encrypted outsourced data. Therefore, the main contributions of this research project are listed below.

- A new security scheme that works in the semi-honest but curious adversarial model using asymmetric bilinear pairings.
- Novel ABSE algorithms for the asymmetric setting, allowing constructions for recommended security levels.
- Experimental evaluation of the scheme that demonstrates the feasibility of its use in real environments.
- Experimental prototype that allows the evaluation of new cryptographic and information retrieval techniques in the cloud storage environment.

1.7 Thesis outline

The rest of this thesis is organized as follows: Chapter 2 presents the background of the research project and its basic concepts. Chapter 3 gives an overview about cloud storage systems and the related work in the literature. The searchable data encryption scheme proposed in this project is described in Chapter 4. Chapter 5 presents a series of experiments and its results. Finally, in Chapter 6 is given the conclusion of this research project and the possible directions for future work.

2

Background

This chapter presents the theoretical framework and basic concepts that supports this research project. First, the fundamentals of Public-Key Cryptography and Pairing-Based Cryptography are described. Then, Identity-Based Encryption and Attribute-Based Encryption are defined in order to put into context Attribute-Based Searchable Encryption and how it guarantees data confidentiality while preserving retrieval capabilities from data in ciphertext form.

2.1 Groups

A group $(\mathbb{G}, \diamond)$ is a mathematical structure consisting of a finite set of elements $\mathbb{G}$, with order n (also denoted by $|\mathbb{G}|$), to which an algebraic group operation $\diamond$ is associated. The operation $\diamond$ is considered as a binary operation since it is defined between any two elements $a, b \in \mathbb{G}$ [14, 45]:

$$a \diamond b \to c,$$

where $c \in \mathbb{G}$. That is, if $a, b \in \mathbb{G}$, then the result of the binary operation defined over them will be also in the set $\mathbb{G}$. This is called the *closure* property, and it implies that $\mathbb{G}$ is closed under the binary operation. A group must also hold the following properties [45]:

- **Closure.** $\forall\ a, b \in \mathbb{G}$, then $a \diamond b \in \mathbb{G}$.
- **Associativity.** $\forall\ a, b, c \in \mathbb{G}$, then $a \diamond (b \diamond c) = (a \diamond b) \diamond c$.
- **Existence of identity element.** $\exists\ e \in \mathbb{G}$, such that $\forall\ a \in \mathbb{G} : a \diamond e = a$.
- **Existence of inverse element.** $\forall\ a \in \mathbb{G} : \exists\ a^{-1} \mid a \diamond a^{-1} = e$.

Within a group there is an element g, called *generator*, from which the rest of the elements in $\mathbb{G}$ can be obtained by means of the successive application of $\diamond$ over g (that is, for any $h \in \mathbb{G}, h = g \diamond g \diamond \cdots \diamond g$). In other words, all the elements of $\mathbb{G}$ can be obtained by computing g^k: applying $\diamond$ operation k times to g. The generator's order is the smallest integer n that generates the identity element when g is operated by $\diamond$: $g^n = e$. Considering the group $\mathbb{G} = \{e, h_1, h_2, ..., h_{p-1}\}$, it is said to be cyclic if the elements within the group start to be repeated when computing $g^{n+1} = h_1$, and so on. Then, if $\mathbb{G}$ is a cyclic group with generator g, it is denoted by $\mathbb{G} = \langle g \rangle$.

Table 2.1: Groups $\mathbb{Z}_p$ and $\mathbb{Z}_p^*$, and its properties.

Property	$\mathbb{Z}_p$	$\mathbb{Z}_p^*$
Closure	$(a + b) \in \mathbb{G}$	$(a * b) \in \mathbb{G}$
Associativity	$a + (b + c) = (a + b) + c$	$a * (b * c) = (a * b) * c$
Identity element	$a + 0 = a$	$a * 1 = a$
Inverse element	$a + a^{-1} = 0$	$a * a^{-1} = 1$

Examples of groups are the additive group $\mathbb{Z}_p = \{0, 1, 2, ..., p-1\}$, with operator $+$ with reduction modulo p, and the multiplicative group $\mathbb{Z}_p^* = \{1, 2, 3, ..., p-1\}$, with operator $*$ with

reduction modulo p , where p is a large prime. Both of them satisfy the properties described above, as shown in Table 2.1 [45].

2.2 Public-Key Cryptography

Public-Key Cryptography (PKC), or asymmetric encryption, was invented in the mid 1970's with Diffie-Hellman algorithm, and it is based on the Discrete Logarithm Problem (DLP). The DLP is defined over cyclic groups: given a group $\mathbb{G}$, with generator g, and a positive integer a, $g^a = b$ is easy to calculate, but given $\{g, b\}$ it is hard to find the value of a.

$$f(x) = y : g^a = b, \text{ linear complexity}$$

$$f'(y) = x : \{g, b\} = a, \text{ exponential complexity}$$

PKC's first application was the Diffie-Hellman protocol to allow a pair of entities the establishment of a pre-shared key that enables a secure message exchange between them over an insecure communication channel. PKC arised from the need to solve the problem of encryption keys distribution when data is stored and transmitted over a network, such as the Internet, as well as to preserve data privacy and integrity.

Asymmetric encryption is based on number theory, mainly in the use of prime numbers to generate a pair of keys mathematically related, but not identical, to encrypt and decrypt data, rather than one key that later has to be shared with all those authorized users through a secure communication channel, as carried out in symmetric encryption algorithms. In other words, the use of a key pair in PKC solves the key distribution problem of symmetric encryption schemes by allowing secure message exchange through insecure communication channels. This key pair consists in a public key k_{pub} and a private key k_{priv}. As the name suggests, k_{pub} is made publicly available to be used by anyone to encrypt data, and k_{priv} must be kept secret and only available for its owner to decrypt a ciphertext, as shown in Figure 2.1. Because the key pair is mathematically related, data encrypted with a k_{pub_a}

can only be decrypted with its corresponding k_{priv_a} [45].

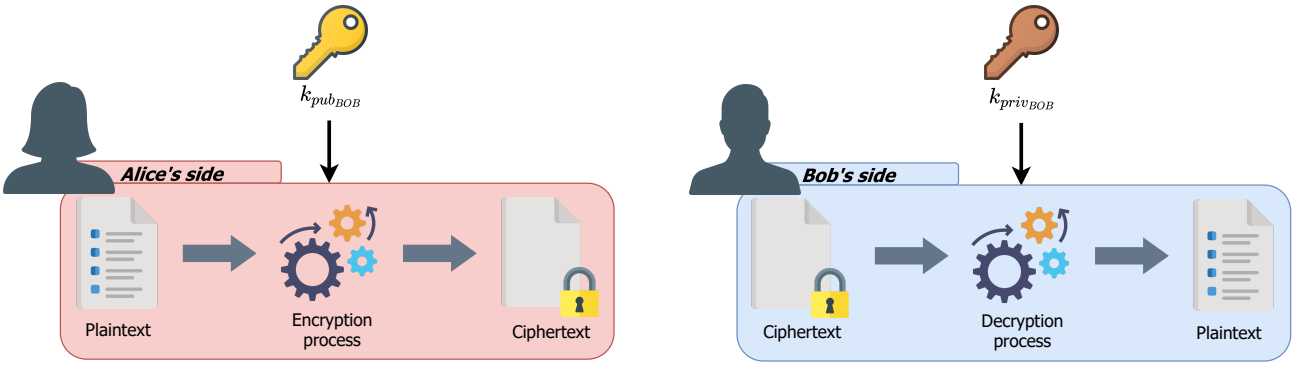


Figure 2.1: Asymmetric encryption.

2.3 Elliptic Curve Cryptography

An elliptic curve E is a group defined over a finite field $\mathbb{F}_q$ by a non-singular Weierstrass equation of the form:

$$y^2 = x^3 + Ax + B,$$

where $A, B \in \mathbb{F}_q$ are constants. In other words, an elliptic curve is essentially a set of points $E(\mathbb{F}_q) = (x, y) \in \mathbb{F}_q \times \mathbb{F}_q$, with order q, satisfying a simple equation that has distinct roots. In this way it is ensured that the curve is non-singular and the Elliptic Curve Discrete Logarithm Problem (ECDLP) for the set of points on E is hard to solve. Let $P \in E(\mathbb{F}_q)$ be a point of prime order n and $Q \in \langle P \rangle$, the ECDLP consists in finding an integer l that holds $Q = lP$ [29, 40]. In this case, E forms an additive group since the group operator is an addition of points $lP = P + P + \cdots + P$. This problem is difficult enough as it is an instance of DLP. But, in the ECDLP smaller fields can be chosen in comparison to those needed for cryptosystems based on DLP. Unlike the DLP defined for multiplicative groups, the best known algorithm to solve the ECDLP takes fully exponential time, more precisely $O(\sqrt{q})$. Also, there is an extra point on E called *point at infinity* ∞ which is the location of the identity element for the additive group. So E is the set

$E = \{(x, y) : y^2 = x^3 + Ax + B\} \cup \{\infty\}$ [40].

Elliptic Curve Cryptography (ECC) is a public-key cryptography method proposed by Victor Miller and Neal Koblitz in 1985. It is based on the set of points on an elliptic curve whose security relies on the hardness of the ECDLP [34]. ECC provides the same functionality as other PKC schemes, so, it can be used for data encryption, digital signatures or for key exchange. One of its main advantages is that it can reach a desired security level with relatively smaller key size than the ones needed, for example, by other PKC systems such as RSA [51]. Although scalar multiplication is the essential operation of any ECC scheme, there are other operations that can be performed over elliptic curves, such as point addition and point doubling. However, scalar multiplication is the most expensive operation since it involves the successive execution of point additions which further requires performing another operations, like inversion or squaring. Because of its mathematical properties, all of these operations produce a result point on the same curve [2].

2.4 Pairing-Based Cryptography

Let the additive groups $\mathbb{G}_1$, $\mathbb{G}_2$, with generators g_1 and g_2, respectively, and the multiplicative group $\mathbb{G}_T$ be cyclic groups of prime order r. Defined over cyclic groups, a bilinear paring e is a mapping $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. The pairing is *symmetric* when $\mathbb{G}_1 = \mathbb{G}_2$, and when $\mathbb{G}_1 \neq \mathbb{G}_2$ the pairing is said to be *asymmetric*. A bilinear paring e must fulfill the following properties:

- **Bilinearity:** $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab},\ \forall\ g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2, \{a, b\} \in \mathbb{Z}_r^*$
- **Non-degeneracy:** $e(g_1^a, g_2^b) \neq 1$
- **Computability:** There is an efficient algorithm to compute $e(g_1, g_2)$.

As stated by the bilinearity property, if $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$, then $e(g_1, g_2)$ is a generator of $\mathbb{G}_T$; as a consequence, the DLP in $\mathbb{G}_1$ can be efficiently reduced to the DLP in $\mathbb{G}_T$. Thus, a hard problem in one group ($\mathbb{G}_1$) is reduced to an easier problem in other group ($\mathbb{G}_T$). Let (P, Q) be an instance

of the DLP in $\mathbb{G}_1$, where $Q = xP$, then $e(P,Q) = e(P,xP) = e(P,P)^x$. So, $\log_P Q = \log_g h$, because $g = e(P,P)$ and $h = e(P,Q)^x$ are elements of $\mathbb{G}_T$. Moreover, non-degeneracy property entails that the mapping must not send the elements of $\mathbb{G}_1$ and $\mathbb{G}_2$ to the identity element of $\mathbb{G}_T$, i.e., the mapping cannot be the trivial map [29].

The Diffie-Hellman Problem (DHP) states that given a cyclic group $\mathbb{G}$ with generator g, if a and b are two values randomly chosen from $\mathbb{Z}_r^*$, it is easy to compute both g^a and g^b. Nonetheless, given g, g^a and g^b, the computation of g^{ab} is intractable. By its definition, DHP is closely related to DLP since g^{ab} could be efficiently solved by computing a by means of the discrete logarithm of g^a and then computing g^{ab} through the exponentiation $(g^b)^a$. Pairing-Based Cryptography (PBC) relies its security on the Bilinear Diffie-Hellman Problem (BDHP), considered as intractable and originally defined over a symmetric pairing. It consists in computing $e(g,g)^{abc}$ given g, g^a, g^b and g^c. Hardness of the BDHP implies that if the DHP in $\mathbb{G}_1$ can be efficiently solved, then an instance of the BDHP can be solved by computing $e(g,g)^{ab}$ and later $e(g^{ab},g)^c = e(g,g)^{abc}$; and if the DHP in $\mathbb{G}_T$ can be efficiently solved, then another instance of the BDHP can be solved by computing $g = e(g,g)$, $g^{ab} = e(g^a,g^b)$, $g^c = e(g,g^c)$ and then g^{abc} [3, 21].

2.5 Identity-Based Encryption

The Identity-Based Encryption (IBE) scheme was introduced in 1984 by Shamir, and is probably the most popular application of bilinear pairings. Even though, it was until 2001 when Boneh and Franklin proposed the first practical IBE scheme. The main advantage of this scheme is the use of a user identity, like an e-mail address, a phone number, etc., as the public key instead of some random generated number. Thus, as any string that uniquely identifies a person can be used, the need for looking up public keys on a directory or the use of certificates is eliminated. However, the corresponding user private key sk_u needs to be derived from the public key using another secret value. As depicted in Figure 2.2, users can obtain the private key corresponding to their identity

from a Key Generation Center (KGC) who possess a key pair $\{a, aP\}$, where the private key a is randomly selected from $a \in [1, \ldots, r-1]$ and aP is the public key. Once the authenticity of a user is proven, the KGC sends the corresponding private key to the user over a secure communication channel [28].

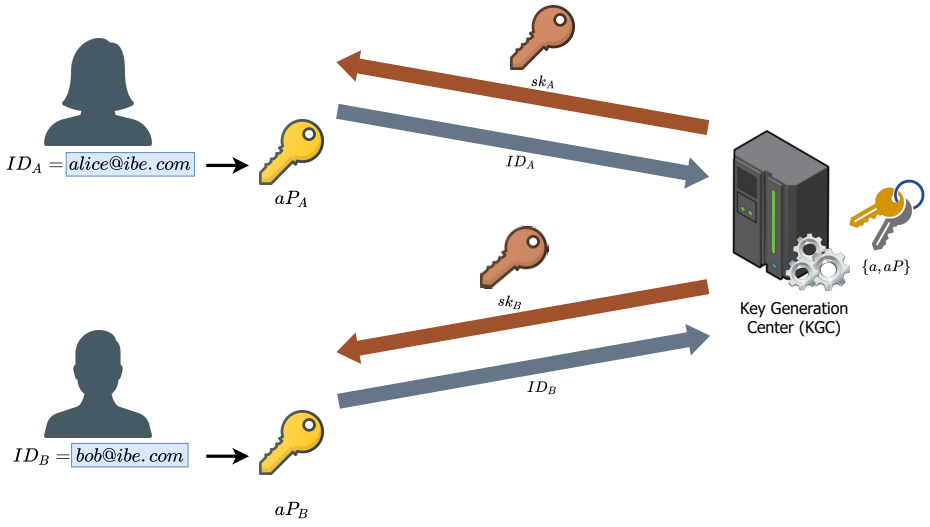


Figure 2.2: Users' secret keys generation on IBE scheme.

As said before, the IBE scheme uses bilinear pairings $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$, where $\mathbb{G}_1$ and $\mathbb{G}_T$ are cyclic groups and $\mathbb{G}_1$ has a generator P, for which the Bilinear Diffie-Hellman Problem (BDHP) is intractable. Also, two cryptographic hash functions are used:

$$H_1 : \{0,1\}^* \to \mathbb{G}_1 \setminus \{\infty\}$$

$$H_2 : \mathbb{G}_T \to \{0,1\}^l,$$

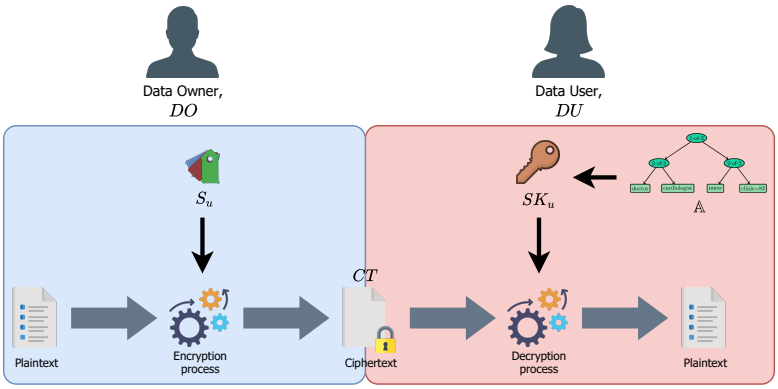
where H_1 is a hash function that maps a set of bits to a point in the elliptic curve, and H_2 is another hash function that maps a point in $\mathbb{G}_T$ to a plaintext of length l [29].

2.6 Attribute-Based Encryption

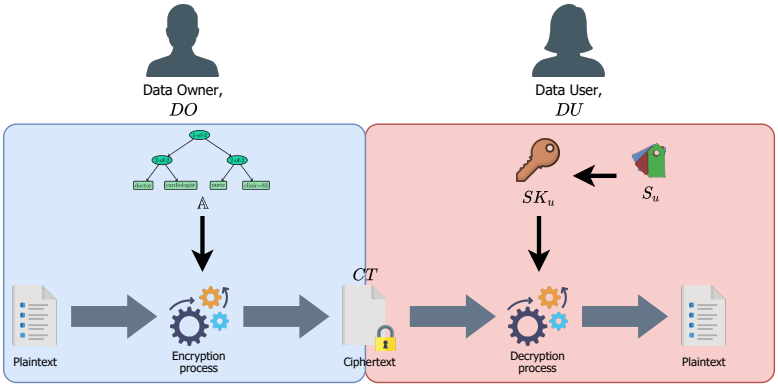
Attribute-Based Encryption (ABE) has its foundations as an access control mechanism that uses IBE, but where an entity could be identified by a set of descriptive attributes rather than a string that

changes for each specific user. In this sense, it is a public-key cryptography technique that allows secure data sharing with multiple users, while offers great flexibility of data access management [1, 39]. Instead of using traditional public or private keys, in ABE data is encrypted by the data owner's attributes specification that a potential user must possess to be able to decrypt a message using his secret key. Either the attributes specification or the user's secret key could be associated with an access policy. And it is specifically this access policy defined over a set of attributes the one that establishes the access control mechanisms for the encryptor's data [24].

It is worth mentioning that, as it is a many-to-many encryption scheme, the data owner does not have to know in advance all the potential decryptor users. This feature represents one of the main advantages of this scheme, since it enables a fine-grained access control without incurring in



(a) KP-ABE.



(b) CP-ABE.

Figure 2.3: Attribute-Based Encryption schemes.

the storage and communication burden associated with other PKC schemes [43]. In this way, and opposed to other cryptographic techniques, ABE is better suited for cloud storage and data sharing scenarios because encryptor's data remain confidential even on untrusted storage environments, such as with HBC or SHBC service providers.

ABE can be categorized in two fields: Ciphertext-Policy Attribute-Based Encryption (CP-ABE) and Key-Policy Attribute-Based Encryption (KP-ABE). As shown in Figure 2.3, its difference relies on how the access policy (associated to an access structure $\mathbb{A}$) is used to encrypt data or to generate the decryption key; and, as a consequence, how users' attributes are used in the encryption and decryption tasks [1]. In KP-ABE the ciphertext CT is based on a set of attributes S_u and the users' secret keys have embedded the access policy. In contrast, in CP-ABE the ciphertext is associated to certain access structure $\mathbb{A}$ and users' secret keys SK_u are based on attributes [24, 43].

2.7 Ciphertext-Policy Attribute-Based Encryption

Ciphertext-Policy Attribute-Based Encryption (CP-ABE) is a practical implementation of ABE proposed in [5] by John Bethencourt, Amit Sahai and Brent Waters. Like in KP-ABE, it uses an access policy closely related to an access structure and a set of attributes, but the way and location these are specified is the opposite. As stated in Section 2.6, in KP-ABE a user can decrypt a message if the access structure embedded in her secret key satisfies the attributes associated to the ciphertext. Nonetheless, in CP-ABE the set of attributes is used to describe a data user, and these are associated to the secret key SK_u, while the access structure $\mathbb{A}$ is embedded into the ciphertext CT. Thus, a data user can decrypt a message only if the attributes embedded in SK_u satisfy the access policy attached to the ciphertext [24].

Because of its definition, CP-ABE is considered most appropriate to implement access control mechanisms conceptually closer to Role-Based Access Control (RBAC), like the ones needed for data stored in the cloud. Not only data confidentiality and fine-grained access control are guaranteed,

but it also makes easier the users revocation and the system scalability. Users growth is efficiently handled since the complexity of the scheme's operations is given by the amount of attributes used in both $\mathbb{A}$ and SK_u, instead of the number of authorized system users [24, 43].

CP-ABE is made up of four main algorithms [5]: *Setup*, *Encrypt*, *KeyGeneration* and *Decrypt*. On the one hand, the setup algorithm takes as input a security parameter λ and generates a public key PK and a master key MK; the encryption algorithm takes as input PK, a message M and the access structure $\mathbb{A}$ to produce the ciphertext CT. On the other hand, the key generation algorithm takes as input MK and a set of attributes S_u, and outputs a private secret key SK_u; finally, the decryption algorithm takes as input PK, CT and a SK_u, and only if the attributes in SK_u satisfy $\mathbb{A}$ the ciphertext is decrypted returning the message M. An access control policy AP is represented by boolean expressions of attributes containing either logical gates (**AND** $\wedge$, **OR** $\vee$) or threshold gates (***k-of-n***). Access structures usually describe access control policies by means of access trees, where the root and the internal nodes represent logical gates and the leaves describe attributes.

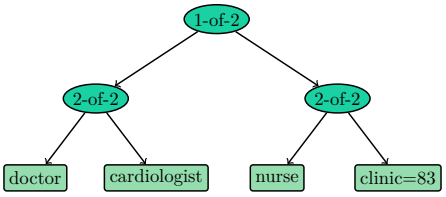
In CP-ABE, data is encrypted by means of an access policy which has associated a set of attributes, instead of using traditional public or private keys. Because access structures represent access control policies, an AP imposes restrictions on the access structure being used regarding having or not all the attributes specified on it. In other words, in order to access the plaintext of some encrypted data, the decryptor's attributes set in SK_u must satisfy the access structure embedded into the ciphertext, which is determined by the access control policy defined by the encryptor. As said before, an access control policy could be represented either by logical or threshold gates. Let's consider the following example: some clinical records were encrypted in such a way it can only be accessed either by a doctor who is specialized in Cardiology or by a nurse assigned to Clinic 83. This access restriction can be easily translated into a boolean expression with logical gates, as shown in Figure 2.4(a), which is also equal to the one shown in Figure 2.4(b) expressed with threshold gates in reverse Polish notation. Figure 2.4(c) shows the tree access structure $\mathbb{A}$ which represents both boolean expressions.

$AP =$ (doctor **AND** cardiologist) **OR** (nurse **AND** clinic=83)

(a) Access policy: logical gates.

$AP =$ doctor cardiologist **2-of-2** nurse clinic=83 **2-of-2 1-of-2**

(b) Access policy: threshold gates.



(c) Tree access structure.

Figure 2.4: Example of an access control policy and its access tree.

For tree-based access structures, it is employed the Shamir's Secret Sharing scheme (SSS) to distribute a secret s, needed to implement the CP-ABE encryption algorithm. The secret is divided into sub-secrets s' and distributed in $\mathbb{A}$ among the n attributes considered in the threshold gates of the access policy according to the polynomial $P(x)$ with degree $k-1$ given in Equation 2.1, where a_i represents random coefficients and s is the secret to be distributed:

$$P(x) = a_{k-1}x^{k-1} + a_{k-2}x^{k-2} + ... + a_1x^1 + s \tag{2.1}$$

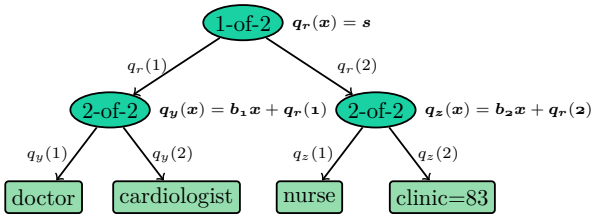
$$P(0) = s$$

Then, each sub-secret s' assigned to a leaf node $y \in \mathbb{A}$ has to be protected through an exponentiation as follows: given the additive groups $\mathbb{G}_0$, $\mathbb{G}_1$, the multiplicative group $\mathbb{G}_T$ and the bilinear pairing e, compute $[C_y, C'_y]^+$ as expressed in Equation 2.2, where g_0 is the generator of $\mathbb{G}_0$, $attr_y$ is the attribute corresponding to leaf node y, and $H(\cdot)$ is a hash function $H : \{0,1\}^* \rightarrow \mathbb{G}_1$ that maps a set of bits to a point in $\mathbb{G}_1$.

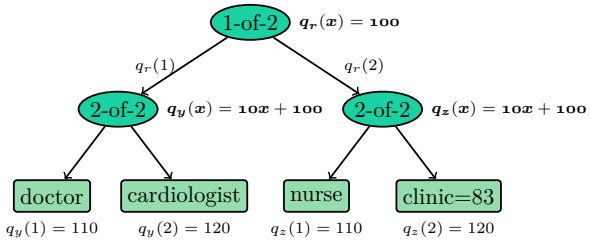
$$C_y = g_0^{s'} \tag{2.2}$$

$$C'_y = H(attr_y)^{s'}$$

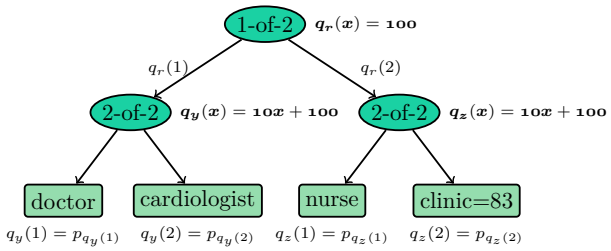
On the decryption process, $P(x)$ can be reconstructed by using at least k of n sub-secrets



(a) SSS diagram.

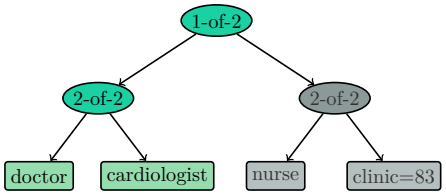


(b) Secret distribution.

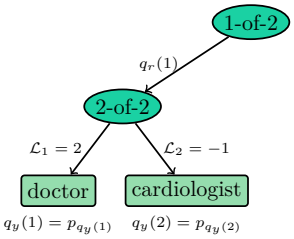


(c) Secret encryption.

1) Secret sharing process.



(d) Access structure for S_u.



(e) Pruned access structure.

2) Secret reconstruction process.

Figure 2.5: Example of secret sharing and reconstruction processes in CP-ABE.

previously distributed in $\mathbb{A}$. Once the polynomial has been reconstructed, s can be computed in a bottom-up process, i.e., starting from the leaf nodes to the root, by applying Equation 2.3, where $\mathcal{L}_j$ is a Lagrange interpolation coefficient and is computed according to Equation 2.4:

$$s = \sum_{j=1}^{k-1} y_j * \mathcal{L}_j, \tag{2.3}$$

$$\mathcal{L}_j = \prod_{i=1}^{k-1} \frac{-x_i}{x_j - x_i}, i \neq j \tag{2.4}$$

The secret of the root node is indirectly recovered through a pairing exponentiation $e(g_0, g_1)^{rs}$. Given a user secret key SK_u, an access structure $\mathbb{A}$, which could be simplified based on the attributes $S_u \in SK_u$ or not, and a ciphertext CT, $e(g_0, g_1)^{rs}$ is computed as denoted in Equation 2.5, where $[d_y, d'_y]^+$ are values associated to leaf y, taken from SK_u, $[C_y, C'_y]^+$ values are obtained from CT, $\mathcal{L}_y$ is the Lagrange's coefficient for leaf y, and r is a random number in $\mathbb{Z}_r$.

$$\prod_{y \in \mathbb{A}} \frac{e(d_y, C_y)^{\mathcal{L}_y s'}}{e(d'_y, C'_y)} \equiv e(g_0, g_1)^{rs} \tag{2.5}$$

Figure 2.5 shows the complete encryption and decryption processes described above. Let AP be the access policy previously considered in Figure 2.4 ("*doctor* ***AND*** *cardiologist* ***OR*** *nurse* ***AND*** *clinic=83*"), the SSS diagram of $\mathbb{A}$ is the one depicted in subfigure 2.5(a). Given the secret value $s = 100$, and the random coefficients $a_1 = 10$ and $a_2 = 10$, s is distributed over both branches of $\mathbb{A}$ through the polynomials $q_y(x) = 10x + 100$, and $q_z(x) = 10x + 100$ and then encrypted and transformed into the values $[p_{q_y}]^+$ and $[p_{q_z}]^+$, respectively, as follows:

$$q_y(1) = 10(1) + 100 = \boxed{110}$$
$$C_{q_y(1)} = g_0^{q_y(1)}$$
$$C'_{q_y(1)} = H(\textit{doctor})^{q_y(1)}$$
$$p_{q_y(1)} = \{C_{q_y(1)}, C'_{q_y(1)}\}$$

$$q_z(1) = 10(1) + 100 = \boxed{110}$$
$$C_{q_z(1)} = g_0^{q_z(1)}$$
$$C'_{q_z(1)} = H(\textit{nurse})^{q_z(1)}$$
$$p_{q_z(1)} = \{C_{q_z(1)}, C'_{q_z(1)}\}$$

$$q_y(2) = 10(2) + 100 = \boxed{120}$$
$$C_{q_y(2)} = g_0^{q_y(2)}$$
$$C'_{q_y(2)} = H(\textit{cardiologist})^{q_y(2)}$$
$$p_{q_y(2)} = \{C_{q_y(2)}, C'_{q_y(2)}\}$$

$$q_z(2) = 10(2) + 100 = \boxed{120}$$
$$C_{q_z(2)} = g_0^{q_z(2)}$$
$$C'_{q_z(2)} = H(\textit{clinic=83})^{q_z(2)}$$
$$p_{q_z(2)} = \{C_{q_z(2)}, C'_{q_z(2)}\}$$

Finally, considering the attributes set $S_u = \{doctor, cardiologist\}$, its access structure is depicted

in subfigure 2.5(d), which could be simplified or pruned in order to verify if S_u satisfies AP in a more efficient and faster way, as shown in subfigure 2.5(e). The secret reconstruction is done by calculating the Lagrange's coefficients for q_{y_1} and q_{y_2}, which means computing the pairing exponentiation $e(g_0, g_1)^{r \cdot q_r(1)}$ needed to decrypt and return a message M, where $q_r(1) = \mathcal{L}_1 \cdot q_y(1) + \mathcal{L}_2 \cdot q_y(2)$, i.e., the secret s:

$$\mathcal{L}_1 = \frac{-2}{1-2} = \frac{-2}{-1} = \boxed{2} \qquad \mathcal{L}_2 = \frac{-1}{2-1} = \frac{-1}{1} = \boxed{-1}$$

Even though the tree-based approach is widely used for describing access control policies, the matrix-based version proposed by Brent Waters in [52] allows an enhanced flexibility in the access policies definition. In other words, since any monotonic policy can be converted into a matrix, the matrix-based approach of CP-ABE enables the implementation of schemes that require the use of highly expressive access policies to enforce more natural access control mechanisms, i.e., closer to RBAC. Furthermore, by using the Linear Secret Sharing scheme (LSSS) proposed in [26] it guarantees reduced ciphertexts, making it more efficient and a better choice for practical implementations. Under such approach it is used a formatted Boolean formula (FBF) to represent the access control policy, and a $n \times l$ matrix $\mathtt{Mat}$ instead of an access tree. Here, n is the total number of attributes included in the access policy and l the minimum amount of attributes required to satisfy the access matrix; besides, $\rho(i)$ is used to associate the i-th row of $\mathtt{Mat}$ ($\mathtt{Mat}_i$) to attribute y in FBF.

On the encryption process, shares of a secret s are distributed across the different attributes specified in FBF according to the LSSS matrix. Those shares form a column vector $\mathbf{v} = \{s, s_1, s_2, ..., s_{n-1}\}$, where s is the secret to be distributed and s_j are values randomly chosen from $\mathbb{Z}_p^*$, which is multiplied by each row $\mathtt{Mat}_i$ in order to get the scalar value λ_i belonging to $\rho(i)$. Additionally, for each $\rho(i)$ it is chosen a random value $r_i \in \mathbb{Z}_p^*$ needed to compute $[C_{y_i}, C'_{y_i}]^+$ according to Equation 2.6. Given the groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$, the bilinear pairing e, and considering the attributes in FBF are mapped to $\mathbb{G}_1$: g_2 is the generator of $\mathbb{G}_2$, h is an element of the public key PK, and $H(\cdot)$ is the hash function $H : \{0,1\}^* \rightarrow \mathbb{G}_1$.

$$C_{y_i} = g_2^{r_i} \tag{2.6}$$
$$C'_{y_i} = h^{\lambda_i} \times H(\rho(i))^{-r_i}$$

Given a user secret key SK_u for an attributes set S_u and a ciphertext CT, if S_u satisfy the access matrix embedded in CT, let $I = \{i : \rho(i) \in S_u\}$, $[\lambda_i]^+$ be valid shares of a secret s, and $\omega_i \in \mathbb{Z}_p$ be the set of constants such that $s = \sum_{i \in I} \omega_i \lambda_i$. The secret s, needed to decrypt the message M, is recovered as part of the pairing exponentiation $e(g_1, g_2)^{\beta rs}$ computed by Equation 2.7, where $[C_{yi}, C'_{yi}]^+$ are taken from CT and $D', [d_{y_i}]^+ \in SK_u$.

$$e(g_1, g_2)^{\beta rs} = \prod_{i \in I} \left(e(C'_{yi}, D') e(d_{y_i}, C_{yi}) \right)^{\omega_i} \tag{2.7}$$

2.8 Searchable Encryption

Searchable Encryption (SE) is a cryptographic technique that aims to maintain data privacy while retaining users information retrieval functions. In other words, SE allows data owners to store encrypted data on untrusted external servers and ensure the possibility of performing search operations to those authorized users. In this way it is possible to take full advantage of cloud computing and avoid processing and communication overheads on the user's side, for example, by having to download all the available data and decrypt it in order to perform search operations over plaintext data. This approach has multiple applications, among which mainly highlight mail servers, storage systems and databases management [9].

Unlike the classic cloud storage model, the basic SE operation model considers three actors, but adds them new operations [9], as shown in Figure 2.6:

- Data owner, *DO*. Before outsourcing the documents collection D, *DO* extracts the keywords $W_j = \{w_1, w_2, w_3, ..., w_n\}$ from D and uses those keywords to construct a secure index SI. SI is built using a secret key k that is also used for the symmetric encryption of D. Then, *DO* sends both the secure index and the encrypted files CT_D to the cloud service provider.

- Data users, *DU*. In order to access data owners' files, a *DU* must generate special query tokens, known as trapdoors $T_u(w_q)$, for the keywords to search by using the key k.
- Cloud service provider, *CSP*. Stores encrypted files collections D and performs search operations on them using the trapdoors $T_u(w_q)$ sent by *DU*s. The result $D_{rslt} = Search(CT_D, T_u(w_q))$ is the set of encrypted files that are related to the trapdoor specified by *DU*.

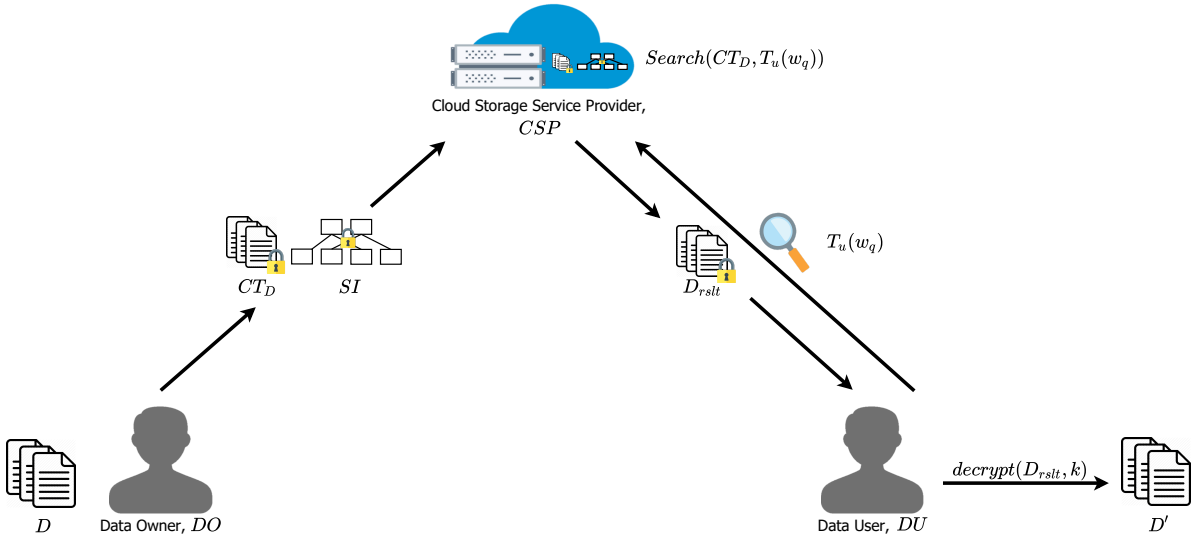


Figure 2.6: Basic Searchable Encryption operation model.

Through SE it is possible to ensure the security of files, keywords, and users search and access patterns. In this way, the risks of information leakage are minimized since the service provider cannot learn from users' queries or from the files that stores [9, 20]. Although the concept of SE is quite simple, the construction of efficient schemes of this type still represents a complex task and remains so far as an open problem.

2.9 Attribute-Based Searchable Encryption

Attribute-Based Searchable Encryption (ABSE) is a SE scheme based on ABE. ABSE allows the creation of a secure index for encrypted data. The secure index is created using an ABE access

control policy AP. Search over encrypted data is achieved by means of tokens created from data users' attributes. So, only those users who possess a set of attributes that cryptographically satisfy AP can have access to the data. Each data user has a unique key that is generated based on the attributes set that each one possess. That is, the data is encrypted under the assumption that the data user's identity is represented by an attribute set [1, 32]. The attributes, or attribute combination, that must be satisfied are defined by the data owner on an access policy, which determines which users meet the established criteria and the search permission could be granted. In this way it is possible to achieve secure authentication, authorization and secure transmission that guarantee information privacy on environments where it is necessary to ensure fine-grained access control among a large number of users. This is the case, for example, of cloud computing [24].

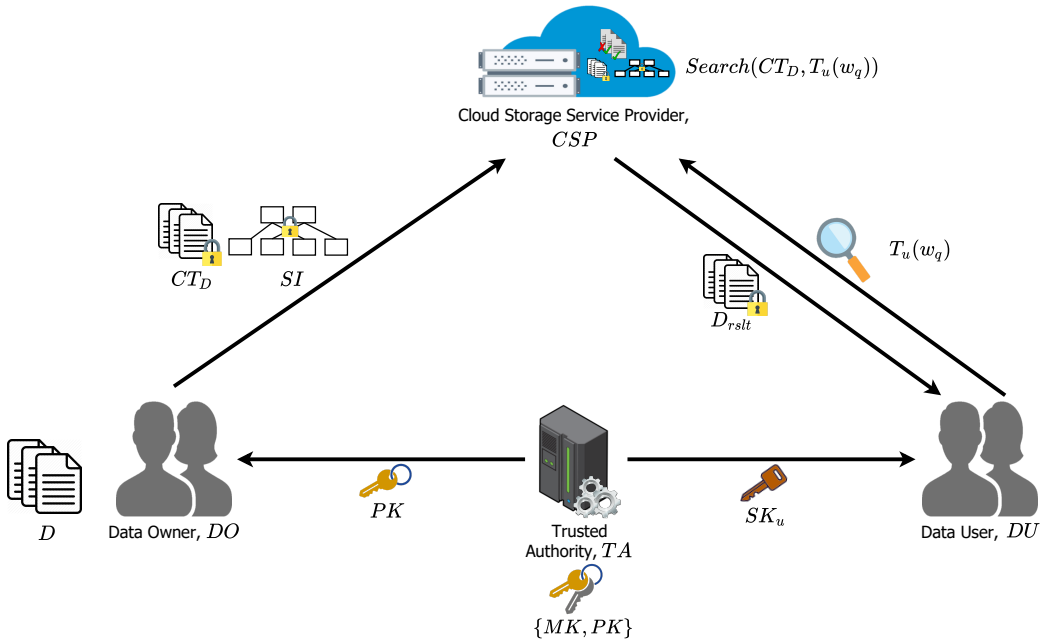


Figure 2.7: Attribute-Based Searchable Encryption operation model.

As it can be seen in Figure 2.7, in ABSE there is an additional actor called trusted authority *TA*, which is responsible for managing the system users' attributes. Given a security parameter λ, *TA* initializes the system generating and distributing a key pair $\{MK, PK\}$, where PK is public and

MK is private, as well as the necessary encryption keys for owners' data. The keys $\{MK, PK\}$ are usually correlated by pairing-based cryptography. On this type of SE scheme, the data owner encrypts either the index, generating a secure index SI, or the data files D, creating the ciphertext CT_D, by using PK and an access policy AP. As said before, each data user is provided with a user secret key SK_u, which is generated by the trusted authority using MK and the corresponding user's attribute set S_u. User's SK_u is necessary for the generation of a query trapdoor $T_u(w_q)$, in order to let the storage service provider to perform search operations on the owners' encrypted data $Search(CT_D, T_u(w_q))$, and then return the matching results D_{rslt} to the data consumer [24, 44].

2.10 Chapter Summary

This chapter presented the background and the basic concepts related to this research project. Sections 2.1 and 2.2 presented the foundations of Public-Key Cryptography (PKC) in order to later introduce Elliptic Curve Cryptography (ECC) and Pairing-Based Cryptography (PBC) in Sections 2.3 and 2.4, respectively. Such sections also described the assumptions on which these schemes rely their security, such as the Discrete Logarithm Problem (DLP) and the Bilinear Diffie-Hellman Problem (BDHP). Section 2.5 described how through Identity-Based Encryption (IBE) it is eliminated the need of using digital certificates or looking up for public keys from a directory. The definition of Attribute-Based Encryption (ABE), its main characteristics and the advantages it has over IBE and other PKC schemes were given in Section **??**. Then, Ciphertext-Policy Attribute-Based Encryption (CP-ABE) and the existing approaches for describing access control policies were explained in Section 2.7. Finally, Searchable Encryption (SE) and Attribute-Based Searchable Encryption (ABSE), as well as their basic operation models were discussed in Sections 2.8 and 2.9, respectively.

3

Related Work

This chapter presents the most representative related works found in the state of the art from both system level and implementation level perspectives of cloud storage systems considering searchable encryption. First, it is introduced an overview of the most relevant ABSE proposals which constitutes the foundations of this thesis project. Then, the existing proposals for addressing the problem of performing searches over encrypted data stored in the cloud, with and without focusing on the attribute-based approach, are described. Derived from the literature review, there are finally presented the essential requirements for secure cloud storage and data sharing systems.

3.1 Overview

One of the first proposals that focus on the SE attribute-based approach is presented by Wenhai Sun et al. in [44]. In that paper is proposed a scalable scheme, which lets users to perform searches of one or several keywords on encrypted data in a multi-owner and multi-user context. Search authorizations are based on the use of CP-ABE, providing fine-grained access control at files level. In this way, search

complexity is reduced since it becomes linear in relation to the number of attributes determined by the data owner, instead of being associated with the number of authorized users. However, a secure index is generated for each file, which is based on an AND gate access structure. In the context of cloud storage and data sharing scenarios, it entails a major drawback when the number of files to be stored by the cloud service provider increases rapidly because the amount of secure indexes to be verified also grows proportionally. On the server side, it implies managing as many access structures as files the owner has, which could lead to a computation overhead when there is a lot of secure indexes to search into. Also, although that work considers users revocation, it is done by re-encrypting the files indexes and updating the users' secret keys that are still considered legitimate. This becomes a process that closely resembles the one required by a symmetric approach.

In [57] it is proposed a scheme that lets data owners to ensure information confidentiality by defining an access control policy. Thus, only the users who possess the attributes defined in the policy will be able to search, and subsequently to access the files contents. In addition to protecting the data owners privacy, users privacy is also guaranteed. This is due to the fact that a data owner does not have to know which keywords users are looking for on the outsourced data. A data user can verify if the cloud service provider has properly executed the search and retrieved all the results that are expected to be obtained. Nonetheless, the scope of this proposal is limited to the use of static data, without considering dynamic data. It is also required that the access credentials for data owners and data users are sent through private communication channels, which cannot be always guaranteed.

The solution proposal presented in [18] describes the implementation of a multi-owner and multi-user scheme based on CP-ABE. This becomes of particular importance because, opposed to most existing proposals, real cloud storage and data sharing scenarios consider many owners who produce data and later share it with many users. That proposal uses a secure index that is encrypted with its technique called EABSE, and a symmetric scheme to encrypt the data owners' files. In this way, only when the attributes of an entity satisfy the access policy, the access to the encrypted data is allowed.

Thus, the service provider could only execute search operations over the encrypted data, but cannot obtain or learn information from the files collection it stores. While a symmetric approach is used to encrypt the files in order to ensure system efficiency, it is not specified how the key distribution problem is solved, which is associated to symmetric encryption. That proposal guarantees privacy of both data owners and data users as trapdoors can be created without the owners become aware of what users want to search. However, the scheme does not consider features like secure index update, users revocation or its implementation on multi-cloud environments.

Authors in [32] present a hybrid encryption scheme that combines the SSE and ABE schemes. On one side, data owners guarantee information confidentiality by encrypting their data using SSE, which allows future searches by users. On the other side, through CP-ABE it is guaranteed fine-grained access control which lets data owners to share their data with multiple users in a secure and efficient way. The scheme considers users revocation as a separate part of the ABE scheme in order to avoid data re-encryption and that unauthorized users can still satisfy the access policy defined by the data owner. Each time a data user requests the cloud service provider to perform a search, the server verifies through a timestamp if the user access authorization has not been canceled. Even if the user has the attributes specified in the access policy defined by the data owner, the timestamp helps to validate if such user is still considered as an authorized user or if his status has changed. A major drawback of this proposal is the lack of expressiveness of the access policy, which makes it much more restrictive. The scheme has not been implemented in a multi-cloud environment where different users or organizations could be using completely different cloud platforms.

In [48] is proposed a hierarchical ABE scheme in which a set of files that share the same access structure can be encrypted together. That is, if all files share the same structure, then they can be encrypted together, instead of being encrypted individually. Based on the TF-IDF model and the attributes assigned to the files, the access structure is built using an ARF[1] tree to organize the documents vectors that will later enable the information retrieval. However, the deep search

[1] Attribute-based retrieval features tree

algorithm designed to achieve data retrieval uses a greedy strategy which does not allow to affirm the way in which the tree nodes are splitting is adequate. In other words, such approach may involve increased processing overheads since greedy algorithms select local optimal solutions at each step even if them do not lead to a global optimal solution. Also, the tree nodes only represent AND gates, which limit the documents' attributes assignment. Finally, the proposal efficiency was only theoretically analyzed and evaluated by simulation, so its effectiveness is not guaranteed in practice.

Table 3.1 shows a comparative summary of the existing proposals in the state of the art mentioned above. This summary describes the main features each proposal has, as well as their respective contributions, both defined by the identifiers $C_1 - C_{10}$.

Table 3.1: Review of most relevant ABSE proposals.

Proposal	Features									
	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}
[44], 2014	✓	✓	✓	✗	✓	✓	✓	Symmetric	Unknown	AES, CP-ABE
[57], 2014	✓	✗	✗	✓	✗	✓	✗	Symmetric	1024-bit DLOG	AES, CP-ABE, DSA, Bloom Filters
[48], 2018	✓	✗	✓	✗	✗	✗	✗	Symmetric	Unknown	AES, CP-ABE, ARF
[18], 2018	✓	✗	✗	✗	✗	✓	✓	Symmetric	Unknown	AES, CP-ABE
[32], 2019	✓	✓	✗	✗	✓	✓	✓	Symmetric	Unknown	SSE, CP-ABE

Notation:

Field	**Meaning**	**Field**	**Meaning**
C_1	Fine-grained access control	C_6	Tokens privacy
C_2	Users revocation	C_7	Multi-owner and multi-user context
C_3	Multi-keyword search	C_8	Pairing type
C_4	Results verification	C_9	Security level
C_5	Secure index update	C_{10}	Algorithms and components used

3.2 System Level

The problem of performing searches on encrypted data stored in the cloud has been addressed from multiple perspectives on several occasions. Even though SSE is the most widely used approach to accomplish searching tasks on encrypted data stored at the cloud, it involves some inconveniences, such as the provision of poor security levels and information leakage. However, the main problem of SSE is the same of the traditional symmetric encryption schemes: a secure and efficient key distribution with authorized users.

In [22] Kamara et al. presents CS2, a proposal focused on searchable cloud storage systems and based on the SSE construction described in [12, 13] by Curtmola et al. However, these specifications were slightly modified to achieve a dynamic index-based SSE scheme that is able to deal with files addition and deletion. CS2 introduces the concept of search authenticators, which are constructed based on Merkle hash trees and incremental hash functions. Those authenticators are used by data users as a tool to verify the correctness from both search operations performed by the cloud service provider and the retrieved results. It also provides two protocols to perform single-keyword searches: standard search and assisted search, where a user selects the files subset that must be retrieved. However, this proposal does not provide a mechanism to share data with other users, and only works as a personal storage system. Because of it, the symmetric key management and distribution is not considered, since the data user is the same as the data owner.

The work presented in [19] guarantees the provision of four basic security services as a part of the cloud storage system. This is done through the use of cryptographic protocols and primitives, along with traditional information retrieval techniques. By a side, data confidentiality is achieved by using symmetric encryption, and the inherent access control to the decryption keys, as well as the users' search permissions, is guaranteed through CP-ABE. On the other hand, users authentication is done by the use of digital certificates and data integrity is assured by digital signatures verification. This proposal allows users to search multiple keywords, and the retrieved results are ranked to return

the user just the top-k most relevant documents. Searching capabilities of the storage provider are endowed by Homomorphic Encryption (HE) due to the possibility of performing arithmetic operations over the ciphertext without revealing its original contents, as if the operations were done with plaintext. Nonetheless, existing HE algorithms are rather inefficient and not feasible for its practical use due to the time complexity associated with operations performing on encrypted data.

In [8] is proposed a framework derived from an adjustment of secure *k*-NN technique, called Secure Inner Product Computation. Such framework grants users multi-keyword ranked search, and uses coordinate matching similarity measure to determine data relevance. It further uses inner product similarity, instead of euclidean distance, to evaluate the precision of the selected similarity measure. Its system model considers the same entities as the simple cloud storage model, and the algorithms involved in the solution deployment are quite similar to those taken into account in SSE or ABSE schemes. The main difference with respect to those schemes is the creation of a symmetric key during the setup phase, which later will be used to build the secure index and to create the query trapdoors needed by the service provider to perform searches over owners' encrypted data. This proposal is not intended to provide data or search access control mechanisms, and do not consider key management and its distribution as part of the solution. Because of it, and in order to achieve system's efficiency, access patterns are not protected, which suppose a potential risk of information leakage.

The authors of [33] propose a first approximation of a protocol based on SSE which aims to guarantee data owners the privacy of their information, regardless of whether data is stored or not across multiple geographic locations. This protocol is also intended to allow users the generation of trapdoors for a given keyword that can be later sent to the cloud service provider in order to perform searching tasks over encrypted data. However, it only remains as a theoretical proposal and its real performance and effectiveness in a cloud environment had not been demonstrated. On the other hand, in [32] the main idea outlined in [33] is carried out and CP-ABE is added in order to provide an efficient access control mechanism over data owners files, and to guarantee that only those

authorized users can have access to them. It also tries to address the problem of users revocation using a revocation list and producing timestamps, both of them managed by a revocation authority. This entity helps the service provider to validate if a user has been blacklisted since the last time he got the decryption key for a given file from the entity responsible for storing all the data owners' decryption keys, and consequently if it is still authorized or not to access to such file. Even though this proposal was displayed and tested in a controlled cloud environment, it is still pending its deployment in a multi-cloud setting where domestic or corporate users must employ different cloud platforms.

The proposal presented in [42] employs the full-text retrieval strategy to properly meet the information needs of cloud storage systems users. Such method consists in the extraction of all the contents of each document included in the files collection that a data owner wants to outsource. The extracted contents are pre-processed in order to eliminate stop words and punctuation marks that are not meaningful or representative for the document's contents. Then, the remaining words help to form compound words that will be grouped according to its similarity. For each created group its corresponding Bloom filter is generated. The set of Bloom filters serves as the basis for a hierarchical tree index used by the service provider to search when a user asks for a given keyword or set of keywords.

In [54] is proposed a cloud information retrieval framework which involves two deployment protocols: document outsourcing and document retrieval. The first protocol consists in the documents pre-processing, the words dictionary definition, and the subsequent secure index construction. The document retrieval protocol considers the query generation given an information need, the search process over the index tables stored at the service provider side, and the found results ranking. To get those files that could meet the user's information need, it also employs different information retrieval models, such as vector space, probabilistic and language models, which work together in order to get the subset of most relevant files from all the results found by the cloud service provider. However, it does not consider the deployment of access control mechanisms, since the distribution and management of the symmetric key used to build the secure index and to generate

the user's trapdoors is done between the owner and each authorized user. That is, the data owners have to define who is authorized to access their data and then share with them the key(s) required to create the query trapdoors used later to be compared against the server's index tables.

On the other hand, the work presented in [49] uses the attribute-based approach to guarantee fine-grained access control over the data owners' files given the data users' search queries, thus allowing an easy data sharing among multiple data users. It also considers the user's attributes revocation by creating a sort of multiple secret key which could be defined as $SK_u = \{sk_1, sk_2, ..., sk_n\}$, where sk_i stands for the *i*-th attribute of the user. That is, for each attribute the user possess, it is created a secret 'sub-key' which makes easier the attributes revocation by updating only the sk related with the attribute that needs to be updated, while the rest of the keys in SK_u remain intact. Nonetheless, such proposal was not proved on a real environment and the performance evaluation was carried out by using simulations just to measure the computational time needed for the public parameters and keys generation, as well as the encryption and decryption of the symmetric keys used to encrypt the owner's files. In other words, the trapdoors generation, the files retrieval, and the user's attributes revocation, which are the core contributions of this work, are not yet evaluated.

In [53] Wu et al. present a verifiable PEKS scheme for a multi-user setting based on DGHV[2] homomorphic encryption. The secure index data structure allows the *CSP* to perform searching operations without requiring trapdoors. *DO*'s files encryption in such proposal is done by means of a proxy re-encryption public key algorithm, but it is not discussed if it is implemented some access control mechanism over the encrypted data. That scheme considers a key generation center responsible for the creation of the system and users' keys as well as of the distribution of the secret keys through secure communication channels. Authors in [17] also propose a scheme based on homomorphic encryption, which is focused on nearest neighbor search on high-dimensional encrypted medical images. In such approach, owners' data is assumed to be encrypted by using a Paillier cryptosystem, but no access control mechanisms are introduced either for data or search. As a tool

[2]Van Dijk, Gentry, Halevi and Vaikuntanathan's Fully Homomorphic Encryption scheme

for ensuring increased security, its system model includes two honest but curious *CSP*s that work together as a federated cloud system, where one server stores *DO*'s encrypted data while the other server stores the secret key of the Paillier cryptosystem. Even though that proposal is able to manage data updates and to transform high-dimensional data into points in a lower dimensional space in order to reduce search response times, it requires the use of secure communication channels for data exchange and each potential user has to be authorized in advance by the data owner.

In Table 3.2 it is provided a comparative summary of the existing cloud-based SE proposals in the literature described above. This summary highlights the main features each proposal has in order to identify its contributions according to the notations and descriptions provided also in Table 3.2.

Table 3.2: Generalities of cloud-based Searchable Encryption systems in the literature.

Proposal	M(k)	SE	S_{type}	AC		IR_B	Exp.	DS	λ[1]	
				data	search				data	search
[22], 2010	✗	SSE	SKS	✗	✗	✓	✓	✗	128	80
[19], 2013	✓	HE	MKS	✓	✓	✓	✓	✓	$\geq$ 128	80
[8], 2014	✗	SIPC	MKS	✗	✗	✓	✓	✗	$\geq$ 128	-
[33], 2016	✗	SSE	SKS	✗	✗	✗	✗	✗	-	-
[42], 2017	✗	H(w) & BF	SKS/MKS	✗	✗	✓	✓	✗	-	128
[54], 2018	✗	SSE	SKS/MKS	✗	✗	✓	✓	✓	-	-
[49], 2018	✗	ABSE	MKS	✓	✓	✗	✗	✓	$\geq$ 128	80
[53], 2018	✓	HE	MKS	✗	✓	✓	✓	✓	-	-
[17], 2018	✓	HE	Other	✗	✗	✓	✓	✓	-	-
[32], 2019	✓	SSE	SKS	✓	✓	✓	✓	✓	-	-
Ours, 2020	✓	ABSE	MKS	✓	✓	✓	✓	✓	$\geq$ 128	$\geq$ 128

[1] In bits

Notation:

Field	Meaning
M(k)	Is the distribution and management of the encryption key considered in the solution?
SE	Technique used to implement searchable capabilities over encrypted data in the system.
S_{type}	What is the searching process based on (keyword, multi-keyword, etc)?
AC_{data}	There is an access control mechanism considered in the system for *DO*'s data?
AC_{search}	There is an access control mechanism considered in the system for *DU*'s search queries?
IR_B	Does it use benchmarks to evaluate the information retrieval efficiency?
Exp.	Does it include an experimental evaluation?
DS	Does it allow data sharing among users?
λ_{data}	Data encryption security level.
λ_{search}	Search scheme security level.

Abbreviation	Meaning	Abbreviation	Meaning
SSE	Searchable Symmetric Encryption	SIPC	Secure inner product computation
SKS	Single-keyword search	H(w) & BF	Hash functions and Bloom Filters
HE	Homomorphic Encryption	ABSE	Attribute-Based Searchable Encryption
MKS	Multi-keyword search		

3.3 Implementation Level

Although in the state of the art there are multiple proposals that uses ABSE to provide searching capabilities over outsourced encrypted data, each proposal has its own characteristics and provides different added functionalities mainly because the minimum functions that must be guaranteed have not been defined yet. For this reason, the standard metrics that should be used to measure the performance of the proposed schemes have also not been established. There are still some challenges that have not been addressed, such as an imbalance between searching capabilities and search efficiency, as well as if it is convenient to manage dynamic secure indexes or not.

As shown in the summary of the existing cloud-based ABSE research works presented in Table 3.3, all the existing proposals uses symmetric pairings, which are unsuitable for providing security levels higher than 80-bit. Most of the proposals use CP-ABE instead of KP-ABE because the access control policy is applied to the plaintext data and the user's secret keys are associated to attributes.

CP-ABE leads to a more efficient deployment for cloud storage and data sharing scenarios since data owners decide which attributes a user must satisfy to access his data. In addition, the predominant type of access structure used is the access tree, which efficiency is also determined by the attributes universe type and policy expressiveness used. For example, if a small attributes universe and a non-monotonic policy is used, it must be verified if a user possess each of the attributes previously defined in the attributes universe. Even when almost all the proposals were experimentally evaluated, just a few of them considered the use of benchmarks to perform the assessment of the information retrieval quality.

Table 3.3: Attribute-based approaches in cloud-based Searchable Encryption systems.

Proposal	e_{type}	ABE	$\mathbb{A}$	U	M	S_{type}	IR_B	Exp.	Reqs.			$R4-\lambda$[1]	
									$R1$	$R2$	$R3$	$R1$	$R2$
[7], 2015	type-1	KP/CP	Tree	L	M	MKS	✗	✓	✗	✓	✗	-	-
[50], 2017	type-1	KP	Matrix	S	M	MKS	✗	✓	✗	✓	✗	-	80
[31], 2018	type-1	CP	Tree	S	M	MKS	✓	✓	✓	✓	✗	-	80
[46], 2018	type-1	Other	Other	S	NM	SKS	✗	✓	✗	✓	✗	-	80
[49], 2018	type-1	CP	Matrix	S	M	MKS	✗	✗	✓	✗	✗	-	80
[18], 2018	type-1	Other	Other	S	NM	SKS	✗	✗	✗	✓	✗	-	80
[27], 2019	type-1	KP	Tree	S	M	MKS	✗	✓	✗	✓	✗	-	80
[55], 2019	type-1	KP	Tree	S	M	SKS	✓	✓	✗	✓	✗	-	80
[56], 2019	type-1	CP	Tree	L	M	SKS	✓	✓	✗	✓	✗	-	80
[32], 2019	type-1	CP	Tree	L	M	SKS	✓	✓	✓	✗	✗	-	80
Ours, 2020	type-3	CP	Tree/Matrix	L	M	MKS	✓	✓	✓	✓	✓	≥ 128	≥ 128

Notation:

Field	**Meaning**
e_{type}	Pairing setting used.
ABE	ABE family which the proposal belongs.
$\mathbb{A}$	Access structure type.
U	Attributes universe type.
M	Kind of access structure being supported (degree of policy expressiveness).
S_{type}	What is the searching process based on (keyword, multi-keyword, etc)?
IR_B	Does it use benchmarks to evaluate the information retrieval efficiency?
Exp.	Does it include an experimental evaluation?
Reqs.	Does the requirements $R1-R3$ are accomplished?
$R4-\lambda$	Achieved security levels.

[1] In bits

Abbreviation	Meaning	Abbreviation	Meaning
KP	Key-Policy Attribute-Based Encryption	MKS	Multi-keyword search
CP	Ciphertext-Policy Attribute-Based Encryption	S	Small universe
L	Large universe	NM	Non-monotonic policy
M	Monotonic policy	SKS	Single-keyword search

Based on the literature review presented so far, in this thesis it is stated that the minimum requirements for the deployment of a security scheme in cloud storage and data sharing scenarios are the following:

R1. Data owners files should be encrypted efficiently with a symmetric cipher, and access control over data could be enforced using ABE, thus allowing secure data sharing.

R2. Searching capabilities for data users are provided by an efficient ABSE construction, which grants fine-grained access control and enables secure information retrieval.

R3. Ranking of information retrieval results.

R4. Security levels must accomplish current standards, such as *NIST SP800-57* and *ECRYPT-CSA D5.4*.

3.4 Essential Requirements for a Cloud Storage and Data Sharing System

As shown in Figure 3.1, the simple cloud storage model includes three main entities: data owner, data user, and cloud service provider. Each one has a certain role within the system model, as described below. The data owner has a files collection that requires a storage repository; he/she selects which data wants to outsource to the cloud service provider and sends such data to it. Also, the owner selectively shares certain data with other users, giving them access permission to the shared files. In this context, the cloud service provider is responsible for storing the data owners' files collections

and managing the data updates for each outsourced collection. When a data owner shares a file or set of files with others, the storage provider must allow the authorized users access the data owner's shared files. Once a data owner shares a file or set of files, a user can look up for data of interest on the pre-shared data owner's files. To do so, the user has to send to the cloud service provider a search query that represents some information need. The service provider performs search operations over the data owner's files collection when a query is received from an authorized user and returns her the search results for the given query. Upon receiving the query results, the user requests the storage provider the download of all the matching files or just a few of them, and the service provider delivers to the data user the requested files.

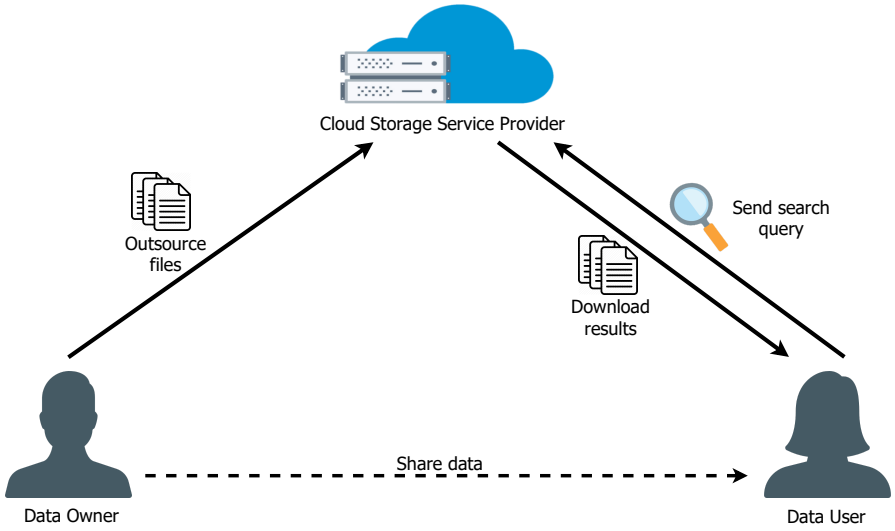


Figure 3.1: Cloud storage system model.

After the literature review of cloud-based searchable encryption systems, either SSE, PEKS or ABSE approaches, the minimum requirements identified for cloud storage systems are listed below.

- Provide users a storage repository to outsource their files collection.
- Guarantee easy data sharing among users of the storage service.
- Manage data updates in the data owners' files collection.
- Manage the data users' access permission to shared files.

- Allow users to search over the files shared with them.
- Deliver the requested files to the data users.

3.5 Chapter Summary

This chapter discussed the most representative related works about searchable encryption in cloud environments, addressed from both system level and implementation level perspectives. The literature review allowed the identification of the contributions made by each proposal as well as the challenges that remain unsolved or could be faced in an alternative way with the aim of improving the results obtained until now. Besides, derived from the review of the state of the art, it were defined the essential requirements that must be attended in order to provide data confidentiality and efficient access control mechanisms while preserving searching and retrieval capabilities in cloud storage systems. The next chapter describes the architecture of FABECS, the security scheme proposed in this thesis to meet the previous stated requirements.

4

Searchable Data Encryption Scheme

This chapter presents the definition of the FABECS architecture, as well as the modules that integrate the proposed scheme in order to provide searching and information retrieval capabilities for secure cloud storage and data sharing scenarios. The description of all the operations performed in the modules is divided and presented focusing on the ABSE insight, which consists in a novel implementation based on asymmetric pairings, and the information retrieval insight, which incorporates the full-text retrieval method.

4.1 System Model

This thesis presents a novel security approach called FABECS, for cloud storage, sharing and retrieval of encrypted data fully constructed on the basis of Attribute-Based Encryption (ABE). It provides confidentiality for outsourced data and access control over its encryption keys, by means of ABE, as well as privacy and access control for searching and retrieval of encrypted outsourced data through ABSE. FABECS is built on the foundations of DET-ABE [36] and CP-ABSE [56], both relying on

the tree-based version of the CP-ABE scheme proposed in [5] (hereafter referred to as BSW07), but also in the DET-ABE extended version, known as AES4SeC [37], which is based on the matrix version of CP-ABE proposed in [52] (onwards referred to as W11). In other words, our proposal was designed to be compliant with both tree-based and matrix-based approaches of CP-ABE.

The system model considered in this thesis is shown in Figure 4.1; as the cloud storage system model, it includes three main entities: data owner *DO*, data user *DU*, and cloud service provider *CSP*. The data owner has a files collection that wants to outsource to a cloud storage service. To do so, *DO* first builds from the documents in the files collection a secure index by using *FABECS.Proc* and *FABECS.Index*, and then encrypts the files by using the symmetric cipher AES and encrypts the symmetric key by using CP-ABE to create a digital envelope. Finally, *DO* sends both the encrypted files collection and the secure index to the service provider. The cloud service provider stores the data owners' files collection and updates its secure searchable index in order to be able to perform searching tasks when a user requests it. When data users have an information need, they create a trapdoor that represents an encrypted version of a query and forces the service provider to look into its searchable index without knowing what exactly it is looking for. After receiving a query trapdoor and performing a search, the *CSP* ranks the results by using *FABECS.Rank* and returns to the users the most relevant results found. The notation used in this chapter is shown in Table 4.1.

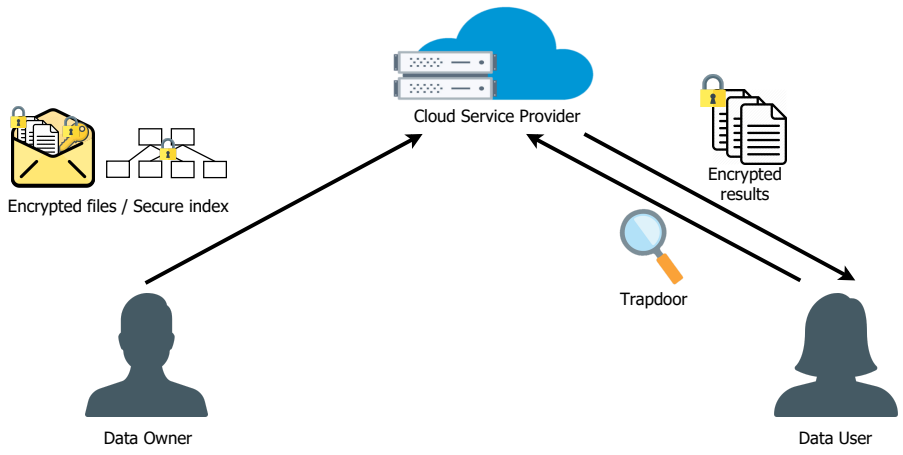


Figure 4.1: System model of the solution proposal.

Table 4.1: Notation and descriptions.

Notation	Description	Notation	Description
λ	Security level.	Q	Data user's search query.
P_λ	System public parameters.	K	Original query keywords.
PK	CP-ABE master public key.	QW	Set of query words given by a data user.
MK	CP-ABE master private key.	w_q	Data user's query word ($w_q \in Q$).
SK_u	CP-ABE user secret key.	w_s	Single query word.
S_u	User's attributes set.	w_c	Query compound word.
$\mathbb{A}$	Access structure.	n_1	Number of single keywords in QW.
D	Data owner's files collection.	n_2	Number of compound keywords in QW.
W_j	Words dictionary for a document $d_j \in D$.	T_ω	Set of user's query trapdoors.
M_{w_i}	Membership value of a word $w_i \in W_j$.	$T_u(w_q)$	Data user's trapdoor for a given query word w_q.
w_i	Index word from a word dictionary W_j	E_{w_c}	Membership entropy for a compound word.
CT_{w_i}	Encrypted index word w_i.	S_{w_c}	Ranking score weight for a query word.
SI	Secure index.	S_{d_j}	Ranking score of a document.
k	Symmetric encryption key.	SSI	Secure searchable index.
CT	CP-ABE ciphertext.	D_{rslt}	Resulting documents for a user query set.

4.2 Proposed ABSE Scheme

As said before, FABECS considers the same entities as the basic cloud storage system model, but some of the operations each actor is responsible to perform are slightly modified in order to enable searching capabilities over encrypted files collections. This section describes the proposed solution approach, as well as the details of the operations involved in the system model. Figure 4.2 depicts the whole system model and the components interaction internally, on each actor's side, and externally, when an entity sends or requests data to another entity. The data owner defines an access structure $\mathbb{A}$ which will be applied to both the files collection and the words dictionary derived from such collection to prevent unauthorized access. Depending on *DO*'s criteria, the access structure could be either different or the same for each item. The encrypted data is then outsourced to the cloud service provider, who is responsible for storing the users' files and the secure searchable index. Data users needs are expressed as a query for which a query trapdoor is created and later sent to the service provider. When the *CSP* receives a trapdoor it searches for the embedded query over its secure searchable index and ranks the results matching the query, so that only the most relevant

results are returned to *DU*. Finally, the data user decrypts the retrieved files by using SK_u to recover the symmetric key needed to decrypt and obtain the files in plaintext form.

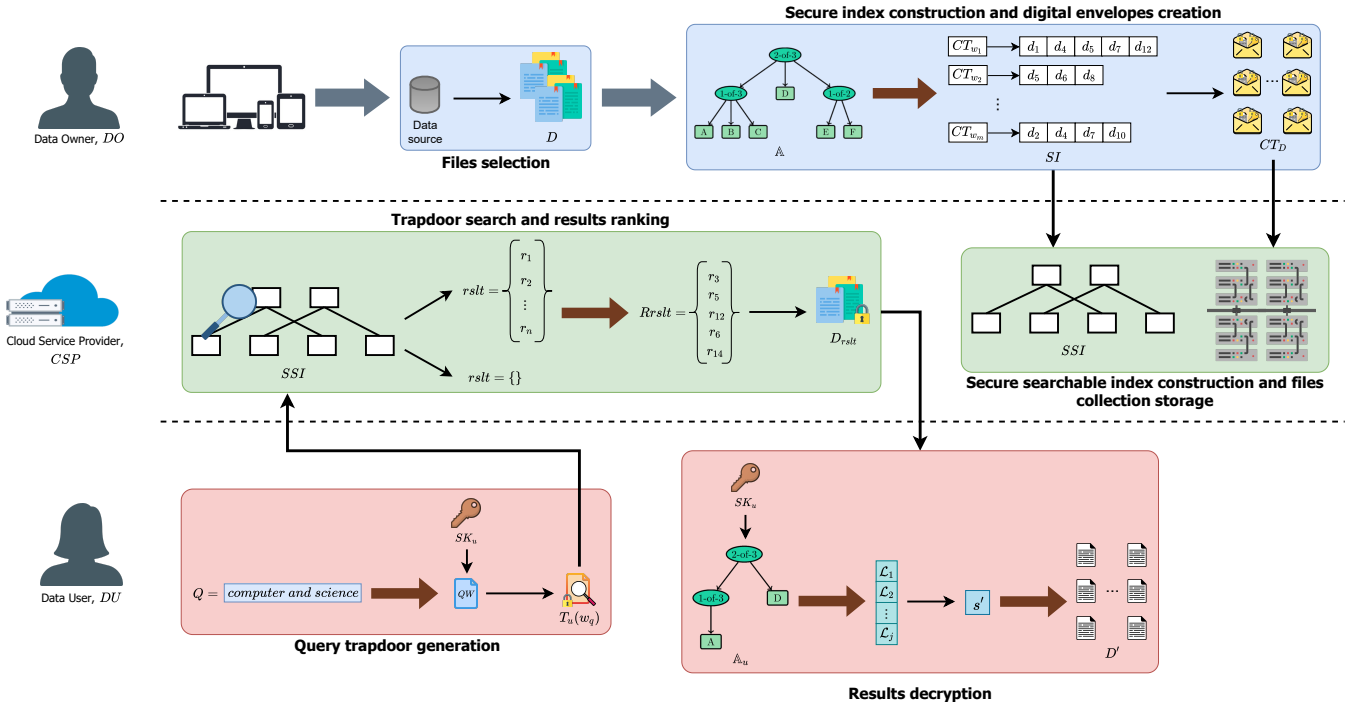


Figure 4.2: Diagram of the solution proposal.

4.2.1 Data Owners' Operations

Before outsourcing a files collection, the data owner has to perform the system initialization in order to set up the system public parameters P_λ, the master public key PK and the master private key MK. Given a security level λ, the system initialization process consists in the definition of the bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ that serves as the basis of the subsequent CP-ABSE and DET-ABE operations. There are also defined two secure hash functions H_1 and H_2 that maps a set of bits to an element of $\mathbb{G}_1$ or $\mathbb{G}_2$ (depending on the version), and to an element of $\mathbb{Z}_r$, respectively. Algorithm 1 shows the operations sequence of the system initialization for the tree-based approach of FABECS, while Algorithm 2 shows the operations related to its matrix-based approach.

Algorithm 1 FABECS.Setup (BSW07)

Require: λ

Ensure: P_λ, PK, MK

1: Generate three multiplicative groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ of prime order r and a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, where g_1 is the generator of $\mathbb{G}_1$ and g_2 is the generator of $\mathbb{G}_2$.

2: Generate two secure hash functions $H_1 : \{0,1\}^* \to \mathbb{G}_2$ and $H_2 : \{0,1\}^* \to \mathbb{Z}_r$.

3: Define Lagrange coefficients

$$\mathcal{L}_j = \prod_{i=1}^{k-1} \frac{-x_i}{x_j - x_i}, i \neq j$$

4: Randomly choose $\{\alpha, \beta\} \leftarrow \mathbb{Z}_r^*$.

5: Compute g_2^α, $h = g_1^\beta$ and $e(g_1, g_2)^\alpha$.

6: $P_\lambda = \{\mathbb{G}_1, \mathbb{G}_2, e, g_1, g_2, r, H_1, H_2\}$

7: $PK = \{g_1, g_2, h, e(g_1, g_2)^\alpha\}$

8: $MK = \{\beta, g_2^\alpha\}$

9: **return** P_λ, PK, MK

Algorithm 2 FABECS.Setup (W11)

Require: λ

Ensure: P_λ, PK, MK

1: Generate three multiplicative groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ of prime order r and a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, where g_1 is the generator of $\mathbb{G}_1$ and g_2 is the generator of $\mathbb{G}_2$.

2: Generate two secure hash functions $H_1 : \{0,1\}^* \to \mathbb{G}_1$ and $H_2 : \{0,1\}^* \to \mathbb{Z}_r$.

3: Randomly choose $\{\alpha, \beta\} \leftarrow \mathbb{Z}_r^*$.

4: Compute g_1^α, $h = g_1^\beta$ and $e(g_1, g_2)^\alpha$.

5: $P_\lambda = \{\mathbb{G}_1, \mathbb{G}_2, e, g_1, g_2, r, H_1, H_2\}$

6: $PK = \{g_1, g_2, h, e(g_1, g_2)^\alpha\}$

7: $MK = \{g_1^\alpha\}$

8: **return** P_λ, PK, MK

Afterwards, *DO* defines the access structure $\mathbb{A}$ used to enforce access control over the files encryption symmetric key and the index words. The access structure could be either an access control policy AP, if it is used the Shamir's secret sharing version of the proposed scheme, or a formatted Boolean formula FBF, if it is used the linear secret sharing version. Both AP and FBF contain a boolean expression over a set of attributes that must be satisfied by any data user that requests access to the encrypted data. That is, if a data user u wants to access the plaintext of a particular encrypted item, u's attributes must cryptographically satisfy the ciphertext's pre-defined access formula in AP or in FBF.

Algorithm 3 CP-ABSE.EncInd (W11)

Require: $P_\lambda, PK, w_i, \mathbb{A}$

Ensure: CT_w

1: Randomly choose a secret value $s \leftarrow Z_r^*$.
2: Compute $C' = e(g_1^{H_2(w_i)s}, g_2) \times e(g_1, g_2)^{\alpha s}$ and $C = g_2^s$.
3: **for** each $\rho(i)$, attribute j at row $\texttt{Mat}_i$ **do**
4: Randomly choose $r_i, \mathbf{v} = \{s, s_1, s_2, ..., s_{n-1}\} \leftarrow Z_r^*$.
5: Compute $\lambda_i = \mathbf{v} \times \texttt{Mat}_i$
6: Compute $C_{y_i} = g_2^{r_i}$ and $C'_{y_i} = h^{\lambda_i} \times H_1(\rho(i))^{-r_i}$.
7: **end for**
8: $CT_w = \{C', C, [C_{y_i}, C'_{y_i}]^+\}$
9: **return** CT_w

Now, the data owner selects from the data source the files D to be outsourced to the cloud and establishes an appropriate $\mathbb{A}$. Then, a words dictionary W_j is generated from each one of the selected documents. So, the contents of each file in the collection are extracted and pre-processed in order to get the keywords used to build the secure index SI. Each index word $w_i \in W_j$ is encrypted by applying either Algorithm 3 or Algorithm 4, which takes as input the pre-defined access structure

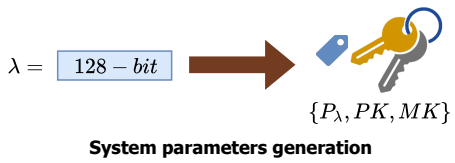
and the index word, as well as P_λ and PK. The encrypted words dictionary W'_j serves as the basis to build SI, which contains each encrypted word and the respective list of files that contain such index word.

Algorithm 4 CP-ABSE.EncInd (BSW07)

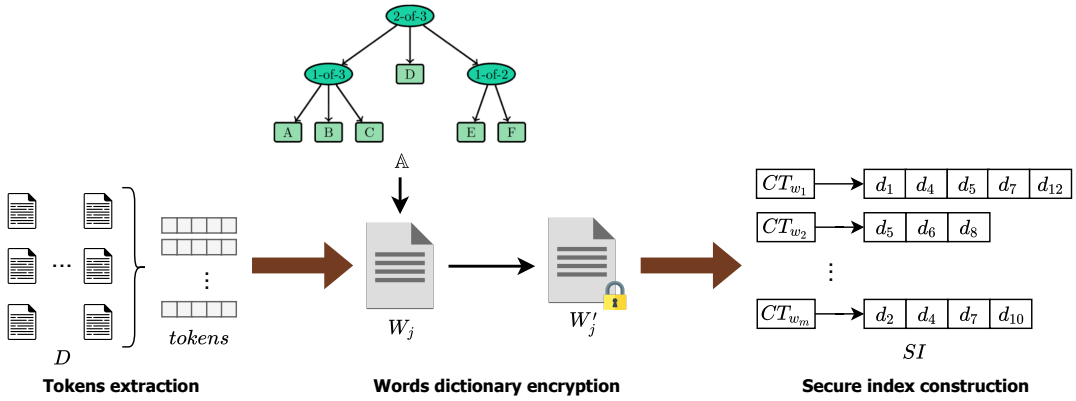
Require: $P_\lambda, PK, w_i, \mathbb{A}$

Ensure: CT_w

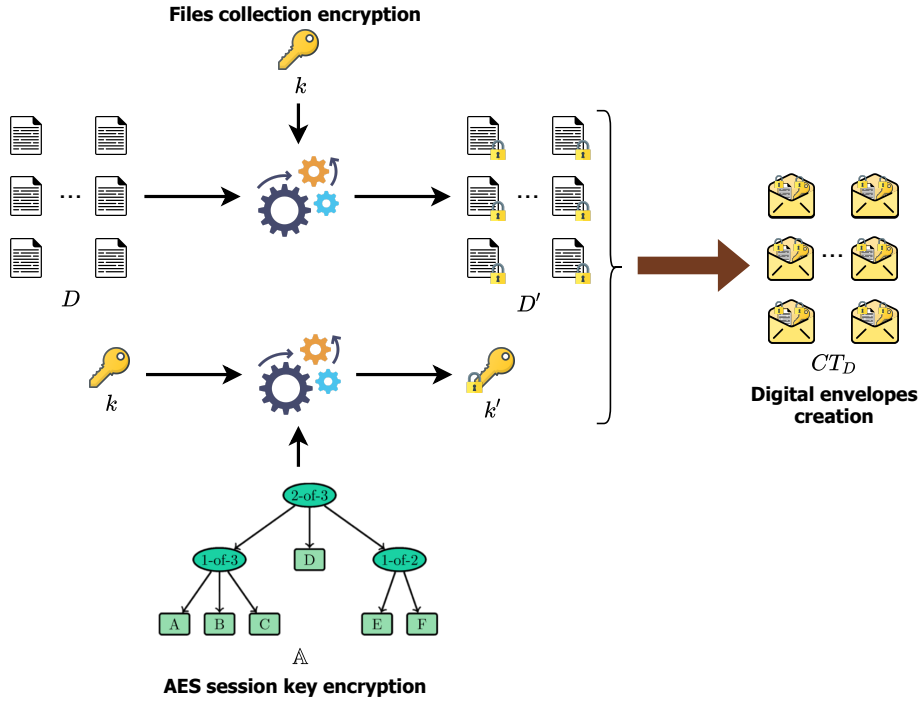
1: Randomly choose a secret value $s \leftarrow Z_r^*$.
2: Compute $C' = e(g_1^{H_2(w_i)s}, g_2) \times e(g_1, g_2)^{\alpha s}$ and $C = h^s$.
3: Let y be the root node of $\mathbb{A}$.
4: **for** each node x in $\mathbb{A}$ (in a top-down manner, starting from y) **do**
5: **if** x is the root node y **then**
6: Randomly choose a polynomial q_y for y with degree $d_y = k_y - 1$ and sets $q_y(0) = s$.
7: Randomly set d_y other points of q_y to completely define it.
8: **else**
9: Randomly choose a polynomial q_x for x with degree $d_x = k_x - 1$.
10: Set $q_x(0) = q_{parent(x)}(index(x))$.
11: Randomly choose d_x other points to completely define q_x.
12: **end if**
13: **end for**
14: **for** each leaf node $y \in \mathbb{A}$ containing attribute j **do**
15: Compute $C_y = g_1^{q_y(0)}$ and $C'_y = H_1(j)^{q_y(0)}$.
16: **end for**
17: $CT_w = \{C', C, [C_y, C'_y]^+\}$
18: **return** CT_w



(a) System initialization.



(b) Secure index creation.



(c) Digital envelope generation.

Figure 4.3: Operations performed at the data owners' side.

After building the secure index, *DO* encrypts the files collection D by means of the symmetric cipher AES. To do so, *DO* uses the *DET-ABE.encrypt* algorithm, which takes as input each file $d_j \in D$, the master public key PK and the pre-defined access structure $\mathbb{A}$, or a new one. The owner's data is encrypted with an AES session key k, which is later encrypted with CP-ABE using PK and $\mathbb{A}$. The result of both encryption processes is a digital envelope CT_D containing both the encrypted file and the encrypted key. Finally, each CT_D generated and the secure index are sent to the external server. The whole process described above is outlined in Figure 4.3.

4.2.2 Data Users' Operations

As described in Section 2.7, each data user possesses a set of attributes S_u that describes the user's characteristics. Those characteristics could be simple, for example, the user's last name, age or social security number; or more complex, such as a job position or a specific condition the user meets. The user's secret key SK_u is associated to S_u and used to decrypt a ciphertext only if S_u satisfies the access structure $\mathbb{A}$ embedded into a ciphertext CT. SK_u is created by using *FABECS.KeyGen* algorithm, which takes as input S_u, the system public parameters P_λ, and the master private key MK. Algorithms 5 and 6 show the operations sequence of the users' secret keys generation for the matrix-based and tree-based approach, respectively.

Algorithm 5 FABECS.KeyGen (W11)

Require: P_λ, MK, S_u

Ensure: SK_u

1: Randomly choose $r \leftarrow Z_r^*$.
2: Compute $D = g_1^{(\alpha+\beta r)}$ and $D' = g_2^r$.
3: **for** each attribute $y \in S_u$ **do**
4: Compute $d_y = H_1(y)^r$.
5: **end for**
6: $SK_u = \{D, D', [d_y]^+\}$
7: **return** SK_u

Algorithm 6 FABECS.KeyGen (BSW07)

Require: P_λ, MK, S_u

Ensure: SK_u

1: Randomly choose $r \leftarrow Z_r^*$.
2: Compute $D = g_2^{\frac{\alpha+r}{\beta}}$, $D' = g_2^{\frac{1}{\beta}}$ and g_2^r.
3: **for** each attribute $y \in S_u$ **do**
4: Randomly choose $r_j \leftarrow Z_r^*$.
5: Compute $d_y = g_2^r \times H_1(y)^{r_j}$, $d'_y = g_1^{r_j}$.
6: **end for**
7: $SK_u = \{D, D', [d_y, d'_y]^+\}$
8: **return** SK_u

When data users have an specific information need, they express that need through a query string Q for which a trapdoor is computed by using *DU*'s secret key. For each query word $w_q \in Q$, either single w_s or compound w_c, it is created a query trapdoor $T_u(w_q)$ by using Algorithm 7, for the BSW07 version of CP-ABSE, or Algorithm 8, for CP-ABSE matrix-based approach. Both algorithms receive as input w_q and the user's SK_u, and output the tuple $T_u(w_q)$, which contains an encrypted version of the user query word derived from the attributes embedded in SK_u. Then, $T_u(w_q)$ is sent to the external server to initiate the search and retrieval process.

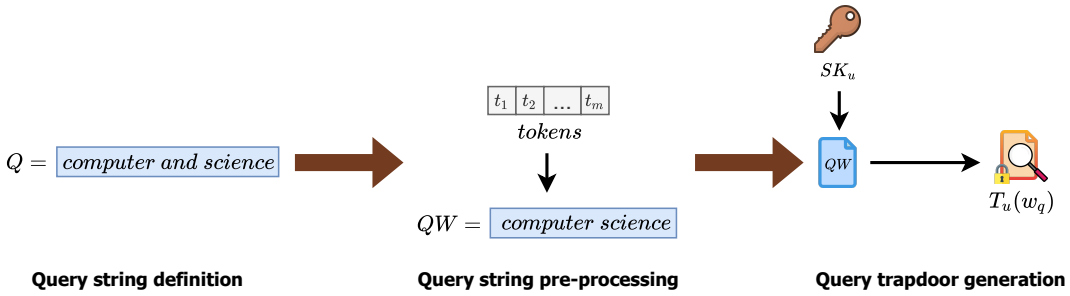
Algorithm 7 CP-ABSE.TrpDr (BSW07)

Require: SK_u, w_q

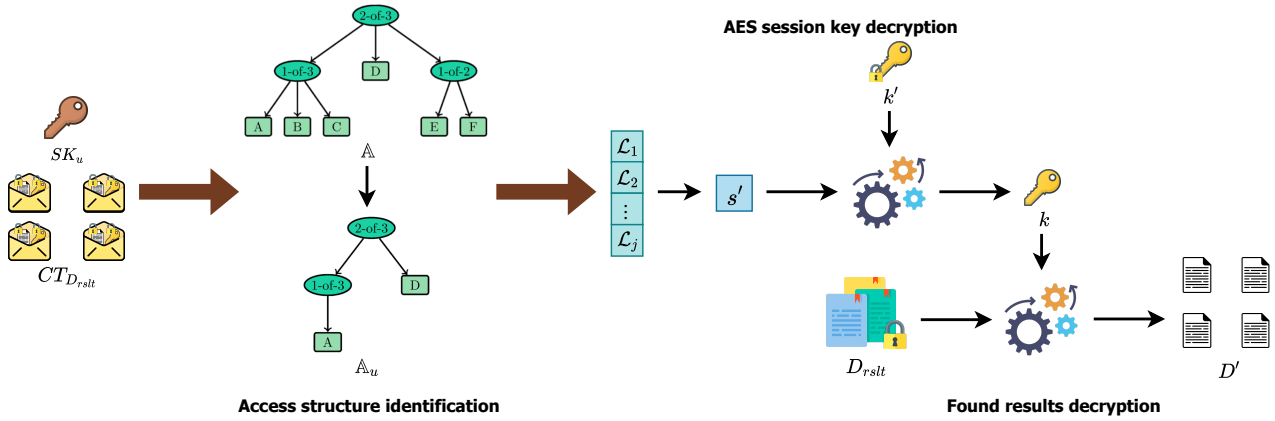
Ensure: $T_u(w_q)$

1: Compute $T = D'^{H_2(w_q)} \times D$.
2: $t_y = d_y \in SK_u$
3: $t'_y = d'_y \in SK_u$
4: $T_u(w_q) = \{T, [t_y, t'_y]^+\}$
5: **return** $T_u(w_q)$

(a) Users' secret key generation.



(b) Trapdoor generation.



(c) Found results decryption.

Figure 4.4: Operations performed at the data users' side.

On the other hand, after sending the search request to the service provider and receiving from it the found results D_{rslt} for a given query, *DU* decrypts the retrieved files set. To do so, *DU* uses *DET-ABE.decrypt* algorithm, which takes as input SK_u and the digital envelope of each $d_j \in D_{rslt}$. First, it is recovered the session key k, encrypted with CP-ABE, and then the document's ciphertext is decrypted by using the AES cipher and k. The data users' operations described so far are illustrated in Figure 4.4.

Algorithm 8 CP-ABSE.TrpDr (W11)

Require: SK_u, w_q

Ensure: $T_u(w_q)$

1: Compute $T = g_1^{H_2(w_q)} \times D$.

2: $T' = D' \in SK_u$

3: $t_{y_i} = d_y \in SK_u$

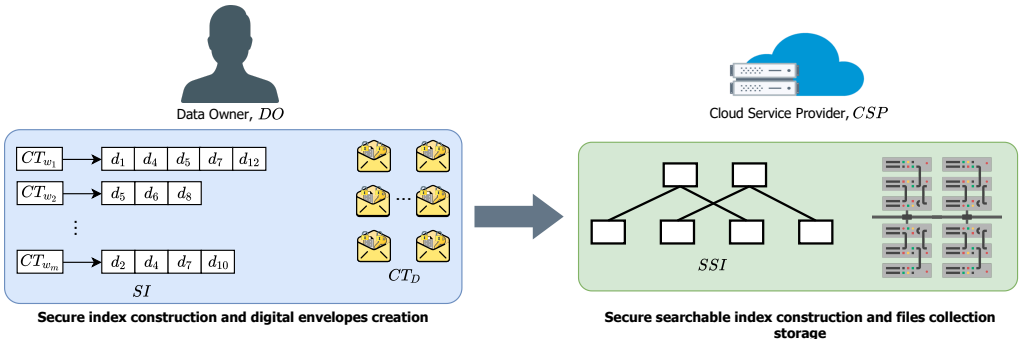
4: $T_u(w_q) = \{T, T', [t_{y_i}]^+\}$

5: **return** $T_u(w_q)$

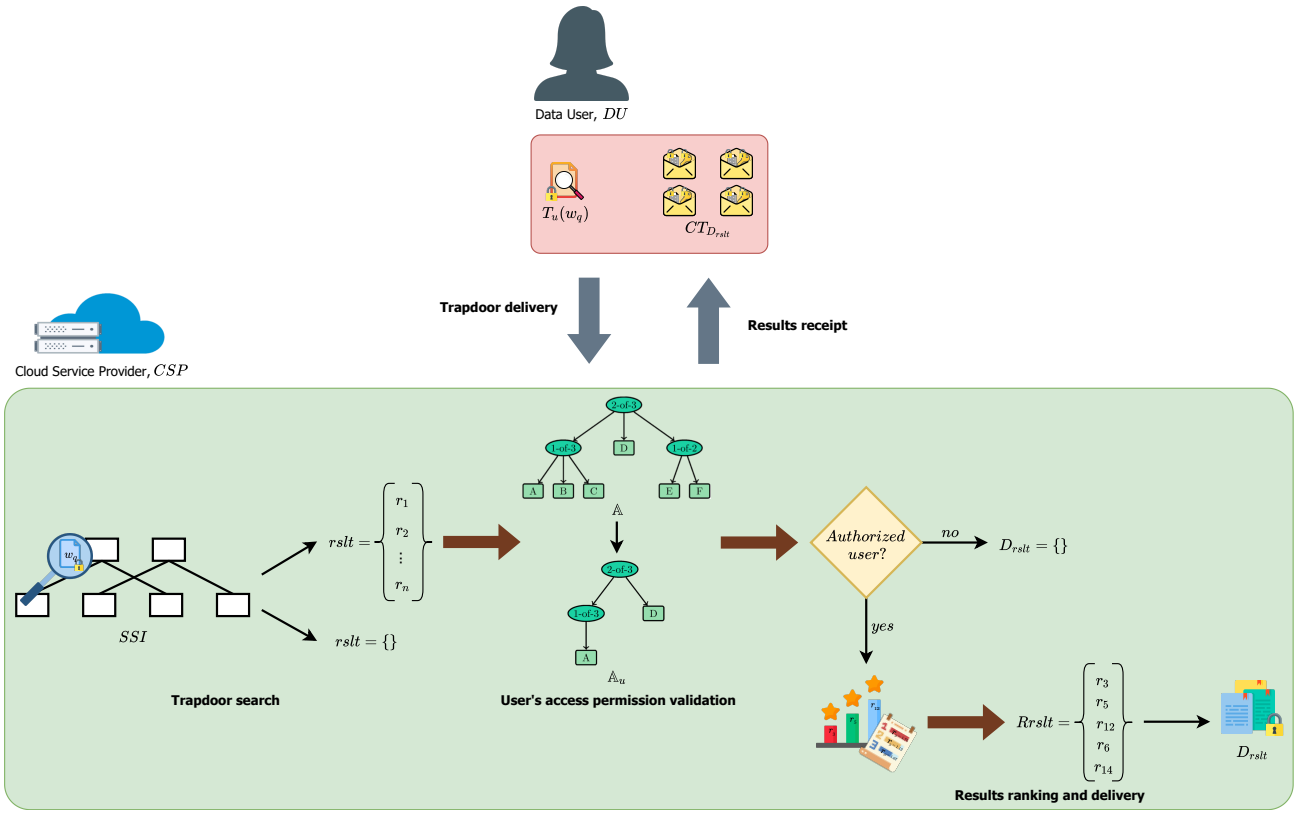
4.2.3 Cloud Service Provider's Operations

The cloud service provider serves as an intermediary between data owners and data users. As shown in Figure 4.5, it is responsible for storing the data owners' files and for the provision of a mechanism to easily share data with other users of the storage service. Such mechanism must involve both an efficient indexing technique to allow searching over encrypted data and a ranking method to return just the most relevant found results to a *DU* after a query search is executed. When the *CSP* receives a query trapdoor $T_u(w_q)$ from a *DU*, it searches over its secure searchable index SSI for the encrypted user query. A match of w_q with an encrypted keyword CT_{w_i} in SSI will be successful only if $w_q = w_i$ and S_u satisfies the access structure embedded in CT_{w_i}.

To do so, it is used either Algorithm 9, for the tree-based approach of CP-ABSE, or Algorithm 10, for matrix-based CP-ABSE. Both algorithms receive as input the query trapdoor submitted by *DU* and the secure index record of the match found. The embedded values in $T_u(w_q)$ are compared with the encrypted version of the found record CT_{w_q} and, if the user attributes cryptographically satisfy the access structure associated to that record, it is added to the results list $rslt$, otherwise it is ignored. Then, the found results in $rslt$ are ranked according to their relevance to *DU*'s query in descending order. Finally, the ranked documents D_{rslt} are delivered to the user.



(a) Data owners' files storage.



(b) Search and retrieval of encrypted data.

Figure 4.5: Operations performed at the service provider's side.

Algorithm 9 CP-ABSE.Search (BSW07)

Require: $CT_w, T_u(w_q)$
Ensure: {**true** $\vee$ **false**}

for each leaf node $y \in \mathbb{A}$ **do**
  Let j denote the attribute associated with the leaf node y.
  if $j \in S_u$ **then**
    Compute

$$F_y = \frac{e(C_y, t_y)}{e(t'_y, C'_y)} = e(g_1, g_2)^{rq_y(0)}$$

  else
    Define $F_y = \perp$.
  end if
end for
for each non-leaf node x in $\mathbb{A}$ (in a top-down manner) **do**
  Let S_x denote an arbitrary k_x-sized set of child nodes z such that $F_z \neq \perp$.
  if no such set exists **then**
    Define $F_x = \perp$.
  else
    Using Lagrange interpolation compute

$$F_x = \prod_{z \in S_x} F_z^{\Delta_{i,S'_x}(0)} = \prod_{z \in S_x} \left(e(g_1, g_2)^{rq_z(0)}\right)^{\Delta_{i,S'_x}(0)} = \prod_{z \in S_x} \left(e(g_1, g_2)^{rq_{parent(z)(index(z))}}\right)^{\Delta_{i,S'_x}(0)}$$

$$F_x = \prod_{z \in S_x} e(g_1, g_2)^{rq_x(i) \cdot \Delta_{i,S'_x}(0)} = e(g_1, g_2)^{rq_x(0)},$$

    where $i = index(z), S'_x = (\forall z \in S_x : index(z))$, and Δ_{i,S'_x} is the Lagrange coefficient.
  end if
end for
Let y be the root node of $\mathbb{A}$.
if $F_y = \perp$ **then**
  return false (The access tree $\mathbb{A}$ is not satisfied by the attributes in S_u)
else
  Recursively compute $F_y = \prod_{y \in \mathbb{A}} \frac{e(C_y, t_y)^{\mathcal{L}_y s'}}{e(t'_y, C'_y)} \equiv e(g_1, g_2)^{rs}$
  Compute $C_T = \frac{e(C,T)}{e(g_1,g_2)^{rs}}$
  if C_T is equal to C' **then**
    return true ($w \equiv w_q$ and the attributes in S_u satisfies $\mathbb{A}$)
  else
    return false
  end if
end if

Algorithm 10 CP-ABSE.Search (W11)

Require: $CT_w, T_u(w_q)$

Ensure: {**true** $\vee$ **false**}

for each $\rho(i)$, attribute j at row $\mathtt{Mat}_i$ **do**
  Let j denote the attribute associated with $\rho(i)$.
  if $j \in S_u$ **then**
    Compute

$$F_y = e(C'_{y_i}, T')e(t_{y_i}, C_{y_i}) = e(g_1, g_2)^{\beta rs}$$

  else
    Define $F_y = \perp$.
  end if
end for
if $F_y = \perp$ **then**
  return false (The matrix $(\mathtt{Mat}, \rho)$ is not satisfied by the attributes in S_u)
else
  Recursively compute $F_y = \prod_{i \in I} \left(e(C'_{y_i}, T')e(t_{y_i}, C_{y_i})\right)^{\omega_i} \equiv e(g_1, g_2)^{\beta rs}$
  Compute $C_T = \frac{e(T,C)}{e(g_1,g_2)^{\beta rs}}$
  if C_T is equal to C' **then**
    return true ($w \equiv w_q$ and the attributes in S_u satisfies $(\mathtt{Mat}, \rho)$)
  else
    return false
  end if
end if

4.3 Proposed Encrypted Data Retrieval Scheme

The FABECS model for the information retrieval task over encrypted data is divided into four parts, as depicted in Figure 4.6:

- **Proc.** It involves the data owner's documents pre-processing to output all the index words

from its contents to effectively represent it.

- **Index.** It consists in the construction of the secure full-text retrieval index and its maintenance mechanism.
- **Query.** It involves the execution of the full-text retrieval algorithm over the secure inverted index to answer a data user information need.
- **Rank.** It consists in the execution of the results ranking after a query is executed by the *CSP*.

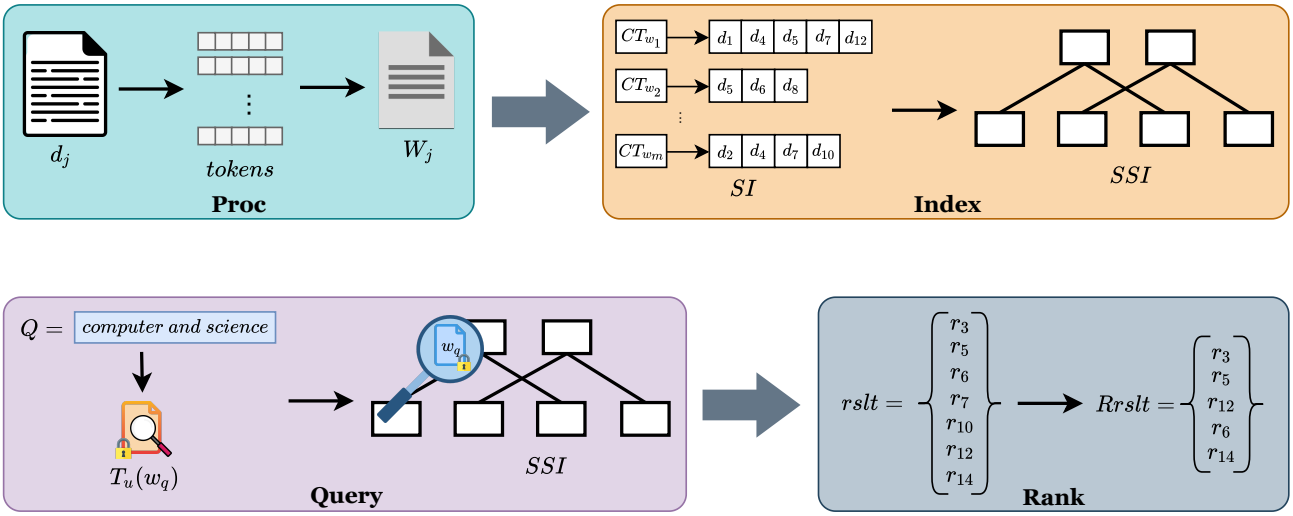


Figure 4.6: Information retrieval model.

Broadly speaking, FABECS.Proc is responsible for the extraction of keywords of a document, its pre-processing and membership measure in order to let FABECS.Index to build the secure full-text retrieval index. The full-text retrieval search engine of FABECS.Query executes the user query over the inverted index and measures the similarity between the resulting documents and the data user query. Finally, by FABECS.Rank the found documents are ordered according to its relevance and based on the query similarity score and index word membership computed by FABECS.Query and FABECS.Proc, respectively.

4.3.1 FABECS.Proc

This module is executed on *DO*'s side. A data owner has a large amount of documents D that wants to outsource to the cloud storage service. Because the documents contains sensitive information that should be shared only with authorized entities, D is encrypted before its outsourcing. To let the *CSP* to perform searches over encrypted data, *DO* builds a secure index from D. To do so, the whole contents of each document d_j are extracted in order to get a preliminary version of the words dictionary W_j that represent in a better way the topic addressed by each file. The tokens obtained from the document's contents are pre-processed in such a way to remove punctuation marks and stop words that are not meaningful to describe the document. This process is depicted in Figure 4.7.

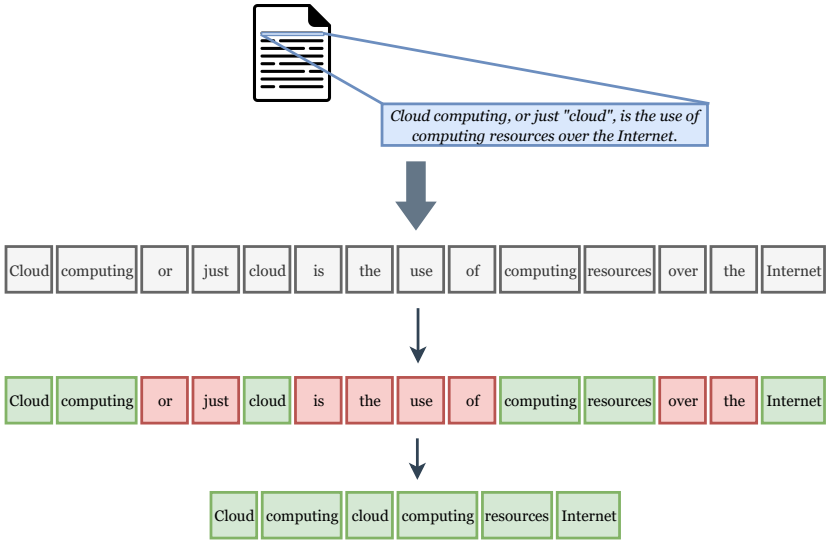


Figure 4.7: Words dictionary pre-processing.

As shown in Figure 4.8, the remaining tokens are used to form compound words of length at most k. The compound words generated will be later indexed along with the single words identified in the document. In this way, it is eliminated the need of knowing the index words offset position to allow the *CSP* to search for query phrases. Indexing both single and compound words without considering words offset position helps avoiding attacks such as known plaintext or statistical attacks.

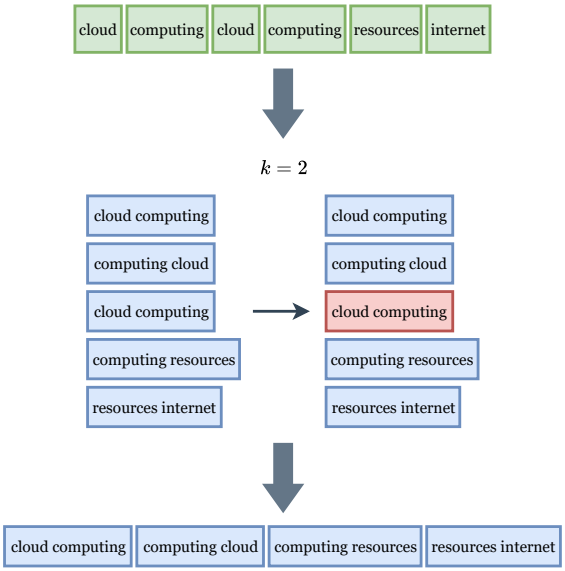


Figure 4.8: Compound words generation.

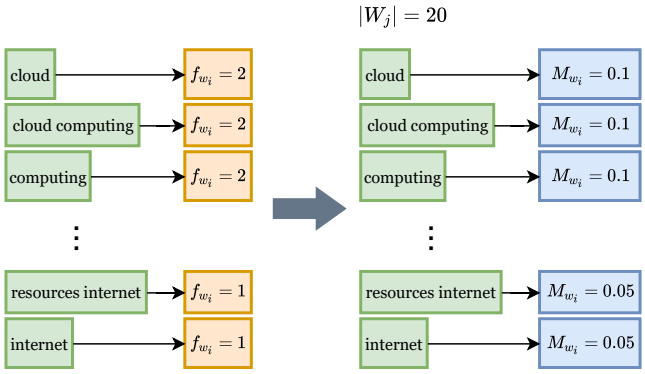


Figure 4.9: Index words and its membership values.

The words membership M_{w_i} for each w_i in the identified dictionary is also computed in this stage. It is used to measure the semantic similarity of each index word w_i and a document d_j. In other words, it is helpful to determine how well w_i represents d_j's contents. M_{w_i} is computed by Equation 4.1 given w_i's frequency f_{w_i} and the total number of the index words $|W_j|$ derived from d_j's word dictionary W_j.

$$M_{w_i} = \frac{f_{w_i}}{|W_j|} \tag{4.1}$$

As a result, this module outputs the single and compound words and its respective membership values that will be used to build the secure index, as it is shown in Figure 4.9.

4.3.2 FABECS.Index

Once the index words W_j for a document d_j have been identified, the data owner generates its encrypted version by applying *CP-ABSE.EncInd* algorithm to get the tuple CT_{w_i}. This algorithm uses the system public key PK and an access structure $\mathbb{A}$ defined by *DO*. It consists in the mapping of each index word w_i to an element in the group $\mathbb{Z}_r^*$ to later compute a pairing multiplication. Also, each attribute in $\mathbb{A}$ is protected through an exponentiation in the multiplicative group $\mathbb{G}_T$. The access structure defined over the index words allows to control who can later search for a particular word from this word dictionary.

Because of the cloud server is not completely trusted and in order to provide full confidentiality to *DO*, each index word, either single or compound, is mapped to a fixed length value by applying a hash function as shown in Figure 4.10. For each index word, the value obtained from the application of a hash function serves as the key for this word in the secure index. Each key $h(w_i)$ in the secure index is associated to a word membership M_{w_i}, an encrypted word CT_{w_i}, and the path of the documents containing such word $path_{d_j}$. Algorithm 11 shows the complete process before the secure index creation. It outputs the words dictionary for each owner's document, and receives as input each d_j, the maximum number of single words in a compound word k, the stop words list $stopwords$, and the pre-defined access structure.

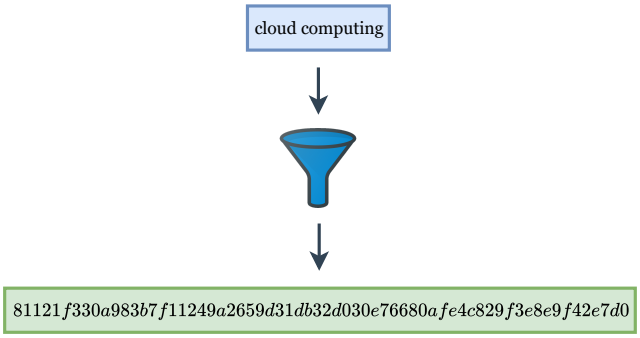


Figure 4.10: Key derivation of an index word.

Finally, *DO* uploads both the secure full-text index and the encrypted files collection to the cloud storage service as stated in Figure 4.11.

Algorithm 11 Words dictionary generation for a data owner's document.

Require: d_j, k, $stopwords$, $\mathbb{A}$
Ensure: W_j

Extract the whole document's contents $tokens$.
for each token t in $tokens$ **do**
  if t is a stopword $s \in stopwords$ **then**
    continue
  else
    Add t to words list W.
  end if
end for

for each word w_i in W **do**
  Compute the hash $h(w_i) = \text{SHA-256}(w_i)$.
  if $h(w_i)$ is already in W_j **then**
    Update index word frequency f_{w_i}.
  else
    Compute $CT_{w_i} = \text{CP-ABSE.EncInd}(P_\lambda, PK, w_i, \mathbb{A})$.
    Add $\{h(w_i), CT_{w_i}\}$ to W_j.
  end if

  for $j = 2$ to k **do**
    Set compound word $w_c = \text{append}(w_i, w_{i+j})$.
    Compute the hash $h(w_c) = \text{SHA-256}(w_c)$.
    if $h(w_c)$ is already in W_j **then**
      Update compound index word frequency.
    else
      Compute $CT_{w_c} = \text{CP-ABSE.EncInd}(P_\lambda, PK, w_c, \mathbb{A})$.
      Add $\{h(w_c), CT_{w_c}\}$ to W_j.
    end if
  end for
end for

Compute the total number of index words $|W_j|$.
for each index word $h(w_i)$ in W_j **do**
  Compute index word membership value $M_{w_i} = \frac{f_{w_i}}{|W_j|}$.
  Update $h(w_i)$ membership in W_j.
end for

return W_j

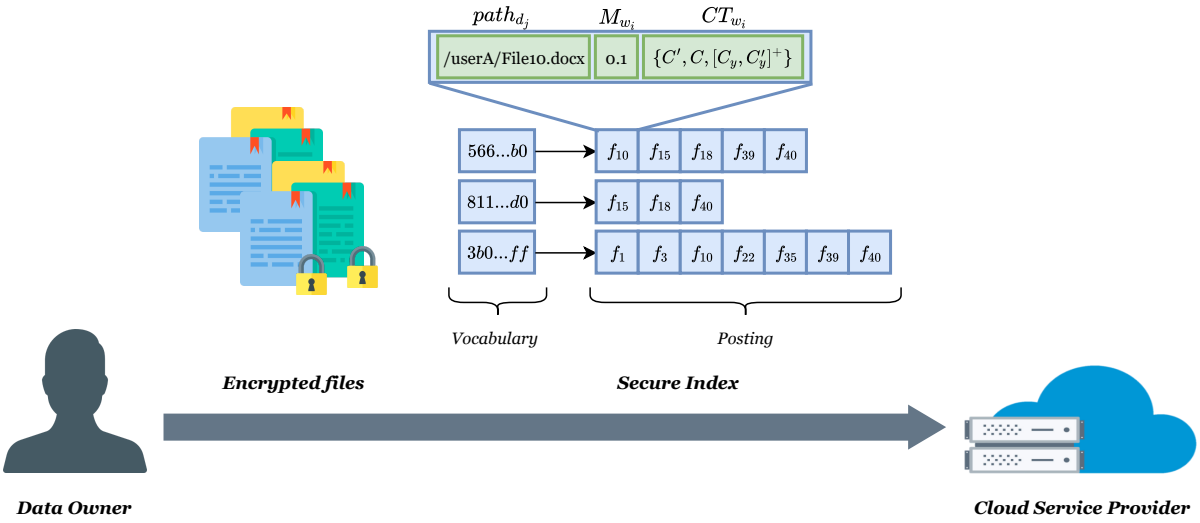


Figure 4.11: *DO*'s encrypted files upload.

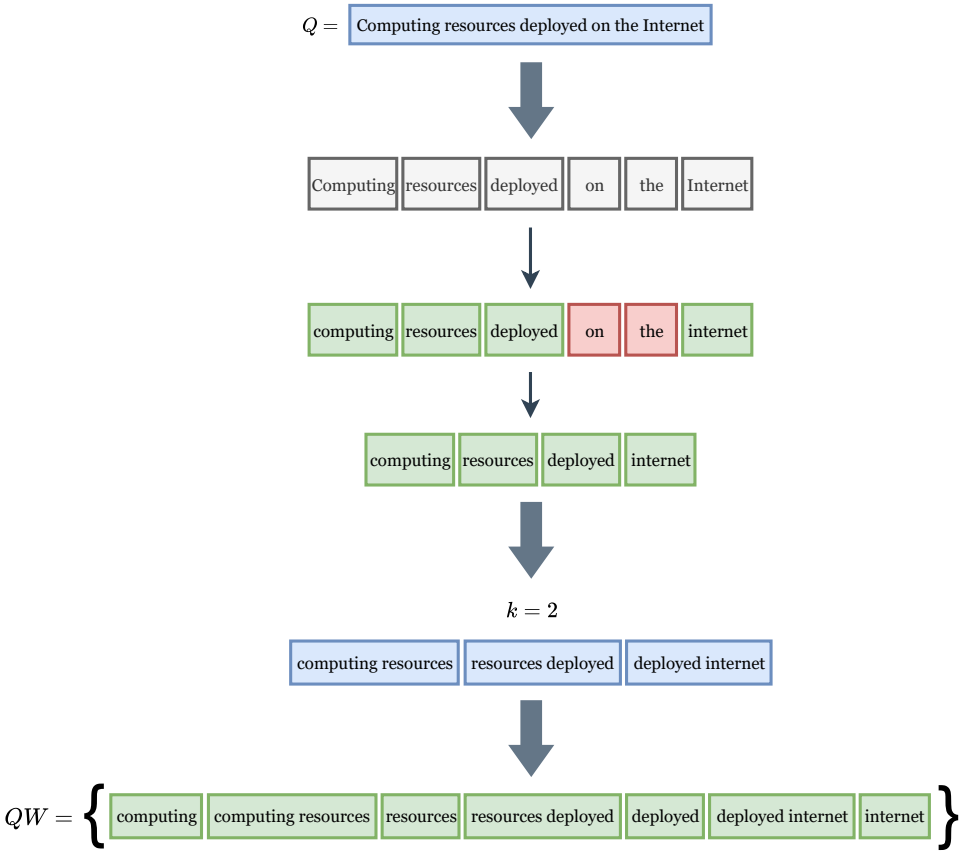


Figure 4.12: Generation of the set of query compound words.

4.3.3 FABECS.Query

When data users have an information need and want to search over *DO*'s documents, they generate the set of query compound words $QW = \{w_{q_1}, w_{q_2}, ..., w_{q_r}\}$ based on original keywords $K = \{w_{s_1}, w_{s_2}, ..., w_{s_n}\}$ given as a query on a traditional search engine application. As described in Algorithm 12, the query compound words are obtained in the same way a data owner gets the index words before data outsourcing. On *DU*'s side, the search query Q is pre-processed in order to eliminate stop words and get the set of tokens K. The single words w_s in K are used to form query compound words w_c of length at most k. Both w_s and w_c form the set QW, for which the corresponding trapdoors T_ω will be later computed. This process is shown in Figure 4.12.

Then, the similarity between every query compound word and the original user query is measured. In such approach, each $w_{s_i} \in K$ has a ranking score weight $S_{w_s} = 1$, and for each $w_c \in QW$ its ranking score weight S_{w_c} is calculated by using Equation 4.2. Also, the membership entropy E_{w_c} is computed for each query compound word by means of Equation 4.3. The value of E_{w_c} is used to measure the membership approximation of a long compound word and its partial words. That is, it allows the evaluation of how much information is embedded on every w_c. Both the ranking weight and the membership entropy will later help the *CSP* to set a score for each found result for T_ω.

$$S_{w_c} = \sum_{w_s \in w_c} S_{w_s} \times e^{\frac{n_1}{|w_c|}-1} \tag{4.2}$$

$$E_{w_c} = -\sum_{w_s \in w_c} \log_2 \left(\frac{S_{w_s}}{\sum_{w_s \in w_c} S_{w_s}} \right) \tag{4.3}$$

Using SK_u, *DU* computes the query trapdoors $T_u(w_q)$ for each query word identified in previous steps. Each $T_u(w_q)$ is generated by using the *CP-ABSE.TrpDr* algorithm, and it consists in mapping the query word w_q to an element in the group $\mathbb{Z}_r^*$ and its later multiplication with the elements D and D' from SK_u. The values $[d_y, d'_y]^+$ or just $[d_y]^+$ related to the attributes the user possess are taken from SK_u and stored as part of the query trapdoor. Finally, a hash function is applied to the

Algorithm 12 Query compound words generation.

Require: Q, k, $stopwords$
Ensure: QW

Extract all the query keywords $tokens$.
for each token t in $tokens$ **do**
  if t is a stopword $s \in stopwords$ **then**
    continue
  else
    Add t to words list K.
  end if
end for

for each word w_s in K **do**
  Add w_s to QW.
  for $j = 2$ to k **do**
    Set compound word $w_c = \text{append}(w_s, w_{s_{i+j}})$.
    Add w_c to QW.
  end for
end for

for each w_q in QW **do**
  if w_q is in K **then**
    Set $S_{w_q} = 1$.
  else
    Compute compound word ranking score weight

$$S_{w_q} = \sum_{\forall w_s \in w_q} S_{w_s} \times e^{\frac{n_1}{|w_q|}-1}$$

    Compute compound word membership entropy

$$E_{w_q} = -\sum_{w_s \in w_q} \log_2 \left(\frac{S_{w_s}}{\sum_{w_s \in w_q} S_{w_s}} \right)$$

  end if
  Update w_q ranking score weight S_{w_q} in QW.
  Create query trapdoor $T_u(w_q) = \text{CP-ABSE.TrpDr}(SK_u, w_q)$.
end for

return QW

query word to get the key $h(w_q)$ used for its identification in the secure index stored by the *CSP*.

Each $h(w_q)$ in the trapdoors set T_ω is associated to a tuple $T_u(w_q)$, a word ranking score S_{w_q}, and a word membership entropy E_{w_q}. As it is shown in Figure 4.13, the trapdoors set T_ω is sent by *DU* to the cloud storage service provider to request for searching.

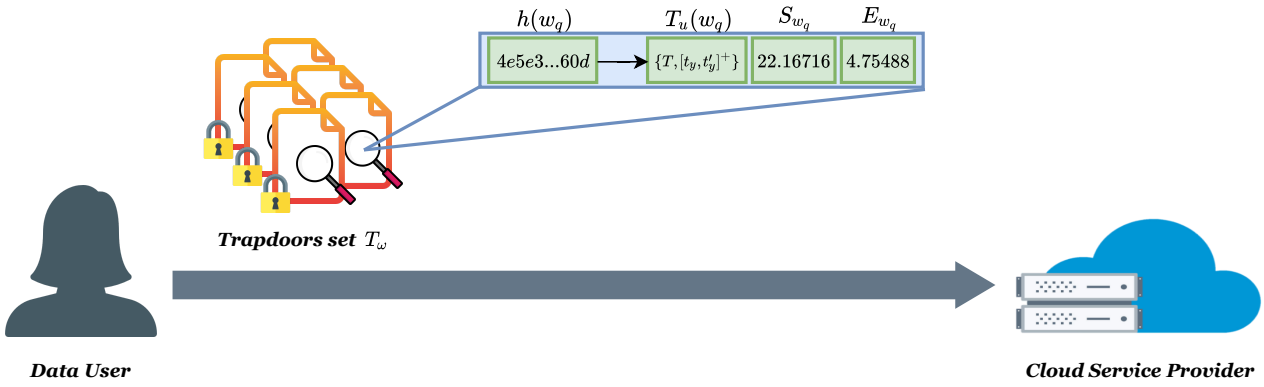


Figure 4.13: *DU*'s search request.

When the *CSP* receives T_ω, it looks for each query word key $h(w_q) \in T_\omega$ in the secure searchable index. If a match is found, the algorithm *CP-ABSE.Search* is executed to validate if the data user has the attributes required to access the documents containing the query word w_q. The elements in $T_u(w_q)$ are used to compute the value C_T that should be compared against C', which is stored in the secure index. $C_T \equiv C'$ only if S_u (embedded in SK_u, used to create the trapdoor) satisfies the access structure used to encrypt the keyword in the index being tested, and if the keyword in the query is the same as the keyword in the index. Otherwise, the service provider notifies the user that no results have been found for the given query.

4.3.4 FABECS.Rank

On the service provider's side, the found results D_{rslt} after a successful search process are ranked according to its relevance as depicted in Figure 4.14. To do so, the *CSP* computes each $d_j \in D_{rslt}$ ranking score S_{d_j} by using Equation 4.4 and based on the pre-calculated M_{w_q}, S_{w_q} and E_{w_q} values of the query words in QW. As the sum of the membership entropies of the query compound words

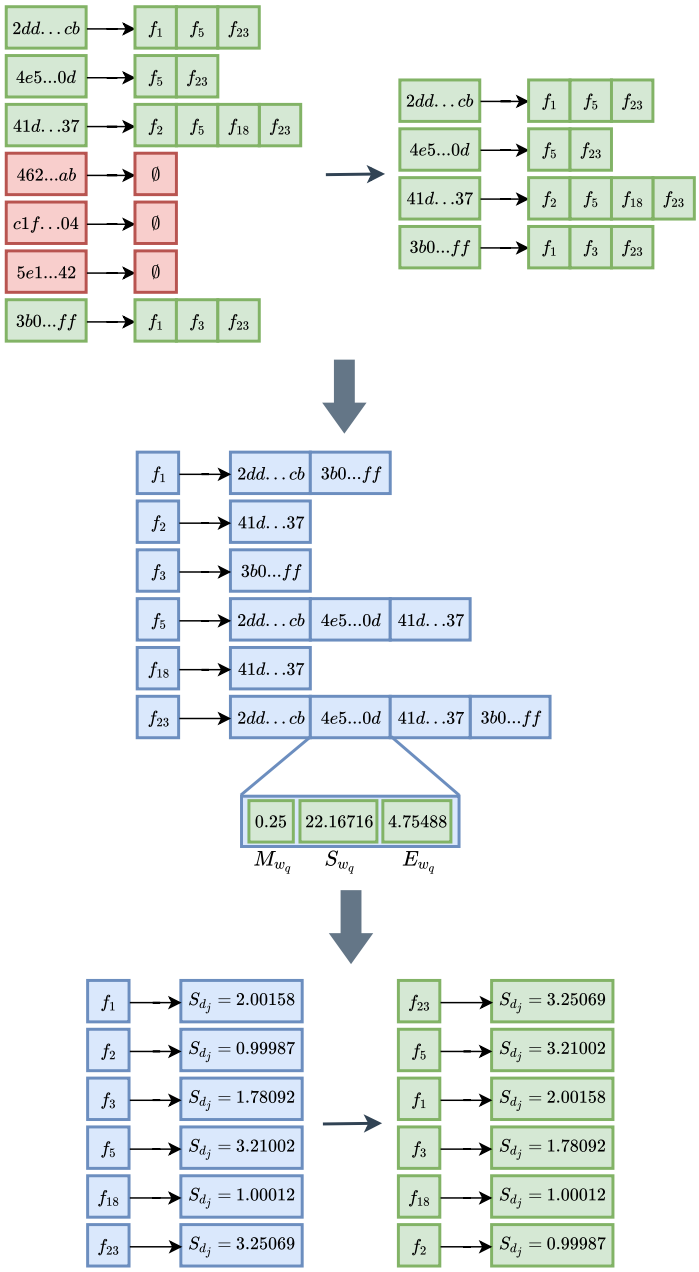


Figure 4.14: Results ranking process.

$\sum_{\forall w_c \in QW} E_{w_c}$ is a constant value, it is calculated before executing the ranking algorithm. The results obtained from the secure index are rearranged in order to facilitate the computation of the ranking score for each resulting document. All the matching query words for each d_j are merged into a list along with its membership and weight values.

Algorithm 13 Query results ranking.
Require: α, β, T_ω, $rslt$
Ensure: $Rrslt$

1: **for** each result r_i in $rslt$ **do**
2: Compute score $\sum s_{w_q} \times M_{w_q}$ for each matching trapdoor $T_u(w_q)$ in T_ω.
3: Compute document's ranking score

$$S_{d_j} = \alpha \times \sum (s_{w_q} \times M_{w_q}) - \beta \times \sum_{\forall w_q \in QW} E_{w_c}$$

4: Add r_i and S_{d_j} to scored results list $Rrslt$.
5: **end for**
6: **return** $Rrslt$

When all the documents' ranking scores are computed, the *CSP* ranks the results in descending order according to Algorithm 13 and retrieves the ranked results list $Rrslt$ to the data user. Finally, *DU* downloads the retrieved documents D_{rslt} and decrypts them with *DET-ABE.decrypt* to access the information in plaintext form.

$$S_{d_j} = \alpha \times \sum_{\forall w_c \in succ(d_i, w_c)} (S_{w_c} \times M_{w_c}) - \beta \times \sum_{\forall w_c \in QW} E_{w_c}, \tag{4.4}$$

$$(\alpha + \beta = 1; \alpha, \beta \geq 0)$$

4.4 Chapter Summary

This chapter described in Section 4.1 the system model considered in the FABECS architecture and in Sections 4.2 and 4.3 the algorithms involved on its development. For a greater transparency in the explanation, the description of all the operations involved in the proposal deployment was divided into two main parts: the data encryption scheme and the retrieval scheme over the encrypted data. The former is focused on the security insight, addressed by means of the implementation of both ABE and ABSE as the core of the solution, describing the details of each process and where it take place. The second one, from the information retrieval viewpoint, incorporates the full-text

retrieval technique and uses an inverted index structure with the aim of ensuring the operations efficiency. Such insight was also split into four parts from the operations needed for the index words identification, the construction of the secure index to the execution of the search over the secure searchable index and the results ranking. The next chapter describes the experiments done for the assessment of both the system performance and the retrieval effectiveness, as well as the analysis of the results obtained.

5

Experiments and Results

This chapter describes the experimental settings, methods and materials used for the evaluation of FABECS. The experimental evaluation considers the security level provided, the system performance and the information retrieval effectiveness as the metrics to be assessed. The results obtained from each experiment are analyzed and compared against representative works in the literature.

5.1 Methods and Materials

In order to carry out the experimental evaluation of the proposal described in Chapter 4, the required methods and materials are exposed in this section. Broadly speaking, the proposal could be divided in two main components: the first one is responsible for providing confidentiality and access control while the second one is in charge of the indexing and information retrieval tasks. The information security component contains modules for system's parameters initialization, keys issuing, index words encryption, and the owners' files and symmetric keys encryption management, as well as for files decryption on the side of authorized users. On the other hand, the information retrieval component

consists of three modules for encrypted words indexing, searching over a secure index, ranking and retrieval.

Table 5.1: Technological infrastructure features.

Features		Device	
		Laptop	Desktop computer
Processor	Model	Intel Core i7 7th.Gen.	Intel Core 2 Duo
	Base frequency	2.70 GHz	3.06 GHz
	Cores	2 physical, 4 logical	2 physical
	Cache	4 MB	3 MB
	Architecture	64-bit	64-bit
Graphics	Model	Intel HD Graphics 620	AMD ATI Radeon HD 4670
	Base frequency	300 MHz	675 MHz
	Display memory	128 MB	256 MB VRAM
System memory	RAM	16 GB	16 GB
	Speed	2133 MHz	1067 MHz
	Type	DDR4-2133 SDRAM	DDR3 SDRAM
Storage	Type	HDD	HDD
	Capacity	1 TB	1 TB
	RPM	7200 rpm	7200 rpm
Operating system		Windows 10	macOS High Sierra

Moreover, there are used some pre-defined test data to evaluate the efficiency of FABECS. Such test data consist in plaintext files, access control policies under different settings to encrypt the plaintext files, several user's attributes sets to generate different configurations for the secret keys, benchmark datasets to perform the quality assessment of the information retrieval, and data queries which reflect multiple user information needs. The specific details of the test data are further described as part of the particular settings for each experiment. Regarding the software and development tools used for the solution proposal deployment, the jPBC library was used for implementing the CP-ABSE engine, which relies on pairing-based cryptography. The DET-ABE API

was also used, which provides an implementation of the digital envelope technique using AES for data confidentiality and CP-ABE for fine-grained access control. The technological infrastructure used for development and experimental evaluation includes a laptop, with the role of *DO* and/or *DU*, and a desktop computer, responsible of the *CSP*'s tasks, which main features are shown in Table 5.1.

5.2 Experimental Evaluation

This section describes the experiments carried out to evaluate the security scheme called FABECS proposed in this thesis as an enabler for secure storage, sharing and retrieval of encrypted data in the cloud. The experimental evaluation involves the definition of the experiments needed to accomplish such purpose, the objectives each one pursues, as well as a brief discussion of the results obtained. All the experiments were executed at least 31 times according to the Central Limit Theorem (CLT) [35], which establishes that as a sample size increases, the mean sampling distribution tends to approach to a normal distribution, especially for samples sizes greater than or equal to thirty elements. In this context, and because there could be several factors that might influence the obtained results (e.g. background processes or other applications execution, etc.), the experimental evaluation was done based on the CLT in order to get more reliable results.

5.2.1 Experiment 1: Security Level Provided

This experiment has the purpose of quantifying the impact of deploying FABECS for supporting different security levels λ on the system's response time, specifically for the operations of keys generation, data encryption and data decryption. That is, to determine the system behavior when creating and using keys of different sizes, as well as how the type of elliptic curve used to generate the asymmetric pairing affects the provision of higher security levels. The impact of the security level being used is most evident in the keys generation operations, which involves the creation of

Table 5.2: Experiment 1 settings.

Component	λ	No. attributes
$\{PK, MK\}$	382-bit	-
	462-bit	-
	638-bit	-
SK_u	382-bit	10, 20, 50
	462-bit	10, 20, 50
	638-bit	10, 20, 50
k	128-bit	-
	192-bit	-
	256-bit	-

the system keys $\{PK, MK\}$, the data users' secret keys SK_u and the AES symmetric keys k. The experiment settings are described in Table 5.2.

Asymmetric pairings in FABECS were created using the available tools in jPBC library for Barreto-Naehrig curves (BN curves). With the recent advances for computing discrete logarithms, the key sizes for the recommended security levels were updated. Previously, a BN256 curve was believed to provide a security level of 128-bit. Nonetheless, after the update that curve was reduced just to λ =100-bit. After exTNFS[1] attack, in [30] it was estimated the minimum groups size of BN curves as 383-bit (BN384 curve) for 128-bits of security. However, in [4] the recommendation is to use a BN curve with groups size of 462-bit (BN464 curve) to achieve λ =128-bit. Therefore, the values of Table 1.1, shown in Section 1, were modified resulting in Table 5.3.

In order to accomplish a 128-bit security level we considered both the BN384 and BN464 curves for FABECS experimental evaluation. Additionally, we included the BN640 curve, which has a greater order and is expected to provide a security level greater than 128-bit, but less than 192-bit. For this reason, in Table 5.2 are shown two different sets of values for λ: the one related to CP-ABE operations, 382-bit, 462-bit and 638-bit (corresponding to the size of $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ in the

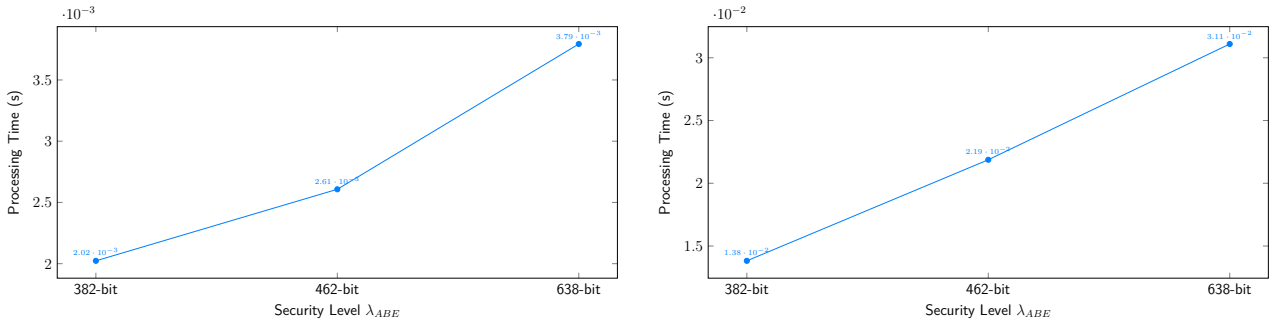
[1] Extended Tower Number Field Sieve

asymmetric pairing), and the one corresponding to the AES symmetric keys.

Table 5.3: Updated key sizes for the recommended security levels.

λ	Validity Period	Symmetric Algorithm	Pairing Settings				BN Curve
			$\mathbb{G}_1$	$\mathbb{G}_2$	$\mathbb{G}_T$	$\mathbb{Z}_r$	
80-bit	Outdated	AES 128-bit	320-bit	640-bit	1920-bit	160-bit	-
112-bit	2016 – 2030	AES 128-bit	448-bit	896-bit	2688-bit	224-bit	-
128-bit	2030 – 2040	AES 128-bit	512-bit	1024-bit	3072-bit	256-bit	BN256
128a-bit	-	-	768-bit	1536-bit	4608-bit	384-bit	BN384
128b-bit	-	-	928-bit	1856-bit	5568-bit	464-bit	BN464
>128-bit	>2030	AES 192-bit	1280-bit	2560-bit	7680-bit	640-bit	BN640

As stated before, the response time is affected by the security level being used. In order to determine the costs associated with the deployment of FABECS for a given recommended security level, the time to generate the private and public keys, the users' secret keys, and the AES encryption keys under the different settings was measured and the results obtained are presented as follows. As depicted in Figure 5.1(a), for the W11 construction, the time required to generate the system public and private keys, $\{PK, MK\}$, is less than 0.004 seconds for each λ (around 4 milliseconds). Generating a pair $\{PK, MK\}$ for λ =382-bit takes 2.02 milliseconds, which represents the lower time needed to generate the system key pair. For λ =462-bit it takes 2.61 milliseconds, and when λ =638-bit the time required is about 3.79 milliseconds. On the other hand, the system keys generation under the BSW07 approach takes at least 13.82 milliseconds, when using 382-bit as security level, and a maximum of 31.1 milliseconds, when λ =638-bit. As shown in Figure 5.1(b), the response time for the BSW07 system keys generation is almost the double of the previous result for both 462-bit and 638-bit security levels, unlike the results achieved with W11. However, the range between the lowest and highest achieved values in the W11 construction is barely 1.77 milliseconds, whereas for BSW07 the difference in the results of λ =382-bit and λ =638-bit reaches 17.28 milliseconds.



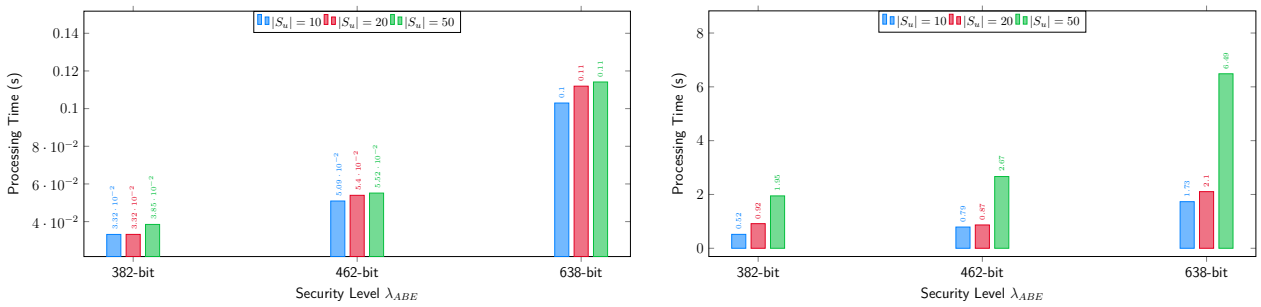
(a) W11. (b) BSW07.

Figure 5.1: System keys generation.

The time required to generate the users' secret keys was measured taking into account different amounts of user attributes since the key generation algorithm also depends on that parameter. Broadly speaking, it takes less time to create a SK_u for lower security levels and for a smaller number of attributes related to the key. Nonetheless, as shown in Figure 5.2(a), under the W11 approach the number of attributes in S_u does not have a significant impact in the keys generation response time, which is mainly affected by the security level being used. Such behavior is due to the fact that the operations complexity of the users' secret keys generation relies in the computation of D and D' instead of the computation of $[d_y]^+$, which depends on the number of attributes the user possess. The results range for the first security level is about 5.31 milliseconds, 4.21 milliseconds for λ =462-bit, and 11.1 milliseconds for λ =638-bit. The difference between the highest values of each security level is 16.65 milliseconds, in the case of 382-bit and 462-bit, and 58.94 milliseconds for 462-bit and 638-bit. From these results, it is stated that the use of higher security levels has a preponderant influence in the system response time for the SK_u generation.

Figure 5.2(b) shows the results obtained from the BSW07 construction under the pre-defined attributes configuration. As with W11, there is a considerable complexity when computing the elements D and D', which are independent from the number of attributes a user owns. However, in this case there is also an extra cost related to the computation of the tuple $[d_y, d'_y]^+$ dependent on the number of attributes in S_u. The additional complexity is mainly associated to the computation

of two exponentiations and one multiplication in the underlying group. Therefore, the response time for the secret keys generation in the BSW07 approach is affected by both the security level used and the number of attributes related to S_u. As in the case of W11 construction, the lower response times are obtained with λ =382-bit when considering ten attributes while the highest results are obtained for λ =638-bit and fifty attributes. But the results range for the 382-bit security level is close to 1.43 seconds, 1.9 seconds for 462-bit, and up to 4.75 seconds for λ =638-bit. In other words, the response times of BSW07 are clearly higher than those of W11.



(a) W11. (b) BSW07.

Figure 5.2: Users' secret keys generation.

This experiment concludes with the evaluation of the symmetric encryption keys generation time. As shown in Figure 5.3, the time needed to create the AES encryption keys is very low compared with the time for keys generation in CP-ABSE and DET-ABE. For the three security levels, a symmetric key is generated between 8.37 and 13.26 microseconds. The lower the security level used the lower time needed to generate the encryption keys, the same behavior observed when creating the system public and private keys.

For both system keys and AES keys generation, the impact of deploying each security level in the system's response time is very low, since it took only a few milliseconds, at most, to create the system's public and private keys and the symmetric keys, respectively. Nonetheless, in the case of the user's secret keys generation in W11 approach, even when the number of attributes that each user owns may affect the system's response times, the deployment of different security levels entails the

substantial differences. The opposite occurs with the BSW07 construction, where both the amount of attributes and the security level used greatly determine the time required to generate the user's secret keys.

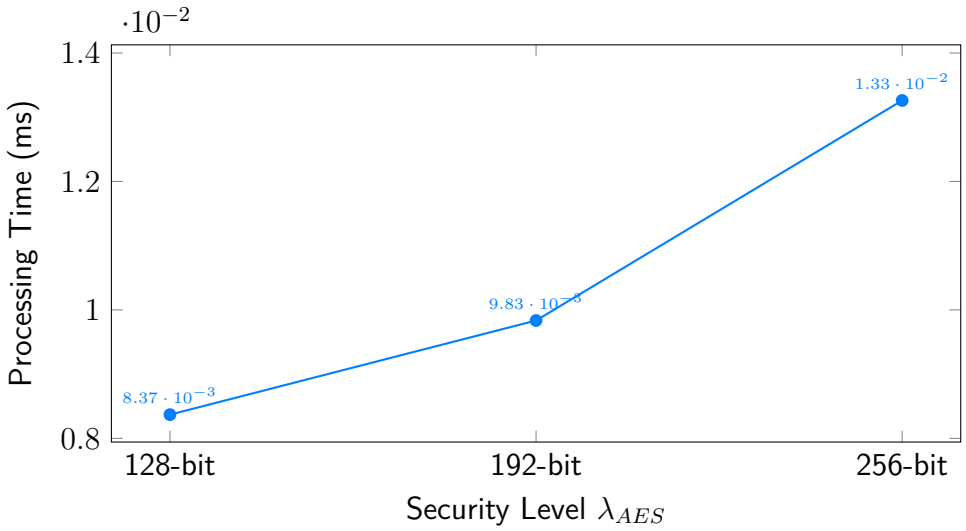


Figure 5.3: AES keys generation.

5.2.2 Experiment 2: Scheme's Efficiency

In this experiment, the objective is to demonstrate the feasibility of the scheme implementation in practical cases, which is predominantly influenced by the response times of each operation requested by the user. Several configurations were applied on different datasets, allowing the evaluation of the prototype's behavior under different circumstances and perspectives. It was evaluated the response time for the creation of a digital envelope, which consists in the encryption of different amounts of files and files of different size, using different key sizes. Regarding the words indexing process, it was measured the time needed for the encryption of different amounts of keywords using access policies with different amount of attributes. Finally, for the information retrieval process, it was measured the response time for the query trapdoors T_ω generation, the user's access permission validation as well as the retrieved results decryption. The experiment settings are described in Table 5.4.

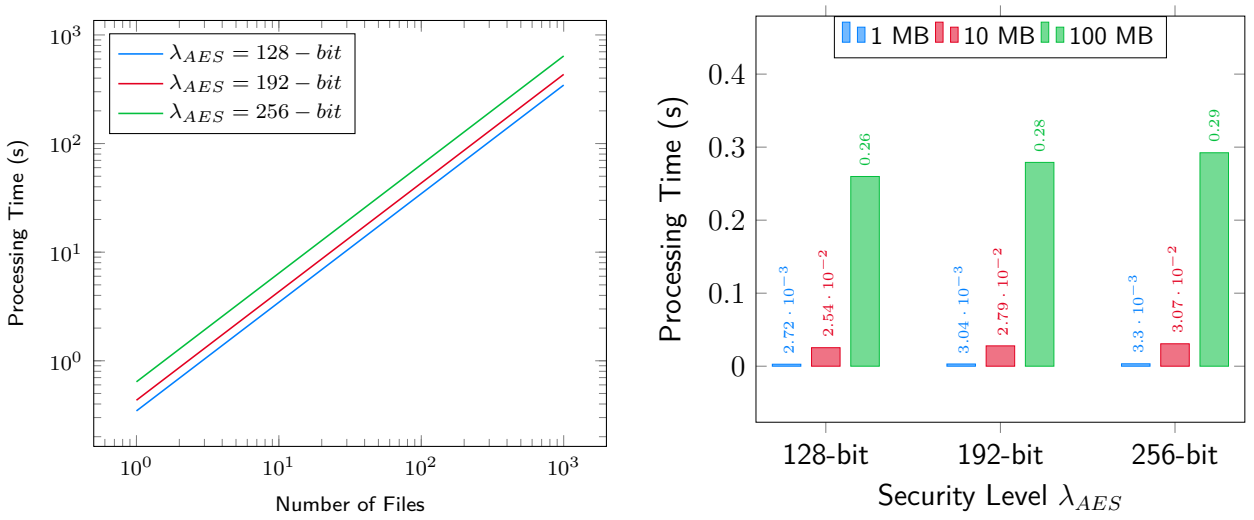
Table 5.4: Experiment 2 settings.

Module	Component	λ	No. attributes	No. files	Files size
DET	$Enc(D)$	128-bit	-	1 – 1000	1MB, 10MB, 100MB
		192-bit	-	1 – 1000	1MB, 10MB, 100MB
		256-bit	-	1 – 1000	1MB, 10MB, 100MB
	$Enc(K)$	382-bit	5, 10, 15	-	-
		462-bit	5, 10, 15	-	-
		638-bit	5, 10, 15	-	-
$WordsInd$	$Enc(W_j)$	382-bit	5, 10, 15	-	-
		462-bit	5, 10, 15	-	-
		638-bit	5, 10, 15	-	-
$Query$	T_ω	382-bit	5, 10, 15	-	-
		462-bit	5, 10, 15	-	-
		638-bit	5, 10, 15	-	-
	$Search(T_\omega)$	382-bit	5, 10, 15	-	-
		462-bit	5, 10, 15	-	-
		638-bit	5, 10, 15	-	-
$RsltsRet$	$Dec(K)$	382-bit	5, 10, 15	-	-
		462-bit	5, 10, 15	-	-
		638-bit	5, 10, 15	-	-
	$Dec(D_{rslt})$	128-bit	-	1 – 1000	1MB, 10MB, 100MB
		192-bit	-	1 – 1000	1MB, 10MB, 100MB
		256-bit	-	1 – 1000	1MB, 10MB, 100MB

5.2.2.1 DET

In order to evaluate the response time for the creation of a digital envelope, it was measured the time needed to encrypt files with different sizes, along with the required time to encrypt the symmetric key for data encryption using different access structures configurations. The first part of this evaluation consists in the files encryption by using the AES symmetric cipher. The performance of AES was evaluated in two forms. In the first one, data encryption was done over a batch of n files, varying the value of n, where each file has a size of 5 MB. The results of this setting are shown in Figure 5.4(a). This results reveal the linear complexity of the cipher regarding the amount of the input data, the

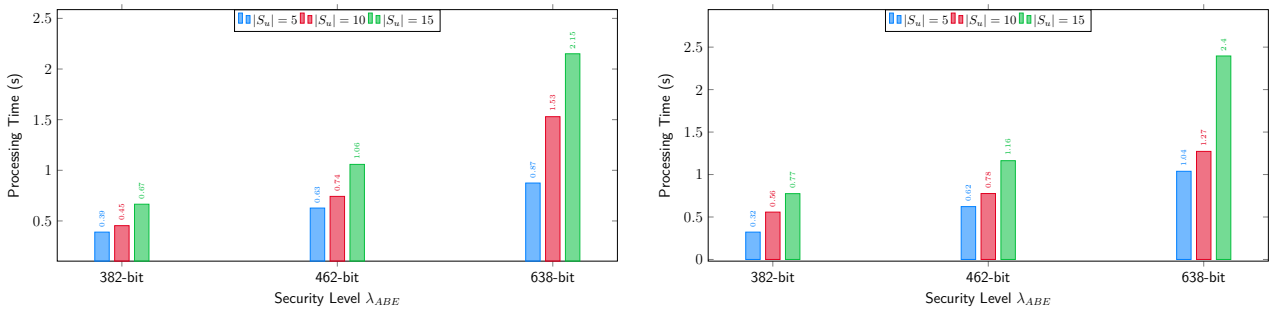
same behavior for the three AES security levels. In the second test, the AES cipher was evaluated over fixed size data for the three security levels. The results of this test are shown in Figure 5.4(b). In this case, the results reveal that there is a relatively low extra cost by using greater security levels, but what really affects the response time is the data size.



(a) By files amount.

(b) By files size.

Figure 5.4: Files encryption.



(a) W11.

(b) BSW07.

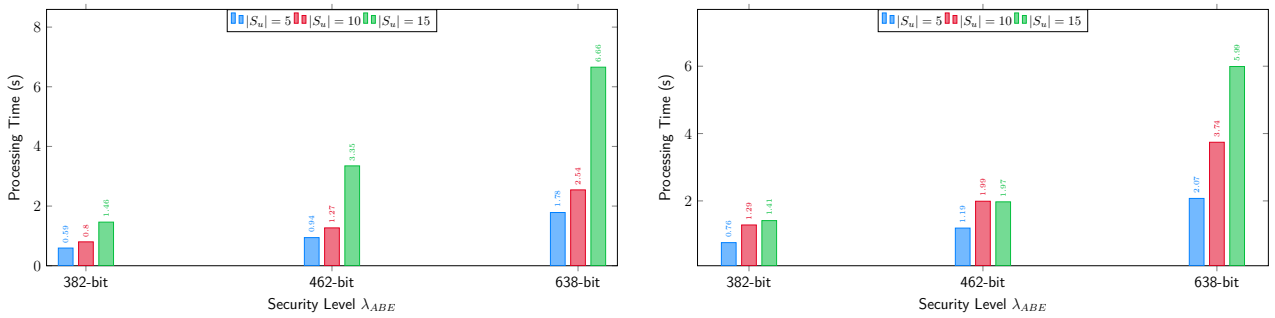
Figure 5.5: AES keys encryption.

The next part of the digital envelope creation involves encrypting the symmetric key by using CP-ABE with an access structure with different amount of attributes associated. Figure 5.5 depicts the

system response times for the AES keys encryption for both W11 and BSW07 approaches (subfigures 5.5(a) and 5.5(b), respectively) using five, ten and fifteen attributes. As already observed in previous results, using a stronger security level requires more time to perform the keys encryption. That is, the lower encryption times are obtained when using λ =382-bit and the higher encryption times when using λ =638-bit. The results achieved by both constructions are quite similar since there is a minimum difference of 10 milliseconds, when using five attributes with λ =462-bit, and a maximum difference of 260 milliseconds, when using ten attributes and 638-bit as security level. Besides, in all cases increasing the number of attributes does have a significant impact in the keys encryption time. Particularly in the case of W11, such behavior is the opposite of those results obtained when creating the users' secret keys, where the number of attributes a user possess has a very little influence on the keys generation time.

5.2.2.2 WordsInd

In this case, it is evaluated the time required to encrypt the index keywords, identified from the contents of the owners' files. Such evaluation considers five, ten and fifteen attributes in the access structure used to encrypt a keyword with CP-ABSE, for each of the three ABE security levels. As expected, using a smaller number of attributes associated to the access structure $\mathbb{A}$ and lower security levels results in less time to perform the keyword's encryption. Figure 5.6(a) shows the results obtained for each configuration to encrypt a keyword, either single or compound, with CP-ABSE W11. As it can be noted at each security level, there is relatively small difference between the usage of five and ten attributes (with a maximum of 0.7583 seconds), while there is a substantial difference between using ten and fifteen attributes, where the results difference runs between 0.66141 and 4.1136 seconds. Figure 5.6(b) shows the results achieved by CP-ABSE BSW07 under the same settings. In this case, when λ =382-bit and λ =462-bit there is a maximum difference of 18.1 milliseconds between encrypting a keyword with an access structure that considers ten or fifteen attributes. Meanwhile, when λ =638-bit such difference increases up to 2.24 seconds.



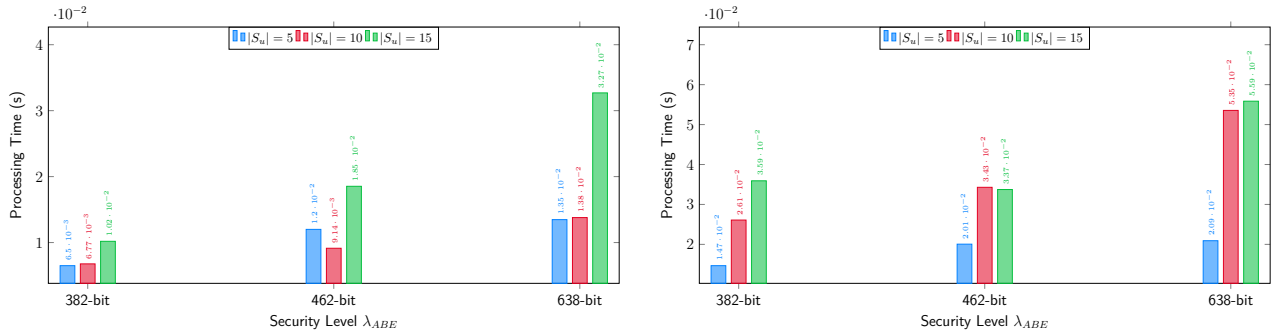
(a) W11. (b) BSW07.

Figure 5.6: Index keywords encryption.

5.2.2.3 Query

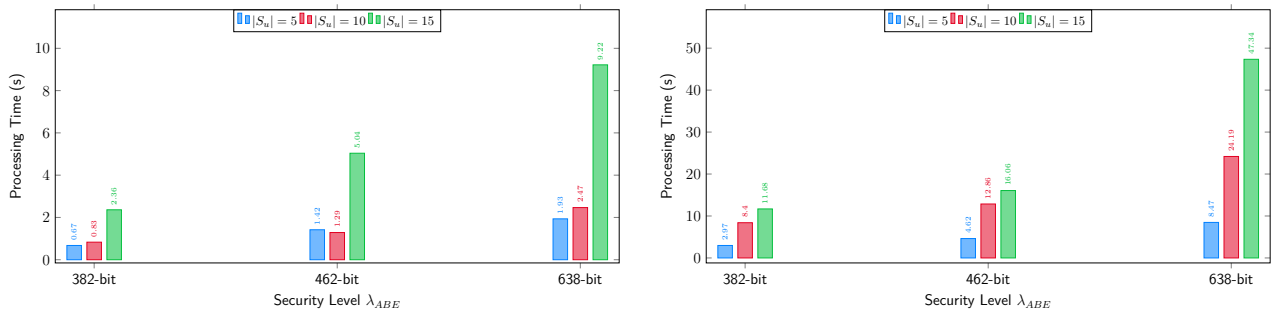
As a part of the information retrieval process, the time needed to create the query trapdoors was measured using different attributes settings and a original query keyword. Figure 5.7(a) shows the response times comparison for each λ_{ABE} in the W11 approach. Only except for the case of five attributes when λ =462-bit, the lower the security level is and the less amount of attributes is used, the lower the time required to create the query trapdoors. The behavior observed in this experiment is similar to the index keywords encryption: there is a minimum difference between the results obtained when using five and ten attributes, but when using ten and fifteen the difference between them tends to increase in a few milliseconds (up to 18.91 milliseconds). On the other hand, Figure 5.7(b) presents the results for the trapdoors generation with CP-ABSE BSW07. As in the W11 construction, the response times do not exceed a second, but the lower differences occur between the results obtained when using ten and fifteen attributes, particularly in the cases of λ =462-bit and λ =638-bit, instead of when considering five and ten attributes for all security levels.

Figure 5.8(a) shows the results obtained when searching an encrypted keyword, either single or compound, with CP-ABSE W11 on the *CSP*'s side by means of a query trapdoor sent by a data user. As with the trapdoors generation, for λ =462-bit there is a little increase in the response time considering five attributes in the user's secret key regarding the one achieved when using ten



(a) W11. (b) BSW07.

Figure 5.7: Trapdoors generation.



(a) W11. (b) BSW07.

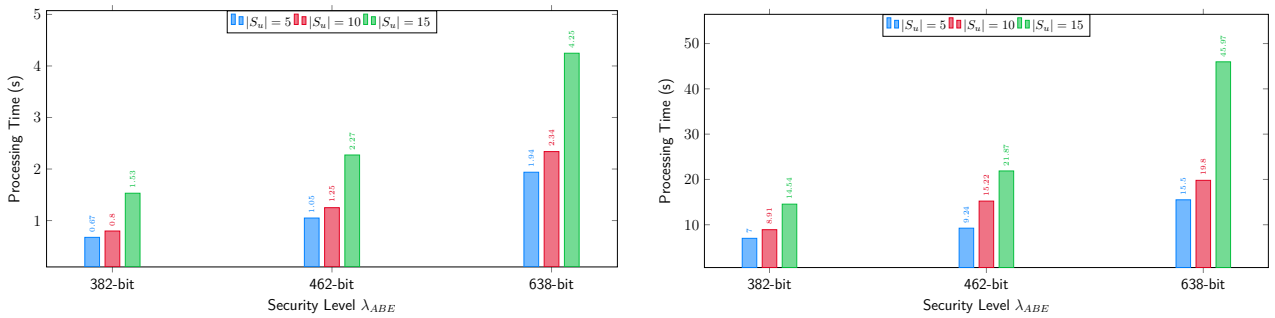
Figure 5.8: Queries search and users' access permission validation.

attributes. For the remaining results, the behavior registered remains the same: higher security levels involve higher response times, and viceversa. Nonetheless, the increase ratio between the results for five and ten attributes is relatively small for all the security levels, unlike the case of fifteen attributes, where the ratio is almost doubled as it is increased the security level used. Unlike the results observed for the index keywords encryption and the trapdoors generation, when searching for a trapdoor with BSW07 construction, in some cases it takes more than five times the amount of time required to perform the same operation than with W11 approach. So, in all cases the response times achieved with BSW07 approach are higher than the ones of CP-ABSE W11. Figure 5.8(b) presents the results obtained for a trapdoor search and the validation of the access permission of a user with CP-ABSE BSW07. As expected, it clearly shows that using higher security levels and more attributes in S_u will

lead to longer response times.

5.2.2.4 RsltsRet

After receiving the results from a given query, those results must be decrypted so that the user can access the information in plaintext. As with the digital envelope creation, this process consists in two steps: first the symmetric key decryption, and then the data decryption with that key. As in the case of the keys encryption, the number of attributes associated to the access structure embedded into the ciphertext has an important effect in the keys decryption response time. In other words, if a reduced number of attributes is considered in $\mathbb{A}$, then less time will be needed to perform the decryption operations. In addition, for both W11 and BSW07 constructions, lower security levels will have also lower response times, as shown in Figure 5.9. However, it is BSW07 approach the one with the highest response times. While the maximum value achieved by W11 construction is around 4.25 seconds (obtained when decrypting a key with fifteen attributes associated to $\mathbb{A}$ and λ =638-bit), for BSW07 construction the minimum response time achieved is of 7 seconds (considering five attributes and λ =382-bit) and the maximum is up to 46 seconds (using fifteen attributes when λ =638-bit).

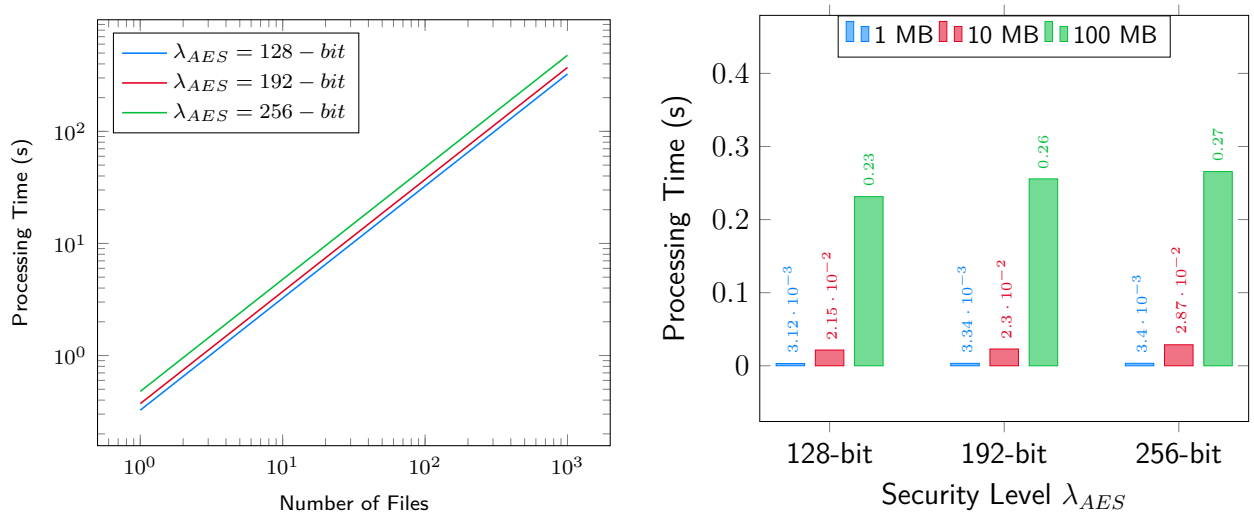


(a) W11. (b) BSW07.

Figure 5.9: AES keys decryption.

Once having the symmetric key, the resulting files are finally decrypted. In Figure 5.10 it is observed that the results obtained are quite similar to those of the encryption process. On the one hand, decryption time is almost identical to that needed for the encryption of the same amount of

files. On the other hand, when decrypting files of sizes 1 MB and 10 MB the response time remains lower than 35 milliseconds, regardless of the security level used. But in the case of decrypting 100 MB files, the system's response times are between 0.231 and 0.266 seconds, opposed to encryption process where the time runs between 0.26 and 0.293 seconds. In fact, only the results of the 1 MB files decryption are higher than the encryption response times, for 10 MB and 100 MB all the results achieved are lower than the encryption results. Nonetheless, the behavior observed remains unchanged since the decryption of smaller size files takes less time than processing large files, no matter which security level is used.



(a) By files amount.

(b) By files size.

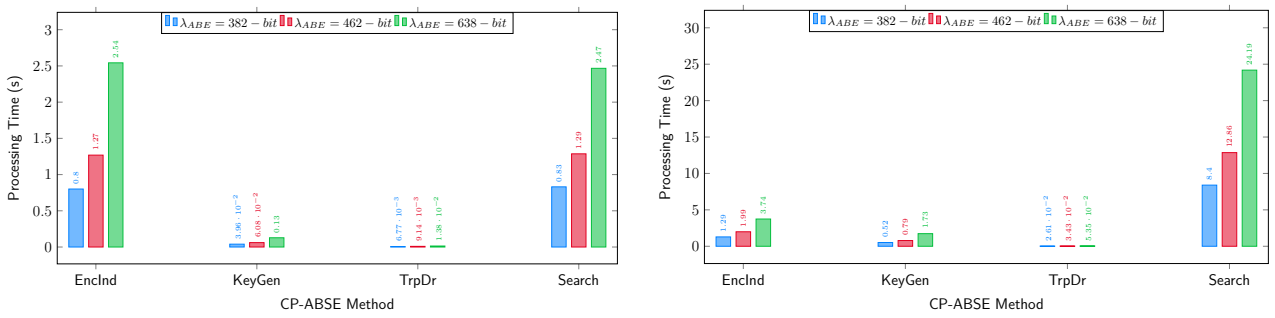
Figure 5.10: Files decryption.

5.2.2.5 CP-ABSE Operations

As described so far, some operations in the different modules analyzed are the most time-consuming. Among those operations are found two of the four methods of CP-ABSE: *CP-ABSE.EncInd* and *CP-ABSE.Search*. Figure 5.11(a) depicts the response times comparison between the four CP-ABSE algorithms of W11 construction on the basis of the achieved results considering ten attributes, which have been presented above. Such setting clearly shows that both *CP-ABSE.EncInd* and

CP-ABSE.Search have similar response times, being the latter the one which requires a little bit more time to finish its execution for security levels 382-bit and 462-bit (about 17.80 and 30.08 milliseconds). Nonetheless, for λ =638-bit *CP-ABSE.EncInd* is the one which requires a few more time to be completed, more precisely 75.80 milliseconds. *CP-ABSE.TrpDr* is the less time-consuming method, since it only requires the computation of T and values $\{T', [t_{y_i}]^+\}$ are just taken from user's SK_u. As already stated, *CP-ABSE.KeyGen* algorithm is not significantly affected by the amount of attributes as it is by the security level used, so its behavior will be almost the same if other scenario is considered.

Almost the same results were obtained with BSW07 construction, as shown in Figure 5.11(b). Nonetheless, in this case even when *CP-ABSE.EncInd* requires higher response times than *CP-ABSE.KeyGen* and *CP-ABSE.TrpDr*, it is *CP-ABSE.Search* the most time-consuming method. When λ =382-bit, it needs more than eight seconds to finish its execution, while for λ =638-bit it is required almost three times this value, specifically 24.18 seconds. All the results of BSW07 CP-ABSE methods are higher than the response times achieved by W11 construction, which proves that the matrix-based approach is more efficient than the tree-based.

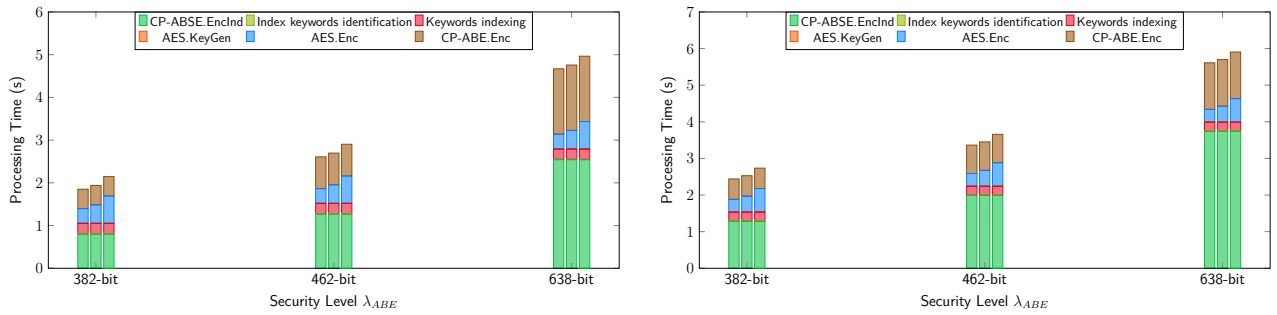


(a) W11. (b) BSW07.

Figure 5.11: Processing times for each CP-ABSE method.

The whole set of operations performed in the scheme could be divided into two parts: data upload and data download. The former considers all the needed operations to outsource an encrypted

files collection, including the index words identification and the encrypted keywords indexing, which are constant time for each security level. Figure 5.12 shows the total processing times for each operation involved in the upload process of both W11 and BSW07 constructions (subfigures 5.12(a) and 5.12(b), respectively). The response times presented correspond to the results obtained when encrypting a 5 MB file (DET-ABE) and one keyword (CP-ABSE) and considering ten attributes in the access control policy, as in the comparison between CP-ABSE methods. Within each CP-ABE security level λ_{ABE} the three AES security levels λ_{AES} were evaluated in order to quantify the time contribution of its deployment for each λ_{ABE}. Such configuration is only relevant to *AES.KeyGen* and *AES.Enc* methods, because the remaining operations are either time constant or not directly related to the symmetric cipher. As already noticed, the most time-comsuming methods are *CP-ABSE.EncInd*, *CP-ABE.Enc*, *AES.Enc*, and now the encrypted keywords indexing, in that order. Although such behavior is common to both W11 and BSW07 approaches, the BSW07 response times are higher than the ones achieved with W11.

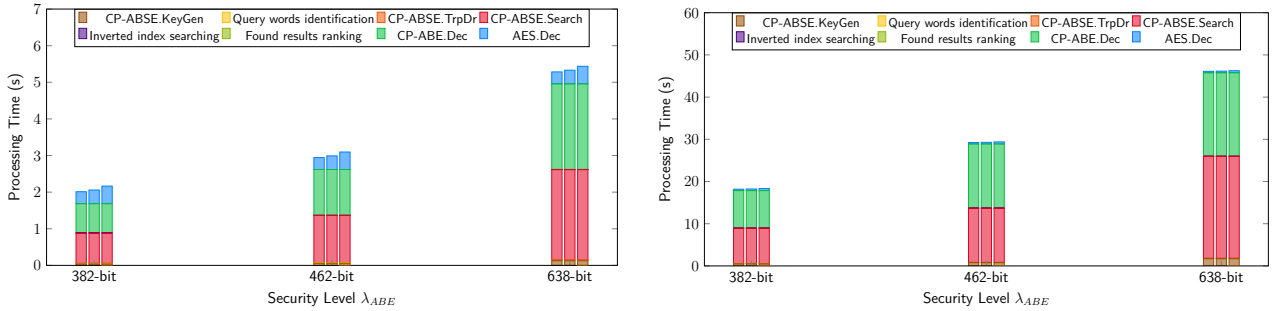


(a) W11. (b) BSW07.

Figure 5.12: Total processing times for upload operations.

The download process involves the execution of eight operations, performed on both data users' and service provider's side, unlike the files collection outsourcing that is only performed by data owners. In this case, the most time-consuming operations for the W11 construction are *CP-ABSE.Search*, *CP-ABE.Dec* and *AES.Dec*, followed by *CP-ABSE.KeyGen*, *CP-ABSE.TrpDr* and

the query keywords identification, but not as much as their predecessors. For BSW07 construction, *CP-ABSE.Search* and *CP-ABE.Dec* are also the most time-consuming methods. However, the third operation with a higher response time is *CP-ABSE.KeyGen*, instead of *AES.Dec*. Here, only in the *AES.Dec* operation are also evaluated the three λ_{AES} along with each λ_{ABE}, and the constant time operations are the query keywords identification and its searching on the inverted index, as well as the found results ranking. The total processing times for each operation in the W11 download process are presented in Figure 5.13(a) while the corresponding results of BSW07 approach are shown in Figure 5.13(b).



(a) W11. (b) BSW07.

Figure 5.13: Total processing times for download operations.

5.2.3 Experiment 3: Information Retrieval Effectiveness

In this experiment, the quality of the information retrieval process is evaluated. In this way, the LISA Test Collection and its benchmark was used. It is a dataset containing 5,999 text files (LISA.Document), 35 information needs, and some possible query keywords, about different topics (LISA.Query), and a list of the relevant documents that are expected to be retrieved for each query (LISA.rel). The experimental evaluation considered only 10 of the 35 queries available, which are listed below along with the number of relevant documents expected.

- Set 1 $\mathbf{QW}_1$: 2 documents
- Set 2 $\mathbf{QW}_2$: 5 documents
- Set 3 $\mathbf{QW}_3$: 1 document
- Set 4 $\mathbf{QW}_4$: 1 document
- Set 5 $\mathbf{QW}_5$: 3 documents
- Set 6 $\mathbf{QW}_6$: 2 documents
- Set 7 $\mathbf{QW}_7$: 11 documents
- Set 8 $\mathbf{QW}_8$: 3 documents
- Set 9 $\mathbf{QW}_9$: 6 documents
- Set 10 $\mathbf{QW}_{10}$: 2 documents

In information retrieval systems, a document is considered as relevant if it appropriately satisfies an information need, not only because the words specified in a user query are located within its contents. In this context, the precision P is a measure of exactness and refers to the portion of the results returned by the cloud storage server that are relevant to the data user. On the other hand, recall R is a measure of completeness that refers to the portion of relevant documents that were returned to the user as a result of a given query.

Although due to its definitions precision and recall seem practically identical, each one allows performing a different assessment of the quality of the search results of an information retrieval system. This metrics can be explained more clearly using a contingency table or confusion matrix. This table keeps the records of the returned results by the information retrieval system, allowing the analysis of the relationship between the relevance and retrieval variables. In Table 5.5, *TP* represents the documents returned by the retrieval system, which are also relevant to the user; *FP* refers to the documents returned, but considered as non-relevant to the user; *FN* is interpreted as those relevant documents that were not returned by the system; and *TN* refers to the documents that are both irrelevant to the user and not returned by the retrieval system.

Table 5.5: Contingency table.

	Relevant	Non-relevant
Retrieved	*TP*	*FP*
Not retrieved	*FN*	*TN*

Given the above, the precision and recall equations are expressed as follows in Equations 5.1 and 5.2, respectively.

$$Precision = P = \frac{TP}{TP + FP} \tag{5.1}$$

$$Recall = R = \frac{TP}{TP + FN} \tag{5.2}$$

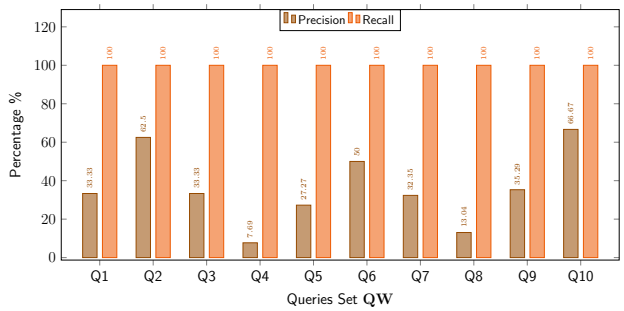
Both precision and recall are based on the complete list of unordered retrieved documents because are set-based measures. To evaluate ranked retrieval results it is necessary to extend the traditional measures or to define new ones that are able of considering ranked documents. In this context, average precision is the average of the precision values at each position where a relevant document is retrieved by the system and mean average precision MAP provides a single-figure measure of quality across recall levels, i.e., a mean value of the average precision of a set of queries.

The experimental evaluation was carried out for three possible values of $Top-k$: (1) considering all the retrieved results $Top - k = *$, (2) where it is just considered a fixed number of retrieved documents $Top - k = \#RL$, based on the number of expected relevant results for each queries set, and (3) considering each result until all relevant documents are retrieved $Top - k = RL$. Each configuration of $Top - k$ was evaluated under two scenarios: a) including at most four synonyms for each keyword in the queries set (SYN case), and b) without taking into account any synonym (NO-SYN case). Finally, the experiments were conducted for different values of Full-Text $k = \{1, 2, 3, 4, 5\}$ in order to examine if adding more query compound words to the queries set has or not a significant effect on the retrieved results ranking.

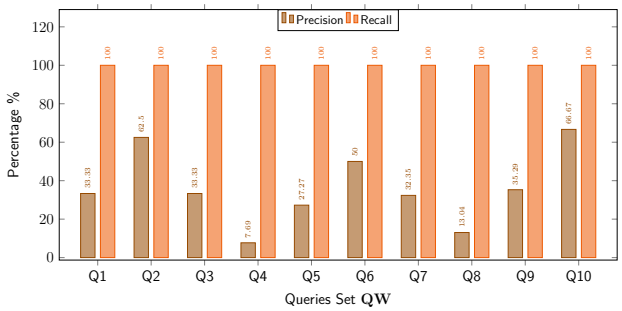
5.2.3.1 All Retrieved Results

As said before, this experiment evaluates precision and recall metrics taking into account the full set of retrieved results, i.e., $Top - k = *$. Using this setting ensures that all relevant documents are retrieved; however, a large number of non-relevant results are also retrieved by the system. At best,

one or two non-relevant documents can be retrieved for a given query. But, in the worst scenario, the amount of non-relevant results can reach up to or more than twice the number of expected relevant results. The experimentation revealed that the ranking score between relevant and non-relevant documents is quite similar, since there is no significant gap that helps to determine from which value the list of non-relevant documents starts. In fact, in some cases the difference between the score of a document doc_B and the previous result doc_A tends to decrease, which suggests doc_B is closely related to doc_A. Also, when $k = 2$, $k = 3$, $k = 4$, and $k = 5$ the rank of the retrieved documents remain the same, opposite to $k = 1$ where in some particular cases the results ranking differs.

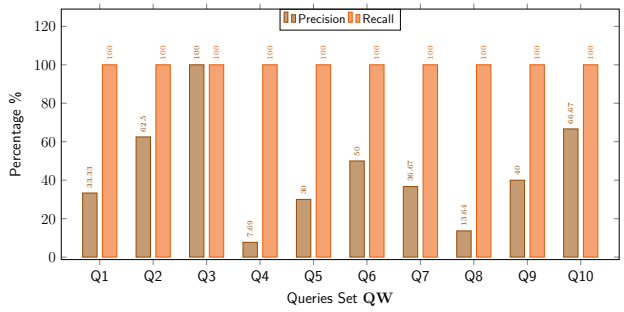


(a) $k = 1$

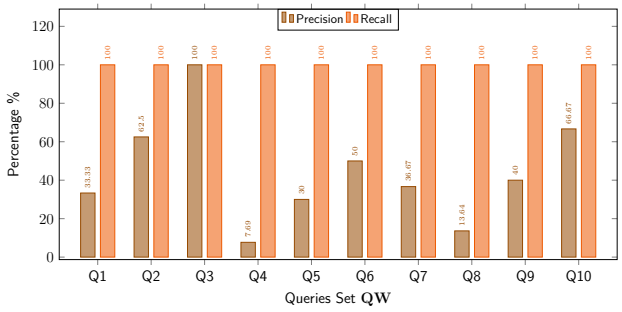


(b) $k = 2 - 5$

1) SYN case.



(c) $k = 1$



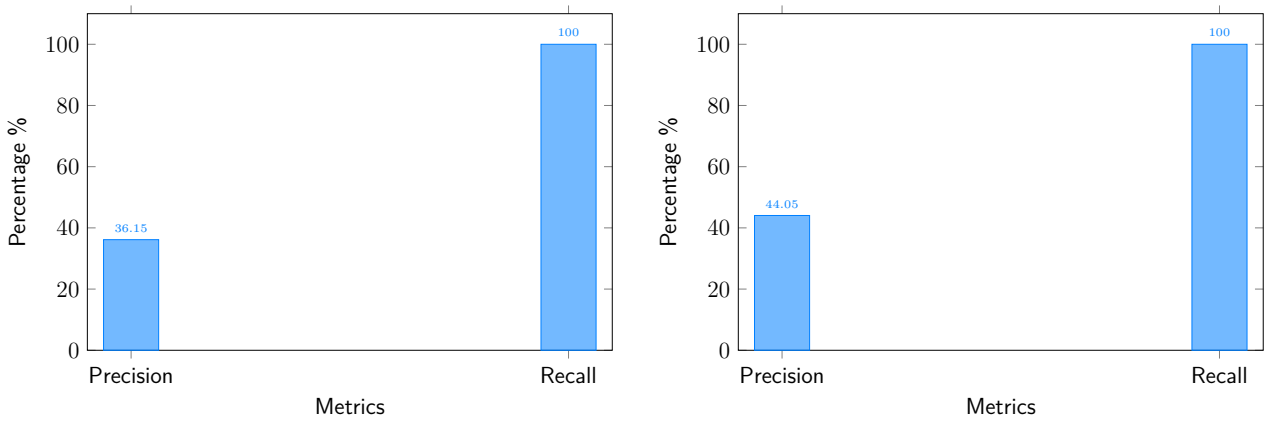
(d) $k = 2 - 5$

2) NO-SYN case.

Figure 5.14: Metrics evaluation for each query: $Top - k = *$.

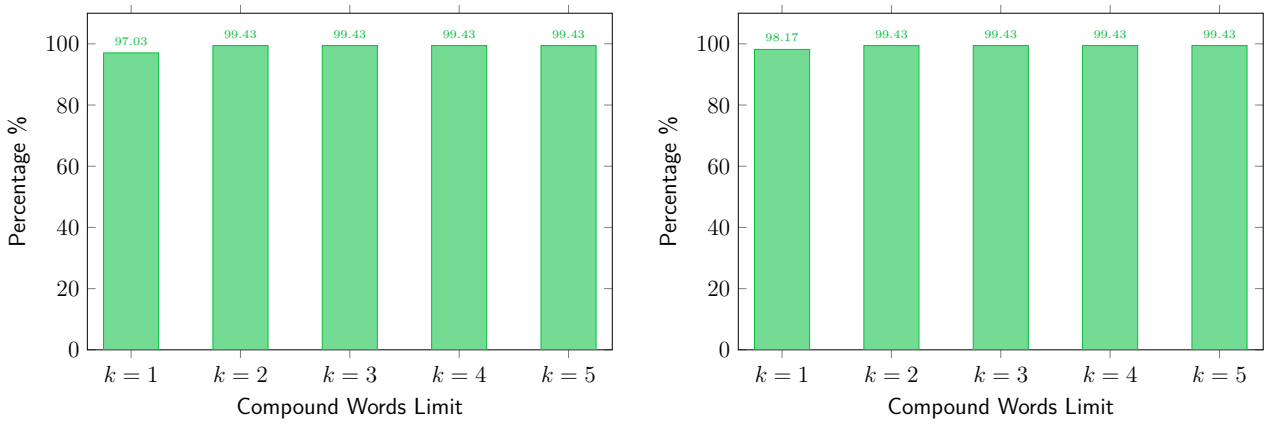
Given the above, the precision and recall graphs were divided into two groups within the scenarios

under analysis: (a) when $k = 1$, and (b) when $k = 2 - 5$ ($k = 2 \equiv k = 3 \equiv k = 4 \equiv k = 5$). Figure 5.14 shows the percentage of precision and recall achieved in each queries set in both cases with words synonyms and without words synonyms. Because all retrieved results are considered, the precision and recall values for $k = 1$ and $k = 2 - 5$ are exactly the same within each scenario. Four of ten queries sets achieve better precision results in the NO-SYN case rather than in the SYN case: $\mathbf{QW}_3$, which rises from 33.33% up to 100%, $\mathbf{QW}_5$, which rises from 27.27% to 30%, $\mathbf{QW}_7$, which goes from 32.35% to 36.66%, and $\mathbf{QW}_9$, which increases from 35.29% to 40%.



(a) SYN case (b) NO-SYN case

Figure 5.15: Overall metrics evaluation: $Top - k = *$.



(a) SYN case (b) NO-SYN case

Figure 5.16: Mean average precision: $Top - k = *$.

Figure 5.15 shows the overall precision and recall results achieved in the $Top-k=*$ setting. As expected by the individual results of the queries set, a higher precision is achieved in the NO-SYN case and the recall in both cases remains the same. Moreover, Figure 5.16 shows the mean average precision, which is identical for both SYN case and NO-SYN case when $k=2-5$, but higher in the NO-SYN case when $k=1$.

As the mean average precision for $k=2-5$ is higher than the one of $k=1$, and because the rank of the retrieved results for $k=2$ to $k=5$ remain the same, the value $k=3$ could be taken as a valid midpoint that guarantees all the expected relevant results would be retrieved by the system. Besides, from such value it could be obtained a significant set of query compound words w_c that helps to increase the ranking score of each found document, since the results ranking is based on both the index words memberships and the query compound words weight values. In other words, the weight of each $w_c \in \mathbf{QW}$ contributes to increase the documents' ranking score by giving priority to those results containing query compound words with higher weight values, which are usually those longer query compound words. A longer w_c can provide more information than single query words because if a document matches a long query compound word, it will also match all the partial words in w_c simultaneously. Even though longer query compound words carry more information than shorter ones, a reasonable value of k can ensure greater efficiency in the search process, while providing enough query compound words that help to improve the ranking score of each relevant result found.

In this sense, Figure 5.17 shows the precision-recall curves for the queries set with at least two expected relevant results when $k=3$. As by its definition precision and recall are set-based measures, the precision and recall values for each curve in the figure are the same considered as part of the assessment of the queries mean average precision. For all queries set, except for one, the precision value remains constant in 100% while the recall progressively increases as the rest of the expected relevant documents are obtained. For queries set $\mathbf{QW}_2$ the precision achieved for the fifth result drops until 71.42% because the last relevant document is ranked in seventh place by the retrieval

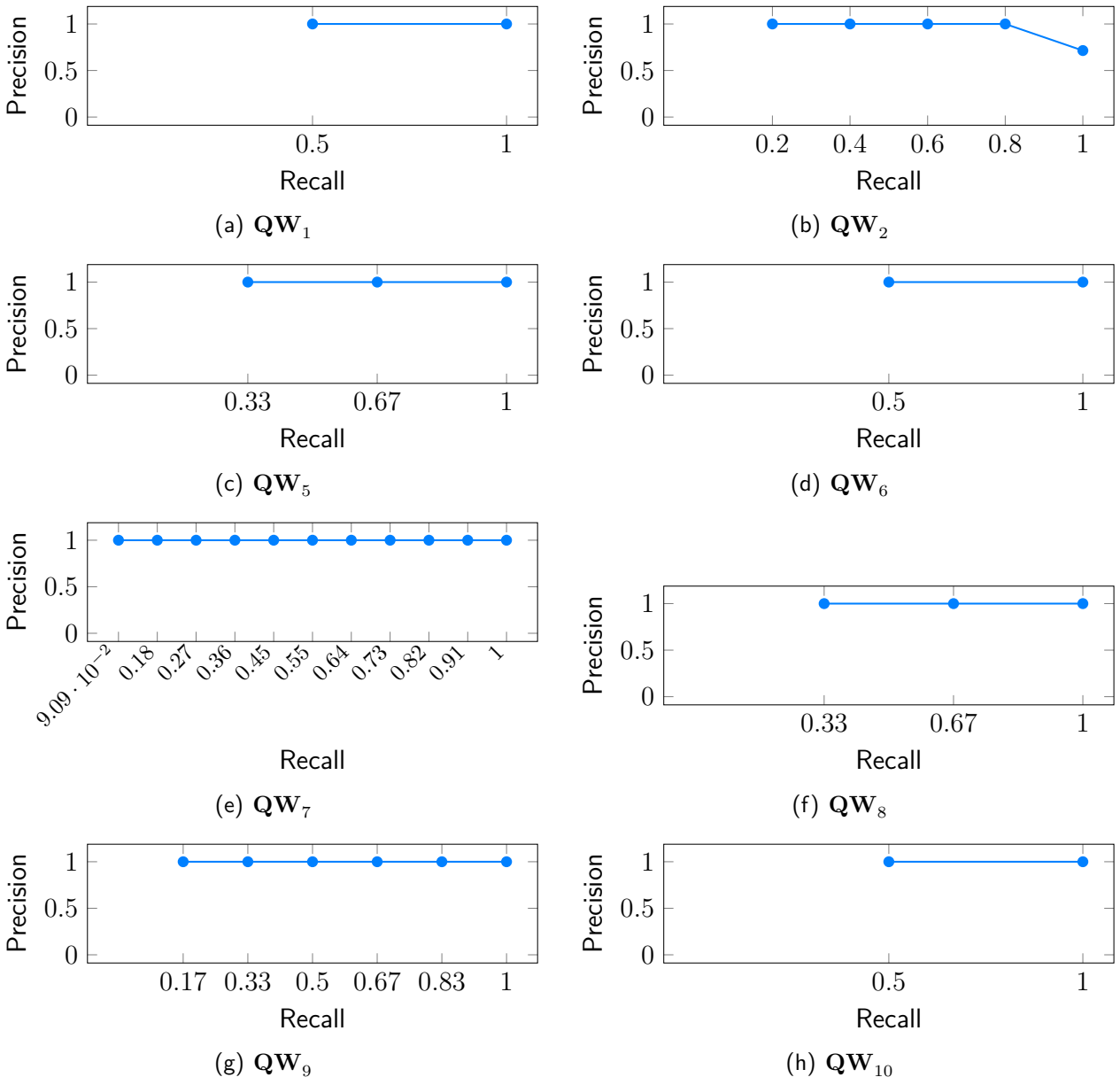


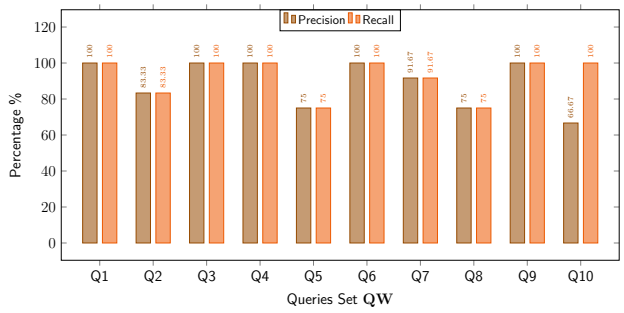
Figure 5.17: Precision-recall curves for each queries set when $k = 3$: $Top - k = *$.

system.

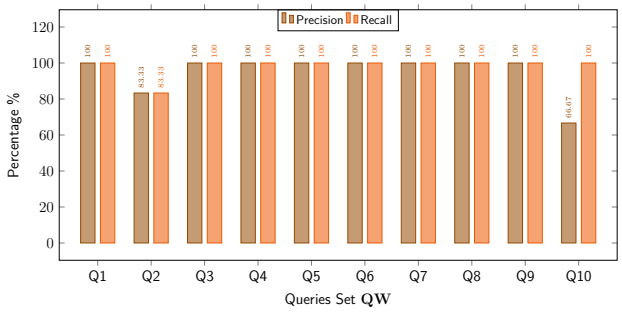
5.2.3.2 *Fixed Number of Retrieved results*

This experiment just considers a fixed number of retrieved results from the whole set. In other words, the $Top - k$ value for each queries set is based on the number of expected relevant results

$\#RL$, which significantly decreases the amount of non-relevant results that are also retrieved by the system. As in the first configuration, the rank of the retrieved results when $k = 1$ is in some cases completely different from those of $k = 2 - 5$. And, again, there is a very small difference among the ranking scores of relevant and non-relevant results, of barely a few decimals between each result.

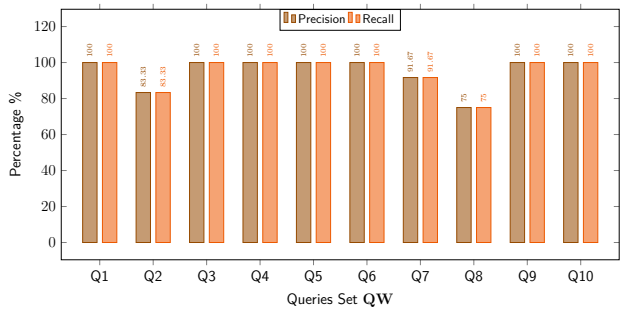


(a) $k = 1$

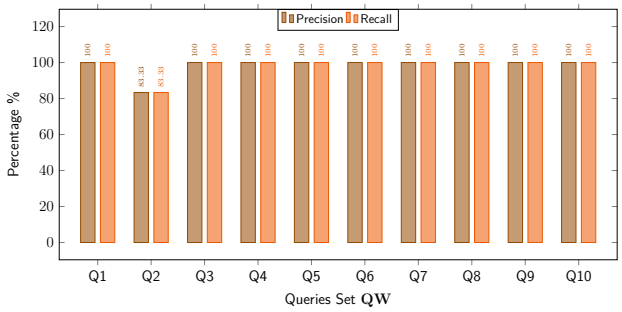


(b) $k = 2 - 5$

1) SYN case.



(c) $k = 1$



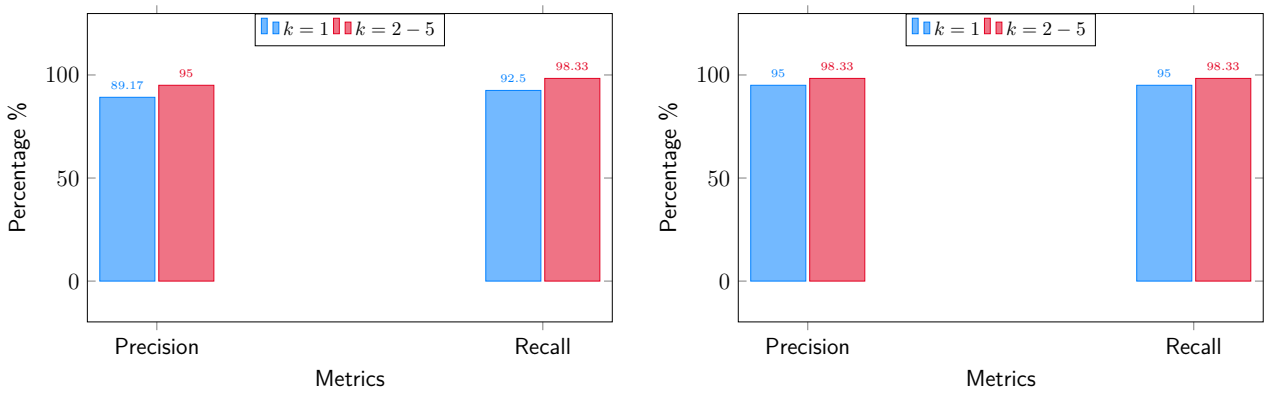
(d) $k = 2 - 5$

2) NO-SYN case.

Figure 5.18: Metrics evaluation for each query: $Top - k = \#RL$.

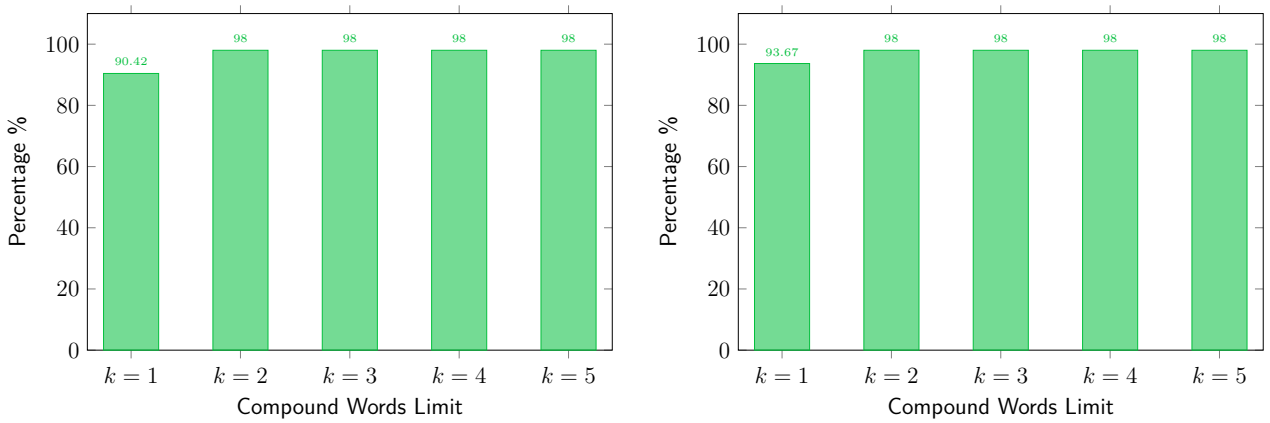
Figure 5.18 shows the percentage of precision and recall reached for each queries set in both SYN case and NO-SYN case scenarios. Opposite to the all retrieved results setting, the precision values achieved are higher than 70% and in all cases, except for $\mathbf{QW}_{10}$ in the SYN case, identical to its respective recall values. Figure 5.19 shows the overall precision and recall results reached in the two groups within the SYN case and NO-SYN case. As shown by the individual results of the queries sets,

it is achieved a higher precision value for both $k = 1$ and $k = 2 - 5$ in the NO-SYN case compared with the SYN case. Unlike the previous results, there is a small difference between the precision and recall measures in the SYN case, but for the NO-SYN case both values are equal. Figure 5.20 shows the mean average precision, which is again identical for both scenarios when $k = 2 - 5$, but higher in the NO-SYN case when $k = 1$.



(a) SYN case (b) NO-SYN case

Figure 5.19: Overall metrics evaluation: $Top - k = \#RL$.

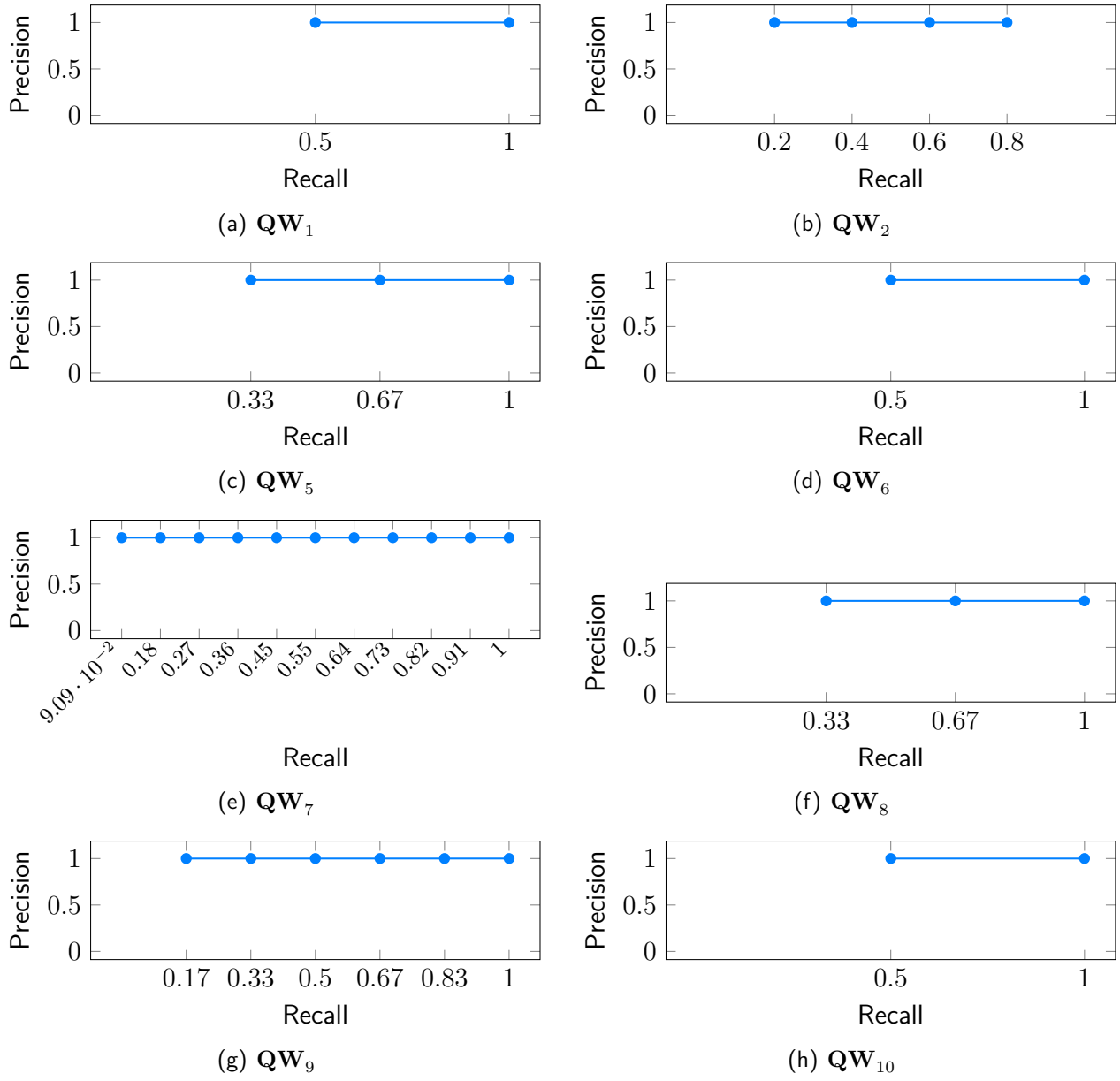


(a) SYN case (b) NO-SYN case

Figure 5.20: Mean average precision: $Top - k = \#RL$.

As it can be seen in Figure 5.21, the results of the precision-recall curves for the queries set when $k = 3$ are almost identical to those achieved in the previous setting. It is possible because, when

considering values of $k > 1$, the ranking algorithm has more information provided by longer query compound words to rank the expected results in the first positions. In this way, the precision of each queries set remains in 100% and the recall values proportionally increases up to 100%, except for $\mathbf{QW}_2$ where there is a missing relevant document and a maximum recall of 80% is reached.



(a) $\mathbf{QW}_1$ (b) $\mathbf{QW}_2$

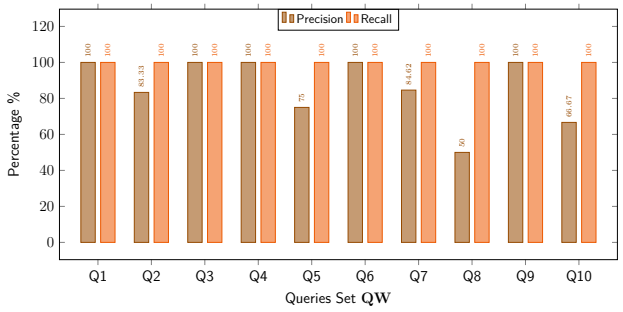
(c) $\mathbf{QW}_5$ (d) $\mathbf{QW}_6$

(e) $\mathbf{QW}_7$ (f) $\mathbf{QW}_8$

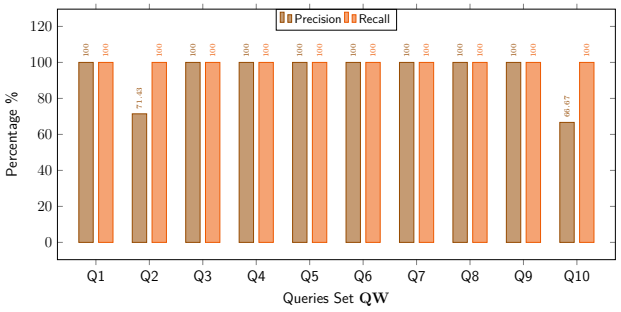
(g) $\mathbf{QW}_9$ (h) $\mathbf{QW}_{10}$

Figure 5.21: Precision-recall curves for each queries set when $k = 3$: $Top - k = \#RL$.

5.2.3.3 Top relevant retrieved results

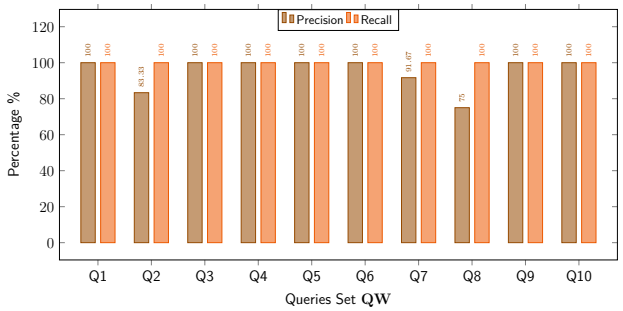


(a) $k = 1$

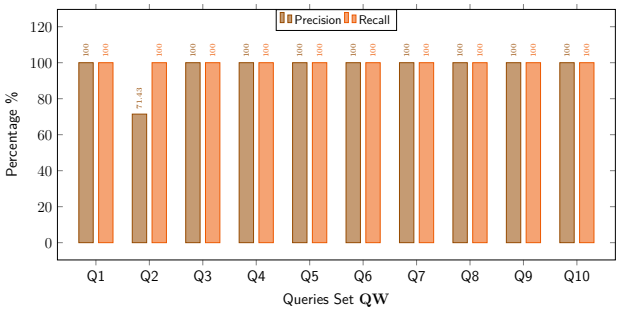


(b) $k = 2 - 5$

1) SYN case.



(c) $k = 1$



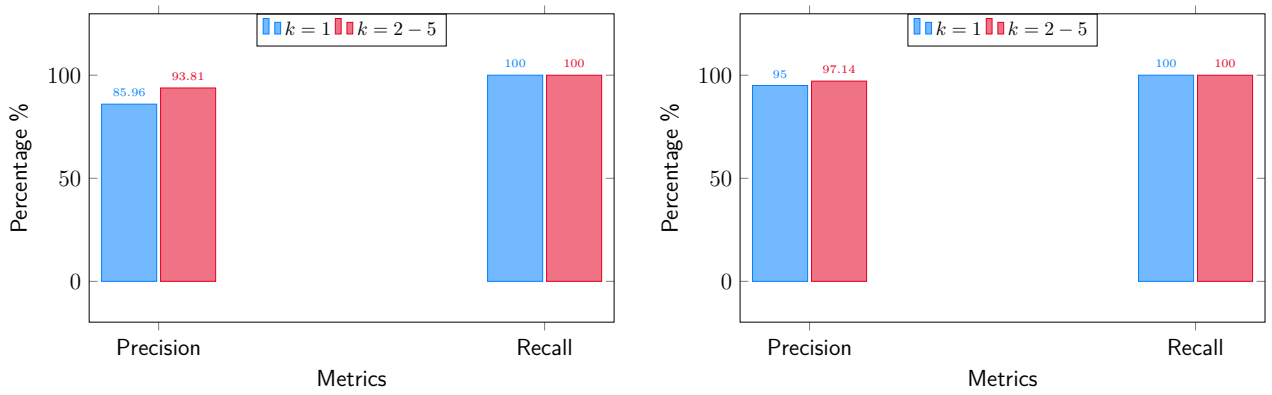
(d) $k = 2 - 5$

2) NO-SYN case.

Figure 5.22: Metrics evaluation for each query: $Top - k = RL$.

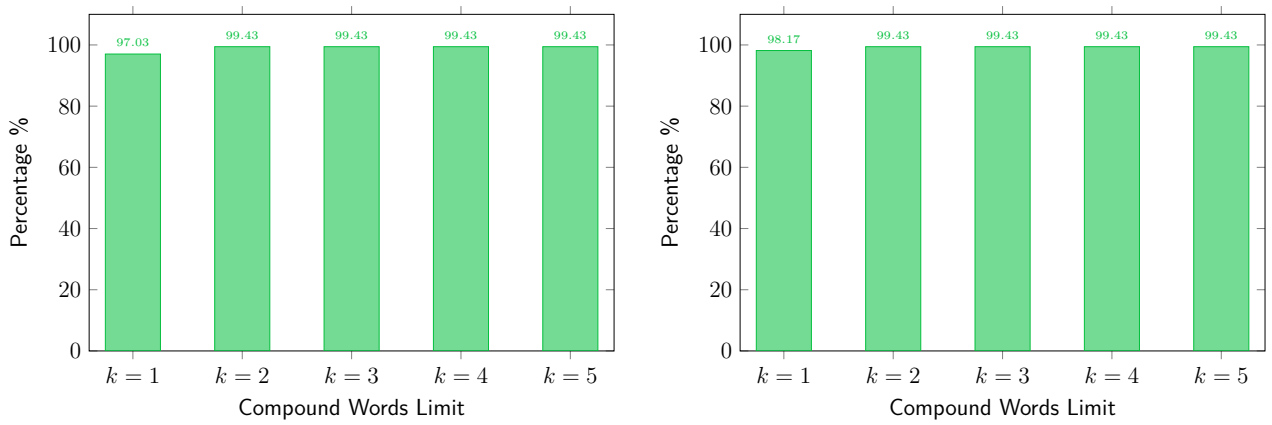
The third experiment is quite similar to the previous one, but it was slightly modified to consider each retrieved result until all the relevant documents were found, i.e., $Top - k = RL$. In this way, for some queries sets the value of $Top - k = RL$ is the same of $Top - k = \#RL$. Figure 5.22 shows the percentage of precision and recall achieved for each queries set in both SYN case and NO-SYN case scenarios. Alike the fixed number of retrieved results, the precision values achieved are higher than 50%, but not in all cases they are identical to its respective recall values. When $k = 1$, four of ten queries sets achieve better precision results in the NO-SYN case rather than in the SYN case:

$\mathbf{QW}_5$, which rises from 75% up to 100%, $\mathbf{QW}_7$, which goes from 84.61% to 91.67%, $\mathbf{QW}_8$, which increases from 50% to 75%, and $\mathbf{QW}_{10}$, which rises from 66.66% up to 100%. When $k = 2 - 5$, just $\mathbf{QW}_{10}$ goes from 66.66% to 100%.



(a) SYN case (b) NO-SYN case

Figure 5.23: Overall metrics evaluation: $Top - k = RL$.

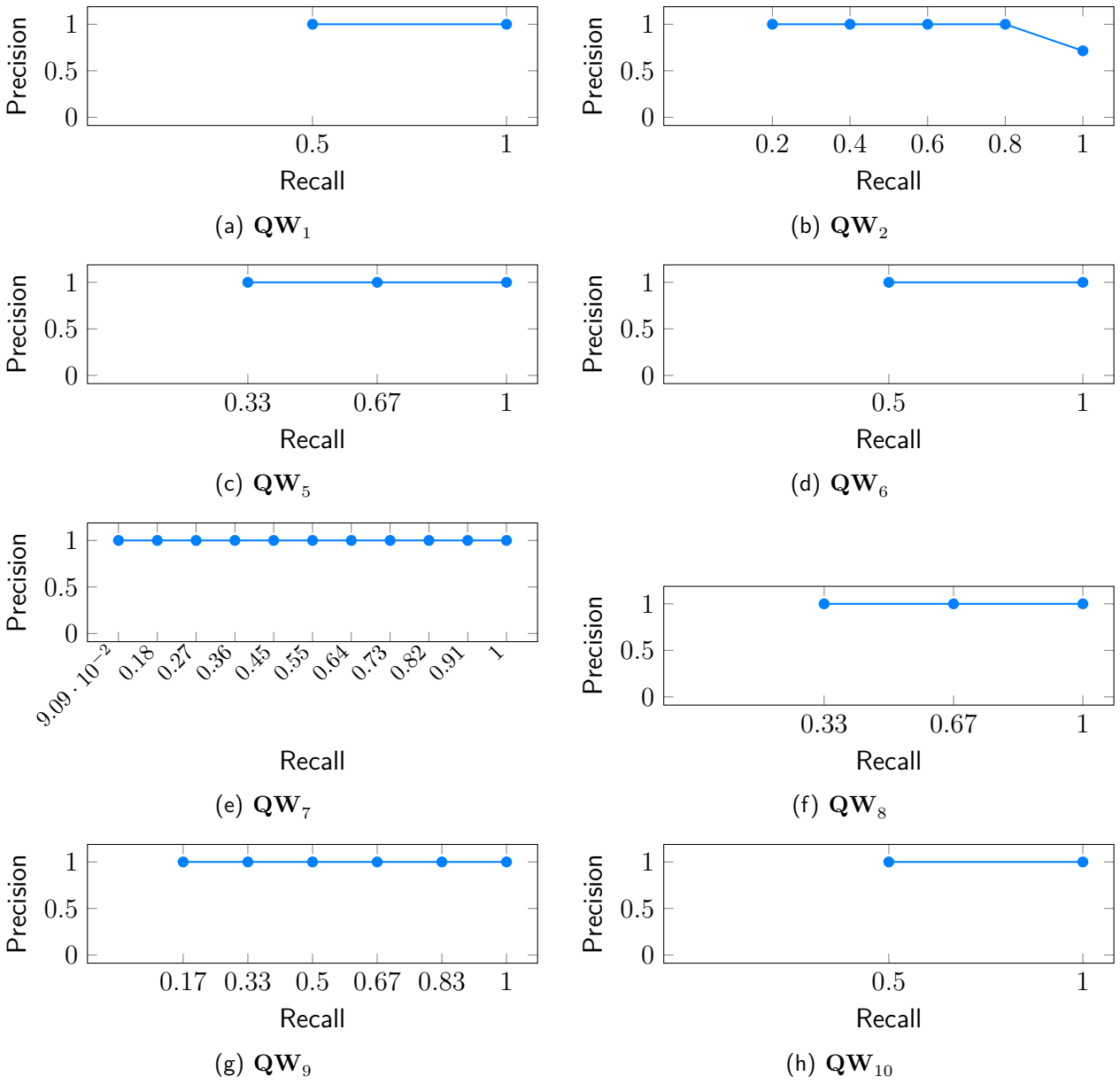


(a) SYN case (b) NO-SYN case

Figure 5.24: Mean average precision: $Top - k = RL$.

Figure 5.23 shows the overall precision and recall results achieved in the $Top - k = RL$ setting in the two groups within the SYN case and NO-SYN case. As in the previous experiments, it is achieved a higher precision in the NO-SYN case, and also the recall in both scenarios remains the

same. Figure 5.24 shows the mean average precision, which remain identical for both SYN case and NO-SYN case when $k = 2 - 5$, but slightly higher in the NO-SYN case when $k = 1$. Finally, Figure 5.25 shows the precision-recall curves for the queries set when $k = 3$, which are the same as those obtained by the all retrieved results configuration.



(a) $\mathbf{QW}_1$ (b) $\mathbf{QW}_2$ (c) $\mathbf{QW}_5$ (d) $\mathbf{QW}_6$ (e) $\mathbf{QW}_7$ (f) $\mathbf{QW}_8$ (g) $\mathbf{QW}_9$ (h) $\mathbf{QW}_{10}$

Figure 5.25: Precision-recall curves for each queries set when $k = 3$: $Top - k = RL$.

Derived from the analysis of the achieved results, it can be concluded that, if no limit of retrieved

results is specified, the distinction among relevant and non-relevant documents cannot be easily established at a glance just by the ranking score each retrieved result reaches. The key to get as much relevant documents as possible relies on the definition of meaningful queries that properly represent an information need by considering those query keywords that could describe in a better way the contents of the expected results. The three experiments showed better results when no words synonyms are considered in the queries set and for values of full-text with $k > 1$. Nonetheless, it is not entirely necessary to use values for k greater than two, or even three depending on the individual queries of each queries set, since it is preserved the retrieved results ranking and it only changes the score for each document.

As shown in $Top - k = *$ and $Top - k = RL$ experiments, a large number of non-relevant results leads to lower precision values because such measure takes into account all the retrieved results, relevant and non-relevant. So, for a higher precision value the best setting seems to be $Top - k = \#RL$. Recall values only decrease if some relevant documents are not retrieved, as in the case of experiment two, where the minimum recall value is 92.5% when $k = 1$ in the SYN case and the maximum achieved recall is about 98% when $k = 2 - 5$ for both scenarios. Regarding mean average precision, it is also affected if not all relevant documents are retrieved by the system. In the particular case of $Top - k = \#RL$, the maximum value for $k = 1$ is about 93% and for $k = 2 - 5$ is of 98%, while in both experiments one and three the maximum values reached are about 98% and 99%, respectively. Considering all this results, the setting that can achieve an appropriate balance between precision, recall and mean average precision is $Top - k = RL$.

5.3 CP-ABSE Results Comparison

This section presents the performance comparison between the two asymmetric constructions presented in this thesis and the CP-ABSE symmetric proposal described in [56]. The experimental outcomes prove that both asymmetric constructions achieve better results than the existing scheme in the state of the art. The results comparison considers three possible ABE security levels: 80-bit,

which is now outdated, 112-bit, and 128-bit, which is the maximum level with acceptable response times for a *Type A* elliptic curve. Besides, the access structure $\mathbb{A}$, used to encrypt the index keywords, and the user's attributes set S_u, used to generate the secret keys, considers a fixed value of ten attributes.

The system keys generation results of the three CP-ABSE constructions are depicted in Figure 5.26. Although all the results are limited to a few milliseconds, the highest response times are achieved by the Type-I BSW07 construction (hereafter referred to as BSW07-A), with a range of 78.2 milliseconds between its highest and lowest values. The next higher response times are obtained with Type-III BSW07 construction (hereafter referred to as BSW07-F), where 5.54 milliseconds is the time required to generate a system key pair for λ =80-bit, 7.69 milliseconds when λ =112-bit, and 8.42 milliseconds for λ =128-bit. Finally, the lower results are achieved by W11 construction, with a range of barely 3.32 milliseconds between the minimum response time (2.27 milliseconds for λ =80-bit) and the maximum value (5.59 milliseconds for 128-bit security level).

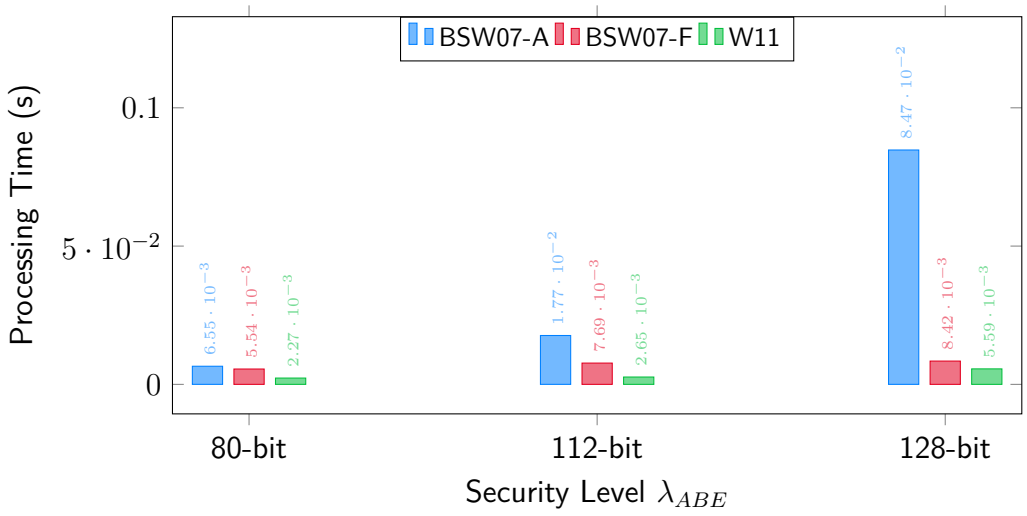


Figure 5.26: System keys generation comparison.

As shown in Figure 5.27, the behavior of the three constructions is the same as that of $\{PK, MK\}$ generation: BSW07-A construction achieves the highest response times followed by BSW07-F, and W11 achieves the lowest results. Nonetheless, the response times obtained increase

for all instances, from a minimum of 7.33 milliseconds (in the case of W11 and λ =80-bit) to a maximum of 911.35 milliseconds (BSW07-A construction with λ =128-bit). As said before in Section 5.2.1, the number of attributes a user owns does not have a significant effect on the response times for the users' secret keys generation under the W11 approach. Thus, it is only affected by the security level used. On the other hand, the results of the two tree-based constructions are influenced by both the number of attributes associated with the secret key and the security level under evaluation.

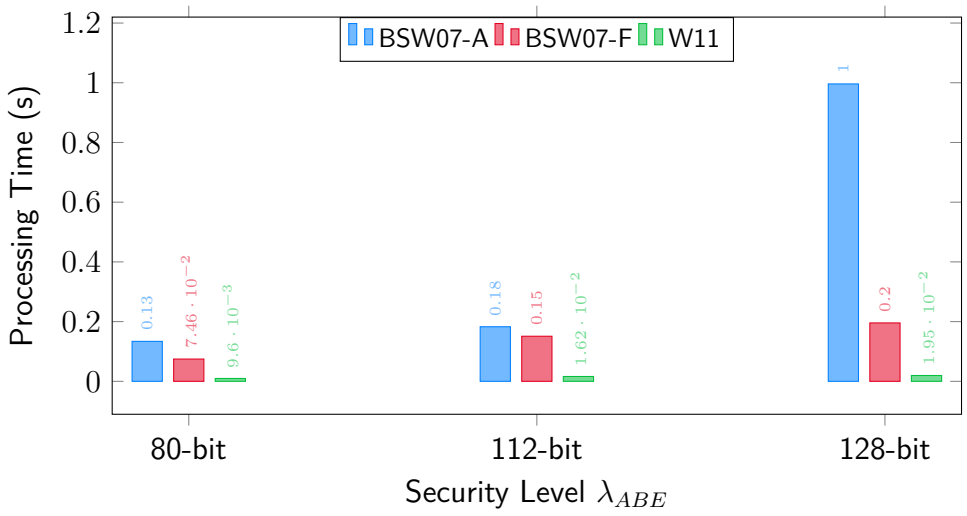
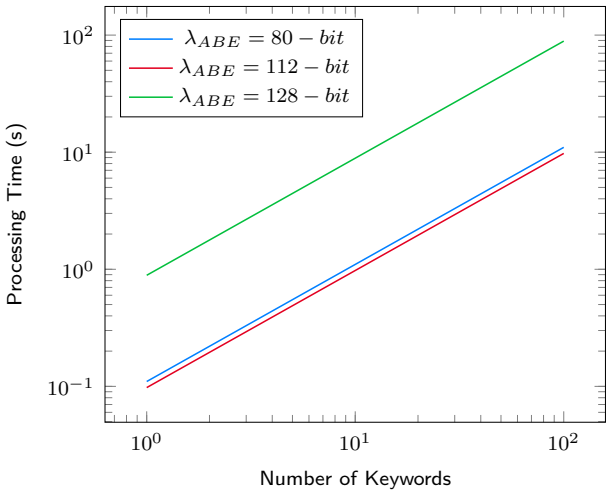


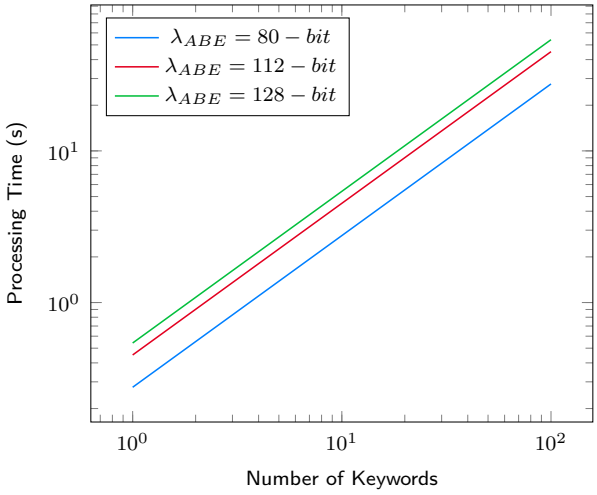
Figure 5.27: Users' secret keys generation comparison.

Figure 5.28 shows the achieved results for the encryption of an index keyword, either single or compound, up to one hundred. As when evaluating higher security levels, both BSW07-F and W11 constructions over *Type F* curve increase the response time as the security level also increases. Even though for BSW07-A construction this behavior is mostly preserved, the lower response times are obtained with λ =112-bit. BSW07-F and W11 constructions obtain higher response times for λ =80-bit and λ =112-bit than the symmetric construction. However, for the highest security level under evaluation, the results of both BSW07-F and W11 constructions are lower than the one obtained with BSW07-A. In this way, the difference in the response times for encrypting a keyword between W11 and BSW07-A is about 0.5757 seconds, while between BSW07-F and BSW07-A is close to 0.3475 seconds. Because of its linear behavior, such difference increases up to almost a minute when

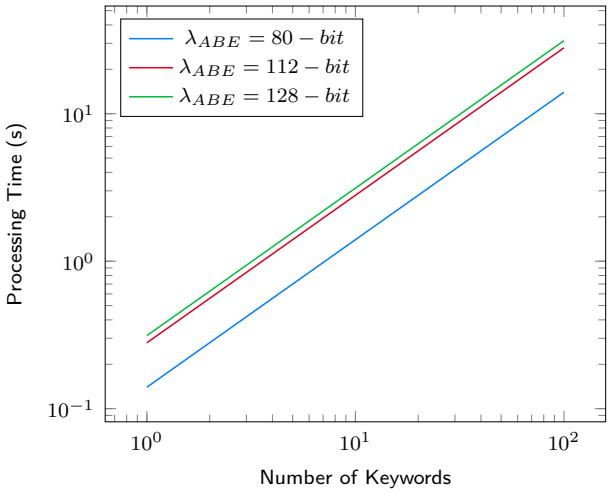
encrypting a set of one hundred index keywords.



(a) BSW07-A [56]



(b) BSW07-F

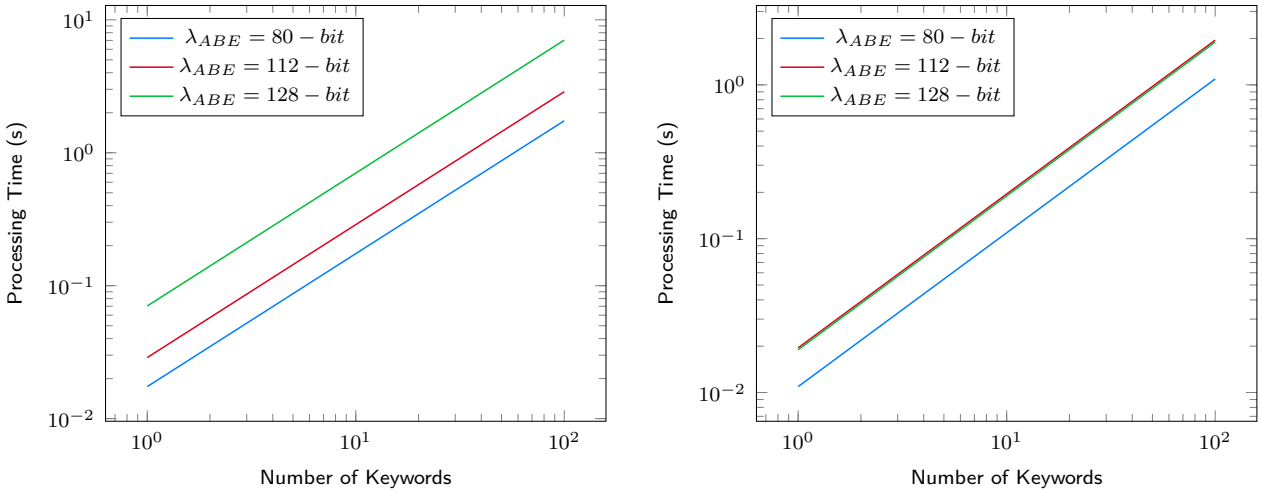


(c) W11

Figure 5.28: Index keywords encryption comparison.

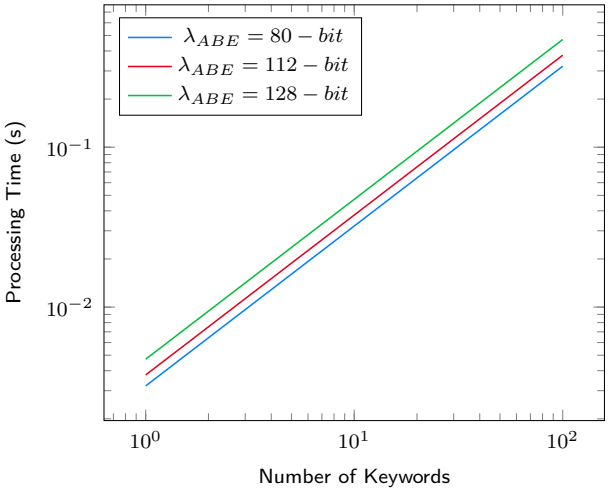
In the case of trapdoors generation, both BSW07-F and W11 achieve higher response times with λ =112-bit rather than with λ =128-bit. However, those results are much lower than the ones accomplished by the BSW07-A construction. As shown in Figure 5.29, the symmetric approach obtains a minimum value of 17.41 milliseconds and a maximum of 70.35 milliseconds to generate the trapdoor for a query keyword. This means that the time required to process a hundred trapdoors

will range between 1.741 and 7.035 seconds for the lowest and highest security level, respectively. Nonetheless, considering the final scenario when generating one hundred trapdoors, the BSW07-F construction achieves a minimum response time of 1.092 seconds and near 2 seconds as the maximum value. On the other hand, with the W11 approach it is required at least 0.3213 seconds to generate a set of one hundred trapdoors and at most 0.472 seconds. Therefore, it is evident that the response time of both asymmetric constructions is much lower than those registered by their symmetric counterpart.



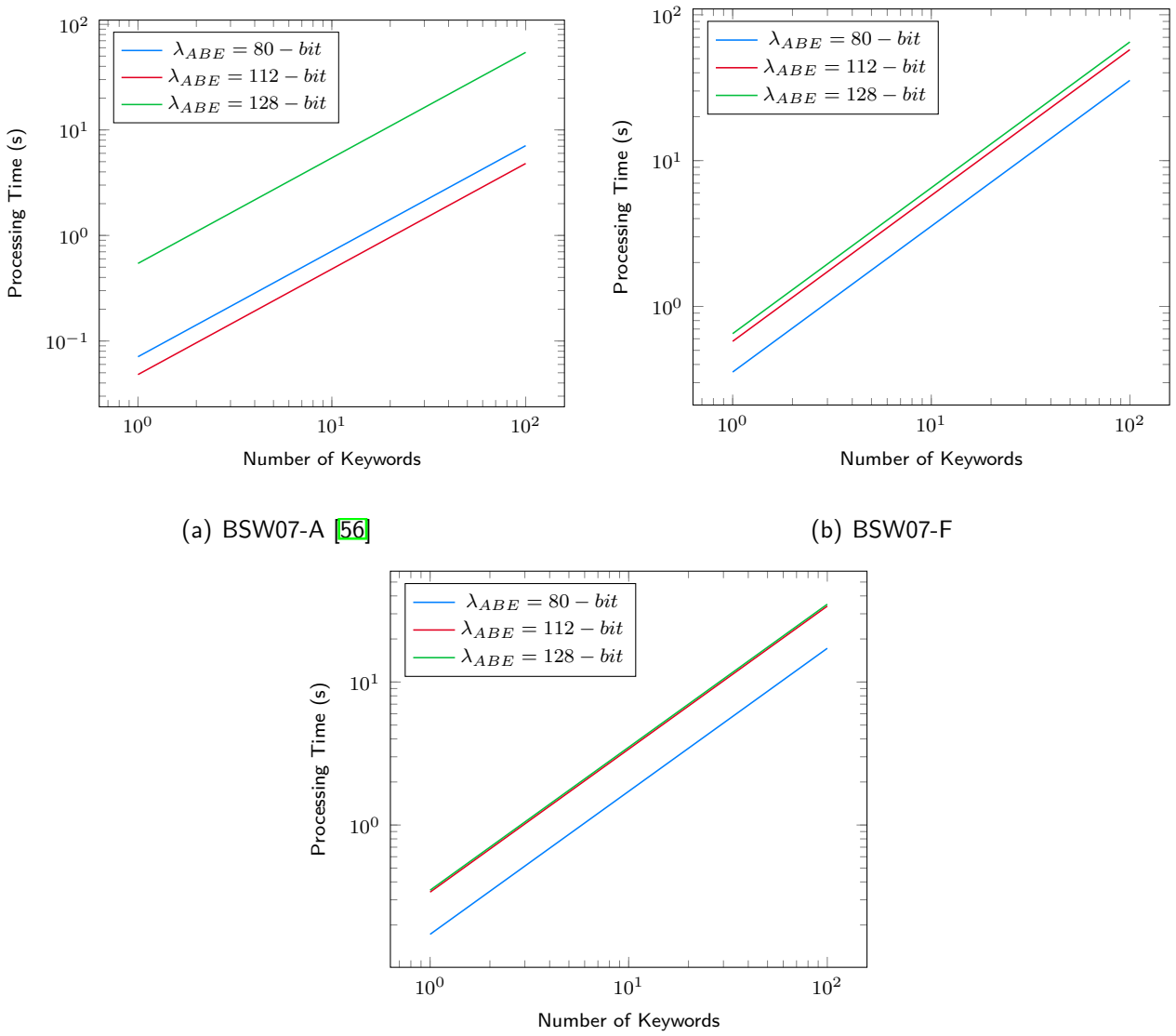
(a) BSW07-A [56]

(b) BSW07-F



(c) W11

Figure 5.29: Trapdoors generation comparison.



(a) BSW07-A [56]

(b) BSW07-F

(c) W11

Figure 5.30: Queries search and users' access permission validation comparison.

Finally, the response time for the search process was evaluated as the previous algorithms considering from one to one hundred query keywords. In such evaluation, BSW07-A obtains lower response times with λ =112-bit, and there is a noticeable difference between the results of both λ =80-bit and λ =112-bit with respect to those of λ =128-bit. Although for the lower security levels the W11 construction requires more time for its execution, when λ =128-bit it is taken about

55% less time to perform the queries search and users' access permission validation than the one needed with the BSW07-A approach. In this experiment, all the results obtained with BSW07-F are much higher than both BSW07-A and W11. The results of this experiment are shown in Figure 5.30.

5.4 Chapter Summary

This chapter started by describing in Section 5.1 the methods and materials used to carry out the experimental evaluation of FABECS. Section 5.2 detailed the experiments objectives and the settings used on the evaluation of each metric defined. It also presented the achieved results and their corresponding analysis. Finally, Section 5.3 presented the performance comparison between the asymmetric constructions detailed in Chapter 4 and the existing symmetric scheme in the state of the art. The accomplished results show that it is feasible the deployment of a fully attribute-based encryption scheme, able to guarantee confidentiality, access control and searching capabilities while meeting standard security levels, on cloud storage and data sharing scenarios. The next chapter presents the conclusions derived from the development of this research project as well as its technological, scientific and academic contributions. It also points out the possible directions for future work.

6

Conclusions and Future Work

This chapter highlights the contributions of the development of this research project, the conclusions drawn from both the literature review and the experimental evaluation carried out, and the possible activities or features that could be considered as part of future work in order to improve the results obtained so far.

Derived from the review of the state of the art, it can be concluded that ABE is the cryptographic technique better suited for cloud storage and data sharing scenarios because of the characteristics it has, such as fine-grained access control, easy data sharing among multiple users and the provision of access control mechanisms. In this context, ABE can be used to accomplish two objectives: enforce fine-grained access control over owner's data and ensure secure information retrieval. However, even though the existing ABSE proposals can deal with some of the challenges identified in current cloud storage systems, there are still some aspects that must be reconsidered. Such aspects include the pairing type used, the accomplishment of standard security levels, the symmetric keys efficient

distribution and management, as well as the improvement of the operations efficiency.

In this research project, for the first time it were presented and evaluated Type-III constructions for both BSW07 and W11 CP-ABE approaches. Such constructions are part of a fully attribute-based encryption scheme, called FABECS, which aims to guarantee sharing and retrieval of encrypted data on cloud storage scenarios. The novel ABSE algorithms allow the deployment of efficient constructions compliant with the security levels recommended by current standards. Assuming either the HBC or SHBC adversarial model, FABECS is intended to preserve both data confidentiality and information retrieval capabilities, while providing an effective and efficient mechanism for fine-grained access control over encrypted data. Therefore, confidentiality is ensured by using AES and access control is achieved by means of ABE. Meanwhile, secure searching and information retrieval is accomplished by using ABSE. In such a way, FABECS is able to provide efficient encryption at three different levels: 1) bulk encryption of outsourced data, 2) access control enforcing and symmetric encryption keys management by means of digital envelopes creation, and 3) novel constructions of an attribute-based searchable encryption scheme.

The achieved results from the experimental evaluation considering Barreto-Naehrig curves prove the feasibility of the proposed constructions for its practical deployment in cloud storage and data sharing scenarios, not only because of its efficiency, but also because of its correctness and efficacy. As described in Section 5.3, both Type-III constructions achieved better results in the response times for the higher security level under evaluation, λ =128-bit, than the Type-I construction reported in the literature. Nonetheless, while both asymmetric constructions achieved good results, it is the W11 approach where lower response times were obtained for all the searching related operations.

Even when in most of the cases the system's response time could be considered as acceptable, there are some results that can be improved by applying acceleration strategies. Mainly those obtained in the index keywords and session keys encryption operations, as well as the users' access permissions validation and session keys decryption. On the one hand, the processing of the attributes associated to the access structure could be done by using the manager worker parallel pattern in

order to process several attributes blocks at a time. On the other hand, large files processing could be done by using the divide and conquer pattern. Besides, response times for data encryption in FABECS could be improved. Such improvement can be done by exploring the use of GPUs for the AES encryption and decryption processes, since AES cipher involves performing repetitive operations in the different data blocks to be processed. Also, FABECS could be implemented and evaluated with other efficient pairing friendly curves, such as the Barreto-Lynn-Scott (BLS), whose use in practical applications is recently being promoted.

Bibliography

[1] Alston, A. (2017). Attribute-based encryption for attribute-based authentication, authorization, storage, and transmission in distributed storage systems. Technical report, Cornell University. arXiv:1705.06002v1.

[2] Anita Aghaie, S. P. R. (2016). Efficient elliptic curve point multiplication with montgomery ladder algorithm. Technical report, Department of Electrical and Microelectronic Engineering, Institute of Technology, Rochester, NY, USA.

[3] Bao, F., Deng, R. H., and Zhu, H. (2003). Variations of diffie-hellman problem. In *Information and Communications Security*, pages 301–312, Berlin, Heidelberg. Springer Berlin Heidelberg.

[4] Barbulescu, R. and Duquesne, S. (2019). Updating key size estimations for pairings. *Journal of Cryptology*, 32:12981336.

[5] Bethencourt, J., Sahai, A., and Waters, B. (2007). Ciphertext-policy attribute-based encryption. In *2007 IEEE Symposium on Security and Privacy (SP '07)*, pages 321–334.

[6] Boneh, D., Crescenzo, G. D., Ostrovsky, R., and Persiano, G. (2004). Public key encryption with keyword search. In *International Conference on the Theory and Applications of Cryptographic Techniques. Advances in Cryptology - EUROCRYPT 2004*, pages 506–522. Springer, Berlin, Heidelberg.

[7] Bouabana-Tebibel, T. and Kaci, A. (2015). Parallel Search over Encrypted Data Under Attribute Based Encryption on the Cloud Computing. *Comput. Secur.*, 54(C):77–91.

[8] Cao, N., Wang, C., Li, M., Ren, K., and Lou, W. (2014). Privacy-Preserving Multi-Keyword Ranked Search over Encrypted Cloud Data. *IEEE Trans. Parallel Distrib. Syst.*, 25(1):222–233.

[9] Chen, Y. W. J. W. X. (2016). Secure searchable encryption: A survey. *Communications and Information Networks*, Vol. 1(No. 4):52–65.

[10] Cisco Networking Academy (2016a). CCNA R&S 6.0 Bridging. Technical report, Cisco Systems, Inc.

[11] Cisco Networking Academy (2016b). Connecting networks. Technical report, Cisco Systems, Inc.

[12] Curtmola, R., Garay, J., Kamara, S., and Ostrovsky, R. (2006). Searchable symmetric encryption: Improved definitions and efficient constructions. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS 06, page 7988, New York, NY, USA. Association for Computing Machinery.

[13] Curtmola, R., Garay, J., Kamara, S., and Ostrovsky, R. (2011). Searchable symmetric encryption: Improved definitions and efficient constructions. *Journal of Computer Security*, 19:895–934.

[14] Ferguson, N., Schneier, B., and Kohno, T. (2010). *Cryptography Engineering: Design Principles and Practical Applications*. Wiley Publishing.

[15] Giry, D. (2018). NIST recommendation for key management (2016). https://www.keylength.com/en/4/. Last accessed: July, 2019.

[16] Grance, P. M. T. (2011). *The NIST Definition of Cloud Computing*. National Institute of Standards and Technology, Gaithersburg, MD, special publication 800-145 edition.

[17] Guo, C., Su, S., Choo, K. R., and Tang, X. (2018). A fast nearest neighbor search scheme over outsourced encrypted medical images. *IEEE Transactions on Industrial Informatics*, pages 1–1.

[18] Guo, W., Dong, X., Cao, Z., and Shen, J. (2018). Efficient Attribute-Based Searchable Encryption on Cloud Storage. In *Journal of Physics: Conference Series*, volume 1087. IOP Publishing.

[19] Hernandez-de la Rosa, S. D. (2013). Secure Storage and Search of Documents in a Cloud Environment. Master's thesis, Cinvestav Tamaulipas.

[20] Ismail, K. C. J. N. W. and Radman, A. (2017). Searchable encryption: A review. *Security and its Applications*, pages 79–88.

[21] Joux, A. and Nguyen, K. (2003). Separating decision diffie-hellman from computational diffie-hellman in cryptographic groups. *J. Cryptology*, 16:239–247.

[22] Kamara, S., Papamanthou, C., and Roeder, T. (2011). CS2: A Searchable Cryptographic Cloud Storage System. Technical Report MSR-TR-2011-58, Microsoft Research Center.

[23] Koo, D., Hur, J., and Yoon, H. (2013). Secure and efficient data retrieval over encrypted data using attribute-based encryption in cloud storage. *Computers & Electrical Engineering*, 39:34–46.

[24] Kumar, P. P., Kumar, P. S., and Alphonse, P. (2018). Attribute based encryption in cloud computing: A survey, gap analysis, and future directions. *Network and Computer Applications*, 108:37–52.

[25] Lauter, S. K. K. (2010). Cryptographic cloud storage. In *Proceedings of Financial Cryptography: Workshop on Real-Life Cryptographic Protocols and Standardization 2010*, volume 6054, pages 136–149. Microsoft Corporation, Springer.

[26] Liu, Z., Cao, Z., and Wong, D. S. (2014). Efficient Generation of Linear Secret Sharing Scheme Matrices from Threshold Access Trees.

[27] Mamta and Gupta, B. (2019). An efficient KP design framework of attribute-based searchable encryption for user level revocation in cloud. *Concurrency and Computation: Practice and Experience*. e5291 CPE-18-1614.R1.

[28] Meffert, D. (2009). Bilinear pairings in cryptography. Master's thesis, Radboud Universiteit Nijmege.

[29] Menezes, A. (2009). An introduction to pairing-based cryptography. *Contemporary Mathematics. Recent Trends in Cryptography*, 477:216–234.

[30] Menezes, A., Sarkar, P., and Singh, S. (2016). Challenges with assessing the impact of NFS advances on the security of pairing-based cryptography. *IACR Cryptol. ePrint Arch.*, 2016:1102.

[31] Miao, Y., Ma, J., Liu, X., Weng, J., Li, H., and Li, H. (2018). Lightweight Fine-Grained Search over Encrypted Data in Fog Computing. *IEEE Transactions on Services Computing*, pages 1–1.

[32] Michalas, A. (2018). The lord of the shares: Combining attribute-based encryption and searchable encryption for flexible data sharing. In *SAC '19: Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*.

[33] Michalas, A. and Yigzaw, K. Y. (2016). LocLess: Do you Really Care Where Your Cloud Files Are? *2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 515–520.

[34] Miller, V. S. (1986). Use of elliptic curves in cryptography. In *Advances in Cryptology — CRYPTO '85 Proceedings*, pages 417–426, Berlin, Heidelberg. Springer Berlin Heidelberg.

[35] Montgomery, D. (2013). *Applied Statistics and Probability for Engineers*. John Wiley and Sons, Incorporated, 6th. edition.

[36] Morales-Sandoval, M. and Diaz-Perez, A. (2015). DET-ABE: A Java API for Data Confidentiality and Fine-Grained Access Control from Attribute Based Encryption. In *Information Security Theory and Practice*, pages 104–119. Springer International Publishing.

[37] Morales-Sandoval, M., González-Compeán, J. L., Díaz-Pérez, A., and Sosa-Sosa, V. J. (2018). A pairing-based cryptographic approach for data security in the cloud. *International Journal of Information Security*, 17(4):441–461.

[38] Rouse, M. and Bigelow, S. J. (2017). What is cloud computing? `http://searchcloudcomputing.techtarget.com/definition/cloud-computing`. Last accessed: September, 2017.

[39] Sahai, A. and Waters, B. (2005). Fuzzy Identity-Based Encryption. In *Advances in Cryptology – EUROCRYPT 2005*, pages 457–473, Berlin, Heidelberg. Springer Berlin Heidelberg.

[40] Silverman, J. H. (2006). An introduction to the theory of elliptic curves.

[41] Song, D. X., Wagner, D., and Perrig, A. (2000). Practical techniques for searches on encrypted data. In *Proceeding 2000 IEEE Symposium on Security and Privacy. S&P 2000*, pages 44–55. IEEE.

[42] Song, W., Wang, B., Wang, Q., Peng, Z., Lou, W., and Cui, Y. (2017). A privacy-preserved full-text retrieval algorithm over encrypted data for cloud storage applications. *Journal of Parallel and Distributed Computing*, 99:14 – 27.

[43] Sosa-Sosa, V. J., Morales-Sandoval, M., Telles-Hurtado, O., and Gonzalez-Compean, J. L. (2017). Protecting data in the cloud: An assessment of practical digital envelopes from attribute based encryption. In *KDCloudApps 2017*. INSTICC, SciTePress.

[44] Sun, W., Yu, S., Lou, W., Hou, Y. T., and Li, H. (2014). Protecting your right: Attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud. In *IEEE*

INFOCOM 2014 - IEEE Conference on Computer Communications, volume 108, pages 226–234. IEEE.

[45] Vacca, J. R. (2013). *Computer and Information Security Handbook*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2nd. edition.

[46] Wang, H., Dong, X., Cao, Z., and Li, D. (2018a). Secure and Efficient Attribute-Based Encryption with Keyword Search. *The Computer Journal*, 61(8):1133–1142.

[47] Wang, J. and Kissel, Z. A. (2015). *Introduction to Network Security*. Wiley Publishing, Inc., 2nd. edition.

[48] Wang, N., Fu, J., Bhargava, B. K., and Zeng, J. (2018b). Efficient retrieval over documents encrypted by attributes in cloud computing. *IEEE Transactions on Information Forensics and Security*, 13.

[49] Wang, S., Ye, J., and Zhang, Y. (2018c). A keyword searchable attribute-based encryption scheme with attribute update for cloud storage. *PLOS ONE*, 13(5):1–19.

[50] Wang, S., Zhao, D., and Zhang, Y. (2017). Searchable attribute-based encryption scheme with attribute revocation in cloud storage. *PLOS ONE*, 12(8):1–20.

[51] Washington, L. C. (2008). *Elliptic Curves: Number Theory and Cryptography*. Chapman & Hall/CRC, 2nd. edition.

[52] Waters, B. (2011). Ciphertext-Policy Attribute-Based Encryption: An Expressive, Efficient, and Provably Secure Realization. In *Public Key Cryptography – PKC 2011*, pages 53–70, Berlin, Heidelberg. Springer Berlin Heidelberg.

[53] Wu, D. N., Gan, Q. Q., and Wang, X. M. (2018). Verifiable public key encryption with keyword search based on homomorphic encryption in multi-user setting. *IEEE Access*, 6:42445–42453.

[54] Yang, Z., Tang, J., and Liu, H. (2018). Cloud Information Retrieval: Model Description and Scheme Design. *IEEE Access*, 6:15420–15430.

[55] Yin, H., Xiong, Y., Zhang, J., Ou, L., Liao, S., and Qin, Z. (2019a). A Key-Policy Searchable Attribute-Based Encryption Scheme for Efficient Keyword Search and Fine-Grained Access Control over Encrypted Data. *Electronics*, 8(3):265.

[56] Yin, H. B., Zhang, J., Xiong, Y., Ou, L., Li, F., Liao, S., and Li, K. (2019b). CP-ABSE: A Ciphertext-Policy Attribute-Based Searchable Encryption Scheme. *IEEE Access*, 7:5682–5694.

[57] Zheng, Q., Xu, S., and Ateniese, G. (2014). Vabks: Verifiable attribute-based keyword search over outsourced encrypted data. *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, pages 522–530.