DESIGNING SECURE ATTRIBUTE-BASED VERIFIABLE DATA STORAGE AND RETRIEVAL SCHEMES IN CLOUD COMPUTING ENVIRONMENT

A Thesis submitted in partial fulfilment of the requirements for the award of the degree of

Doctor of Philosophy

in

Mathematics

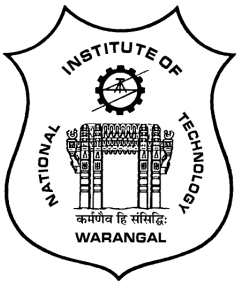
by

Sourav Bera

(Roll No: 719102)

Under the supervision of

Dr. Y. Sreenivasa Rao



DEPARTMENT OF MATHEMATICS

NATIONAL INSTITUTE OF TECHNOLOGY

WARANGAL

TELANGANA STATE - 506 004

JUNE 2024

CERTIFICATE

This is to certify that the thesis entitled **"Designing Secure Attribute-Based Verifiable Data Storage and Retrieval Schemes in Cloud Computing Environment"**, submitted to the Department of Mathematics, National Institute of Technology Warangal, is a record of bonafide research work carried out by **Mr. Sourav Bera**, Roll No. 719102, for the award of Degree of Doctor of Philosophy in Mathematics under my supervision. The contents of the thesis have not been submitted elsewhere for the award of any degree or diploma.

Date:

Dr. Y. Sreenivasa Rao
(Supervisor)
Assistant Professor
Department of Mathematics
National Institute of Technology Warangal
Telangana State - 506 004
India

DECLARATION

This is to certify that the work presented in the thesis entitled **"Designing Secure Attribute-Based Verifiable Data Storage and Retrieval Schemes in Cloud Computing Environment"**, is a bonafide work done by me under the supervision of **Dr. Y. Sreenivasa Rao**, Assistant Professor, Department of Mathematics, National Institute of Technology Warangal and has not been submitted elsewhere for the award of any degree or diploma.

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea / data / fact /source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Sourav Bera

Roll No. 719102

Date:

Dedicated to

My Teachers & Family Members

and

Lord Ramakrishna Paramahangsha Dev

Acknowledgements

I express my sincere thanks to the National Institute of Technology Warangal (NITW) in particular Department of Mathematics, for giving me an opportunity to pursue my Ph.D and supporting me during my research tenure in the Institute.

I would like to take this opportunity to express my utmost gratitude to my beloved supervisor Dr. Y. Sreenivasa Rao, who has always been a source of inspiration for me. Words are inadequate to express my thankfulness for his fatherhood support, guidance and encouragement throughout my research work. His thought provoking comments, suggestions and friendly guidance enabled me in enhancing my research skills. Without his generous help and support it would not be possible for me to complete this work. I shall ever remain indebted to him.

It is my privilege to express thanks to my teachers Prof. D. Srinivasacharya, Dr. R.S. Selvaraj, Dr. E. Satyanarayana, Department of Mathematics, for their inspiring lectures and discussions which paved a basis for my research work. I am also thankful to all faculty members and non teaching staff of the department.

I express my sincere thanks to Prof. R. Padmavathy, Doctoral Scrutiny Committee member, Department of Computer Science and Engineering, for his constant encouragement and knowledgeable inputs for my research work. I express my gratitude to my childhood teacher Mr. Jivan Krishna Das for his enormous help to make me reach upto this level.

I would like to express my gratitude to my seniors K. Sita Ramana, R. Shravan Kumar, Avinabh Srivastava, Atul Kumar Srivastava, K. Naresh, G. Shiva Kumar Reddy, Smriti Tiwari, and friends Pankaj Barman, Bappa Mondal, G. Prithvi Raju, Bikash Modak, Subhabrata Rath, Suryakant Prasad, Brijesh Kumar, R. Naresh, Nidhi Humnekar, Krishnendu Nayak and all my fellow research scholars for their friendship and companionship.

I owe special thanks to my parents and family members for their constant source of encouragement and tremendous care.

I wish to thank all people who helped me in one way or the other towards my success in research endeavors.

- Sourav Bera

List of Symbols and Abbreviations

Symbols

$\mathbb{N}$	:	$\{1, 2, 3, \ldots\}$
$x \xleftarrow{\text{u}} \mathcal{X}$	:	x is randomly chosen from the set $\mathcal{X}$ using uniform distribution
$[t]$	:	$\{1, 2, \ldots, t\}$, where $t \in \mathbb{N}$.
$[\ell_1, \ell_2, \ldots, \ell_k]$	:	$[\ell_1] \times [\ell_2] \times \cdots \times [\ell_k]$, where $\ell_1, \ell_2, \ldots, \ell_k \in \mathbb{N}$.
$\lvert\mathcal{X}\rvert$	:	cardinality of the set $\mathcal{X}$
$\vec{1}_n$	:	an n-length vector of the form $(1, 0, \cdots, 0)$
$\vec{0}_n$	:	an n-length zero vector $(0, 0, \cdots, 0)$
V_{ij} (resp. v_{ij})	:	jth element of $\vec{V}_i$ (resp. $\vec{v}_i$)
V_{ikj} (resp. v_{ikj})	:	jth element of $\vec{V}_{ik}$ (resp. $\vec{v}_{ik}$)
$\lvert\rvert$	:	string concatenation
$\mathbb{Z}_p$	:	the set of integers modulo a prime number p
$\mathbb{Z}_p^*$	:	$\mathbb{Z}_p - \{0\}$
$\{0,1\}^t$	:	the set of binary strings of length $t \in \mathbb{N}$
$\{0,1\}^*$	:	the set of all finite length binary strings
$\lvert\mathcal{F}\rvert$	:	the number of elements in $\mathcal{F}$, where $\mathcal{F}$ is a finite set

Abbreviations

ABE	Attribute-Based Encryption
ABS	Attribute-Based Signature
ABSC	Attribute-Based Signcryption
ABSE	Attribute-Based Searchable Encryption
ABVOD	ABE with verifiable outsourced decryption
ABPRE	Attribute-Based Proxy Re-Encryption
PREKS	Proxy Re-Encryption with Keyword Search
CP-ABE	Ciphertext-Policy ABE
CPA	Chosen Plaintext Attack
CCA	Chosen Ciphertext Attack
DNF	Disjunctive Normal Form
DLin	Decisional Linear
DBDH	Decisional Bilinear Diffie-Hellman

q-DHE	q-Diffie-Hellman Exponent
KP-ABE	Key-Policy ABE
LSSS	Linear Secret-Sharing Scheme
MSP	Monotone Span Program
PPT	Probabilistic Polynomial Time
PHR	Personal Health Record
DO	Data Owner
DU	Data User
EMR	Electronic Medical Record
KGAs	Keyword value Guessing Attacks

Abstract

Cloud computing technology is a novel storage and computing paradigm that enables individuals and organizations to store data, share data with the intended group of users, and retrieve data when required. It greatly improves peoples' data storage and sharing, and data retrieval capabilities by providing flexible, less expensive, and quality services. For data security and privacy concerns, fine-grained data access control, authenticated and secure data storage, authorized data searching, and self-verifiability of the corrrectness of search results are of critical importance. Attribute-based cryptographic framework is a promising solution for applications requiring fine-grained access control. However, a significant computation cost that rises in complexity with access policy complexity affects the majority of attribute-based cryptosystems. Because of this, their usefulness in resource-constrained environments may be compromised. Hence, this thesis aims at designing secure and efficient attribute-based cryptographic schemes with data storage, data sharing, and data retrieval in cloud computing environments.

The contributions of the thesis are threefold. We, first, propose a lightweight online-offline attribute-based data storage and retrieval scheme with Boolean keyword search mechanism. The computationally intensive tasks are either offloaded to the cloud or offline phase and the lightweight operations are carried out by the data user, which makes the scheme lightweight.

Next, we design a verifiable and Boolean keyword searchable attribute-based signcryption scheme in a cloud-based Electronic Medical Record (EMR) management system. The scheme allows EMR owners to store and share their personal EMRs with specific healthcare professionals. It uses disjunctive normal form encryption policy to make the scheme communicationally efficient. Both the aforementioned schemes achieve data owner (DO) privacy, data and DO authenticity, non-interactive verifiability, fine-grained access control over encrypted data, Boolean keyword search, keyword privacy, outsourced decryption, and provable security.

Further, to achieve efficient data sharing functionality along with data searching, we propose an attribute-based proxy re-encryption scheme with Boolean keyword search mechanism. We prove that the scheme is adaptive chosen ciphertext attack secure at both the original and re-encrypted ciphertext, and chosen keyword attack secure on both ciphertext and token.

Keywords: Attribute-based encryption, attribute-based signature and signcryption, bilinear pairing, constant decryption cost, linear-secret sharing scheme, data

storage and retrieval, data sharing, attribute-based searchable encryption, attribute-based proxy re-encryption, Boolean keyword search, search results verification.

List of Published Papers

1. **Sourav Bera**, Suryakant Prasad, Y Sreenivasa Rao, Ashok Kumar Das, and Youngho Park. Designing attribute-based verifiable data storage and retrieval scheme in cloud computing environment. *Journal of Information Security and Applications*, vol. 75, pp. 103482, Elsevier, 2023.
https://doi.org/10.1016/j.jisa.2023.103482

2. **Sourav Bera**, Suryakant Prasad, and Y Sreenivasa Rao. Verifiable and boolean keyword searchable attribute-based signcryption for electronic medical record storage and retrieval in cloud computing environment. *The Journal of Supercomputing*, vol. 79, pp. 1-59, Springer, 2023.
https://doi.org/10.1007/s11227-023-05416-8

3. **Sourav Bera**, and Y Sreenivasa Rao. Searchable Attribute-Based Proxy re-encryption : Keyword Privacy, Verifiable Expressive Search and Outsourced Decryption. *SN Computer Science*, vol. 5, pp. 1-24, Springer, 2024.
https://doi.org/10.1007/s42979-024-02646-2

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 General Introduction

In our day to day life, data plays an indispensable role in decision making for both private and public domains. Due to this fact, the voluminous personal data is maintained by both individuals and organizations, and the necessary data is being shared with various kinds of users more often. Data management becomes a critical concern in such scenarios. The emergence of cloud computing paradigm greatly reduces the cost of personal data management and maintenance. Data storing and sharing, and data retrieval are the two key components in cloud computing technology. To facilitate data sharing and ensure secure data transmission to the designated data users (DUs) in public networking settings, the data owners (DOs) outsource their data to the cloud server. The cloud stores the data at storage servers. Next, the cloud performs a search operation and transmits the outcome of the search to the user through a wireless channel subsequent to obtaining a data retrieval request from a DU. In bringing various benefits, cloud computing technology creates new challenges including data security and data access control. The sensitive outsourced data should only be accessed by authorized users.

To realize data confidentiality and fine-grained data access control over encrypted data, Attribute-Based Encryption (ABE) schemes [75, 4, 85] are acknowledged as being extremely prevalent. Each user in the ABE framework has a unique collection of attributes that act as their public key. Attributes can be elements like a users designation, affiliation, or other typical abstract credentials. Based on whether the secret key or the ciphertext is linked to an access policy, ABE is categorized into Key-Policy ABE (KP-ABE) [75, 28] and Ciphertext-Policy ABE (CP-ABE) [4, 85]. A set of attributes are appended to the ciphertext and an access policy is associated

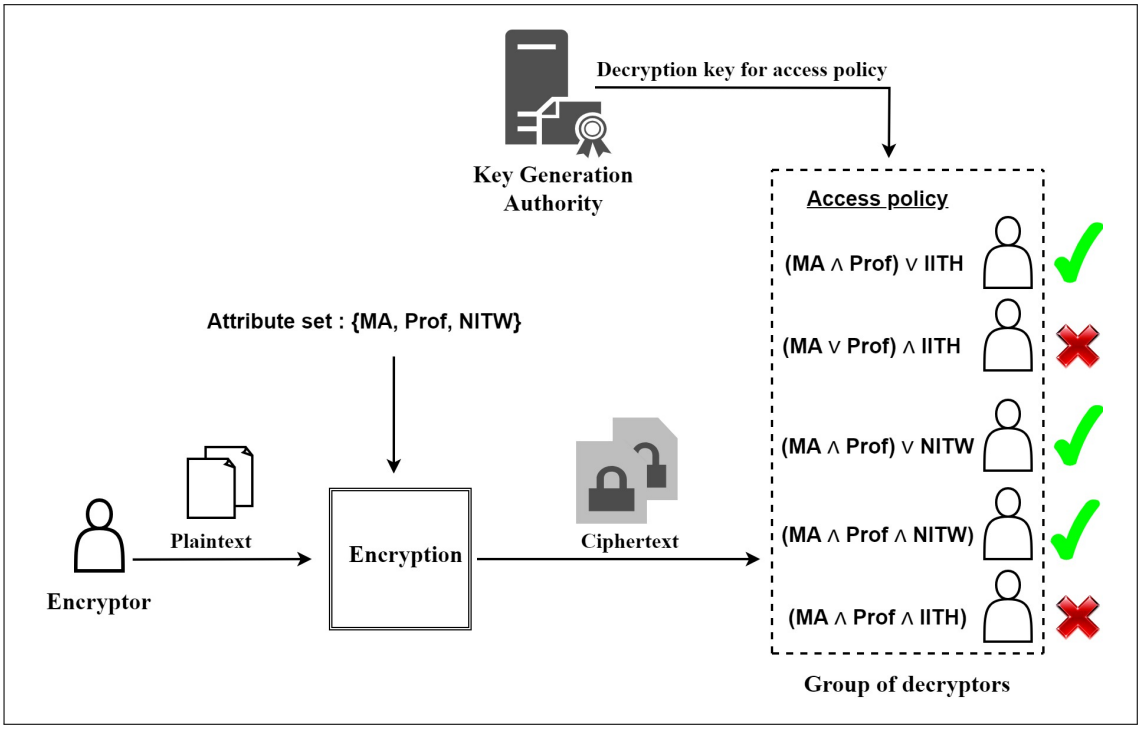


Figure 1.1: KP-ABE

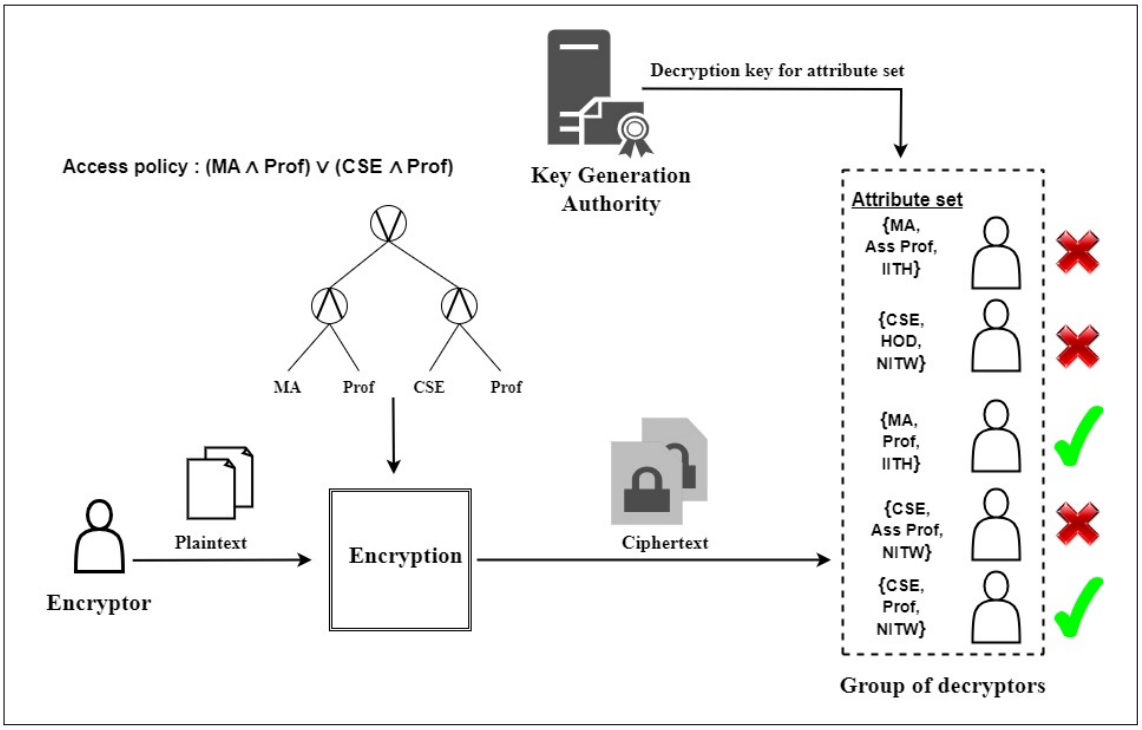


Figure 1.2: CP-ABE

to the secret key in the KP-ABE framework (as shown in Figure 1.1). The CP-ABE framework (as shown in Figure 1.2) associates the secret key with a set of

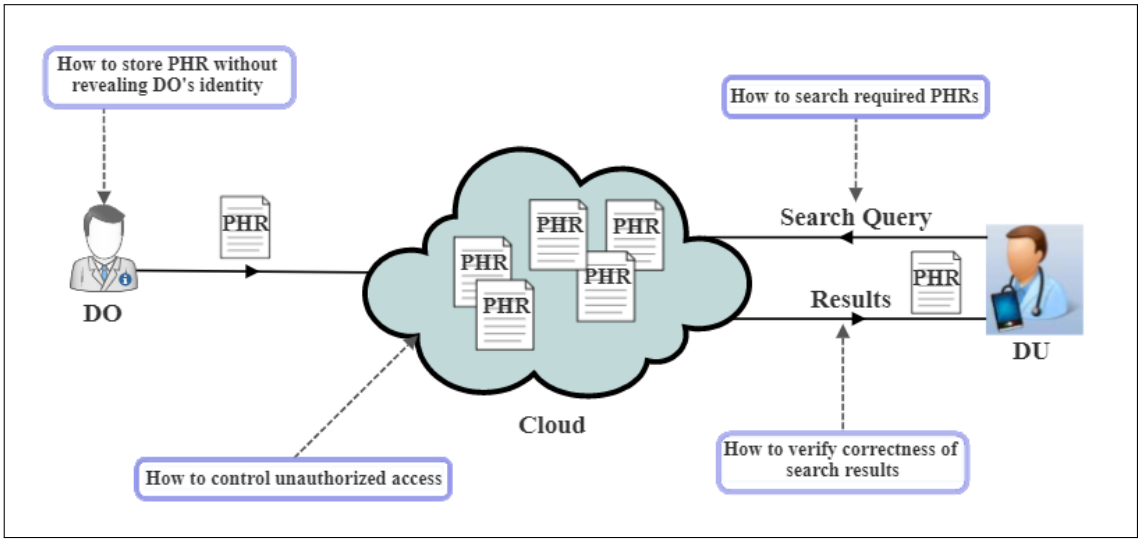


Figure 1.3: Cloud-Based PHR Management System

attributes and uses an access policy to create the ciphertext. An access policy over an attribute universe U is represented by a Boolean formula, in which attributes are connected together with the assistance of the logical operators AND($\wedge$), OR($\vee$). A DU receives a secret key for his attribute set and an access policy is involved in ciphertext generation. And, the DU is able to access the encrypted data if the attribute set satisfies the access policy.

Consider, as a motivating example, a cloud-based Personal Health Record (PHR) management system (displayed in Figure 1.3), where the DOs (such as patients) upload their PHRs to the public cloud for storing and sharing with the specified DUs, such as physicians, medical insurance agent, etc. Since the PHR contains sensitive data, such as disease information, the DO's privacy/anonymity must be ensured while sharing PHRs with DUs; else, the DO will be identified on social platform [70]. Apart from allowing DOs to anonymously impose fine-grained PHR access control, permitting the DUs to retrieve encrypted PHRs of their interest from a cloud is also an important problem.

A DU may search for his patient's medical history related to a particular disease, say Covid-19 or heart disease, but does not want to know about other diseases. Hence, how to enable DUs to efficiently filter out required PHRs by specifying suitable keywords is another crucial problem. Aiming to achieve fine-grained access control and keyword-based data retrieval from the cloud, Attribute-Based Searchable Encryption (ABSE) primitives [92, 81, 88, 13, 12] have been proposed. However, these schemes cannot provide the data and DO authenticity, and DO anonymity, which are crucial for PHR management systems. But Attribute-Based Signcryption

(ABSC) [15, 72, 70], a logical integration of ABE [76, 29, 85] and Attribute-Based Signature (ABS) [57, 72], provides the data and DO authenticity, and DO anonymity in addition to the fine-grained access control over encrypted data.

To access the required patients' PHRs stored at the cloud, a DU creates a data retrieval request using appropriate keywords and delegates the same to the cloud. Next, the cloud locates suitable PHRs and partially decrypts them. Given these transformed PHRs, the DU (e.g., a doctor) treats the patients accordingly. Sometimes a patients inaccurate treatment may result from an untrusted cloud occasionally returning a transformed PHR in the correct format but containing incorrect PHR information [40]. This could be serious threat for the patient's life. So, the difficulty of allowing a data user (DU) to independently check the precision of search outcomes acquired from the cloud becomes a topic of considerable interest. Realizing such verification mechanism in the context of ABSC is quite challenging if we integrate the ABSC with keyword search functionality.

In a medical data sharing system, a covid-19 patient Harry wants to test whether he is covid-positive or not in a clinic. The clinic should be located within 4 km from London, the doctors should be senior doctor and appointed as a covid-specialist. Harry encrypts his medical record using an access policy $\Gamma = \{$*covid specialist* $\wedge$ *senior doctor* $\wedge$ *Location : within 4 km from London*$\}$ and a keyword set. The clinics satisfying Γ can decrypt the encrypted medical record. However, they couldn't get to the exact record by typing the keywords. Instead, the clinic needs to decrypt all the medical records satisfying Γ and then satisfy the keyword set to get the intended medical record. Also, in case, the clinic wants to share the medical record with some other junior doctors of the hospital located within 10 km from Nottigham, it needs to decrypt the encrypted record sent by Harry and thereafter encodes the medical record using an access policy $\Gamma' = \{$*covid specialist* $\wedge$ *junior doctor* $\wedge$ *Location : within 10 km from Nottigham*$\}$ and a keyword set. This method is not suitable since the clinic has to perform n pairs of encryption and decryption processes for n number of patients. Consequently, this system is extremely inefficient regarding data searching and sharing.

The majority of attribute-based cryptosystems are significantly affected by a computation cost that increases in complexity with access policy complexity. As a result, their effectiveness in resource-constrained environments may be impeded. Hence, it is highly desirable to construct attribute-based cryptosystems which feature low computation cost for the environments equipped with computational capability constrained devices.

An insecure design [26] may result from a straightforward combination of attribute-based keyword policy search primitives, ABE with verifiable outsourced decryption (ABVOD), and ABS. Specifically, the design may be vulnerable to Keyword value Guessing attacks (KGAs) and chosen ciphertext attacks. Moreover, it is much more difficult to realize the verifiability mechanism in such designs.

Functionalities. The issues discussed so far motivate us to achieve the following functionalities in the attribute-based cryptographic schemes.

(i) *Fine-grained data access control over encrypted data*: Only authorized DUs with valid trapdoors can access the data stored in cloud. A ciphertext and the associated data retrieval token will not reveal anything regarding the actual plaintext to even the cloud sever.

(ii) *Data and DO authenticity*: The cloud server accepts a ciphertext for storage only when the DOs signature is valid. An authorized DO who has obtained a valid signing key from key generation authority (KGA) can create a verifiable correct signature, thereby unauthorized DOs cannot store their data in the cloud server.

(iii) *DO anonymity*: The ciphertext leaks no information about the DOs attributes involved in the corresponding signing policy. The cloud server can just check that whether the signature satisfies the signing policy, but cannot determine the set of signing attributes originally used to create the signature components.

(iv) *Keyword policy search over encrypted data*: Our approach enables an expressive and versatile keyword search over encoded data, where a data user sends a Boolean query to the cloud and gets back the required search results in a single search request.

(v) *Keyword privacy*: To protect the privacy of the keywords, our scheme assigns a generic name to each keyword value; furthermore, solely the generic keyword names are encoded in both the ciphertext and the search token. The actual value of a keyword can not be deduced from merely the ciphertext and its token in the absence of the cloud secret key.

(vi) *Constant decryption cost for DUs*: Our scheme supports the outsourced decryption mechanism. Most of the decryption calculations are outsourced to the cloud, so DUs can recover a message with a constant number of lightweight operations.

(vii) *Non-interactive search results verification*: Our schemes enable a DU to check the correctness of search, transform and signature verification operations executed by the cloud *without interacting* with any authority. We use *verify-then-decrypt* framework, wherein the DU first verifies the correctness of the results obtained from the cloud and next recovers the plaintext accordingly.

(viii) *Data sharing*: Along with data searching, our scheme enables a data sharing mechanism, where a DU can share the encrypted data sent by a DO with another DU without decrypting it.

(ix) *Keyword set updating*: Prior to sharing a ciphertext with others, our method facilitates the update of the keyword set, hence allowing for future modification of the keyword set. The ability to easily modify the ciphertext keyword set depending on the data-sharingrecord makes this characteristic convenient for DOs.

1.2 Objectives of the Thesis

The main objectives of the thesis are to design the following attribute-based cryptographic schemes.

1. Designing a computationally efficient attribute-based verifiable data storage and retrieval scheme in the cloud computing environment supporting (i) data and DO authenticity (ii) DO anonymity, (iii) fine-grained data access control, (iv) keyword policy search over encrypted data, (v) keyword privacy, (vi) constant decryption cost for DUs, and (vii) non-interactive search results verification.

2. Constructing a communicationally efficient attribute-based searchable signcryption scheme for electronic medical record (EMR) storage and retrieval that supports simultaneously the aforementioned functionalities.

3. Designing a ciphertext-policy searchable attribute-based proxy re-encryption scheme achieving (i) keyword policy search over encrypted data, (ii) data sharing, (iii) constant decryption cost for DUs, (iv) keyword privacy, (v) keyword set updating, and (vi) search results verification.

1.3 Thesis Summary

The thesis is organized into six chapters, with **Chapter 1** providing a general introduction to attribute-based cryptographic primitives and their applications to cloud-based data storage and retrieval system, and **Chapter 6** as the conclusion. **Chapter 2** presents the cryptographic preliminary notions to make the thesis self-contained. Additionally, this chapter discusses the previous works that are most relevant to the main theme of the thesis. In Chapters 3 to 5, we present our contributions. Each presented attribute-based cryptosystem includes the description of a system model along with the security proof. Chapters 3 to 5 are summarized below.

In **Chapter 3**, we first propose a computationally efficient attribute-based verifiable data storage and retrieval scheme in the cloud computing environment. Our scheme supports data and DO authenticity, DO anonymity, keyword privacy, keyword policy search over encrypted data, non-interactive verifiability, and outsourced unsigncryption. To make our scheme lightweight, we make use of the online-offline framework. The heavy computation tasks are offloaded either to the cloud or to the offline phase, while only the lightweight operations are executed at the DU's device or the online phase, which makes the scheme computationally efficient. In order to preserve keyword privacy, we divide each keyword into a generic keyword name and a keyword value, and attach only the generic keyword names with the ciphertext and trapdoor. The cloud server is assigned with a cloud public key and cloud secret key pair. We make use of the cloud public key in trapdoor generation so that only the cloud server can perform test operation with its cloud secret key. In order to validate the cloud's test, transform and signature verification, the DU re-randomizes the trapdoor and the decryption key using a few random numbers. We have compared the achieved functionalities, and theoretical computation and communication costs with the existing schemes [1, 62]. Also, we compare the experimental outcomes with [1] regarding storage expenses and execution durations. The experimental results and theoretical analysis show that our scheme is computationally efficient compared to the existing ABSE schemes [1, 62].

In **Chapter 4**, we propose a communicationally efficient Boolean searchable attribute-based signcryption scheme for EMR storage and retrieval in the cloud computing environment. Along with achieving data and DO authenticity, DO anonymity, keyword privacy, and non-interactive search results verifiability, the proposed scheme reduces the size of the ciphertext significantly compared to the existing searchable signcryption schemes. We utilize a disjunctive normal form (DNF) en-

cryption policy which makes the scheme communicationally efficient. It enables the number of pairing computations in the transform algorithm to be independent of the number of encryption attributes in the encryption policy. To prevent KGAs on ciphertexts, we adopt the linear-splitting technique, which splits each keyword ciphertext component into two complementary randomized components and re-randomizes every keyword trapdoor component to match the split components in the ciphertext. To prevent KGAs on trapdoors, the cloud is allocated with a pair of public and secret keys. In order to validate the accuracy of the search results returned by the cloud, the DU makes use of the re-randomization technique on the trapdoor and decryption key components. The attained functionalities, as well as the theoretical computation and communication costs, were assessed in comparison to those of the relevant existing schemes [53, 66]. We found our scheme to be rich in functionality. Furthermore, an analysis of execution times and storage expenses is conducted to compare the experimental outcomes with those in [53, 66]. The theoretical analysis and experimental findings demonstrate that our scheme is communicationally efficient.

In **Chapter 5**, we have addressed the open problem presented by Ge et al. [25], which was to provide a new attribute-based proxy re-encryption with keyword search scheme (ABPRE-KS) for enabling more expressive keyword search. The schemes in Chapter 3 and Chapter 4 do not provide a data sharing mechanism along with a data storage and retrieval framework, where a DU can share the encrypted data sent by the DO with another user without decrypting it. Here, we have proposed a searchable attribute-based proxy re-encryption scheme that simultaneously provides data storage, data sharing, and data searching framework. Our scheme supports a keyword set updating mechanism, where a DU can modify the keyword set attached to the ciphertext prior to sharing it with another DU. To delegate the re-encryption task to the cloud server, the DU creates and assigns a re-encryption key to the server. After checking the validity of the re-encryption key and the original ciphertext, the cloud server re-encrypts and stores the re-encrypted ciphertext in it. To increase the search efficiency in the attribute-based proxy re-encryption (ABPRE) framework, the DU generates Boolean formula-based search query. Finally, we have compared the achieved functionalities, and theoretical computation and communication costs with the existing relevant schemes [25, 48]. The theoretical analysis along with achieved rich functionalities demonstrates the efficiency of our scheme.

Chapter 2

Cryptographic Preliminaries and Literature Survey

The objective of this chapter is twofold. First, we discuss the preliminary concepts that are necessary to understand the cryptographic primitives and their security proofs described in the forthcoming chapters. Second, the presentation of literature review, which covers the previous works that are most relevant to the main theme of this thesis.

2.1 Introduction

The cryptographic preliminaries commence with a description of key notations and terminologies necessary for the thesis to be presented in a clear and concise manner. Since all cryptosystems included in this thesis rely on bilinear maps (or pairings), we briefly review pairings, without giving the details of computation of such pairings on elliptic curve groups. Note that a pairing-based cryptosystem can be fully understood without any knowledge of elliptic curves. One can find an excellent exposition on elliptic curves and pairings in [61],[56]. A scheme is provably secure if breaking the scheme leads to a solution for a mathematical problem that is believed to be hard (no one can solve the problem in polynomial time). We discuss several problems that are assumed to be hard to solve in polynomial time. Security of various schemes presented in later chapters are based on the hardness of these problems. A few useful building blocks used in our proposed constructions will also be described briefly.

In literature review, we discuss the major functionality of some existing attribute-based cryptosystems (in Section 2.7) such as ABE schemes, attribute-based sig-

nature and signcryption schemes, attribute-based searchable encryption schemes, searchable attribute-based signcryption schemes, attribute-based proxy re-encryption schemes, proxy re-encryption with keyword search schemes. This survey is not exhaustive and covers only the schemes that are more relevant to the schemes we address to design in the subsequent chapters.

2.2 Bilinear Pairing

We provide a brief introduction to (cryptographic) bilinear pairing that would be useful for designing different kinds of secure cryptographic protocols, including the schemes proposed in this thesis. Bilinear pairing is originally used in cryptology to reduce the discrete logarithm problem on a certain class of elliptic curves over a finite field to the discrete logarithm problem on a finite field [17],[60].

Definition 1. *Let $\mathbb{G}$ and $\mathbb{G}_T$ be two multiplicative cyclic groups of prime order p. Also, let g be a generator of $\mathbb{G}$. The bilinear pairing tuple is defined as $\Sigma := \langle p, g, \mathbb{G}, \mathbb{G}_T, e \rangle$, where $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is a bilinear map satisfying three properties:*

1. *$e(h_1 \cdot h_2, k) = e(h_1, k) \cdot e(h_2, k)$ and $e(h, k_1 \cdot k_2) = e(h, k_1) \cdot e(h, k_2)$, for all $h, h_1, h_2, k, k_1, k_2 \in \mathbb{G}$.*
2. *$e(g, g) \neq 1$, where 1 denotes the identity element of $\mathbb{G}_T$.*
3. *e is efficiently computable in polynomial time.*

Remark 1. *e is symmetric as $e(g^a, g^b) = e(g, g)^{ab} = e(g^b, g^a)$, for all $a, b \in \mathbb{Z}_p$ (using property 1).*

Example 2.1. Modified Weil pairing ([7]) and Tate pairing ([17], [19]) functions on elliptic curves over finite fields give typical implementations of bilinear pairings.

2.3 Hardness Assumptions

This section presents five important hard problems: (i) Decisional Bilinear Diffie-Hellman (DBDH) problem [28], (ii) q-Diffie-Hellman Exponent (q-DHE) Problem [10], (iii) q-1 problem [35], (iv) q-2 problem [35], (v) Decisional Linear (DLin) Problem [13]. The security of our proposed constructions relies on the hardness of these problems.

2.3.1 DBDH Problem

Given a bilinear pairing tuple $\Sigma = (p, g, \mathbb{G}, \mathbb{G}_T, e)$ and the DBDH problem instance $\mathcal{Y} = \langle \Sigma, G_1 := g^{\phi_1}, G_2 := g^{\phi_2}, G_3 := g^{\phi_3}, Z \rangle$, where ϕ_1, ϕ_2, ϕ_3 chosen secretly from $\mathbb{Z}_p^*$, the task for an adversary $\mathscr{A}$ is to ascertain if Z is equal to $e(g,g)^{\phi_1\phi_2\phi_3}$ or Z has been selected randomly from $\mathbb{G}_T$. The advantage for any PPT adversary $\mathscr{A}$ is described as $\mathsf{Adv}_{\mathscr{A}}^{\mathsf{DBDH}}$
$\stackrel{\text{def}}{=} \big|\Pr[\mathscr{A}(\Sigma, G_1, G_2, G_3, Z{=}e(g,g)^{\phi_1\phi_2\phi_3}){=}0] - \Pr[\mathscr{A}(\Sigma, G_1, G_2, G_3, Z{=}\text{random}){=}0]\big|$
The DBDH assumption says $\mathsf{Adv}_{\mathscr{A}}^{\mathsf{DBDH}}$ is negligible for all PPT adversary $\mathcal{A}$.

2.3.2 q-DHE Problem

Given a bilinear pairing tuple $\Sigma = (p, g, \mathbb{G}, \mathbb{G}_T, e)$ and the q-DHE problem instance $\mathcal{Y} = \langle \Sigma, g, \{g^{\phi^i}\}_{i\in[2q], i\neq q+1} \rangle$, where ϕ chosen secretly from $\mathbb{Z}_p^*$, the task for an adversary $\mathcal{A}$ is to compute $g^{\phi^{q+1}}$. The advantage for any PPT adversary $\mathscr{A}$ is described as $\mathsf{Adv}_{\mathscr{A}}^{q-\mathsf{DHE}} \stackrel{\text{def}}{=} \Pr[g^{\phi^{q+1}} \leftarrow \mathscr{A}(\mathcal{Y})]$.

2.3.3 q-1 Problem

Given a bilinear pairing tuple $\Sigma = (p, g, \mathbb{G}, \mathbb{G}_T, e)$ and the q-1 problem instance

$$\begin{aligned}\mathcal{Y} \;=\; & \big\langle \Sigma, g, g^{\beta}, \{g^{\phi^i}, g^{\psi_j}, g^{\beta\psi_j}, g^{\phi^i\psi_j}, g^{\phi^i/\psi_j^2}\}_{(i,j)\in[q,q]}, \{g^{\phi^i/\psi_j}\}_{(i,j)\in[2q,q], i\neq q+1}, \\ & \{g^{\phi^i\psi_j/\psi_{j'}^2}\}_{(i,j,j')\in[2q,q,q], j\neq j'}, \{g^{\beta\phi^i\psi_j/\psi_{j'}}, g^{\beta\phi^i\psi_j/\psi_{j'}^2}\}_{(i,j,j')\in[q,q,q], j\neq j'}, Z \big\rangle,\end{aligned}$$

where $\beta, \phi, \psi_1, \ldots, \psi_q$ chosen secretly from $\mathbb{Z}_p^*$, the task for an adversary $\mathscr{A}$ is to ascertain if Z is equal to $e(g,g)^{\beta\phi^{q+1}}$ or Z has been randomly selected from $\mathbb{G}_T$. The advantage for any PPT adversary $\mathscr{A}$ is described as
$\mathsf{Adv}_{\mathscr{A}}^{q\text{-}1} \stackrel{\text{def}}{=} \big|\Pr[1 \leftarrow \mathscr{A}(\mathcal{Y}) | Z = e(g,g)^{\beta\phi^{q+1}}] - \Pr[1 \leftarrow \mathscr{A}(\mathcal{Y}) | Z = \text{random}]\big|$.

2.3.4 q-2 Problem

Given a bilinear pairing tuple $\Sigma = (p, g, \mathbb{G}, \mathbb{G}_T, e)$ and the q-2 problem instance

$$\begin{aligned}\mathcal{Y} \;:=\; & \big\langle \Sigma, g, g^{\phi_1}, g^{\phi_2}, g^{\phi_3}, g^{(\phi_1\phi_3)^2}, \{g^{\psi_i}, g^{\phi_1\phi_3\psi_i}, g^{\phi_1\phi_3/\psi_i}, g^{\phi_1^2\phi_3\psi_i}, g^{\phi_2/\psi_i^2}, g^{\phi_2^2/\psi_i^2}\}_{i\in[q]}, \\ & \{g^{\phi_1\phi_3\psi_i/\psi_j}, g^{\phi_2\psi_i/\psi_j^2}, g^{\phi_1\phi_2\phi_3\psi_i/\psi_j^2}, g^{(\phi_1\phi_3)^2\psi_i/\psi_j}\}_{(i,j)\in[q,q], i\neq j}, Z \big\rangle,\end{aligned}$$

where $\phi_1, \phi_2, \phi_3, \psi_1, \ldots, \psi_q$ chosen secretly from $\mathbb{Z}_p^*$, the task for an adversary $\mathscr{A}$ is to ascertain if Z is equal to $e(g,g)^{\phi_1\phi_2\phi_3}$ or Z has been randomly selected from $\mathbb{G}_T$.

The advantage for any PPT adversary $\mathscr{A}$ is described as
$\mathsf{Adv}_{\mathscr{A}}^{q\text{-}2} \stackrel{\text{def}}{=} \big|\Pr[1 \leftarrow \mathscr{A}(\mathcal{Y})|Z = e(g,g)^{\phi_1\phi_2\phi_3}] - \Pr[1 \leftarrow \mathscr{A}(\mathcal{Y})|Z = \text{random}]\big|$.

2.3.5 DLin Problem

Given a bilinear pairing tuple $\Sigma = (p, g, \mathbb{G}, \mathbb{G}_T, e)$ and the DLin problem instance $\mathcal{Y} := \big\langle \Sigma, g, g^{\phi_1}, g^{\phi_2}, g^{\phi_1\phi_3}, g^{\phi_2\phi_4}, Z \big\rangle$, where g is a random generator of $\mathbb{G}$, (unknown) $\phi_1, \phi_2, \phi_3, \phi_4$ chosen secretly from $\mathbb{Z}_p^*$, the task for an adversary $\mathscr{A}$ is to ascertain if Z is equal to $g^{\phi_3+\phi_4}$ or Z has been selected randomly from $\mathbb{G}$. The advantage for any PPT adversary $\mathscr{A}$ is described as
$\mathsf{Adv}_{\mathscr{A}}^{\mathsf{DLin}} \stackrel{\text{def}}{=} \big|\Pr[1 \leftarrow \mathscr{A}(\mathcal{Y})|Z = g^{\phi_3+\phi_4}] - \Pr[1 \leftarrow \mathscr{A}(\mathcal{Y})|Z = \text{random}]\big|$.

2.4 Building Blocks

2.4.1 Access Policy

An access policy over the attribute universe U is represented by a Boolean formula, in which attributes are linked together with the assistance of the logical operators AND ($\wedge$), OR ($\vee$), and NOT ($\neg$). However, in our scheme, we use Boolean formula consisting of AND ($\wedge$), OR ($\vee$) gates.

Let $\Gamma = \{(att_1 \vee att_2 \vee att_3) \wedge (att_4 \vee att_5) \wedge att_6\}$ be an access policy, where att_i's are attributes for all $i = 1, 2, 3, 4, 5$,
6. A set $A \subset U$ is authorized with respect to Γ if A satisfies Γ (in this case we write $A \models \Gamma$). Otherwise, A is said to be unauthorized with respect to Γ (in this case we write $A \not\models \Gamma$). Here the set $\{att_1, att_4, att_6\}$ is an authorized set and the set $\{att_1, att_2, att_6\}$ is an unauthorized set with respect to Γ.

If $A \models \Gamma$ implies $A_1 \models \Gamma$ for all $A_1 \supset A$, then Γ is said to be monotone access policy. From now on, unless stated otherwise, by an access policy we mean a monotone access policy.
The ordered tuple $(\mathcal{K}, \psi)$ is said to be a monotonic span program (MSP), where $\mathcal{K}$ is a matrix of dimension $\ell \times n$ with entries from a field $\mathbb{F}$ and $\psi : [\ell] \to U$ is a function that labels rows of $\mathcal{K}$. If a MSP $(\mathcal{K}, \psi)$ represents a monotone access policy Γ, we denote it as $\Gamma = (\mathcal{K}, \psi)$. In this case, $A \models \Gamma$ (or $\Gamma(A) = \text{true}$) implies that there exists $\vec{\omega} := (\omega_1, \omega_2, \ldots, \omega_\ell) \in \mathbb{F}^\ell$ such that $\vec{\omega} \cdot \mathcal{K} = \vec{1}_n$ with $\omega_i = 0$, for all i satisfying $\psi(i) \notin A$. $A \not\models \Gamma$ (or $\Gamma(A) = \text{false}$) implies that there exists $\vec{b} := (b_1, b_2, \ldots, b_n) \in \mathbb{F}^n$ such that $\vec{b} \cdot \mathcal{K}^{(i)} = 0$, for all i satisfying $\psi(i) \in A$, for any matrix $\mathcal{K}$, the ith row is denoted as $\mathcal{K}^{(i)}$.

Disjunctive Normal Form (DNF). Our methodology employs the disjunctive normal form (DNF), a particular type of the Boolean formula. If $B_i \subset U$ for all $i \in [m]$ and the attributes in U are represented by u, then a DNF Boolean formula can be expressed as

$$\Gamma = \big(\wedge_{u \in B_1} u \big) \vee \big(\wedge_{u \in B_2} u \big) \vee \cdots \vee \big(\wedge_{u \in B_m} u \big), \text{where } m \in \mathbb{N}$$

In short, $\Gamma = B_1 \vee B_2 \vee \cdots \vee B_m$. Let $A \subset U$ be an arbitrary set. Then

$$A \models \Gamma \Leftrightarrow A \supseteq B_i, \quad \text{for some i} \in [m]$$

If y be an attribute in U, then $\Gamma \wedge y = (B_1 \wedge \{y\}) \vee (B_2 \wedge \{y\}) \vee \cdots \vee (B_m \wedge \{y\})$. Therefore, if $y \notin A$, $A \not\models \Gamma \wedge y$.

2.4.2 Linear Secret-Sharing Scheme

Let $\mathcal{K}$ be a share-generating matrix of size $\ell \times n$ and ψ be a row labeling function that maps rows of $\mathcal{K}$ to attributes in access policy Γ. A Linear Secret-Sharing Scheme (LSSS) for the MSP $\Gamma := (\mathcal{K}_{\ell \times n}, \psi)$ is comprised of the subsequent two polynomial time procedures.

1. $\mathsf{Share}(\mathcal{K}, \psi, \alpha)$. To distribute shares of a secret $\alpha \in \mathbb{Z}_p^*$, it chooses $w_2, w_3, \ldots, w_n \xleftarrow{\text{u}} \mathbb{Z}_p^*$ and sets $\lambda_i := \vec{\mathcal{K}}^{(i)} \cdot \vec{w}$, where $\vec{w} := (\alpha, w_2, w_3, \ldots, w_n)$. Finally, it assigns the ith share λ_i to the ith row.

2. $\mathsf{Reconstruct}(\mathcal{K}, \psi, A)$. To rebuild the shared secret α, it takes $(\mathcal{K}, \psi)$ and a set A of attributes as input. If $A \models \Gamma$, it generates a secret reconstruction vector $\vec{z} := (z_1, z_2, \ldots, z_\ell) \in \mathbb{Z}_p^\ell$ satisfying $\sum_{i \in [\ell]} z_i \cdot \vec{\mathcal{K}}^{(i)} = \vec{1}_n$ and $z_i = 0$, whenever $\psi(i) \notin A$. Let $I := \{i | \psi(i) \in A\}$, then $\sum_{i \in I} z_i \lambda_i = \alpha$ if $A \models \Gamma$. If $A \not\models \Gamma$, it outputs $\perp$.

Remark 2. *Using the Gaussian elimination method, the constants z_i can be calculated in polynomial time in the size of the matrix $\mathcal{K}$.*

Given a valid set $\{\lambda_i : i \in I\}$ of secret shares and computed reconstruction vector $\vec{a}$, the secret α can be recovered as follows.

Since $A \models \Gamma$, we get $\vec{z} \cdot \mathcal{K} = \vec{1}_n$. Consider $I = \{i \in [\ell] : z_i \neq 0\}$. Then

$$\sum_{i \in I} z_i \lambda_i = \sum_{i \in I} z_i \vec{\mathcal{K}}^{(i)} \cdot \vec{w} = \vec{w} \cdot \left(\sum_{i \in I} z_i \vec{\mathcal{K}}^{(i)} \right) = \vec{w} \cdot (\vec{z} \cdot \mathcal{K}) = (\alpha, w_2, w_3, \ldots, w_n) \cdot \vec{1}_n = \alpha$$

Convertion of Access policy into LSSS Matrix

The following section outlines a useful approach for transforming Access policy into an LSSS Matrix. A binary access tree can be used to represent it, with the leaf nodes representing the characteristics and the inside nodes representing the operator name AND ($\wedge$) or OR ($\vee$) gates.

- Let a binary access tree T represents an access policy.
- Start by assigning the root node r of the access tree T with the vector $\vec{v}_r = (1)$, which is a vector of length 1.
- Make the global variable C's initial value 1 to count. Next, we denote as u_y to every node y with a vector u_y by moving down the levels of T as follows:
  - If $y = \vee$, then
    1. $\vec{u}_{\text{left }(y)} = \vec{u}_{\text{right}(y)} = \vec{u}_y$, where left (y) and right (y) are left and right childs of the node y, respectively.
    2. $x = x + 0$.
  - If $y = \wedge$, then
    1. $\vec{u}_{\text{left }(y)} = \left(\vec{u}'_y, 1\right)$, where $\overrightarrow{u'}_y = \left(\vec{u}_y, \overrightarrow{0}\right)$, $\overrightarrow{0}$ being a vector of length zero, $x - |\vec{u}_y|$. Here, $|\vec{u}_y|$ is the length of the vector $\vec{u}_y$. Note that if $x = |\vec{u}_y|$, then $\overrightarrow{u'}_y = \vec{u}_y$.
    2. $u_{\text{right }(y)} = (\overrightarrow{0}, -1)$, where $\overrightarrow{0}$ is the zero vector of length x.
    3. $x = x + 1$.

Example 2.2. Let a, b, c, d, e, f be attributes and consider the access policy $\Big(\big(a \vee b \vee c\big) \wedge (d \vee e)\Big) \wedge f$. The equivalent access tree is given in Figure 2.1. Converting the above boolean formula we got the following matrix

$$\mathcal{K} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & -1 \\ 0 & 0 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

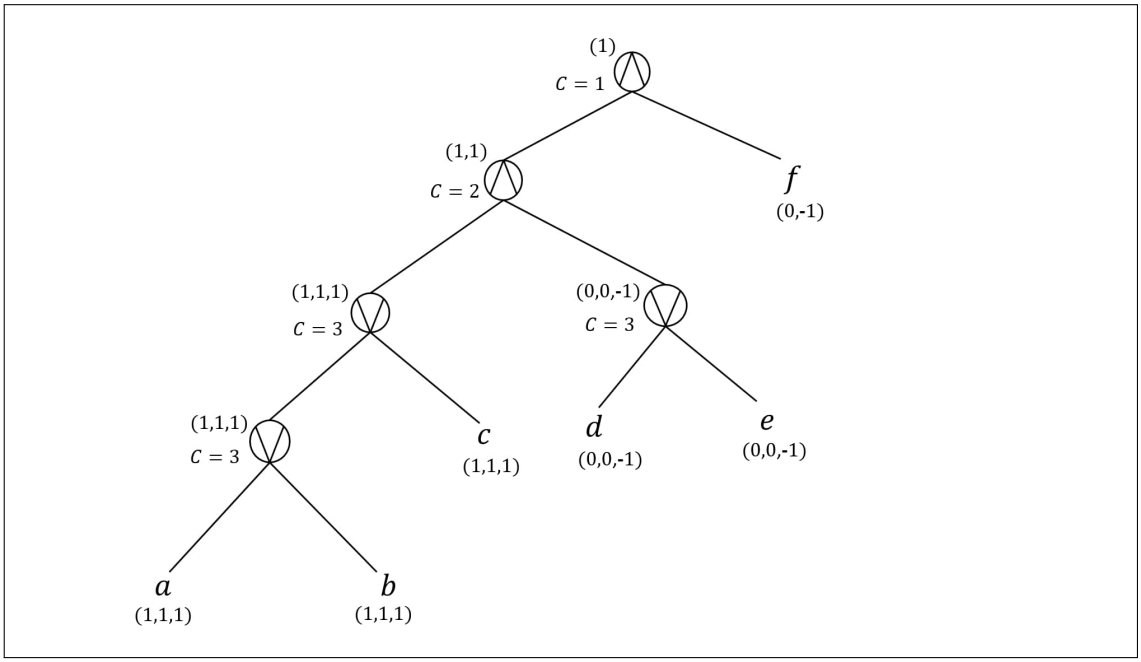


Figure 2.1: Access Tree for the Access Policy $\Big(\big(a \vee b \vee c\big) \wedge (d \vee e)\Big) \wedge f$

Let $p = 13$ and $\alpha, z_2, z_3 \in \mathbb{Z}_{13}$
and $\rho : \{1,2,3,4,5,6\} \to \{a,b,c,d,e,f\}$ be the row labeling function defined by $\rho(1) = a, \rho(2) = b, \rho(3) = c, \rho(4) = d, \rho(5) = e, \rho(6) = f$.
Consider the secret $\alpha = 11$ and $\vec{v} = (\alpha, z_2, z_3\ ) = (11, 9, 2)$
Secret share of $a : \vec{\mathcal{K}}^{(1)} \cdot \vec{v} = (1,1,1) \cdot (11,9,2) = 22 \bmod 13 = 9$
Secret share of $b : \vec{\mathcal{K}}^{(2)} \cdot \vec{v} = (1,1,1) \cdot (11,9,2) = 22 \bmod 13 = 9$
Secret share of $c : \vec{\mathcal{K}}^{(3)} \cdot \vec{v} = (1,1,1) \cdot (11,9,2) = 22 \bmod 13 = 9$
Secret share of $d : \vec{\mathcal{K}}^{(4)} \cdot \vec{v} = (0,0,-1) \cdot (11,9,2) = -2 \bmod 13 = 11$
Secret share of $e : \vec{\mathcal{K}}^{(5)} \cdot \vec{v} = (0,0,-1) \cdot (11,9,2) = -2 \bmod 13 = 11$
Secret share of $f : \vec{\mathcal{K}}^{(6)} \cdot \vec{v} = (0,-1,0) \cdot (7,9,2) = -9 \bmod 13 = 4$
Let $L = \{c, e, f\}$ be a set of valid users satisfying the access policy: $\big(((a \vee b) \vee c) \wedge (d \vee e)\big) \wedge f$. So $I = \{i \in [6] \mid \rho(i) \in L\} = \{3, 5, 6\}$. To construct the secret α for L, set $\vec{a} = (0, 0, a_3, 0, a_5, a_6)$ such that $\vec{a} \cdot \mathcal{K} = (1, 0, 0)$, i.e., $\sum_{i \in [\ell]} a_i \cdot \vec{\mathcal{K}}^{(i)} = \vec{1}$ and get the following system of equations.

$$\begin{aligned} a_3 &= 1 \\ a_3 - a_6 &= 0 \\ a_3 - a_5 &= 0 \end{aligned}$$

We get $a_3 = a_5 = a_6 = 1$, solving the above equations.

Finally, we get the secret by the equation $\sum_{i \in I} a_i \lambda_i = \alpha$.
Hence, $\alpha = 0 \times 9 + 0 \times 9 + 1 \times 9 + 0 \times 11 + 1 \times 11 + 1 \times 4 = 9 + 11 + 4 = 11$.

2.5 Key Derivation Function

If a function that outputs a cryptographically secure secret key, given an initial keying material as input, then the function is called a key derivation function [38]. Upon receiving an initial secret key (SK_I) as input, the key derivation function $\mathsf{KDF} : SK_I \rightarrow \{0,1\}^\kappa$ produces a secret key string with κ bits of length.

Definition 2. *The key derivation function KDF with the output length κ, is secure if for any PPT adversary $\mathcal{A}$, $\left|\Pr[\mathcal{A}(\mathsf{KDF}(\mathcal{I})) = 1] - \Pr[\mathcal{A}(R) = 1]\right|$ is negligible, where $\mathcal{I}$ is an initial secret key recieved from keying source and R is chosen uniformly at random from $\{0,1\}^\kappa$.*

2.6 Hash Function

In general, a hash function is a (deterministic) mapping that takes bit strings of any length and compress them into bit strings of a fixed length. For practical applications, hash functions should be easy to compute, i.e., given a string x, computing the hash of x should be feasible in time polynomial in the size of x.

Let $H : X \rightarrow Y$ be a hash function. The domain X of H could be infinite or finite, but the range Y is always a finite set. If X is finite, it is assumed that $|X| \geq |Y|$ (or $|X| \geq 2|Y|$ which is stronger condition). In case X is infinite, by the pigeon-hole principal, there must exist distinct $x_1, x_2 \in X$ with $H(x_1) = H(x_2)$. The pair (x_1, x_2) is called a collision for the hash function H. For cryptographic uses, the essential requirement is that it is "hard" to find such collisions. The following is a desirable property of hash functions utilized in cryptography.
Collision resistant: a hash function H is collision resistant if it is infeasible for any PPT algorithm to find $x_1, x_2 \in X$ with $x_1 \neq x_2$ such that $H(x_1) = H(x_2)$.

2.7 Literature Survey

2.7.1 Attribute-Based Encryption

Attribute-based cryptography, a branch of public-key cryptography, has been an active area of research in recent years. The concept of ABE was introduced by Sahai

and Waters [76]. Every user in this system has their own unique set of attributes that act as their public key. The user secret key is created by incorporating an access policy over user's attributes and the ciphertext is generated based on a set of data receivers' attributes. This kind of encryption is called the KP-ABE [76, 29, 73]. The dual mechanism is called CP-ABE [4, 85] wherein the user receives a secret key for his attribute set and an access policy is involved in ciphertext generation. With either architecture, decryption will work as long as the list of attributes matches the access policy. In this thesis, we consider CP-ABE mechanism since it is more flexible to realize fine-grained access control over encrypted data. With the goal of enhancing the security and efficiency, diverse ABE frameworks have been constructed in [43, 33, 32, 74, 36].

2.7.2 ABE with Outsourced Decryption

The standard ABE designs suffer from huge computation cost in both encryption and decryption algorithms, that usually grows with the number of attributes used in the process. To make the encryption process lightweight, Hohenberger and Waters [36] split the process into offline encryption and online encryption. In the former, expensive operations (e.g., exponentiation, pairing) are performed while the latter utilizes only light computations such as hashing, XORing, modular addition, modular multiplication etc. In an attempt to reduce the decryption cost at DU's end, Green et al. [30] framed a mechanism in which one can delegate the decryption process to third-party decryption service providers (e.g., cloud service provider). And, the cloud server converts the original ciphertext into another form so that the DU needs to perform quite a few operations to obtain the plaintext, resulting in the DU is able to recover the plaintext very efficiently. Later, Lai et al. [40] identified that the method suggested in [30] is not feasible to verify the correctness of the transformed ciphertext returned by the cloud, and designed an ABVOD. A formal security model for verifiability is also formulated in [40]. The technique used in [40] introduces significant computation and communication overhead in encryption process. Following, several ABVOD primitives with different features (like low computation cost, short ciphertext size, outsourced key-issuing, correctness of partially decrypted ciphertexts for the unauthorized users) have been proposed in [45, 44, 51, 68, 59], some of them are generic.

2.7.3 Attribute-Based Signature

Maji et al. [57] established the concept of ABS alongside the advancement of ABE. ABS has been utilized in several contexts such as attribute-based messaging, attribute-based authentication and trust-negotiation, the disclosure of secret [57], and the implementation of an attribute-based anonymous credential system [78], etc. In general, ABS schemes are classified as

(i) Signature-Policy ABS: In the Signature-Policy ABS [57, 10] design, a set of attributes is used to construct the signing key, and a signing policy that is met by the signer's attribute set is used to sign a message.

(ii) Key-Policy ABS: In Key-Policy ABS [71] design, the signing key is computed according to an signing policy over signers attributes and a message is signed with an attribute set satisfying the signing policy associated with the signing key.

In Signature-Policy ABS, the specific set of signing attributes used to form the signature cannot be determined. One can only have the knowledge that a signer with a set of attributes that meet the signing policy has signed the message. This feature is called *signer anonymity*. In Key-Policy ABS, the signer privacy guarantees that the signature of a message for the attribute set A does not disclose any information about the signing policy $\mathcal{P}$ of the signer except the fact that $\mathcal{P}$ is satisfied by A.

2.7.4 Attribute-Based Signcryption

Combining the ABE and ABS, ABSC scheme realizes data and DO authenticity and, fine-grained data access control simultaneously. Various ABSC constructions have been suggested to improve efficiency, security, and expressiveness since Gagné et al. [18] introduced the study of signcryption in an attribute-based architechture [15, 10, 67, 70, 69]. To facilitate the sharing of PHRs in the cloud, Rao subsequently developed a Ciphertext-Policy ABSC (CP-ABSC) in [70] that, in comparison to the CP-ABSC schemes [10, 15, 67], displays a small ciphertext-size. The CP-ABSC [67] achieves non-selective security in composite-order bilinear group setting wherein the group order is a product of three primes. Composite-order group constructions are not very practical in the sense that they impose high computation burden on the system. Recently, Rao [69] has proposed the concept of an online-offline architecture within ABSC for the purpose of implementing a lightweight signcryption. In order to reduce the computation overhead of unsigncryption algorithm, outsourced

ABSC schemes [14, 2] have been suggested, building upon the outsourcing decryption technique developed by Green et al. [30]. With the help of certain ciphertext components, the DU may confirm the accuracy of the converted ciphertext using the technique suggested in [14], which is an adaptation of the ABSC [70]. Belguith et al. ABSC [2] allows for access policy updates without the need to re-issue users' secret keys or re-signcryption of cipherext. In order to verify that the converted ciphertext acquired in [2] is accurate, the DU must communicate with a semi-trusted edge server.

2.7.5 Attribute-Based Searchable Encryption

To efficiently search over encrypted data, Boneh et al. [6] introduced public key encryption with keyword search, wherein a storage server, not aware of the underlying plaintext or the keyword, verifies if a keyword selected by the DU is same as the keyword linked to the ciphertext. The combination of ABE and searchable encryption, called ABSE, provides data confidentiality, fine-grained access control, and data retrieval mechanism simultaneously. In [92, 81], for the first time the search functionality is incorporated in ABE framework. The DU queries the cloud by sending a trapdoor corresponding to a keyword and obtains back partially decrypted matching ciphertexts. ABSE can be broadly classified into three groups (i) ABSE supporting single keyword search [49, 26, 88, 54, 64, 63, 62], (ii) ABSE supporting multi-keyword search [83, 91, 1, 77, 87, 62] and (iii) ABSE supporting keyword policy search [12]. In [49, 26], the authors constructed ABSE with keyword update and ciphertext re-encryption in key-policy and ciphertext-policy setting, respectively. Although the ABSE [88] supports user revocation and traitor tracing functionalities, it fails to resist KGAs. The scheme [83] focuses on outsourcing key generation, encryption and decryption tasks whereas the verification of the partially decrypted ciphertext returned by the cloud server is ignored. Less expressive AND gate access policy is used in [91] to encrypt a data document and a conjunctive keyword search policy is employed to generate a trapdoor. In [12], Cui et al. suggested a generic construction of ABSE that simultaneously supports fine-grained data access control, keyword privacy and expressive data searching. But it may not resist KGAs.

2.7.6 Searchable Attribute-Based Signcryption

By integrating the key-policy ABS from [72], the CP-ABE from [70], and the keyword search scheme from [92], a searchable ABSC with single keyword search is presented in [53]. Combining the KP-ABSC [72] with the single keyword search framework [92], another ABSC with single keyword search is provided in [52]. Based on the threshold-policy CP-ABE [21] and ABS [20], and B+tree-based data retrieval framework, Obiri et al. [66] designed a single keyword searchable ABSC scheme to share EMRs with intended DUs. The massive computational overhead of unsigncryption in [66] is left to DUs. Varri et al. [82] suggested a multi-keyword searchable ABSC in cloud storage employing a multi-dimensional B+-tree framework, in which DUs obtain search trapdoors from DOs. The workload of DUs is heavy and the scheme cannot offer signer anonymity and search results verification functionalities. Recently, Bera et al. [3] devised a novel secure, lightweight online-offline keyword policy searchable ABSC for cloud environments.

2.7.7 Attribute-Based Proxy Re-Encryption

In order to effectively communicate the encrypted data with other DUs, Mambo and Okamoto [58] proposed proxy re-encryption (PRE), where a proxy with a re-encryption key can transform original ciphertext to another ciphertext of the same message without knowing the original message. Later, Liang et al. [50] proposed a PRE in attribute-based setting, known as ABPRE. In [55], Luo et al. suggested a ciphertext-policy ABPRE (CP-ABPRE) scheme supporting a less expressive AND gate access policy with positive and negative attributes. Later, Liang et al. [46, 47] introduced different types of CP-ABPRE schemes with improved security. Later, two key policy ABPRE (KP-ABPRE) schemes were proposed by Ge et al. [23, 27], which are selective and adaptive model secure, respectively. In 2021, Ge et al. [22] introduced verifiability and fairness in CP-ABPRE scheme. Recently, the direct user revocation mechanism is introduced in ABPRE framework by Ge et al. [24].

2.7.8 Proxy Re-Encryption with Keyword Search

Shao et al. [79] initiated proxy re-encryption with keyword search (PREKS), which enables a sender to transfer keyword search capacity to new users. Later, various PREKS schemes were introduced in [89, 16]. However, all of the schemes [79, 89, 16] support single keyword search. Later, Wang et al. [84] introduced conjunctive

keyword search function to increase the search efficiency. Shi et al. [80] introduced PREKS scheme in attribute-based setting. In order to facilitate keyword updates and ensure chosen ciphertext attack (CCA) security in the random oracle model, a KP-ABPRE with keyword search scheme was developed in [48]. However, the scheme [48] does not support independent token generation framework. Later, a searchable CP-ABPRE scheme was proposed by Ge et al. [25], which supports independent token generation framework. Prior to this work, a searchable KP-ABPRE scheme [37] was introduced, which is IND-CPA secure, but it does not support search result verfication mechanism. However, all the schemes [80, 48, 25, 37] support less efficient single keyword search framework.

2.8 Chapter Summary

This chapter can be viewed as an integration of two parts: (i) cryptographic preliminaries (Sections 2.2 to 2.6) and (ii) study of literature (Section 2.7). First part provides the necessary tools for understanding of the topics given in the next chapters, including pairings, hard problems, access structures, LSSS, hash functions and KDF. In the second part, we concentrate on providing appropriate related work pertaining to the primitives presented in the later chapters.

Chapter 3

Attribute-Based Verifiable Data Storage and Retrieval Scheme in Cloud Computing Environment

In this chapter, we propose a secure lightweight Attribute-Based verifiable Data Storage and data Retrieval Scheme (ABDSRS) for cloud environments that attains the following features: (i) lightweight design, (ii) provably secure, (iii) fine-grained data access control, (iv) DO anonymity, (v) data and DO authenticity, (vi) keyword policy search over encrypted data, (vii) keyword privacy, and (viii) search results verification. ABDSRS employs attribute-based online-offline mechanism in which only authorized DOs can anonymously upload data to the cloud. And, a DU can search over encrypted data using keyword policy. ABDSRS enables a DU to independently check the precision of search outcomes acquired from the cloud. ABDSRS is lightweight in the sense that the heavy computations are offloaded either to the cloud or to offline phase, while only lightweight operations are executed at the DU device. We formalize more general security definitions of ABDSRS by considering various possible adversarial capabilities and present rigorous security analysis. We also conduct experiments to evaluate ABDSRS's performance.

The work presented in this chapter is based on our published research article given below.
Sourav Bera, Suryakant Prasad, Y Sreenivasa Rao, Ashok Kumar Das, and Youngho Park. Designing attribute-based verifiable data storage and retrieval scheme in cloud computing environment. *Journal of Information Security and Applications*, vol. 75, pp. 103482, Elsevier, 2023. https://doi.org/10.1016/j.jisa.2023.103482

3.1 Introduction

Cloud computing is a new paradigm for computing and storage that allows people and organizations to store data, share it with specific users, and retrieve it when needed. By offering adaptable, affordable, and high-quality services, it significantly enhances peoples' capacities for data sharing, storage, and retrieval. To effectively deal with issues with data security and privacy, it is essential to establish verified storage for data, implement fine grained data access control, provide efficient data searching mechanism, and verify the accuracy of search outcomes. However, simultaneously accomplishing the aforementioned functionalities is quite difficult.

An insecure framework [26] may arise from a straightforward combination of attribute-based keyword policy search mechanism, ABVOD, and ABS. Specifically, the framework may be vulnerable to KGAs and CCA. Moreover, it is much more difficult to realize the verifiability mechanism within such architechtures. This requirement arises from the fact that the DU needs to confirm that three cloud-based operations-search, transform, and verifying the signature-are performed correctly. Within ABVOD, the DU only validates the transform algorithm executed by the cloud. It is far from simple to combine ABE and ABS into one basic. Several ABSC mechanisms have been shown to be vulnerable in [72, 70].

In order to address the technical challenges being identified, we propose a secure lightweight ABDSRS for cloud environments by adapting the techniques of [8, 13, 42, 69, 90] and introducing some new technical ideas. Our ABDSRS is a novel secure lightweight online-offline attribute-based policy searchable signcryption cryptosystem equipped with the entities: DO, the Trapdoor Generation Center (TGC), DU, the cloud, and the Key Generation Center (KGC), that supports simultaneously outsourcing unsigncryption, correctness verification of search results and keyword privacy.

Briefly, in ABDSRS, a signature key and a decryption key are assigned to the DO and the DU respectively, by KGC. The DO uses signcryption to secure the confidential data by using an encryption policy, signature policy, and a collection of keywords. The resulting ciphertext is then sent to the cloud for storage. The cloud accepts the ciphertext if the DO is authorized. In order to access necessary data from the cloud, a DU first obtains a trapdoor from TGC for a keyword policy. Subsequently, the cloud receives a data retrieval request generated by the DU. The test algorithm is carried out by the cloud, which finds matching ciphertexts after getting the data retrieval request. After that, the cloud creates the "transformed

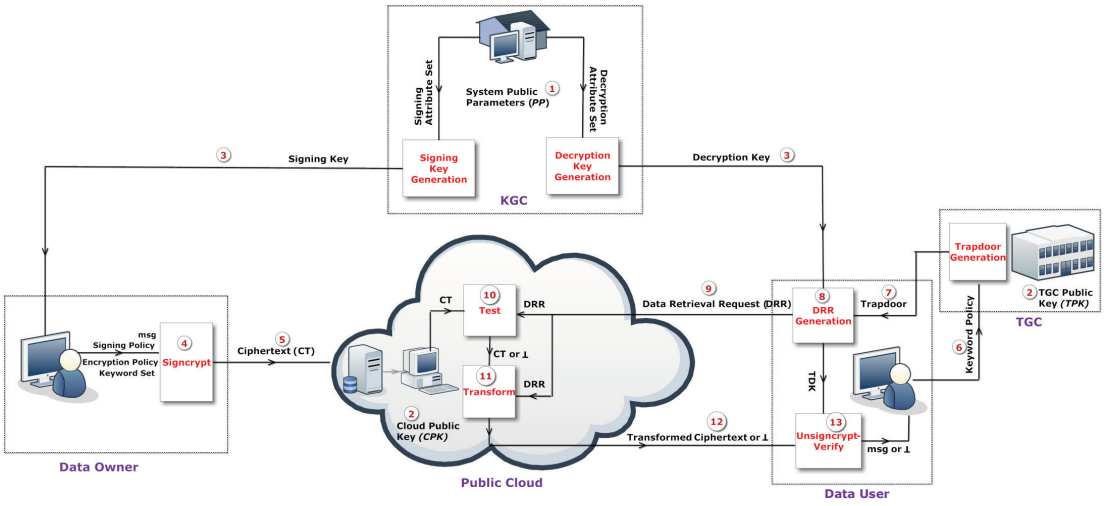


Figure 3.1: Architecture of ABDSRS

ciphertexts", which are matching ciphertexts that have been partially decrypted. Afterwards, the DU receives the transformed ciphertexts from the cloud. Finally, the DU validates the accuracy of the tasks executed by the cloud and restores the original data using lightweight calculations (the detailed model of ABDSRS is presented in Section 3.2.1).

Chapter Organization. The rest of the chapter is organized as follows. In Section 3.2, we formally define proposed ABDSRS and its security notions. Section 3.3 presents the ABDSRS's comprehensive construction. The security analysis of ABDSRS is provided in Section 3.4. Section 3.5 then addresses ABDSRS's performance. The chapter is concluded in Section 3.6.

3.2 Security of ABDSRS

3.2.1 System Model

The five entities of ABDSRS's architecture are shown in Figure 3.1: DO, KGC, TGC, DU and cloud.

(i) **KGC.** It is a completely trusted entity that is responsible for assigning a signing key and a decryption key to DO and DU, respectively.

(ii) **DO.** It signcrypts its personal message using an encryption policy, a signing policy and a keyword set and generates the ciphertext.

(iii) **TGC.** The creation of a trapdoor for the keyword policy obtained from the DU is the responsibility of this trustworthy entity. Next, the produced trapdoor is

sent to the relevant DU using a secure connection.

(iv) **Cloud.** It offers data storage and retrieval services. After receiving the ciphertext outsourced by DO, it checks the integrity of the ciphertext and stores it. Also, when receiving a data retrieval request from DU, it first locates all the matching ciphertexts (if the keyword set in ciphertext satisfies the keyword policy attached to the data retrieval request, then those ciphertexts are called matching ciphertexts) by performing the test operation. Then, it executes the transform operation to create partially unsigncrypted matching ciphertexts called transformed ciphertexts and sends them to DU.

(v) **DU.** It can obtain the stored ciphertext in the cloud based on its requirements. First, it generates a data retrieval request using the trapdoor, sends it to the cloud, and finally obtains the corresponding transformed ciphertext from the cloud. Next, it validates the accuracy of the transformed ciphertext returned by the cloud and finally gets back the original data by unsigncrypting the transformed ciphertext.

3.2.2 Security Models

In this section, we formally define the security of ABDSRS in terms of data confidentiality, data unforgeability, DO privacy, verifiability and keyword privacy. Based on the system model presented above, our proposed ABDSRS (for the ease of understanding, we present in Table 4.1 the notations used in ABDSRS) with signing attribute[1] universe U_s, encryption attribute[2] universe U_e and keyword universe U_t (let U be the attribute universe such that $U = U_s \cup U_e \cup U_t$) that supports signing policy $\mathcal{P}_s$ over U_s, encryption policy $\mathcal{P}_e$ over U_e and keyword policy $\mathcal{P}_t$ over U_t with the message space $\mathcal{M}$ involves five entities: KGC, DOs, cloud, TGC and DUs, and consists of the following seven phases.

(1) *System Initialization.* First, KGC generates the system public parameters $\mathcal{PP}$ and the system master secret key $\mathcal{MK}$ by running KGC-Setup algorithm with the input security parameter $1^{\wp}$, and initializes the system by announcing $\mathcal{PP}$. $\mathcal{MK}$ is kept secret by KGC. Next, by taking $\mathcal{PP}$ as input, TGC and cloud create their public and secret key pair respectively $[\mathcal{TPK}, \mathcal{TSK}]$ and $[\mathcal{CPK}, \mathcal{CSK}]$ by executing TGC-Setup and Cloud-Setup algorithms. They make $\mathcal{TPK}$ and $\mathcal{CPK}$ public for every entity in the system while $\mathcal{TSK}$ (resp. $\mathcal{CSK}$) is known only to TGC (resp. the cloud).

[1]We will use signing attribute and DOs attribute interchangeably.
[2]We will use encryption attribute, decryption attribute and DUs attribute interchangeably.

Table 3.1: Notations used in our ABDSRS

KGC (resp. TGC)	:	key (resp. trapdoor) generation center
$\mathcal{PP}$ (resp. $\mathcal{MK}$)	:	system public parameters(resp. system master secret key)
$\mathcal{TPK}$ (resp. $\mathcal{TSK}$)	:	TGC public key (resp. secret key)
$\mathcal{CPK}$ (resp. $\mathcal{CSK}$)	:	cloud public key (resp. secret key)
$\mathcal{SK}_{A_s}$	:	signing key for signing attribute set A_s
$\mathcal{DK}_{A_d}$	:	decryption key for the decryption attribute set A_d
$\mathcal{DRR}$	:	data retrieval request
$\mathcal{TDK}$	:	secret transformation decryption key
$\mathcal{IC}$ (resp. $\mathcal{IT}, \mathcal{IDR}$)	:	intermediate ciphertext (resp. trapdoor, data retrieval request)
msg	:	data file or plaintext
$\mathcal{P}_e$ (resp. $\mathcal{P}_s$, $\mathcal{P}_t$)	:	encryption (resp. signing, keyword) policy
$\mathcal{CT}$	:	ciphertext for $\mathcal{P}_s, \mathcal{P}_e$, keyword set W
$\mathcal{CT}_{out}$	:	transformed ciphertext
$\mathcal{TK}_{A_d}$	:	transform key for the decryption attribute set A_d derived from $\mathcal{DK}_{A_d}$
$\widetilde{\mathcal{TD}}_{\mathcal{P}_t}$	:	trapdoor for $\mathcal{P}_t$ returned by TGC
$\mathcal{TD}_{\mathcal{P}_t^\circ}$	:	transform trapdoor for $\mathcal{P}_t$ derived by DU from $\widetilde{\mathcal{TD}}_{\mathcal{P}_t}$

- KGC-Setup$(1^\wp) \rightarrow [\mathcal{PP}, \mathcal{MK}]$.
- TGC-Setup$(\mathcal{PP}) \rightarrow [\mathcal{TPK}, \mathcal{TSK}]$.
- Cloud-Setup$(\mathcal{PP}) \rightarrow [\mathcal{CPK}, \mathcal{CSK}]$.

(2) *DO and DU Key Generation.* In this phase, KGC generates and issues signing key to DO and decryption key to DU, by running sKeyGen and dKeyGen algorithms, respectively.

- sKeyGen$(\mathcal{PP}, \mathcal{MK}, A_s) \rightarrow \mathcal{SK}_{A_s}$. Taking $\mathcal{PP}$, $\mathcal{MK}$ and a signing attribute set $A_s \subset U_s$, this algorithm produces a signing key $\mathcal{SK}_{A_s}$.
- dKeyGen$(\mathcal{PP}, \mathcal{MK}, \mathcal{CPK}, A_d) \rightarrow \mathcal{DK}_{A_d}$. Given $\mathcal{PP}, \mathcal{MK}$, a decryption attribute set $A_d \subset U_e$ and $\mathcal{CPK}$, this algorithm returns a decryption key $\mathcal{DK}_{A_d}$.

(3) *Signcryption.* The DO executes this algorithm. In the offline phase offSigncrypt, DO prepares the intermediate ciphertext $\mathcal{IC}$, which will be used in the online phase. Then, once $\mathcal{IC}$ and the plaintext msg becomes accessible, the online phase onSigncrypt produces the final ciphertext $\mathcal{CT}$.

- offSigncrypt$(\mathcal{PP}, \mathcal{SK}_{A_s}, \mathcal{TPK}, \mathcal{CPK}, \mathcal{P}_s) \rightarrow \mathcal{IC}$. On input $\mathcal{PP}, \mathcal{SK}_{A_s}, \mathcal{TPK}$, $\mathcal{CPK}$, and a signing policy $\mathcal{P}_s$ such that $\mathcal{P}_s(A_s) = true$, this algorithm creates the intermediate ciphertext $\mathcal{IC}$.
- onSigncrypt$(\mathcal{PP}, \mathcal{IC}, msg, \mathcal{P}_e, W) \rightarrow \mathcal{CT}$. Taking $\mathcal{PP}, \mathcal{IC}$, a set W of keywords, a plaintext $msg \in \mathcal{M}$ and an encryption policy $\mathcal{P}_e$, this algorithm

produces a ciphertext $\mathcal{CT}$. The set W° (of keywords that includes only generic names) and the policies $\mathcal{P}_e, \mathcal{P}_s$ will be incorporated in $\mathcal{CT}$. Let $W := \{[\mathcal{W}_1 : w_1], \ldots, [\mathcal{W}_\varsigma : w_\varsigma]\}$, where $\mathcal{W}_i$ represents the generic keyword name and w_i represents the associated keyword value. In this case, $W^\circ := \{\mathcal{W}_1, \ldots, \mathcal{W}_\varsigma\}$.

(4) *Keyword Policy Trapdoor Generation.* This algorithm is executed by the TGC. In the offline phase offTrapGen, TGC precomputes the intermediate trapdoor $\mathcal{IT}$, which will be utilized in the online phase. Next, once $\mathcal{IT}$ is ready to be used and the keyword policy $\mathcal{P}_t$ is received, the online phase onTrapGen creates and sends the final trapdoor $\widetilde{\mathcal{TD}}_{\mathcal{P}_t}$ to the DU.

- offTrapGen$(\mathcal{PP}, \mathcal{CPK}, \mathcal{TSK}) \rightarrow \mathcal{IT}$. Taking $\mathcal{PP}$, $\mathcal{CPK}$ and $\mathcal{TSK}$ as input, this algorithm outputs $\mathcal{IT}$.
- onTrapGen$(\mathcal{PP}, \mathcal{IT}, \mathcal{TSK}, \mathcal{P}_t) \rightarrow \widetilde{\mathcal{TD}}_{\mathcal{P}_t}$. On input $\mathcal{PP}, \mathcal{IT}, \mathcal{TSK}$, a keyword policy $\mathcal{P}_t$, this algorithm produces the trapdoor $\widetilde{\mathcal{TD}}_{\mathcal{P}_t}$ for $\mathcal{P}_t$.

(5) *Data Retrieval Request Generation.* When sending the keyword policy $\mathcal{P}_t$ to TGC, the DU runs offline data retrieval request generation algorithm offDataRetReq to precompute the intermediate data retrieval request $\mathcal{IDR}$. After getting back the trapdoor $\widetilde{\mathcal{TD}}_{\mathcal{P}_t}$, the DU performs the online data retrieval request generation algorithm onDataRetReq to generate the actual data retrieval request $\mathcal{DRR}$, and sends $\mathcal{DRR}$ to the cloud.

- offDataRetReq$(\mathcal{PP}, \mathcal{DK}_{A_d}, \mathcal{P}_t) \rightarrow \mathcal{IDR}$. Taking $\mathcal{PP}, \mathcal{DK}_{A_d}, \mathcal{P}_t$ as input, this algorithm generates $\mathcal{IDR}$.
- onDataRetReq$(\mathcal{PP}, \mathcal{IDR}, \widetilde{\mathcal{TD}}_{\mathcal{P}_t}) \rightarrow [\mathcal{DRR}, \mathcal{TDK}]$. Taking $\mathcal{PP}, \mathcal{IDR}, \widetilde{\mathcal{TD}}_{\mathcal{P}_t}$, it creates the data retrieval request $\mathcal{DRR}$ and the secret transformation decryption key $\mathcal{TDK}$. Note that $\mathcal{DRR}$ contains two components $\mathcal{TD}_{\mathcal{P}_t^\circ}$ and $\mathcal{TK}_{A_d}$, where the former is transform trapdoor (which is derived from $\widetilde{\mathcal{TD}}_{\mathcal{P}_t}$) for $\mathcal{P}_t$ and the latter is transform key (which is derived from $\mathcal{DK}_{A_d}$) for the decryption attribute set A_d. That is, $\mathcal{DRR} := \langle \mathcal{TD}_{\mathcal{P}_t^\circ}, \mathcal{TK}_{A_d} \rangle$. Only $\mathcal{P}_t^\circ$ will be included in $\mathcal{TD}_{\mathcal{P}_t^\circ}$, and hence in $\mathcal{DRR}$, where $\mathcal{P}_t^\circ$ is the policy $\mathcal{P}_t$ with only generic names of the keywords.

(6) *Data Retrieval.* Upon receiving the data retrieval request $\mathcal{DRR} := \langle \mathcal{TD}_{\mathcal{P}_t^\circ}, \mathcal{TK}_{A_d} \rangle$ from DU, the cloud server responds the request as follows. The cloud server first

performs the Test algorithm on the stored ciphertexts and identifies the matching ciphertexts obeying the property that the (blinded) keyword set W° in the ciphertext satisfies the (blinded) keyword policy $\mathcal{P}_t^{\circ}$ embedded in the trapdoor (i.e., $\mathcal{P}_t^{\circ}(W^{\circ}) = true$ and hence $\mathcal{P}_t(W) = true$). After that, it sends the DU the converted ciphertexts $\mathcal{CT}_{out}$ after running the Transform algorithm upon the matching ciphertexts.

- $\mathsf{Test}(\mathcal{PP}, \mathcal{CT}, \mathcal{CSK}, \mathcal{DRR}) \rightarrow \mathcal{CT}$ or $\perp$. Taking $\mathcal{PP}, \mathcal{CT}, \mathcal{CSK}$ and $\mathcal{DRR}$ as input, the test algorithm returns the ciphertext $\mathcal{CT}$ if $\mathcal{P}_t(W) = true$; otherwise, returns $\perp$.
- $\mathsf{Transform}(\mathcal{PP}, \mathcal{CT}, \mathcal{CSK}, \mathcal{DRR}). \rightarrow \mathcal{CT}_{out}$ or $\perp$. Taking $\mathcal{PP}, \mathcal{CT}, \mathcal{CSK}$ and $\mathcal{DRR}$ as input, it returns the transformed ciphertext $\mathcal{CT}_{out}$ if $\mathcal{CT}$ is the output of Test; and returns $\perp$ if Test outputs $\perp$.

(7) *Unsigncryption and Verify.* Once the transformed ciphertext $\mathcal{CT}_{out}$ is received by a DU from the cloud, it is unsigncrypted using the user's secret transformation decryption key $\mathcal{TDK}$. Additionally, the user verifies the accuracy of the Test and Transform operations carried out by the cloud through the execution of the subsequent algorithm.

- $\mathsf{Unsigncrypt\text{-}Verify}(\mathcal{PP}, \mathcal{CT}_{out}, \mathcal{TDK}) \rightarrow msg$ or $\perp$. Taking $\mathcal{PP}$, a transformed ciphertext $\mathcal{CT}_{out}$ and the transformation decryption key $\mathcal{TDK}$ as input, it outputs the message msg if the search results are correct and the signature verification is done correctly by the cloud; otherwise, outputs $\perp$ indicating that search results are incorrect.

Data Confidentiality

Unauthorized entities should not decrypt a ciphertext stored in the cloud, which is ensured by the security property called data confidentiality. In ABDSRS, only the cloud can provide the ciphertext storage service that any authorized DU can use. So, a DU with a valid trapdoor can't perform search operation and decrypt the ciphertext if it doesn't know the cloud secret key ($\mathcal{CSK}$). Therefore, an adversary can manifest as either a Type-1 adversary, which refers to an unauthorized entity possessing $\mathcal{CSK}$, or a Type-2 adversary, which refers to an authorized entity that is ignorant of $\mathcal{CSK}$. The following defines this ABDSRS security concept using the IND-CCA2 security game, wherein the ciphertexts are indistinguishable under an adaptive chosen ciphertext attack.

IND-CCA2 Security for Type-1 Adversary

The scenario is depicted in the following security game $\mathsf{Game}^{\mathsf{IND\text{-}CCA2}}_{\mathsf{Type\text{-}1}}$, which poses a challenger $\mathscr{C}$ against a Type-1 adversary $\mathscr{A}$.

Experiment $\mathsf{Game}^{\mathsf{IND\text{-}CCA2}}_{\mathsf{Type\text{-}1}}(1^{\wp})$

1. $\mathcal{P}^{\star}_{e} \leftarrow \mathscr{A}(1^{\wp})$;
2. $[\mathcal{PP}, \mathcal{MK}] \leftarrow \mathsf{KGC\text{-}Setup}(1^{\wp}), [\mathcal{TPK}, \mathcal{TSK}] \leftarrow \mathsf{TGC\text{-}Setup}(\mathcal{PP})$, $[\mathcal{CPK}, \mathcal{CSK}] \leftarrow \mathsf{Cloud\text{-}Setup}(\mathcal{PP})$;
3. $[msg^{\star}_{0}, msg^{\star}_{1}, \mathcal{P}^{\star}_{s}, W^{\star}, \mathsf{st}] \leftarrow \mathscr{A}^{\mathcal{O}_1}(\mathcal{PP}, \mathcal{TPK}, \mathcal{CPK}, \mathcal{CSK})$, where $|msg^{\star}_{0}| = |msg^{\star}_{1}|$;
4. $\mathcal{CT}^{\star} \leftarrow \mathsf{onSigncrypt}\big(\mathcal{PP}, \mathsf{offSigncrypt}(\mathcal{PP}, \mathcal{SK}_{A_s}, \mathcal{TPK}, \mathcal{CPK}, \mathcal{P}^{\star}_{s}), msg^{\star}_{i}, \mathcal{P}^{\star}_{e}, W^{\star}\big)$, where $i \xleftarrow{\text{u}} \{0,1\}, A_s \xleftarrow{\text{u}} 2^{U_s} \ni \mathcal{P}^{\star}_{s}(A_s) = true, \mathcal{SK}_{A_s} \leftarrow \mathsf{sKeyGen}(\mathcal{PP}, \mathcal{MK}, A_s)$;
5. $i' \leftarrow \mathscr{A}^{\mathcal{O}_2}(\mathcal{P}^{\star}_{e}, \mathcal{PP}, \mathcal{TPK}, \mathcal{CPK}, \mathcal{CSK}, msg^{\star}_{0}, msg^{\star}_{1}, \mathcal{P}^{\star}_{s}, W^{\star}, \mathsf{st}, \mathcal{CT}^{\star})$.

where $\mathcal{O}_1 := \{\mathcal{O}_{\mathcal{SKG}}, \mathcal{O}_{\mathcal{DRR}}, \mathcal{O}_{\mathcal{SC}}, \mathcal{O}_{\mathcal{DR\text{-}UV}}\}$ and $\mathcal{O}_2 := \{\mathcal{O}_{\mathcal{SKG}}, \mathcal{O}_{\mathcal{DRR}}, \mathcal{O}_{\mathcal{SC}}, \mathcal{O}'_{\mathcal{DR\text{-}UV}}\}$ are two sets of oracles (defined below), 2^{U_s} is the set of all non-empty subsets of the signing attribute universe U_s, and st is state information maintained by $\mathscr{A}$.

- *Signing key generation oracle* $\mathcal{O}_{\mathcal{SKG}}(A_s)$: on input a signing attribute set A_s, it returns the signing key $\mathcal{SK}_{A_s}$ to $\mathscr{A}$.

- *Data retrieval request generation oracle* $\mathcal{O}_{\mathcal{DRR}}(A_d, \mathcal{P}_t)$: on input a decryption attribute set A_d and a keyword policy $\mathcal{P}_t$, it performs as follows.
  (i) In case $\mathcal{P}^{\star}_{e}(A_d) = true$, it computes $[\mathcal{DK}_{A_d}, \widetilde{\mathcal{TD}}_{\mathcal{P}_t}, \mathcal{TK}_{A_d}, \mathcal{TD}_{\mathcal{P}^{\circ}_{t}}, \mathcal{TDK}]$ and returns the data retrieval request $\mathcal{DRR} := \langle \mathcal{TD}_{\mathcal{P}^{\circ}_{t}}, \mathcal{TK}_{A_d} \rangle$ to $\mathscr{A}$.
  (ii) In case $\mathcal{P}^{\star}_{e}(A_d) = false$, it computes $[\mathcal{DK}_{A_d}, \widetilde{\mathcal{TD}}_{\mathcal{P}_t}]$ and returns the same to $\mathscr{A}$. Note that, in this case, $\mathscr{A}$ can generate $\mathcal{DRR} := \langle \mathcal{TD}_{\mathcal{P}^{\circ}_{t}}, \mathcal{TK}_{A_d} \rangle$ and $\mathcal{TDK}$.

- *Signcryption oracle* $\mathcal{O}_{\mathcal{SC}}(msg, \mathcal{P}_s, \mathcal{P}_e, W)$: on input a message msg, a signing policy $\mathcal{P}_s$, an encryption policy $\mathcal{P}_e$ and a keyword set W, it returns to $\mathscr{A}$ the corresponding ciphertext $\mathcal{CT}$.

- *Data retrieval cum unsigncrypt-verify oracle* $\mathcal{O}_{\mathcal{DR\text{-}UV}}(\mathcal{CT}, A_d, \mathcal{P}_t)$: on input a ciphertext $\mathcal{CT}$, a decryption attribute set A_d and a keyword policy $\mathcal{P}_t$, it returns either msg or $\perp$ to $\mathscr{A}$.

- $\mathcal{O}'_{\mathcal{DR\text{-}UV}}$ is same as $\mathcal{O}_{\mathcal{DR\text{-}UV}}$, except that $\mathscr{A}$ is not permitted to query $\mathcal{O}'_{\mathcal{DR\text{-}UV}}$ using the input $[\mathcal{CT}^{\star}, A_d, \mathcal{P}_t]$ satisfying $\mathcal{P}^{\star}_{e}(A_d) = true \wedge \mathcal{P}_t(W^{\star}) = true$, here $W^{\star}$ is the keyword set of $\mathcal{CT}^{\star}$.

If $i' = i$, $\mathscr{A}$ wins the game. The advantage of $\mathscr{A}$ in winning the above game is described as $\mathsf{Adv}^{\mathsf{IND\text{-}CCA2}}_{\mathsf{Type\text{-}1}}(1^{\wp}) \stackrel{\text{def}}{=} \big|\Pr[i' = i] - 1/2\big|$.

IND-CCA2 Security for Type-2 Adversary

The scenario is depicted in the following security game $\mathsf{Game}^{\mathsf{IND\text{-}CCA2}}_{\mathsf{Type\text{-}2}}$, which poses a challenger $\mathscr{C}$ against a Type-2 adversary $\mathscr{A}$.

Experiment $\mathsf{Game}^{\mathsf{IND\text{-}CCA2}}_{\mathsf{Type\text{-}2}}(1^{\wp})$

1. $[\mathcal{PP}, \mathcal{MK}] \leftarrow \mathsf{KGC\text{-}Setup}(1^{\wp}), [\mathcal{TPK}, \mathcal{TSK}] \leftarrow \mathsf{TGC\text{-}Setup}(\mathcal{PP})$, $[\mathcal{CPK}, \mathcal{CSK}] \leftarrow \mathsf{Cloud\text{-}Setup}(\mathcal{PP})$;
2. $[msg_0^{\star}, msg_1^{\star}, \mathcal{P}_e^{\star}, \mathcal{P}_s^{\star}, W^{\star}, \mathsf{st}] \leftarrow \mathscr{A}^{\mathcal{O}_3}(\mathcal{PP}, \mathcal{TPK}, \mathcal{CPK})$, where $|msg_0^{\star}| = |msg_1^{\star}|$;
3. $\mathcal{CT}^{\star} \leftarrow \mathsf{onSigncrypt}\big(\mathcal{PP}, \mathsf{offSigncrypt}(\mathcal{PP}, \mathcal{SK}_{A_s}, \mathcal{TPK}, \mathcal{CPK}, \mathcal{P}_s^{\star}), msg_i^{\star}, \mathcal{P}_e^{\star}, W^{\star}\big)$; where $i \xleftarrow{\text{u}} \{0,1\}, A_s \xleftarrow{\text{u}} 2^{U_s} \ni \mathcal{P}_s^{\star}(A_s) = true, \mathcal{SK}_{A_s} \leftarrow \mathsf{sKeyGen}(\mathcal{PP}, \mathcal{MK}, A_s)$;
4. $i' \leftarrow \mathscr{A}^{\mathcal{O}_4}(\mathcal{PP}, \mathcal{TPK}, \mathcal{CPK}, msg_0^{\star}, msg_1^{\star}, \mathcal{P}_e^{\star}, \mathcal{P}_s^{\star}, W^{\star}, \mathsf{st}, \mathcal{CT}^{\star})$.

where $\mathcal{O}_3 := \{\mathcal{O}_{\mathcal{SKG}}, \mathcal{O}'_{\mathcal{DRR}}, \mathcal{O}_{\mathcal{SC}}, \mathcal{O}_{\mathcal{DR}\text{-}\mathcal{UV}}\}$ and $\mathcal{O}_4 := \{\mathcal{O}_{\mathcal{SKG}}, \mathcal{O}'_{\mathcal{DRR}}, \mathcal{O}_{\mathcal{SC}}, \mathcal{O}'_{\mathcal{DR}\text{-}\mathcal{UV}}\}$ are two sets of oracles which are defined below.

- *Data retrieval request generation oracle* $\mathcal{O}'_{\mathcal{DRR}}(A_d, \mathcal{P}_t)$: on input a decryption attribute set A_d and a keyword policy $\mathcal{P}_t$, it returns $[\mathcal{DK}_{A_d}, \widetilde{\mathcal{TD}}_{\mathcal{P}_t}]$ to $\mathscr{A}$. Note that $\mathscr{A}$ can generate $\mathcal{DRR} := \langle \mathcal{TD}_{\mathcal{P}_t^{\circ}}, \mathcal{TK}_{A_d} \rangle$ and $\mathcal{TDK}$ by using the received $\mathcal{DK}_{A_d}$ and $\widetilde{\mathcal{TD}}_{\mathcal{P}_t}$.

The other oracles are essentially the same as that of $\mathsf{Game}^{\mathsf{IND\text{-}CCA2}}_{\mathsf{Type\text{-}1}}(1^{\wp})$.

If $i' = i$, $\mathscr{A}$ wins the game. The advantage of $\mathscr{A}$ in winning the above game is described as $\mathsf{Adv}^{\mathsf{IND\text{-}CCA2}}_{\mathsf{Type\text{-}2}}(1^{\wp}) \stackrel{\text{def}}{=} \big|\Pr[i' = i] - 1/2\big|$.

Definition 3. *The ABDSRS is said to be IND-CCA2 secure if* $\mathsf{Adv}^{\mathsf{IND\text{-}CCA2}}_{\mathsf{Type\text{-}1}}(1^{\wp})$ *and* $\mathsf{Adv}^{\mathsf{IND\text{-}CCA2}}_{\mathsf{Type\text{-}2}}(1^{\wp})$ *are negligible, for all PPT Type-1 and Type-2 adversaries, respectively.*

Unforgeability of the Data

It detects the inability of an external malicious entity or an unauthorized DO to generate a valid signature, thereby ensuring the signature verification mechanism is successful. And, even if unauthorized DOs collude and pool their signing attributes such that the collection of attributes satisfies a signing policy whereas none of the single DOs would satisfy the policy on its own, they cannot create a ciphertext with valid signature for that signing policy. Existential unforgeability against Chosen Message Attack (EUF-CMA) model defines this ABDSRS security concept.

EUF-CMA Security

The security game $\mathsf{Game}_{\mathscr{A}}^{\mathsf{EUF\text{-}CMA}}$ involves a model that consists of a challenger $\mathscr{C}$ and an adversary $\mathscr{A}$. The game works as follows.

Experiment $\mathsf{Game}_{\mathscr{A}}^{\mathsf{EUF\text{-}CMA}}(1^{\wp})$

1. $\mathcal{P}_s^{\star} \leftarrow \mathscr{A}(1^{\wp})$;
2. $[\mathcal{PP}, \mathcal{MK}] \leftarrow \mathsf{KGC\text{-}Setup}(1^{\wp}), [\mathcal{TPK}, \mathcal{TSK}] \leftarrow \mathsf{TGC\text{-}Setup}(\mathcal{PP})$, $[\mathcal{CPK}, \mathcal{CSK}] \leftarrow \mathsf{Cloud\text{-}Setup}(\mathcal{PP})$;
3. $\mathcal{CT}^{\star} \leftarrow \mathscr{A}^{\mathcal{O}}(\mathcal{PP}, \mathcal{TPK}, \mathcal{CPK}, \mathcal{CSK})$.

where $\mathcal{CT}^{\star} = \mathcal{CT}^{\star}_{(\mathcal{P}_s^{\star}, \mathcal{P}_e^{\star}, W^{\star})}$, $\mathcal{O} := \{\mathcal{O}'_{\mathcal{SKG}}, \mathcal{O}'_{\mathcal{DRR}}, \mathcal{O}_{\mathcal{SC}}, \mathcal{O}_{\mathcal{DR}\text{-}\mathcal{UV}}\}$ is a set of oracles and

- *Signing key generation oracle* $\mathcal{O}'_{\mathcal{SKG}}(A_s)$: on input a signing attribute set A_s with the restriction that $\mathcal{P}_s^{\star}(A_s) = false$, it outputs and transmits the signing key $\mathcal{SK}_{A_s}$ to $\mathscr{A}$.

Note that the other oracles are similar to that of $\mathsf{Game}_{\mathsf{Type\text{-}2}}^{\mathsf{IND\text{-}CCA2}}(1^{\wp})$.

The adversary $\mathscr{A}$ wins this game if there exist $A_d, \mathcal{P}_t$ satisfying the following simultaneously: $\mathcal{P}_e^{\star}(A_d) = true, \mathcal{P}_t(W^{\star}) = true, \mathcal{O}_{\mathcal{DR}\text{-}\mathcal{UV}}(\mathcal{CT}^{\star}, A_d, \mathcal{P}_t) = msg^{\star} \neq \perp$, and $\mathscr{A}$ did not query for $\mathcal{O}_{\mathcal{SC}}$ using the input $(msg^{\star}, \mathcal{P}_s^{\star}, \mathcal{P}_e^{\star}, W^{\star})$. The advantage of $\mathscr{A}$ in this game is defined as $\mathsf{Adv}_{\mathscr{A}}^{\mathsf{EUF\text{-}CMA}}(1^{\wp}) \stackrel{\text{def}}{=} \text{Prob}[\mathscr{A} \text{ wins the game}]$.

Definition 4. *The ABDSRS is said to be EUF-CMA secure if* $\mathsf{Adv}_{\mathscr{A}}^{\mathsf{EUF\text{-}CMA}}(1^{\wp})$ *is negligible, for any* $\mathscr{A}$.

DO Privacy

This guarantees that the ciphertext does not reveal the set of attributes used in signing process. No one, including the cloud server, a DU, or any other adversary, can deduce the set of DO's attributes used to generate a signature from a certain ciphertext. The scenario is depicted in the following security game $\mathsf{Game}_{\mathscr{A}}^{\mathsf{DO\text{-}Privacy}}$, which poses a challenger $\mathscr{C}$ against an adversary $\mathscr{A}$. The game works as descried below.

Experiment $\mathsf{Game}_{\mathscr{A}}^{\mathsf{DO\text{-}Privacy}}(1^{\wp})$

1. $[\mathcal{PP}, \mathcal{MK}] \leftarrow \mathsf{KGC\text{-}Setup}(1^{\wp}), [\mathcal{TPK}, \mathcal{TSK}] \leftarrow \mathsf{TGC\text{-}Setup}(\mathcal{PP})$, $[\mathcal{CPK}, \mathcal{CSK}] \leftarrow \mathsf{Cloud\text{-}Setup}(\mathcal{PP})$;
2. $[A_s^{(0)}, A_s^{(1)}, msg, \mathcal{P}_s, \mathcal{P}_e, W, \mathsf{st}] \leftarrow \mathscr{A}(\mathcal{PP}, \mathcal{MK}, \mathcal{TPK}, \mathcal{TSK}, \mathcal{CPK}, \mathcal{CSK})$, where $\mathcal{P}_s(A_s^{(0)}) = true = \mathcal{P}_s(A_s^{(1)})$;
3. $\mathcal{CT}^{\star} \leftarrow \mathsf{onSigncrypt}\big(\mathcal{PP}, \mathsf{offSigncrypt}(\mathcal{PP}, \mathcal{SK}_{A_s^{(i)}}, \mathcal{TPK}, \mathcal{CPK}, \mathcal{P}_s), msg, \mathcal{P}_e, W\big)$, where $i \xleftarrow{\mathrm{u}} \{0,1\}, \mathcal{SK}_{A_s^{(i)}} \leftarrow \mathsf{sKeyGen}(\mathcal{PP}, \mathcal{MK}, A_s^{(i)})$;
4. $i' \leftarrow \mathscr{A}(\mathcal{PP}, \mathcal{MK}, \mathcal{TPK}, \mathcal{TSK}, \mathcal{CPK}, \mathcal{CSK}, A_s^{(0)}, A_s^{(1)}, msg, \mathcal{P}_s, \mathcal{P}_e, W, \mathsf{st}, \mathcal{CT}^{\star})$.

In this game, $\mathscr{A}$ need not to query any oracle and it can compute required components by itself because $\mathscr{A}$ is given access to system master secret key, cloud secret key and trapdoor secret key.

If $i' = i$, $\mathscr{A}$ wins the game. The advantage of $\mathscr{A}$ in winning the above game is $\mathsf{Adv}_{\mathscr{A}}^{\mathsf{DO\text{-}Privacy}}(1^{\wp}) \stackrel{\text{def}}{=} \text{Prob}[i' = i]$.

Definition 5. *The ABDSRS is said to provide DO privacy if* $\mathsf{Adv}_{\mathscr{A}}^{\mathsf{DO\text{-}Privacy}}(1^{\wp}) = \frac{1}{2}$, *for any* $\mathscr{A}$.

Verifiability

The verifiability ensures that DU can check whether the transformed ciphertext made by the cloud is correct. That is, an authorized DU can verify the correctness of the test, transform and signature verification operations done by the cloud. Specifically, given a challenge ciphertext for the message $msg^{\star}$, the malicious cloud cannot create a transformed ciphertext that gives a message not in the set $\{msg^{\star}, \perp\}$ and passes the verification mechanism. The model is formulated by a security game $\mathsf{Game}_{\mathscr{A}}^{\mathsf{verifiability}}$, presented below, between an authorized DU $\mathscr{C}$ and the malicious cloud $\mathscr{A}$. where $\widetilde{\mathcal{O}} := \{\mathcal{O}_{\mathcal{SKG}}, \mathcal{O}''_{\mathcal{DRR}}, \mathcal{O}_{\mathcal{SC}}, \mathcal{O}_{\mathcal{DR}\text{-}\mathcal{UV}}\}$ and

- *Data retrieval request generation oracle* $\mathcal{O}''_{\mathcal{DRR}}(A_d, \mathcal{P}_t)$: on input a keyword policy $\mathcal{P}_t$ and a decryption attribute set A_d, it outputs $[\mathcal{DK}_{A_d}, \widetilde{\mathcal{TD}}_{\mathcal{P}_t}, \mathcal{TK}_{A_d}, \mathcal{TD}_{\mathcal{P}_t^{\circ}}, \mathcal{TDK}]$ and sends the data retrieval request $\mathcal{DRR} := \langle \mathcal{TD}_{\mathcal{P}_t^{\circ}}, \mathcal{TK}_{A_d} \rangle$ to the adversary $\mathscr{A}$. Next, it stores the tuple $[\![A_d, \mathcal{P}_t, \mathcal{TK}_{A_d}, \mathcal{TD}_{\mathcal{P}_t^{\circ}}, \mathcal{TDK}]\!]$ in table $\mathsf{Tab}''_{\mathcal{DRR}}$.

Experiment $\mathsf{Game}_{\mathscr{A}}^{\mathsf{verifiability}}(1^{\wp})$

1. $[\mathcal{PP}, \mathcal{MK}] \leftarrow \mathsf{KGC\text{-}Setup}(1^{\wp}), [\mathcal{TPK}, \mathcal{TSK}] \leftarrow \mathsf{TGC\text{-}Setup}(\mathcal{PP})$, $[\mathcal{CPK}, \mathcal{CSK}] \leftarrow \mathsf{Cloud\text{-}Setup}(\mathcal{PP})$;
2. $[msg^{\star}, \mathcal{P}_e^{\star}, \mathcal{P}_s^{\star}, W^{\star}, \mathsf{st}] \leftarrow \mathscr{A}^{\tilde{\mathcal{O}}}(\mathcal{PP}, \mathcal{TPK}, \mathcal{CPK}, \mathcal{CSK})$;
3. $\mathcal{CT}^{\star} \leftarrow \mathsf{onSigncrypt}\big(\mathcal{PP}, \mathsf{offSigncrypt}(\mathcal{PP}, \mathcal{SK}_{A_s}, \mathcal{TPK}, \mathcal{CPK}, \mathcal{P}_s^{\star}), msg^{\star}, \mathcal{P}_e^{\star}, W^{\star}\big)$; where $A_s \xleftarrow{\mathrm{u}} 2^{U_s} \ni \mathcal{P}_s^{\star}(A_s) = true, \mathcal{SK}_{A_s} \leftarrow \mathsf{sKeyGen}(\mathcal{PP}, \mathcal{MK}, A_s)$;
4. $[A_d, \mathcal{P}_t, \mathcal{CT}_{out}] \leftarrow \mathscr{A}^{\tilde{\mathcal{O}}}(\mathcal{PP}, \mathcal{TPK}, \mathcal{CPK}, \mathcal{CSK}, msg^{\star}, \mathcal{P}_e^{\star}, \mathcal{P}_s^{\star}, W^{\star}, \mathsf{st}, \mathcal{CT}^{\star})$, where $\mathcal{P}_e^{\star}(A_d) = true \wedge \mathcal{P}_t(W^{\star}) = true$.

Suppose that the tuple $[\![A_d, \mathcal{P}_t, \mathcal{TK}_{A_d}, \mathcal{TD}_{\mathcal{P}_t^{\circ}}, \mathcal{TDK}]\!]$ is in table $\mathsf{Tab}''_{\mathcal{DRR}}$. If not, it can be generated by querying the oracle $\mathcal{O}''_{\mathcal{DRR}}$ with the input $(A_d, \mathcal{P}_t)$.
The adversary $\mathscr{A}$ can win the game
if $\mathsf{Unsigncrypt\text{-}Verify}(\mathcal{PP}, \mathcal{CT}_{out}, \mathcal{TDK}) \notin \{msg^{\star}, \perp\}$.

Definition 6. *The ABDSRS is verifiable if the advantage of $\mathscr{A}$ in the game $\mathsf{Game}_{\mathscr{A}}^{\mathsf{verifiability}}$, defined as $\mathsf{Adv}_{\mathscr{A}}^{\mathsf{verifiability}}(1^{\wp}) \stackrel{\mathrm{def}}{=} Prob[\mathscr{A}\ wins]$, is negligible, for all PPT adversaries $\mathscr{A}$.*

Keyword Privacy

This guarantees that the ciphertext reveals nothing about the keyword values it contains. The cloud server (referred to as Type-1 adversary) cannot determine which ciphertext uses which set of keyword values without knowledge of the corresponding "valid" trapdoor. A trapdoor is considered valid if it contains a keyword policy that accepts the set of keywords linked to the ciphertext. The Type-2 adversary, which refers to an authorized entity that is ignorant of $\mathcal{CSK}$, is unable to identify which ciphertext utilizes which set of keyword values, even though the adversary knows the corresponding valid trapdoors. Security in terms of keyword privacy is defined subsequently as indistinguishability against chosen keyword set attack (in short, IND-CKA).

IND-CKA Security for Type-1 Adversary

The scenario is depicted in the following security game $\mathsf{Game}_{\mathsf{Type\text{-}1}}^{\mathsf{IND\text{-}CKA}}$, which poses a challenger $\mathscr{C}$ against a Type-1 adversary $\mathscr{A}$.

Experiment $\mathsf{Game}^{\mathsf{IND\text{-}CKA}}_{\mathsf{Type\text{-}1}}(1^{\wp})$

1. $W_0^\star, W_1^\star \leftarrow \mathscr{A}(1^{\wp})$, where $|W_0^\star| = |W_1^\star|$ and $W_0^{\star\circ} = W_1^{\star\circ}$;
2. $[\mathcal{PP}, \mathcal{MK}] \leftarrow \mathsf{KGC\text{-}Setup}(1^{\wp}), [\mathcal{TPK}, \mathcal{TSK}] \leftarrow \mathsf{TGC\text{-}Setup}(\mathcal{PP})$, $[\mathcal{CPK}, \mathcal{CSK}] \leftarrow \mathsf{Cloud\text{-}Setup}(\mathcal{PP})$;
3. $[msg^\star, \mathcal{P}_e^\star, \mathcal{P}_s^\star, \mathsf{st}] \leftarrow \mathscr{A}^{\widehat{\mathcal{O}}}(\mathcal{PP}, \mathcal{TPK}, \mathcal{CPK}, \mathcal{CSK})$;
4. $\mathcal{CT}^\star \leftarrow \mathsf{onSigncrypt}\big(\mathcal{PP}, \mathsf{offSigncrypt}(\mathcal{PP}, \mathcal{SK}_{A_s}, \mathcal{TPK}, \mathcal{CPK}, \mathcal{P}_s^\star), msg^\star, \mathcal{P}_e^\star, W_i^\star\big)$, where $i \xleftarrow{\text{u}} \{0,1\}, A_s \xleftarrow{\text{u}} 2^{U_s} \ni \mathcal{P}_s^\star(A_s) = true, \mathcal{SK}_{A_s} \leftarrow \mathsf{sKeyGen}(\mathcal{PP}, \mathcal{MK}, A_s)$;
5. $i' \leftarrow \mathscr{A}^{\widehat{\mathcal{O}}}(\mathcal{PP}, \mathcal{TPK}, \mathcal{CPK}, \mathcal{CSK}, msg^\star, \mathcal{P}_e^\star, \mathcal{P}_s^\star, W_0^\star, W_1^\star, \mathsf{st}, \mathcal{CT}^\star)$.

where $\widehat{\mathcal{O}} := \{\mathcal{O}_{\mathcal{SKG}}, \mathcal{O}'''_{\mathcal{DRR}}, \mathcal{O}_{test}\}$ and

- *Data retrieval request generation oracle* $\mathcal{O}'''_{\mathcal{DRR}}(A_d, \mathcal{P}_t)$: on input a decryption attribute set A_d and a keyword policy $\mathcal{P}_t$ with the condition that $\mathcal{P}_t(W_0^\star) = false \wedge \mathcal{P}_t(W_1^\star) = false$, it computes the data retrieval request $\mathcal{DRR} := \langle \mathcal{TD}_{\mathcal{P}_t^\circ}, \mathcal{TK}_{A_d} \rangle$ and sends the same to $\mathscr{A}$.

- *Test oracle* $\mathcal{O}_{test}(\mathcal{CT}, \mathcal{P}_t)$: Taking a ciphertext $\mathcal{CT}$ and a keyword policy $\mathcal{P}_t$ obeying the condition $\mathcal{P}_t(W_0^\star) = false \wedge \mathcal{P}_t(W_1^\star) = false$, it returns the output of Test algorithm.

IND-CKA Security for Type-2 Adversary

The security game $\mathsf{Game}^{\mathsf{IND\text{-}CKA}}_{\mathsf{Type\text{-}2}}$ involves a model that consists of a challenger $\mathscr{C}$ and an adversary $\mathscr{A}$. The game works as follows.

Experiment $\mathsf{Game}^{\mathsf{IND\text{-}CKA}}_{\mathsf{Type\text{-}2}}(1^{\wp})$

1. $[\mathcal{PP}, \mathcal{MK}] \leftarrow \mathsf{KGC\text{-}Setup}(1^{\wp}), [\mathcal{TPK}, \mathcal{TSK}] \leftarrow \mathsf{TGC\text{-}Setup}(\mathcal{PP})$, $[\mathcal{CPK}, \mathcal{CSK}] \leftarrow \mathsf{Cloud\text{-}Setup}(\mathcal{PP})$;
2. $[msg^\star, \mathcal{P}_e^\star, \mathcal{P}_s^\star, W_0^\star, W_1^\star, \mathsf{st}] \leftarrow \mathscr{A}^{\mathcal{O}_5}(\mathcal{PP}, \mathcal{TPK}, \mathcal{CPK})$, where $|W_0^\star| = |W_1^\star|$ and $W_0^{\star\circ} = W_1^{\star\circ}$;
3. $\mathcal{CT}^\star \leftarrow \mathsf{onSigncrypt}\big(\mathcal{PP}, \mathsf{offSigncrypt}(\mathcal{PP}, \mathcal{SK}_{A_s}, \mathcal{TPK}, \mathcal{CPK}, \mathcal{P}_s^\star), msg^\star, \mathcal{P}_e^\star, W_i^\star\big)$, where $i \xleftarrow{\text{u}} \{0,1\}, A_s \xleftarrow{\text{u}} 2^{U_s} \ni \mathcal{P}_s^\star(A_s) = true, \mathcal{SK}_{A_s} \leftarrow \mathsf{sKeyGen}(\mathcal{PP}, \mathcal{MK}, A_s)$;
4. $i' \leftarrow \mathscr{A}^{\mathcal{O}_6}(\mathcal{PP}, \mathcal{TPK}, \mathcal{CPK}, msg^\star, \mathcal{P}_e^\star, \mathcal{P}_s^\star, W_0^\star, W_1^\star, \mathsf{st}, \mathcal{CT}^\star)$.

where $\mathcal{O}_5 := \{\mathcal{O}_{\mathcal{SKG}}, \mathcal{O}'_{\mathcal{DRR}}, \mathcal{O}'_{test}\}$ and $\mathcal{O}_6 := \{\mathcal{O}_{\mathcal{SKG}}, \mathcal{O}'_{\mathcal{DRR}}, \mathcal{O}''_{test}\}$,

- *Test oracle* $\mathcal{O}'_{test}(\mathcal{CT}, \mathcal{P}_t)$: Given a ciphertext $\mathcal{CT}$ and a keyword policy $\mathcal{P}_t$, it returns the output of Test algorithm.

- $\mathcal{O}''_{test}(\mathcal{CT}, \mathcal{P}_t)$: This is same as $\mathcal{O}'_{test}(\mathcal{CT}, \mathcal{P}_t)$, except that $\mathscr{A}$ is not permitted to query $\mathcal{O}''_{test}$ using the input $(\mathcal{CT}^\star, \mathcal{P}_t)$ such that $\mathcal{P}_t(W_i^\star) = true$.

For $\mathsf{k} \in \{1, 2\}$, the adversary wins the game $\mathsf{Game}^{\mathsf{IND\text{-}CKA}}_{\mathsf{Type\text{-}k}}$ if $i' = i$. The adversary's advantage in $\mathsf{Game}^{\mathsf{IND\text{-}CKA}}_{\mathsf{Type\text{-}k}}$ is defined as $\mathsf{Adv}^{\mathsf{IND\text{-}CKA}}_{\mathsf{Type\text{-}k}}(1^\wp) \stackrel{\text{def}}{=} \left|\text{Prob}[i' = i] - 1/2\right|$.

Definition 7. *The ABDSRS is said to be IND-CKA secure if* $\mathsf{Adv}^{\mathsf{IND\text{-}CKA}}_{\mathsf{Type\text{-}1}}(1^\wp)$ *and* $\mathsf{Adv}^{\mathsf{IND\text{-}CKA}}_{\mathsf{Type\text{-}2}}(1^\wp)$ *are negligible, for all PPT Type-1 and Type-2 adversaries, respectively.*

3.3 ABDSRS Construction

Prior to presenting our ABDSRS, we first describe our techniques that are used to achieve the key characteristics of ABDSRS.

Let p be a prime, and $\mathbb{G}$ and $\mathbb{G}_T$ be cyclic groups of order p. We employ a bilinear pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ and eight collision-resistant hash functions $\{H_i\}_{i=1}^{8}$, the description of these functions is given in KGC-Setup algorithm presented below. To reduce the number of hash functions, we use the hash function $H_5 : \{0,1\}^* \to \mathbb{Z}_p^*$ to map either a binary string or a $\mathbb{G}_T$ element to an element of $\mathbb{Z}_p^*$. In the latter case, we first convert the $\mathbb{G}_T$ element into a binary string, then we use H_5. The attribute universe $U = \{0,1\}^*$. Using a hash function $H_1 : \{0,1\}^* \to \mathbb{G}$, the signing attributes are converted to random elements of $\mathbb{G}$. Using another hash function $H_t : \{0,1\}^* \to \mathbb{Z}_p^*$, one can map each encryption attribute/keyword string to an element of $\mathbb{Z}_p^*$. Hence, for brevity, we treat both encryption attributes as well as keywords are elements of $\mathbb{Z}_p^*$. All the access policies we consider in this system use MSP representation (described in Section 2.4.1). We assume that the signing attributes involved in signing policy are all distinct. However, this is acceptable from a practical view point due to the following fact. To sign a data file, the DO formulates a signing policy that accepts its signing attribute set. This can be done in many ways. For instance, if A_s is a signing attribute set, then DO can always create a policy of the form $\mathcal{P}_s := \breve{\mathcal{P}}_s \vee \breve{A}_s$ satisfying $\mathcal{P}_s(A_s) = true$, where $\breve{\mathcal{P}}_s$ is any signing policy with distinct attributes which are also different from those in A_s and $\breve{A}_s \subset A_s$.

We employ suitably and significantly modified version of the attribute-based online-offline signcryption scheme (ABOOSC) proposed in [69] as the building block due to the following fact. ABOOSC supports a large attribute universe and expressive monotone boolean function access policies, and realizes constant size public

parameters, DO authentication, lightweight signcryption and DO anonymity, simultaneously.

- *Lightweight Design.* We implement an online-offline framework in signcryption, trapdoor generation and data retrieval request generation algorithms. The offline phase carries out majority of the required computations including all the heavy computations such as pairing, exponentiation etc. whereas during online phase comparatively quite a few light computations (e.g., hashing, modular addition and modular multiplication etc.) are performed. Using the idea suggested in [90], we make the number of required pairing computations constant (precisely, 14) during the process of identifying matching ciphertexts. The number of pairings used in transform operation is independent of the number of keywords and signing attributes, and depends only on the number of encryption attributes associated with the ciphertext. Precisely, this number is $2\ell_e+7$, where ℓ_e is the number of encryption attributes. To make unsigncryption lightweight, we employ *outsourcing mechanism*. The cloud partially unsigncrypts the ciphertext using the trapdoor received from a DU and sends the transformed ciphertext to the DU. Then, in order to retrieve the plaintext and ensure that the cloud operations were valid, the DU just has to run two $\mathbb{G}_T$ multiplications, two $\mathbb{G}_T$ exponentiations, two hash function calculations, and one key derivation function KDF computation.

- *Keyword Privacy.* In order to ensure the keyword privacy, the technique used in [42, 13] is adopted. Namely, each keyword is split into generic keyword name and a keyword value. Precisely, the structure of each keyword is [generic keyword name : keyword value]. In the ciphertext, we include only the set W° of keywords that includes only generic names. And, a trapdoor is attached with the keyword policy with generic names of the keywords only. For instance, if the keyword policy is $\mathcal{P}_t := [\mathsf{university} : \mathsf{ABC}] \wedge ([\mathsf{faculty} : \mathsf{asst\text{-}prof}] \vee [\mathsf{student} : \mathsf{research\text{-}scholar}])$, then the policy $\mathcal{P}_t^{\circ} := \mathsf{university} \wedge (\mathsf{faculty} \vee \mathsf{student})$ will be attached to a trapdoor. Therefore, both the cloud and users are not familiar to the precise keyword values $\mathsf{ABC}, \mathsf{asst\text{-}prof}, \mathsf{research\text{-}scholar}$. Many pairing-based searchable encryption schemes are not secure against KGAs [9]. More specifically, the actual value of the keyword concealed within a ciphertext (resp. a keyword trapdoor) can be deduced by an adversary through the appropriate pairing of the ciphertext's components with the public parameters.

  (i) To prevent KGAs on ciphertexts, we adopt the *linear-splitting* technique suggested in [8, 13] in combination with the technique proposed in [90]. The

former splits each keyword ciphertext component into two complementary randomized components and re-randomizes every keyword trapdoor component to match the splitted components in the ciphertext. The latter makes the number of required pairing computations constant.

(ii) To prevent KGAs on trapdoors, a pair of public and secret keys are associated with the cloud. Additionally, the generation of trapdoors utilises the public key of the cloud in a manner that mandates knowledge of the cloud secret key for the search operation. Therefore, information regarding the keyword values will not be disclosed from a trapdoor to an adversary other than the cloud. Only the cloud server with its secret key can perform search operations and can identify the matching ciphertexts, the cloud can acquire knowledge of the keyword values encoded in the trapdoor by performing offline KGAs.

- *Verifiability.* Verifying the accuracy of the cloud's test, transform, and signature verification procedures independently is the most challenging task. The following novel technical concepts are used to accomplish this. The ciphertext $\mathcal{CT}$ consists of $\langle \Omega_s, \Omega_e, \Omega_k, \mathsf{tag2}, E_2, \varepsilon_2, \eta \rangle$, where Ω_s (resp. Ω_e, Ω_k) consists of signature (resp. encryption, keyword) components. The message msg is encrypted as $ct := msg \oplus \mathsf{KDF}\big(g_T^{\beta+\frac{1}{\theta}}\big)$, where g_T is a public parameter and β, θ are random exponents. Note that $\mathsf{tag2} := H_3\big(H_2(g_T^{\beta+\frac{1}{\theta}})||ct\big)$. To recover the message, one needs compute $g_T^{\beta+\frac{1}{\theta}} = g_T^{\beta} \cdot g_T^{\frac{1}{\theta}}$. The DU re-randomizes the trapdoor $\widetilde{\mathcal{TD}_{\mathcal{P}_t}}$, received from TGC, with the random number $\breve{\tau}/t_r$ and produces the actual trapdoor $\mathcal{TD}_{\mathcal{P}_t}$. Also, the DU suitably re-randomizes the decryption key with two random numbers $\breve{\tau}, \gamma$. Hence, in the generation of transformed ciphertext $\mathcal{CT}_{out}$, the term $g_T^{\beta} = e(g,g)^{\alpha\beta}$ is masked as $e(g,g)^{(\alpha\beta)/\gamma} \cdot e(g_{14},g)^{(\beta\breve{\tau})/\gamma}$ which is denoted by Δ_2. To cancel the term $e(g_{14},g)^{(\beta\breve{\tau})/\gamma}$, cloud computes $\breve{X}_1$ (given in Equation (4.4)), X_2 (given in Equation (4.3) and the product $\breve{X}_1 X_2$ correctly yields $\Delta_1 := e(g_{14},g)^{(\beta\breve{\tau})/t_r}$ if $\mathcal{CT}$ is a matching ciphertext. This verifies the correctness of test operation done by the cloud. Next, the cloud runs the signature verification algorithm of [69] with random exponent φ, and recovers the term $g_T^{\varphi/\theta}$ if the ciphertext contains a legitimate signature. Next, the cloud computes $\Delta_3 := \big(g_T^{\varphi/\theta}\big)^{1/\varphi} = g_T^{1/\theta}$. Finally, the DU receives the transformed ciphertext $\mathcal{CT}_{out} := \langle \Delta_1, \Delta_2, \Delta_3, ct, \mathsf{tag2} \rangle$ from the cloud. Since the cloud does not know $\langle tdk_1 := t_r, tdk_2 := \gamma \rangle$ set by DU, it cannot decrypt any original ciphertext. Upon receiving $\mathcal{CT}_{out}$, DU computes $\Lambda := (\Delta_1)^{-tdk_1} \cdot (\Delta_2)^{tdk_2} \cdot \Delta_3$ and checks whether

$H_3\big(H_2(\Lambda)||ct\big) \stackrel{?}{=} \mathsf{tag2}$. If all the operations done by the cloud are correct, Λ correctly gives $g_T^{\beta+\frac{1}{\theta}}$ and hence $H_3\big(H_2(\Lambda)||ct\big) = \mathsf{tag2}$. In this case, the DU recovers the message msg as $msg = ct \oplus \mathsf{KDF}(\Lambda)$.

- *DO authentication.* An authorized DO who has obtained a valid signing key from KGC can create a verifiable correct signature. The DO includes the components $\sigma' := g^{\delta'}, \sigma'' := g_4^{\delta''}, \Gamma := H_7(g_T^{1/\theta}) \oplus H_8\big(e(\sigma', Y_c)^{\delta''}\big)$ in Ω_s, where $Y_c := g_4^{\mathcal{CSK}}$ is the cloud public key. Upon receiving the data storage request from DO, the cloud executes the signature verification algorithm of [69] with random exponent ω and computes the term Δ_0 as given in Fig. 3.2. Next, it verifies the identity $\Gamma \stackrel{?}{=} H_7(\Delta_0^{1/\omega}) \oplus H_8\big(e(\sigma', \sigma'')^{\mathcal{CSK}}\big)$. If the DO has a valid signing key, Δ_0 correctly produces $g_T^{\omega/\theta}$ and hence the identity is true. In this case, the cloud accepts the ciphertext for storage; otherwise it rejects the DO's request. Note here that only the cloud can verify the authenticity of a DO using its secret key $\mathcal{CSK}$.

The description of our ABDSRS is as follows.

1. **System Initialization**
   This phase is specified by the following three setup algorithms.

   $\mathsf{KGC\text{-}Setup}(1^\wp)$. Taking input the security parameter $1^\wp$, this algorithm produces the system public parameters $\mathcal{PP}$ and the system master secret key $\mathcal{MK}$ in the following way.

   - Select a pairing tuple $\Sigma := \langle p, \mathbb{G}, \mathbb{G}_T, e\rangle$ (the details are given in Section 2.2).
   - Sample $g, g_1, g_2, \ldots, g_{15} \xleftarrow{\text{u}} \mathbb{G}$.
   - Choose $\alpha \xleftarrow{\text{u}} \mathbb{Z}_p^*$ and compute $g_T := e(g, g)^\alpha$.
   - Select the message space $\mathcal{M} := \{0,1\}^{\ell_{msg}}$. Choose a key derivation function $\mathsf{KDF} : \mathbb{G}_T \to \{0,1\}^{\ell_{msg}}$.
   - Pick eight collision-resistant hash functions $H_1 : \{0,1\}^* \to \mathbb{G}$, $H_2 : \mathbb{G}_T \to \{0,1\}^{\ell_{H_2}}$, $H_3 : \{0,1\}^* \to \{0,1\}^{\ell_{H_3}}, H_4 : \{0,1\}^* \to \mathbb{Z}_p^*$, $H_5 : \{0,1\}^* \to \mathbb{Z}_p^*, H_6 : \{0,1\}^* \to \mathbb{G}$, $H_7 : \mathbb{G}_T \to \{0,1\}^{\ell_H}$, $H_8 : \mathbb{G}_T \to \{0,1\}^{\ell_H}$ where H_1 and H_6 are two independent hash functions, H_4 and H_5 are two independent hash functions, and H_7 and H_8 are again two independent hash functions.
   - The system public parameters $\mathcal{PP} := \langle \Sigma, g_T, g, \{g_i\}_{i=1}^{15}, \mathcal{M}, \{H_i\}_{i=1}^{8}, \mathsf{KDF}\rangle$.

- The system master secret key $\mathcal{MK} := g^{\alpha}$.

◇ TGC-Setup($\mathcal{PP}$). Taking $\mathcal{PP}$ as input, this algorithm (run by TGC) generates TGC's public and secret keys $\mathcal{TPK}$ and $\mathcal{TSK}$, respectively, as described below.

- Pick $\varpi_1, \varpi_2, \varpi_3, \varpi_4, \tau \xleftarrow{u} \mathbb{Z}_p^*$ and compute $g_{16} := g^{\varpi_1}, g_{17} := g^{\varpi_2}$, $g_{18} := g^{\varpi_3}, g_{19} := g^{\varpi_4}$ and $h_T := e(g, g_{15})^{\tau}$.
- Output $\mathcal{TPK} := \langle g_{16}, g_{17}, g_{18}, g_{19}, h_T \rangle$ and $\mathcal{TSK} := \langle \varpi_1, \varpi_2, \varpi_3, \varpi_4, \tau \rangle$.

◇ Cloud-Setup($\mathcal{PP}$). On input $\mathcal{PP}$, this algorithm (run by cloud) produces cloud's public and secret keys $\mathcal{CPK}$ and $\mathcal{CSK}$, respectively, in the following way.

- Choose $r_c \xleftarrow{u} \mathbb{Z}_p^*$ and calculate $Y_c := g_4^{r_c}$.
- Output $\mathcal{CPK} := Y_c$ and $\mathcal{CSK} := r_c$.

2. **DO and DU Key Generation**

This phase contains the following two key generation algorithms (run by KGC). KGC issues signing key to DO and decryption key to DU.

◇ sKeyGen($\mathcal{PP}, \mathcal{MK}, A_s$). Given input $\mathcal{PP}$, $\mathcal{MK}$ and a signing attribute set $A_s \subset \{0,1\}^*$, this algorithm returns the signing key $\mathcal{SK}_{A_s}$ for A_s as described below.

- Choose $r' \xleftarrow{u} \mathbb{Z}_p^*$ and compute $S_1 := g^{\alpha} g_4^{r'}, S_2 := g^{r'}$, $S_{3,y} := (H_1(y))^{r'}, \forall y \in A_s$.
- The signing key $\mathcal{SK}_{A_s} := \langle A_s, S_1, S_2, \{S_{3,y}\}_{y \in A_s} \rangle$.

◇ dKeyGen($\mathcal{PP}, \mathcal{MK}, \mathcal{CPK}, A_d$). Taking input $\mathcal{PP}$, $\mathcal{MK}, \mathcal{CPK}$ and a decryption attribute set $A_d \subset \mathbb{Z}_p^*$, this algorithm generates the decryption key $\mathcal{DK}_{A_d}$ for A_d as follows.

- Pick $r \xleftarrow{u} \mathbb{Z}_p^*$ and compute $D_1 := g^{\alpha} Y_c^r, D_2 := g^r$.
- For each $x \in A_d$, pick $r_x \xleftarrow{u} \mathbb{Z}_p^*$, and calculate $D_{3,x} := g^{r_x}, D_{4,x} := (g_2^x g_1)^{r_x} g_3^{-r}$.
- The decryption key $\mathcal{DK}_{A_d} := \langle A_d, D_1, D_2, \{D_{3,x}, D_{4,x}\}_{x \in A_d} \rangle$.

3. **Signcryption**

To signcrypt a plaintext, the DO carries out this phase using two sub-algorithms: offline signcryption and online signcryption.

◇ $\mathsf{offSigncrypt}(\mathcal{PP}, \mathcal{SK}_{A_s}, \mathcal{TPK}, \mathcal{CPK}, \mathcal{P}_s)$. Given input $\mathcal{PP}, \mathcal{SK}_{A_s}, \mathcal{TPK}, \mathcal{CPK}$ and a signing policy $\mathcal{P}_s := (\mathbf{M}_s, \rho_s)$ (where $\mathbf{M}_s$ represents a matrix with dimension $\ell_s \times n_s$) satisfying $\mathcal{P}_s(A_s) = true$, it generates the intermediate ciphertext $\mathcal{IC}$ by carrying out the steps given below.

- As $\mathcal{P}_s(A_s) = true$, we can calculate $\vec{a} := (a_1, a_2, \ldots, a_{\ell_s}) \leftarrow \mathsf{Reconstruct}(\mathbf{M}_s, \rho_s, A_s)$, satisfying $\vec{a} \cdot \mathbf{M}_s = \vec{1}_{n_s}$, i.e., $\sum_{i \in [\ell_s]} a_i \cdot \vec{M}_s^{(i)} = \vec{1}_{n_s}$ and $a_i = 0$ $\forall i$ satisfying $\rho_s(i) \notin A_s$. For the matrix $\mathbf{M}_s$, the ith row is denoted as $\vec{M}_s^{(i)}$.
- Sample $(b_1, b_2, \ldots, b_{\ell_s}) \xleftarrow{\text{u}} \big\{(b_1, b_2, \ldots, b_{\ell_s}) \in \mathbb{Z}_p^{\ell_s} | \sum_{i \in [\ell_s]} b_i \vec{M}_s^{(i)} = \vec{0}_{n_s}\big\}$.
- Re-randomize the signing key $\mathcal{SK}_{A_s}$ as follows: pick $r'' \xleftarrow{\text{u}} \mathbb{Z}_p^*$ and set

$$\begin{aligned}\mathcal{SK}_{A_s} &:= \langle A_s, S_1 := S_1 g_4^{r''}, S_2 := S_2 g^{r''}, \{S_{3,y} := S_{3,y}(H_1(y))^{r''}\}_{y \in A_s}\rangle \\ &= \langle A_s, S_1 := g^{\alpha} g_4^{\breve{r}}, S_2 := g^{\breve{r}}, \{S_{3,y} := (H_1(y))^{\breve{r}}\}_{y \in A_s}\rangle, \\ &\qquad \text{where } \breve{r} := r' + r''\end{aligned}$$

- Select $\beta, \theta, o_1, o_2, \delta', \delta'' \xleftarrow{\text{u}} \mathbb{Z}_p^*$ and set

$$\Omega_s' := \begin{pmatrix} \mathcal{P}_s, \theta, o_1, \sigma' := g^{\delta'}, \sigma'' := g_4^{\delta''}, \Gamma := H_7(g_T^{1/\theta}) \oplus H_8\big(e(\sigma', Y_c)^{\delta''}\big), \\ \sigma := S_1^{\frac{1}{\theta}} \big(g_5^{o_1} g_6\big)^{\beta} \prod_{i \in [\ell_s]} \big(S_{3,\rho_s(i)}^{\frac{a_i}{\theta}} \cdot H_1(\rho_s(i))^{o_2 b_i}\big), \\ \big\{\sigma_i := S_2^{\frac{a_i}{\theta}} g^{o_2 b_i}\big\}_{i \in [\ell_s]} \end{pmatrix}$$

- Pick $\xi', \eta, \delta''' \xleftarrow{\text{u}} \mathbb{Z}_p^*$ and for each $j \in [\mathfrak{m}_e]$, select $\lambda_j', t_j, \zeta_j \xleftarrow{\text{u}} \mathbb{Z}_p^*$, where $\mathfrak{m}_e \in \mathbb{N}$; and set

$$\Omega_e' := \begin{pmatrix} \beta, \xi', \eta, \{\lambda_j', t_j, \zeta_j\}_{j \in [\mathfrak{m}_e]}, \partial := H_5\big(e(\sigma', Y_c)^{\delta'''}\big), \\ E' := g_4^{\delta'''}, E_1 := g^{\beta}, E_2 := (g_7^{\xi'} g_8^{\eta} g_9)^{\beta}, \\ \big\{\vec{E}_j := \big(g_4^{\lambda_j'} g_3^{t_j}, (g_2^{\zeta_j} g_1)^{t_j}, g^{t_j}\big)\big\}_{j \in [\mathfrak{m}_e]}, \\ \mathsf{tag1} := H_2\big(g_T^{\beta + \frac{1}{\theta}}\big), \mathsf{key} := \mathsf{KDF}\big(g_T^{\beta + \frac{1}{\theta}}\big) \end{pmatrix}$$

- Choose $f_1, f_2, f_3, f_4 \xleftarrow{\text{u}} \mathbb{Z}_p^*$ and for each $j \in [\mathfrak{m}_k]$,

pick $\breve{t}_j, w'_j, \pi_{j1}, \pi_{j2} \xleftarrow{\text{u}} \mathbb{Z}_p^*$, where $\mathfrak{m}_k \in \mathbb{N}$; and set

$$\Omega'_k := \left(\begin{array}{l} k_T := h_T^{\beta}, , \vec{L} := \big(g_{16}^{f_1}, g_{17}^{f_2}, g_{18}^{f_3}, g_{19}^{f_4}\big), \\ \{\vec{K}_j, \vec{k}_j, \breve{t}_j, w'_j\}_{j\in[\mathfrak{m}_k]}, \text{ where} \\ \vec{K}_j := \big((g_{11}^{w'_j} g_{12})^{\breve{t}_j} g_{10}^{-\beta}, (g_{11}^{w'_j} g_{12})^{\breve{t}_j} g_{13}^{-\beta}\big), \\ \vec{k}_j := (\breve{t}_j - \pi_{j1} - f_1, \pi_{j1} - f_2, \breve{t}_j - \pi_{j2} - f_3, \pi_{j2} - f_4) \end{array} \right)$$

- The intermediate ciphertext $\mathcal{IC} := \langle \Omega'_s, \Omega'_e, \Omega'_k \rangle$.

◇ onSigncrypt$(\mathcal{PP}, \mathcal{IC}, msg, \mathcal{P}_e, W)$. Given input $\mathcal{PP}$, an encryption policy $\mathcal{P}_e := (\mathbf{M}_e, \rho_e)$ (where $\mathbf{M}_e$ represents a matrix of dimension $\ell_e \times n_e$), $\mathcal{IC}$, a plaintext $msg \in \mathcal{M}$, and a set $W := \{[\mathcal{W}_1 : w_1], \ldots, [\mathcal{W}_\varsigma : w_\varsigma]\}$ of keywords (where $\mathcal{W}_i$ represents generic keyword name and w_i represents corresponding keyword value), it generates the final ciphertext $\mathcal{CT}$ as follows.

- Compute $ct := msg \oplus \mathsf{key}, \mathsf{tag2} := H_3(\mathsf{tag1}||ct)$ and $(\lambda_1, \ldots, \lambda_{\ell_e}) \leftarrow \mathsf{Share}(\mathbf{M}_e, \rho_e, \beta \cdot \partial)$.
- Set $\vec{\varepsilon_i} := \big(\lambda_i - \lambda'_i, t_i(\rho_e(i) - \zeta_i)\big), \forall i \in [\ell_e]$ and

$$\Omega_e := \langle \mathcal{P}_e, ct, E', E_1, \{\vec{E}_i, \vec{\varepsilon_i}\}_{i\in[\ell_e]} \rangle.$$

- Compute $\breve{o}_1 := H_4(ct||\Gamma||E'||\mathcal{P}_e||\mathcal{P}_s||W^\circ)$, $\chi := \beta(\breve{o}_1 - o_1)$ and set

$$\Omega_s := \langle \mathcal{P}_s, \sigma', \sigma'', \Gamma, \sigma, \{\sigma_i\}_{i\in[\ell_s]}, \chi \rangle.$$

- Calculate $u_i := \breve{t}_i(w_i - w'_i)$, for all $i \in [\varsigma]$ and set

$$\Omega_k := \langle W^\circ, k_T, \vec{L}, \{\vec{K}_i, \vec{k}_i, u_i\}_{i\in[\varsigma]} \rangle.$$

- Compute $\xi := H_5(\Omega_s||\Omega_e||\Omega_k||\mathsf{tag2})$ and $\varepsilon_2 := \beta(\xi - \xi')$.
- The final ciphertext $\mathcal{CT} := \langle \Omega_s, \Omega_e, \Omega_k, \mathsf{tag2}, E_2, \varepsilon_2, \eta \rangle$.

Remark 3. *Now, DO outsources the ciphertext $\mathcal{CT}$ to the cloud. Then, cloud accepts $\mathcal{CT}$ for storage if the DO is legitimate. Figure 3.2 illustrates the data storage phase, which is given at the end of the construction.*

Remark 4. *The distribution of the ciphertext*

$$\mathcal{CT} := \left(\begin{array}{l} \Omega_s := \langle \mathcal{P}_s, \sigma', \sigma'', \Gamma, \sigma, \{\sigma_i\}_{i\in[\ell_s]}, \chi\rangle, \\ \Omega_e := \langle \mathcal{P}_e, ct, E', E_1, \{\vec{E}_i := (E_{i1}, E_{i2}, E_{i3}), \vec{\varepsilon}_i := (\varepsilon_{i1}, \varepsilon_{i2})\}_{i\in[\ell_e]}\rangle, \\ \Omega_k := \langle W^{\circ}, k_T, \vec{L} := (L_1, L_2, L_3, L_4), \{\vec{K}_j := (K_{j1}, K_{j2}), \\ \qquad \vec{k}_j := (k_{j1}, k_{j2}, k_{j3}, k_{j4}), u_j\}_{j\in[\varsigma]}\rangle, \\ \mathsf{tag2}, E_2, \varepsilon_2, \eta \end{array} \right)$$

of a data file msg for the signing policy $\mathcal{P}_s := (\mathbf{M}_s, \rho_s)$*, encryption policy* $\mathcal{P}_e := (\mathbf{M}_e, \rho_e)$*, where* $\mathbf{M}_s$ *(resp.* $\mathbf{M}_e$*) represents a matrix of dimension* $\ell_s \times n_s$ *(resp.* $\ell_e \times n_e$*), and keyword set* $W := \{[\mathcal{W}_1 : w_1], \ldots, [\mathcal{W}_\varsigma : w_\varsigma]\}$ *is of the form* $\{\sigma', \sigma'', \sigma, \sigma_i, E', E_1, E_2, E_{i1}, E_{i2}, E_{i3}, L_1, L_2, L_3, L_4, K_{j1}, K_{j2}\} \subset \mathbb{G}$*,* $\Gamma \in \{0,1\}^{\ell_H}$*,* $ct \in \{0,1\}^{\ell_{msg}}$*,* $k_T \in \mathbb{G}_T$*,* $\mathsf{tag2} \in \{0,1\}^{\ell_{H_3}}$*,* $\{\chi, \varepsilon_{i1}, \varepsilon_{i2}, k_{j1}, k_{j2}, k_{j3}, k_{j4}, u_j, \varepsilon_2, \eta\} \subset \mathbb{Z}_p^*$ *and*

$$\left\{ \begin{array}{l} E_1 := g^{\beta}, \Gamma := H_7(g_T^{1/\theta}) \oplus H_8\big(e(g,g_4)^{\delta'\cdot\delta''\cdot\mathcal{CSK}}\big), ct := msg \oplus \mathsf{KDF}\big(g_T^{\beta+\frac{1}{\theta}}\big), \\ \mathsf{tag2} := H_3\big(H_2(g_T^{\beta+\frac{1}{\theta}})||ct\big), \sigma' := g^{\delta'}, \sigma'' := g_4^{\delta''}, \\ \sigma := g^{\frac{\alpha}{\theta}} g_4^{\frac{\breve{r}}{\theta}} \big(g_5^{\breve{o}_1} g_6\big)^{\beta} \big(\prod_{i\in[\ell_s]} H_1\big(\rho_s(i)\big)^{\frac{\breve{r}a_i}{\theta}+o_2 b_i}\big) g_5^{-\chi}, \sigma_i := g^{\frac{\breve{r}a_i}{\theta}+o_2 b_i}, E' := g_4^{\delta'''}, \\ E_{i1} := g_4^{\lambda_i} g_3^{t_i} \cdot g_4^{-\varepsilon_{i1}}, E_{i2} := (g_2^{\rho_e(i)} g_1)^{t_i} \cdot g_2^{-\varepsilon_{i2}}, E_{i3} := g^{t_i}, \\ L_1 := g_{16}^{f_1}, L_2 := g_{17}^{f_2}, L_3 := g_{18}^{f_3}, L_4 := g_{19}^{f_4}, \\ K_{j1} := (g_{11}^{w_j} g_{12})^{\breve{t}_j} g_{10}^{-\beta} \cdot g_{11}^{-u_j}, K_{j2} := (g_{11}^{w_j} g_{12})^{\breve{t}_j} g_{13}^{-\beta} \cdot g_{11}^{-u_j}, \\ k_{j1} := \breve{t}_j - \pi_{j1} - f_1, k_{j2} := \pi_{j1} - f_2, k_{j3} := \breve{t}_j - \pi_{j2} - f_3, k_{j4} := \pi_{j2} - f_4, \\ k_T := e(g, g_{15})^{\tau\beta}, E_2 := (g_7^{\xi} g_8^{\eta} g_9)^{\beta} \cdot g_7^{-\varepsilon_2} \end{array} \right\} \tag{3.1}$$

where $\theta, \beta, \delta', \delta'', \delta''', \breve{r}, o_2, \eta, t_i, f_1, f_2, f_3, f_4, \breve{t}_j, \pi_{j1}, \pi_{j2}, \chi, \varepsilon_2, \varepsilon_{i1}, \varepsilon_{i2}, u_j$ *are random exponents,* λ_i *is the* i*th share of* $\beta \cdot H_5\big(e(g,g_4)^{\delta'\delta'''\cdot\mathcal{CSK}}\big)$ *with respect to* $(\mathbf{M}_e, \rho_e)$*,* $\breve{o}_1 := H_4(ct||\Gamma||E'||\mathcal{P}_e||\mathcal{P}_s||W^{\circ})$*,* $\xi := H_5(\Omega_s||\Omega_e||\Omega_k||\mathsf{tag2})$*,* $(a_1, a_2, \ldots, a_{\ell_s})$ *and* $(b_1, b_2, \ldots, b_{\ell_s})$ *are vectors satisfying respectively* $\sum_{i\in[\ell_s]} a_i \cdot \vec{M}_s^{(i)} = \vec{1}_{n_s}$ *and* $\sum_{i\in[\ell_s]} b_i \cdot \vec{M}_s^{(i)} = \vec{0}_{n_s}$*,* α *is the system master key,* τ *is TGC secret key,* $\mathcal{CSK}$ *is cloud secret key, and others are public parameters of the system.* □

We use this distribution of $\mathcal{CT}$ *in correctness and security analysis of the proposed system.*

4. **Keyword Policy Trapdoor Generation** To generate a trapdoor for a keyword policy, the TGC performs this phase in two sub-algorithms: offline trapdoor generation and online trapdoor generation.

◇ $\mathsf{offTrapGen}(\mathcal{PP}, \mathcal{CPK}, \mathcal{TSK})$. Given input $\mathcal{PP}, \mathcal{CPK}$ and $\mathcal{TSK}$, it produces the intermediate trapdoor $\mathcal{IT}$.

- Pick $f, \breve{f}, f' \xleftarrow{u} \mathbb{Z}_p^*$, and for each $i \in [\mathfrak{m}_t]$, choose $\breve{r}_i, \breve{r}'_i, \kappa_i, \vartheta'_i \xleftarrow{u} \mathbb{Z}_p^*$, here $\mathfrak{m}_t \in \mathbb{N}$.
- Set $\mathcal{IT} := \left(\begin{array}{l} T_0 := g^f, T_1 := g^{\breve{f}}, T_2 := g_4^{f'}, \mathsf{ot} := e(T_1, Y_c)^{f'}, \\ \{\breve{r}_i, \breve{r}'_i, \kappa_i, \vartheta'_i, \varrho_i, \vec{\breve{T}}_{1i}, \vec{T}_{2i}\}_{i \in [\mathfrak{m}_t]}, \text{ where} \\ \varrho_i := \varpi_1 \varpi_2 \breve{r}_i + \varpi_3 \varpi_4 \breve{r}'_i - f, \\ \vec{\breve{T}}_{1i} := \big(g_{15}^{\vartheta'_i} \cdot g_{10}^{\varpi_1 \varpi_2 \breve{r}_i + \varpi_3 \varpi_4 \breve{r}'_i}, g_{13}^{\varpi_1 \varpi_2 \breve{r}_i + \varpi_3 \varpi_4 \breve{r}'_i}\big), \\ \vec{T}_{2i} := \big((g_{11}^{\kappa_i} \cdot g_{12})^{-\breve{r}_i \varpi_1}, (g_{11}^{\kappa_i} \cdot g_{12})^{-\breve{r}_i \varpi_2}, (g_{11}^{\kappa_i} \cdot g_{12})^{-\breve{r}'_i \varpi_3}, \\ \qquad (g_{11}^{\kappa_i} \cdot g_{12})^{-\breve{r}'_i \varpi_4}\big) \end{array} \right)$

◇ $\mathsf{onTrapGen}(\mathcal{PP}, \mathcal{IT}, \mathcal{TSK}, \mathcal{P}_t)$. On input $\mathcal{PP}, \mathcal{IT}, \mathcal{TSK}$ and a keyword policy $\mathcal{P}_t := (\mathbf{M}_t, \rho_t^\circ, \{w_{\rho_t^\circ(i)}\}_{i \in [\ell_t]})$ (where $\mathbf{M}_t$ represents a matrix of dimension $\ell_t \times n_t$, the rows of $\mathbf{M}_t$ are mapped to generic keyword names via the function ρ_t° and the associated keyword value is denoted as $\{w_{\rho_t^\circ(i)}\}_{i \in [\ell_t]}$), it generates the trapdoor $\widetilde{\mathcal{TD}}_{\mathcal{P}_t}$.

- Compute $(\vartheta_1, \ldots, \vartheta_{\ell_t}) \leftarrow \mathsf{Share}(\mathbf{M}_t, \rho_t^\circ, \tau \cdot H_5(\mathsf{ot}))$
- For each $i \in [\ell_t]$, calculate $B_{1i} := \vartheta_i - \vartheta'_i$ and $\vec{B}_{2i} := \big(\breve{r}_i(-w_{\rho_t^\circ(i)} + \kappa_i)\varpi_1, \breve{r}_i(-w_{\rho_t^\circ(i)} + \kappa_i)\varpi_2, \breve{r}'_i(-w_{\rho_t^\circ(i)} + \kappa_i)\varpi_3, \breve{r}'_i(-w_{\rho_t^\circ(i)} + \kappa_i)\varpi_4\big)$.
- Set $\vec{\breve{T}}_{1i} := (\breve{T}_{1i1}, H_6(\mathsf{ot} || \mathcal{P}_t^\circ || M_t^{(i)}) \cdot \breve{T}_{1i2})$ for $i \in [\ell_t]$.
- The trapdoor $\widetilde{\mathcal{TD}}_{\mathcal{P}_t} := \langle \mathcal{P}_t, T_0, T_1, T_2, \{\varrho_i, \vec{\breve{T}}_{1i}, \vec{T}_{2i}, B_{1i}, \vec{B}_{2i}\}_{i \in [\ell_t]} \rangle$.

5. **Data Retrieval Request Generation**
Once the DU receives the trapdoor $\widetilde{\mathcal{TD}}_{\mathcal{P}_t}$ from TGC, it proceeds to execute two algorithms described below in order to create a data retrieval request ($\mathcal{DRR}$).

◇ $\mathsf{offDataRetReq}(\mathcal{PP}, \mathcal{DK}_{A_d}, \mathcal{P}_t)$. On input $\mathcal{PP}, \mathcal{DK}_{A_d}, \mathcal{P}_t$, this offline data retrieval request generation algorithm produces the intermediate data retrieval request $\mathcal{IDR}$. Here $\mathcal{DK}_{A_d} := \langle A_d, D_1, D_2, \{D_{3,x}, D_{4,x}\}_{x \in A_d} \rangle$.

- Choose $\gamma, \breve{\tau}, t_r \xleftarrow{u} \mathbb{Z}_p^*$, calculate $trk := \breve{\tau}/t_r$, $(\breve{\vartheta}_1, \ldots, \breve{\vartheta}_{\ell_t}) \leftarrow \mathsf{Share}(\mathbf{M}_t, \rho_t^\circ, trk)$ and $\{g_{14}^{\breve{\vartheta}_i}\}_{i \in [\ell_t]}$.
Note that $\mathcal{P}_t^\circ := (\mathbf{M}_t, \rho_t^\circ)$.

- Set $\mathcal{IDR} := \left(\begin{array}{l} \{g_{14}^{\breve{\vartheta}_i}\}_{i\in[\ell_t]}, \\ \mathcal{TK}_{A_d} := \langle A_d, T_1', T_2', \{T_{3,x}', T_{4,x}'\}_{x\in A_d}\rangle, \text{ where} \\ T_1' := (D_1 \cdot g_{14}^{\breve{\tau}})^{1/\gamma}, T_2' := D_2^{1/\gamma}, T_{3,x}' := D_{3,x}^{1/\gamma}, \\ T_{4,x}' := D_{4,x}^{1/\gamma}, \mathcal{TDK} := \langle tdk_1, tdk_2\rangle, \\ \text{where } tdk_1 := t_r, tdk_2 := \gamma \end{array} \right)$

◇ $\mathsf{onDataRetReq}(\mathcal{PP}, \mathcal{IDR}, \widetilde{\mathcal{TD}}_{\mathcal{P}_t})$. Taking $\mathcal{PP}, \mathcal{IDR}, \widetilde{\mathcal{TD}}_{\mathcal{P}_t}$, it generates a secret transformation decryption key $\mathcal{TDK} := \langle tdk_1, tdk_2\rangle$ and the data retrieval request $\mathcal{DRR}$. Note that $\mathcal{TDK}$ will be used to decrypt the transformed ciphertexts received from the cloud.

- Compute $T_{1i2} := g_{14}^{\breve{\vartheta}_i} \cdot \widetilde{T}_{1i2}$ and set $T_{1i1} := \widetilde{T}_{1i1}$, and hence $\vec{T}_{1i} := (\widetilde{T}_{1i1}, T_{1i2})$.
- Set $\mathcal{DRR} := \langle \mathcal{TD}_{\mathcal{P}_t^\circ}, \mathcal{TK}_{A_d}\rangle$, where $\mathcal{TD}_{\mathcal{P}_t^\circ} := \langle \mathcal{P}_t^\circ, T_0, T_1, T_2, \{\varrho_i, \vec{T}_{1i}, \vec{T}_{2i}, B_{1i}, \vec{B}_{2i}\}_{i\in[\ell_t]}\rangle$ is transform trapdoor for $\mathcal{P}_t$ and $\mathcal{TK}_{A_d}$ is transform key for the decryption attribute set A_d.

Remark 5. *The distribution of the data retrieval request*

$$\mathcal{DRR} := \left(\begin{array}{l} \mathcal{TD}_{\mathcal{P}_t^\circ} := \left\langle \begin{array}{l} \mathcal{P}_t^\circ, T_0, T_1, T_2, \{\varrho_i, \vec{T}_{1i} := (T_{1i1}, T_{1i2}), \\ \vec{T}_{2i} := (T_{2i1}, T_{2i2}, T_{2i3}, T_{2i4})\}_{i\in[\ell_t]}, \\ \{B_{1i}, \vec{B}_{2i} := (B_{2i1}, B_{2i2}, B_{2i3}, B_{2i4})\}_{i\in[\ell_t]} \end{array} \right\rangle, \\ \mathcal{TK}_{A_d} := \left\langle \; A_d, T_1', T_2', \{T_{3,x}', T_{4,x}'\}_{x\in A_d} \; \right\rangle \end{array} \right)$$

of a keyword policy $\mathcal{P}_t := (\mathbf{M}_t, \rho_t^\circ, \{w_{\rho_t^\circ(i)}\}_{i\in[\ell_t]})$ *is of the form* $\{T_0, T_1, T_2, T_{1i1}, T_{1i2}, T_{2i1}, T_{2i2}, T_{2i3}, T_{2i4}, T_1', T_2', T_{3,x}', T_{4,x}'\} \subset \mathbb{G}$, $\{\varrho_i, B_{1i}, B_{2i1}, B_{2i2}, B_{2i3}, B_{2i4}\} \subset \mathbb{Z}_p^*$ *and*

$$\left\{ \begin{array}{l} T_0 := g^f, T_1 := g^{\breve{f}}, T_2 := g_4^{f'}, \varrho_i := \varpi_1\varpi_2\breve{r}_i + \varpi_3\varpi_4\breve{r}_i' - f, \\ T_{1i1} := g_{15}^{\vartheta_i} \cdot g_{10}^{\varpi_1\varpi_2\breve{r}_i + \varpi_3\varpi_4\breve{r}_i'} \cdot g_{15}^{-B_{1i}}, \\ T_{1i2} := g_{14}^{\breve{\vartheta}_i} \cdot H_6\big(e(g,g_4)^{\breve{f}f'\cdot\mathcal{CSK}} || \mathcal{P}_t^\circ || M_t^{(i)}\big) \cdot g_{13}^{\varpi_1\varpi_2\breve{r}_i + \varpi_3\varpi_4\breve{r}_i'}, \\ T_{2i1} := \big(g_{11}^{w_{\rho_t^\circ(i)}} \cdot g_{12}\big)^{-\breve{r}_i\varpi_1} \cdot g_{11}^{-B_{2i1}}, T_{2i2} := \big(g_{11}^{w_{\rho_t^\circ(i)}} \cdot g_{12}\big)^{-\breve{r}_i\varpi_2} \cdot g_{11}^{-B_{2i2}}, \\ T_{2i3} := \big(g_{11}^{w_{\rho_t^\circ(i)}} \cdot g_{12}\big)^{-\breve{r}_i'\varpi_3} \cdot g_{11}^{-B_{2i3}}, T_{2i4} := \big(g_{11}^{w_{\rho_t^\circ(i)}} \cdot g_{12}\big)^{-\breve{r}_i'\varpi_4} \cdot g_{11}^{-B_{2i4}}, \\ T_1' := g^{\alpha/\gamma} g_4^{(\mathcal{CSK}\cdot\widehat{r})} g_{14}^{\breve{\tau}/\gamma}, T_2' := g^{\widehat{r}}, T_{3,x}' := g^{\widehat{r}_x}, T_{4,x}' := (g_2^x g_1)^{\widehat{r}_x} g_3^{-\widehat{r}}, \end{array} \right\} \tag{3.2}$$

where $f, \breve{f}, f', \breve{r}_i, \breve{r}_i', \widehat{r}, \gamma, \breve{\tau}, \widehat{r}_x, t_r, B_{1i}, B_{2i1}, B_{2i2}, B_{2i3}, B_{2i4}$ *are random exponents (elements in* $\mathbb{Z}_p^*$*),* ϑ_i *(resp.* $\breve{\vartheta}_i$*) is the* i*th share of* $\tau \cdot H_5(e(g,g_4)^{\breve{f}f'\cdot\mathcal{CSK}})$ *(resp.* $trk := \breve{\tau}/t_r$*) with respect to the policy* $(\mathbf{M}_t, \rho_t^\circ)$*,* $\langle \tau, \varpi_1, \varpi_2, \varpi_3, \varpi_4\rangle$ *is TGC's*

secret key, α is system's master secret key, $\mathcal{CSK}$ is cloud secret key, and others are system public parameters. □

We use this distribution of $\mathcal{DRR}$ in correctness and security analysis of ABD-SRS.

6. **Data Retrieval**

The cloud uses two steps, test and transform, to finish this phase after receiving $\mathcal{DRR}$ from a DU.

◇ $\mathsf{Test}(\mathcal{PP}, \mathcal{CT}, \mathcal{DRR}, \mathcal{CSK})$. Taking $\mathcal{PP}, \mathcal{CT}, \mathcal{DRR} := \langle \mathcal{TD}_{\mathcal{P}_t^\circ}, \mathcal{TK}_{A_d} \rangle, \mathcal{CSK}$ as input, the cloud server outputs either $\mathcal{CT}$ or $\perp$ by carrying out the following steps. $\mathcal{TD}_{\mathcal{P}_t^\circ}$ is parsed as $\mathcal{TD}_{\mathcal{P}_t^\circ} := \langle \mathcal{P}_t^\circ, T_0, T_1, T_2, \{\varrho_i, \vec{T}_{1i}, \vec{T}_{2i}, B_{1i}, \vec{B}_{2i}\}_{i\in[\ell_t]} \rangle$.

- Compute $\xi := H_5(\Omega_s || \Omega_e || \Omega_k || \mathsf{tag2})$ and check

$$e\big(E_2 \cdot g_7^{\varepsilon_2}, g\big) \stackrel{?}{=} e\big(g_7^{\xi} g_8^{\eta} g_9, E_1\big)$$

  If above equation do not hold, output $\perp$. Otherwise, proceed further.

- Set $\vec{a}' := (a'_1, a'_2, \ldots, a'_{\ell_t}) \leftarrow \mathsf{Reconstruct}(\mathbf{M}_t, \rho_t^\circ, W^\circ)$ and calculate

$$\begin{aligned} \mathsf{ot} &:= e(T_1, T_2)^{\mathcal{CSK}} \\ X_1 &:= e\Big(E_1, \prod_{i\in[\ell_t]} \big(T_{1i1} \cdot g_{15}^{B_{1i}}\big)^{a'_i}\Big) \cdot e\Big(g, \prod_{i\in[\ell_t]} \big(K_{\rho_t^\circ(i)1} \cdot g_{11}^{u_{\rho_t^\circ(i)}}\big)^{a'_i \varrho_i}\Big) \\ &\quad \cdot e\Big(T_0, \prod_{i\in[\ell_t]} \big(K_{\rho_t^\circ(i)1} \cdot g_{11}^{u_{\rho_t^\circ(i)}}\big)^{a'_i}\Big) \end{aligned} \tag{3.3}$$

$$\begin{aligned} X_2 &:= e\Big(g_{16}, \prod_{i\in[\ell_t]} \big(T_{2i2} \cdot g_{11}^{B_{2i2}}\big)^{a'_i k_{\rho_t^\circ(i)1}}\Big) \cdot e\Big(L_1, \prod_{i\in[\ell_t]} \big(T_{2i2} \cdot g_{11}^{B_{2i2}}\big)^{a'_i}\Big) \\ &\quad \cdot e\Big(g_{17}, \prod_{i\in[\ell_t]} \big(T_{2i1} \cdot g_{11}^{B_{2i1}}\big)^{a'_i k_{\rho_t^\circ(i)2}}\Big) \cdot e\Big(L_2, \prod_{i\in[\ell_t]} \big(T_{2i1} \cdot g_{11}^{B_{2i1}}\big)^{a'_i}\Big) \\ &\quad \cdot e\Big(g_{18}, \prod_{i\in[\ell_t]} \big(T_{2i4} \cdot g_{11}^{B_{2i4}}\big)^{a'_i k_{\rho_t^\circ(i)3}}\Big) \cdot e\Big(L_3, \prod_{i\in[\ell_t]} \big(T_{2i4} \cdot g_{11}^{B_{2i4}}\big)^{a'_i}\Big) \\ &\quad \cdot e\Big(g_{19}, \prod_{i\in[\ell_t]} \big(T_{2i3} \cdot g_{11}^{B_{2i3}}\big)^{a'_i k_{\rho_t^\circ(i)4}}\Big) \cdot e\Big(L_4, \prod_{i\in[\ell_t]} \big(T_{2i3} \cdot g_{11}^{B_{2i3}}\big)^{a'_i}\Big) \end{aligned} \tag{3.4}$$

- Verify that $X_1 X_2 \stackrel{?}{=} k_T^{H_5(\mathsf{ot})}$. If it does not hold, the algorithm outputs

$\perp$. Otherwise, the trapdoor $\mathcal{TD}_{\mathcal{P}_t^\circ}$ is matched by the ciphertext $\mathcal{CT}$, i.e., $\mathcal{P}_t^\circ(W^\circ) = true$ ($\mathcal{P}_t(W) = true$, implicitly). As a result, the matching ciphertext $\mathcal{CT}$ is generated.

Note. Even though we mentioned $\mathcal{DRR}$ in the input of Test algorithm, actually Test uses only the trapdoor $\mathcal{TD}_{\mathcal{P}_t^\circ}$ (in fact, $\widetilde{\mathcal{TD}}_{\mathcal{P}_t}$). We use this fact to answer test queries in IND-CKA security.

Correctness. By Remark 5, one can see that if $\mathcal{P}_t(W) = true$, then

$$\begin{aligned} X_1 &= e\big(g, g_{15}\big)^{\beta\tau H_5(\mathsf{ot})} \cdot e\Big(g, \prod_{i\in[\ell_t]} \big(g_{11}^{w_{\rho_t^\circ(i)}} \cdot g_{12}\big)^{\breve{t}_{\rho_t^\circ(i)}\cdot a'_i\cdot(\varpi_1\varpi_2\breve{r}_i+\varpi_3\varpi_4\breve{r}'_i)}\Big) \\ X_2 &= e\Big(g, \prod_{i\in[\ell_t]} \big(g_{11}^{w_{\rho_t^\circ(i)}} \cdot g_{12}\big)^{-\breve{t}_{\rho_t^\circ(i)}\cdot a'_i\cdot(\varpi_1\varpi_2\breve{r}_i+\varpi_3\varpi_4\breve{r}'_i)}\Big) \\ X_1X_2 &= e\big(g, g_{15}\big)^{\beta\tau H_5(\mathsf{ot})} = k_T^{H_5(\mathsf{ot})} \end{aligned}$$

◇ $\mathsf{Transform}(\mathcal{PP}, \mathcal{CT}, \mathcal{DRR}, \mathcal{CSK})$. Taking $\mathcal{PP}, \mathcal{CT}, \mathcal{DRR} := \langle \mathcal{TD}_{\mathcal{P}_t^\circ}, \mathcal{TK}_{A_d}\rangle$, $\mathcal{CSK}$ as input, the cloud server outputs either transformed ciphertext $\mathcal{CT}_{out}$ or $\perp$ by executing the following steps. $\mathcal{TK}_{A_d}$ is parsed as $\mathcal{TK}_{A_d} := \langle A_d, T'_1, T'_2, \{T'_{3,x}, T'_{4,x}\}_{x\in A_d}\rangle$.

- If $\mathsf{Test}(\mathcal{PP}, \mathcal{CT}, \mathcal{DRR}, \mathcal{CSK}) \to \perp$ or $\mathcal{P}_e(A_d) = false$, output $\perp$. Otherwise, proceed further.
- Consider X_2 from previous algorithm and compute

$$\begin{aligned} \breve{X}_1 &:= e\Big(E_1, \prod_{i\in[\ell_t]} \big(T_{1i2}\cdot H_6(\mathsf{ot}||\mathcal{P}_t^\circ||M_t^{(i)})^{-1}\big)^{a'_i}\Big) \\ &\quad \times e\Big(g, \prod_{i\in[\ell_t]} \big(K_{\rho_t^\circ(i)2}\cdot g_{11}^{u_{\rho_t^\circ(i)}}\big)^{a'_i\varrho_i}\Big) \\ &\quad \times e\Big(T_0, \prod_{i\in[\ell_t]} \big(K_{\rho_t^\circ(i)2}\cdot g_{11}^{u_{\rho_t^\circ(i)}}\big)^{a'_i}\Big) \qquad (3.5) \\ \Delta_1 &:= \breve{X}_1X_2 \end{aligned}$$

- Since $\mathcal{P}_e(A_d) = true$, calculate $\vec{a}'' := (a''_1, a''_2, \ldots, a''_{\ell_e}) \leftarrow \mathsf{Reconstruct}(\mathbf{M}_e, \rho_e, A_d)$,

$$\begin{aligned} \partial &:= H_5\big(e(\sigma', E')^{\mathcal{CSK}}\big) \\ X_3 &:= e(T'_1, E_1) \end{aligned}$$

$$
\begin{aligned}
X_4 &:= e\Big(T'_2, \prod_{i\in[\ell_e]} \big(E_{i1}\cdot g_4^{\varepsilon_{i1}}\big)^{a''_i}\Big) \\
&\quad \times \prod_{i\in[\ell_e]} \Big(e\big(T'_{3,\rho_e(i)}, E_{i2}\cdot g_2^{\varepsilon_{i2}}\big)^{-1}\cdot e\big(T'_{4,\rho_e(i)}, E_{i3}\big)\Big)^{a''_i} \\
\Delta_2 &:= X_3\cdot (X_4)^{-\mathcal{CSK}/\partial}
\end{aligned}
$$

- Choose $\varphi \xleftarrow{u} \mathbb{Z}_p^*$ and obtain $(\lambda''_1,\ldots,\lambda''_{\ell_s}) \leftarrow \mathsf{Share}(\mathbf{M}_s,\rho_s,\varphi)$
- Calculate $\breve{o} := H_4(ct||\Gamma||E'||\mathcal{P}_e||\mathcal{P}_s||W^\circ)$,

$$
\Delta_3 := \left(\frac{e(\sigma\cdot g_5^{\chi}, g^{\varphi})}{e\big(g_5^{\breve{o}}g_6, E_1^{\varphi}\big)\cdot\prod_{i\in[\ell_s]} e\big(g_4^{\lambda''_i}H_1(\rho_s(i))^{\varphi},\sigma_i\big)}\right)^{1/\varphi} \tag{3.6}
$$

- The transformed ciphertext $\mathcal{CT}_{out} := \langle \Delta_1,\Delta_2,\Delta_3,ct,\mathsf{tag2}\rangle$.

Correctness. By Remark 4 and Remark 5, we can see that if $\mathsf{Test}(\mathcal{PP},\mathcal{CT},\mathcal{DRR},\mathcal{CSK}) \to \mathcal{CT}$ and $\mathcal{P}_e(A_d) = true$, then

$$
\begin{aligned}
\breve{X}_1 &= e\big(g,g_{14}\big)^{\beta\cdot trk}\cdot e\Big(g,\prod_{i\in[\ell_t]}\big(g_{11}^{w_{\rho_t^\circ(i)}}\cdot g_{12}\big)^{\breve{t}_{\rho_t^\circ(i)}\cdot a'_i(\varpi_1\varpi_2\breve{r}_i+\varpi_3\varpi_4\breve{r}'_i)}\Big) \\
X_2 &= e\Big(g,\prod_{i\in[\ell_t]}\big(g_{11}^{w_{\rho_t^\circ(i)}}\cdot g_{12}\big)^{-\breve{t}_{\rho_t^\circ(i)}\cdot a'_i(\varpi_1\varpi_2\breve{r}_i+\varpi_3\varpi_4\breve{r}'_i)}\Big) \\
\Delta_1 &= \breve{X}_1X_2 = e\big(g,g_{14}\big)^{\beta\cdot trk} = e(g,g_{14})^{(\beta\cdot\breve{\tau})/t_r} \\
X_3 &= e(g,g)^{(\alpha\beta)/\gamma}\cdot e(Y_c,g)^{\widehat{r}\beta}\cdot e(g_{14},g)^{(\beta\breve{\tau})/\gamma},\ \text{where } Y_c = g_4^{\mathcal{CSK}} \\
X_4 &= e(g,g_4)^{\widehat{r}\beta\partial} \\
\Delta_2 &= X_3\cdot(X_4)^{-\mathcal{CSK}/\partial} = e(g,g)^{(\alpha\beta)/\gamma}\cdot e(g_{14},g)^{(\beta\breve{\tau})/\gamma} \\
\Delta_3 &= e(g,g)^{\frac{\alpha}{\theta}} = g_T^{1/\theta}
\end{aligned}
$$

Remark 6. *The cloud can avoid the computation of Equation (3.6) as follows. On receiving the ciphertext $\mathcal{CT}$, the cloud computes Δ_0 (given in Figure 3.2) and verifies the identity $\Gamma \stackrel{?}{=} H_7(\Delta_0^{1/\omega})\oplus H_8\big(e(\sigma',\sigma'')^{\mathcal{CSK}}\big)$. If it is true, the cloud can store the ciphertext as $[\mathcal{CT},\Delta_0^{1/\omega}]$. Before sending to DU, the cloud verifies the integrity of the term $\Delta_0^{1/\omega}$ using the same identity $\Gamma \stackrel{?}{=} H_7(\Delta_0^{1/\omega})\oplus H_8\big(e(\sigma',\sigma'')^{\mathcal{CSK}}\big)$ and sets $\Delta_3 := \Delta_0^{1/\omega}$. In this case, the transform algorithm requires only $2\ell_e+7$ pairing evaluations instead of $2\ell_e+\ell_s+8$.*

7. **Unsigncryption and Verify**

The DU runs the following algorithm to recover the original plaintext.

◇ $\mathsf{Unsigncrypt\text{-}Verify}(\mathcal{PP}, \mathcal{CT}_{out}, \mathcal{TDK})$. On input $\mathcal{PP}$, $\mathcal{CT}_{out}$ and the secret transformation decryption key $\mathcal{TDK} := \langle tdk_1, tdk_2 \rangle$, this algorithm outputs the data file msg or $\perp$.

- Compute $\Lambda := (\Delta_1)^{-tdk_1} \cdot (\Delta_2)^{tdk_2} \cdot \Delta_3$.
- Check whether $H_3\big(H_2(\Lambda)||ct\big) \stackrel{?}{=} \mathsf{tag2}$.
  If this condition is not met, the error value $\perp$ is returned to confirm that the cloud deceptively returns a false search result.
- Else, Output $msg = ct \oplus \mathsf{KDF}(\Lambda)$.

Correctness. Note that $tdk_1 := t_r, tdk_2 := \gamma$. If the cloud honestly returns a valid transformed ciphertext $\mathcal{CT}_{out}$, then

$$\begin{aligned}
\Lambda &= \big(e(g, g_{14})^{\beta\check{r}/t_r}\big)^{-t_r} \big(e(g,g)^{(\alpha\beta)/\gamma} \cdot e(g_{14}, g)^{(\beta\check{r})/\gamma}\big)^{\gamma} g_T^{1/\theta} \\
&= g_T^{\beta+\frac{1}{\theta}} \\
H_3\big(H_2(\Lambda)||ct\big) &= H_3\big(\mathsf{tag1}||ct\big) = \mathsf{tag2} \\
ct \oplus \mathsf{KDF}(\Lambda) &= msg \oplus \mathsf{KDF}\big(g_T^{\beta+\frac{1}{\theta}}\big) \oplus \mathsf{KDF}\big(g_T^{\beta+\frac{1}{\theta}}\big) = msg
\end{aligned}$$

Remark 7. *As shown in Figure 3.2, a DU can retrieve the required data from the cloud.*

3.4 Security Proof of ABDSRS

In this section, we provide the security analysis of our ABDSRS under the assumption that KDF is secure.

Lemma 1. *Suppose the number of rows and the number of columns in the challenge encryption policy are at most q. Then, ABDSRS demonstrates IND-CCA2 security against a Type-1 adversary in the random oracle model, assuming the hardness of the q-1 problem (presented in Section 2.3.3).*

Proof. Let a PPT Type-1 adversary $\mathscr{A}$ breaks IND-CCA2 security (modeled as a game $\mathsf{Game}^{\mathsf{IND\text{-}CCA2}}_{\mathsf{Type\text{-}1}}$ in Section 3.2.2) of our ABDSRS with non-negligible advantage, then a challenger $\mathscr{C}$ can solve q-1 problem by communicating with $\mathscr{A}$ as in $\mathsf{Game}^{\mathsf{IND\text{-}CCA2}}_{\mathsf{Type\text{-}1}}$. $\mathscr{C}$ is given the q-1 problem instance $\big\langle \Sigma, g, g^{\beta}, \{g^{\phi^i}, g^{\psi_j}, g^{\beta\psi_j}, g^{\phi^i\psi_j}, g^{\phi^i/\psi_j^2}\}_{(i,j)\in[q,q]}, \{g^{\phi^i/\psi_j}\}_{(i,j)\in[2q,q],i\neq q+1}, \{g^{\phi^i\psi_j/\psi_{j'}^2}\}_{(i,j,j')\in[2q,q,q],j\neq j'},$

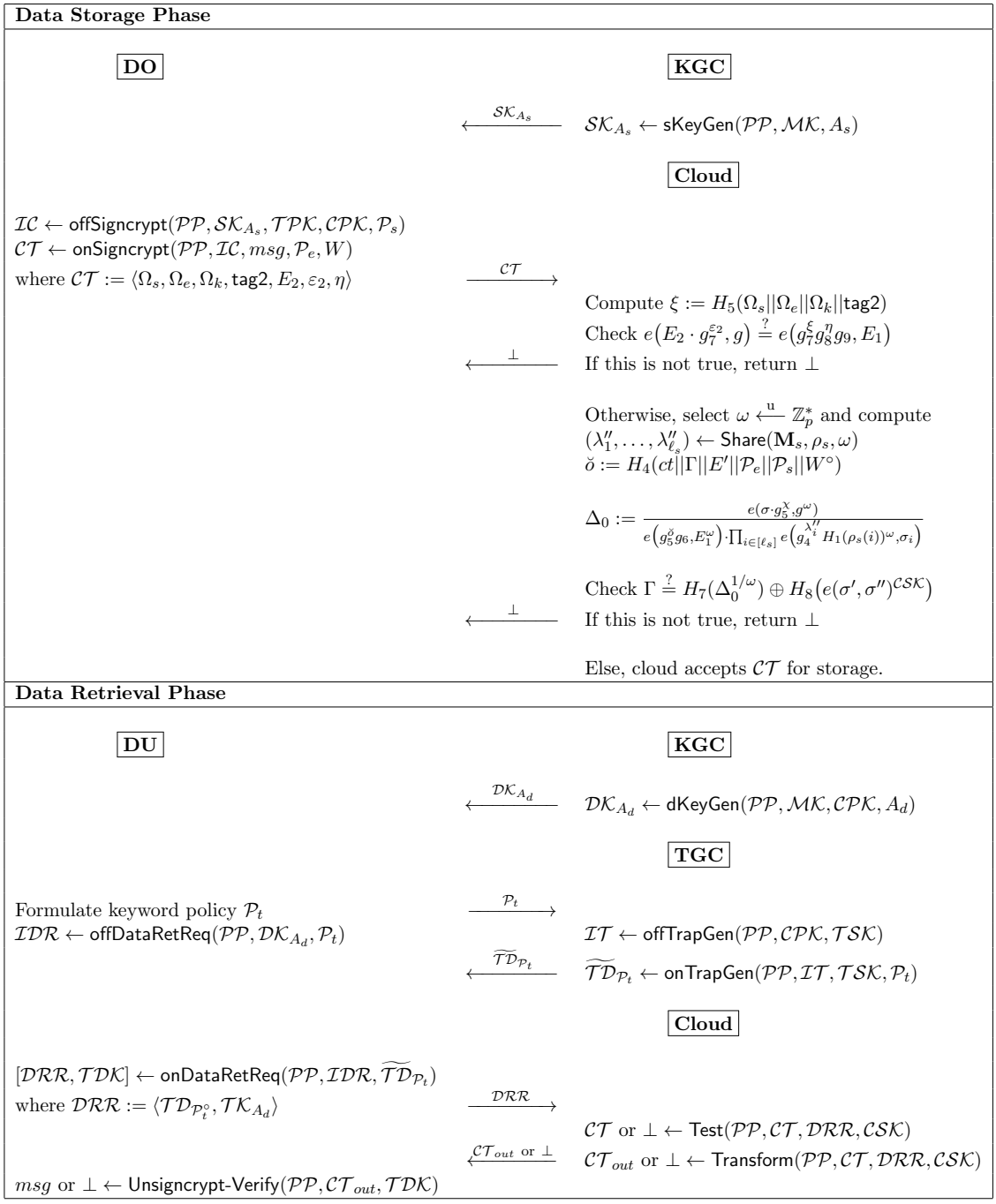


Figure 3.2: Data storage and data retrieval phases.

$\big\{g^{\beta\phi^i\psi_j/\psi_{j'}}, g^{\beta\phi^i\psi_j/\psi_{j'}^2}\big\}_{(i,j,j') \in [q,q,q], j \neq j'}, Z\big\rangle$. In order to ascertain if Z is equal to $e(g,g)^{\beta\phi^{q+1}}$ or Z has been randomly selected from $\mathbb{G}_T$, $\mathscr{C}$ interacts with $\mathscr{A}$ as described below.

(1) $\mathscr{A}$ sends the challenge encryption policy $\mathcal{P}_e^{\star} := (\mathbf{M}_e^{\star}, \rho_e^{\star})$ to $\mathscr{C}$, where $\mathbf{M}_e^{\star}$ is an $\ell_e^{\star} \times n_e^{\star}$ matrix with $\ell_e^{\star}, n_e^{\star} \leq q$. Let $\vec{M}_e^{\star(i)} := (M_{e1}^{\star(i)}, M_{e2}^{\star(i)}, \ldots, M_{en_e^{\star}}^{\star(i)})$ be the ith

row of $\mathbf{M}_e^\star$.

(2) $\mathscr{C}$ samples $\alpha' \xleftarrow{\text{u}} \mathbb{Z}_p^*$ and sets $g_T := e(g,g)^{\alpha'} \cdot e(g^\phi, g^{\phi^q})$ (i.e., the system master secret key $\mathcal{MK}$ is defined implicitly as g^α where $\alpha := \alpha' + \phi^{q+1}$). Next, $\mathscr{C}$ picks $z, z_1, z_2, \ldots, z_{15} \xleftarrow{\text{u}} \mathbb{Z}_p^*$ and defines

$$
\begin{array}{lll}
g := g & g_5 := g^{z_5} & g_{10} := g^{z_{10}} \\
g_1 := g^{z_1} \cdot \prod_{(j,\iota)\in[\ell_e^\star, n_e^\star]} \left(g^{\phi^\iota/\psi_j^2}\right)^{-\rho_e^\star(j) M_{e\iota}^{\star(j)}} & g_6 := g^{z_6} & g_{11} := g^{z_{11}} \\
g_2 := g^{z_2} \cdot \prod_{(j,\iota)\in[\ell_e^\star, n_e^\star]} \left(g^{\phi^\iota/\psi_j^2}\right)^{M_{e\iota}^{\star(j)}} & g_7 := g^\phi \cdot g^{z_7} & g_{12} := g^{z_{12}} \\
g_3 := g^{z_3} \cdot \prod_{(j,\iota)\in[\ell_e^\star, n_e^\star]} \left(g^{\phi^\iota/\psi_j}\right)^{M_{e\iota}^{\star(j)}} & g_8 := (g^\phi)^z \cdot g^{z_8} & g_{13} := g^{z_{13}} \\
g_4 := g^\phi & g_9 := (g^\phi)^{z_4} \cdot g^{z_9} & g_{14} := g^{z_{14}} \\
 & & g_{15} := g^{z_{15}}
\end{array}
$$

$\mathscr{C}$ chooses eight anti-collision hash functions $H_1 : \{0,1\}^* \to \mathbb{G}$, $H_2 : \mathbb{G}_T \to \{0,1\}^{\ell_{H_2}}$, $H_3 : \{0,1\}^* \to \{0,1\}^{\ell_{H_3}}$, $H_4 : \{0,1\}^* \to \mathbb{Z}_p^*$, $H_5 : \{0,1\}^* \to \mathbb{Z}_p^*$, $H_6 : \{0,1\}^* \to \mathbb{G}$, $H_7 : \mathbb{G}_T \to \{0,1\}^{\ell_H}$, $H_8 : \mathbb{G}_T \to \{0,1\}^{\ell_H}$, in which $\mathscr{C}$ simulates H_1 as follows.

To answer H_1 hash queries, $\mathscr{C}$ maintains a table Tab_{H_1}. If one submits a signing attribute y, $\mathscr{C}$ answers in the following way. If the tuple $[\![y, H_1(y) := g^{\upsilon_y}]\!]$ exists in Tab_{H_1}, returns g^{υ_y}. Otherwise, $\mathscr{C}$ picks $\upsilon_y \xleftarrow{\text{u}} \mathbb{Z}_p^*$, returns g^{υ_y} and inserts the new tuple $[\![y, H_1(y) := g^{\upsilon_y}]\!]$ into Tab_{H_1}.

$\mathscr{C}$ sets $\mathcal{PP} := \langle \Sigma, g_T, g, \{g_i\}_{i=1}^{15}, \mathcal{M}, \mathsf{KDF}, \{H_i\}_{i=1}^{8} \rangle$, where the message space is $\mathcal{M} := \{0,1\}^{\ell_{msg}}$ and a key derivation function is denoted as $\mathsf{KDF} : \mathbb{G}_T \to \{0,1\}^{\ell_{msg}}$. Lastly, $\mathscr{C}$ selects $\tau, \varpi_1, \varpi_2, \varpi_3, \varpi_4, r_c \xleftarrow{\text{u}} \mathbb{Z}_p^*$, computes $g_{16} := g^{\varpi_1}, g_{17} := g^{\varpi_2}, g_{18} := g^{\varpi_3}, g_{19} := g^{\varpi_4}, h_T := e(g, g_{15})^\tau, Y_c := g_4^{r_c}$ and sets $\mathcal{TPK} := \langle h_T, g_{16}, g_{17}, g_{18}, g_{19} \rangle, \mathcal{TSK} := \langle \tau, \varpi_1, \varpi_2, \varpi_3, \varpi_4 \rangle$, $\mathcal{CPK} := Y_c, \mathcal{CSK} := r_c$. $\mathscr{A}$ obtains the tuple $[\mathcal{PP}, \mathcal{TPK}, \mathcal{CPK}, \mathcal{CSK}]$.

(3) $\mathscr{A}$ queries signing key generation oracle $\mathcal{O}_{\mathcal{SKG}}(A_s)$, data retrieval request generation oracle $\mathcal{O}_{\mathcal{DRR}}(A_d, \mathcal{P}_t)$, signcryption oracle $\mathcal{O}_{\mathcal{SC}}(msg, \mathcal{P}_s, \mathcal{P}_e, W)$ and data retrieval cum unsigncrypt-verify oracle $\mathcal{O}_{\mathcal{DR}\text{-}\mathcal{UV}}(\mathcal{CT}, A_d, \mathcal{P}_t)$, with the respective inputs. Then $\mathscr{C}$ answers these queries as described below.

- $\mathcal{O}_{\mathcal{SKG}}(A_s)$: $\mathscr{C}$ chooses $\tilde{r} \xleftarrow{\text{u}} \mathbb{Z}_p^*$, implicitly defines $r' := \tilde{r} - \phi^q$ and returns the signing key $\mathcal{SK}_{A_s} := \left\langle A_s, S_1 := g^{\alpha'} g_4^{\tilde{r}}, S_2 := g^{\tilde{r}} \left(g^{\phi^q}\right)^{-1}, \{S_{3,y} := g^{\tilde{r}\upsilon_y} \left(g^{\phi^q}\right)^{-\upsilon_y}\}_{y \in A_s} \right\rangle$ to $\mathscr{A}$.
- $\mathcal{O}_{\mathcal{DRR}}(A_d, \mathcal{P}_t)$: $\mathscr{C}$'s response is one of the following two types.

(i) If $\mathcal{P}_e^\star(A_d) = true$, then $\mathscr{C}$ selects $f, \breve{f}, f', \breve{r}_i, \breve{r}'_i, \widehat{r}, \widehat{r}_x, B_{1i}, B_{2i1}, B_{2i2}, B_{2i3}, B_{2i4}, d, d', d'' \xleftarrow{\text{u}} \mathbb{Z}_p^*$, and implicitly sets $\gamma := \alpha/d, \breve{\tau} := (\alpha d'')/d, t_r := (\alpha d'')/(dd')$. Next, $\mathscr{C}$ computes the data retrieval request

$$\mathcal{DRR} := \left(\begin{array}{l} \mathcal{TD}_{\mathcal{P}_t^\circ} := \left\langle \begin{array}{l} \mathcal{P}_t^\circ, T_0, T_1, T_2, \{\varrho_i, \vec{T}_{1i} := (T_{1i1}, T_{1i2})\}_{i\in[\ell_t]}, \\ \{\vec{T}_{2i} := (T_{2i1}, T_{2i2}, T_{2i3}, T_{2i4})\}_{i\in[\ell_t]}, \\ \{B_{1i}, \vec{B}_{2i} := (B_{2i1}, B_{2i2}, B_{2i3}, B_{2i4})\}_{i\in[\ell_t]} \end{array} \right\rangle, \\ \mathcal{TK}_{A_d} := \left\langle \; A_d, T'_1, T'_2, \{T'_{3,x}, T'_{4,x}\}_{x\in A_d} \; \right\rangle \end{array} \right)$$

of the keyword policy $\mathcal{P}_t := (\mathbf{M}_t, \rho_t^\circ, \{w_{\rho_t^\circ(i)}\}_{i\in[\ell_t]})$ and the decryption attribute set A_d as follows: Compute $T'_1 := g^d g_4^{r_c \cdot \widehat{r}} g_{14}^{d''}$ and ϑ_i (resp. $\breve{\vartheta}_i$) is the ith share of $\tau \cdot H_5(e(T_1, T_2)^{r_c})$ (resp. $trk := \breve{\tau}/t_r = d'$) with respect to the policy $(\mathbf{M}_t, \rho_t^\circ)$, and calculate the other components of $\mathcal{DRR}$ as in Equation (3.2). Note that, in this case, $\mathscr{C}$ does not know $\mathcal{TDK} := \langle t_r, \gamma \rangle$. However, since $\mathscr{C}$ knows TGC's secret key $\langle \tau, \varpi_1, \varpi_2, \varpi_3, \varpi_4 \rangle$, the above $\mathcal{DRR}$ is properly distributed according to Remark 5.

(ii) Suppose $\mathcal{P}_e^\star(A_d) = false$. Then, $\mathscr{C}$ computes $\vec{\delta} := (\delta_1, \delta_2, \ldots, \delta_{n_e^\star}) \in \mathbb{Z}_p^{n_e^\star}$ such that $\delta_1 = -1$ and $\vec{\delta} \cdot \vec{M}_e^{\star(i)} = 0$, $\forall i \in \{i | \rho_e^\star(i) \in A_d\}$. Now, $\mathscr{C}$ picks $\breve{r} \xleftarrow{\text{u}} \mathbb{Z}_p^*$, implicitly defines $r := r_c^{-1}(\breve{r} + \sum_{i\in[n_e^\star]} \delta_i \cdot \phi^{q+1-i})$ and calculates the decryption key $\mathcal{DK}_{A_d} := \langle A_d, D_1, D_2, \{D_{3,x}, D_{4,x}\}_{x\in A_d} \rangle$ as given below.

$$D_1 := g^{\alpha'} g_4^{\breve{r}} \prod_{i=2}^{n_e^\star} \left(g^{\phi^{q+2-i}}\right)^{\delta_i}, \quad D_2 := g^{\breve{r} r_c^{-1}} \prod_{i\in[n_e^\star]} \left(g^{\phi^{q+1-i}}\right)^{\delta_i r_c^{-1}}.$$

For each attribute $x \in A_d$, $\mathscr{C}$ chooses $\breve{r}_x \xleftarrow{\text{u}} \mathbb{Z}_p^*$ and implicitly defines $r_x := \breve{r}_x + r \sum_{\substack{i'\in[\ell_e^\star] \\ \rho_e^\star(i')\notin A_d}} \dfrac{\psi_{i'}}{x - \rho_e^\star(i')}$, and computes

$$\begin{aligned} D_{3,x} \;:=\; & g^{\breve{r}_x} \prod_{\substack{i'\in[\ell_e^\star] \\ \rho_e^\star(i')\notin A_d}} \left(g^{\psi_{i'}}\right)^{(\breve{r} r_c^{-1})/(x-\rho_e^\star(i'))} \\ & \times \prod_{\substack{(i,i')\in[n_e^\star,\ell_e^\star] \\ \rho_e^\star(i')\notin A_d}} \left(g^{\psi_{i'}\phi^{q+1-i}}\right)^{(\delta_i r_c^{-1})/(x-\rho_e^\star(i'))}, \end{aligned}$$

$$
\begin{aligned}
D_{4,x} \;:=\; & (g_2^x g_1)^{\check{r}_x} \cdot (D_{3,x}/g^{\check{r}_x})^{xz_2+z_1} \\
& \times \prod_{\substack{(i',j,\iota)\in[\ell_e^\star,\ell_e^\star,n_e^\star] \\ \rho_e^\star(i')\notin A_d}} \Big(g^{\phi^\iota \psi_{i'}/\psi_j^2}\Big)^{\check{r}r_c^{-1}(x-\rho_e^\star(j))M_{e\iota}^{\star(j)}/(x-\rho_e^\star(i'))} \\
& \cdot \prod_{\substack{(i,i',j,\iota)\in[n_e^\star,\ell_e^\star,\ell_e^\star,n_e^\star] \\ \rho_e^\star(i')\notin A_d,(j\neq i'\vee i\neq\iota)}} \Big(g^{\psi_{i'}\phi^{q+1-i+\iota}/\psi_j^2}\Big)^{r_c^{-1}\delta_i(x-\rho_e^\star(j))M_{e\iota}^{\star(j)}/(x-\rho_e^\star(i'))} \\
& \times g_3^{-\check{r}r_c^{-1}} \prod_{i\in[n_e^\star]} \Big(g^{\phi^{q+1-i}}\Big)^{-z_3\delta_i r_c^{-1}} \prod_{\substack{(i,j,\iota)\in[n_e^\star,\ell_e^\star,n_e^\star] \\ i\neq\iota}} \Big(g^{\phi^{q+1-i+\iota}/\psi_j}\Big)^{-r_c^{-1}\delta_i M_{e\iota}^{\star(j)}}
\end{aligned}
$$

Since $\mathscr{C}$ knows TGC's secret key $\mathcal{TSK} := \langle \tau, \varpi_1, \varpi_2, \varpi_3, \varpi_4 \rangle$, it computes the trapdoor as $\widetilde{\mathcal{TD}}_{\mathcal{P}_t} \leftarrow \mathsf{onTrapGen}(\mathcal{PP}, \mathcal{IT}, \mathcal{TSK}, \mathcal{P}_t)$, where $\mathcal{IT} \leftarrow \mathsf{offTrapGen}(\mathcal{PP}, \mathcal{CPK}, \mathcal{TSK})$. Finally, $\mathscr{C}$ sends $[\mathcal{DK}_{A_d}, \widetilde{\mathcal{TD}}_{\mathcal{P}_t}]$ to $\mathscr{A}$. In this case, both $\mathscr{A}$ and $\mathscr{C}$ can generate $\mathcal{DRR} := \langle \mathcal{TD}_{\mathcal{P}_t^\circ}, \mathcal{TK}_{A_d} \rangle$ and $\mathcal{TDK}$ by using $[\mathcal{DK}_{A_d}, \widetilde{\mathcal{TD}}_{\mathcal{P}_t}]$.

- $\mathcal{O}_{\mathcal{SC}}(msg, \mathcal{P}_s, \mathcal{P}_e, W)$: $\mathscr{C}$ selects a set A_s satisfying $\mathcal{P}_s(A_s) = true$, generates $\mathcal{SK}_{A_s} \leftarrow \mathcal{O}_{\mathcal{SKG}}(A_s)$ and outputs the ciphertext
  $\mathcal{CT} \leftarrow \mathsf{onSigncrypt}(\mathcal{PP}, \mathcal{IC}, msg, \mathcal{P}_e, W)$, where
  $\mathcal{IC} \leftarrow \mathsf{offSigncrypt}(\mathcal{PP}, \mathcal{SK}_{A_s}, \mathcal{TPK}, \mathcal{CPK}, \mathcal{P}_s)$.
- $\mathcal{O}_{\mathcal{DR}\text{-}\mathcal{UV}}(\mathcal{CT}, A_d, \mathcal{P}_t)$: Firstly, $\mathscr{C}$ calculates $\mathcal{DRR} \leftarrow \mathcal{O}_{\mathcal{DRR}}(A_d, \mathcal{P}_t)$. If $\mathsf{Test}(\mathcal{PP}, \mathcal{CT}, \mathcal{DRR}, \mathcal{CSK}) \rightarrow \perp$ or $\mathcal{P}_e(A_d) = false$, it returns $\perp$. Otherwise, $\mathscr{C}$'s response can be one of two types given subsequently. The ciphertext is parsed as $\mathcal{CT} := \langle \Omega_s, \Omega_e, \Omega_k, \mathsf{tag2}, E_2, \varepsilon_2, \eta \rangle$, where
  $\Omega_e := \langle \mathcal{P}_e, ct, E', E_1, \{\vec{E}_i, \vec{\varepsilon}_i\}_{i\in[\ell_e]} \rangle$, $\Omega_s := \langle \mathcal{P}_s, \sigma', \sigma'', \Gamma, \sigma, \{\sigma_i\}_{i\in[\ell_s]}, \chi \rangle$ and $\Omega_k := \langle W^\circ, k_T, \vec{L}, \{\vec{K}_i, \vec{k}_i, u_i\}_{i\in[\varsigma]} \rangle$.
  (i) In case $\mathcal{P}_e^\star(A_d) = true$, $\mathscr{C}$ does not have the knowledge of the secret transformation decryption key $\mathcal{TDK}$ according to the simulation of $\mathcal{O}_{\mathcal{DRR}}(A_d, \mathcal{P}_t)$ and hence it proceeds in the following way. If $\xi + z\eta + z_4 = 0$ (this happens with prob. at most $1/p$), where $\xi := H_5(\Omega_s||\Omega_e||\Omega_k||\mathsf{tag2})$, then $\mathscr{C}$ aborts. Else it calculates
  $\mathcal{Y}_1 := e(E_1, g^{\alpha'}) \cdot e\big(E_2 \cdot g_7^{\varepsilon_2}/E_1^{\xi z_7+\eta z_8+z_9}, (g^{\phi^q})^{(\xi+z\eta+z_4)^{-1}}\big) = g_T^\beta$,
  $\Delta_3 = g_T^{1/\theta}$ (using Equation (3.6)) and $\Lambda := \mathcal{Y}_1\Delta_3 = g_T^{\beta+\frac{1}{\theta}}$.
  Next, $\mathscr{C}$ checks whether $H_3\big(H_2(\Lambda)||ct\big) \stackrel{?}{=} \mathsf{tag2}$. If this is not true, it returns $\perp$. Otherwise, it returns $msg = ct \oplus \mathsf{KDF}(\Lambda)$ to $\mathscr{A}$.
  (ii) In case $\mathcal{P}_e^\star(A_d) = false$, $\mathscr{C}$ knows $\mathcal{TDK}$. Hence, it computes the transformed ciphertext $\mathcal{CT}_{out} \leftarrow \mathsf{Transform}(\mathcal{PP}, \mathcal{CT}, \mathcal{DRR}, \mathcal{CSK})$ and

returns to $\mathscr{A}$ the output of $\mathsf{Unsigncrypt\text{-}Verify}(\mathcal{PP}, \mathcal{CT}_{out}, \mathcal{TDK})$.

Once this query phase is done, $\mathscr{A}$ outputs a signing policy $\mathcal{P}_s^\star$, two messages $msg_0^\star, msg_1^\star \in \mathcal{M}$, and a set $W^\star$ of keywords.

(4) Let $\mathcal{P}_s^\star := (\mathbf{M}_s^\star, \rho_s^\star)$, where $\mathbf{M}_s^\star$ represents a matrix of dimension $\ell_s^\star \times n_s^\star$. $\mathscr{C}$ picks $i \xleftarrow{\text{u}} \{0,1\}$, picks a set A_s satisfying $\mathcal{P}_s^\star(A_s) = true$, calculates $\vec{a} := (a_1, a_2, \ldots, a_{\ell_s^\star}) \leftarrow \mathsf{Reconstruct}(\mathbf{M}_s^\star, \rho_s^\star, A_s)$ satisfying $\sum_{i\in[\ell_s^\star]} a_i \cdot \vec{M}_s^{\star(i)} = \vec{1}_{n_s^\star}$ and $a_i = 0$ for all $i \in \{i | \rho_s^\star(i) \notin A_s\}$, samples $(b_1, b_2, \ldots, b_{\ell_s^\star}) \xleftarrow{\text{u}} \big\{(b_1, b_2, \ldots, b_{\ell_s^\star}) \in \mathbb{Z}_p^{\ell_s^\star} | \sum_{i\in[\ell_s^\star]} b_i \cdot \vec{M}_s^{\star(i)} = \vec{0}_{n_s^\star}\big\}$. To compute the challenge ciphertext

$$\mathcal{CT}^\star := \left(\begin{array}{l} \Omega_s^\star := \langle \mathcal{P}_s^\star, \sigma'^\star, \sigma''^\star, \Gamma^\star, \sigma^\star, \{\sigma_i^\star\}_{i\in[\ell_s^\star]}, \chi^\star \rangle, \\ \Omega_e^\star := \langle \mathcal{P}_e^\star, ct^\star, E'^\star, E_1^\star, \{\vec{E}_i^\star := (E_{i1}^\star, E_{i2}^\star, E_{i3}^\star), \vec{\varepsilon}_i^\star := (\varepsilon_{i1}^\star, \varepsilon_{i2}^\star)\}_{i\in[\ell_e^\star]} \rangle, \\ \Omega_k^\star := \left\langle \begin{array}{l} W^{\star\circ}, k_T^\star, \vec{L}^\star := (L_1^\star, L_2^\star, L_3^\star, L_4^\star), \\ \{\vec{K}_j^\star := (K_{j1}^\star, K_{j2}^\star), \vec{k}_j^\star := (k_{j1}^\star, k_{j2}^\star, k_{j3}^\star, k_{j4}^\star), u_j^\star\}_{j\in[\varsigma]} \end{array} \right\rangle, \\ \mathsf{tag2}^\star, E_2^\star, \varepsilon_2^\star, \eta^\star \end{array} \right)$$

of the message $msg_i^\star$ for the signing policy $\mathcal{P}_s^\star := (\mathbf{M}_s^\star, \rho_s^\star)$, encryption policy $\mathcal{P}_e^\star := (\mathbf{M}_e^\star, \rho_e^\star)$ and keyword set $W^\star := \{[\mathcal{W}_1 : w_1], \ldots, [\mathcal{W}_\varsigma : w_\varsigma]\}$, $\mathscr{C}$ chooses $\theta, \delta', \delta'', \delta''', \breve{r}_1, o_2, f_1, f_2, f_3, f_4, \breve{t}_j, \pi_{j1}, \pi_{j2}, \chi^\star, \varepsilon_2^\star, \varepsilon_{i1}^\star, \varepsilon_{i2}^\star, u_j^\star \xleftarrow{\text{u}} \mathbb{Z}_p^*$, implicitly sets $\breve{r} := \breve{r}_1 - \phi^q, t_i := -\beta\partial\psi_i$,

$$\lambda_i := \vec{M}_e^{\star(i)} \cdot (\beta\partial, \beta\partial\phi + \tilde{o}_2, \beta\partial\phi^2 + \tilde{o}_3, \ldots, \beta\partial\phi^{n_e^\star - 1} + \tilde{o}_{n_e^\star})$$

$$= \sum_{j\in[n_e^\star]} M_{ej}^{\star(i)} \beta\partial\phi^{j-1} + \sum_{j=2}^{n_e^\star} \tilde{o}_j M_{ej}^{\star(i)},$$

where $\partial := H_5\big(e(g, g_4)^{\delta'\delta'''r_c}\big)$, and computes the components of $\mathcal{CT}^\star$ in the following way.

$E_1^\star := g^\beta, \Gamma^\star := H_7(g_T^{1/\theta}) \oplus H_8\big(e(g, g_4)^{\delta'\cdot\delta''\cdot r_c}\big)$,

$ct^\star := msg_i^\star \oplus \mathsf{KDF}\big(Z \cdot e(g^\beta, g)^{\alpha'} \cdot g_T^{1/\theta}\big)$, $\mathsf{tag2}^\star := H_3\big(H_2(Z \cdot e(g^\beta, g)^{\alpha'} \cdot g_T^{1/\theta}) || ct^\star\big)$,

$E'^\star := g_4^{\delta'''}, \sigma'^\star := g^{\delta'}, \sigma''^\star := g_4^{\delta''}$,

$\sigma^\star := g^{\frac{\alpha'}{\theta}} g_4^{\frac{\breve{r}_1}{\theta}} (g^\beta)^{z_5\breve{o}_1 + z_6} g_5^{-\chi^\star} \prod_{i\in[\ell_s^\star]} \big(g^{\upsilon_{\rho_s^\star(i)}(\frac{\breve{r}_1 a_i}{\theta} + o_2 b_i)} (g^{\phi^q})^{-\upsilon_{\rho_s^\star(i)}\cdot\frac{a_i}{\theta}}\big)$,

$\sigma_i^\star := g^{\frac{\breve{r}_1 a_i}{\theta} + o_2 b_i} (g^{\phi^q})^{-\frac{a_i}{\theta}}$,

$E_{i1}^\star := g_4^{\sum_{j=2}^{n_e^\star} \tilde{o}_j M_{ej}^{\star(i)}} \cdot (g^{\beta\psi_i})^{-z_3\partial} \cdot g_4^{-\varepsilon_{i1}^\star} \cdot \prod_{\substack{(j,\iota)\in[\ell_e^\star, n_e^\star] \\ j\neq i}} \big(g^{\beta\phi^\iota\psi_i/\psi_j}\big)^{-\partial M_{e\iota}^{\star(j)}}$,

$E_{i2}^\star := (g^{\beta\psi_i})^{-\partial(z_2\rho_e^\star(i) + z_1)} \cdot g_2^{-\varepsilon_{i2}^\star} \cdot \prod_{\substack{(j,\iota)\in[\ell_e^\star, n_e^\star] \\ j\neq i}} \big(g^{\beta\phi^\iota\psi_i/\psi_j^2}\big)^{(\rho_e^\star(j) - \rho_e^\star(i))\partial M_{e\iota}^{\star(j)}}$,

$E^{\star}_{i3} := (g^{\beta\psi_i})^{-\partial}$,
$L^{\star}_1 := g^{f_1}_{16}, L^{\star}_2 := g^{f_2}_{17}, L^{\star}_3 := g^{f_3}_{18}, L^{\star}_4 := g^{f_4}_{19}$,
$K^{\star}_{j1} := (g^{w_j}_{11} g_{12})^{\breve{t}_j} (g^{\beta})^{-z_{10}} \cdot g^{-u^{\star}_j}_{11}, K^{\star}_{j2} := (g^{w_j}_{11} g_{12})^{\breve{t}_j} (g^{\beta})^{-z_{13}} \cdot g^{-u^{\star}_j}_{11}$,
$k^{\star}_{j1} := \breve{t}_j - \pi_{j1} - f_1, k^{\star}_{j2} := \pi_{j1} - f_2, k^{\star}_{j3} := \breve{t}_j - \pi_{j2} - f_3, k^{\star}_{j4} := \pi_{j2} - f_4$,
$k^{\star}_T := e(g^{\beta}, g_{15})^{\tau}, E^{\star}_2 := (g^{\beta})^{z_7\xi + z_8\eta^{\star} + z_9} \cdot g^{-\varepsilon^{\star}_2}_7$
where $\breve{o}_1 := H_4(ct^{\star}||\Gamma^{\star}||E'^{\star}||\mathcal{P}^{\star}_e||\mathcal{P}^{\star}_s||W^{\circ\star}), \xi := H_5(\Omega^{\star}_s||\Omega^{\star}_e||\Omega^{\star}_k||\mathsf{tag2}^{\star})$,
$\eta^{\star} := (-\xi - z_4)/z$ and $W^{\circ\star} := \{\mathcal{W}_1, \mathcal{W}_2, \ldots, \mathcal{W}_{\varsigma}\}$. Lastly, $\mathscr{C}$ sends the challenge ciphertext $\mathcal{CT}^{\star}$ to $\mathscr{A}$.

(5) Now, $\mathscr{A}$ queries for the oracles like step (3), with the obvious restriction that it cannot query data retrieval cum unsigncrypt-verify oracle with the input $[\mathcal{CT}^{\star}, A_d, \mathcal{P}_t]$ satisfying $\mathcal{P}^{\star}_e(A_d) = true \wedge \mathcal{P}_t(W^{\star}) = true$. Once this query phase is over, $\mathscr{A}$ outputs a guess i' of i.

$\mathscr{A}$ wins the game if $i' = i$. Therefore, if $\mathscr{A}$ wins, $\mathscr{C}$ will claim that $Z = e(g,g)^{\beta\phi^{q+1}}$; otherwise, $\mathscr{C}$ claims that Z is a random element of $\mathbb{G}_T$.

When $Z = e(g,g)^{\beta\phi^{q+1}}$, according to Remark 4, the challenge ciphertext $\mathcal{CT}^{\star}$ is an accurately derived ciphertext as in original construction. Note that if Z has been randomly selected from $\mathbb{G}_T$, then $\mathcal{CT}^{\star}$ is independent of i in $\mathscr{A}$'s view. In this case, $\mathscr{A}$'s guess is random and its advantage is 0. Thus, if $\mathscr{A}$ has non-negligible advantage in winning the game, $\mathscr{C}$ can solve q-1 problem with non-negligible advantage. □

Lemma 2. *ABDSRS demonstrates IND-CCA2 security against PPT Type-2 adversary assuming the hardness of the DBDH problem (described in Section 2.3.1).*

Proof. Let a PPT Type-2 adversary $\mathscr{A}$ that breaks the IND-CCA2 security (modeled as a game $\mathsf{Game}^{\mathsf{IND\text{-}CCA2}}_{\mathsf{Type\text{-}2}}$ in Section 3.2.2) of our ABDSRS with non-negligible advantage, then a challenger $\mathscr{C}$ can solve DBDH problem by communicating with $\mathscr{A}$ as in $\mathsf{Game}^{\mathsf{IND\text{-}CCA2}}_{\mathsf{Type\text{-}2}}$. $\mathscr{C}$ is given the DBDH problem instance $\langle \Sigma, g, G_1, G_2, G_3, Z \rangle$, where $G_1 := g^{\phi_1}$,
$G_2 := g^{\phi_2}, G_3 := g^{\phi_3}$ (note that ϕ_1, ϕ_2, ϕ_3 are unknown to $\mathscr{C}$). In order to ascertain if Z is equal to $e(g,g)^{\beta\phi^{q+1}}$ or Z has been randomly selected from $\mathbb{G}_T$, $\mathscr{C}$ interacts with $\mathscr{A}$ as described below.

(1) $\mathscr{C}$ picks $\alpha, z_4 \xleftarrow{u} \mathbb{Z}^*_p$ and sets $g_T := e(g,g)^{\alpha}, g_4 := g^{z_4}$. Next, it chooses $g_1, g_2, g_3, g_5, \ldots, g_{15} \xleftarrow{u} \mathbb{G}$, and eight collision-resistant hash functions $\{H_i\}^8_{i=1}$ (as described in the construction). Now, $\mathscr{C}$ sets $\mathcal{PP} := \langle \Sigma, g_T, g, \{g_i\}^{15}_{i=1}, \mathcal{M}, \mathsf{KDF}, \{H_i\}^8_{i=1} \rangle$, where the message space is $\mathcal{M} := \{0,1\}^{\ell_{msg}}$ and a key derivation function is denoted as $\mathsf{KDF} : \mathbb{G}_T \rightarrow \{0,1\}^{\ell_{msg}}$, and $\mathcal{MK} := g^{\alpha}$. Lastly, $\mathscr{C}$ selects

$\tau, \varpi_1, \varpi_2, \varpi_3, \varpi_4 \xleftarrow{u} \mathbb{Z}_p^*$, computes $g_{16} := g^{\varpi_1}, g_{17} := g^{\varpi_2}, g_{18} := g^{\varpi_3}, g_{19} := g^{\varpi_4}, h_T := e(g, g_{15})^{\tau}, Y_c := G_1^{z_4}$ and sets $\mathcal{TPK} := \langle h_T, g_{16}, g_{17}, g_{18}, g_{19}\rangle, \mathcal{TSK} := \langle \tau, \varpi_1, \varpi_2, \varpi_3, \varpi_4\rangle$, $\mathcal{CPK} := Y_c$, and implicitly sets $\mathcal{CSK} := \phi_1$. $\mathscr{C}$ sends the tuple $[\mathcal{PP}, \mathcal{TPK}, \mathcal{CPK}]$ to $\mathscr{A}$.

(2) $\mathscr{A}$ queries signing key generation oracle $\mathcal{O}_{\mathcal{SKG}}(A_s)$, data retrieval request generation oracle $\mathcal{O}'_{\mathcal{DRR}}(A_d, \mathcal{P}_t)$, signcryption oracle $\mathcal{O}_{\mathcal{SC}}(msg, \mathcal{P}_s, \mathcal{P}_e, W)$ and data retrieval cum unsigncrypt-verify oracle $\mathcal{O}_{\mathcal{DR}\text{-}\mathcal{UV}}(\mathcal{CT}, A_d, \mathcal{P}_t)$. Since $\mathscr{C}$ knows the system master secret key $\mathcal{MK}$ and TGC secret key $\mathcal{TSK}$, it can answer the $\mathscr{A}$'s queries by running suitable algorithms of ABDSRS. Once this query phase is over, $\mathscr{A}$ sends to $\mathscr{C}$ two messages $msg_0^\star, msg_1^\star \in \mathcal{M}$, an encryption policy $\mathcal{P}_e^\star$, a signing policy $\mathcal{P}_s^\star$ and a keyword set $W^\star$.

(3) Let $\mathcal{P}_s^\star := (\mathbf{M}_s^\star, \rho_s^\star)$ and $\mathcal{P}_e^\star := (\mathbf{M}_e^\star, \rho_e^\star)$, where $\mathbf{M}_s^\star$ (resp. $\mathbf{M}_e^\star$) is an $\ell_s^\star \times n_s^\star$ (resp. $\ell_e^\star \times n_e^\star$) matrix. $\mathscr{C}$ samples $i \xleftarrow{u} \{0,1\}$, formulates a signing attribute set A_s such that $\mathcal{P}_s^\star(A_s) = true$, calculates $\vec{a} := (a_1, a_2, \ldots, a_{\ell_s^\star}) \leftarrow \mathsf{Reconstruct}(\mathbf{M}_s^\star, \rho_s^\star, A_s)$ satisfying $\sum_{i\in[\ell_s^\star]} a_i \cdot \vec{M}_s^{\star(i)} = \vec{1}_{n_s^\star}$ and $a_i = 0$ for all $i \in \{i | \rho_s^\star(i) \notin A_s\}$, picks $(b_1, b_2, \ldots, b_{\ell_s^\star}) \xleftarrow{u} \big\{(b_1, b_2, \ldots, b_{\ell_s^\star}) \in \mathbb{Z}_p^{\ell_s^\star} \,|\, \sum_{i\in[\ell_s^\star]} b_i \cdot \vec{M}_s^{\star(i)} = \vec{0}_{n_s^\star}\big\}$. To compute the challenge ciphertext

$$\mathcal{CT}^\star := \left(\begin{array}{l} \Omega_s^\star := \langle \mathcal{P}_s^\star, \sigma'^\star, \sigma''^\star, \Gamma^\star, \sigma^\star, \{\sigma_i^\star\}_{i\in[\ell_s^\star]}, \chi^\star\rangle, \\ \Omega_e^\star := \langle \mathcal{P}_e^\star, ct^\star, E'^\star, E_1^\star, \{\vec{E}_i^\star := (E_{i1}^\star, E_{i2}^\star, E_{i3}^\star), \vec{\varepsilon}_i^\star := (\varepsilon_{i1}^\star, \varepsilon_{i2}^\star)\}_{i\in[\ell_e^\star]}\rangle, \\ \Omega_k^\star := \left\langle \begin{array}{l} W^{\star\circ}, k_T^\star, \vec{L}^\star := (L_1^\star, L_2^\star, L_3^\star, L_4^\star), \\ \{\vec{K}_j^\star := (K_{j1}^\star, K_{j2}^\star), \vec{k}_j^\star := (k_{j1}^\star, k_{j2}^\star, k_{j3}^\star, k_{j4}^\star), u_j^\star\}_{j\in[\varsigma]} \end{array} \right\rangle, \\ \mathsf{tag2}^\star, E_2^\star, \varepsilon_2^\star, \eta^\star \end{array}\right)$$

of the message $msg_i^\star$ for the signing policy $\mathcal{P}_s^\star := (\mathbf{M}_s^\star, \rho_s^\star)$, encryption policy $\mathcal{P}_e^\star := (\mathbf{M}_e^\star, \rho_e^\star)$ and keyword set $W^\star := \{[\mathcal{W}_1 : w_1], \ldots, [\mathcal{W}_\varsigma : w_\varsigma]\}$, $\mathscr{C}$ chooses $\theta, \beta, \delta'', \breve{r}, o_2, \eta, t_i, f_1, f_2, f_3, f_4, \breve{t}_j, \pi_{j1}, \pi_{j2}, \chi^\star, \varepsilon_2^\star, \varepsilon_{i1}^\star, \varepsilon_{i2}^\star, u_j^\star, v_2, \ldots, v_{n_e^\star} \xleftarrow{u} \mathbb{Z}_p^*$ and computes the components of $\mathcal{CT}^\star$ as follows. Set $\sigma'^\star := G_2, E'^\star := G_3^{z_4}$, $\Gamma^\star := H_7(g_T^{1/\theta}) \oplus H_8\big(e(G_1, G_2)^{\delta'' z_4}\big)$, $E_{i1}^\star := g_4^{\vec{M}_e^{\star(i)}\cdot(\beta H_5(Z^{z_4}), v_2, \ldots, v_{n_e^\star})} g_3^{t_i} \cdot g_4^{-\varepsilon_{i1}^\star}$, and compute the other components of $\mathcal{CT}^\star$ as in Equation (3.1), where $\lambda_i := \vec{M}_e^{\star(i)} \cdot (\beta H_5(Z^{z_4}), v_2, \ldots, v_{n_e^\star})$ is the ith share of $\beta H_5(Z^{z_4})$ with respect to $(\mathbf{M}_e^\star, \rho_e^\star)$. Lastly, $\mathscr{C}$ sends $\mathcal{CT}^\star$ to $\mathscr{A}$.

(4) Now, $\mathscr{A}$ queries similar to step (2), except that $\mathscr{A}$ is not permitted to query data retrieval cum unsigncrypt-verify oracle with the input $[\mathcal{CT}^\star, A_d, \mathcal{P}_t]$ satisfying

$\mathcal{P}_e^\star(A_d) = true \wedge \mathcal{P}_t(W^\star) = true$. When this query phase is over, $\mathscr{A}$ announces a guess i' of i.

$\mathscr{A}$ wins the game if $i' = i$. Therefore, if $\mathscr{A}$ wins, $\mathscr{C}$ will claim that $Z = e(g,g)^{\phi_1\phi_2\phi_3}$; otherwise, $\mathscr{C}$ claims that Z is a random element of $\mathbb{G}_T$.

From Remark 4, if Z is equal to $e(g,g)^{\phi_1\phi_2\phi_3}$, the challenge ciphertext $\mathcal{CT}^\star$ is an accurately derived ciphertext similar to the ciphertext of original ABDSRS construction. If Z is randomly chosen from $\mathbb{G}_T$, then $\mathcal{CT}^\star$ is not dependent on i in $\mathscr{A}$'s view; resulting in $\mathscr{A}$'s guess is random and its advantage is 0. Thus, if $\mathscr{A}$ has a non-negligible advantage in winning the game, $\mathscr{C}$ can solve DBDH problem with non-negligible advantage. □

We derive the subsequent theorem by combining Lemma 1 with Lemma 2.

Theorem 1 (Data Confidentiality)**.** *Suppose the number of rows and the number of columns in the challenge encryption policy are at most q. Then, ABDSRS is IND-CCA2 secure in the random oracle model, under the assumption that q-1 and DBDH problems are hard.*

Theorem 2 (Data Unforgeability)**.** *ABDSRS demonstrates EUF-CMA security in the random oracle model assuming the hardness of the q-DHE problem (given in Section 2.3.2), if the challenge signing policy has a maximum of q columns. .*

Proof. Let a PPT adversary $\mathscr{A}$ breaks EUF-CMA security (modeled as a game $\mathsf{Game}_{\mathscr{A}}^{\mathsf{EUF\text{-}CMA}}$ in Section 3.2.2) of our ABDSRS with non-negligible advantage, then a challenger $\mathscr{C}$ can solve q-DHE problem by communicating with $\mathscr{A}$. Given the q-DHE problem instance $\langle \Sigma, g, \{g^{\phi^i}\}_{i\in[2q],i\neq q+1}\rangle$, the task of $\mathscr{C}$ is to compute $g^{\phi^{q+1}}$. We show below how this can be done.

(1) $\mathscr{A}$ sends the challenge signing policy $\mathcal{P}_s^\star := (\mathbf{M}_s^\star, \rho_s^\star)$ to $\mathscr{C}$, where $\mathbf{M}_s^\star$ is an $\ell_s^\star \times n_s^\star$ matrix with $n_s^\star \leq q$. Let $\vec{M}_s^{\star(i)} := (M_{s1}^{\star(i)}, M_{s2}^{\star(i)}, \ldots, M_{sn_s^\star}^{\star(i)})$ be the ith row of $\mathbf{M}_s^\star$.

(2) $\mathscr{C}$ chooses $\alpha', \theta \xleftarrow{u} \mathbb{Z}_p^*$ and sets $g_T := e(g,g)^{\theta\alpha'} e(g^\phi, g^{\phi^q})^\theta$ by implicitly defining $\alpha := \theta\alpha' + \theta\phi^{q+1}$. Next, $\mathscr{C}$ picks $z_1, z_2, z_3, z_5, \ldots, z_9 \xleftarrow{u} \mathbb{Z}_p^*$ and $g_{10}, \ldots, g_{15} \xleftarrow{u} \mathbb{G}$, and defines $g := g$, $g_4 := g^\phi, g_i := g^{z_i}$ for $i = 1, 2, 3, 5, \ldots, 9$. $\mathscr{C}$ selects eight collision-resistant hash functions $\{H_i\}_{i=1}^8$ (as mentioned in the construction), and simulates H_1 and H_4 as explained below.

H_1 Hash Queries: To answer H_1 hash queries, $\mathscr{C}$ maintains a table Tab_{H_1}. If one submits a signing attribute y, $\mathscr{C}$ answers in the following way. If the tuple

$[\![y, \upsilon_y, H_1(y)]\!]$ exists in Tab_{H_1}, $\mathscr{C}$ returns $H_1(y)$. Otherwise, $\mathscr{C}$ picks $\upsilon_y \xleftarrow{\text{u}} \mathbb{Z}_p^*$, returns

$$H_1(y) := \begin{cases} g^{\upsilon_y} \prod_{j \in [n_s^\star]} \left(g^{\phi^j}\right)^{-M_{sj}^{\star(i)}}, & \text{if } y \text{ is the attribute of some } i\text{th row of } \mathbf{M}_s^\star \\ g^{\upsilon_y}, & \text{otherwise,} \end{cases},$$

and inserts the new tuple $[\![y, \upsilon_y, H_1(y)]\!]$ into Tab_{H_1}.

H_4 Hash Queries: To answer H_4 hash queries, $\mathscr{C}$ maintains a table Tab_{H_4}. These queries are of two types. (i) Queries being submitted by $\mathscr{A}$. When $\mathscr{A}$ submits the input $(ct, \Gamma, E', \mathcal{P}_e, \mathcal{P}_s, W^\circ)$, $\mathscr{C}$ responds as follows. If the tuple $[\![(ct, \Gamma, E', \mathcal{P}_e, \mathcal{P}_s, W^\circ), \tilde{o}]\!]$ exists in Tab_{H_4}, $\mathscr{C}$ returns $\tilde{o}$ as
$H_4(ct||\Gamma||E'||\mathcal{P}_e||\mathcal{P}_s||W^\circ) := \tilde{o}$. Else, $\mathscr{C}$ selects $\tilde{o} \xleftarrow{\text{u}} \mathbb{Z}_p^*$, returns $\tilde{o}$ and inserts the new tuple $[\![(ct, \Gamma, E', \mathcal{P}_e, \mathcal{P}_s, W^\circ), \tilde{o}]\!]$ into Tab_{H_4}. (ii) Queries being conducted by $\mathscr{C}$ during signcryption oracle simulation (which will be discussed in signcryption oracle execution given below).

Next, $\mathscr{C}$ sets $\mathcal{PP} := \langle \Sigma, g_T, g, \{g_i\}_{i=1}^{15}, \mathcal{M}, \mathsf{KDF}, \{H_i\}_{i=1}^{8} \rangle$, where $\mathcal{M} := \{0,1\}^{\ell_{msg}}$ is the space of all messages, and KDF is the key derivation function. Lastly, $\mathscr{C}$ selects $\tau, \varpi_1, \varpi_2, \varpi_3, \varpi_4, r_c \xleftarrow{\text{u}} \mathbb{Z}_p^*$, computes $g_{16} := g^{\varpi_1}, g_{17} := g^{\varpi_2}, g_{18} := g^{\varpi_3}$, $g_{19} := g^{\varpi_4}, h_T := e(g, g_{15})^\tau, Y_c := g_4^{r_c}$ and sets $\mathcal{TPK} := \langle h_T, g_{16}, g_{17}, g_{18}, g_{19} \rangle$, $\mathcal{TSK} := \langle \tau, \varpi_1, \varpi_2, \varpi_3, \varpi_4 \rangle$, $\mathcal{CPK} := Y_c, \mathcal{CSK} := r_c$.
$\mathscr{C}$ sends $[\mathcal{PP}, \mathcal{TPK}, \mathcal{CPK}, \mathcal{CSK}]$ to $\mathscr{A}$.

(3) Now, $\mathscr{A}$ queries signing key generation oracle $\mathcal{O}'_{\mathcal{SKG}}(A_s)$, data retrieval request generation oracle $\mathcal{O}'_{\mathcal{DRR}}(A_d, \mathcal{P}_t)$, signcryption oracle $\mathcal{O}_{\mathcal{SC}}(msg, \mathcal{P}_s, \mathcal{P}_e, W)$ and data retrieval cum unsigncrypt-verify oracle $\mathcal{O}_{\mathcal{DR\text{-}UV}}(\mathcal{CT}, A_d, \mathcal{P}_t)$. Then $\mathscr{C}$ responds to these queries as explained below.

- $\mathcal{O}'_{\mathcal{SKG}}(A_s)$: $\mathscr{A}$ submits a signing attribute set A_s such that $\mathcal{P}_s^\star(A_s) = false$. Then, $\mathscr{C}$ calculates $\vec{\delta} := (\delta_1, \delta_2, \ldots, \delta_{n_s^\star}) \in \mathbb{Z}_p^{n_s^\star}$ such that $\delta_1 = -1$ and $\vec{\delta} \cdot \vec{M}_s^{\star(i)} = 0$, $\forall i \in \{i | \rho_s^\star(i) \in A_s\}$. Now, $\mathscr{C}$ picks $r'_0 \xleftarrow{\text{u}} \mathbb{Z}_p^*$, implicitly defines $r' := r'_0 + \sum_{\iota \in [n_s^\star]} \theta \delta_\iota \phi^{q-\iota+1}$ and returns the signing key $\mathcal{SK}_{A_s} := \langle A_s, S_1, S_2, \{S_{3,y}\}_{y \in A_s} \rangle$ to $\mathscr{A}$, where

$$S_1 := g^{\theta\alpha'} g_4^{r'_0} \prod_{\iota=2}^{n_s^\star} \left(g^{\phi^{q-\iota+2}}\right)^{\theta\delta_\iota}, \; S_2 := g^{r'_0} \prod_{\iota \in [n_s^\star]} \left(g^{\phi^{q-\iota+1}}\right)^{\theta\delta_\iota},$$

$$S_{3,y} := \begin{cases} S_2^{\upsilon_y} \prod_{j\in[n_s^\star]} \left(g^{\phi^j}\right)^{-r_0' M_{sj}^{\star(i)}} \prod_{(\iota,j)\in[n_s^\star,n_s^\star],\iota\neq j} \left(g^{\phi^{q-\iota+1+j}}\right)^{-\theta\delta_\iota M_{sj}^{\star(i)}}, \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{if } \rho_s^\star(i) = y \\ S_2^{\upsilon_y}, \text{otherwise.} \end{cases}$$

- $\mathcal{O}'_{\mathcal{DRR}}(A_d, \mathcal{P}_t) : \mathscr{A}$ submits a decryption attribute set A_d and a keyword policy $\mathcal{P}_t$. $\mathscr{C}$ picks $r_0, r_x \xleftarrow{\text{u}} \mathbb{Z}_p^*$, implicitly sets $r := r_0 - \theta r_c^{-1}\phi^q$ and calculates the decryption key $\mathcal{DK}_{A_d} := \langle A_d, D_1, D_2, \{D_{3,x}, D_{4,x}\}_{x\in A_d}\rangle$, where

$$D_1 := g^{\theta\alpha'} g_4^{r_0 r_c}, D_2 := g^{r_0}\left(g^{\phi^q}\right)^{-\theta r_c^{-1}}, D_{3,x} := g^{r_x}, D_{4,x} := (g_2^x g_1)^{r_x} D_2^{-z_3}.$$

  Since $\mathscr{C}$ knows TGC's secret key $\mathcal{TSK} := \langle \tau, \varpi_1, \varpi_2, \varpi_3, \varpi_4\rangle$, it computes the trapdoor as $\widetilde{\mathcal{TD}}_{\mathcal{P}_t} \leftarrow \mathsf{onTrapGen}(\mathcal{PP}, \mathcal{IT}, \mathcal{TSK}, \mathcal{P}_t)$, where $\mathcal{IT} \leftarrow \mathsf{offTrapGen}(\mathcal{PP}, \mathcal{CPK}, \mathcal{TSK})$. Lastly, $\mathscr{C}$ sends $[\mathcal{DK}_{A_d}, \widetilde{\mathcal{TD}}_{\mathcal{P}_t}]$ to $\mathscr{A}$.

- $\mathcal{O}_{\mathcal{SC}}(msg, \mathcal{P}_s, \mathcal{P}_e, W)$: Let $W := \{[\mathcal{W}_1 : w_1], \ldots, [\mathcal{W}_\varsigma : w_\varsigma]\}$, $\mathcal{P}_s := (\mathbf{M}_s, \rho_s)$, and $\mathcal{P}_e := (\mathbf{M}_e, \rho_e)$, where $\mathbf{M}_s$ (resp. $\mathbf{M}_e$) is a matrix of size $\ell_s \times n_s$ (resp. $\ell_e \times n_e$). $\mathscr{C}$ chooses a signing attribute set A_s such that $\mathcal{P}_s(A_s) = true$, calculates $\vec{a} := (a_1, a_2, \ldots, a_{\ell_s}) \leftarrow \mathsf{Reconstruct}(\mathbf{M}_s, \rho_s, A_s)$ satisfying $\sum_{i\in[\ell_s]} a_i \cdot \vec{M}_s^{(i)} = \vec{1}_{n_s}$ and $a_i = 0$ for all $i \in \{i|\rho_s(i) \notin A_s\}$, samples $(b_1, b_2, \ldots, b_{\ell_s}) \xleftarrow{\text{u}} \left\{(b_1, b_2, \ldots, b_{\ell_s}) \in \mathbb{Z}_p^{\ell_s} | \sum_{i\in[\ell_s]} b_i \cdot \vec{M}_s^{(i)} = \vec{0}_{n_s}\right\}$. To generate a ciphertext

$$\mathcal{CT} := \begin{pmatrix} \Omega_s := \langle \mathcal{P}_s, \sigma', \sigma'', \Gamma, \sigma, \{\sigma_i\}_{i\in[\ell_s]}, \chi\rangle, \\ \Omega_e := \langle \mathcal{P}_e, ct, E', E_1, \{\vec{E}_i := (E_{i1}, E_{i2}, E_{i3}), \vec{\varepsilon}_i := (\varepsilon_{i1}, \varepsilon_{i2})\}_{i\in[\ell_e]}\rangle, \\ \Omega_k := \left\langle \begin{array}{l} W^\circ, k_T, \vec{L} := (L_1, L_2, L_3, L_4), \\ \{\vec{K}_j := (K_{j1}, K_{j2}), \vec{k}_j := (k_{j1}, k_{j2}, k_{j3}, k_{j4}), u_j\}_{j\in[\varsigma]}, \end{array} \right\rangle, \\ \mathsf{tag2}, E_2, \varepsilon_2, \eta \end{pmatrix}$$

  of the message msg, $\mathscr{C}$ picks $\tilde{o}, \beta, \delta', \delta'', \delta''', \breve{r}, o_2, \eta, t_i, f_1, f_2, f_3, f_4, \breve{t}_j, \pi_{j1}, \pi_{j2}, \chi, \varepsilon_2, \varepsilon_{i1}, \varepsilon_{i2}, u_j \xleftarrow{\text{u}} \mathbb{Z}_p^*$, implicitly sets $\breve{o}_1 = H_4(ct||\Gamma||E'||\mathcal{P}_e||\mathcal{P}_s||W^\circ) := \tilde{o} - (\beta z_5)^{-1}\phi^{q+1}$ (this is of type (ii) query mentioned above), and computes the components of $\mathcal{CT}$ in the following manner.
  Set $\sigma := g^{\alpha'} g_4^{\frac{\breve{r}}{\delta}} g_5^{\beta\tilde{o}} g_6^{\beta}\left(\prod_{i\in[\ell_s]} H_1\left(\rho_s(i)\right)^{\frac{\breve{r}a_i}{\theta} + o_2 b_i}\right) g_5^{-\chi}$ and calculate the other components of $\mathcal{CT}$ as in Equation (3.1). Finally, $\mathscr{C}$ sends this $\mathcal{CT}$ to $\mathscr{A}$.

- $\mathcal{O}_{\mathcal{DR}\text{-}\mathcal{UV}}(\mathcal{CT}, A_d, \mathcal{P}_t)$: First, $\mathscr{C}$ obtains $[\mathcal{DK}_{A_d}, \widetilde{\mathcal{TD}}_{\mathcal{P}_t}] \leftarrow \mathcal{O}'_{\mathcal{DRR}}(A_d, \mathcal{P}_t)$. Next, it computes $\mathcal{IDR} \leftarrow \mathsf{offDataRetReq}(\mathcal{PP}, \mathcal{DK}_{A_d}, \mathcal{P}_t)$, $[\mathcal{DRR}, \mathcal{TDK}] \leftarrow \mathsf{onDataRetReq}(\mathcal{PP}, \mathcal{IDR}, \widetilde{\mathcal{TD}}_{\mathcal{P}_t})$. Next, it executes

$\mathsf{Test}(\mathcal{PP}, \mathcal{CT}, \mathcal{DRR}, \mathcal{CSK})$, $\mathsf{Transform}(\mathcal{PP}, \mathcal{CT}, \mathcal{DRR}, \mathcal{CSK})$ and $\mathsf{Unsigncrypt\text{-}Verify}(\mathcal{PP}, \mathcal{CT}_{out}, \mathcal{TDK})$ algorithms, in that order. $\mathscr{C}$ sends the final output to $\mathscr{A}$.

When this query phase is over, $\mathscr{A}$ outputs a forgery ciphertext $\mathcal{CT}^\star$ of the message $msg^\star$ for the encryption policy $\mathcal{P}_e^\star$, keyword set $W^\star$, and the signing policy $\mathcal{P}_s^\star$.

$\mathscr{A}$ wins the game if all the following conditions are true.
(i) $\exists A_d, \mathcal{P}_t \ni \mathcal{P}_e^\star(A_d) = true \wedge \mathcal{P}_t(W^\star) = true$,
(ii) $\mathsf{Test}(\mathcal{PP}, \mathcal{CT}^\star, \mathcal{DRR} := \langle \mathcal{TD}_{\mathcal{P}_t^\circ}, \mathcal{TK}_{A_d} \rangle, \mathcal{CSK}) = \mathcal{CT}^\star$,
(iii) $\mathsf{Transform}(\mathcal{PP}, \mathcal{CT}^\star, \mathcal{DRR}, \mathcal{CSK}) = \mathcal{CT}_{out}^\star$,
(iv) $\mathsf{Unsigncrypt\text{-}Verify}(\mathcal{PP}, \mathcal{CT}_{out}^\star, \mathcal{TDK}) = msg^\star \neq \perp$ and
(v) $\mathscr{A}$ did not query for $\mathcal{O}_{\mathcal{SC}}$ using the input $(msg^\star, \mathcal{P}_s^\star, \mathcal{P}_e^\star, W^\star)$.
The ciphertext $\mathcal{CT}^\star$ is parsed as $\mathcal{CT}^\star := \big\langle \Omega_s^\star := \langle \mathcal{P}_s^\star, \sigma'^\star, \sigma''^\star, \Gamma^\star, \sigma^\star, \{\sigma_i^\star\}_{i\in[\ell_s^\star]}, \chi^\star \rangle$, $\Omega_e^\star := \langle \mathcal{P}_e^\star, ct^\star, E'^\star, E_1^\star, \{\vec{E}_i^\star, \vec{\varepsilon}_i^\star\}_{i\in[\ell_e^\star]} \rangle, \Omega_k^\star, \mathsf{tag2}^\star, E_2^\star, \varepsilon_2^\star, \eta^\star \big\rangle$.

From Remark 4, conditions (i)-(iv) imply that
$\sigma^\star := g^{\frac{\alpha}{\theta}} g_4^{\frac{\breve{r}}{\theta}} \big(g_5^{\breve{o}_1} g_6\big)^{\beta} \big(\prod_{i\in[\ell_s^\star]} H_1\big(\rho_s^\star(i)\big)^{\frac{\breve{r}a_i}{\theta} + o_2 b_i} \big) g_5^{-\chi^\star}$, $\sigma_i^\star := g^{\frac{\breve{r}a_i}{\theta} + o_2 b_i}$, $E_1^\star := g^\beta$, where $\breve{r}, \beta, o_2, \chi^\star$ are random exponents, $\breve{o}_1 := H_4(ct^\star || \Gamma^\star || E'^\star || \mathcal{P}_e^\star || \mathcal{P}_s^\star || W^{\star\circ})$, $(a_1, a_2, \ldots, a_{\ell_s^\star})$ and $(b_1, b_2, \ldots, b_{\ell_s^\star})$ are vectors satisfying respectively $\sum_{i\in[\ell_s^\star]} a_i \cdot \vec{M}_s^{\star(i)} = \vec{1}_{n_s^\star}$ and $\sum_{i\in[\ell_s^\star]} b_i \cdot \vec{M}_s^{\star(i)} = \vec{0}_{n_s^\star}$. Condition (v) implies that
$\breve{o}_1 := H_4(ct^\star || \Gamma^\star || E'^\star || \mathcal{P}_e^\star || \mathcal{P}_s^\star || W^{\star\circ}) = \tilde{o}$ (this is of type (i) query, $\mathscr{C}$ can obtain this value from Tab_{H_4}). Now, $\mathscr{C}$ can calculate the unknown value $g^{\phi^{q+1}}$ as

$$g^{\phi^{q+1}} = \frac{\sigma^\star \cdot g_5^{\chi^\star}}{g^{\alpha'} \big(E_1^\star\big)^{\tilde{o}z_5 + z_6} \prod_{i\in[\ell_s^\star]} \big(\sigma_i^\star\big)^{\upsilon_{\rho_s^\star(i)}}}$$

Therefore, if the advantage of $\mathscr{A}$ in the game $\mathsf{Game}_{\mathscr{A}}^{\mathsf{EUF\text{-}CMA}}$ is non-negligible, then $\mathscr{C}$ can solve the q-DHE problem with non-negligible advantage. □

Theorem 3 (DO Privacy)**.** *ABDSRS preserves DO privacy.*

Proof. The challenger $\mathscr{C}$ interacts with an adversary $\mathscr{A}$ as in $\mathsf{Game}_{\mathscr{A}}^{\mathsf{DO\text{-}Privacy}}$ (formulated in Section 3.2.2).

(1) $\mathscr{C}$ computes $[\mathcal{PP}, \mathcal{MK}] \leftarrow \mathsf{KGC\text{-}Setup}(1^\wp)$, $[\mathcal{TPK}, \mathcal{TSK}] \leftarrow \mathsf{TGC\text{-}Setup}(\mathcal{PP})$, $[\mathcal{CPK}, \mathcal{CSK}] \leftarrow \mathsf{Cloud\text{-}Setup}(\mathcal{PP})$; and sends $[\mathcal{PP}, \mathcal{MK}, \mathcal{TPK}, \mathcal{TSK}, \mathcal{CPK}, \mathcal{CSK}]$ to $\mathscr{A}$.

(2) In this step, $\mathscr{A}$ need not to query any oracle and it can compute required components by itself because $\mathscr{A}$ has the knowledge of system master secret key, cloud secret key and trapdoor secret key. $\mathscr{A}$ sends to $\mathscr{C}$ an encryption policy $\mathcal{P}_e$, a message msg, a signing policy $\mathcal{P}_s$, a keyword set W and two signing attribute sets $A_s^{(0)}, A_s^{(1)}$ obeying the condition $\mathcal{P}_s(A_s^{(0)}) = true = \mathcal{P}_s(A_s^{(1)})$.

(3) $\mathscr{C}$ samples $i \xleftarrow{\text{u}} \{0,1\}$, computes the signing key $\mathcal{SK}_{A_s^{(i)}} \leftarrow \mathsf{sKeyGen}(\mathcal{PP}, \mathcal{MK}, A_s^{(i)})$, the challenge ciphertext $\mathcal{CT}^\star \leftarrow \mathsf{onSigncrypt}\big(\mathcal{PP}, \mathsf{offSigncrypt}(\mathcal{PP}, \mathcal{SK}_{A_s^{(i)}}, \mathcal{TPK}, \mathcal{CPK}, \mathcal{P}_s), msg, \mathcal{P}_e, W\big)$, and sends $\mathcal{CT}^\star$ to $\mathscr{A}$.

(4) $\mathscr{A}$ outputs its guess i' of i.

From Remark 4, the signature $\Omega_s := \langle \mathcal{P}_s, \sigma', \sigma'', \Gamma, \sigma, \{\sigma_i\}_{i\in[\ell_s]}, \chi\rangle$ of a data file msg for the signing policy $\mathcal{P}_s := (\mathbf{M}_s, \rho_s)$ is of the form

$$\Gamma := H_7(g_T^{1/\theta}) \oplus H_8\big(e(g,g_4)^{\delta'\cdot\delta''\cdot\mathcal{CSK}}\big), \sigma' := g^{\delta'}, \sigma'' := g_4^{\delta''},$$

$$\sigma := g^{\frac{\alpha}{\theta}} g_4^{\frac{\breve{r}}{\theta}} \big(g_5^{\breve{o}_1} g_6\big)^{\beta} \Big(\prod_{i\in[\ell_s]} H_1\big(\rho_s(i)\big)^{\frac{\breve{r}a_i}{\theta} + o_2 b_i} \Big) g_5^{-\chi}, \sigma_i := g^{\frac{\breve{r}a_i}{\theta} + o_2 b_i},$$

where $\theta, \beta, \delta', \delta'', \breve{r}, o_2$, are random elements of $\mathbb{Z}_p^*$, $\breve{o}_1 := H_4(ct||\Gamma||E'||\mathcal{P}_e||\mathcal{P}_s||W^\circ)$, $ct := msg \oplus \mathsf{KDF}\big(g_T^{\beta+\frac{1}{\theta}}\big)$, $(a_1, a_2, \ldots, a_{\ell_s})$ and $(b_1, b_2, \ldots, b_{\ell_s})$ are vectors satisfying respectively $\sum_{i\in[\ell_s]} a_i \cdot \vec{M}_s^{(i)} = \vec{1}_{n_s}$ and $\sum_{i\in[\ell_s]} b_i \cdot \vec{M}_s^{(i)} = \vec{0}_{n_s}$, α is the system master key, $\mathcal{CSK}$ is cloud secret key, and others are public parameters.

Since o_2 is random, each $\frac{\breve{r}a_i}{\theta} + o_2 b_i$ is random, and hence all the components $\sigma', \sigma'', \Gamma, \sigma, \sigma_i$ of the signature are random elements from adversary's point of view. That is, the signature is independent of the signing key being used to generate it. Therefore, the challenge ciphertext gives no information about i in $\mathsf{Game}_{\mathscr{A}}^{\mathsf{DO\text{-}Privacy}}$ to the adversary $\mathscr{A}$, resulting in $\mathscr{A}$ has to output just random guess i'. In this case, $\text{Prob}[i' = i] = 1/2$. Hence, $\mathscr{A}$'s advantage $\mathsf{Adv}_{\mathscr{A}}^{\mathsf{DO\text{-}Privacy}}(1^\wp) \stackrel{\text{def}}{=} \text{Prob}[i' = i] = 1/2$. Thus, ABDSRS provides DO privacy. □

Theorem 4 (Verifiability)**.** *ABDSRS is verifiable under the assumption that H_3 is a collision-resistant hash function.*

Proof. If there exists an adversary $\mathscr{A}$ that wins the verifiability game $\mathsf{Game}_{\mathscr{A}}^{\mathsf{verifiability}}$ (presented in Section 3.2.2) of ABDSRS, then an authorized DU $\mathscr{C}$ identifies a collision for the hash function H_3 by communicating with $\mathscr{A}$ in the following way.

(1) $\mathscr{C}$ computes $[\mathcal{PP}, \mathcal{MK}] \leftarrow \mathsf{KGC\text{-}Setup}(1^{\wp})$, $[\mathcal{TPK}, \mathcal{TSK}] \leftarrow \mathsf{TGC\text{-}Setup}(\mathcal{PP})$, $[\mathcal{CPK}, \mathcal{CSK}] \leftarrow \mathsf{Cloud\text{-}Setup}(\mathcal{PP})$; and sends $[\mathcal{PP}, \mathcal{TPK}, \mathcal{CPK}, \mathcal{CSK}]$ to $\mathscr{A}$. The secret keys $\mathcal{MK}$ and $\mathcal{TSK}$ are kept secret by $\mathscr{C}$.

(2) $\mathscr{A}$ adaptively queries signing key generation oracle $\mathcal{O}_{\mathcal{SKG}}(A_s)$, data retrieval request generation oracle $\mathcal{O}''_{\mathcal{DRR}}(A_d, \mathcal{P}_t)$, signcryption oracle $\mathcal{O}_{\mathcal{SC}}(msg, \mathcal{P}_s, \mathcal{P}_e, W)$ and data retrieval cum unsigncrypt-verify oracle $\mathcal{O}_{\mathcal{DR}\text{-}\mathcal{UV}}(\mathcal{CT}, A_d, \mathcal{P}_t)$. As $\mathscr{C}$ has the system master secret key $\mathcal{MK}$ and TGC secret key $\mathcal{TSK}$, it simulates $\mathscr{A}$'s queries properly. At the end of this phase, $\mathscr{A}$ announces an encryption policy $\mathcal{P}_e^{\star}$, a message $msg^{\star}$, a signing policy $\mathcal{P}_s^{\star}$, a keyword set $W^{\star}$, and sends $[msg^{\star}, \mathcal{P}_e^{\star}, \mathcal{P}_s^{\star}, W^{\star}]$ to $\mathscr{C}$.

(3) $\mathscr{C}$ selects a set A_s satisfying $\mathcal{P}_s^{\star}(A_s) = true$ and obtains a signing key $\mathcal{SK}_{A_s} \leftarrow \mathsf{sKeyGen}(\mathcal{PP}, \mathcal{MK}, A_s)$. Next, it computes $\mathcal{CT}^{\star} \leftarrow \mathsf{onSigncrypt}\big(\mathcal{PP}, \mathcal{IC}, msg^{\star}, \mathcal{P}_e^{\star}, W^{\star}\big)$, where $\mathcal{IC} \leftarrow \mathsf{offSigncrypt}(\mathcal{PP}, \mathcal{SK}_{A_s}, \mathcal{TPK}, \mathcal{CPK}, \mathcal{P}_s^{\star})$. Then, the ciphertext $\mathcal{CT}^{\star}$ will be given to $\mathscr{A}$. Here $\mathcal{CT}^{\star} := \langle \Omega_s^{\star}, \Omega_e^{\star}, \Omega_k^{\star}, \mathsf{tag2}^{\star}, E_2^{\star}, \varepsilon_2^{\star}, \eta^{\star} \rangle$.

(4) Again $\mathscr{A}$ queries the oracles $\mathcal{O}_{\mathcal{SKG}}, \mathcal{O}''_{\mathcal{DRR}}, \mathcal{O}_{\mathcal{SC}}, \mathcal{O}_{\mathcal{DR}\text{-}\mathcal{UV}}$ and obtains the respective responses as in step (2). At the end of this phase, $\mathscr{A}$ outputs a decryption attribute set A_d, a keyword policy $\mathcal{P}_t$ and a transformed ciphertext $\mathcal{CT}_{out} := \langle \Delta_1, \Delta_2, \Delta_3, ct, \mathsf{tag2}^{\star} \rangle$ such that $\mathcal{P}_e^{\star}(A_d) = true \wedge \mathcal{P}_t(W^{\star}) = true$.

Suppose that the tuple $[\![A_d, \mathcal{P}_t, \mathcal{TK}_{A_d}, \mathcal{TD}_{\mathcal{P}_t^{\circ}}, \mathcal{TDK}]\!]$ is in table $\mathsf{Tab}''_{\mathcal{DRR}}$, where $\mathcal{TDK} := \langle tdk_1, tdk_2 \rangle$. If not, it can be generated by querying the oracle $\mathcal{O}''_{\mathcal{DRR}}$ with the input $(A_d, \mathcal{P}_t)$. Since $\mathscr{A}$ can break the verifiability of ABDSRS, $\mathscr{C}$ recovers a message $msg \leftarrow \mathsf{Unsigncrypt\text{-}Verify}(\mathcal{PP}, \mathcal{CT}_{out}, \mathcal{TDK})$ with the property that $msg \notin \{msg^{\star}, \perp\}$ as follows. $\mathscr{C}$ (i) computes $\Lambda := (\Delta_1)^{-tdk_1} \cdot (\Delta_2)^{tdk_2} \cdot \Delta_3$ (ii) observes $H_3\big(H_2(\Lambda)||ct\big) = \mathsf{tag2}^{\star}$ and (iii) obtains $msg = ct \oplus \mathsf{KDF}(\Lambda)$.

If $\Lambda^{\star}$ and $ct^{\star}$ are the respective components of Λ and ct being used in the generation of $\mathcal{CT}^{\star}$, then there are two possibilities $\Lambda \neq \Lambda^{\star}$ or $\Lambda = \Lambda^{\star}$. Note that the condition (ii) implies that $H_3\big(H_2(\Lambda)||ct\big) = H_3\big(H_2(\Lambda^{\star})||ct^{\star}\big)$.

In case $\Lambda \neq \Lambda^{\star}$, $H_2(\Lambda)||ct \neq H_2(\Lambda^{\star})||ct^{\star}$ and hence the pair $\big(H_2(\Lambda)||ct, H_2(\Lambda^{\star})||ct^{\star}\big)$ forms a collision for the hash function H_3.

In case $\Lambda = \Lambda^{\star}$, since $ct \oplus \mathsf{KDF}(\Lambda) = msg \neq msg^{\star} = ct^{\star} \oplus \mathsf{KDF}(\Lambda^{\star})$, we have that $ct \neq ct^{\star}$. So, $H_2(\Lambda)||ct \neq H_2(\Lambda^{\star})||ct^{\star}$ and hence the pair $\big(H_2(\Lambda)||ct, H_2(\Lambda^{\star})||ct^{\star}\big)$ causes H_3 to collide.

$\mathscr{C}$ detects a collision for H_3 in each scenario. Due to the collision-resistance nature of H_3, it is impossible for $\mathscr{A}$ to gain a non-negligible advantage and win $\mathsf{Game}^{\mathsf{verifiability}}_{\mathscr{A}}$. Therefore, we prove ABDSRS to be verifiable. □

Remark 8. *Note that the components X_1 and X_2 (computed in Equations (3.3) and (3.4)) can also be computed as*

$$X_1 := e\Big(E_1, \prod_{i\in[\ell_t]} \big(T_{1i1}\cdot g_{15}^{B_{1i}}\big)^{a'_i}\Big)\cdot \prod_{i\in[\ell_t]} e\Big(\underbrace{g^{\varrho_i}\cdot T_0}_{:=\varrho_i^{\times}}, \big(K_{\rho_t^{\circ}(i)1}\cdot g_{11}^{u_{\rho_t^{\circ}(i)}}\big)^{a'_i}\Big) \tag{3.7}$$

$$\begin{aligned} X_2 := \prod_{i\in[\ell_t]} \Big\{ & e\Big(\underbrace{g_{16}^{k_{\rho_t^{\circ}(i)1}}\cdot L_1}_{:=k^{\times}_{\rho_t^{\circ}(i)1}}, \big(T_{2i2}\cdot g_{11}^{B_{2i2}}\big)^{a'_i}\Big)\cdot e\Big(\underbrace{g_{17}^{k_{\rho_t^{\circ}(i)2}}\cdot L_2}_{:=k^{\times}_{\rho_t^{\circ}(i)2}}, \big(T_{2i1}\cdot g_{11}^{B_{2i1}}\big)^{a'_i}\Big) \\ & \times e\Big(\underbrace{g_{18}^{k_{\rho_t^{\circ}(i)3}}\cdot L_3}_{:=k^{\times}_{\rho_t^{\circ}(i)3}}, \big(T_{2i4}\cdot g_{11}^{B_{2i4}}\big)^{a'_i}\Big)\cdot e\Big(\underbrace{g_{19}^{k_{\rho_t^{\circ}(i)4}}\cdot L_4}_{:=k^{\times}_{\rho_t^{\circ}(i)4}}, \big(T_{2i3}\cdot g_{11}^{B_{2i3}}\big)^{a'_i}\Big)\Big\} \end{aligned} \tag{3.8}$$

Hence, one can run Test *algorithm using Equations (3.7) and (3.8) instead of Equations (3.3) and (3.4), respectively. Due to this fact, we can modify the distribution of $\mathcal{DRR}$ (presented in Remark 5) and $\mathcal{CT}$ (given in Remark 4) as follows.*

- *$m\mathcal{DRR}$ **(modified distribution of $\mathcal{DRR}$):** It is the same as $\mathcal{DRR}$, except where the components $\big(T_0, \{\varrho_i\}_{i\in[\ell_t]}\big)$ are replaced by $\{\varrho_i^{\times}\}_{i\in[\ell_t]}$, here $\varrho_i^{\times} := g^{\varrho_i}\cdot T_0 = g^{\varpi_1\varpi_2\check{r}_i+\varpi_3\varpi_4\check{r}'_i}$ (from Equation (3.2)). More specifically,*

$$m\mathcal{DRR} := \begin{pmatrix} \mathcal{TD}_{\mathcal{P}_t^{\circ}} := \Big\langle \begin{matrix} \mathcal{P}_t^{\circ}, T_1, T_2, \\ \{\varrho_i^{\times}, \vec{T}_{1i} := (T_{1i1}, T_{1i2}), \vec{T}_{2i} := (T_{2i1}, T_{2i2}, T_{2i3}, T_{2i4})\}_{i\in[\ell_t]}, \\ \{B_{1i}, \vec{B}_{2i} := (B_{2i1}, B_{2i2}, B_{2i3}, B_{2i4})\}_{i\in[\ell_t]} \end{matrix} \Big\rangle, \\ \mathcal{TK}_{A_d} := \Big\langle\ A_d, T'_1, T'_2, \{T'_{3,x}, T'_{4,x}\}_{x\in A_d}\ \Big\rangle \end{pmatrix} \tag{3.9}$$

where $\varrho_i^{\times} := g^{\varpi_1\varpi_2\check{r}_i+\varpi_3\varpi_4\check{r}'_i}$ and the other components are identical to that of Equation (3.2).

- *$m\mathcal{CT}$ **(modified distribution of $\mathcal{CT}$):** This is the same as $\mathcal{CT}$ except that the components $\big(\vec{L}, \{\vec{k}_j\}_{j\in[\varsigma]}\big)$ are replaced by the components $\{\vec{k}_j^{\times}\}_{j\in[\varsigma]}$, where $\vec{k}_j^{\times} := (k_{j1}^{\times}, k_{j2}^{\times}, k_{j3}^{\times}, k_{j4}^{\times})$ and $k_{j1}^{\times} := g_{16}^{k_{j1}}\cdot L_1 = g_{16}^{\check{t}_j-\pi_{j1}}, k_{j2}^{\times} := g_{17}^{k_{j2}}\cdot L_2 = g_{17}^{\pi_{j1}}$, $k_{j3}^{\times} := g_{18}^{k_{j3}}\cdot L_3 = g_{18}^{\check{t}_j-\pi_{j2}}, k_{j4}^{\times} := g_{19}^{k_{j4}}\cdot L_4 = g_{19}^{\pi_{j2}}$ (from Equation (3.1)).*

We use these modified distributions to answer adversary's queries in IND-CKA security proof (for Type-1 adversary) presented in Lemma 3. □

Table 3.2: The sequence of $2\varsigma + 2$ games $\mathsf{G}_{real}, \mathsf{G}_0, \mathsf{G}_1, \ldots, \mathsf{G}_{2\varsigma}$.

Game	*The Challenge Ciphertext* $\mathcal{CT}^\star$
G_{real}	$\Omega_k := \left\langle \begin{array}{l} W^\circ, k_T, \{\vec{K}_j := (K_{j1}, K_{j2}), u_j\}_{j\in[\varsigma]}, \\ \vec{k}_1^\times := (k_{11}^\times, k_{12}^\times, k_{13}^\times, k_{14}^\times), \vec{k}_2^\times := (k_{21}^\times, k_{22}^\times, k_{23}^\times, k_{24}^\times), \ldots, \vec{k}_\varsigma^\times := (k_{\varsigma 1}^\times, k_{\varsigma 2}^\times, k_{\varsigma 3}^\times, k_{\varsigma 4}^\times), \end{array} \right\rangle$
G_0	$\Omega_k := \left\langle \begin{array}{l} W^\circ, \boxed{r_T}, \{\vec{K}_j := (K_{j1}, K_{j2}), u_j\}_{j\in[\varsigma]}, \\ \vec{k}_1^\times := (k_{11}^\times, k_{12}^\times, k_{13}^\times, k_{14}^\times), \vec{k}_2^\times := (k_{21}^\times, k_{22}^\times, k_{23}^\times, k_{24}^\times), \ldots, \vec{k}_\varsigma^\times := (k_{\varsigma 1}^\times, k_{\varsigma 2}^\times, k_{\varsigma 3}^\times, k_{\varsigma 4}^\times), \end{array} \right\rangle$
G_1	$\Omega_k := \left\langle \begin{array}{l} W^\circ, \boxed{r_T}, \{\vec{K}_j := (K_{j1}, K_{j2}), u_j\}_{j\in[\varsigma]}, \\ \vec{k}_1^\times := (\boxed{R_{11}}, k_{12}^\times, k_{13}^\times, k_{14}^\times), \vec{k}_2^\times := (k_{21}^\times, k_{22}^\times, k_{23}^\times, k_{24}^\times), \ldots, \vec{k}_\varsigma^\times := (k_{\varsigma 1}^\times, k_{\varsigma 2}^\times, k_{\varsigma 3}^\times, k_{\varsigma 4}^\times), \end{array} \right\rangle$
G_2	$\Omega_k := \left\langle \begin{array}{l} W^\circ, \boxed{r_T}, \{\vec{K}_j := (K_{j1}, K_{j2}), u_j\}_{j\in[\varsigma]}, \\ \vec{k}_1^\times := (\boxed{R_{11}}, k_{12}^\times, k_{13}^\times, k_{14}^\times), \vec{k}_2^\times := (\boxed{R_{21}}, k_{22}^\times, k_{23}^\times, k_{24}^\times), \ldots, \vec{k}_\varsigma^\times := (k_{\varsigma 1}^\times, k_{\varsigma 2}^\times, k_{\varsigma 3}^\times, k_{\varsigma 4}^\times), \end{array} \right\rangle$
$\vdots$	
G_ς	$\Omega_k := \left\langle \begin{array}{l} W^\circ, \boxed{r_T}, \{\vec{K}_j := (K_{j1}, K_{j2}), u_j\}_{j\in[\varsigma]}, \\ \vec{k}_1^\times := (\boxed{R_{11}}, k_{12}^\times, k_{13}^\times, k_{14}^\times), \vec{k}_2^\times := (\boxed{R_{21}}, k_{22}^\times, k_{23}^\times, k_{24}^\times), \ldots, \vec{k}_\varsigma^\times := (\boxed{R_{\varsigma 1}}, k_{\varsigma 2}^\times, k_{\varsigma 3}^\times, k_{\varsigma 4}^\times), \end{array} \right\rangle$
$\mathsf{G}_{\varsigma+1}$	$\Omega_k := \left\langle \begin{array}{l} W^\circ, \boxed{r_T}, \{\vec{K}_j := (K_{j1}, K_{j2}), u_j\}_{j\in[\varsigma]}, \\ \vec{k}_1^\times := (\boxed{R_{11}}, k_{12}^\times, \boxed{R_{13}}, k_{14}^\times), \vec{k}_2^\times := (\boxed{R_{21}}, k_{22}^\times, k_{23}^\times, k_{24}^\times), \ldots, \vec{k}_\varsigma^\times := (\boxed{R_{\varsigma 1}}, k_{\varsigma 2}^\times, k_{\varsigma 3}^\times, k_{\varsigma 4}^\times), \end{array} \right\rangle$
$\mathsf{G}_{\varsigma+2}$	$\Omega_k := \left\langle \begin{array}{l} W^\circ, \boxed{r_T}, \{\vec{K}_j := (K_{j1}, K_{j2}), u_j\}_{j\in[\varsigma]}, \\ \vec{k}_1^\times := (\boxed{R_{11}}, k_{12}^\times, \boxed{R_{13}}, k_{14}^\times), \vec{k}_2^\times := (\boxed{R_{21}}, k_{22}^\times, \boxed{R_{23}}, k_{24}^\times), \ldots, \vec{k}_\varsigma^\times := (\boxed{R_{\varsigma 1}}, k_{\varsigma 2}^\times, k_{\varsigma 3}^\times, k_{\varsigma 4}^\times), \end{array} \right\rangle$
$\vdots$	
$\mathsf{G}_{2\varsigma}$	$\Omega_k := \left\langle \begin{array}{l} W^\circ, \boxed{r_T}, \{\vec{K}_j := (K_{j1}, K_{j2}), u_j\}_{j\in[\varsigma]}, \\ \vec{k}_1^\times := (\boxed{R_{11}}, k_{12}^\times, \boxed{R_{13}}, k_{14}^\times), \vec{k}_2^\times := (\boxed{R_{21}}, k_{22}^\times, \boxed{R_{23}}, k_{24}^\times), \ldots, \vec{k}_\varsigma^\times := (\boxed{R_{\varsigma 1}}, k_{\varsigma 2}^\times, \boxed{R_{\varsigma 3}}, k_{\varsigma 4}^\times), \end{array} \right\rangle$

Lemma 3. *If the challenge keyword set has at most q keywords, then ABDSRS provides IND-CKA security against PPT Type-1 adversary, under the assumption that q-2 and DLin problems (given in Sections 2.3.4 and 2.3.5, respectively) are hard.*

Proof. We prove this security notion (modeled as $\mathsf{Game}^{\mathsf{IND\text{-}CKA}}_{\mathsf{Type\text{-}1}}$ in Section 3.2.2) employing a hybrid experiment which consisting of a sequence of games between a challenger $\mathscr{C}$ and a Type-1 adversary $\mathscr{A}$. The individual games differ in how $\mathscr{C}$ constructs the challenge ciphertext given to $\mathscr{A}$. Let $\mathcal{CT}^\star := \langle \Omega_s, \Omega_e, \Omega_k, \mathsf{tag2}, E_2, \varepsilon_2, \eta \rangle$ be the challenge ciphertext given to $\mathscr{A}$ during IND-CKA game. Let ς be the size of the keyword set used in computation of $\mathcal{CT}^\star$ such that $\varsigma \leq q$. The sequence of $2\varsigma + 2$ games $\mathsf{G}_{real}, \mathsf{G}_0, \mathsf{G}_1, \ldots, \mathsf{G}_{2\varsigma}$ is defined in Table 3.2. In these games, only the components in Ω_k are modified (precisely, some components are replaced by random elements) and all other elements of $\mathcal{CT}^\star$ remain unchanged. For brevity, we omit the components $\Omega_s, \Omega_e, \mathsf{tag2}, E_2, \varepsilon_2, \eta$ from $\mathcal{CT}^\star$ and present only the components of Ω_k in Table 3.2. Let r_T be the random element of $\mathbb{G}_T$ and $R_{11}, R_{13}, R_{21}, R_{23}, \ldots, R_{\varsigma 1}, R_{\varsigma 3}$ be random elements of $\mathbb{G}$. Note that all the components of Ω_k in the challenge ciphertext (computed in $\mathsf{G}_{2\varsigma}$) are random elements and hence the challenge ciphertext is independent of the two keyword sets submitted by $\mathscr{A}$. But, the challenge ciphertext of G_{real} is well formed. The challenge ciphertext in $\mathsf{G}_{2\varsigma}$ reveals nothing about its keyword set. To complete the proof, we show that the transitions from G_{real} to

G_0 to G_1 to ... to $\mathsf{G}_{2\varsigma}$ are all computationally indistinguishable (in Claims 1, 2 and 3).

Claim 1. *Under the hardness assumption of q-2 problem, no PPT Type-1 adversary can distinguish between the games* G_{real} *and* G_0 *with non-negligible advantage.*

Proof. Assume there is a Type-1 adversary $\mathscr{A}$ that distinguishes between the games G_{real} and G_0 with non-negligible advantage. Next, we introduce a challenger $\mathscr{C}$ which solves q-2 problem with non-negligible advantage by interacting with $\mathscr{A}$ as follows. Given the q-2 problem instance $\big\langle \Sigma, g, g^{\phi_1}, g^{\phi_2}, g^{\phi_3}, g^{(\phi_1\phi_3)^2}, \{g^{\psi_i}, g^{\phi_1\phi_3\psi_i}, g^{\phi_1\phi_3/\psi_i}, g^{\phi_1^2\phi_3\psi_i}, g^{\phi_2/\psi_i^2}, g^{\phi_2^2/\psi_i^2}\}_{i\in[q]}, \{g^{\phi_1\phi_3\psi_i/\psi_j}, g^{\phi_2\psi_i/\psi_j^2}, g^{\phi_1\phi_2\phi_3\psi_i/\psi_j^2}, g^{(\phi_1\phi_3)^2\psi_i/\psi_j}\}_{(i,j)\in[q,q],i\neq j}, Z\big\rangle$, the task for $\mathscr{C}$ is to ascertain if Z is equal to $e(g,g)^{\phi_1\phi_2\phi_3}$ or Z has been randomly selected from $\mathbb{G}_T$.

(1) $\mathscr{A}$ announces two challenge keyword sets $W_0^\star := \{[\mathcal{W}_1 : w_1^{(0)}], \ldots, [\mathcal{W}_\varsigma : w_\varsigma^{(0)}]\}$ and $W_1^\star := \{[\mathcal{W}_1 : w_1^{(1)}], \ldots, [\mathcal{W}_\varsigma : w_\varsigma^{(1)}]\}$. Note that these two sets satisfy the conditions $|W_0^\star| = |W_1^\star| = \varsigma$ and $W_0^{\star\circ} = W_1^{\star\circ} = \{\mathcal{W}_1, \ldots, \mathcal{W}_\varsigma\}$ of $\mathsf{Game}_{\mathsf{Type\text{-}1}}^{\mathsf{IND\text{-}CKA}}$.

(2) Firstly, $\mathscr{C}$ samples $\mu \xleftarrow{\text{u}} \{0,1\}$ and sets $W_\mu^\star := \{[\mathcal{W}_1 : w_1^{(\mu)}], \ldots, [\mathcal{W}_\varsigma : w_\varsigma^{(\mu)}]\}$. Then, $\mathscr{C}$ selects $\alpha', z_i \xleftarrow{\text{u}} \mathbb{Z}_p^*, i \in \{1, \ldots, 9, 11, \ldots, 15\}$, implicitly defines $\alpha := \alpha'\phi_2$ and sets $g_T := e(g, g^{\phi_2})^{\alpha'}, g_i := g^{z_i}$ for $i \in \{1, 2, \ldots, 9, 14, 15\}$, $g_{10} := g^{\phi_1}, g_{11} := g^{z_{11}} \prod_{j\in[\varsigma]} g^{\phi_2/\psi_j^2}, g_{12} := g^{z_{12}} \prod_{j\in[\varsigma]} (g^{\phi_1\phi_3/\psi_j})\big(g^{\phi_2/\psi_j^2}\big)^{-w_j^{(\mu)}}$, $g_{13} := g^{\phi_1} g^{z_{13}}$. Next, it chooses eight collision-resistant hash functions $\{H_i\}_{i=1}^8$ (as mentioned in construction) and sets $\mathcal{PP} := \langle \Sigma, g_T, g, \{g_i\}_{i=1}^{15}, \mathcal{M}, \mathsf{KDF}, \{H_i\}_{i=1}^8\rangle$, where $\mathcal{M} := \{0,1\}^{\ell_{msg}}$ is the message space and KDF is the key derivation function, and $\mathcal{MK} := (g^{\phi_2})^{\alpha'}$. Lastly, $\mathscr{C}$ picks $\tau_0, \varpi_1, \varpi_2, \varpi_3, \varpi_4, r_c \xleftarrow{\text{u}} \mathbb{Z}_p^*$, calculates $g_{16} := g^{\varpi_1}, g_{17} := g^{\varpi_2}, g_{18} := g^{\varpi_3}, g_{19} := g^{\varpi_4}$, $h_T := e(g, g_{15})^{\tau_0} \cdot e(g^{\phi_1}, g^{\phi_2})^{z_{15}}, Y_c := g_4^{r_c}$, and sets $\mathcal{TPK} := \langle h_T, g_{16}, g_{17}, g_{18}, g_{19}\rangle$, $\mathcal{TSK} := \langle \tau := \tau_0 + \phi_1\phi_2, \varpi_1, \varpi_2, \varpi_3, \varpi_4\rangle$, $\mathcal{CPK} := Y_c$, $\mathcal{CSK} := r_c$. Note that $\tau := \tau_0 + \phi_1\phi_2$ is implicitly defined. $\mathscr{C}$ sends $[\mathcal{PP}, \mathcal{TPK}, \mathcal{CPK}, \mathcal{CSK}]$ to $\mathscr{A}$.

(3) $\mathscr{A}$ queries signing key generation oracle $\mathcal{O}_{\mathcal{SKG}}(A_s)$, data retrieval request generation oracle $\mathcal{O}'''_{\mathcal{DRR}}(A_d, \mathcal{P}_t)$ and test oracle $\mathcal{O}_{test}(\mathcal{CT}, \mathcal{P}_t)$. Then $\mathscr{C}$ responds as follows.

- $\mathcal{O}_{\mathcal{SKG}}(A_s)$: $\mathscr{C}$ chooses $r' \xleftarrow{\text{u}} \mathbb{Z}_p^*$, sets $S_1 := (g^{\phi_2})^{\alpha'} g_4^{r'}, S_2 := g^{r'}$, $S_{3,y} := (H_1(y))^{r'}, \forall y \in A_s$, and returns $\mathcal{SK}_{A_s} := \langle A_s, S_1, S_2, \{S_{3,y}\}_{y\in A_s}\rangle$.
- $\mathcal{O}'''_{\mathcal{DRR}}(A_d, \mathcal{P}_t)$: $\mathscr{A}$ submits a decryption attribute set A_d and a keyword policy $\mathcal{P}_t$ with the condition that $\mathcal{P}_t(W_0^\star) = false \wedge \mathcal{P}_t(W_1^\star) = false$. Hence

$\mathcal{P}_t(W^{\star}_{\mu}) = false$. Let $\mathcal{P}_t := (\mathbf{M}_t, \rho^{\circ}_t, \{w_{\rho^{\circ}_t(i)}\}_{i\in[\ell_t]})$, where $\mathbf{M}_t$ is a matrix of size $\ell_t \times n_t$. Since $\mathcal{P}_t(W^{\star}_{\mu}) = false$, $\mathscr{C}$ can compute a vector $\vec{\delta} := (\delta_1, \delta_2, \ldots, \delta_{n_t}) \in \mathbb{Z}_p^{n_t}$ such that $\delta_1 = -1$ and $\vec{M}^{(i)}_t \cdot \vec{\delta} = 0$, $\forall i \in \{i | \rho^{\circ}_t(i) \in W^{\star\circ}_{\mu}\}$. It selects $o_2, \ldots, o_{n_t}, \breve{f}, f' \xleftarrow{\text{u}} \mathbb{Z}^*_p$, sets $T_1 := g^{\breve{f}}$, $T_2 := g_4^{f'}, \breve{\partial} := H_5(e(T_1, T_2)^{r_c})$, and implicitly sets $\vec{\vartheta} := (0, o_2, \ldots, o_{n_t}) - (\tau_0 + \phi_1\phi_2)\breve{\partial}\vec{\delta}$. For each row $i \in [\ell_t]$, its share is

$$\vartheta_i = \vec{M}^{(i)}_t \cdot \vec{\vartheta} = \vec{M}^{(i)}_t \cdot (0, o_2, \ldots, o_{n_t}) - \tau_0\breve{\partial}\vec{M}^{(i)}_t \cdot \vec{\delta} - \phi_1\phi_2\breve{\partial}\vec{M}^{(i)}_t \cdot \vec{\delta}$$

It samples $\gamma_0, \breve{\tau}_0, o'_2, \ldots, o'_{n_t} \xleftarrow{\text{u}} \mathbb{Z}^*_p$, implicitly defines $\breve{\tau} := \breve{\tau}_0\breve{\partial}\phi_1\phi_2 z_{15}/z_{14}$, $trk := \phi_1\phi_2\breve{\partial}z_{15}/z_{14}$, $tdk_1 := \breve{\tau}_0, tdk_2 := \gamma = \gamma_0\phi_2$, $\breve{\vec{\vartheta}} := (0, o'_2, \ldots, o'_{n_t}) - (\phi_1\phi_2\breve{\partial}z_{15}/z_{14})\vec{\delta}$.
In this case, the share of the row $i \in [\ell_t]$ is

$$\breve{\vartheta}_i := \vec{M}^{(i)}_t \cdot \breve{\vec{\vartheta}} = \vec{M}^{(i)}_t \cdot (0, o'_2, \ldots, o'_{n_t}) - (\phi_1\phi_2\breve{\partial}z_{15}/z_{14})\vec{M}^{(i)}_t \cdot \vec{\delta}$$

$\mathscr{C}$ calculates the trapdoor $\mathcal{TD}_{\mathcal{P}^{\circ}_t} := \langle \mathcal{P}^{\circ}_t, T_1, T_2, \{\varrho_i^{\times}, \vec{T}_{1i}, \vec{T}_{2i}, B_{1i}, \vec{B}_{2i}\}_{i\in[\ell_t]}\rangle$ (mentioned in Equation (3.9)) in the following way.
Case-1: For the row $i \in [\ell_t]$ where $\rho^{\circ}_t(i) \in W^{\star\circ}_{\mu}$. In this case, $\vec{M}^{(i)}_t \cdot \vec{\delta} = 0$ and therefore $\vartheta_i = \vec{M}^{(i)}_t \cdot (0, o_2, \ldots, o_{n_t}) \overset{\text{let}}{=} \mathsf{vt}_i$
and $\breve{\vartheta}_i = \vec{M}^{(i)}_t \cdot (0, o'_2, \ldots, o'_{n_t}) \overset{\text{let}}{=} \mathsf{ut}_i$.
Choose $\breve{r}_i, \breve{r}'_i, B_{1i}, B_{2i1}, B_{2i2}, B_{2i3}, B_{2i4} \xleftarrow{\text{u}} \mathbb{Z}^*_p$, set

$$\begin{aligned}
\varrho_i^{\times} &:= g^{\varpi_1\varpi_2\breve{r}_i + \varpi_3\varpi_4\breve{r}'_i}, \\
T_{1i1} &:= g_{15}^{\mathsf{vt}_i} \cdot g_{10}^{\varpi_1\varpi_2\breve{r}_i + \varpi_3\varpi_4\breve{r}'_i} \cdot g_{15}^{-B_{1i}}, \\
T_{1i2} &:= g_{14}^{\mathsf{ut}_i} \cdot H_6\big(e(g, g_4)^{\breve{f}f'\cdot r_c} || \mathcal{P}^{\circ}_t || M^{(i)}_t\big) \cdot g_{13}^{\varpi_1\varpi_2\breve{r}_i + \varpi_3\varpi_4\breve{r}'_i},
\end{aligned}$$

and compute the other components of $\mathcal{TD}_{\mathcal{P}^{\circ}_t}$ as in Equation (3.2).
Case-2: For the row $i \in [\ell_t]$ where $\rho^{\circ}_t(i) \notin W^{\star\circ}_{\mu}$ (i.e., $w_{\rho^{\circ}_t(i)} \neq w^{(\mu)}_j, \forall j \in [\varsigma]$). In this case, $\vec{M}^{(i)}_t \cdot \vec{\delta} \neq 0$, and $\mathscr{C}$ cannot compute $\phi_1\phi_2\breve{\partial}\vec{M}^{(i)}_t \cdot \vec{\delta}$ and $(\phi_1\phi_2\breve{\partial}z_{15}/z_{14})\vec{M}^{(i)}_t \cdot \vec{\delta}$. However, by properly defining $\breve{r}_i, \breve{r}'_i$, $\mathscr{C}$ is able to calculate $\varrho_i^{\times}, T_{1i1}, T_{1i2}, T_{2i1}, T_{2i2}, T_{2i3}, T_{2i4}$.
Choose $r_i, r'_i, B_{1i}, B_{2i1}, B_{2i2}, B_{2i3}, B_{2i4} \xleftarrow{\text{u}} \mathbb{Z}^*_p$, implicitly define

$$\breve{r}_i := r_i + \frac{\breve{\partial}\phi_2 z_{15}(\vec{M}^{(i)}_t \cdot \vec{\delta})}{2\varpi_1\varpi_2} - \sum_{j\in[\varsigma]} \frac{\phi_1\phi_3\psi_j\breve{\partial}z_{15}(\vec{M}^{(i)}_t \cdot \vec{\delta})}{2\varpi_1\varpi_2(w_{\rho^{\circ}_t(i)} - w^{(\mu)}_j)}$$

$$\breve{r}_i' \;:=\; r_i' + \frac{\breve{\partial}\phi_2 z_{15}(\vec{M}_t^{(i)}\cdot\vec{\delta})}{2\varpi_3\varpi_4} - \sum_{j\in[\varsigma]} \frac{\phi_1\phi_3\psi_j\breve{\partial}z_{15}(\vec{M}_t^{(i)}\cdot\vec{\delta})}{2\varpi_3\varpi_4(w_{\rho_t^\circ(i)} - w_j^{(\mu)})}$$

and compute

$$\varrho_i^{\times} := g^{\varpi_1\varpi_2 r_i + \varpi_3\varpi_4 r_i'} \cdot (g^{\phi_2})^{\breve{\partial}z_{15}\vec{M}_t^{(i)}\cdot\vec{\delta}} \cdot \prod_{j\in[\varsigma]} (g^{\phi_1\phi_3\psi_j})^{\frac{-\breve{\partial}z_{15}(\vec{M}_t^{(i)}\cdot\vec{\delta})}{w_{\rho_t^\circ(i)} - w_j^{(\mu)}}}$$

$$T_{1i1} := g_{15}^{\mathsf{vt}_i - \breve{\partial}\tau_0\vec{M}_t^{(i)}\cdot\vec{\delta}} \cdot g_{10}^{\varpi_1\varpi_2 r_i + \varpi_3\varpi_4 r_i'} \cdot g_{15}^{-B_{1i}} \cdot \prod_{j\in[\varsigma]} (g^{\phi_1^2\phi_3\psi_j})^{\frac{-\breve{\partial}z_{15}(\vec{M}_t^{(i)}\cdot\vec{\delta})}{w_{\rho_t^\circ(i)} - w_j^{(\mu)}}}$$

$$T_{1i2} := g_{14}^{\mathsf{ut}_i} \cdot H_6\big(e(g,g_4)^{\breve{f}f'\cdot r_c} || \mathcal{P}_t^\circ || M_t^{(i)}\big) \cdot (\varrho_i^{\times})^{z_{13}} \cdot (g^{\phi_1})^{\varpi_1\varpi_2 r_i + \varpi_3\varpi_4 r_i'} \times \prod_{j\in[\varsigma]} (g^{\phi_1^2\phi_3\psi_j})^{\frac{-\breve{\partial}z_{15}(\vec{M}_t^{(i)}\cdot\vec{\delta})}{w_{\rho_t^\circ(i)} - w_j^{(\mu)}}}$$

$$T_{2i1} := \big(g_{11}^{w_{\rho_t^\circ(i)}} \cdot g_{12}\big)^{-r_i\varpi_1} \cdot g_{11}^{-B_{2i1}} \cdot (g^{\phi_2})^{\frac{-(z_{11}w_{\rho_t^\circ(i)} + z_{12})\breve{\partial}z_{15}(\vec{M}_t^{(i)}\cdot\vec{\delta})}{2\varpi_2}} \prod_{j\in[\varsigma]} \big(g^{\phi_1\phi_3\psi_j}\big)^{\frac{(z_{11}w_{\rho_t^\circ(i)} + z_{12})\breve{\partial}z_{15}(\vec{M}_t^{(i)}\cdot\vec{\delta})}{2\varpi_2(w_{\rho_t^\circ(i)} - w_j^{(\mu)})}} \prod_{(j,\iota)\in[\varsigma,\varsigma]} \big(g^{(\phi_1\phi_3)^2\psi_\iota/\psi_j}\big)^{\frac{\breve{\partial}z_{15}(\vec{M}_t^{(i)}\cdot\vec{\delta})}{2\varpi_2(w_{\rho_t^\circ(i)} - w_j^{(\mu)})}} \times \prod_{j\in[\varsigma]} \big(g^{\phi_2^2/\psi_j^2}\big)^{\frac{-\breve{\partial}z_{15}(\vec{M}_t^{(i)}\cdot\vec{\delta})(w_{\rho_t^\circ(i)} - w_j^{(\mu)})}{2\varpi_2}} \times \prod_{(j,\iota)\in[\varsigma,\varsigma], j\neq\iota} \big(g^{\phi_1\phi_2\phi_3\psi_\iota/\psi_j^2}\big)^{\frac{\breve{\partial}z_{15}(\vec{M}_t^{(i)}\cdot\vec{\delta})(w_{\rho_t^\circ(i)} - w_j^{(\mu)})}{2\varpi_2(w_{\rho_t^\circ(i)} - w_\iota^{(\mu)})}}$$

The components $T_{2i2}, T_{2i3}, T_{2i4}$ can be computed as T_{2i1} since these components have the term $(g_{11}^{w_{\rho_t^\circ(i)}} \cdot g_{12})$ in common and $\breve{r}_i, \breve{r}_i'$ have the same structure. Now, $\mathscr{C}$ generates the transform key $\mathcal{TK}_{A_d} := \langle A_d, T_1', T_2', \{T_{3,x}', T_{4,x}'\}_{x\in A_d}\rangle$, where $\widehat{r}, \widehat{r}_x \xleftarrow{\mathrm{u}} \mathbb{Z}_p^*$ and

$$T_1' := g^{\alpha'/\gamma_0} g_4^{r_c\cdot\widehat{r}} (g^{\phi_1})^{\breve{\tau}_0\breve{\partial}z_{15}/\gamma_0}, T_2' := g^{\widehat{r}}, T_{3,x}' := g^{\widehat{r}_x}, T_{4,x}' := (g_2^x g_1)^{\widehat{r}_x} g_3^{-\widehat{r}}$$

Lastly, $\mathscr{C}$ hands over $m\mathcal{DRR} := [\mathcal{TD}_{\mathcal{P}_t^\circ}, \mathcal{TK}_{A_d}]$ to $\mathscr{A}$. According to Remark 8, the generated data retrieval request for $(A_d, \mathcal{P}_t)$ is accuarately derived.

- $\mathcal{O}_{test}(\mathcal{CT}, \mathcal{P}_t) : \mathscr{A}$ submits a ciphertext $\mathcal{CT}$ and a keyword policy $\mathcal{P}_t$ with the condition that $\mathcal{P}_t(W_0^\star) = false \wedge \mathcal{P}_t(W_1^\star) = false$. Hence $\mathcal{P}_t(W_\mu^\star) = false$. Then, $\mathscr{C}$ chooses a decryption attribute set A_d, obtains $m\mathcal{DRR} := [\mathcal{TD}_{\mathcal{P}_t^\circ}, \mathcal{TK}_{A_d}] \leftarrow \mathcal{O}'''_{\mathcal{DRR}}(A_d, \mathcal{P}_t)$, and sends the outcome of $\mathsf{Test}(\mathcal{PP}, \mathcal{CT}, m\mathcal{DRR}, \mathcal{CSK})$ to $\mathscr{A}$. Note that the choice of A_d doest not affect the output of Test algorithm because it uses only $\mathcal{TD}_{\mathcal{P}_t^\circ}$ from $m\mathcal{DRR}$.

When this query phase is over, $\mathscr{A}$ submits an encryption policy $\mathcal{P}_e^\star$, a message $msg^\star$, and a signing policy $\mathcal{P}_s^\star$.

(4) To generate the challenge ciphertext

$$m\mathcal{CT}^\star := \begin{pmatrix} \Omega_s := \langle \mathcal{P}_s^\star, \sigma', \sigma'', \Gamma, \sigma, \{\sigma_i\}_{i\in[\ell_s]}, \chi\rangle, \\ \Omega_e := \langle \mathcal{P}_e^\star, ct, E', E_1, \{\vec{E}_i := (E_{i1}, E_{i2}, E_{i3}), \vec{\varepsilon}_i := (\varepsilon_{i1}, \varepsilon_{i2})\}_{i\in[\ell_e]}\rangle, \\ \Omega_k := \langle W_\mu^{\star\circ}, k_T, \{\vec{K}_j := (K_{j1}, K_{j2}), \vec{k}_j^{\times} := (k_{j1}^{\times}, k_{j2}^{\times}, k_{j3}^{\times}, k_{j4})^{\times}, u_j\}_{j\in[\varsigma]}\rangle, \\ \mathsf{tag2}, E_2, \varepsilon_2, \eta \end{pmatrix}$$

of the message $msg^\star$ for $\mathcal{P}_s^\star := (\mathbf{M}_s, \rho_s)$, $\mathcal{P}_e^\star := (\mathbf{M}_e, \rho_e)$, where $\mathbf{M}_s$ (resp. $\mathbf{M}_e$) is a matrix of size $\ell_s \times n_s$ (resp. $\ell_e \times n_e$), and the keyword set $W_\mu^\star := \{[\mathcal{W}_1 : w_1^{(\mu)}], \ldots, [\mathcal{W}_\varsigma : w_\varsigma^{(\mu)}]\}$, $\mathscr{C}$ samples $\theta, \delta', \delta'', \delta''', \breve{r}, o_2, \eta, t_i, , \pi_{j1}, \pi_{j2}, \chi, \varepsilon_2, \varepsilon_{i1}, \varepsilon_{i2}, u_j, \tilde{o}_2, \ldots, \tilde{o}_{\ell_e} \xleftarrow{\mathrm{u}} \mathbb{Z}_p^*$, implicitly defines $\beta := \phi_3, \breve{t}_j := \psi_j$ and sets

$$\begin{aligned} &E_1 := g^{\phi_3}, \Gamma := H_7(g_T^{1/\theta}) \oplus H_8\big(e(g, g_4)^{\delta'\cdot\delta''\cdot r_c}\big), \\ &ct := msg^\star \oplus \mathsf{KDF}\big(e(g^{\phi_2}, E_1)^{\alpha'} \cdot g_T^{1/\theta}\big), \\ &\mathsf{tag2} := H_3\big(H_2(e(g^{\phi_2}, E_1)^{\alpha'} \cdot g_T^{1/\theta})||ct\big), \sigma' := g^{\delta'}, \sigma'' := g_4^{\delta''}, \\ &\sigma := (g^{\phi_2})^{\frac{\alpha'}{\theta}} g_4^{\frac{\breve{r}}{\theta}} E_1^{\breve{o}_1 z_5 + z_6} \big(\textstyle\prod_{i\in[\ell_s]} H_1\big(\rho_s(i)\big)^{\frac{\breve{r}a_i}{\theta} + o_2 b_i}\big) g_5^{-\chi}, \sigma_i := g^{\frac{\breve{r}a_i}{\theta} + o_2 b_i}, E' := g_4^{\delta'''}, \\ &E_{i1} := E_1^{z_4 \partial M_{e1}^{(i)}} g_4^{\sum_{j=2}^{\ell_e} \tilde{o}_j M_{ej}^{(i)}} g_3^{t_i} \cdot g_4^{-\varepsilon_{i1}}, E_{i2} := (g_2^{\rho_e(i)} g_1)^{t_i} \cdot g_2^{-\varepsilon_{i2}}, E_{i3} := g^{t_i}, \end{aligned}$$

$$\begin{aligned} &K_{j1} := (g^{\psi_j})^{z_{11} w_j^{(\mu)} + z_{12}} \cdot g_{11}^{-u_j} \cdot \textstyle\prod_{\iota\in[\varsigma]} \big(g^{\phi_2\psi_j/\psi_\iota^2}\big)^{(w_j^{(\mu)} - w_\iota^{(\mu)})} \cdot \prod_{\iota\in[\varsigma], \iota\neq j} g^{\phi_1\phi_3\psi_j/\psi_\iota}, \\ &K_{j2} := K_{j1} \cdot E_1^{-z_{13}}, \\ &k_{j1}^{\times} := (g^{\psi_j})^{\varpi_1} g^{-\pi_{j1}\varpi_1}, k_{j2}^{\times} := g^{\pi_{j1}\varpi_2}, k_{j3}^{\times} := (g^{\psi_j})^{\varpi_3} g^{-\pi_{j2}\varpi_3}, k_{j4}^{\times} := g^{\pi_{j2}\varpi_4}, \\ &k_T := e(E_1, g_{15})^{\tau_0} \cdot Z^{z_{15}}, E_2 := E_1^{z_7\xi + z_8\eta + z_9} \cdot g_7^{-\varepsilon_2} \end{aligned}$$

where $\partial := H_5\big(e(g, g_4)^{\delta'\delta'''\cdot r_c}\big)$, $\lambda_i := (\phi_3\partial, \tilde{o}_2, \ldots, \tilde{o}_{\ell_e}) \cdot \vec{M}_e^{(i)} = \phi_3\partial M_{e1}^{(i)} + \sum_{j=2}^{\ell_e} \tilde{o}_j M_{ej}^{(i)}$ is the ith share of $\beta\partial := \phi_3\partial$ with respect to $(\mathbf{M}_e, \rho_e)$, $\breve{o}_1 := H_4(ct||\Gamma||E'||\mathcal{P}_e^\star||\mathcal{P}_s^\star||W_\mu^{\star\circ})$, $\xi := H_5(\Omega_s||\Omega_e||\Omega_k||\mathsf{tag2})$, $(a_1, a_2, \ldots, a_{\ell_s})$ and $(b_1, b_2, \ldots, b_{\ell_s})$ are vectors satis-

fying respectively $\sum_{i\in[\ell_s]} a_i \cdot \vec{M}_s^{(i)} = \vec{1}_{n_s}$ and $\sum_{i\in[\ell_s]} b_i \cdot \vec{M}_s^{(i)} = \vec{0}_{n_s}$. Lastly, $\mathscr{C}$ hands over the challenge ciphertext to $\mathscr{A}$.

(5) In this step, $\mathscr{A}$ queries the oracles similar to step (3). Once this query phase is over, $\mathscr{A}$ makes a guess μ'.

If $\mu' = \mu$, $\mathscr{A}$ wins, and in this case, $Z = e(g,g)^{\phi_1\phi_2\phi_3}$; otherwise, $\mathscr{C}$ claims that Z is a random element of $\mathbb{G}_T$.

If $Z = e(g,g)^{\phi_1\phi_2\phi_3}$, then $k_T := e(E_1, g_{15})^{\tau_0} \cdot Z^{z_{15}} = e(g, g_{15})^{\tau\beta}$ and hence $\mathscr{A}$'s view is identical to the original game G_{real}. On the other hand, if Z is a random element then k_T is a random element as well and hence $\mathscr{A}$'s view is identical to the game G_0. Therefore, if $\mathscr{A}$ can distinguish between G_{real} and G_0 with non-negligible advantage, $\mathscr{C}$ has a non-negligible advantage in solving q-2 problem. (*of Claim 1)*

Claim 2. *Under the hardness assumption of DLin problem, no PPT Type-1 adversary can distinguish between the games* G_l *and* G_{l+1} *with non-negligible advantage,* $l \in \{0, 1, \ldots, \varsigma - 1\}$.

Proof. Let a Type-1 adversary $\mathscr{A}$ distinguishes between the games G_l and G_{l+1} with non-negligible advantage, then we construct a challenger $\mathscr{C}$ that solves DLin problem with non-negligible advantage by interacting with $\mathscr{A}$ as follows. Given the DLin problem instance $\langle \Sigma, g, g^{\phi_1}, g^{\phi_2}, g^{\phi_1\phi_3}, g^{\phi_2\phi_4}, Z\rangle$, the task for $\mathscr{C}$ is to determine whether $Z = g^{\phi_3+\phi_4}$ or Z is a random element of $\mathbb{G}$. As in [8], we write DLin problem as $\langle \Sigma, g, g^{\phi_1}, g^{\phi_2}, g^{\phi_1\phi_3}, \mathcal{D}, g^{\psi}\rangle$ for ψ such that $g^{\psi} = Z$, and consider the task of deciding whether $\mathcal{D} = g^{\phi_2(\psi-\phi_3)}$.

(1) $\mathscr{A}$ submits two challenge keyword sets $W_0^\star := \{[\mathcal{W}_1 : w_1^{(0)}], \ldots, [\mathcal{W}_\varsigma : w_\varsigma^{(0)}]\}$ and $W_1^\star := \{[\mathcal{W}_1 : w_1^{(1)}], \ldots, [\mathcal{W}_\varsigma : w_\varsigma^{(1)}]\}$. Note that these two sets satisfy the conditions $|W_0^\star| = |W_1^\star| = \varsigma$ and $W_0^{\star\circ} = W_1^{\star\circ} = \{\mathcal{W}_1, \ldots, \mathcal{W}_\varsigma\}$ of $\mathsf{Game}^{\mathsf{IND\text{-}CKA}}_{\mathsf{Type\text{-}1}}$.

(2) Firstly, $\mathscr{C}$ samples $\mu \xleftarrow{\text{u}} \{0,1\}$ and sets $W_\mu^\star := \{[\mathcal{W}_1 : w_1^{(\mu)}], \ldots, [\mathcal{W}_\varsigma : w_\varsigma^{(\mu)}]\}$. Then, $\mathscr{C}$ selects $\alpha, \tau, \varpi_3, \varpi_4, r_c, z_i \xleftarrow{\text{u}} \mathbb{Z}_p^*, i \in \{1, \ldots, 10, 12, \ldots, 15\}$, sets $g_T := e(g,g)^\alpha, g_i := g^{z_i}$ for $i \in \{1, \ldots, 10, 13, 14, 15\}$, $g_{11} := (g^{\phi_2})^\tau$, $g_{12} := g^{z_{12}}(g^{\phi_2})^{-\tau w_l^{(\mu)}}$, $g_{16} := g^{\phi_2}, g_{17} := g^{\phi_1}, g_{18} := g^{\varpi_3}, g_{19} := g^{\varpi_4}$, $h_T := e(g, g_{15})^\tau, Y_c := g_4^{r_c}$. Next, it chooses eight collision-resistant hash functions $\{H_i\}_{i=1}^8$ (as described in the construction) and sets $\mathcal{PP} := \langle \Sigma, g_T, g, \{g_i\}_{i=1}^{15}, \mathcal{M}, \mathsf{KDF}, \{H_i\}_{i=1}^8\rangle$, $\mathcal{MK} := g^\alpha$, $\mathcal{TPK} := \langle h_T, g_{16}, g_{17}, g_{18}, g_{19}\rangle$, $\mathcal{TSK} := \langle \tau, \phi_2, \phi_1, \varpi_3, \varpi_4\rangle$, $\mathcal{CPK} := Y_c$, $\mathcal{CSK} := r_c$. Note that $\varpi_1 := \phi_2, \varpi_2 := \phi_1$ are implicitly defined. $\mathscr{C}$ sends $[\mathcal{PP}, \mathcal{TPK}, \mathcal{CPK}, \mathcal{CSK}]$ to $\mathscr{A}$.

(3) $\mathscr{C}$ simulates $\mathscr{A}$'s queries as follows.

- $\mathcal{O}_{\mathcal{SKG}}(A_s) : \mathscr{C}$ returns $\mathcal{SK}_{A_s} \leftarrow \mathsf{sKeyGen}(\mathcal{PP}, \mathcal{MK}, A_s)$ since $\mathscr{C}$ knows $\mathcal{MK}$.
- $\mathcal{O}'''_{\mathcal{DRR}}(A_d, \mathcal{P}_t)$: $\mathscr{A}$ submits a decryption attribute set A_d and a keyword policy $\mathcal{P}_t$ with the condition that $\mathcal{P}_t(W_0^\star) = false \wedge \mathcal{P}_t(W_1^\star) = false$. Hence $\mathcal{P}_t(W_\mu^\star) = false$. Let $\mathcal{P}_t := (\mathbf{M}_t, \rho_t^\circ, \{w_{\rho_t^\circ(i)}\}_{i\in[\ell_t]})$, where $\mathbf{M}_t$ represents a matrix of dimesion $\ell_t \times n_t$. $\mathscr{C}$ calculates $\mathcal{TD}_{\mathcal{P}_t^\circ} := \langle \mathcal{P}_t^\circ, T_1, T_2, \{\varrho_i^\times, \vec{T}_{1i}, \vec{T}_{2i}, B_{1i}, \vec{B}_{2i}\}_{i\in[\ell_t]}\rangle$ (mentioned in Equation (3.9)) as described below. Choose $\breve{f}, f', \breve{\tau}, t_r, r_i, r'_i, B_{1i}, B_{2i1}, B_{2i2}, B_{2i3}, B_{2i4}, \breve{o}_2, \ldots, \breve{o}_{n_t}, \breve{o}'_2, \ldots, \breve{o}'_{n_t} \xleftarrow{\text{u}} \mathbb{Z}_p^*$, $i \in [\ell_t]$, implicitly define

$$\breve{r}_i := \frac{r_i\tau(w_{\rho_t^\circ(i)} - w_l^{(\mu)})}{\phi_2\tau(w_{\rho_t^\circ(i)} - w_l^{(\mu)}) + z_{12}} \text{ and } \breve{r}'_i := r'_i + \frac{z_{12}\phi_1 r_i}{\varpi_3\varpi_4\big(\phi_2\tau(w_{\rho_t^\circ(i)} - w_l^{(\mu)}) + z_{12}\big)},$$

and set $\breve{\partial} := H_5\big(e(g,g_4)^{\breve{f}f'\cdot r_c}\big), \vartheta_i := \vec{M}_t^{(i)} \cdot (\tau\breve{\partial}, \breve{o}_2, \ldots, \breve{o}_{n_t}), \breve{\vartheta}_i := \vec{M}_t^{(i)} \cdot (\breve{\tau}/t_r, \breve{o}'_2, \ldots, \breve{o}'_{n_t})$, $T_1 := g^{\breve{f}}, T_2 := g_4^{f'}$,

$$\begin{aligned}
\varrho_i^\times &:= (g^{\phi_1})^{r_i} g^{\varpi_3\varpi_4 r'_i},\\
T_{1i1} &:= g_{15}^{\vartheta_i} \cdot (g^{\phi_1})^{r_i z_{10}} g^{\varpi_3\varpi_4 r'_i z_{10}} \cdot g_{15}^{-B_{1i}},\\
T_{1i2} &:= g_{14}^{\breve{\vartheta}_i} \cdot H_6\big(e(g,g_4)^{\breve{f}f'\cdot r_c} || \mathcal{P}_t^\circ || M_t^{(i)}\big) \cdot (g^{\phi_1})^{r_i z_{13}} g^{\varpi_3\varpi_4 r'_i z_{13}},\\
T_{2i1} &:= (g^{\phi_2})^{-r_i\tau(w_{\rho_t^\circ(i)} - w_l^{(\mu)})} \cdot g_{11}^{-B_{2i1}},\\
T_{2i2} &:= (g^{\phi_1})^{-r_i\tau(w_{\rho_t^\circ(i)} - w_l^{(\mu)})} \cdot g_{11}^{-B_{2i2}},\\
T_{2i3} &:= \big(g_{11}^{w_{\rho_t^\circ(i)}} \cdot g_{12}\big)^{-r'_i\varpi_3} (g^{\phi_1})^{-z_{12}r_i/\varpi_4} \cdot g_{11}^{-B_{2i3}},\\
T_{2i4} &:= \big(g_{11}^{w_{\rho_t^\circ(i)}} \cdot g_{12}\big)^{-r'_i\varpi_4} (g^{\phi_1})^{-z_{12}r_i/\varpi_3} \cdot g_{11}^{-B_{2i4}}.
\end{aligned}$$

Next, $\mathscr{C}$ generates the transform key $\mathcal{TK}_{A_d} := \langle A_d, T'_1, T'_2, \{T'_{3,x}, T'_{4,x}\}_{x\in A_d}\rangle$, where $\gamma, \widehat{r}, \widehat{r}_x \xleftarrow{\text{u}} \mathbb{Z}_p^*$ and $T'_1 := g^{\alpha/\gamma} g_4^{(r_c\cdot\widehat{r})} g_{14}^{\breve{r}/\gamma}, T'_2 := g^{\widehat{r}}, T'_{3,x} := g^{\widehat{r}_x}$, $T'_{4,x} := (g_2^x g_1)^{\widehat{r}_x} g_3^{-\widehat{r}}$. Finally, $\mathscr{C}$ sends $m\mathcal{DRR} := [\mathcal{TD}_{\mathcal{P}_t^\circ}, \mathcal{TK}_{A_d}]$ to $\mathscr{A}$. According to Remark 8, the generated data retrieval request for $(A_d, \mathcal{P}_t)$ is accurately derived.

- $\mathcal{O}_{test}(\mathcal{CT}, \mathcal{P}_t)$: The simulation is similar to that of Claim 1.

Once the query phase is complete, $\mathscr{A}$ outputs an encryption policy $\mathcal{P}_e^\star$, a message $msg^\star$, and a signing policy $\mathcal{P}_s^\star$.

(4) To compute the challenge ciphertext

$$m\mathcal{CT}^{\star} := \begin{pmatrix} \Omega_s := \langle \mathcal{P}_s^{\star}, \sigma', \sigma'', \Gamma, \sigma, \{\sigma_i\}_{i\in[\ell_s]}, \chi\rangle, \\ \Omega_e := \langle \mathcal{P}_e^{\star}, ct, E', E_1, \{\vec{E}_i := (E_{i1}, E_{i2}, E_{i3}), \vec{\varepsilon}_i := (\varepsilon_{i1}, \varepsilon_{i2})\}_{i\in[\ell_e]}\rangle, \\ \Omega_k := \langle W_{\mu}^{\star\circ}, k_T, \{\vec{K}_j := (K_{j1}, K_{j2}), \vec{k}_j^{\times} := (k_{j1}^{\times}, k_{j2}^{\times}, k_{j3}^{\times}, k_{j4})^{\times}, u_j\}_{j\in[\varsigma]}\rangle, \\ \mathsf{tag2}, E_2, \varepsilon_2, \eta \end{pmatrix}$$

of the message $msg^{\star}$ for $\mathcal{P}_s^{\star} := (\mathbf{M}_s, \rho_s)$, $\mathcal{P}_e^{\star} := (\mathbf{M}_e, \rho_e)$, where $\mathbf{M}_s$ (resp. $\mathbf{M}_e$) is a matrix of size $\ell_s \times n_s$ (resp. $\ell_e \times n_e$), and the keyword set $W_{\mu}^{\star} := \{[\mathcal{W}_1 : w_1^{(\mu)}], \ldots, [\mathcal{W}_{\varsigma} : w_{\varsigma}^{(\mu)}]\}$, $\mathscr{C}$ calculates the components of Ω_k in the following manner.

For $j \in [\varsigma]$ and $j \neq l$, $\mathscr{C}$ picks $\beta, \breve{t}_j, u_j, \pi_{j1}, \pi_{j2} \xleftarrow{\text{u}} \mathbb{Z}_p^*$, and sets
$k_T := h_T^{\beta}, K_{j1} := (g_{11}^{w_j^{(\mu)}} g_{12})^{\breve{t}_j} g_{10}^{-\beta} \cdot g_{11}^{-u_j}, K_{j2} := (g_{11}^{w_j^{(\mu)}} g_{12})^{\breve{t}_j} g_{13}^{-\beta} \cdot g_{11}^{-u_j}$,
$k_{j1}^{\times} := g_{16}^{\breve{t}_j - \pi_{j1}}, k_{j2}^{\times} := g_{17}^{\pi_{j1}}, k_{j3}^{\times} := g_{18}^{\breve{t}_j - \pi_{j2}}, k_{j4}^{\times} := g_{19}^{\pi_{j2}}$.

For $j = l$, $\mathscr{C}$ implicitly defines $\breve{t}_l := \psi, \pi_{l1} := \phi_3$, chooses $u_l, \pi_{l2} \xleftarrow{\text{u}} \mathbb{Z}_p^*$, and sets
$K_{l1} := (g^{\psi})^{z_{12}} g_{10}^{-\beta} \cdot g_{11}^{-u_l}, K_{l2} := (g^{\psi})^{z_{12}} g_{13}^{-\beta} \cdot g_{11}^{-u_l}, k_{l1}^{\times} := \mathcal{D}, k_{l2}^{\times} := g^{\phi_1\phi_3}$,
$k_{l3}^{\times} := (g^{\psi})^{\varpi_3} g_{18}^{-\pi_{l2}}, k_{l4}^{\times} := g_{19}^{\pi_{l2}}$. $\mathscr{C}$ computes the other components of $m\mathcal{CT}^{\star}$ as in Equation (3.1). $\mathscr{A}$ receives this $m\mathcal{CT}^{\star}$.

(5) Again, $\mathscr{A}$ queries similar to described in step (3). Once this query phase is over, $\mathscr{A}$ makes a guess μ'.

If $\mu' = \mu$, $\mathscr{A}$ wins and $\mathscr{C}$ will claim that $\mathcal{D} = g^{\phi_2(\psi - \phi_3)}$; otherwise, $\mathscr{C}$ claims that $\mathcal{D}$ is a random element of $\mathbb{G}$.

If $\mathcal{D} = g^{\phi_2(\psi-\phi_3)}$, then $k_{l1}^{\times} := g_{16}^{\breve{t}_l - \pi_{l1}}, k_{l2}^{\times} := g_{17}^{\pi_{l1}}$ and hence $\mathscr{A}$'s view is identical to G_l. On the other hand, if $\mathcal{D}$ is a random element of $\mathbb{G}$, then $\mathscr{A}$'s view is identical to G_{l+1}. Therefore, if $\mathscr{A}$ can distinguish between G_l and G_{l+1} with non-negligible advantage, $\mathscr{C}$ has a non-negligible advantage in solving DLin problem. *(of Claim 2)*

Claim 3. *There is no PPT Type-1 adversary that can distinguish between the games G_l and G_{l+1}, $l \in \{\varsigma, \varsigma+1, \ldots, 2\varsigma-1\}$, with non-negligible advantage, under the assumption that DLin problem is hard.*

Proof. The proof is almost identically to that of Claim 2, except where the simulation is done over g_{18} and g_{19} instead of g_{16} and g_{17}. *(of Claim 3)*

This completes the proof of Lemma 3. □

Lemma 4. *ABDSRS demonstrates IND-CKA security against PPT Type-2 adversary assuming the hardness of the DBDH problem.*

Proof. Assume, a PPT Type-2 adversary $\mathscr{A}$ breaks IND-CKA security (modeled as a game $\mathsf{Game}^{\mathsf{IND\text{-}CKA}}_{\mathsf{Type\text{-}2}}$ in Section 3.2.2) of our ABDSRS with non-negligible advantage, then, a challenger $\mathscr{C}$ can solve DBDH problem by communicating with $\mathscr{A}$. Given the DBDH problem instance $\langle \Sigma, g, G_1, G_2, G_3, Z \rangle$, where $G_1 := g^{\phi_1}, G_2 := g^{\phi_2}, G_3 := g^{\phi_3}$ (note that ϕ_1, ϕ_2, ϕ_3 are unknown to $\mathscr{C}$), the task for $\mathscr{C}$ is to ascertain if Z is equal to $e(g,g)^{\phi_1\phi_2\phi_3}$ or Z has been randomly selected from $\mathbb{G}_T$.

(1) This step is identical to the step (1) of Lemma 2.

(2) In this step, firstly $\mathscr{A}$ queries signing key generation oracle $\mathcal{O}_{\mathcal{SKG}}(A_s)$, data retrieval request generation oracle $\mathcal{O}'_{\mathcal{DRR}}(A_d, \mathcal{P}_t)$ and test oracle $\mathcal{O}'_{test}(\mathcal{CT}, \mathcal{P}_t)$. $\mathscr{C}$ answers as described subsequently.

- $\mathcal{O}_{\mathcal{SKG}}(A_s) : \mathscr{C}$ returns $\mathcal{SK}_{A_s} \leftarrow \mathsf{sKeyGen}(\mathcal{PP}, \mathcal{MK}, A_s)$ since $\mathscr{C}$ knows $\mathcal{MK}$.
- $\mathcal{O}'_{\mathcal{DRR}}(A_d, \mathcal{P}_t) : \mathscr{C}$ generates the trapdoor $\widetilde{\mathcal{TD}}_{\mathcal{P}_t} := \langle \mathcal{P}_t, T_0, T_1, T_2, \{\varrho_i, \vec{\widetilde{T}}_{1i} := (\widetilde{T}_{1i1}, \widetilde{T}_{1i2}), \vec{T}_{2i} := (T_{2i1}, T_{2i2}, T_{2i3}, T_{2i4}), B_{1i}, \vec{B}_{2i} := (B_{2i1}, B_{2i2}, B_{2i3}, B_{2i4})\}_{i\in[\ell_t]} \rangle$ as follows. Choose $f, \breve{r}_i, \breve{r}'_i, B_{1i}, B_{2i1}, B_{2i2}, B_{2i3}, B_{2i4}, v_2, \ldots, v_{n_t} \xleftarrow{\text{u}} \mathbb{Z}^*_p$ and compute

$$
\begin{aligned}
&T_0 := g^f, T_1 := G_2, T_2 := G_3^{z_4}, \varrho_i := \varpi_1\varpi_2\breve{r}_i + \varpi_3\varpi_4\breve{r}'_i - f,\\
&\widetilde{T}_{1i1} := g_{15}^{\vec{M}_t^{(i)}\cdot(\tau H_5(Z^{z_4}), v_2, \ldots, v_{n_t})} \cdot g_{10}^{\varpi_1\varpi_2\breve{r}_i + \varpi_3\varpi_4\breve{r}'_i} \cdot g_{15}^{-B_{1i}},\\
&\widetilde{T}_{1i2} := H_6\big(Z^{z_4} || \mathcal{P}_t^\circ || M_t^{(i)}\big) \cdot g_{13}^{\varpi_1\varpi_2\breve{r}_i + \varpi_3\varpi_4\breve{r}'_i},\\
&T_{2i1} := \big(g_{11}^{w_{\rho_t^\circ(i)}} \cdot g_{12}\big)^{-\breve{r}_i\varpi_1} \cdot g_{11}^{-B_{2i1}}, T_{2i2} := \big(g_{11}^{w_{\rho_t^\circ(i)}} \cdot g_{12}\big)^{-\breve{r}_i\varpi_2} \cdot g_{11}^{-B_{2i2}},\\
&T_{2i3} := \big(g_{11}^{w_{\rho_t^\circ(i)}} \cdot g_{12}\big)^{-\breve{r}'_i\varpi_3} \cdot g_{11}^{-B_{2i3}}, T_{2i4} := \big(g_{11}^{w_{\rho_t^\circ(i)}} \cdot g_{12}\big)^{-\breve{r}'_i\varpi_4} \cdot g_{11}^{-B_{2i4}},
\end{aligned}
$$

  where $\mathcal{P}_t := (\mathbf{M}_t, \rho_t^\circ, \{w_{\rho_t^\circ(i)}\}_{i\in[\ell_t]})$ and $\vartheta_i := \vec{M}_t^{(i)} \cdot (\tau H_5(Z^{z_4}), v_2, \ldots, v_{n_t})$ is the ith share of $\tau H_5(Z^{z_4})$. Since $\mathscr{C}$ knows $\mathcal{MK}$, it computes the decryption key $\mathcal{DK}_{A_d} \leftarrow \mathsf{dKeyGen}(\mathcal{PP}, \mathcal{MK}, \mathcal{CPK}, A_d)$ and sends the tuple $[\mathcal{DK}_{A_d}, \widetilde{\mathcal{TD}}_{\mathcal{P}_t}]$ to $\mathscr{A}$.
- $\mathcal{O}'_{test}(\mathcal{CT}, \mathcal{P}_t) : \mathscr{C}$ computes the trapdoor $\widetilde{\mathcal{TD}}_{\mathcal{P}_t}$ as in the simulation of $\mathcal{O}'_{\mathcal{DRR}}$, and sends the outcome of $\mathsf{Test}(\mathcal{PP}, \mathcal{CT}, \widetilde{\mathcal{TD}}_{\mathcal{P}_t}, \mathcal{CSK})$ to $\mathscr{A}$.

When this query phase is over, $\mathscr{A}$ sends to $\mathscr{C}$ a message $msg^\star$, an encryption policy $\mathcal{P}_e^\star$, a signing policy $\mathcal{P}_s^\star$ and two keyword sets $W_0^\star := \{[\mathcal{W}_1 : w_1^{(0)}], \ldots, [\mathcal{W}_\varsigma : w_\varsigma^{(0)}]\}$ and $W_1^\star := \{[\mathcal{W}_1 : w_1^{(1)}], \ldots, [\mathcal{W}_\varsigma : w_\varsigma^{(1)}]\}$.

(3) $\mathscr{C}$ selects $\mu \xleftarrow{\text{u}} \{0,1\}$, and a singing attribute set A_s such that $\mathcal{P}_s^\star(A_s) = true$. Next, it obtains $\mathcal{CT}^\star \leftarrow \mathsf{onSigncrypt}(\mathcal{PP}, \mathcal{IC}, msg^\star, \mathcal{P}_e^\star, W_\mu^\star)$,

where $\mathcal{IC} \leftarrow \mathsf{offSigncrypt}(\mathcal{PP}, \mathcal{SK}_{A_s}, \mathcal{TPK}, \mathcal{CPK}, \mathcal{P}_s^{\star})$,
$\mathcal{SK}_{A_s} \leftarrow \mathsf{sKeyGen}(\mathcal{PP}, \mathcal{MK}, A_s)$, and sends the challenge ciphertext $\mathcal{CT}^{\star}$ to $\mathscr{A}$. This is possible because $\mathscr{C}$ is having the system master secret key $\mathcal{MK}$ and TGC's secret key $\mathcal{TSK}$.

(4) $\mathscr{A}$ queries like step (2), except that $\mathscr{A}$ is not permitted to query test oracle using the input $(\mathcal{CT}^{\star}, \mathcal{P}_t)$ such that $\mathcal{P}_t(W_{\mu}^{\star}) = true$. $\mathscr{C}$ responds as in step (2). When this query phase is over, $\mathscr{A}$ makes a guess μ'.

If $\mu' = \mu$, $\mathscr{A}$ wins the game and $\mathscr{C}$ will claim that Z is equal to $e(g,g)^{\phi_1\phi_2\phi_3}$; otherwise, $\mathscr{C}$ claims that Z is a random element of $\mathbb{G}_T$.

If $Z = e(g,g)^{\phi_1\phi_2\phi_3}$, the challenge ciphertext $\mathcal{CT}^{\star}$ is properly simulated. If Z is randomly chosen from $\mathbb{G}_T$, then $\mathcal{CT}^{\star}$ does not depend on μ in $\mathscr{A}$'s perspective; resulting in $\mathscr{A}$'s advantage is 0. As a result, if $\mathscr{A}$ has a non-negligible advantage in winning the game, $\mathscr{C}$ has a non-negligible advantage in solving DBDH problem. □

We derive the subsequent theorem by combining Lemma 3 with Lemma 4.

Theorem 5 (Chosen Keyword set Attack Security)**.** *Suppose the challenge keyword set has at most q keywords. Then, ABDSRS is IND-CKA secure, under the assumption that q-2, DLin and DBDH problems are hard.*

3.5 Performance

This section, along with all of the tables and figures, makes use of the notations described below.

ℓ_e (resp. ℓ_s)	:	total number of attributes in an encryption (resp. signing) policy
$\|W\|$ or ς	:	size of a keyword set ascribed to a ciphertext
M_G (resp. E_G)	:	one multiplication (resp. exponentiation) execution time in $\mathbb{G}$
$\|A_d\|$	:	number of DU's attributes
M_T (resp. E_T)	:	one multiplication (resp. exponentiation) execution time on $\mathbb{G}_T$ element
I_T	:	one inversion execution time on $\mathbb{G}_T$ element
n	:	number of attributes in encryption attribute universe
P	:	one pairing computation execution time
ℓ_t	:	total number of keywords within a trapdoor/trapdoor keyword policy

H	:	one hash function calculation execution time
$\|A_s\|$	:	number of DO's attributes
L (resp. L_T, L_p)	:	size of an element of $\mathbb{G}$ (resp, $\mathbb{G}_T$, $\mathbb{Z}_p$)
n_m	:	maximum number of possible values for each encryption attribute
$\|sig\|$	:	size of an identity-based signature
T_{veri}	:	one signature verification time
$\|c_s\|$	:	size of a commitment
L_H	:	hash function's output length
S_p (resp. M_p, I_p)	:	one subtraction/addition (resp. multiplication, inversion) execution time in $\mathbb{Z}_p$
T_{kdf}	:	execution time of KDF
T_{open}	:	one commitment verification time

As already mentioned, the appealing features−(i) provably secure, (ii) lightweight design, (iii) fine-grained data access control, (iv) data and DO authenticity, (v) DO anonymity, (vi) keyword policy search over encrypted data, (vii) keyword privacy, and (viii) search results verification− provided by ABDSRS have not been taken into account concurrently in previous literature studies. Next, we conduct a comparison between ABDSRS and the following attribute-based cryptosystems:

1) Attribute-based online-offline signcryption scheme (ABOOSC) [69]
2) ABSC with outsourced unsigncryption [14, 2]
3) ABSE supporting single keyword search mechanism [88, 64, 63]
4) ABSE supporting multi-keyword search framework [1, 62]
5) ABSE supporting keyword policy search [12]

Table 3.3: Functionality features comparison

	[69]	[14]	[2]	[88]	[12]	[64]	[63]	[62]	[1]	ABDSRS
Fine-grained data access control	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Data and DO authenticity	✓	✓	✓							✓
Do anonymity	✓	✓	✓							✓
Online-offline signcryption/encryption	✓						✓		✓	✓
Outsourced unsigncryption/decryption		✓	✓	✓			✓	✓	✓	✓
Verification of transformed ciphertext		✓	✓	✓				✓	✓	✓
Boolean formula search over encrypted data					✓					✓
Keyword privacy				✓	✓	✓	✓			✓
Secure against KGAs										✓
Non-interactive search results verification					✓				✓	✓

Table 3.3 summarizes the functionality comparison. We theoretically compare our ABDSRS with the ABSE sechemes [88, 64, 63, 1, 62] regarding the computation

Table 3.4: Computation costs comparison

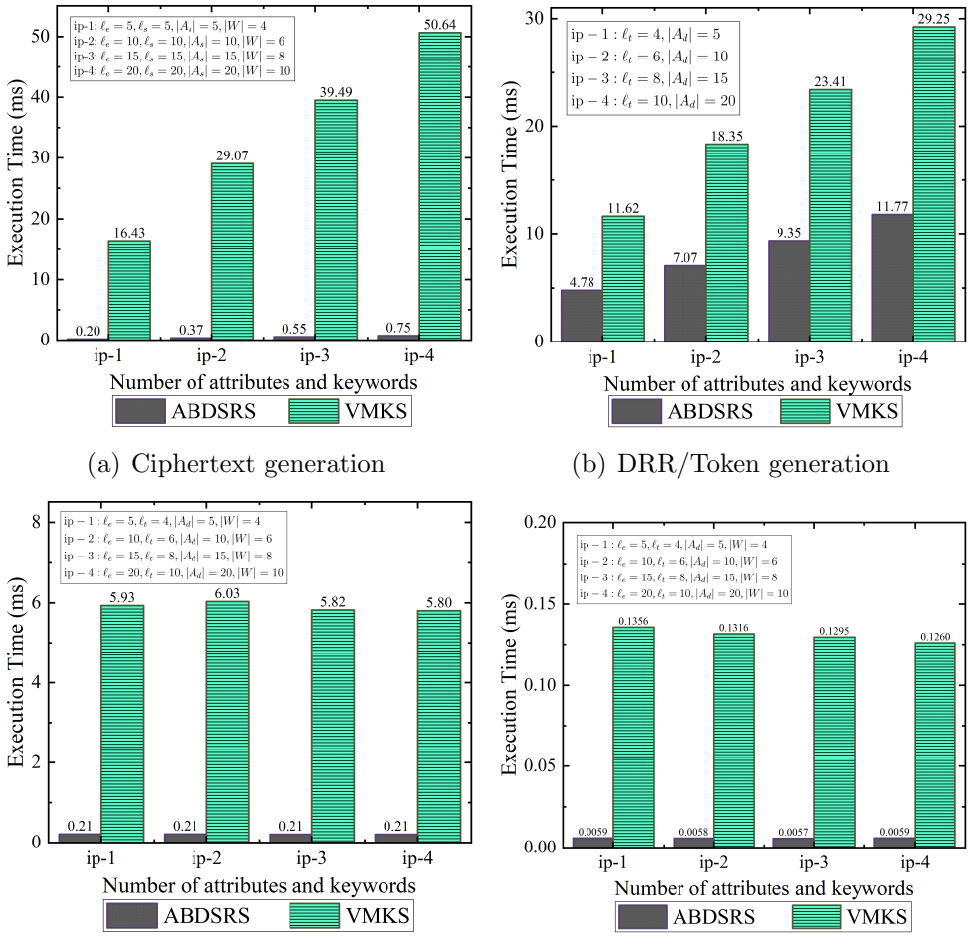
Scheme	Ciphertext Generation	DRR/Token Generation	Search Results Verification	Decryption
[88]	$4E_G + 3E_T + O(\ell_e)M_p + O(\ell_e)I_p$	$8M_p + 5I_p + H$	-	$2M_T + 2I_T + E_T + H$
[64]	$O(n_m \cdot n)E_G + 3E_T$	$O(n)E_G$	-	$O(1)E_G + O(1)E_T + 3P + H$
[63]	$O(\ell_e \cdot \varsigma)E_G + O(\ell_e)S_p + O(\ell_e)M_p$	$O(\|A_d\|)E_G + O(\|A_d\|)M_G$	-	$M_T + I_T + E_T$
[62]	$O(v + \varsigma)E_G + 2E_T + P + H$	$3E_G + O(\ell_t)S_p + \ell_t H$	$O(1)E_G + O(1)M_G + 2P + 2H$	$2M_T + I_T + P$
[1]	$O(\ell_e)M_G + O(\ell_e)E_G$	$\|A_d\|M_G + 5E_G$ $\ell_t E_T + \ell_t H$	$E_T + I_T + M_T +$ $T_{veri} + T_{open} + H$	$E_T + I_T + M_T + 2H$
ABDSRS	$O(\ell_e + \varsigma)S_p + O(\ell_e + \varsigma)M_p + 2H$	$\ell_t M_G$	$2M_T + 2E_T + 2H$	T_{kdf}

Table 3.5: Communication and storage costs comparison

Scheme	Public Params. Size	Signing Key Size	Decryption Key Size	Ciphertext (CT) Size	DRR/Token Size	Transformed CT Size
[88]	$5L + 2L_T$	-	$O(\|A_d\|)L +$ $L_T + 4L_p$	$4L + 2L_T + 2\ell_e L_p$	$\|A_d\|L + L_T + 7L_p$	$3L_T$
[64]	$O(n_m \cdot n)L + L_T$	-	$O(n)L + L_T$	$O(n_m \cdot n)L + 3L_T$	$O(n)L + L_p$	$2L + L_T$
[63]	$O(n)L + L_T$	-	$O(\|A_d)L + 3L_p$	$O(n)L + 2L_T + O(n)L_p$	$2L$	$2L_T$
[62]	$O(n)L + L_T$	-	$(\|A_d\| + 6)L + 4L_p$	$O(v + \varsigma)L + 2L_T$	$2L + L_p$	$L + 2L_T$
[1]	$O(n)L + 2L_T$	-	$\|A_d\|L$	$(2\varsigma + 2\ell_e)L + 2L_T$	$(\|A_d\| + 4)L +$ $\ell_t L_H + L_p$	$\|sig\| + L_T + \|c_s\|$
ABDSRS	$21L + 2L_T$	$(\|A_s\| + 2)L$	$(2\|A_d\| + 2)L$	$O(\ell_e + \ell_s + \varsigma)L +$ $L_t + L_p + 2L_H$	$O(\ell_t + \|A_d\|)L +$ $O(\ell_t)L_p$	$3L_T + L_H$

cost and communication cost in Tables 3.4 and 3.5, respectively. The comparison of experimental results with [1] (the construction in [1] is called as VMKS) is presented in Figures 3.3 and 3.4. In Figure 3.3, we compare computational efficiency of ABDSRS with VMKS. The DU executes the Signcrypt, TokenGen, Search Result Verify and Decrypt algorithms. Hence, the computing expenses of these algorithms are of utmost importance in scenarios with low processing power gadgets and the comparisons are made in terms of these algorithms only, in Table 3.4 and Figure 3.3. The communication overhead of ABDSRS and VMKS is illustrated in Figure 3.4. Note that in Table 3.5, the size of the ciphertext component corresponding to the plaintext message is not considered while calculating the ciphertext size and the transformed ciphertext size. But, this is included in experimental results.

The proposed ABDSRS is built on the top of ABOOSC [69]. One cannot implement online-offline mechanism in ABSC schemes [14, 2] due to the fact that for each attribute string, one or two random group elements are included in system public parameters. This increases the number of required (expensive) modular exponentiations during signcryption process as well. Hence, the schemes [14, 2] may not be a wise choice to employ as a building block in designing lightweight ABDSRS. Although supporting policy search framework, one drawback of the generic (policy search) ABSE [12] is that the data encryption phase performs two encryptions

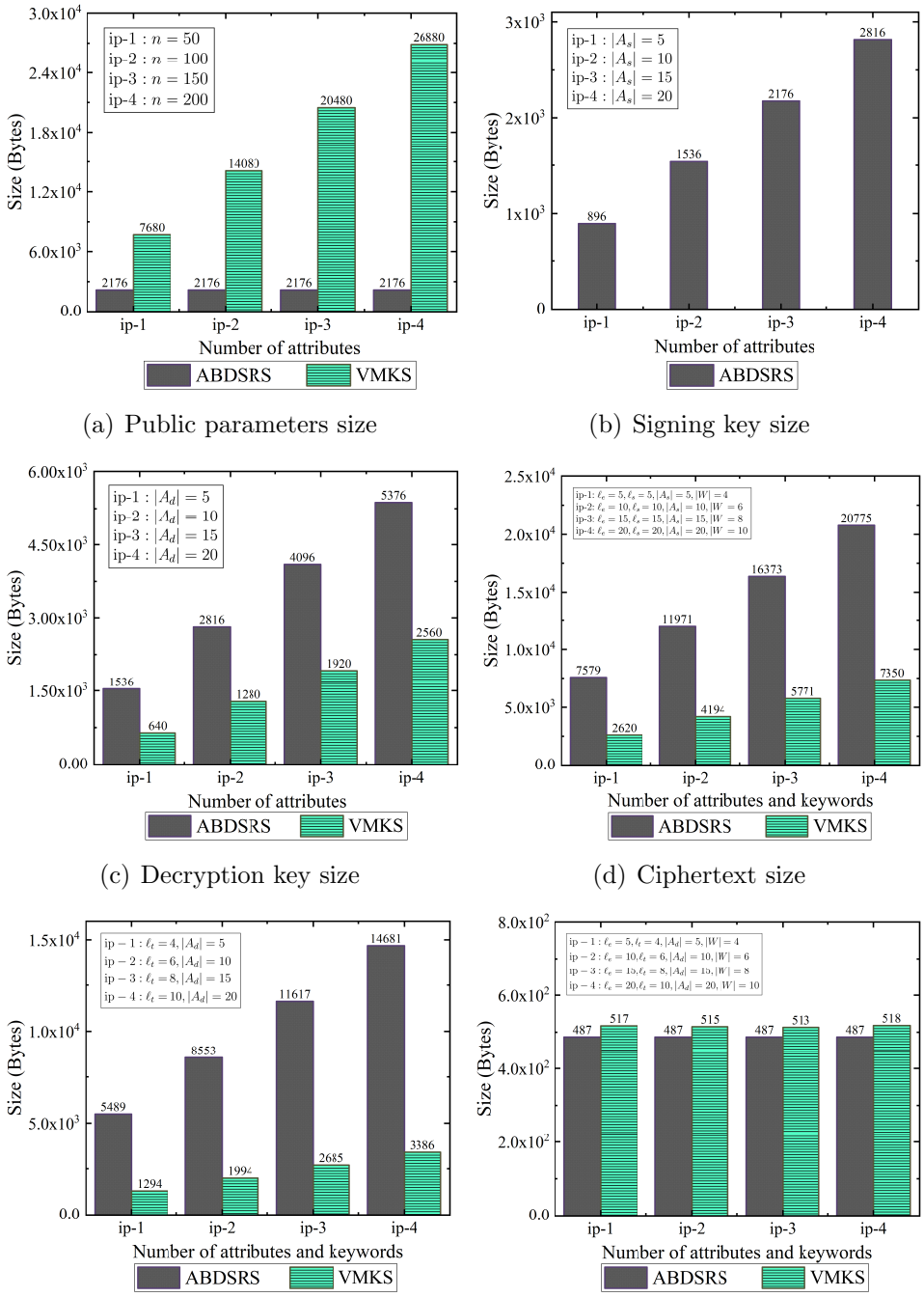


(a) Ciphertext generation (b) DRR/Token generation

(c) Search results verification (d) Decryption

Figure 3.3: Execution time (in ms) of ABDSRS and VMKS [1]

of ABE along with keyword set encryption, and the decryption phase performs encryption in addition to the decryption process. Besides, it suffers from KGAs like [88]. Hence, the ABSE [12] cannot further be extended to design lightweight ABDSRS. The single keyword search ABSE [88, 64, 63] and multi-keyword ABSE [1, 62] execute expensive exponentiation and modular division operations during data encryption and trapdoor generation phases. But, ABDSRS utilizes lightweight modular difference and multiplication operations only. According to Tables 3.4 and 3.5, the ABSE [64] computation and communication costs increase linearly with the number of attributes inattribute universe, with the exception of the time of decryption t and the size of the modified ciphertext. In [1], the online-offline framework is implemented only at sensor nodes because of their low computing power. However, the mobile terminal performs expensive exponentiation operations in the generation of the final ciphertext which is being sent to the cloud. To alleviate both DO and



(a) Public parameters size (b) Signing key size

(c) Decryption key size (d) Ciphertext size

(e) DRR/Token size (f) Transformed ciphertext size

Figure 3.4: Communication cost and storage cost (in bytes) of ABDSRS and VMKS [1]

DU computation burden in ABDSRS, the heavy computation is migrated either to offline phase or to the cloud server. In sum, ABDSRS is computationally efficient compared to the existing ABSE schemes [88, 64, 63, 1, 62, 12].

From Table 3.5, only ABDSRS and [88] achieve constant size system public parameters. The size of the decryption key in ABDSRS is asymptotically comparable to that of the existing ABSE schemes [88, 64, 63, 1, 62, 12], whereas the size of the transformed ciphertext is comparable to that of [88, 64, 63, 1, 62, 12]. To achieve remarkable features such as DO authenticity, DO anonymity, keyword privacy, Boolean formula search and KGAs secure concurrently, ABDSRS sacrifices small size of the ciphertext and token when compared to the ABSE schemes [88, 64, 63, 1, 62, 12]. In [62], DU depends on a trusted public verifier to verify the accuracy of the search results. The fact that the public verifier must always be online in order to validate the search results might make it less useful in practice. ABDSRS, [12] and [1] are the only schemes supporting non-interactive search results verification. But [12] is a generic construction. In subsequent section, we present the experimental analysis of ABDSRS and [1].

The execution of the implementation is conducted on a laptop equipped with an Intel(R) Core(TM) i5-10300H CPU working at a frequency of 2.50 GHz and utilizing 8-GB of RAM. The laptop has 64-bit Ubuntu 20.04 LTS installed on Oracle VM VirtualBox - 6.1.22 memory of 2GB. PBC Library is explored, and a type-A elliptic curve with a prime number group order of 160 bits is selected for experimentation. The curve is $y^2 = x^3 + x$ over a 512-bit finite field.

The running time in milliseconds (ms) of various algorithms of the proposed ABDSRS and VMKS [1] is presented in Figure 3.3. The communication and storage costs (in bytes) of ABDSRS and VMKS are analyzed in Figure 3.4. We use AND-gate encryption policy of the form $a_1 \wedge a_2 \wedge \cdots \wedge a_{\ell_e}$, where $\ell_e = 5, 10, 15, 20$. With each encryption policy, we construct the corresponding DU's decryption key that contains exactly $\ell_e (= |A_d|)$ attributes. We fix the total number of keywords within a keyword policy, $\ell_t = 4, 6, 8, 10$, according to [39]. And, we formulate the keyword policy in the form of an AND-gate $b_1 \wedge b_2 \wedge \cdots \wedge b_{\ell_t}$, where $\ell_t = 4, 6, 8, 10$. In such case, the size of the keyword set ascribed to a ciphertext is $|W| = 4, 6, 8, 10$, respectively. Note that AND-gate policies require the maximum execution time to complete the respective tasks. Total 10 trials are conducted for each experiment, and bar graphs are used to show the average results.

It is observed from Figure 3.4 that the size of the system public parameters is constant in ABDSRS while that for VMKS is linear in the size of the attribute universe. But, the sizes of the decryption key, ciphertext and token in ABDSRS are more than that in VMKS. This is due to the fact that the ciphertext includes a DO signature unlike VMKS, and both the token and the ciphertext include some addi-

tional components to realize remarkable functionalities like keyword privacy, KGAs secure, and keyword policy search simultaneously. However, ABDSRS's transformed ciphertext size is less than that in VMKS.

From Figure 3.3(a), one can observe that when $\ell_e = \ell_s = |A_s| = 20, |W| = 10$, VMKS requires 50.64 ms to generate a ciphertext, while that for our ABDSRS is only 0.75 ms. This is due to the fact that the onSigncrypt algorithm in ABDSRS performs only subtraction and multiplication operations in $\mathbb{Z}_p$, and the time taken by onSigncrypt algorithm is independent of the complexity of the signing policy. As shown in Figure 3.3(b), to create a DRR/token for $\ell_t = 10$ and $|A_d| = 20$, VMKS consumes 29.25 ms, but ABDSRS can create the same DRR with in 11.77 ms. For search results verification, VMKS consumes 5.89 ms on average, whereas ABDSRS consumes 0.21 ms. This can be noticed from Figure 3.3(c). From Figure 3.3(d), VMKS takes 0.1306 ms and ABDSRS takes 0.0058 ms on average to recover the original plaintext. Note that the time taken by verification and decryption processes does not depend on the number of required attributes and keywords in both the schemes. In sum, the experimental results presented in Figure 3.3 exhibit that ABDSRS is computationally efficient when compared to VMKS.

3.6 Chapter Summary

In this chapter, a new notion of an attribute-based cryptosystem (termed as ABDSRS) is presented to support fine-grained data access control, authenticated and secure data storage, efficient data searching, DO anonymity, self-verification of search results and keyword privacy. To deploy in a network equipped with resource-constrained devices, the operations performed in DOs' and DUs' devices in ABDSRS are kept lightweight. In terms of data unforgeability,data confidentiality, keyword privacy, verifiability, and DO privacy, we explicitly defined and demonstrated the security of ABDSRS. The performance as well as property comparison illustrate the efficiency and practicality of the designed ABDSRS.

Chapter 4

Verifiable and Boolean Keyword Searchable Attribute-Based Signcryption for Electronic Medical Record Storage and Retrieval in Cloud Computing Environment

In the preceding chapter, we introduced an online-offline attribute-based searchable signcryption scheme supporting data and DO authenticity, fine-grained data access control, DO anonymity, outsourced unsigncryption, non-interactive search results verification, keyword policy search, and keyword privacy. Along with achieving all the aforementioned functionalities, the proposed searchable signcryption scheme in this chapter, called MediCare, supports an authorized search mechanism where only authorized users can perform the search operation. Additionally, the sizes of the ciphertext, token, decryption key, and transformed ciphertext are reduced.

In this chapter, we propose a searchable attribute-based signcryption for EMR storage and retrieval in cloud computing environment, termed as MediCare, that supports simultaneously the functionalities: *EMR owner authenticity and anonymity,*

The work presented in this chapter is based on our published research article given below.
Sourav Bera, Suryakant Prasad, and Y Sreenivasa Rao. Verifiable and boolean keyword searchable attribute-based signcryption for electronic medical record storage and retrieval in cloud computing environment. *The Journal of Supercomputing*, vol. 79, pp. 1-59, Springer, 2023. https://doi.org/10.1007/s11227-023-05416-8

fine-grained EMR access control, keyword privacy, encrypted EMRs searching based on Boolean formula, constant decryption cost for EMR users, provably secure and independent search results verifiablity. We employ attribute-based framework in designing MediCare. Only authorized EMR owners can anonymously upload EMRs to the cloud, and an EMR user can search over encrypted EMRs using Boolean formula keyword policy. MediCare enables an EMR user to independently check the precision of search outcomes acquired from the cloud. We establish broader security models for MediCare followed by comprehensive security analysis. We also conduct experiments to evaluate MediCare's performance.

4.1 Introduction

Consider a cloud-based EMR management system, where patients, who are the EMR Owners (EOs) upload their EMRs to a public cloud for storing and sharing with the specified EMR Users (EUs), such as physicians, health insurance providers, nurses, etc. In bringing various benefits, the cloud-based EMR management system creates new challenges including data (i.e., EMRs, in this scenario) security and data access control. The sensitive outsourced data should only be accessed by authorized users. Due to the inclusion of sensitive information such as disease names, it is crucial to provide anonymity for the EO when sharing EMRs with EUs. Failure to do so might result in the EO being easily identified within a social network [70].

In addition to these two problems, enabling the EUs to retrieve EMRs of their interest from a cloud is also an important problem. Precisely, an EU may search for his patient's medical history related to a particular disease, say diabetes or heart disease, but does not want to know about other diseases. Hence, how to enable EUs to efficiently filter out required EMRs by specifying suitable keywords is a crucial problem. To access the required patients' EMRs stored at the cloud, an EU creates a data retrieval request using appropriate keywords and delegates the same to the cloud. Next, the cloud locates suitable EMRs and sends them to the EU. Sometimes a patients inaccurate treatment may result from an untrusted cloud occasionally returning a transformed EMR in the correct format but containing incorrect EMR information [40]. This could be serious threat for the patient's life. Overall, the difficulty of allowing an EU to independently check the precision of search outcomes acquired from the cloud becomes another challenging problem.

The issues discussed so far motivate us to construct an EMR management system that supports simultaneously the following functionalities: (i) fine-grained data

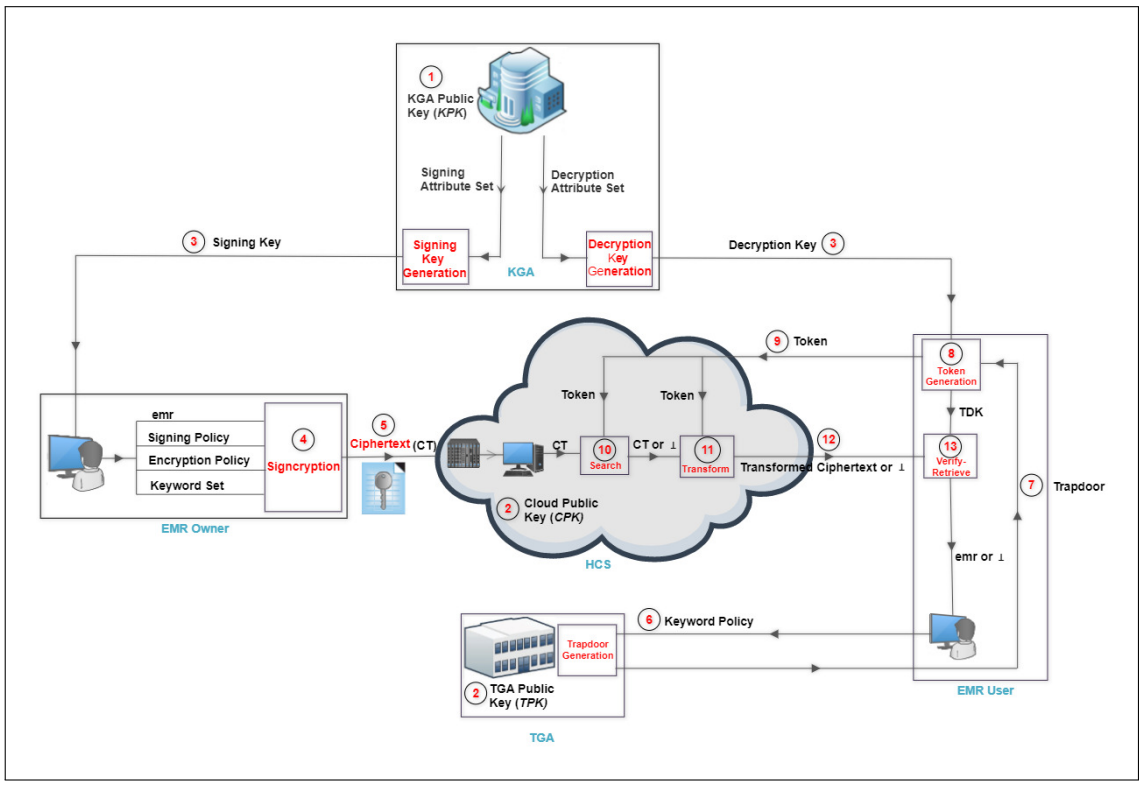


Figure 4.1: Architecture of MediCare

access control, (ii) data and EO authenticity, (iii) Boolean formula based keyword search, (iv) EO anonymity, (v) keyword privacy, (vi) non-interactive search results verification, and (vii) provably secure. With this end in view, we employ attribute-based cryptographic techniques. Such EMR management system, to the best of our knowledge, has not been proposed so far.

Chapter Organization. The rest of the chapter is organized as follows. In Section 4.2, we formally define proposed MediCare and its corresponding security models. The detailed construction of MediCare is presented in Section 4.3. Section 4.4 elaborates the security proofs of MediCare. Following this, the evaluation of MediCare's performance is presented in Section 4.5. Lastly, Section 4.6 concludes the chapter.

4.2 Security of MediCare

4.2.1 System Model

The five components that make up MediCare's architecture are shown in Figure 4.1:

1. **KGA.** It is a completely trusted entity that is responsible for assigning a signing key for a signing attribute[1] set and a decryption key for a decryption attribute[2] set to EO and EU, respectively.

2. **EO.** It possesses EMRs and desires to preserve them in HCS to facilitate sharing and ensure reliable distribution to the specified EUs. The EO signcrypts its own EMR using a signing policy (which accepts the attribute set associated with EO's signing key), an encryption policy (which decides the group of EUs who have the right to view the EMR), and a set of suitable keywords about the EMR. It sends the signcrypted EMR or ciphertext to HCS.

3. **TGA.** The creation of a trapdoor for the keyword policy obtained from the EU is the responsibility of this trustworthy entity. The generated trapdoor is then transmitted via a secure connection to the appropriate EU.

4. **HCS.** It offers services for storing and retrieving EMRs. It stores the ciphertexts outsourced by EOs. The HCS conducts search and transform operations in response to an EU's EMR retrieval request, and then sends the EU the resulting "partially decrypted matching ciphertexts" (also known as "transformed ciphertexts"). Note that a ciphertext is called matching ciphertext if its keyword set satisfies the keyword policy associated with the trapdoor.

5. **EU.** This is an entity who wants to access the ciphertexts stored at HCS. The EU queries the HCS by sending a token and obtains back the corresponding transformed ciphertexts. Then, the EU verifies the accuracy of the search, transform and signature verification processes performed by HCS, and retrieves the original EMRs.

4.2.2 Security Models

This section provides a formal definition of the security measures implemented in MediCare, which includes DO privacy, data unforgeability, data confidentiality, keyword privacy, and verifiability. To enhance comprehension, we have included the notations used in MediCare in Table 4.1. Our proposed MediCare (shown in Figure 4.1), consists of the following five phases, supports signing policy Γ_s over signing attribute universe U_s, encryption policy Γ_e over encryption attribute universe U_e

[1]We will use signing attribute and EO attribute interchangeably.
[2]We will use encryption attribute, decryption attribute and EU attribute interchangeably.

Table 4.1: Notations used in MediCare

KGA (resp. TGA)	:	key (resp. trapdoor) generation authority
EO (resp. EU)	:	EMR owner (resp. EMR user)
HCS	:	Healthcare Cloud Service
$\mathcal{KPK}$	:	KGA public parameters
$\mathcal{MK}$	:	system master secret
$\mathcal{PP}$	:	system global public parameters
$\mathcal{SK}_{A_s}$	:	signing key for the signing attribute set A_s
$\mathcal{TPK}$ (resp. $\mathcal{TSK}$)	:	TGA public key (resp. TGA secret key)
$\mathcal{CPK}$ (resp. $\mathcal{CSK}$)	:	HCS public key (resp. HCS secret key)
$\mathcal{DK}_{A_d}$	:	decryption key for the decryption attribute set A_d
emr	:	EMR file or plaintext
Γ_e (resp. Γ_s, Γ_t)	:	encryption (resp. signing, keyword) policy
$\mathcal{TDK}$	:	secret transformation decryption key
$\mathcal{CT}$	:	ciphertext for Γ_s, Γ_e and keyword set W
$\mathcal{CT}_u$	:	stored ciphertext of $\mathcal{CT}$
$\mathcal{CT}_{tr}$	:	transformed ciphertext
$\mathcal{TK}_{A_d}$	:	transform key for A_d derived from $\mathcal{DK}_{A_d}$
$\widetilde{\mathcal{TD}}_{\Gamma_t}$	:	trapdoor for Γ_t returned by TGA
$\mathcal{TD}_{\Gamma_t^\circ}$	:	transform trapdoor for Γ_t derived by EU from $\widetilde{\mathcal{TD}}_{\Gamma_t}$

and keyword policy Γ_t over keyword universe U_t. Define the attribute universe $U := U_s \cup U_e \cup U_t$ and $\mathcal{M}$:= plaintext space.

1. *System Setup*

   First, KGA generates the system public parameters $\mathcal{KPK}$ and the system master secret $\mathcal{MK}$ by running KGA-Setup algorithm with the security parameter 1^κ as input. $\mathcal{MK}$ is kept secret by KGA.

   Next, by taking $\mathcal{KPK}$ as input, TGA and HCS create their public and secret key pair respectively $[\mathcal{TPK}, \mathcal{TSK}]$ and $[\mathcal{CPK}, \mathcal{CSK}]$ by executing TGA-Setup and HCS-Setup algorithms. They make $\mathcal{TPK}$ and $\mathcal{CPK}$ public, while $\mathcal{TSK}$ and $\mathcal{CSK}$ are kept secret by TGA and HCS, respectively.

   (a) KGA-Setup$(1^\kappa) \rightarrow [\mathcal{KPK}, \mathcal{MK}]$

   (b) TGA-Setup$(\mathcal{KPK}) \rightarrow [\mathcal{TPK}, \mathcal{TSK}]$

   (c) HCS-Setup$(\mathcal{KPK}) \rightarrow [\mathcal{CPK}, \mathcal{CSK}]$

   Set the system global public parameters $\mathcal{PP} := \langle \mathcal{KPK}, \mathcal{CPK}, \mathcal{TPK} \rangle$.

2. *Registration*

In this phase, KGA issues signing key $\mathcal{SK}_{A_s}$ to EO and decryption key $\mathcal{DK}_{A_d}$ to EU, by running $\mathsf{sKeyGen}$ and $\mathsf{dKeyGen}$ algorithms, respectively.

(a) $\mathsf{sKeyGen}(\mathcal{PP}, \mathcal{MK}, A_s) \rightarrow \mathcal{SK}_{A_s}$. Taking $\mathcal{PP}$, $\mathcal{MK}$ and a signing attribute set $A_s \subset U_s$, this algorithm produces a signing key $\mathcal{SK}_{A_s}$.

(b) $\mathsf{dKeyGen}(\mathcal{PP}, \mathcal{MK}, A_d) \rightarrow \mathcal{DK}_{A_d}$. On input $\mathcal{PP}$, $\mathcal{MK}$ and a decryption attribute set $A_d \subset U_e$, this algorithm returns a decryption key $\mathcal{DK}_{A_d}$.

3. *Ciphertexts Uploading*
When an EO wants to outsource its EMR *emr* to HCS for storing and sharing, it signcrypts *emr* by executing $\mathsf{Signcrypt}$ algorithm and creates the corresponding ciphertext $\mathcal{CT}$.
Next, EO delivers the ciphertext $\mathcal{CT}$ to the HCS.

(a) $\mathsf{Signcrypt}(\mathcal{PP}, \mathcal{SK}_{A_s}, \Gamma_s, \Gamma_e, W, emr) \rightarrow \mathcal{CT}$. It takes as input $\mathcal{PP}, \mathcal{SK}_{A_s}$, a signing policy Γ_s such that $\Gamma_s(A_s) = 1$, an encryption policy Γ_e, a keyword set W and an EMR $emr \in \mathcal{M}$, and outputs a ciphertext $\mathcal{CT}$. It should be noted that Γ_s, Γ_e and the set W° (of keywords that includes only generic names) are being incorporated in $\mathcal{CT}$.
Let $W := \{[\mathcal{W}_1 : w_1], \ldots, [\mathcal{W}_\varsigma : w_\varsigma]\}$, where $\mathcal{W}_i$ represents the generic keyword name and w_i represents the associated keyword value. In this case, $W^\circ := \{\mathcal{W}_1, \ldots, \mathcal{W}_\varsigma\}$.

After verifying the authenticity of the EMR and EO, HCS uploads the ciphertext.

4. *EMR Retrieval Token Generation*
EU sends a keyword policy Γ_t to TGA, requesting a trapdoor. After executing the $\mathsf{TrapGen}$ algorithm, TGA provides the EU with the corresponding trapdoor $\widetilde{\mathcal{TD}}_{\Gamma_t}$.
Next, the EU performs $\mathsf{TokenGen}$ algorithm and creates an EMR retrieval request token *token* and a secret transformation decryption key $\mathcal{TDK}$.
Lastly, the EU sends *token* to the HCS.

(a) $\mathsf{TrapGen}(\mathcal{PP}, \mathcal{TSK}, \Gamma_t) \rightarrow \widetilde{\mathcal{TD}}_{\Gamma_t}$. On input $\mathcal{PP}, \mathcal{TSK}$, a keyword policy Γ_t, this algorithm produces the trapdoor $\widetilde{\mathcal{TD}}_{\Gamma_t}$ for the policy Γ_t.

(b) $\mathsf{TokenGen}(\mathcal{PP}, \widetilde{\mathcal{TD}}_{\Gamma_t}, \mathcal{DK}_{A_d}) \rightarrow [token, \mathcal{TDK}]$. It takes as input $\mathcal{PP}$, $\widetilde{\mathcal{TD}}_{\Gamma_t}, \mathcal{DK}_{A_d}$, and outputs the EMR retrieval request *token* and the secret transformation decryption key $\mathcal{TDK}$. Note that *token* contains two

components $\mathcal{TD}_{\Gamma_t^\circ}$ and $\mathcal{TK}_{A_d}$, where the former is transform trapdoor (which is derived from $\widetilde{\mathcal{TD}}_{\Gamma_t}$) for Γ_t and the latter is transform key (which is derived from $\mathcal{DK}_{A_d}$) for the decryption attribute set A_d. That is, $token := \langle \mathcal{TD}_{\Gamma_t^\circ}, \mathcal{TK}_{A_d} \rangle$. Only Γ_t° will be included in $\mathcal{TD}_{\Gamma_t^\circ}$, and hence in $token$, where Γ_t° is the policy Γ_t with only generic names of the keywords.

5. *EMR Retrieval*

Upon receiving the EMR retrieval request $token$ from EU, the HCS first performs the Search operation to identify the matching ciphertexts.
After, it generates the transformed ciphertexts $\mathcal{CT}_{tr}$ by applying Transform operation to the matching ciphertexts.
Next, EU receives $\mathcal{CT}_{tr}$ from HCS and verifies the accuracy of Search and Transform algorithms performed by HCS with its secret transformation decryption key $\mathcal{TDK}$, then accordingly EU recovers the emr, by executing Verify-Retrieve algorithm.

(a) $\mathsf{Search}(\mathcal{PP}, \mathcal{CT}_u, token, \mathcal{CSK}) \rightarrow \mathcal{CT}_u$ or $\perp$. On input $\mathcal{PP}, \mathcal{CT}_u, token$, $\mathcal{CSK}$, the search algorithm outputs the ciphertext $\mathcal{CT}_u$ if $\Gamma_t(W) = 1$; otherwise, outputs $\perp$.

(b) $\mathsf{Transform}(\mathcal{PP}, \mathcal{CT}_u, token, \mathcal{CSK}) \rightarrow \mathcal{CT}_{tr}$ or $\perp$. Taking $\mathcal{PP}, \mathcal{CT}_u, token$, $\mathcal{CSK}$ as input, this transform algorithm outputs the transformed ciphertext $\mathcal{CT}_{tr}$ if Search outputs $\mathcal{CT}_u$ and $\Gamma_e(A_d) = 1$; and outputs $\perp$ otherwise.

(c) $\mathsf{Verify\text{-}Retrieve}(\mathcal{PP}, \mathcal{CT}_{tr}, \mathcal{TDK}) \rightarrow emr$ or $\perp$. Taking $\mathcal{PP}$, $\mathcal{CT}_{tr}$ and $\mathcal{TDK}$, it outputs emr if the search result $\mathcal{CT}_u$ sent by HCS is correct (i.e., the keyword set associated with $\mathcal{CT}_u$ satisfies the keyword policy included in $token$ and the decryption attribute set annotated to $token$ satisfies the encryption policy associated with $\mathcal{CT}_u$) as well as $\mathcal{CT}$ includes a legitimate signature that adheres to the signing policy specified within $\mathcal{CT}$. Otherwise, it outputs $\perp$.

Data Confidentiality

Unauthorized entities should not decrypt a ciphertext stored in the cloud, which is ensured by the security property called data confidentiality. In MediCare, only the cloud can provide the ciphertext storage service that any authorized EU can use. So, a EU with a valid token can't perform search operation and decrypt the

Experiment $\mathsf{Game}^{\mathsf{IND\text{-}CCA2}}_{\mathsf{Type\text{-}1}}(1^\kappa)$

1. $y^\star \leftarrow \mathscr{A}(1^\kappa)$//where $y^\star \in U_e$//
2. $[\mathcal{KPK}, \mathcal{MK}] \leftarrow \mathsf{KGA\text{-}Setup}(1^\kappa), [\mathcal{TPK}, \mathcal{TSK}] \leftarrow \mathsf{TGA\text{-}Setup}(\mathcal{KPK})$, $[\mathcal{CPK}, \mathcal{CSK}] \leftarrow \mathsf{HCS\text{-}Setup}(\mathcal{KPK})$
3. $[emr_0^\star, emr_1^\star, \Gamma_s^\star, \Gamma_e^\star, W^\star, \mathsf{st}] \leftarrow \mathscr{A}^{\mathcal{O}_1}(\mathcal{PP}, \mathcal{CSK})$ //where $|emr_0^\star| = |emr_1^\star|$, $\mathcal{O}_1 := \{\mathcal{O}_{\mathcal{SKG}}, \mathcal{O}_{\mathcal{TG}}, \mathcal{O}_{\mathcal{CG}}, \mathcal{O}_{\mathcal{ER}}\}$//
4. $\mathcal{CT}^\star \leftarrow \mathsf{Signcrypt}(\mathcal{PP}, \mathcal{SK}_{A_s}, \Gamma_s^\star, \Gamma_e^\star \wedge y^\star, W^\star, emr_i^\star)$//where $i \xleftarrow{\text{u}} \{0,1\}$, $A_s \xleftarrow{\text{u}} 2^{U_s} \ni \Gamma_s^\star(A_s) = 1, \mathcal{SK}_{A_s} \leftarrow \mathsf{sKeyGen}(\mathcal{PP}, \mathcal{MK}, A_s)$//
5. $i' \leftarrow \mathscr{A}^{\mathcal{O}_2}(y^\star, \mathcal{PP}, \mathcal{CSK}, emr_0^\star, emr_1^\star, \Gamma_s^\star, \Gamma_e^\star, W^\star, \mathsf{st}, \mathcal{CT}^\star)$ //where $\mathcal{O}_2 := \{\mathcal{O}_{\mathcal{SKG}}, \mathcal{O}_{\mathcal{TG}}, \mathcal{O}_{\mathcal{CG}}, \mathcal{O}'_{\mathcal{ER}}\}$//

Experiment $\mathsf{Game}^{\mathsf{IND\text{-}CCA2}}_{\mathsf{Type\text{-}2}}(1^\kappa)$

1. $[\mathcal{KPK}, \mathcal{MK}] \leftarrow \mathsf{KGA\text{-}Setup}(1^\kappa), [\mathcal{TPK}, \mathcal{TSK}] \leftarrow \mathsf{TGA\text{-}Setup}(\mathcal{KPK})$, $[\mathcal{CPK}, \mathcal{CSK}] \leftarrow \mathsf{HCS\text{-}Setup}(\mathcal{KPK})$
2. $[emr_0^\star, emr_1^\star, \Gamma_e^\star, \Gamma_s^\star, W^\star, \mathsf{st}] \leftarrow \mathscr{A}^{\mathcal{O}_3}(\mathcal{PP})$ //where $|emr_0^\star| = |emr_1^\star|$, $\mathcal{O}_3 := \{\mathcal{O}_{\mathcal{SKG}}, \mathcal{O}'_{\mathcal{TG}}, \mathcal{O}_{\mathcal{CG}}, \mathcal{O}_{\mathcal{ER}}\}$//
3. $\mathcal{CT}^\star \leftarrow \mathsf{Signcrypt}(\mathcal{PP}, \mathcal{SK}_{A_s}, \Gamma_s^\star, \Gamma_e^\star, W^\star, emr_i^\star)$//where $i \xleftarrow{\text{u}} \{0,1\}$, $A_s \xleftarrow{\text{u}} 2^{U_s} \ni \Gamma_s^\star(A_s) = 1, \mathcal{SK}_{A_s} \leftarrow \mathsf{sKeyGen}(\mathcal{PP}, \mathcal{MK}, A_s)$//
4. $i' \leftarrow \mathscr{A}^{\mathcal{O}_4}(\mathcal{PP}, emr_0^\star, emr_1^\star, \Gamma_s^\star, \Gamma_e^\star, W^\star, \mathsf{st}, \mathcal{CT}^\star)$ //where $\mathcal{O}_4 := \{\mathcal{O}_{\mathcal{SKG}}, \mathcal{O}'_{\mathcal{TG}}, \mathcal{O}_{\mathcal{CG}}, \mathcal{O}'_{\mathcal{ER}}\}$//

Experiment $\mathsf{Game}^{\mathsf{EUF\text{-}CMA}}_{\mathscr{A}}(1^\kappa)$

1. $\Gamma_s^\star \leftarrow \mathscr{A}(1^\kappa)$
2. $[\mathcal{KPK}, \mathcal{MK}] \leftarrow \mathsf{KGA\text{-}Setup}(1^\kappa), [\mathcal{TPK}, \mathcal{TSK}] \leftarrow \mathsf{TGA\text{-}Setup}(\mathcal{KPK})$, $[\mathcal{CPK}, \mathcal{CSK}] \leftarrow \mathsf{HCS\text{-}Setup}(\mathcal{KPK})$
3. $\mathcal{CT}^\star \leftarrow \mathscr{A}^{\mathcal{O}}(\mathcal{PP}, \mathcal{CSK})$ //where $\mathcal{O} := \{\mathcal{O}'_{\mathcal{SKG}}, \mathcal{O}'_{\mathcal{TG}}, \mathcal{O}_{\mathcal{CG}}, \mathcal{O}_{\mathcal{ER}}\}$//

Experiment $\mathsf{Game}^{\mathsf{EO\text{-}Anonymity}}_{\mathscr{A}}(1^\kappa)$

1. $[\mathcal{KPK}, \mathcal{MK}] \leftarrow \mathsf{KGA\text{-}Setup}(1^\kappa), [\mathcal{TPK}, \mathcal{TSK}] \leftarrow \mathsf{TGA\text{-}Setup}(\mathcal{KPK})$, $[\mathcal{CPK}, \mathcal{CSK}] \leftarrow \mathsf{HCS\text{-}Setup}(\mathcal{KPK})$
2. $[A_s^{(0)}, A_s^{(1)}, emr, \Gamma_e, \Gamma_s, W, \mathsf{st}] \leftarrow \mathscr{A}(\mathcal{PP}, \mathcal{MK}, \mathcal{TSK}, \mathcal{CSK})$ //where $\Gamma_s(A_s^{(0)}) = 1 = \Gamma_s(A_s^{(1)})$//
3. $\mathcal{CT}^\star \leftarrow \mathsf{Signcrypt}(\mathcal{PP}, \mathcal{SK}_{A_s^{(i)}}, \Gamma_s, \Gamma_e, W, emr)$ //where $i \xleftarrow{\text{u}} \{0,1\}, \mathcal{SK}_{A_s^{(i)}} \leftarrow \mathsf{sKeyGen}(\mathcal{PP}, \mathcal{MK}, A_s^{(i)})$//
4. $i' \leftarrow \mathscr{A}(\mathcal{PP}, \mathcal{MK}, \mathcal{TSK}, \mathcal{CSK}, A_s^{(0)}, A_s^{(1)}, emr, \Gamma_e, \Gamma_s, W, \mathsf{st}, \mathcal{CT}^\star)$

Experiment $\mathsf{Game}^{\mathsf{verifiability}}_{\mathscr{A}}(1^\kappa)$

1. $[\mathcal{KPK}, \mathcal{MK}] \leftarrow \mathsf{KGA\text{-}Setup}(1^\kappa), [\mathcal{TPK}, \mathcal{TSK}] \leftarrow \mathsf{TGA\text{-}Setup}(\mathcal{KPK})$, $[\mathcal{CPK}, \mathcal{CSK}] \leftarrow \mathsf{HCS\text{-}Setup}(\mathcal{KPK})$
2. $[emr^\star, \Gamma_e^\star, \Gamma_s^\star, W^\star, \mathsf{st}] \leftarrow \mathscr{A}^{\widetilde{\mathcal{O}}}(\mathcal{PP}, \mathcal{CSK})$ //where $\widetilde{\mathcal{O}} := \{\mathcal{O}_{\mathcal{SKG}}, \mathcal{O}''_{\mathcal{TG}}, \mathcal{O}_{\mathcal{CG}}, \mathcal{O}_{\mathcal{ER}}\}$//
3. $\mathcal{CT}^\star \leftarrow \mathsf{Signcrypt}(\mathcal{PP}, \mathcal{SK}_{A_s}, \Gamma_s^\star, \Gamma_e^\star, W^\star, emr^\star)$ //where $A_s \xleftarrow{\text{u}} 2^{U_s} \ni \Gamma_s^\star(A_s) = 1, \mathcal{SK}_{A_s} \leftarrow \mathsf{sKeyGen}(\mathcal{PP}, \mathcal{MK}, A_s)$//
4. $[A_d, \Gamma_t, \mathcal{CT}_{tr}] \leftarrow \mathscr{A}^{\widetilde{\mathcal{O}}}(\mathcal{PP}, \mathcal{CSK}, emr^\star, \Gamma_e^\star, \Gamma_s^\star, W^\star, \mathsf{st}, \mathcal{CT}^\star)$ //where $\Gamma_e^\star(A_d) = 1 \wedge \Gamma_t(W^\star) = 1$

Figure 4.2: Security games.

Experiment $\mathsf{Game}^{\mathsf{IND\text{-}CKA}}_{\mathsf{Type\text{-}1}}(1^\kappa)$

1. $W_0^\star, W_1^\star \leftarrow \mathscr{A}(1^\kappa)$ //where $|W_0^\star| = |W_1^\star|$ and $W_0^{\star\circ} = W_1^{\star\circ}$//
2. $[\mathcal{KPK}, \mathcal{MK}] \leftarrow \mathsf{KGA\text{-}Setup}(1^\kappa), [\mathcal{TPK}, \mathcal{TSK}] \leftarrow \mathsf{TGA\text{-}Setup}(\mathcal{KPK})$, $[\mathcal{CPK}, \mathcal{CSK}] \leftarrow \mathsf{HCS\text{-}Setup}(\mathcal{KPK})$
3. $[emr^\star, \Gamma_e^\star, \Gamma_s^\star, \mathsf{st}] \leftarrow \mathscr{A}^{\widehat{\mathcal{O}}}(\mathcal{PP}, \mathcal{CSK})$ //where $\widehat{\mathcal{O}} := \{\mathcal{O}_{\mathcal{SKG}}, \mathcal{O}'''_{\mathcal{TG}}, \mathcal{O}_{search}\}$//
4. $\mathcal{CT}^\star \leftarrow \mathsf{Signcrypt}(\mathcal{PP}, \mathcal{SK}_{A_s}, \Gamma_s^\star, \Gamma_e^\star, W_i^\star, emr^\star)$ //where $i \xleftarrow{\text{u}} \{0,1\}$, $A_s \xleftarrow{\text{u}} 2^{U_s} \ni \Gamma_s^\star(A_s) = 1, \mathcal{SK}_{A_s} \leftarrow \mathsf{sKeyGen}(\mathcal{PP}, \mathcal{MK}, A_s)$//
5. $i' \leftarrow \mathscr{A}^{\widehat{\mathcal{O}}}(\mathcal{PP}, \mathcal{CSK}, emr^\star, \Gamma_e^\star, \Gamma_s^\star, W_0^\star, W_1^\star, \mathsf{st}, \mathcal{CT}^\star)$.

Experiment $\mathsf{Game}^{\mathsf{IND\text{-}CKA}}_{\mathsf{Type\text{-}2}}(1^\kappa)$

1. $[\mathcal{KPK}, \mathcal{MK}] \leftarrow \mathsf{KGA\text{-}Setup}(1^\kappa), [\mathcal{TPK}, \mathcal{TSK}] \leftarrow \mathsf{TGA\text{-}Setup}(\mathcal{KPK})$, $[\mathcal{CPK}, \mathcal{CSK}] \leftarrow \mathsf{HCS\text{-}Setup}(\mathcal{KPK})$
2. $[emr^\star, \Gamma_e^\star, \Gamma_s^\star, W_0^\star, W_1^\star, \mathsf{st}] \leftarrow \mathscr{A}^{\mathcal{O}_5}(\mathcal{PP})$ //where $|W_0^\star| = |W_1^\star|$, $W_0^{\star\circ} = W_1^{\star\circ}, \mathcal{O}_5 := \{\mathcal{O}_{\mathcal{SKG}}, \mathcal{O}'_{\mathcal{TG}}, \mathcal{O}'_{search}\}$//
3. $\mathcal{CT}^\star \leftarrow \mathsf{Signcrypt}(\mathcal{PP}, \mathcal{SK}_{A_s}, \Gamma_s^\star, \Gamma_e^\star, W_i^\star, emr^\star)$ //where $i \xleftarrow{\text{u}} \{0,1\}$, $A_s \xleftarrow{\text{u}} 2^{U_s} \ni \Gamma_s^\star(A_s) = 1, \mathcal{SK}_{A_s} \leftarrow \mathsf{sKeyGen}(\mathcal{PP}, \mathcal{MK}, A_s)$//
4. $i' \leftarrow \mathscr{A}^{\mathcal{O}_6}(\mathcal{PP}, emr^\star, \Gamma_e^\star, \Gamma_s^\star, W_0^\star, W_1^\star, \mathsf{st}, \mathcal{CT}^\star)$ //where $\mathcal{O}_6 := \{\mathcal{O}_{\mathcal{SKG}}, \mathcal{O}'_{\mathcal{TG}}, \mathcal{O}''_{search}\}$//

Figure 4.3: Security games.

ciphertext if it doesn't know the cloud secret key ($\mathcal{CSK}$). Therefore, an adversary can manifest as either a Type-1 adversary, which refers to an unauthorized entity possessing $\mathcal{CSK}$, or a Type-2 adversary, which refers to an authorized entity that is ignorant of $\mathcal{CSK}$. The following defines this MediCare security concept using the IND-CCA2 security game, wherein the ciphertexts are indistinguishable under an adaptive chosen ciphertext attack.

IND-CCA2 Security for Type-1 Adversary

The scenario is depicted in the security game $\mathsf{Game}^{\mathsf{IND\text{-}CCA2}}_{\mathsf{Type\text{-}1}}$ (given in Figure 4.2), which poses a challenger $\mathscr{C}$ against a Type-1 adversary $\mathscr{A}$.

In this game, $\mathcal{O}_1 := \{\mathcal{O}_{\mathcal{SKG}}, \mathcal{O}_{\mathcal{TG}}, \mathcal{O}_{\mathcal{CG}}, \mathcal{O}_{\mathcal{ER}}\}$ and $\mathcal{O}_2 := \{\mathcal{O}_{\mathcal{SKG}}, \mathcal{O}_{\mathcal{TG}}, \mathcal{O}_{\mathcal{CG}}, \mathcal{O}'_{\mathcal{ER}}\}$ are two sets of oracles (defined below), 2^{U_s} is the set of all non-empty subsets of the signing attribute universe U_s, and st is the state information maintained by $\mathscr{A}$.

- *Signing key generation oracle* $\mathcal{O}_{\mathcal{SKG}}(A_s)$: on input a signing attribute set A_s, it returns the signing key $\mathcal{SK}_{A_s}$ to $\mathscr{A}$.
- *Token generation oracle* $\mathcal{O}_{\mathcal{TG}}(A_d, \Gamma_t)$: on input a decryption attribute set A_d

and a keyword policy Γ_t, it performs as follows.
(i) In case $y^\star \in A_d$, it computes the EMR retrieval request token *token* and the secret transformation decryption key $\mathcal{TDK}$, and returns *token* to $\mathscr{A}$.
(ii) In case $y^\star \notin A_d$, it computes $[\mathcal{DK}_{A_d}, \widetilde{\mathcal{TD}}_{\Gamma_t}]$ and returns the same to $\mathscr{A}$. Note that, in this case, $\mathscr{A}$ can generate *token* and $\mathcal{TDK}$ from $\mathcal{DK}_{A_d}$ and $\widetilde{\mathcal{TD}}_{\Gamma_t}$.

- *Ciphertext generation oracle* $\mathcal{O}_{\mathcal{CG}}(emr, \Gamma_s, \Gamma_e, W)$: on input an encryption policy Γ_e, a plaintext file *emr*, a signing policy Γ_s, and a set W of keywords, it generates and forwards the ciphertext $\mathcal{CT}$ to $\mathscr{A}$.

- *EMR retrieval oracle* $\mathcal{O}_{\mathcal{ER}}(\mathcal{CT}, A_d, \Gamma_t)$: on input a ciphertext $\mathcal{CT}$, a decryption attribute set A_d and a keyword policy Γ_t, it returns to $\mathscr{A}$ either *emr* or $\perp$.

- $\mathcal{O}'_{\mathcal{ER}}$ is same as $\mathcal{O}_{\mathcal{ER}}$, except that $\mathscr{A}$ is not permitted to query $\mathcal{O}'_{\mathcal{ER}}$ using the input $(\mathcal{CT}^\star, A_d, \Gamma_t)$ satisfying $y^\star \in A_d$ and $\Gamma_t(W^\star) = 1$, here $W^\star$ is the keyword set of $\mathcal{CT}^\star$.

IND-CCA2 Security for Type-2 Adversary

The scenario is depicted in the security game $\mathsf{Game}^{\mathsf{IND\text{-}CCA2}}_{\mathsf{Type\text{-}2}}$ (described in Figure 4.2), which poses a challenger $\mathscr{C}$ against a Type-2 adversary $\mathscr{A}$.

In the game, $\mathcal{O}_3 := \{\mathcal{O}_{\mathcal{SKG}}, \mathcal{O}'_{\mathcal{TG}}, \mathcal{O}_{\mathcal{CG}}, \mathcal{O}_{\mathcal{ER}}\}$ and $\mathcal{O}_4 := \{\mathcal{O}_{\mathcal{SKG}}, \mathcal{O}'_{\mathcal{TG}}, \mathcal{O}_{\mathcal{CG}}, \mathcal{O}'_{\mathcal{ER}}\}$ are two sets of oracles which are defined below.

- *Token generation oracle* $\mathcal{O}'_{\mathcal{TG}}(A_d, \Gamma_t)$: on input a decryption attribute set A_d and a keyword policy Γ_t, it returns $[\mathcal{DK}_{A_d}, \widetilde{\mathcal{TD}}_{\Gamma_t}]$ to $\mathscr{A}$. Note that $\mathscr{A}$ can generate *token* and $\mathcal{TDK}$ from $\mathcal{DK}_{A_d}$ and $\widetilde{\mathcal{TD}}_{\Gamma_t}$.

The other oracles are essentially the same as that of $\mathsf{Game}^{\mathsf{IND\text{-}CCA2}}_{\mathsf{Type\text{-}1}}(1^\kappa)$.

If $i' = i$, $\mathscr{A}$ wins $\mathsf{Game}^{\mathsf{IND\text{-}CCA2}}_{\mathsf{Type\text{-}k}}(1^\kappa)$, where $\mathsf{k} \in \{1, 2\}$. The adversary's advantage in $\mathsf{Game}^{\mathsf{IND\text{-}CCA2}}_{\mathsf{Type\text{-}k}}(1^\kappa)$ is described as $\mathsf{Adv}^{\mathsf{IND\text{-}CCA2}}_{\mathsf{Type\text{-}k}}(1^\kappa) \stackrel{\text{def}}{=} \left|\Pr[i' = i] - 1/2\right|$.

Definition 8. *The MediCare is said to be IND-CCA2 secure if* $\mathsf{Adv}^{\mathsf{IND\text{-}CCA2}}_{\mathsf{Type\text{-}1}}(1^\kappa)$ *and* $\mathsf{Adv}^{\mathsf{IND\text{-}CCA2}}_{\mathsf{Type\text{-}2}}(1^\kappa)$ *are negligible, for all PPT Type-1 and Type-2 adversaries, respectively.*

Unforgeability of the Data

It identifies the inability of an external malicious entity or an unauthorized EO to generate a valid signature, thereby ensuring the signature verification mechanism is successful. And, even if unauthorized EOs collude and pool their signing attributes

such that the collection of attributes satisfies a signing policy whereas none of the single EOs would satisfy the policy on its own, they cannot create a ciphertext with valid signature for that signing policy. Existential unforgeability against Chosen Message Attack (EUF-CMA) model defines this MediCare security concept.

EUF-CMA Security

The security game $\mathsf{Game}_{\mathscr{A}}^{\mathsf{EUF\text{-}CMA}}$ (presented in Figure 4.2) involves a model that consists of a challenger $\mathscr{C}$ and an adversary $\mathscr{A}$. In this game, $\mathcal{CT}^{\star} = \mathcal{CT}^{\star}_{(\Gamma_s^{\star}, \Gamma_e^{\star}, W^{\star})}$, $\mathcal{O} := \{\mathcal{O}'_{\mathcal{SKG}}, \mathcal{O}'_{\mathcal{TG}}, \mathcal{O}_{\mathcal{CG}}, \mathcal{O}_{\mathcal{ER}}\}$ is a set of oracles and

- *Signing key generation oracle* $\mathcal{O}'_{\mathcal{SKG}}(A_s)$: on input a signing attribute set A_s with the condition that $\Gamma_s^{\star}(A_s) = 0$, it returns the signing key $\mathcal{SK}_{A_s}$ to $\mathscr{A}$.

Note that the other oracles are similar to that of $\mathsf{Game}_{\mathsf{Type\text{-}2}}^{\mathsf{IND\text{-}CCA2}}(1^{\kappa})$.

The adversary $\mathscr{A}$ wins this game if there exist A_d, Γ_t satisfying the following simultaneously: $\Gamma_e^{\star}(A_d) = 1, \Gamma_t(W^{\star}) = 1$, $\mathcal{O}_{\mathcal{ER}}(\mathcal{CT}^{\star}, A_d, \Gamma_t) = emr^{\star} \neq \perp$, and $\mathscr{A}$ was never queried to the ciphertext generation oracle $\mathcal{O}_{\mathcal{CG}}$ with the input $(emr^{\star}, \Gamma_s^{\star}, \Gamma_e^{\star}, W^{\star})$.

The advantage of $\mathscr{A}$ in this game is defined as
$\mathsf{Adv}_{\mathscr{A}}^{\mathsf{EUF\text{-}CMA}}(1^{\kappa}) \stackrel{\text{def}}{=} \Pr[\mathscr{A} \text{ wins the game}]$.

Definition 9. *The MediCare demonstrates EUF-CMA security if* $\mathsf{Adv}_{\mathscr{A}}^{\mathsf{EUF\text{-}CMA}}(1^{\kappa})$ *becomes negligible, for any PPT adversaries* $\mathscr{A}$*.*

EO Anonymity

This guarantees that the ciphertext does not reveal the set of attributes used in signing process. No one, including the HCS, an EU, or any other adversary, can deduce the set of EO's attributes used to generate a signature from a certain ciphertext. The scenario is depicted in the following security game $\mathsf{Game}_{\mathscr{A}}^{\mathsf{EO\text{-}Anonymity}}$ (detailed in Figure 4.2), which poses a challenger $\mathscr{C}$ against an adversary $\mathscr{A}$.

In this game, $\mathscr{A}$ does not need to query any oracle, however, it can compute required components by itself because $\mathscr{A}$ has the knowledge of system master secret, HCS secret key and TGA secret key.

If $i' = i$, $\mathscr{A}$ wins the game. Therefore, $\mathscr{A}$'s advantage in winning the game is $\mathsf{Adv}_{\mathscr{A}}^{\mathsf{EO\text{-}Anonymity}}(1^{\kappa}) \stackrel{\text{def}}{=} \Pr[i' = i]$.

Definition 10. *The MediCare is said to provide EO anonymity if* $\mathsf{Adv}_{\mathscr{A}}^{\mathsf{EO\text{-}Anonymity}}(1^{\kappa}) = \frac{1}{2}$*, for all PPT adversaries* $\mathscr{A}$*.*

Verifiability

The verifiability ensures that EU can check whether the transformed ciphertext made by HCS is correct. That is, an authorized EU can check the accuracy of the search, transform and signature verification algorithms done by HCS. Specifically, given a challenge ciphertext for the EMR file $emr^{\star}$, the malicious HCS cannot create a transformed ciphertext that gives a plaintext not in the set $\{emr^{\star}, \perp\}$ and passes the verification mechanism. The model is formulated by a security game $\mathsf{Game}_{\mathscr{A}}^{\mathsf{verifiability}}$, presented in Figure 4.2, between an authorized EU $\mathscr{C}$ and the malicious HCS $\mathscr{A}$. In this game, $\widetilde{\mathcal{O}} := \{\mathcal{O}_{\mathcal{SKG}}, \mathcal{O}''_{\mathcal{TG}}, \mathcal{O}_{\mathcal{CG}}, \mathcal{O}_{\mathcal{ER}}\}$ and

- *Token generation oracle* $\mathcal{O}''_{\mathcal{TG}}(A_d, \Gamma_t)$: on input a decryption attribute set A_d and a keyword policy Γ_t, it computes $[\mathcal{DK}_{A_d}, \widetilde{\mathcal{TD}}_{\Gamma_t}, \mathcal{TK}_{A_d}, \mathcal{TD}_{\Gamma_t^{\circ}}, \mathcal{TDK}]$ and sends $token := \langle \mathcal{TD}_{\Gamma_t^{\circ}}, \mathcal{TK}_{A_d} \rangle$ to the adversary $\mathscr{A}$. Next, it stores the tuple $[\![A_d, \Gamma_t, token, \mathcal{TDK}]\!]$ in table $\mathsf{Tab}''_{\mathcal{ER}}$.

The adversary $\mathscr{A}$ wins the game If $\mathsf{Verify\text{-}Retrieve}(\mathcal{PP}, \mathcal{CT}_{tr}, \mathcal{TDK}) \notin \{emr^{\star}, \perp\}$, where $\mathcal{TDK}$ is taken from the tuple $[\![A_d, \Gamma_t, token, \mathcal{TDK}]\!]$ which is in table $\mathsf{Tab}''_{\mathcal{ER}}$. Note that if the tuple is not in $\mathsf{Tab}''_{\mathcal{ER}}$, it can be generated by querying the oracle $\mathcal{O}''_{\mathcal{TG}}$ with the input (A_d, Γ_t).

Definition 11. *The MediCare is verifiable if the advantage of $\mathscr{A}$ in the game $\mathsf{Game}_{\mathscr{A}}^{\mathsf{verifiability}}$, defined as $\mathsf{Adv}_{\mathscr{A}}^{\mathsf{verifiability}}(1^{\kappa}) \stackrel{\text{def}}{=} Pr[\mathscr{A}\ wins]$, is negligible, for all PPT adversaries $\mathscr{A}$.*

Keyword Privacy

This guarantees that the ciphertext reveals nothing about the keyword values it contains. The HCS (referred to as Type-1 adversary) cannot determine which ciphertext uses which set of keyword values without knowledge of the corresponding "valid" trapdoor. A trapdoor is considered valid if it contains a keyword policy that accepts the set of keywords linked to the ciphertext. The Type-2 adversary, which refers to an authorized entity that is ignorant of $\mathcal{CSK}$, is unable to identify which ciphertext utilizes which set of keyword values, even though the adversary knows the corresponding valid trapdoors. Security in terms of keyword privacy is defined subsequently as indistinguishability against chosen keyword set attack (in short, IND-CKA).

IND-CKA Security for Type-1 Adversary

The scenario is depicted in the following security game $\mathsf{Game}^{\mathsf{IND\text{-}CKA}}_{\mathsf{Type\text{-}1}}$ (described in Figure 4.3), which poses a challenger against a Type-1 adversary $\mathscr{A}$. Here, $\widehat{\mathcal{O}} := \{\mathcal{O}_{\mathcal{SKG}}, \mathcal{O}'''_{\mathcal{TG}}, \mathcal{O}_{search}\}$ and

- *Token generation oracle* $\mathcal{O}'''_{\mathcal{TG}}(A_d, \Gamma_t)$: on input a decryption attribute set A_d and a keyword policy Γ_t with the condition that $\Gamma_t(W_0^\star) = 0 \wedge \Gamma_t(W_1^\star) = 0$, it computes the EMR retrieval request $token := \langle \mathcal{TD}_{\Gamma_t^\circ}, \mathcal{TK}_{A_d} \rangle$ and sends the same to $\mathscr{A}$.

- *Search oracle* $\mathcal{O}_{search}(\mathcal{CT}, \Gamma_t)$: Taking a ciphertext $\mathcal{CT}$ and a keyword policy Γ_t obeying the condition $\Gamma_t(W_0^\star) = 0 \wedge \Gamma_t(W_1^\star) = 0$, it returns the output of Search algorithm.

IND-CKA Security for Type-2 Adversary

The security game $\mathsf{Game}^{\mathsf{IND\text{-}CKA}}_{\mathsf{Type\text{-}2}}$ (given in Figure 4.3) involves a model that consists of a challenger $\mathscr{C}$ and a Type-2 adversary $\mathscr{A}$.
Here, $\mathcal{O}_5 := \{\mathcal{O}_{\mathcal{SKG}}, \mathcal{O}'_{\mathcal{TG}}, \mathcal{O}'_{search}\}$ and $\mathcal{O}_6 := \{\mathcal{O}_{\mathcal{SKG}}, \mathcal{O}'_{\mathcal{TG}}, \mathcal{O}''_{search}\}$,

- *Search oracle* $\mathcal{O}'_{search}(\mathcal{CT}, \Gamma_t)$: Taking a ciphertext $\mathcal{CT}$ and a keyword policy Γ_t, it returns the output of Search algorithm.

- $\mathcal{O}''_{search}(\mathcal{CT}, \Gamma_t)$: This is same as $\mathcal{O}'_{search}(\mathcal{CT}, \Gamma_t)$, except that $\mathscr{A}$ is not permitted to query $\mathcal{O}''_{search}$ using the input $(\mathcal{CT}^\star, \Gamma_t)$ satisfying $\Gamma_t(W_i^\star) = 1$.

For $\mathsf{k} \in \{1, 2\}$, the adversary wins the game $\mathsf{Game}^{\mathsf{IND\text{-}CKA}}_{\mathsf{Type\text{-}k}}$ if $i' = i$. The adversary's advantage in $\mathsf{Game}^{\mathsf{IND\text{-}CKA}}_{\mathsf{Type\text{-}k}}$ is defined as $\mathsf{Adv}^{\mathsf{IND\text{-}CKA}}_{\mathsf{Type\text{-}k}}(1^\kappa) \stackrel{\text{def}}{=} \big|\Pr[i' = i] - 1/2\big|$.

Definition 12. *The MediCare demonstrates IND-CKA security if* $\mathsf{Adv}^{\mathsf{IND\text{-}CKA}}_{\mathsf{Type\text{-}1}}(1^\kappa)$ *and* $\mathsf{Adv}^{\mathsf{IND\text{-}CKA}}_{\mathsf{Type\text{-}2}}(1^\kappa)$ *are negligible, for all PPT Type-1 and Type-2 adversaries, respectively.*

4.3 MediCare Construction

Let p be a prime, and $\mathbb{G}$ and $\mathbb{G}_T$ be cyclic groups of order p. We utilize a bilinear pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ and seven collision-resistant hash functions $\{H_i\}_{i=1}^7$, described as $H_1 : \{0,1\}^* \to \mathbb{G}, H_2 : \{0,1\}^* \to \mathbb{G}, H_3 : \mathbb{G}_T \to \mathbb{Z}_p^*, H_4 : \mathbb{G}_T \to \{0,1\}^{\ell_{H_4}}$, $H_5 : \{0,1\}^* \to \{0,1\}^{\ell_{H_5}}, H_6 : \{0,1\}^* \to \mathbb{Z}_p^*, H_7 : \{0,1\}^* \to \mathbb{Z}_p^*$, where H_1, H_2 are two independent hash functions, and H_6 and H_7 are two independent hash functions.
Assume $U_e = \{0,1\}^* = U_s$. Both EO and EU may hold the same attribute string, for instance $\mathsf{hospitalABC}$. In this case, $\mathsf{EO||hospitalABC}$ is treated as EO

attribute whereas EU||hospitalABC is considered as EU attribute. Hence, we assume that signing and encryption attributes are all different. Using a hash function $H_1 : \{0,1\}^* \rightarrow \mathbb{G}$, the signing and encryption attributes are converted to random elements of $\mathbb{G}$. Let $U_t = \mathbb{Z}_p^*$. Each keyword string can be mapped to a $\mathbb{Z}_p^*$ element via a hash function of the type $H_t : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$. Hence, for brevity, we treat keywords are elements of $\mathbb{Z}_p^*$. Our encryption policy is represented by a DNF Boolean formula, which means that the computation of the transformed ciphertext component $\mathcal{X}_3$ requires only 2 pairings which is independent of the number of encryption attributes.

System Setup

- KGA-Setup(1^κ). Let $\Sigma := \langle p, \mathbb{G}, \mathbb{G}_T, e\rangle$ be a bilinear pairing tuple. Choose the plaintext space as $\mathcal{M} := \{0,1\}^{\ell_{pt}}$ and a key derivation function $\mathsf{KDF} : \mathbb{G}_T \rightarrow \{0,1\}^{\ell_{pt}}$. The KGA public key $\mathcal{KPK}$ and system master secret $\mathcal{MK}$ are generated in the following way.
  Choose $\alpha \xleftarrow{u} \mathbb{Z}_p^*$ and compute $g_T := e(g,g)^\alpha$,
  Select $g, h, g_1, g_2, \ldots, g_{10} \xleftarrow{u} \mathbb{G}$, and
  Set $\mathcal{KPK} := \langle \Sigma, g_T, g, h, \{g_i\}_{i=1}^{10}, \mathcal{M}, \mathsf{KDF}, \{H_i\}_{i=1}^{7}\rangle$, $\mathcal{MK} := g^\alpha$.

- HCS-Setup($\mathcal{KPK}$). The HCS's public and secret keys $\mathcal{CPK}$ and $\mathcal{CSK}$, respectively, are computed in the following way.
  Choose $\beta \xleftarrow{u} \mathbb{Z}_p^*$, calculate $Y := h^\beta$, and define $\mathcal{CPK} := Y, \mathcal{CSK} := \beta$.

- TGA-Setup($\mathcal{KPK}$). The TGA public key $\mathcal{TPK}$ and TGA secret key $\mathcal{TSK}$ are computed as follows.
  Choose $\gamma, \varpi_1, \varpi_2, \varpi_3, \varpi_4 \xleftarrow{u} \mathbb{Z}_p^*$,
  Calculate $h_T := e(g, g_{10})^\gamma, h_1 := g^{\varpi_1}, h_2 := g^{\varpi_2}, h_3 := g^{\varpi_3}, h_4 := g^{\varpi_4}$,
  Set $\mathcal{TPK} := \langle h_T, h_1, h_2, h_3, h_4\rangle, \mathcal{TSK} := \langle \gamma, \varpi_1, \varpi_2, \varpi_3, \varpi_4\rangle$.

Set $\mathcal{PP} := \langle \mathcal{KPK}, \mathcal{CPK}, \mathcal{TPK}\rangle$.

Registration

- sKeyGen($\mathcal{PP}, \mathcal{MK}, A_s$). This algorithm produces a signing key $\mathcal{SK}_{A_s} := \langle A_s, S, S_0, \{S_x\}_{x \in A_s}\rangle$ where
  $r' \xleftarrow{u} \mathbb{Z}_p^*$, $S := g^\alpha h^{r'}, S_0 := g^{r'}, S_x = H_1(x)^{r'}$.

- dKeyGen($\mathcal{PP}, \mathcal{MK}, A_d$). This algorithm creates a decryption key $\mathcal{DK}_{A_d} := \langle A_d, D, D_0, \{D_y\}_{y \in A_d}\rangle$ where
  $r \xleftarrow{u} \mathbb{Z}_p^*$, $D := g^\alpha Y^r, D_0 := g^r, D_y := H_1(y)^r$.

Ciphertexts Uploading

- $\mathsf{Signcrypt}(\mathcal{PP}, \mathcal{SK}_{A_s}, \Gamma_s, \Gamma_e, W, emr)$. Select a signing policy $\Gamma_s := (\mathbf{M}_s, \rho_s)$ satisfying $\Gamma_s(A_s) = 1$. Here, $\mathbf{M}_s$ represents a matrix of dimension $\ell_s \times n_s$.
  Next, choose an encryption policy $\Gamma_e := B_1 \vee B_2 \vee \cdots \vee B_m$ and a keyword set $W := \{[\mathcal{W} : w]\}$(where the set $W^\circ := \{\mathcal{W}\}$ represents generic keywords names).
  Since $\Gamma_s(A_s) = 1$, calculate $\vec{a} := (a_1, a_2, \ldots, a_{\ell_s}) \leftarrow \mathsf{Reconstruct}(\mathbf{M}_s, \rho_s, A_s)$ satisfying $\vec{a} \cdot \mathbf{M}_s = \vec{1}_{n_s}$, i.e., $\sum_{i\in[\ell_s]} a_i \cdot \vec{M}_s^{(i)} = \vec{1}_{n_s}$ and $a_i = 0\ \forall i$ satisfying $\rho_s(i) \notin A_s$. For the matrix $\mathbf{M}_s$, the ith row is denoted as $\vec{M}_s^{(i)}$. Sample $(b_1, b_2, \ldots, b_{\ell_s}) \xleftarrow{\text{u}} \{(b_1, b_2, \ldots, b_{\ell_s}) \in \mathbb{Z}_p^{\ell_s} | \sum_{i\in[\ell_s]} b_i \cdot \vec{M}_s^{(i)} = \vec{0}_{n_s}\}$. The ciphertext $\mathcal{CT} := \langle \Delta_s, \Delta_e, \Delta_k, \mathsf{tag2}, E_0, \eta \rangle$ is computed as follows.
  - Pick $\theta \xleftarrow{\text{u}} \mathbb{Z}_p^*$, and encode the EMR emr in the following way.

    $$\delta := H_3(g_T^\theta), \mathsf{key} := \mathsf{KDF}(g_T^\theta), k_T := h_T^\theta, ct := emr \oplus \mathsf{key}, \mathsf{tag2} := H_5(\delta || ct)$$

  - Choose $\delta', \delta'', o_2 \xleftarrow{\text{u}} \mathbb{Z}_p^*$, and generate the signature components as

    $$\sigma' := g^{\delta'}, \sigma'' := h^{\delta''}, \mathsf{tag1} := H_4(g_T^{1/\delta} \cdot e(\sigma', Y)^{\delta''}),$$

    $$\sigma := S^{1/\delta}\big(g_1^{o_1} g_2\big)^\theta \prod_{i\in[\ell_s]} \big(S_{\rho_s(i)}^{a_i/\delta} \cdot H_1(\rho_s(i))^{o_2 b_i}\big), \sigma_i := S_0^{a_i/\delta} g^{o_2 b_i}, \forall i \in [\ell_s],$$

    where $o_1 := H_6(ct || \mathsf{tag1} || \Gamma_e || \Gamma_s || W^\circ)$
    The signature components $\Delta_s := \langle \Gamma_s, \sigma', \sigma'', \mathsf{tag1}, \sigma, \{\sigma_i\}_{i\in[\ell_s]} \rangle$.
  - Select $\theta_1, \theta_2, \ldots, \theta_m \xleftarrow{\text{u}} \mathbb{Z}_p^*$. Then, create the encryption components as

    $$E := g^{\theta \cdot H_3(e(\sigma', Y)^{\delta''})}, E_{i1} := g^{\theta_i}, E_{i2} := h^\theta \big(\prod_{y\in B_i} H_1(y)\big)^{\theta_i}$$

    The encryption components $\Delta_e := \langle \Gamma_e, ct, E, \{E_{i1}, E_{i2}\}_{i\in[m]} \rangle$.
  - Pick $t_\mathcal{W}, \pi_{\mathcal{W}1}, \pi_{\mathcal{W}2} \xleftarrow{\text{u}} \mathbb{Z}_p^*$, for all $\mathcal{W} \in W$. Then, calculate the components of keyword index
    $\Delta_k := \langle W^\circ, k_T, \{K_{\mathcal{W}1}, K_{\mathcal{W}2}, K_{\mathcal{W}3}, K_{\mathcal{W}4}, L_{\mathcal{W}1}, L_{\mathcal{W}2}\}_{\mathcal{W}\in W^\circ} \rangle$ as

    $$K_{\mathcal{W}1} := h_1^{t_\mathcal{W} - \pi_{\mathcal{W}1}}, K_{\mathcal{W}2} := h_2^{\pi_{\mathcal{W}1}}, K_{\mathcal{W}3} := h_3^{t_\mathcal{W} - \pi_{\mathcal{W}2}}, K_{\mathcal{W}4} := h_4^{\pi_{\mathcal{W}2}},$$

    $$L_{\mathcal{W}1} := (g_6^w g_7)^{t_\mathcal{W}} g_8^{-\theta}, L_{\mathcal{W}2} := (g_6^w g_7)^{t_\mathcal{W}} g_9^{-\theta}$$

EO | **HCS**

$\mathcal{CT} \leftarrow \mathsf{Signcrypt}(\mathcal{PP}, \mathcal{SK}_{A_s}, \Gamma_s, \Gamma_e, W, emr)$
where $\mathcal{CT} := \langle \Delta_s, \Delta_e, \Delta_k, \mathsf{tag2}, E_0, \eta \rangle$ $\xrightarrow{\mathcal{CT}}$

Compute $\xi := H_7(\Delta_s||\Delta_e||\Delta_k||\mathsf{tag2})$,
$\breve{E} := E^{1/H_3(e(\sigma',\sigma'')^{\mathcal{CSK}})}$,
verify $e(E_0, g) \stackrel{?}{=} e\big(g_3^{\xi} g_4^{\eta} g_5, \breve{E}\big)$
this verifies the integrity of the ciphertext.

$\xleftarrow{\perp}$ If this is not true, return $\perp$
Otherwise, obtain
$\varphi \xleftarrow{\mathrm{u}} \mathbb{Z}_p^*$,
$(\lambda_1'', \ldots, \lambda_{\ell_s}'') \leftarrow \mathsf{Share}(\mathbf{M}_s, \rho_s, \varphi)$,
$o_1 := H_6(ct||\mathsf{tag1}||\Gamma_e||\Gamma_s||W^{\circ})$,

$$\mathcal{Y}_3 := \left(\frac{e(\sigma, g^{\varphi})}{e\big(g_1^{o_1} g_2, (\breve{E})^{\varphi}\big) \cdot \prod_{i \in [\ell_s]} e\big(h^{\lambda_i''} H_1(\rho_s(i))^{\varphi}, \sigma_i\big)} \right)^{1/\varphi}$$

verify $\mathsf{tag1} \stackrel{?}{=} H_4(\mathcal{Y}_3 \cdot e(\sigma', \sigma'')^{\mathcal{CSK}})$
this checks the validity of the EO's signature.

$\xleftarrow{\perp}$ If this is not true, return $\perp$
Else, HCS accepts $\mathcal{CT}$ for storage and
stores $\mathcal{CT}_u := \langle \Delta_e, \Delta_k, \mathsf{tag2}, E_0, \eta, \breve{E}, \mathsf{tag} \rangle$, where $\mathsf{tag} := H_4(\mathcal{Y}_3)$

Figure 4.4: EMR storage phase

– Choose $\eta \xleftarrow{\mathrm{u}} \mathbb{Z}_p^*$, compute $E_0 := (g_3^{\xi} g_4^{\eta} g_5)^{\theta}$,
where $\xi := H_7(\Delta_s||\Delta_e||\Delta_k||\mathsf{tag2})$,
Set the ciphertext $\mathcal{CT} := \langle \Delta_s, \Delta_e, \Delta_k, \mathsf{tag2}, E_0, \eta \rangle$.

Now, EO outsources the ciphertext $\mathcal{CT}$ to HCS. Then, HCS accepts $\mathcal{CT}$ for storage if the EO is legitimate. That is, the EO has a signing key for the signing attribute set which satisfies the signing policy associated with $\mathcal{CT}$. Figure 4.4 depicts the EMR storage phase.

EMR Retrieval Token Generation

- $\mathsf{TrapGen}(\mathcal{PP}, \mathcal{TSK}, \Gamma_t)$. The keyword policy is defined as $\Gamma_t := (\mathbf{M}_t, \rho_t^{\circ}, \{w_{\rho_t^{\circ}(i)}\}_{i \in [\ell_t]})$ (where $\mathbf{M}_t$ represents a matrix of dimension $\ell_t \times n_t$, the rows of $\mathbf{M}_t$ are mapped to generic keyword names via the function ρ_t° and the associated keyword value is denoted as $\{w_{\rho_t^{\circ}(i)}\}_{i \in [\ell_t]}$). The trapdoor $\widetilde{\mathcal{TD}}_{\Gamma_t}$ is generated as follows.

  – Pick $f, f' \xleftarrow{\mathrm{u}} \mathbb{Z}_p^*$, and compute $T_1 := g^f, T_2 := h^{f'}, \mathsf{tt} := e(T_1, Y)^{f'}$,

  – Obtain $(\vartheta_1, \ldots, \vartheta_{\ell_t}) \leftarrow \mathsf{Share}(\mathbf{M}_t, \rho_t^{\circ}, \gamma \cdot H_3(\mathsf{tt}))$,

  – Pick $\breve{r}_i, \breve{r}_i' \xleftarrow{\mathrm{u}} \mathbb{Z}_p^*$, for all $i \in [\ell_t]$, and generate
  $T_{i1} := g_{10}^{\vartheta_i} \cdot g_8^{\varpi_1 \varpi_2 \breve{r}_i + \varpi_3 \varpi_4 \breve{r}_i'}, T_{i2}' := H_2(\mathsf{tt}||\Gamma_t^{\circ}||\vec{M}_t^{(i)}) \cdot g_9^{\varpi_1 \varpi_2 \breve{r}_i + \varpi_3 \varpi_4 \breve{r}_i'}$,
  $T_{i3} := g^{\varpi_1 \varpi_2 \breve{r}_i + \varpi_3 \varpi_4 \breve{r}_i'}$,
  $V_{i1} := (g_6^{w_{\rho_t^{\circ}(i)}} g_7)^{-\breve{r}_i \varpi_1}, V_{i2} := (g_6^{w_{\rho_t^{\circ}(i)}} g_7)^{-\breve{r}_i \varpi_2}$,
  $V_{i3} := (g_6^{w_{\rho_t^{\circ}(i)}} g_7)^{-\breve{r}_i' \varpi_3}, V_{i4} := (g_6^{w_{\rho_t^{\circ}(i)}} g_7)^{-\breve{r}_i' \varpi_4}$,

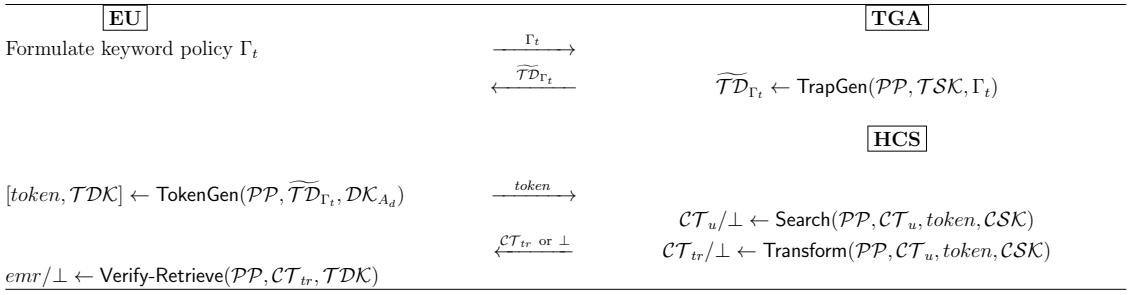


Figure 4.5: EMR retrieval phase

– Set $\widetilde{\mathcal{TD}}_{\Gamma_t} := \langle \Gamma_t, T_1, T_2, \{T_{i1}, T'_{i2}, T_{i3}, V_{i1}, V_{i2}, V_{i3}, V_{i4}\}_{i\in[\ell_t]} \rangle$.

- $\mathsf{TokenGen}(\mathcal{PP}, \widetilde{\mathcal{TD}}_{\Gamma_t}, \mathcal{DK}_{A_d})$. Here $\mathcal{DK}_{A_d} := \langle A_d, D, D_0, \{D_y\}_{y\in A_d} \rangle$. The EMR retrieval request *token* and the secret transformation decryption key $\mathcal{TDK}$ are generated as follows.

  – Select $\tau_1, \tau_2, \tau_3 \xleftarrow{\mathrm{u}} \mathbb{Z}_p^*$, obtain $(\breve{\vartheta}_1, \ldots, \breve{\vartheta}_{\ell_t}) \leftarrow \mathsf{Share}(\mathbf{M}_t, \rho_t^\circ, \tau_3/\tau_1)$, and compute

$$T_{i2} := h^{\breve{\vartheta}_i} \cdot T'_{i2}$$

  Set $\mathcal{TD}_{\Gamma_t^\circ} := \langle \Gamma_t^\circ, T_1, T_2, \{T_{i1}, T_{i2}, T_{i3}, V_{i1}, V_{i2}, V_{i3}, V_{i4}\}_{i\in[\ell_t]} \rangle$.

  – Randomize the decryption key $\mathcal{DK}_{A_d}$ as

$$D' := (D \cdot h^{\tau_3})^{1/\tau_2}, D'_0 := D_0^{1/\tau_2}, D'_y := D_y^{1/\tau_2}$$

  Set $\mathcal{TK}_{A_d} := \langle A_d, D', D'_0, \{D'_y\}_{y\in A_d} \rangle$.

  – Set $token := \langle \mathcal{TD}_{\Gamma_t^\circ}, \mathcal{TK}_{A_d} \rangle$, $\mathcal{TDK} := \langle \tau_1, \tau_2 \rangle$.

Remark 9. *The distribution of the EMR retrieval request token*

$$token := \left(\begin{array}{l} \mathcal{TD}_{\Gamma_t^\circ} := \left\langle \ \Gamma_t^\circ, T_1, T_2, \{T_{i1}, T_{i2}, T_{i3}, V_{i1}, V_{i2}, V_{i3}, V_{i4}\}_{i\in[\ell_t]} \ \right\rangle, \\ \mathcal{TK}_{A_d} := \left\langle \ A_d, D', D'_0, \{D'_y\}_{y\in A_d} \ \right\rangle \end{array} \right)$$

of a keyword policy $\Gamma_t := (\mathbf{M}_t, \rho_t^\circ, \{w_{\rho_t^\circ(i)}\}_{i\in[\ell_t]})$ *and a decryption attribute set* A_d *is of the form*

$$\left\{\begin{array}{l} T_1 := g^f, T_2 := h^{f'}, \\ T_{i1} := g_{10}^{\vartheta_i} \cdot g_8^{\varpi_1\varpi_2\breve{r}_i+\varpi_3\varpi_4\breve{r}'_i}, \\ T_{i2} := h^{\breve{\vartheta}_i} \cdot H_2(e(T_1,Y)^{f'}||\Gamma_t^\circ||\vec{M}_t^{(i)}) \cdot g_9^{\varpi_1\varpi_2\breve{r}_i+\varpi_3\varpi_4\breve{r}'_i}, \\ T_{i3} := g^{\varpi_1\varpi_2\breve{r}_i+\varpi_3\varpi_4\breve{r}'_i}, \\ V_{i1} := (g_6^{w_{\rho_t^\circ(i)}} g_7)^{-\breve{r}_i\varpi_1}, V_{i2} := (g_6^{w_{\rho_t^\circ(i)}} g_7)^{-\breve{r}_i\varpi_2}, \\ V_{i3} := (g_6^{w_{\rho_t^\circ(i)}} g_7)^{-\breve{r}'_i\varpi_3}, V_{i4} := (g_6^{w_{\rho_t^\circ(i)}} g_7)^{-\breve{r}'_i\varpi_4}, \\ D' := g^{\alpha/\tau_2} Y^{\bar{\bar{r}}} h^{\tau_3/\tau_2}, D'_0 := g^{\bar{\bar{r}}}, D'_y := (H_1(y))^{\bar{\bar{r}}} \end{array}\right\} \tag{4.1}$$

where $f, f', \breve{r}_i, \breve{r}'_i, \bar{\bar{r}}, \tau_1, \tau_2, \tau_3$ are random exponents (elements in $\mathbb{Z}_p^*$), ϑ_i (resp. $\breve{\vartheta}_i$) is the ith share of $\gamma \cdot H_3(e(T_1,Y)^{f'})$ (resp. τ_3/τ_1) with respect to the policy $(\mathbf{M}_t, \rho_t^\circ)$, $\langle \gamma, \varpi_1, \varpi_2, \varpi_3, \varpi_4 \rangle$ is TGA's secret key, α is system's master secret, and others are system public parameters.

We use this distribution of token in correctness and security analysis of MediCare. □

EMR Retrieval

- $\mathsf{Search}(\mathcal{PP}, \mathcal{CT}_u, token, \mathcal{CSK})$. The stored ciphertext is parsed as $\mathcal{CT}_u := \langle \Delta_e, \Delta_k, \mathsf{tag2}, E_0, \eta, \breve{E}, \mathsf{tag} \rangle$. The matching ciphertexts are identified by performing the steps given below.
  - Generate $\vec{a}' := (a'_1, a'_2, \ldots, a'_{\ell_t}) \leftarrow \mathsf{Reconstruct}(\mathbf{M}_t, \rho_t^\circ, W^\circ)$ and compute

$$\mathcal{X}_1 := e(\breve{E}, \prod_{i\in[\ell_t]} T_{i1}^{a'_i}) \cdot \prod_{i\in[\ell_t]} e(T_{i3}, L_{\rho_t^\circ(i)1})^{a'_i} \tag{4.2}$$

$$\mathcal{X}_2 := \prod_{i\in[\ell_t]} \left\{ e(K_{\rho_t^\circ(i)1}, V_{i2}) \cdot e(K_{\rho_t^\circ(i)2}, V_{i1}) \cdot e(K_{\rho_t^\circ(i)3}, V_{i4}) \cdot e(K_{\rho_t^\circ(i)4}, V_{i3}) \right\}^{a'_i} \tag{4.3}$$

  - Verify that $\mathcal{X}_1\mathcal{X}_2 \stackrel{?}{=} k_T^{H_3\left(e(T_1,T_2)^{\mathcal{CSK}}\right)}$. If it does not hold, the algorithm outputs $\perp$. Otherwise, the ciphertext $\mathcal{CT}_u$ matches $\mathcal{TD}_{\Gamma_t^\circ}$, i.e., $\Gamma_t^\circ(W^\circ) = 1$ (implicitly $\Gamma_t(W) = 1$). In this case, it outputs the stored ciphertext $\mathcal{CT}_u$.

- $\mathsf{Transform}(\mathcal{PP}, \mathcal{CT}_u, token, \mathcal{CSK})$. Here $\mathcal{CT}_u := \langle \Delta_e, \Delta_k, \mathsf{tag2}, E_0, \eta, \breve{E}, \mathsf{tag} \rangle$. It creates the transformed ciphertexts $\mathcal{CT}_{tr}$ as described below.
  - If $\mathsf{Search}(\mathcal{PP}, \mathcal{CT}_u, token, \mathcal{CSK}) \rightarrow \perp$ or $\Gamma_e(A_d) = 0$, output $\perp$. Otherwise, proceed further.

– Consider $\mathcal{X}_2$ from search algorithm and compute

$$\breve{\mathcal{X}}_1 := e\Big(\breve{E}, \prod_{i\in[\ell_t]} \big(T_{i2}\cdot H_2(e(T_1,T_2)^{\mathcal{CSK}}||\Gamma_t^{\circ}||\vec{M}_t^{(i)})^{-1}\big)^{a'_i}\Big)\cdot \prod_{i\in[\ell_t]} e\Big(T_{i3}, L_{\rho_t^{\circ}(i)2}\Big)^{a'_i} \tag{4.4}$$

$\mathcal{Y}_1 := \breve{\mathcal{X}}_1\mathcal{X}_2$

– As $\Gamma_e(A_d) = 1$, there exists a $j \in [m]$ such that $B_j \subseteq A_d$. Calculate

$$\begin{aligned}
\mathcal{X}_3 &= \frac{e(E_{j2}, D'_0)}{e(E_{j1}, \prod_{y\in B_j} D'_y)}\\
\mathcal{X}_4 &= e(\breve{E}, D')\\
\mathcal{Y}_2 &:= (\mathcal{X}_3)^{-\mathcal{CSK}}\cdot \mathcal{X}_4
\end{aligned}$$

– $\mathcal{CT}_{tr} := \langle \mathcal{Y}_1, \mathcal{Y}_2, ct, \mathsf{tag2}, \mathsf{tag}\rangle$.

Correctness of Search. One can see that if $\Gamma_t(W) = 1$, then

$$\begin{aligned}
\mathcal{X}_1 &= e\big(g, g_{10}\big)^{\theta\cdot\gamma\cdot H_3(tt)}\cdot \prod_{i\in[\ell_t]}\Big\{e\big(g, g_6^{w_{\rho_t^{\circ}(i)}} g_7\big)^{t_{\rho_t^{\circ}(i)}\cdot(\varpi_1\varpi_2\check{r}_i+\varpi_3\varpi_4\check{r}'_i)a'_i}\Big\}\\
\mathcal{X}_2 &= \prod_{i\in[\ell_t]}\Big\{e(g, g_6^{w_{\rho_t^{\circ}(i)}} g_7)^{-t_{\rho_t^{\circ}(i)}(\varpi_1\varpi_2\check{r}_i+\varpi_3\varpi_4\check{r}'_i)a'_i}\Big\}\\
\mathcal{X}_1\mathcal{X}_2 &= e\big(g, g_{10}\big)^{\theta\cdot\gamma\cdot H_3(tt)} = k_T^{H_3\left(e(T_1,T_2)^{\mathcal{CSK}}\right)}
\end{aligned}$$

Correctness of Transform. We can see that if $\Gamma_t(W) = 1$, then

$$\begin{aligned}
\breve{\mathcal{X}}_1 &= e\big(g, h\big)^{\theta\cdot\tau_3/\tau_1}\cdot \prod_{i\in[\ell_t]}\Big\{e\big(g, g_6^{w_{\rho_t^{\circ}(i)}} g_7\big)^{t_{\rho_t^{\circ}(i)}(\varpi_1\varpi_2\check{r}_i+\varpi_3\varpi_4\check{r}'_i)a'_i}\Big\}\\
\mathcal{X}_2 &= \prod_{i\in[\ell_t]}\Big\{e(g, g_6^{w_{\rho_t^{\circ}(i)}} g_7)^{-t_{\rho_t^{\circ}(i)}(\varpi_1\varpi_2\check{r}_i+\varpi_3\varpi_4\check{r}'_i)a'_i}\Big\}\\
\mathcal{Y}_1 &= \breve{\mathcal{X}}_1\mathcal{X}_2 = e\big(g, h\big)^{\theta\cdot\tau_3/\tau_1}
\end{aligned}$$

If $\Gamma_e(A_d) = 1$, then

$$\begin{aligned}
\mathcal{X}_3 &= e\big(g, h\big)^{\theta\cdot r/\tau_2}\\
\mathcal{X}_4 &= e\big(g, g\big)^{\alpha\cdot\theta/\tau_2} e\big(g, h\big)^{\beta r\theta/\tau_2} e\big(g, h\big)^{\theta\tau_3/\tau_2}\\
\mathcal{Y}_2 &= \mathcal{X}_3^{-\mathcal{CSK}}\cdot\mathcal{X}_4 = e\big(g, g\big)^{\alpha\theta/\tau_2} e(g, h)^{\theta\tau_3/\tau_2}
\end{aligned}$$

- $\mathsf{Verify\text{-}Retrieve}(\mathcal{PP}, \mathcal{CT}_{tr}, \mathcal{TDK})$. The secret transformation decryption key is parsed as $\mathcal{TDK} := \langle \tau_1, \tau_2 \rangle$. It checks the accuracy of $\mathcal{CT}_{tr}$ and gets back the EMR emr if $\mathcal{CT}_{tr}$ is valid.
  - Compute $\Lambda := \mathcal{Y}_1^{-\tau_1} \cdot \mathcal{Y}_2^{\tau_2}, \delta^\star := H_3(\Lambda)$.
  - Check whether $H_5(\delta^\star || ct) \stackrel{?}{=} \mathsf{tag2}$.
    If this does not hold, the error value $\perp$ is returned to confirm that the cloud deceptively returns a false search result. Else, it executes the following step.
  - Check whether $H_4(g_T^{1/\delta^\star}) \stackrel{?}{=} \mathsf{tag}$.
    If this does not hold, the error value $\perp$ is returned to confirm that the signature is not valid. Otherwise, it executes the subsequent step.
  - Output $ct \oplus \mathsf{KDF}(\Lambda) = emr$.

As shown in Figure 4.5, an EU can retrieve the required EMRs from HCS.

Correctness of Verify-Retrieve. If $\Gamma_t(W) = 1$ and $\Gamma_e(A_d) = 1$, then

$$\Lambda = \mathcal{Y}_1^{-\tau_1}\mathcal{Y}_2^{\tau_2} = \Big(e\big(g,h\big)^{\theta\tau_3/\tau_1}\Big)^{-\tau_1}\Big(e\big(g,g\big)^{\theta\alpha/\tau_2}\Big)^{\tau_2}\Big(e\big(g,h\big)^{\theta\tau_3/\tau_2}\Big)^{\tau_2} = g_T^{\theta}$$

$$\delta^\star = H_3(\Lambda) = H_3(g_T^\theta) = \delta$$

$$H_5(\delta^\star||ct) = H_5(\delta||ct) = \mathsf{tag2}$$

If $\Gamma_s(A_s) = 1$, then we can see that $\mathcal{Y}_3 = g_T^{1/\delta}$ and hence

$$\begin{aligned} H_4(g_T^{1/\delta^\star}) &= H_4(g_T^{1/\delta}) = H_4(\mathcal{Y}_3) = \mathsf{tag} \\ ct \oplus \mathsf{KDF}(\Lambda) &= emr \oplus \mathsf{KDF}(g_T^\theta) \oplus \mathsf{KDF}(\Lambda) = emr \end{aligned}$$

4.4 Security Proof

We provide the subsequent theorems to demonstrate MediCare's security. For better readability, the proofs are deferred to Appendix A. The following theorems assume that KDF is secure.

Lemma 5. *MediCare demonstrates IND-CCA2 security against PPT Type-1 adversary in the random oracle model, assuming the hardness of the DBDH problem.*

Lemma 6. *MediCare demonstrates IND-CCA2 security against PPT Type-2 adversary, assuming the hardness of the DBDH problem.*

The subsequent theorem is obtained by combining Lemma 5 and Lemma 6.

Theorem 6 (Data Confidentiality)**.** *MediCare provides IND-CCA2 security in the random oracle model, assuming the hardness of the DBDH problem.*

Theorem 7 (Data Unforgeability)**.** *MediCare demonstrates EUF-CMA security in the random oracle model assuming the hardness of the q-DHE problem, if the challenge signing policy has a maximum of q columns.*

Theorem 8 (EO Anonymity)**.** *MediCare preserves EO anonymity.*

Theorem 9 (Verifiability)**.** *MediCare is verifiable under the assumption that H_5 is a collision-resistant hash function.*

Lemma 7. *If the challenge keyword set has at most q keywords, then MediCare demonstrates IND-CKA security against PPT Type-1 adversary, assuming that q-2 and DLin problems are hard.*

Lemma 8. *MediCare provides IND-CKA security against PPT Type-2 adversary, assuming the hardness of the DBDH problem.*

The subsequent theorem is obtained by combining Lemma 7 and Lemma 8.

Theorem 10 (Chosen Keyword Set Attack Security)**.** *Suppose the challenge keyword set has at most q keywords. Then, MediCare is IND-CKA secure, under the assumption that q-2, DLin and DBDH problems are hard.*

4.5 Performance

The remainder of this section, as well as all the tables and figures, utilize the notations described below.

ℓ_e (resp. ℓ_s)	:	total number of attributes in an encryption (resp. signing) policy
$\|W\|$ or ς	:	size of a keyword set ascribed to a ciphertext
M_G (resp. E_G)	:	one multiplication (resp. exponentiation) execution time in $\mathbb{G}$
$\|A_d\|$	:	number of DU's attributes
M_T (resp. E_T)	:	one multiplication (resp. exponentiation) execution time on $\mathbb{G}_T$ element

I_T	:	one inversion execution time on $\mathbb{G}_T$ element
$\|U_e\|$ (resp. $\|U_s\|$)	:	number of attributes in encryption (resp. signing) attribute universe
P	:	one pairing computation execution time
ℓ_t	:	total number of keywords within a trapdoor/trapdoor keyword policy
H	:	one hash function calculation execution time
$\|A_s\|$	:	number of DO's attributes
$\|D_e\|$ (resp. $\|D_s\|$)	:	cardinality of dummy encryption (resp. dummy signing) attribute set
L (resp. L_T, L_p)	:	size of an element of $\mathbb{G}$ (resp, $\mathbb{G}_T$, $\mathbb{Z}_p$)
$L_{AES.CT}$	:	bit-length of one AES-ciphertext of a query keyword
m	:	total number of clauses in DNF encryption policy
$\|msg\|$	:	size of a plaintext/message
$T_{AES.Eec}$ (resp. $T_{AES.Dec}$)	:	one AES encryption (resp. AES decryption) execution time
S_p (resp. M_p, I_p)	:	one subtraction/addition (resp. multiplication, inversion) execution time in $\mathbb{Z}_p$
T_{kdf}	:	execution time of KDF
L_H	:	hash function's output length
$T_{R.Digest}$	:	one root digest execution time

Table 4.2: Functionality Comparison

Scheme	Encryption type	Signature type	Search mechanism	Keyword privacy	EO (Signer) anonymity	Search results verification	Constant decrypt cost	Encryption policy/ dKeyGen policy	Signing policy/ sKeyGen policy	Security
[52]	KP	KP	single keyword	×	✓	×	×	Boolean formula	Boolean formula	IND-CCA2, EUF-CMA, IND-CKA
[53]	CP	KP	single keyword	×	✓	×	×	Boolean formula	Boolean formula	IND-CCA2, EUF-CMA, IND-CKA
[66]	CP	SP	single keyword	✓	×	✓	×	Threshold policy	Threshold policy	IND-CCA2, EUF-CMA
[82]	CP	×	multi-keyword	✓	×	×	×	Boolean formula	×	IND-CPA, EUF-CMA
[3]	CP	SP	Boolean formula	✓	✓	✓	✓	Boolean formula	Boolean formula	IND-CCA2, EUF-CMA, EO-anonymity, IND-CKA, Verifiability
MediCare	CP	SP	Boolean formula	✓	✓	✓	✓	DNF	Boolean formula	IND-CCA2, EUF-CMA, EO-anonymity, IND-CKA, Verifiability

Table 4.3: Comparison of computation cost

Scheme	sKeyGen	dKeyGen	Signcrypt	TokenGen	Search Results Verify	Decrypt
[52]	$O(\|U_s\| \cdot \ell_s)E_G$	$O(\|U_e\| \cdot \ell_e)E_G$	$O(\ell_s + \|A_d\| + 13)E_G$ $+E_T + 5H$	$(2\ell_e + 3)E_G$ $+H$	–	$O(\ell_e)E_G$ $+6P + 4H$
[53]	$O(\|U_s\| \cdot \ell_s)E_G$	$(\|A_d\| + 4)E_G$	$O(\ell_e)E_G + E_T$ $+O(\ell_s \cdot \|A_s\|)M_G$ $+4H$	$O(\|A_d\|)E_G$ $+O(\|A_d\|)M_G$	–	$O(\|A_s\| + \ell_e)M_G$ $+4M_T + 6P$ $+I_T + 4H$
[66]	$O(\|A_s\| \cdot \|U_s\|)E_G$	$O(\|A_d\| \cdot \|U_e\|)E_G$	$O(\|A_s\| + \|D_s\| + 7)E_G$ $+E_T + 2H$	$T_{AES.Dec}$ $+T_{AES.Eec}$	$T_{R.Digest}$	$O(\|A_d\| + \|D_e\|)E_G$ $+7P + 3I_T + 2H$
[3]	$(\|A_s\| + 3)E_G$	$(3\|A_d\| + 4)E_G$	$O(\ell_e + \varsigma)S_p$ $+O(\ell_e + \varsigma)M_p + 2H$	$\ell_t M_G$	$2E_T + 2M_T$ $+2H$	T_{kdf}
MediCare	$(\|A_s\| + 3)E_G$	$(\|A_d\| + 3)E_G$	$O(m + \ell_s + \varsigma)E_G$ $+4E_T + O(m + \varsigma)M_G$ $+P + 6H + T_{kdf}$	$(\ell_t + \|A_d\| + 3)E_G$	$3E_T + I_T$ $+3H$	T_{kdf}

Table 4.4: Comparison of communication cost

Scheme	Signing Key Size	Decryption Key Size	Ciphertext Size	Token Size	Transformed Ciphertext Size
[52]	$O(\ell_s \cdot \|U_s\|)L$	$O(\ell_e \cdot \|U_e\|)L$	$(\|A_d\| + 9)L + \|msg\|$	$(2\ell_e + 2)L$	$6L + \|msg\|$
[53]	$O(\ell_s \cdot \|U_s\|)L$	$(\|A_d\| + 3)L$	$(3\ell_e + 8)L + \|msg\|$	$(\|A_d\| + 4)L$	$(\ell_e + 5)L + \|msg\|$
[66]	$O(\|A_s\| \cdot \|U_s\|)L$	$O(\|A_d\| \cdot \|U_e\|)L$	$5L + L_T$	$L_{AES.CT}$	$5L + L_T$
[3]	$(\|A_s\| + 2)L$	$(2\|A_d\| + 2)L$	$O(\ell_e + \ell_s + \varsigma)L + L_T + 2L_H + \|msg\|$	$O(\ell_t + \|A_d\|)L$	$3L_T + L_H + \|msg\|$
MediCare	$(\|A_s\| + 2)L$	$(\|A_d\| + 2)L$	$O(\ell_s + m + \varsigma)L + \|msg\|$	$O(\ell_t + \|A_d\|)L$	$2L_T + 2L_H + \|msg\|$

In this section, our MediCare and the schemes [53, 52, 66, 3] are compared both theoretically and empirically. The schemes [53, 52, 66] support only small attribute universe setting. In contrast, MediCare and [3] achieve a largeattribute universe structure, which means that KGA can introduce new attributes and keywords as needed rather than fixing all of them at system setup. We implement MediCare and [3] as small universe constructions for a fair comparison.

In Figure 4.6, we compare the computational efficiency of MediCare with [53, 52, 66, 3]. The EU executes the Signcrypt, TokenGen, Search Result Verify and Decrypt algorithms. Hence, the computing expenses of these algorithms are of utmost importance in scenarios with low processing power gadgets and the comparisons are made in terms of these algorithms only, in Table 4.3 and Figure 4.6. The communication overhead of MediCare and [53, 52, 66, 3] is illustrated in Table 4.4 and Figure 4.7.

- **Functionality Comparison.** Table 4.2 summarizes the functionality comparison of MediCare and [53, 52, 66, 82, 3]. The schemes [66, 3] and our MediCare employ the ciphertext-policy framework, whereas [52] makes use of the key-policy setting for both encryption and signing. The scheme in [53] combines CP-ABE and key-policy ABS (KP-ABS) mechanisms. In contrast, the signature component in [82] is not actually an ABS framework, because the unsigncryption process does not contain any signature verification mechanism. Besides, the major limitation of [82] is that the DU has to obtain the token from the corresponding DOs. To issue k different search queries, the DU has to

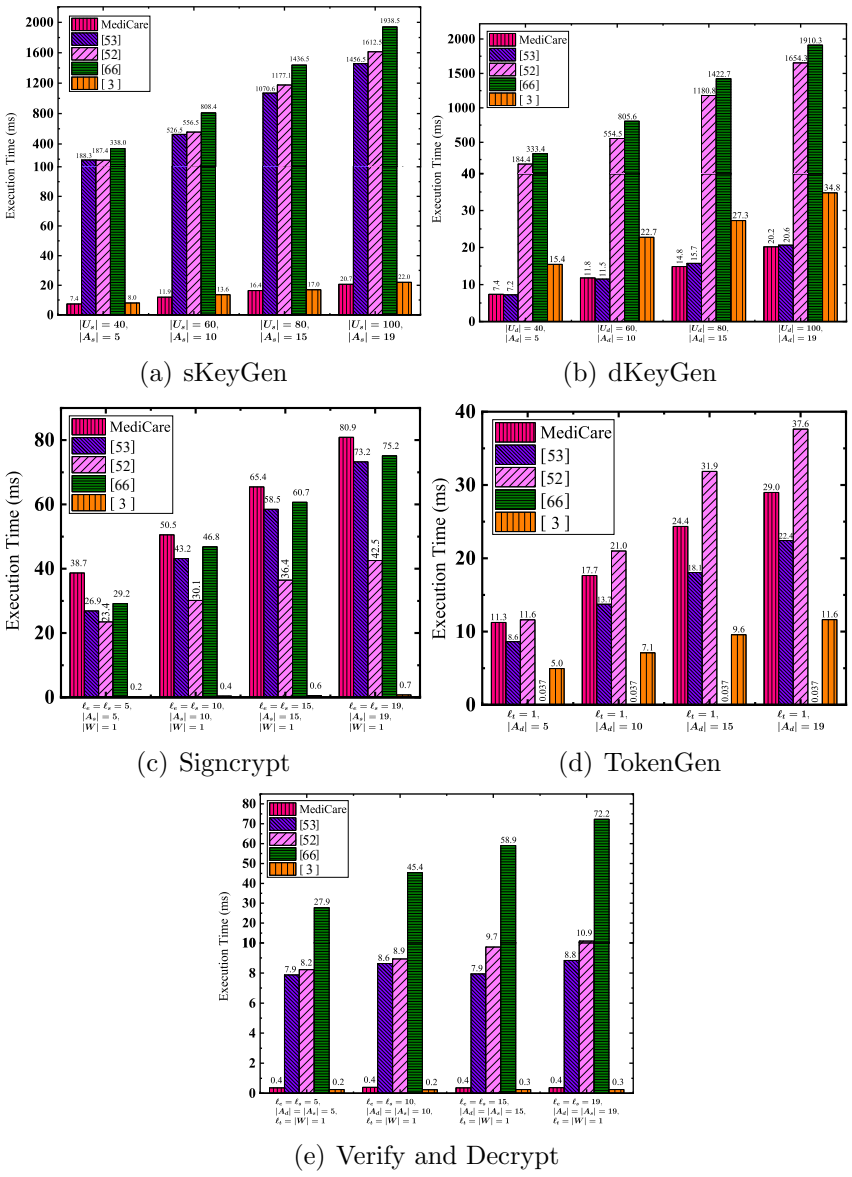


Figure 4.6: Execution time (in ms) of MediCare and [53, 52, 66, 3]

communicate k times with the respective DOs, which adds significant communication overhead on the user side. This type of framework is not suitable for EMR management systems. Because, for instance, to check Electrocardiogram (ECG) reports of all the patients from hospital X, the duty Cardiologist has to obtain tokens from all the heart patients in hospital X. This may not be a wise solution for data retrieval. Furthermore, the scheme [82] is secure against only IND-CPA, but a signcryption scheme should realize IND-CCA2 security. And, the computation cost of DUs is high and the scheme cannot offer search results verification functionality. These limitations make the scheme [82] less

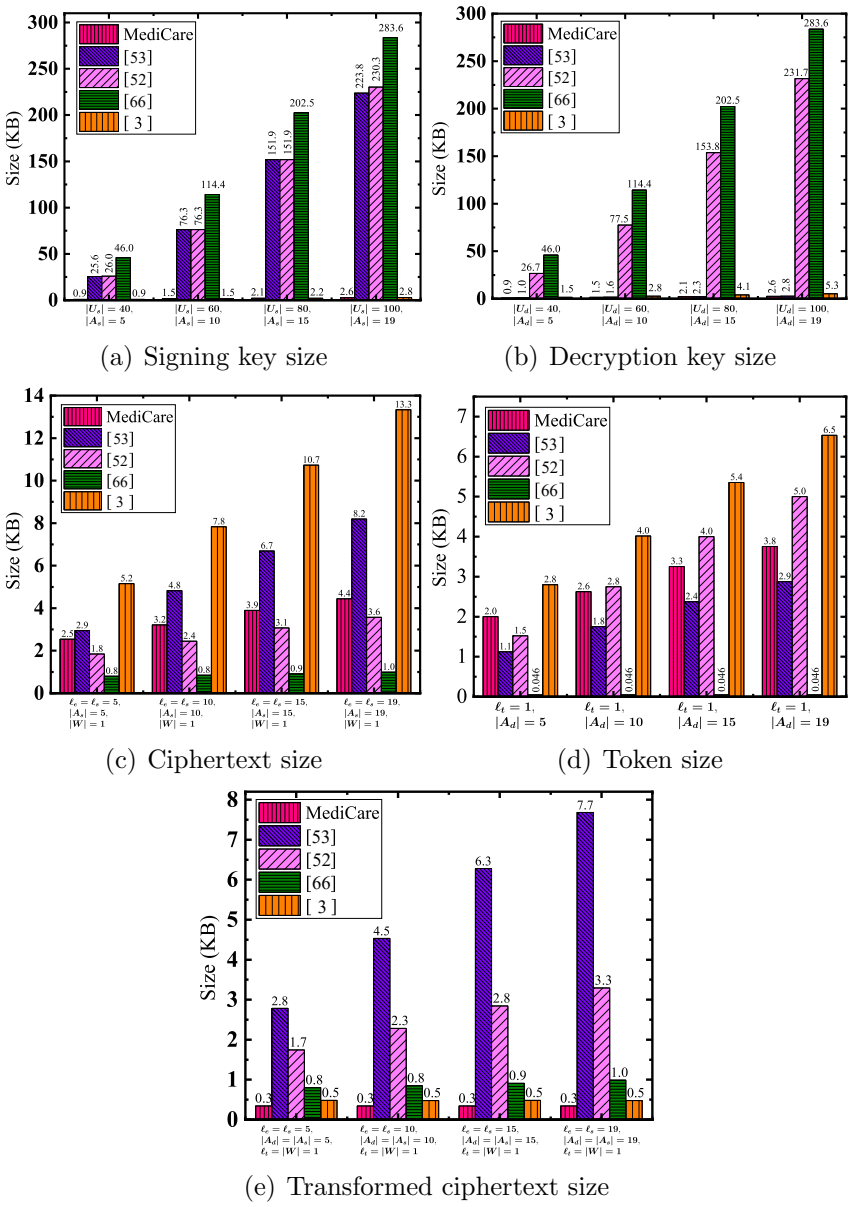


Figure 4.7: Communication cost (in KB) of MediCare and [53, 52, 66, 3]

practical for applications. Due to these reasons, [82] is not considered for experimental comparison. Regarding the data retrieval mode, MediCare and [3] support Boolean formula keyword search, which has great flexibility over those designs that provide single keyword search [53, 52, 66] or multi-keyword search [82]. The schemes [53, 52] (resp. [66, 82]) fail to achieve keyword privacy (resp. signer anonymity). While MediCare and [3] provide stronger signer privacy protections, the KP-ABS architecture makes the anonymity of signers in [53, 52] a less robust version. Observe that only [66, 3] and our Medicare enable

DUs to check the accrucy of search outcomes. Precisely, unlike [66], Medicare and [3] enable DUs verifying the precision of search, transform and signature verification algorithms executed by HCS without interacting with any authority. The access policies of all the schemes except [66] are Boolean formulas. In particular, the scheme in [66] uses less expressive threshold policies, and our MediCare employs expressive DNF policies to reduce the ciphertext size. The time consumed to decrypt a ciphertext is constant in our MediCare and [3], unlike [53, 52, 66, 82]. Since the data stored at HCS is retrieved by the EU, such as a doctor, using a lightweight end-device, like a smartphone, it is crucial for EMR applications to have lightweight constant decryption cost.

- **Experimental Setup.** The execution of the implementation is conducted on a laptop equipped with an Intel(R) Core(TM) i5-10300H CPU working at a frequency of 2.50 GHz and utilizing 8-GB of RAM. The laptop has 64-bit Ubuntu 20.04 LTS installed on Oracle VM VirtualBox - 6.1.22 memory of 2GB. PBC Library is explored, and a type-A elliptic curve with a prime number group order of 160 bits is selected for experimentation. The curve is $y^2 = x^3 + x$ over a 512-bit finite field. The running time in milliseconds (ms) of various algorithms of the proposed MediCare and [53, 52, 66, 3] is presented in Figure 4.6. The various communication costs, in kilobytes (KB), of MediCare and [53, 52, 66, 3] are analyzed in Figure 4.7. The design and development of an access policy have an impact on the amount of time required for execution as well as the cost of communication for attribute-based cryptosystems. To facilitate a comparison of the worst-case complexity, as proposed in reference [65], we employ AND-gate policies characterizedby the structure $c_1 \wedge c_2 \wedge \cdots \wedge c_j$, where $c_1, c_2, \ldots, c_j$ represent keywords or attributes. The schemes [53, 52, 66] support only single keyword search and hence we implement MediCare and [3] as single keyword searchable constructions for fair comparison. That is, $\ell_t = |W| = 1$ for all the schemes. Total 20 trials are conducted for each experiment, and bar graphs are used to show the average results.

- **Performance Analysis.** Below, we present performance of our MediCare compared to that of [53, 52, 66, 3].

  – In comparison to the methods described in [53, 52, 66], our MediCare implementation takes less time to generate the signing key and uses a smaller signing key size (as shown in Figures 4.6(a) and 4.7(a)). As demonstrated

in Tables 5.3 and 5.4, the significance is that in [53, 52, 66], the aforementioned measures increase as the cardinality of the signing attribute universe $|U_s|$ increases, whereas in MediCare, they are just affected by $|A_s|$.

- The time needed to generate the decryption key and the size of the decryption key in MediCare are approximately equal to that of [53] and slightly lower than those in [3]. But, it can be observed (from Figures 4.6(b) and 4.7(b)) that these measures in [52, 66] are extensive compared to our MediCare, since they increases as $|U_e|$ increases.
- Figure 4.6(c) depicts the duration of the signcryption process, whereas Figure 4.7(c) displays the size of the ciphertext for various keywords and attributes. As a result of using an online-offline framework, the signcryption time in [3] is much reduced. Precisely, it splits the process into offline signcryption and online signcryption. In the former, expensive operations (e.g., exponentiation, pairing) are executed while the latter utilizes only light computations such as hashing, XORing, modular multiplication, modular addition etc. However, the ciphertext size in [3] is larger compared to other schemes including MediCare. The ciphertext generation time in [53, 52, 66] is lower than our MediCare. And, the schemes [52, 66] achieve short ciphertext compared to Medicare. The following are the causes for this: (i) in order to maintain the scheme KGAs secure, we add certain extra calculations and group components, (ii) the ciphertext in [53, 52] contains only encryption and signature components, whereas our ciphertext contains keyword components in addition to the encryption and signature components, and (iii) the scheme [66] supports only less expressive threshold access policies, whereas MediCare works for expressive DNF policies.
- The structure of the token in [66] is totally different from [53, 52, 3] and MediCare. More specifically, first DU obtains AES key from the cloud and then encrypts the keyword using the key. The corresponding AES ciphertext of the keyword will be given to the cloud as a token. Hence, the duration of generating token and the size of token in [66] is significantly low (this can be seen from Figures 4.6(d) and 4.7(d)). However, the scheme [66] supports only single keyword search, whereas our MediCare realizes expressive keyword policy search. The single keyword search framework suggested in [66] cannot be extended to support

Boolean formula keyword search. The token generation time (resp. token size) in MediCare is lesser compared to that in [52] (resp. [3]) and longer compared to that in [53, 3] (resp. [53, 52]). This is because (i) we re-randomize some token components to counteract the KGAs on token, (ii) to facilitate outsourced unsigncryption and non-interactive search results verification, the decryption components included inside the token undergo appropriate re-randomization using two random exponents, and (iii) the scheme [3] makes use of online-offline mechanism to generate tokens, however, the token size is larger in [3] compared to MediCare. Table 4.2 shows that the schemes [53, 52, 66] cannot offer the keyword privacy, search results verifiability and outsourced unsigncryption functionalities simultaneously.

– According to Table 4.3, the process of decrypting MediCare only needs one KDF computation, similar to [3]. This is much lower than the decryption expense in [53, 52, 66], which grows as the number of signing and encryption attributes increases. Regardless of the policy's size, MediCare's search results verification time remains constant. Irrespective of the number of attributes and keywords, EU retrieves the plaintext with only one $\mathbb{G}_T$ inversion, three $\mathbb{G}_T$ exponentiations, one KDF calculation, and three hash function calculations. This is what EMR managing systems should strive for. As shown in Figure 4.6(e), our Verify-Retrieve procedure requires a mere 0.4 milliseconds; this is significantly less than the time required in [53, 52, 66] and is comparable to that of [3].

– Table 4.4 and Figure 4.7(e) exhibit that MediCare and [3] possess constant size transformed ciphertext, whereas it grows with the number of attributes in [53, 52, 66]. Precisely, HCS sends a transformed ciphertext of size 0.3 KB to EU in MediCare that is smaller when compared to that of the other schemes [53, 52, 66, 3].

4.6 Chapter Summary

In this chapter, we propose MediCare, a new attribute-based EMR storage and retrieval scheme that supports simultaneously (i) data and EO authenticity (ii) fine-grained data access control, (iii) EO anonymity, (iv) Boolean formula keyword search, (v) constant decryption cost for EUs, (vi) keyword privacy, and (vii) non-interactive search results verification. In order to demonstrate the security assur-

ance, we explicitly define and validate MediCare's security with respect to data confidentiality, data unforgeability, verifiability, EO anonymity, and keyword privacy. The efficiency and practicality of the proposed MediCare are shown via the comparison of its features and performance.

Chapter 5

Searchable Attribute-Based Proxy Re-encryption: Keyword Privacy, Verifiable Expressive Search and Outsourced Decryption

Data storage and data searching mechanisms were considered in the earlier chapters. This chapter presents the data sharing method along with data storage and searching, which was not accomplished by the schemes discussed in the preceding chapters. The data sharing mechanism enables a recipient to share the encrypted data received from the data owner with another recipient without decryption. Furthermore, the schemes in the previous chapters do not allow for the updating of the keyword set, but the design proposed in this chapter does.

In this chapter, we address the open problem posed by Ge et al. in 2020, which was to design a new ABPRE-KS scheme for enabling more expressive keyword search. ABPRE-KS exhibits promising potential in facilitating data searching and sharing through the implementation of one-to-many access control mechanism. However, existing ABPRE-KS schemes support single keyword search framework resulting in low search efficiency and poor user search experience. Also, maintaining keyword privacy, and protecting the outsourced ciphertexts and tokens from KGAs quite challenging in ABPRE-KS framework. To overcome these issues, we pro-

The work presented in this chapter is based on our published research article given below.
Sourav Bera, and Y Sreenivasa Rao. Searchable Attribute-Based Proxy re-encryption : Keyword Privacy, Verifiable Expressive Search and Outsourced Decryption. *SN Computer Science*, vol. 5, pp. 1-24, Springer, 2024. https://doi.org/10.1007/s42979-024-02646-2

pose an attribute-based proxy re-encryption scheme with Boolean keyword search (ABPRE-BKS) in the large attribute universe framework. Our scheme not only offers an efficient Boolean keyword search framework but also it enables constant decryption cost on the DU's side. The DU needs to perform only constant number of computations to recover both the original and the re-encrypted ciphertext. We define ABPRE-BKS and its security models. And, we prove that our scheme achieves ciphertext indistinguishability against adaptive chosen ciphertext attack, ciphertext and token indistinguishability against chosen keyword attack, and non-interactive verifiability. The efficiency of our proposed construction is demonstrated through a comparison of its functionalities and performance with the existing such schemes.

5.1 Introduction

Cloud services have emerged as a significant development in offering widespread and readily available access to a shared and adaptable collection of storage and computing resources. As a result, organizations and individuals are outsourcing their personal records (such as PHRs, identifying information etc.) to cloud servers. However, the task of ensuring the security and privacy of the personal documents that have been outsourced to cloud storage presents significant challenges.

ABE [75, 28, 34] is a promising technology which provides one-to-many access control and data confidentiality for outsourced documents. Each user in the ABE framework has a collection of attributes that acts as their public key. Attributes can be elements like a user's designation, affiliation, or other typical abstract credentials. Based on whether the secret key or the ciphertext is linked to an access policy, ABE is categorizedinto KP-ABE [75], [28] and CP-ABE [86], [5], [11]. A set of attributes are appended to the ciphertext and an access policy is associated to the secret key in the KP-ABE framework. The CP-ABE framework associates the secret key with a set of attributes and uses an access policy to create the ciphertext. The decryption process is successful in both architectures if the attribute set satisfies the access policy. In order to minimize the burden of decryption on the user's end, outsourced decryption privileges in ABE schemes [31, 41] have been proposed. Two crucial procedures are usually necessary for the DO after encrypted data is stored in the cloud: (i) searching for the stored data and (ii) sharing the stored data.

For instance, in a medical data sharing system, a covid-19 patient Harry wants to test whether he is covid-positive or not in a clinic. The clinic should be located within 4 km from London, the doctors should be senior doctor and ap-

pointed as a covid-specialist. Harry encrypts his medical record using an access policy $\Gamma = \{$*covid specialist* $\wedge$ *senior doctor* $\wedge$ *Location : within 4 km from London*$\}$ and a keyword set. The clinics satisfying Γ can decrypt the encrypted medical record. However, they couldn't get to the exact record by typing the keywords. Instead, the clinic needs to decrypt all the medical records satisfying Γ and then satisfy the keyword set to get the intended medical record. Also, in case, the clinic wants to share the medical record with some other junior doctors of the hospital located within 10 km from Nottigham, it needs to decrypt the encrypted record sent by Harry and thereafter encodes the medical record using an access policy $\Gamma' = \{$*covid specialist* $\wedge$ *junior doctor* $\wedge$ *Location : within 10 km from Nottigham*$\}$ and a keyword set. This method is not suitable since the clinic has to perform n pairs of encryption and decryption processes for n number of patients. Consequently, this system is extremely inefficient regarding data searching and sharing.

Suppose, in the above example, Harry attached the keyword set $W = \{$*covid-negative, senior doctor*$\}$ to his encrypted medical record. Now, the clinic founds Harry to be covid-negative after testing and wants to update the keyword set W to $W' = \{$*covid-positive, junior doctor*$\}$ without decrypting the medical record. As traditional ABE does not support keyword set updating mechanism, the clinic has to decrypt the record everytime before encrypting with new keyword set, which makes the system unfeasible.

To solve the above mentioned issues, ABPRE with keyword search schemes [48, 25, 37] are extremely prevalent, where the data searching, re-encryption and keyword set updating tasks are assigned to the cloud server (acts as proxy). Sometimes the semi-trusted cloud server may return incorrect search results that may lead to a wrong treatment to the patient. So, the difficulty of allowing a DU (DU) to independently check the correctness of search outcomes acquired from the cloud becomes a topic of considerable interest in ABPRE framework. Also, how to protect the keywords from KGAs on a ciphertext or token while performing data searching and sharing is another important aspect of security. Existing ABPRE schemes [48, 25, 37] support inefficient single keyword search framework which produces a large number of irrelavant douments, resulting in low search efficiency and poor user search experience. To improve the search efficiency in ABPRE framework, we propose an attribute-based proxy re-encryption scheme providing verifiable Boolean formula-based expressive search, keyword privacy and outsourced decryption.

Chapter Organization. The remainder of the chapter is organized as follows. Our scheme along with its security models is discussed in Section 5.2. In section 5.3, we

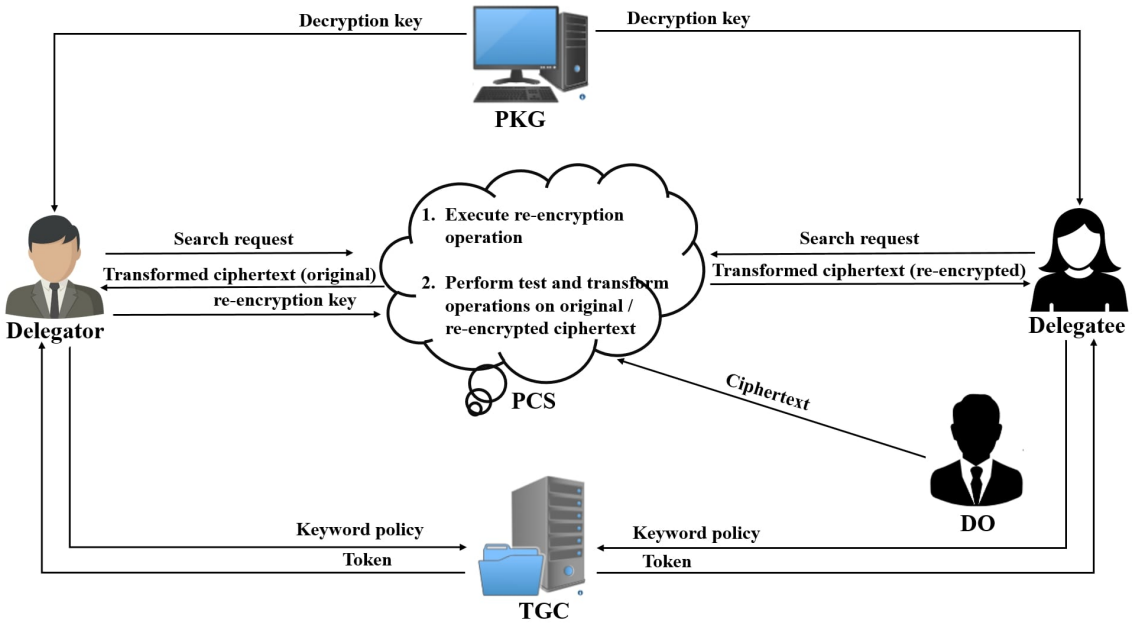


Figure 5.1: System Model of ABPRE-BKS

present the construction of our scheme and its correctness. Section 5.4 provides our ABPRE-BKS security proof. The performance evaluation of ABPRE-BKS is shown in Section 5.5. And, Section 5.6 provides the concluding remarks of the chapter.

5.2 Security of ABPRE-BKS

5.2.1 System Model

We provide the notations utilized in our ABPRE-BKS in Table 5.1. Our ABPRE-BKS mainly consists of six system entities: Private Key Generator (PKG), DO, Public Cloud Server (PCS), Trapdoor Generation Center (TGC), Delegator (recipient of the original ciphertext) and Delegatee (recipient of the re-encrypted ciphertext). It includes the following phases as shown in Figure 5.1.

System Initialization. In this phase, PKG, TGC and PCS create their own public key and secret key pairs, make the public parameters available to other entities in the system and keep the respective secret keys with themselves.

Registration. This phase is executed by PKG. After getting the registration requests from a DU (a delegator or a delegatee), PKG creates and sends the corresponding decryption keys to them.

Table 5.1: Notations used in ABPRE-BKS

PKG	:	Private Key Generator
PCS	:	Public Cloud Server
PP	:	system public parameters
mpk	:	master public key
cpk	:	PCS public key
msk	:	system master secret key
sk_S	:	decryption key corresponding to an attribute set S
rk	:	re-encryption key
csk	:	PCS secret key
sk_{tr}	:	transformed secret key
m	:	message
tpk	:	TGC public key
Γ_e	:	encryption policy
tsk	:	TGC secret key
Γ_t	:	keyword policy
CT	:	ciphertext
tok	:	token for Γ_t and sk_S
CT_m	:	matching ciphertext
CT_{tr}	:	transformed ciphertext
$\mathcal{U}$	:	encryption attribute universe
$\mathcal{U}_t$	:	keyword universe
$U\ (=\mathcal{U}\cup\mathcal{U}_t)$	:	attribute universe

Ciphertext Upload. DO executes this phase in order to encrypt his confidential data and upload the encrypted data (original ciphertext) to PCS.

Proxy Re-encryption. First, the delegator creates a re-encryption key and sends it to PCS. Then, PCS encrypts the original ciphertext using the re-encryption key and a new access policy, and creates the corresponding re-encrypted ciphertext, which cannot further be encrypted.

Ciphertext Retrieval. At first, by sending a Boolean query formula to TGC, a DU generates a search token and a transformed secret key with the help of his own secret key, and sends the search token to PCS. Next, PCS initiates the search process to retrieve all the matching original or re-encrypted ciphertexts and sends the corresponding partially decrypted ciphertexts to the DU.

Verification and Decryption. First, the DU verifies the accuracy of the search outcomes acquired from PCS using his transformed secret key, then decrypts the matching original or re-encrypted ciphertext and gets the original data.

5.2.2 Security Models

Our security model considers the security of the original and re-encrypted ciphertext, keyword privacy, and verifiability. An adversary may appear as a Type-1 adversary, denoting an unauthorized entity in possession of csk, or a Type-2 adversary, indicating an authorized entity unaware of csk. Our ABPRE-BKS is defined in the following algorithms.

- $\mathsf{PKG.Setup}(1^\kappa, U) \rightarrow (mpk, msk)$: Given input a security parameter 1^κ and an attribute universe U, PKG outputs a master public key mpk and a master secret key msk.
- $\mathsf{TGC.Setup}(mpk) \rightarrow (tpk, tsk)$: Taking mpk as input, TGC generates its public and secret keys, tpk and tsk, respectively.
- $\mathsf{PCS.Setup}(mpk) \rightarrow (cpk, csk)$: Taking mpk as input, PCS produces its public and secret keys, cpk and csk, respectively.

The attribute universe U is defined as $U = \mathcal{U} \cup \mathcal{U}_t$, where $\mathcal{U}$ and $\mathcal{U}_t$ are the universes of encryption attributes and keywords, respectively. The tuple (mpk, tpk, cpk) is denoted as the system public parameters PP. Let W be defined as a set containing keywords of the form $[\mathcal{W} : w]$, where $\mathcal{W}$ represents the generic keyword name and w represents the associated keyword value. Note that the entire keyword universe is divided into several categories and each category is identified with a suitable generic name. If $W = \{[\mathcal{W} : w]\}$ is a set of keywords, then we define W° as $W^\circ = \{\mathcal{W}\}$, i.e, W° is the set of generic keyword names corresponding to W.

- $\mathsf{KeyGen}(PP, msk, S) \rightarrow sk_S$: Given PP, msk and an attribute set $S \subset \mathcal{U}$ as input, it outputs a user decryption key sk_S for S.
- $\mathsf{Encrypt}(PP, m, \Gamma_e, W) \rightarrow CT$: Taking PP, an encryption policy Γ_e, a message $m \in \{0,1\}^\lambda$ and a keyword set W as input, DO generates an original ciphertext CT. The set W° and the policy Γ_e will be incorporated in CT.
- $\mathsf{Re\text{-}KeyGen}(PP, sk_S, \Gamma'_e, W') \rightarrow rk$: Taking PP, a user's decryption key sk_S, a keyword set W' and an encryption policy Γ'_e as input, a delegator produces a re-encryption key rk. The key rk can be used to transform a ciphertext under Γ_e and W to another ciphertext under Γ'_e and W', where $S \models \Gamma_e$ and $\Gamma_e \odot \Gamma'_e = \phi$. Note that W' may not equal to W. If $W' \neq W$, it means that the keyword set W in original ciphertext is updated to W' in re-encryption phase.

- $\mathsf{Re\text{-}Encrypt}(PP, csk, CT, rk) \rightarrow CT$: Taking PP, csk, an original ciphertext CT and a re-encryption key rk as input, PCS produces a re-encrypted ciphertext CT.

- $\mathsf{TokenGen}(PP, \Gamma_t, tsk, sk_S) \rightarrow (tok, sk_{tr})$: Taking PP, tsk and a Boolean formula Γ_t over $\mathcal{U}_t$ as input, the TGC generates and sends a trapdoor to a DU. Next, taking PP, the trapdoor obtained from TGC and the user's decryption key sk_S as input, the DU generates a token tok and transformed secret key sk_{tr}.

- $\mathsf{Search}(PP, CT, csk, tok) \rightarrow 1/0$: Taking PP, an original or re-encrypted ciphertext CT, a token tok and the cloud secret key csk as input, PCS returns 1 if $W \models \Gamma_t$; otherwise, returns 0. If PCS returns 1, it generates the associated matching ciphertext CT_m.

- $\mathsf{Transform}(PP, CT_m, tok) \rightarrow CT_{tr} \ / \ \perp$: Given PP, a matching original or re-encrypted ciphertext CT_m and a token tok, it generates the corresponding transformed original (resp. re-encrypted) ciphertext CT_{tr} if $S \models \Gamma_e$ (resp. $S \models \Gamma'_e$) ; otherwise, outputs $\perp$.

- $\mathsf{Verify\text{-}and\text{-}Decrypt}(PP, CT_{tr}, sk_{tr}) \rightarrow m/\perp$: Taking PP, a transformed original or re-encrypted ciphertext CT_{tr} and a transformed secret key sk_{tr} as input, it outputs the message m if the search results returned by PCS are correct; otherwise, outputs $\perp$ indicating that search results are incorrect.

Consistency. Our scheme is *correct* if

$$\Pr\left[\mathsf{Verify\text{-}and\text{-}Decrypt}(PP, CT_{tr}, sk_{tr}) \rightarrow m \;\middle|\; \begin{array}{l} \mathsf{PKG.Setup}(1^\kappa, U) \rightarrow (mpk, msk) \\ \mathsf{TGC.Setup}(mpk) \rightarrow (tpk, tsk) \\ \mathsf{PCS.Setup}(mpk) \rightarrow (cpk, csk) \\ \mathsf{KeyGen}(PP, msk, S) \rightarrow sk_S \\ \mathsf{Encrypt}(PP, m, \Gamma_e, W) \rightarrow CT \\ \mathsf{TokenGen}(PP, \Gamma_t, tsk, sk_S) \rightarrow (tok, sk_{tr}) \\ \mathsf{Search}(PP, CT, csk, tok) \rightarrow 1 \\ \mathsf{Transform}(PP, CT_m, tok) \rightarrow CT_{tr} \end{array}\right] = 1$$

and

$$\Pr\left[\mathsf{Verify\text{-}and\text{-}Decrypt}(PP, CT_{tr}, sk'_{tr}) \rightarrow m \;\middle|\; \begin{array}{l} \mathsf{PKG.Setup}(1^\kappa, U) \rightarrow (mpk, msk) \\ \mathsf{TGC.Setup}(mpk) \rightarrow (tpk, tsk) \\ \mathsf{PCS.Setup}(mpk) \rightarrow (cpk, csk) \\ \mathsf{KeyGen}(PP, msk, S) \rightarrow sk_S \\ \mathsf{Re\text{-}KeyGen}(PP, sk_S, \Gamma'_e, W') \rightarrow rk \\ \mathsf{Re\text{-}Encrypt}(PP, csk, \mathsf{Encrypt}(PP, m, \Gamma_e, W), rk) \rightarrow CT \\ \mathsf{TokenGen}(PP, \Gamma_t, tsk, sk_S) \rightarrow (tok', sk'_{tr}) \\ \mathsf{Search}(PP, CT, csk, tok') \rightarrow 1 \\ \mathsf{Transform}(PP, CT_m, tok') \rightarrow CT_{tr} \end{array}\right] = 1$$

Definition 13. (IND-CCA2-Or-Type-1). *Our scheme is indistinguishable adaptive chosen ciphertext attack secure at original ciphertext against a Type-1 adversary*

$\mathcal{A}$ if there is no PPT $\mathcal{A}$ that can win the subsequent game with a non-negligible advantage against a challenger $\mathcal{B}$.

Init. A challenge attribute $y^\star \in \mathcal{U}$ is output by $\mathcal{A}$.
Setup. $\mathcal{B}$ executes all the Setup algorithms and sends $PP = (mpk, tpk, cpk)$ and csk to $\mathcal{A}$.
Phase I. The oracles listed below are accessible to $\mathcal{A}$.

- $\mathcal{O}_{sk}(S)$: Taking an attribute set S with $y^\star \notin S$, it creates and sends a decryption key sk_S to $\mathcal{A}$.
- $\mathcal{O}_{rk}(S, W', \Gamma'_e)$: On input an attribute set S, a keyword set W' and an encryption policy Γ'_e such that $S \not\models \Gamma'_e$, it returns a re-key rk. Note that the input (S, Γ'_e) should not satisfy the condition $y^\star \in S$ and $\mathcal{O}_{sk}(S')$ for any $S' \models \Gamma'_e$.
- $\mathcal{O}_{re}(CT, S, W', \Gamma'_e)$: On input an original ciphertext CT under Γ_e and W, an attribute set S such that $S \models \Gamma_e$, a keyword set W' and an encryption policy Γ'_e, it returns a re-encrypted ciphertext CT or $\perp$ to $\mathcal{A}$.
- $\mathcal{O}_{token}(\Gamma_t, S)$: It takes input a Boolean keyword formula Γ_t and an attribute set S. If $y^\star \in S$, it returns a token tok to $\mathcal{A}$. And, if $y^\star \notin S$, it returns both token tok and transformed secret key sk_{tr} to $\mathcal{A}$.
- $\mathcal{O}_{search}(\Gamma_t, S, CT)$: Taking input CT, Γ_t, and S, it returns either 0 or 1 to $\mathcal{A}$.
- $\mathcal{O}_{decrypt}(\Gamma_t, S, CT)$: On input CT, Γ_t and S, it outputs a message m or $\perp$.

Challenge. After Phase I is over, $\mathcal{A}$ selects two equal length messages $m_0^\star$ and $m_1^\star$, a challenge keyword set $W^\star$ and a challenge encryption policy $\Gamma_e^\star$, and forwards them to $\mathcal{B}$. Now, $\mathcal{B}$ selects a bit $i \xleftarrow{u} \{0, 1\}$ and sends to $\mathcal{A}$ a challenge ciphertext $CT^\star$ of $m_i^\star$ under the encryption policy $\Gamma_e^\star \wedge y^\star$ and the keyword set $W^\star$.
Phase II. After challenge phase is over, $\mathcal{A}$ queries as in Phase I except the following:

- $\mathcal{O}_{re}(CT^\star, S, W', \Gamma'_e)$. and $\mathcal{O}_{sk}(S')$, if $S \models \Gamma_e^\star \wedge y^\star$ and $S' \models \Gamma'_e$,
- $\mathcal{O}_{decrypt}(\Gamma_t, S, CT)$ if CT is a derivative [1] of $CT^\star$, $S \models \Gamma_e^\star \wedge y^\star$ and $W^\star \models \Gamma_t$.

Guess. $\mathcal{A}$ outputs a bit i' and wins if $i' = i$.
The advantage of $\mathcal{A}$ is defined as $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{IND\text{-}CCA2\text{-}Or\text{-}Type\text{-}1}}(1^\kappa) = \left|\Pr[i' = i] - 1/2\right|$.

[1] The definition is available in [47]

Definition 14. (IND-CCA2-Or-Type-2). *Our scheme is indistinguishable adaptive chosen ciphertext attack secure at original ciphertext against a Type-2 adversary $\mathcal{A}$ if there is no PPT $\mathcal{A}$ that can win the following game with a non-negligible advantage against a challenger $\mathcal{B}$.*

Setup. $\mathcal{B}$ executes all the Setup algorithms to generate $(mpk, msk), (tpk, tsk)$ and (cpk, csk), and sends $PP = (mpk, tpk, cpk)$ to $\mathcal{A}$.
Phase I. $\mathcal{A}$ queries for the oracles $\mathcal{O}_{sk}, \mathcal{O}_{rk}, \mathcal{O}_{re}, \mathcal{O}'_{token}, \mathcal{O}'_{search}$ and $\mathcal{O}_{decrypt}$. The oracles $\mathcal{O}_{sk}, \mathcal{O}_{rk}, \mathcal{O}_{re}$ and $\mathcal{O}_{decrypt}$ are similar to IND-CCA2-Or-Type-1 game and the oracle $\mathcal{O}'_{token}$ and $\mathcal{O}'_{search}$ are described as follows.

- $\mathcal{O}'_{token}(\Gamma_t, S)$: On input a Boolean keyword formula Γ_t and an attribute set S, $\mathcal{B}$ returns a token tok and transformed secret key sk_{tr} to $\mathcal{A}$.
- $\mathcal{O}'_{search}(\Gamma_t, S, CT)$: Taking input CT, Γ_t, and S, it returns either 0 or 1 to $\mathcal{A}$.

Challenge. $\mathcal{A}$ selects two messages $m_0^\star$ and $m_1^\star$ of equal length, a challenge keyword set $W^\star$ and a challenge encryption policy $\Gamma_e^\star$, and forwards them to $\mathcal{B}$. Now, $\mathcal{B}$ selects a bit $i \xleftarrow{u} \{0, 1\}$ and returns a challenge ciphertext
$CT^\star \leftarrow \mathsf{Encrypt}(PP, m_i^\star, \Gamma_e^\star, W^\star)$ to $\mathcal{A}$.
Phase II. $\mathcal{A}$ queries as in Phase I except the following:

- $\mathcal{O}_{re}(CT^\star, S, W', \Gamma'_e)$ and $\mathcal{O}_{sk}(S')$, if $S \models \Gamma_e^\star$ and $S' \models \Gamma'_e$,
- $\mathcal{O}_{decrypt}(\Gamma_t, S, CT)$ if CT is a derivative of $CT^\star$, $S \models \Gamma_e^\star$, and $W^\star \models \Gamma_t$.

Guess. $\mathcal{A}$ produces its guess i' and wins if $i' = i$.
The advantage of $\mathcal{A}$ is $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{IND\text{-}CCA2\text{-}Or\text{-}Type\text{-}2}}(1^\kappa) = \left|\Pr[i' = i] - 1/2\right|$.

Definition 15. (IND-CCA2-Re-Type-1). *Our scheme is indistinguishable adaptive chosen ciphertext attack secure at re-encrypted ciphertext against a Type-1 adversary $\mathcal{A}$ if there is no PPT $\mathcal{A}$ that can win the game described below with a non-negligible advantage against a challenger $\mathcal{B}$.*

Init. A challenge attribute $y^\star \in \mathcal{U}$ is output by $\mathcal{A}$.
Setup. $\mathcal{B}$ performs all the Setup algorithms to generate $(mpk, msk), (tpk, tsk)$ and (cpk, csk) and sends $PP = (mpk, tpk, cpk)$ and csk to $\mathcal{A}$.
Phase I. $\mathcal{A}$ queries for the set $\{\mathcal{O}_{sk}, \mathcal{O}'_{rk}, \mathcal{O}_{token}, \mathcal{O}_{search}, \mathcal{O}_{decrypt}\}$ of oracles, where

- $\mathcal{O}'_{rk}(S, W', \Gamma'_e)$: On input a keyword set W', an attribute set S and an encryption policy Γ'_e, where $S \not\models \Gamma'_e$, it returns a re-key rk to $\mathcal{A}$.

The other oracles are similar to IND-CCA2-Or-Type-1 game.

Challenge. $\mathcal{A}$ selects two equal length messages $m_0^\star$ and $m_1^\star$, a challenge keyword set $W^\star$ and a challenge encryption policy $\Gamma_e^\star$, and forwards them to $\mathcal{B}$. Now, $\mathcal{B}$ selects $i \xleftarrow{u} \{0,1\}$, generates $CT^\star \leftarrow \mathsf{Re\text{-}Encrypt}\big(PP, csk, \mathsf{Encrypt}(PP, m_i^\star, \Gamma_e, W), rk^\star\big)$, where $rk^\star \leftarrow \mathsf{Re\text{-}KeyGen}(PP, sk_S, \Gamma_e^\star \wedge y^\star, W^\star), S \models \Gamma_e$, and forwards $CT^\star$ to $\mathcal{A}$.

Phase II. Similar to Phase I, $\mathcal{A}$ queries with the exception of the following:

- $\mathcal{O}_{decrypt}(\Gamma_t, S, CT^\star)$ if $S \models \Gamma_e^\star \wedge y^\star$ and $W^\star \models \Gamma_t$.

Guess. $\mathcal{A}$ announces its guess i' and if $i' = i$, $\mathcal{A}$ wins.

The advantage of $\mathcal{A}$ is $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{IND\text{-}CCA2\text{-}Re\text{-}Type\text{-}1}}(1^\kappa) = \big|\Pr[i' = i] - 1/2\big|$.

Definition 16. (IND-CCA2-Re-Type-2). *Our scheme is indistinguishable adaptive chosen ciphertext attack secure at re-encrypted ciphertext against a Type-2 adversary $\mathcal{A}$ if there is no PPT $\mathcal{A}$ that can win the game described below with a non-negligible advantage against a challenger $\mathcal{B}$.*

Setup. $\mathcal{B}$ runs all Setup algorithms and generate (mpk, msk), (tpk, tsk), and (cpk, csk). It sends $PP = (mpk, tpk, cpk)$ to $\mathcal{A}$.

Phase I. $\mathcal{A}$ can query the oracles $\mathcal{O}_{sk}, \mathcal{O}'_{rk}, \mathcal{O}'_{token}, \mathcal{O}'_{search}$ and $\mathcal{O}_{decrypt}$. The oracles $\mathcal{O}_{sk}, \mathcal{O}'_{rk}$ and $\mathcal{O}_{decrypt}$ are same as described in the game IND-CCA2-Re-Type-1, and $\mathcal{O}'_{token}$ and $\mathcal{O}'_{search}$ are similar to IND-CCA2-Or-Type-2 game.

Challenge. $\mathcal{A}$ chooses two messages $m_0^\star$ and $m_1^\star$ of equal length, a challenge encryption policy $\Gamma_e^\star$ and a challenge keyword set $W^\star$ and sends them to $\mathcal{B}$. Now, $\mathcal{B}$ selects a bit $i \xleftarrow{u} \{0,1\}$, outputs $CT^\star \leftarrow \mathsf{Re\text{-}Encrypt}\big(PP, csk, \mathsf{Encrypt}(PP, m_i^\star, \Gamma_e, W), rk^\star\big)$, where $rk^\star \leftarrow \mathsf{Re\text{-}KeyGen}(PP, sk_S, \Gamma_e^\star, W^\star), S \models \Gamma_e$, and forwards $CT^\star$ to $\mathcal{A}$.

Phase II. Similar to Phase I, $\mathcal{A}$ queries with the exception of the following:

- $\mathcal{O}_{decrypt}(\Gamma_t, S, CT^\star)$ if $S \models \Gamma_e^\star$ and $W^\star \models \Gamma_t$.

Guess. $\mathcal{A}$ announces its guess i' and if $i' = i$, $\mathcal{A}$ wins.

The advantage $\mathcal{A}$ is $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{IND\text{-}CCA2\text{-}Re\text{-}Type\text{-}2}}(1^\kappa) = \big|\Pr[i' = i] - 1/2\big|$.

Definition 17. ($\mathsf{IND\text{-}CKA_{ct}}$). *Our scheme is indistinguishable chosen keyword attack secure on ciphertext if there is no PPT adversary $\mathcal{A}$ without having the cloud secret key that can win the subsequent game with a non-negligible advantage against a challenger $\mathcal{B}$.*

Setup. $\mathcal{B}$ executes all the Setup algorithms and generates (mpk, msk), (tpk, tsk) and (cpk, csk). Next, it sends $PP = (mpk, tpk, cpk)$ to $\mathcal{A}$.

Phase I. The oracles listed below are accessible to $\mathcal{A}$.

- $\mathcal{O}'_{sk}(S)$: Taking an attribute set S, it produces sk_S.
- $\mathcal{O}'_{rk}(S, W', \Gamma'_e)$: Taking an encryption policy Γ'_e, an attribute set S, and a keyword set W', where $S \not\models \Gamma'_e$, it returns a re-key to $\mathcal{A}$.
- $\mathcal{O}''_{token}(\Gamma_t, S)$: Taking a boolean keyword formula Γ_t and an attribute set S, $\mathcal{B}$ sends a token tok and transformed secret key sk_{tr} to $\mathcal{A}$.
- $\mathcal{O}'_{search}(\Gamma_t, S, CT)$: On input CT, Γ_t, and S, it outputs the search result 0 or 1 to $\mathcal{A}$.
- $\mathcal{O}_{decrypt}(\Gamma_t, S, CT)$: On input CT, Γ_t and S, it outputs a message m or $\perp$.

Challenge. $\mathcal{A}$ sends to $\mathcal{B}$ two equal size keyword sets $W_0^\star$ and $W_1^\star$ $\big($where either $(W_0^\star \models \Gamma_t \wedge W_1^\star \models \Gamma_t)$ or $(W_0^\star \not\models \Gamma_t \wedge W_1^\star \not\models \Gamma_t)$ for all Γ_t submitted to $\mathcal{O}''_{token}, \mathcal{O}'_{search}, \mathcal{O}_{decrypt}$ in Phase I$\big)$, a challenge message $m^\star$ and a challenge encryption policy $\Gamma_e^\star$. Next, $\mathcal{B}$ selects $i \xleftarrow{\text{u}} \{0,1\}$, sends to $\mathcal{A}$ either the challenge ciphertext or the re-encrypted ciphertext $CT^\star$.
Phase II. Queries made by $\mathcal{A}$ are similar to those in Phase I along with the following conditions:

- $\mathcal{O}''_{token}(\Gamma_t, S)$ with either $(W_0^\star \models \Gamma_t \wedge W_1^\star \models \Gamma_t)$ or $(W_0^\star \not\models \Gamma_t \wedge W_1^\star \not\models \Gamma_t)$,
- $\mathcal{O}'_{search}(\Gamma_t, S, CT^\star)$ with either $(W_0^\star \models \Gamma_t \wedge W_1^\star \models \Gamma_t)$ or $(W_0^\star \not\models \Gamma_t \wedge W_1^\star \not\models \Gamma_t)$,
- $\mathcal{O}_{decrypt}(\Gamma_t, S, CT^\star)$ with either $(W_0^\star \models \Gamma_t \wedge W_1^\star \models \Gamma_t)$ or $(W_0^\star \not\models \Gamma_t \wedge W_1^\star \not\models \Gamma_t)$.

Guess. $\mathcal{A}$ announces its guess i' and if $i' = i$, $\mathcal{A}$ wins.
The advantage of $\mathcal{A}$ is $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{IND\text{-}CKA_{ct}}}(1^\kappa) = \big|\Pr[i' = i] - 1/2\big|$.

Definition 18. ($\mathsf{IND\text{-}CKA_{tok}}$). *Our scheme is indistinguishable chosen keyword attack secure on token if there is no PPT adversary $\mathcal{A}$ without having the cloud secret key that can win the subsequent game with a non-negligible advantage against a challenger $\mathcal{B}$.*

Setup. $\mathcal{B}$ executes all the Setup algorithms and generate $(mpk, msk), (tpk, tsk)$, and (cpk, csk). Then, it sends $PP = (mpk, tpk, cpk)$ to $\mathcal{A}$.
Phase I. $\mathcal{A}$ has access to the oracles $\mathcal{O}'_{sk}, \mathcal{O}'_{rk}, \mathcal{O}''_{token}, \mathcal{O}'_{search}$, and $\mathcal{O}_{decrypt}$, which are identical to those in the $\mathsf{IND\text{-}CKA_{ct}}$ game.

Challenge. $\mathcal{A}$ forwards to $\mathcal{B}$ two equal size Boolean keyword formulas $\Gamma^{\star}_{t(0)}$ and $\Gamma^{\star}_{t(1)}$ (where either $W \models \Gamma^{\star}_{t(0)} \wedge W \models \Gamma^{\star}_{t(1)}$ or $W \not\models \Gamma^{\star}_{t(0)} \wedge W \not\models \Gamma^{\star}_{t(1)}$ for all W submitted to $\mathcal{O}'_{rk}$, and for all W attached to CT which is input to $\mathcal{O}'_{search}$ and $\mathcal{O}_{decrypt}$ in Phase I), a challenge attribute set $S^{\star}$. Now, $\mathcal{B}$ selects a bit $i \xleftarrow{u} \{0,1\}$, computes $(tok^{\star}, sk^{\star}_{tr}) \leftarrow \mathsf{TokenGen}(PP, \Gamma^{\star}_{t(i)}, tsk, sk_{S^{\star}})$, and sends $(tok^{\star}, sk^{\star}_{tr})$ to $\mathcal{A}$.
Phase II. $\mathcal{A}$ queries as in Phase I along with the following conditions:

- $\mathcal{O}''_{token}(\Gamma_t, S)$ with $\Gamma^{\circ}_t \notin \{\Gamma^{\star\circ}_{t(0)}, \Gamma^{\star\circ}_{t(1)}\}$,
- $\mathcal{O}'_{search}(\Gamma_t, S^{\star}, CT)$ with either $(W \models \Gamma^{\star}_{t(0)} \wedge W \models \Gamma^{\star}_{t(1)})$ or $(W \not\models \Gamma^{\star}_{t(0)} \wedge W \not\models \Gamma^{\star}_{t(1)})$, where W is involved in CT.
- $\mathcal{O}_{decrypt}(\Gamma_t, S^{\star}, CT)$ with either $(W \models \Gamma^{\star}_{t(0)} \wedge W \models \Gamma^{\star}_{t(1)})$ or $(W \not\models \Gamma^{\star}_{t(0)} \wedge W \not\models \Gamma^{\star}_{t(1)})$, where W is from CT.

Guess. $\mathcal{A}$ makes a guess i' and if $i' = i$, $\mathcal{A}$ wins.
The advantage of $\mathcal{A}$ is $\mathsf{Adv}^{\mathsf{IND\text{-}CKA}_{\mathsf{tok}}}_{\mathcal{A}}(1^{\kappa}) = \left|\Pr[i' = i] - 1/2\right|$.

Definition 19. (Verifiability). *Our scheme is verifiable if there is no PPT adversary $\mathcal{A}$ having the cloud secret key that can win the subsequent game with a non-negligible advantage against a challenger $\mathcal{B}$.*

Setup. $\mathcal{B}$ runs all the Setup algorithms and generates $(mpk, msk), (tpk, tsk)$, and (cpk, csk). It sends $PP = (mpk, tpk, cpk)$ and csk to $\mathcal{A}$.
Phase I. The oracles listed below are accessible to $\mathcal{A}$.

- $\mathcal{O}'_{sk}(S)$: Taking an attribute set S, it creates and sends the user's decryption key sk_S to $\mathcal{A}$.
- $\mathcal{O}''_{rk}(S, W, \Gamma'_e)$: Given an attribute set S, a keyword set W and an encryption policy Γ'_e, where $S \not\models \Gamma'_e$, it returns a re-key to $\mathcal{A}$.
- $\mathcal{O}'''_{token}(\Gamma_t, S)$: On input a Boolean keyword formula Γ_t and an attribute set S, $\mathcal{B}$ sends a token tok to $\mathcal{A}$.
- $\mathcal{O}''_{search}(CT, \Gamma_t, S)$: Taking an attribute set S, a Boolean keyword formula Γ_t, and a ciphertext CT, it sends the search result either 0 or 1 to $\mathcal{A}$.
- $\mathcal{O}_{transform}(\Gamma_t, S, CT)$: Given a ciphertext CT, Γ_t and S, $\mathcal{B}$ returns a transformed ciphertext CT_{tr} or $\perp$.
- $\mathcal{O}_{decrypt}(\Gamma_t, S, CT)$: On input CT, Γ_t and S, it outputs a message m or $\perp$.

Challenge. $\mathcal{A}$ sends to $\mathcal{B}$ a message $m^\star$, a keyword set $W^\star$ and an encryption policy $\Gamma_e^\star$. Now, $\mathcal{B}$ outputs $CT^\star \leftarrow \mathsf{Encrypt}(PP, m^\star, \Gamma_e^\star, W^\star)$ or
$CT^\star \leftarrow \mathsf{Re\text{-}Encrypt}\big(PP, csk, \mathsf{Encrypt}(PP, m^\star, \Gamma_e, W), rk^\star\big)$, where
$rk^\star \leftarrow \mathsf{Re\text{-}KeyGen}(PP, sk_S, \Gamma_e^\star, W^\star), S \models \Gamma_e$, and sends $CT^\star$ to $\mathcal{A}$.
Phase II. $\mathcal{A}$ queries as in Phase I except the following:

- $\mathcal{O}_{transform}(CT^\star, \Gamma_t, S)$ if $S \models \Gamma_e^\star$ and $W^\star \models \Gamma_t$.

Output. $\mathcal{A}$ outputs $CT_{tr}^\star, \Gamma_t^\star$ and $S^\star$.
If $\mathsf{Verify\text{-}and\text{-}Decrypt}(PP, CT_{tr}^\star, sk_{tr}^\star) \notin \{m^\star, \perp\}$, $\mathcal{A}$ wins. Here, $sk_{tr}^\star$ can be produced through a query on $\mathcal{O}'''_{token}(\Gamma_t^\star, S^\star)$.
The advantage of $\mathcal{A}$ is defined as $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{verifiability}}(1^\kappa) = \Pr[\mathcal{A} \text{ wins}]$.

5.3 ABPRE-BKS Construction

We define a Boolean keyword policy as $\Gamma_t = (\mathcal{K}_t, \psi_t^\circ, \{w_{\psi_t^\circ(i)}\}_{i\in[\ell_t]})$. Here $\mathcal{K}_t$ represents a matrix with dimension $\ell_t \times n_t$. The rows of $\mathcal{K}_t$ are mapped to generic keyword names $\mathcal{W}$ via the function ψ_t° and each row in $\mathcal{K}_t$ is associated witha keyword value, denoted by $w_{\psi_t^\circ(i)}$. Let $W = \{[\mathcal{W} : w]\}$ is a keyword set, where $W^\circ = \{\mathcal{W}\}$. If $W \models \Gamma_t$ then there exists a vector $\vec{a} = (a_1, a_2, \ldots, a_{\ell_t})$ which satisfies the following

(i) $\vec{a} \cdot \mathcal{K}_t = \vec{1}_{n_t}$

(ii) $a_i = 0$, for all $i \in [\ell_t]$ such that $\psi_t^\circ(i) \notin W^\circ$
along with $w_{\psi_t^\circ(i)} = w$, for all $i \in [\ell_t]$ satisfying $\psi_t^\circ(i) = \mathcal{W} \in W^\circ$

$\mathsf{PKG.Setup}(1^\kappa, U)$**.** Given the security parameter 1^κ and an attribute universe U, it executes the following steps.

- Choose a pairing tuple $\Delta = (p, g, \mathbb{G}, \mathbb{G}_T, e)$.
- Pick $\alpha, \beta \xleftarrow{u} \mathbb{Z}_p^*$ and set $X = g^\beta, Y = e(g, g)^\alpha$.
- Choose $g_1, g_2, g_3, g_4, g_5, g_6, g_7, g_8, g_9 \xleftarrow{u} \mathbb{G}$.
- Set $M = \{0,1\}^\lambda$ as the space of all the messages.
- Choose ten collision-resistant hash functions: $H_1 : \{0,1\}^{2\lambda} \to \mathbb{Z}_p^*$,
  $H_2 : \mathbb{G}_T \to \{0,1\}^{2\lambda}$, $H_3 : \{0,1\}^* \to \mathbb{G}$, $H_4 : \mathbb{G}_T \to \mathbb{Z}_p^*$,
  $H_5 : \{0,1\}^* \to \{0,1\}^{\ell_{H_5}}$, $H_6 : \mathbb{G}_T \to \mathbb{Z}_p^*$, $H_7 : \{0,1\}^* \to \mathbb{Z}_p^*$,
  $H_8 : \{0,1\}^* \to \mathbb{G}$, $H_9 : \{0,1\}^\lambda \to \mathbb{Z}_p^*$, $H_{10} : \mathbb{G}_T \to \mathbb{Z}_p^*$. Here, H_3 and H_8 are independent hash functions, and H_4,H_6, and H_{10} are independent hash functions.

- Finally, output the master public key $mpk = (\Delta, X, Y, \{g_i\}_{i=1}^{9}, M, \{H_i\}_{i=1}^{10})$ and the master secret key $msk = g^{\alpha}$.

TGC.Setup(mpk). Given input mpk, it

- chooses $\gamma', b_1, b_2, b_3, b_4 \xleftarrow{u} \mathbb{Z}_p^*$, and computes $h_1 = g^{b_1}, h_2 = g^{b_2}, h_3 = g^{b_3}, h_4 = g^{b_4}$ and $h_T = e(g, g_6)^{\gamma'}$,
- outputs TGC public key $tpk = (h_T, h_1, h_2, h_3, h_4)$ and TGC secret key $tsk = (\gamma', b_1, b_2, b_3, b_4)$.

PCS.Setup(mpk). Taking mpk as input, it

- picks $\beta' \xleftarrow{u} \mathbb{Z}_p^*$ and sets $\hat{X} = X^{\beta'}$,
- outputs PCS public key $cpk = \hat{X}$ and PCS secret key $csk = \beta'$.

Set $PP = (mpk, tpk, cpk)$.

KeyGen(PP, msk, S). Given input PP, msk and an attribute set S,

- select $r \xleftarrow{u} \mathbb{Z}_p^*$ and set $K_1 = g^{\alpha+\beta r}, K_2 = g^r, K_x = H_3(x)^r, \forall x \in S$.
- Output decryption key $sk_S = (S, K_1, K_2, \{K_x\}_{x\in S})$.

Encrypt(PP, m, Γ_e, W). Given input PP, a message $m \in \{0,1\}^{\lambda}$, a DNF encryption policy $\Gamma_e = B_1 \vee B_2 \vee \cdots \vee B_n$, and a set $W = \{[\mathcal{W} : w]\}$ of keywords (where $W^{\circ} = \{\mathcal{W}\}$ represents the set of generic keyword names and the set $\{w\}$ the corresponding keyword values), it generates an original ciphertext CT according to the following computations.

- Select $\delta', \delta'' \xleftarrow{u} \mathbb{Z}_p^*, \gamma \xleftarrow{u} \{0,1\}^{\lambda}$, and set

$$\sigma' = g^{\delta'}, \sigma'' = X^{\delta''}, s = H_1(m||\gamma).$$

- Choose $s_i \xleftarrow{u} \mathbb{Z}_p^*, \ \forall i \in [n]$ and compute

$$C_0 = (m||\gamma) \oplus H_2(Y^s), C_1 = g^{sH_6(e(\sigma', \hat{X})^{\delta''})}, C_2 = g_1^s,$$

$$C_{1,i} = X^s\big(\prod_{y\in B_i} H_3(y)\big)^{s_i}, C_{2,i} = g^{s_i}, \ \forall i \in [n]$$

$$\delta = H_4(Y^s), \mathsf{tag} = H_5(\delta||C_0)$$

Set $ct_e = (\Gamma_e, \sigma', \sigma'', C_0, C_1, C_2, \{C_{1,i}, C_{2,i}\}_{i\in[n]})$.

- Choose $\mu_{\mathcal{W}}, \tau_{\mathcal{W}1}, \tau_{\mathcal{W}2} \xleftarrow{u} \mathbb{Z}_p^*$, $\forall \mathcal{W} \in W^\circ$ and compute

$$I_{\mathcal{W}1} = h_1^{\mu_{\mathcal{W}} - \tau_{\mathcal{W}1}}, I_{\mathcal{W}2} = h_2^{\tau_{\mathcal{W}1}}, I_{\mathcal{W}3} = h_3^{\mu_{\mathcal{W}} - \tau_{\mathcal{W}2}}, I_{\mathcal{W}4} = h_4^{\tau_{\mathcal{W}2}},$$

$$I_{\mathcal{W}5} = (g_2^w g_3)^{\mu_{\mathcal{W}}} g_4^{-s}, I_{\mathcal{W}6} = (g_2^w g_3)^{\mu_{\mathcal{W}}} g_5^{-s}, k_T = h_T^s,$$

  Set $ct_k = (W^\circ, k_T, \{I_{\mathcal{W}1}, I_{\mathcal{W}2}, I_{\mathcal{W}3}, I_{\mathcal{W}4}, I_{\mathcal{W}5}, I_{\mathcal{W}6}\}_{\mathcal{W} \in W^\circ})$.

- Choose $\varepsilon \xleftarrow{u} \mathbb{Z}_p^*$ and compute $\bar{C} = (g_7^{\varrho} g_8^{\varepsilon} g_9)^s$, where $\varrho = H_7(ct_e || ct_k || \mathsf{tag})$. Finally, output the original ciphertext $CT = (ct_e, ct_k, \mathsf{tag}, \bar{C}, \varepsilon)$.

Re-KeyGen$(PP, sk_S, \Gamma_e', W')$. Given input PP, a decryption key sk_S, a DNF encryption policy $\Gamma_e' = B_1' \vee B_2' \vee \cdots \vee B_t'$, and a set $W' = \{[\mathcal{W}' : w']\}$ of keywords (where the set $W'^\circ = \{\mathcal{W}'\}$ represents the set of generic keywords names and the set $\{w'\}$ is the corresponding keyword values), it generates a re-encryption key rk by performing the following calculations.

- Select $\delta_1, \delta_2 \xleftarrow{u} \mathbb{Z}_p^*, \phi, \gamma_1 \xleftarrow{u} \{0,1\}^\lambda$, and set

$$\sigma_1 = g^{\delta_1}, \sigma_2 = X^{\delta_2}, s' = H_1(\phi || \gamma_1)$$

- Choose $s_i' \xleftarrow{u} \mathbb{Z}_p^*$, $\forall i \in [t]$, and compute

$$C_0' = (\phi || \gamma_1) \oplus H_2(Y^{s'}), C_1' = g^{s' H_6(e(\sigma_1, \hat{X})^{\delta_2})}, C_{1,i}' = X^{s'} \Big(\prod_{y \in B_i'} H_3(y) \Big)^{s_i'},$$

$$C_{2,i}' = g^{s_i'}, \delta' = H_4(Y^{s'}), \mathsf{tag1} = H_5(\delta' || C_0')$$

  Set $ct_e' = (\Gamma_e', \sigma_1, \sigma_2, C_0', C_1', \{C_{1,i}', C_{2,i}'\}_{i \in [t]})$.

- Choose $\mu_{\mathcal{W}'}', \pi_{\mathcal{W}'1}', \pi_{\mathcal{W}'2}' \xleftarrow{u} \mathbb{Z}_p^*$, $\forall \mathcal{W}' \in W'^\circ$ and calculate

$$I_{\mathcal{W}'1}' = h_1^{\mu_{\mathcal{W}'}' - \pi_{\mathcal{W}'1}'}, I_{\mathcal{W}'2}' = h_2^{\pi_{\mathcal{W}'1}'}, I_{\mathcal{W}'3}' = h_3^{\mu_{\mathcal{W}'}' - \pi_{\mathcal{W}'2}'}, I_{\mathcal{W}'4}' = h_4^{\pi_{\mathcal{W}'2}'},$$

$$I_{\mathcal{W}'5}' = (g_2^{w'} g_3)^{\mu_{\mathcal{W}'}'} g_4^{-s'}, I_{\mathcal{W}'6}' = (g_2^{w'} g_3)^{\mu_{\mathcal{W}'}'} g_5^{-s'}, k_T' = h_T^{s'}$$

  Set $ct_k' = (W'^\circ, k_T', \{I_{\mathcal{W}'1}', I_{\mathcal{W}'2}', I_{\mathcal{W}'3}', I_{\mathcal{W}'4}', I_{\mathcal{W}'5}', I_{\mathcal{W}'6}'\}_{\mathcal{W}' \in W'^\circ})$.

- Choose $\varepsilon' \xleftarrow{u} \mathbb{Z}_p^*$, compute $\varrho' = H_7(ct_e' || ct_k' || \mathsf{tag1} || S)$ and set $\bar{C}' = (g_7^{\varrho'} g_8^{\varepsilon'} g_9)^{s'}$.

- Set $ct_{\Gamma_e'} = (ct_e', ct_k', \mathsf{tag1}, \bar{C}', \varepsilon')$.

- Choose $\eta \xleftarrow{\text{u}} \mathbb{Z}_p^*$, and set

$$rk_0 = K_1^{H_9(\phi)} g_1^{\eta}, rk_1 = g^{\eta}, rk_2 = K_2^{H_9(\phi)},$$

$$rk_x = K_x^{H_9(\phi)}, \ \forall x \in S, rk_3 = ct_{\Gamma'_e}$$

- Finally, output the re-encryption key $rk = (S, rk_0, rk_1, rk_2, \{rk_x\}_{x \in S}, rk_3)$.

Re-Encrypt(PP, csk, CT, rk). Taking PP, csk, an original ciphertext CT and a re-encryption key rk as input, it performs the subsequent computations.

- Check whether the re-encryption key contains valid S, Γ'_e, W' or not by verifying the following equation

$$e\big(C'_1, g_7^{\varrho'} g_8^{\varepsilon'} g_9\big) \stackrel{?}{=} e(g^{H_6(e(\sigma_1,\sigma_2)^{csk})}, \bar{C}')$$

  where $\varrho' = H_7(ct'_e || ct'_k || \mathsf{tag1} || S)$.

- Next, compute $\varrho = H_7(ct_e || ct_k || \mathsf{tag})$ and check validity of the original ciphertext using the subsequent equations,

$$e(C_1, g_1) \stackrel{?}{=} e(g^{H_6(e(\sigma',\sigma'')^{csk})}, C_2) \tag{5.1}$$

$$e\big(C_2, g_7^{\varrho} g_8^{\varepsilon} g_9\big) \stackrel{?}{=} e(g_1, \bar{C}) \tag{5.2}$$

$$\forall i \in [n], \ e(C_{1,i}, g) \stackrel{?}{=} e(C_1^{1/H_6(e(\sigma',\sigma'')^{csk})}, X) \cdot e\big(C_{2,i}, \prod_{y \in B_i} H_3(y)\big) \tag{5.3}$$

  If the three equations (5.1), (5.2) and (5.3) do not hold simultaneously, output $\perp$. Otherwise, execute the following.

- If $S \not\models \Gamma_e$, output $\perp$. Otherwise, $S \supset B_i$ for some $i \in [n]$, then compute

$$T = \frac{e(C_1^{1/H_6(e(\sigma',\sigma'')^{csk})}, rk_0)}{e(C_2, rk_1) \cdot e(C_{1,i}, rk_2) \cdot e\big(C_{2,i}^{-1}, \prod_{x \in B_i} rk_x\big)}$$

- Finally, output the re-encrypted ciphertext $CT = (S, C_0, C_1, C_2, T, rk_3)$.

TokenGen$(PP, \Gamma_t, tsk, sk_S)$. It takes the input PP, a decryption key sk_S, the trapdoor secret key tsk and a Boolean keyword formula $\Gamma_t = (\mathcal{K}_t, \psi_t^{\circ}, \{w_{\psi_t^{\circ}(i)}\}_{i \in [\ell_t]})$, where $\mathcal{K}_t$ represents a matrix with dimension $\ell_t \times n_t$, the rows of $\mathcal{K}_t$ are mapped to generic keyword names $\mathcal{W}$ via the function ψ_t° and each row in $\mathcal{K}_t$ is associated witha keyword value, denoted by $w_{\psi_t^{\circ}(i)}$. Then, it outputs a token and

the corresponding transformed secret key pair (tok, sk_{tr}) in the following way. Note that $\Gamma_t^\circ = (\mathcal{K}_t, \psi_t^\circ)$, which is a keyword policy with only keyword generic names.

- TGC picks $\xi, \xi' \xleftarrow{u} \mathbb{Z}_p^*$ and $\breve{q}_i, \breve{q}_i' \xleftarrow{u} \mathbb{Z}_p^*$, for each $i \in [\ell_t]$ and sets $\Pi_1 = g^\xi$.
- It sets $\vec{\eta} = \big(\gamma' H_{10}(\mathsf{pp}), y_2, y_3, \ldots, y_{\ell_t}\big)$, where $\mathsf{pp} = e(\Pi_1, \hat{X})^{\xi'}$ and $y_2, y_3, \ldots, y_{\ell_t} \xleftarrow{u} \mathbb{Z}_p$. Next, it computes $\zeta_i = \mathcal{K}_t^{(i)} \cdot \vec{\eta}$, for all $i \in [\ell_t]$, where $\mathcal{K}_t^{(i)}$ is the i-th row of $\mathcal{K}_t$. Next, TGC computes and sends the trapdoor $Trp = (\Gamma_t, \Pi_1, \Pi_2, \{\Pi_{i1}, \Pi_{i2}', \Pi_{i3}, U_{i1}, U_{i2}, U_{i3}, U_{i4}\}_{i\in[\ell_t]})$ to the DU, where

$$\Pi_2 = X^{\xi'}, \Pi_{i1} = g_6^{\zeta_i} \cdot g_4^{b_1 b_2 \breve{q}_i + b_3 b_4 \breve{q}_i'},$$

$$\Pi_{i2}' = g_5^{b_1 b_2 \breve{q}_i + b_3 b_4 \breve{q}_i'}, \Pi_{i3} = H_8(\mathsf{pp}||\Gamma_t^\circ||\mathcal{K}_t^{(i)}) \cdot g^{b_1 b_2 \breve{q}_i + b_3 b_4 \breve{q}_i'},$$

$$U_{i1} = (g_2^{w_{\psi_t^\circ(i)}} g_3)^{-\breve{q}_i b_1}, U_{i2} = (g_2^{w_{\psi_t^\circ(i)}} g_3)^{-\breve{q}_i b_2},$$

$$U_{i3} = (g_2^{w_{\psi_t^\circ(i)}} g_3)^{-\breve{q}_i' b_3}, U_{i4} = (g_2^{w_{\psi_t^\circ(i)}} g_3)^{-\breve{q}_i' b_4}$$

- Then, the DU chooses $\phi_1, \phi_2, \phi_3 \xleftarrow{u} \mathbb{Z}_p^*$ and sets $\vec{\eta}' = \big(\phi_3/\phi_1, y_2', y_3', \ldots, y_{\ell_t}'\big)$, where $y_2', y_3', \ldots, y_{\ell_t}' \xleftarrow{u} \mathbb{Z}_p$. Next, it computes $\nu_i = \mathcal{K}_t^{(i)} \cdot \vec{\eta}'$, for all $i \in [\ell_t]$, sets $sk_{tr} = (\phi_1, \phi_2)$ and $tok = (\Gamma_t^\circ, \Pi_1, \Pi_2, \{\Pi_{i1}, \Pi_{i2}, \Pi_{i3}, U_{i1}, U_{i2}, U_{i3}, U_{i4}\}_{i\in[\ell_t]}, D_1, D_2, \{D_y\}_{y\in S})$, where

$$\Pi_{i2} = X^{\nu_i} \Pi_{i2}', D_1 = (K_1 \cdot X^{\phi_3})^{1/\phi_2}, D_2 = (K_2)^{1/\phi_2}, D_y = (K_y)^{1/\phi_2}, \forall y \in S.$$

Remark 10. *The distribution of the token* $tok = (\Gamma_t^\circ, \Pi_1, \Pi_2, \{\Pi_{i1}, \Pi_{i2}, \Pi_{i3}, U_{i1}, U_{i2}, U_{i3}, U_{i4}\}_{i\in[\ell_t]}, D_1, D_2, \{D_y\}_{y\in S})$ *of a keyword policy* $\Gamma_t = (\mathcal{K}_t, \psi_t^\circ, \{w_{\psi_t^\circ(i)}\}_{i\in[\ell_t]})$ *and an attribute set* S *is of the form*

$$\left\{ \begin{array}{l} \Pi_1 = g^\xi, \Pi_2 = X^{\xi'}, \\ \Pi_{i1} = g_6^{\zeta_i} \cdot g_4^{b_1 b_2 \breve{q}_i + b_3 b_4 \breve{q}_i'}, \\ \Pi_{i2} = X^{\nu_i} g_5^{b_1 b_2 \breve{q}_i + b_3 b_4 \breve{q}_i'}, \\ \Pi_{i3} = H_8(\mathsf{pp}||\Gamma_t^\circ||\mathcal{K}_t^{(i)}) \cdot g^{b_1 b_2 \breve{q}_i + b_3 b_4 \breve{q}_i'}, \\ U_{i1} = (g_2^{w_{\psi_t^\circ(i)}} g_3)^{-\breve{q}_i b_1}, U_{i2} = (g_2^{w_{\psi_t^\circ(i)}} g_3)^{-\breve{q}_i b_2}, \\ U_{i3} = (g_2^{w_{\psi_t^\circ(i)}} g_3)^{-\breve{q}_i' b_3}, U_{i4} = (g_2^{w_{\psi_t^\circ(i)}} g_3)^{-\breve{q}_i' b_4} \\ D_1 = g^{\alpha/\phi_2} X^{\ddot{r}} X^{\phi_3/\phi_2}, D_2 = g^{\ddot{r}}, D_y = (H_3(y))^{\ddot{r}} \end{array} \right\} \quad (5.4)$$

where $\xi, \xi', \breve{q}_i, \breve{q}'_i, \ddot{r}, \phi_1, \phi_2, \phi_3$ *are random exponents (elements in* $\mathbb{Z}_p^*$*),* ζ_i *(resp.* ν_i*) is the* i*th share of* $\gamma' \cdot H_{10}(e(\Pi_1, \hat{X})^{\xi'})$ *(resp.* ϕ_3/ϕ_1*) with respect to the policy* $(\mathcal{K}_t, \psi_t^\circ)$*,* $(\gamma', b_1, b_2, b_3, b_4)$ *is trapdoor secret key,* g^α *is system master secret key.*

This distribution is used in our ABPRE-BKS security study.

$\mathsf{Search}(PP, CT, csk, tok)$. Given input PP, CT (an original or re-encrypted ciphertext), csk, and tok, PCS outputs 1 or 0 as follows.

Case 1. If $CT = (ct_e, ct_k, \mathsf{tag}, \bar{C}, \varepsilon)$ is an original ciphertext, then compute $\varrho = H_7(ct_e||ct_k||\mathsf{tag})$ and check validity of CT by the equations (5.1), (5.2) and (5.3). If any of the above equations is not true, return $\perp$.
Otherwise, set $\breve{B} = {C_1}^{1/H_6(e(\sigma', \sigma'')^{csk})}$ and obtain $\vec{a} = (a_1, a_2, \ldots, a_{\ell_t})$ satisfying $\sum_{i \in [\ell_t]} a_i \zeta_i = \gamma' H_{10}(e(\Pi_1, \Pi_2)^{csk})$, by performing the secret recovery phase of LSSS.
Let $\varepsilon_i = H_8(e(\Pi_1, \Pi_2)^{csk}||\Gamma_t^\circ||\mathcal{K}_t^{(i)})$ and calculate

$$J_1 = e\Big(\breve{B}, \prod_{i \in [\ell_t]} \Pi_{i1}^{a_i}\Big) \times \prod_{i \in [\ell_t]} e\Big(\Pi_{i3} \cdot {\varepsilon_i}^{-1}, I_{\psi_t^\circ(i)5}\Big)^{a_i}$$

$$J_2 = \prod_{i \in [\ell_t]} \Big\{ e\Big(I_{\psi_t^\circ(i)1}, U_{i2}\Big) e\Big(I_{\psi_t^\circ(i)2}, U_{i1}\Big) e\Big(I_{\psi_t^\circ(i)3}, U_{i4}\Big) e\Big(I_{\psi_t^\circ(i)4}, U_{i3}\Big) \Big\}^{a_i}$$

and check whether $J_1 J_2 \stackrel{?}{=} {k_T}^{H_{10}(e(\Pi_1, \Pi_2)^{csk})}$. If this is true, output 1 and set the corresponding matching ciphertext as $CT_m = (ct_e, ct_k, \mathsf{tag})$; otherwise, output 0.

Case 2. If $CT = (S, C_0, C_1, C_2, T, rk_3)$ is a re-encrypted ciphertext, then compute $\varrho' = H_7(ct'_e||ct'_k||\mathsf{tag1}||S)$ and check validity of CT by the following equations,

$$e\big(C'_1, g_7^{\varrho'} g_8^{\varepsilon'} g_9\big) \stackrel{?}{=} e(g^{H_6(e(\sigma_1, \sigma_2)^{csk})}, \bar{C}') \tag{5.5}$$

$$\forall i \in [t],\ e(C'_{1,i}, g) \stackrel{?}{=} e({C'_1}^{1/H_6(e(\sigma_1, \sigma_2)^{csk})}, X) \cdot e\big(C'_{2,i}, \prod_{y \in B'_i} H_3(y)\big) \tag{5.6}$$

If anyone of the equations (5.5) and (5.6) is not valid, return $\perp$.
Else, set $\breve{B} = {C_1}^{1/H_6(e(\sigma', \sigma'')^{csk})}$, $\breve{D} = {C'_1}^{1/H_6(e(\sigma_1, \sigma_2)^{csk})}$ and obtain $\vec{a} = (a_1, a_2, \ldots, a_{\ell_t})$ satisfying $\sum_{i \in [\ell_t]} a_i \zeta_i = \gamma' H_{10}(e(\Pi_1, \Pi_2)^{csk})$, by performing the secret recovery phase of LSSS.

Let $\varepsilon_i = H_8(e(\Pi_1, \Pi_2)^{csk} || \Gamma_t^{\circ} || \mathcal{K}_t^{(i)})$ and calculate

$$\begin{aligned} Q_1 &= e\Big(\breve{D}, \prod_{i\in[\ell_t]} \Pi_{i1}^{a_i}\Big) \times \prod_{i\in[\ell_t]} e\Big(\Pi_{i3} \cdot \varepsilon_i^{-1}, I'_{\psi_t^{\circ}(i)5}\Big)^{a_i} \\ Q_2 &= \prod_{i\in[\ell_t]} \Big\{ e\Big(I'_{\psi_t^{\circ}(i)1}, U_{i2}\Big) e\Big(I'_{\psi_t^{\circ}(i)2}, U_{i1}\Big) e\Big(I'_{\psi_t^{\circ}(i)3}, U_{i4}\Big) e\Big(I'_{\psi_t^{\circ}(i)4}, U_{i3}\Big) \Big\}^{a_i} \end{aligned}$$

Check whether $Q_1 Q_2 \stackrel{?}{=} {k'_T}^{H_{10}(e(\Pi_1,\Pi_2)^{csk})}$. If this is true, output 1 and set the corresponding matching ciphertext as $CT_m = (S, C_0, \breve{B}, C_2, T, rk_3)$; otherwise, output 0.

$\mathsf{Transform}(PP, CT_m, tok)$. Given input PP, CT_m and tok, PCS outputs CT_{tr} or $\perp$ as described below.

Case 1. Suppose $CT_m = (ct_e, ct_k, \mathsf{tag})$ is a matching original ciphertext. If $S \not\models \Gamma_e$, output $\perp$. Otherwise, proceed further. Compute

$$\begin{aligned} \breve{J}_1 &= e\Big(\breve{B}, \prod_{i\in[\ell_t]} \Pi_{i2}^{a_i}\Big) \prod_{i\in[\ell_t]} e\Big(\Pi_{i3} \cdot \varepsilon_i^{-1}, I_{\psi_t^{\circ}(i)6}\Big)^{a_i} \\ R_1 &= \breve{J}_1 J_2 \end{aligned}$$

Since $S \models \Gamma_e, B_j \subseteq S$ for some $j \in [n]$,

$$\begin{aligned} J_3 &= \frac{e(C_{1,j}, D_2)}{e(C_{2,j}, \prod_{y\in B_j} D_y)} \\ J_4 &= e(\breve{B}, D_1) \\ R_2 &= J_3^{-1} \cdot J_4 \end{aligned}$$

The transformed ciphertext $CT_{tr} = (R_1, R_2, C_0, C_2, \breve{B}, \mathsf{tag})$.

Case 2. Suppose $CT_m = (S, C_0, \breve{B}, C_2, T, rk_3)$ is a matching re-encrypted ciphertext. If $S' \not\models \Gamma'_e$, output $\perp$. Otherwise, proceed further. Compute

$$\begin{aligned} \breve{Q}_1 &= e\Big(\breve{D}, \prod_{i\in[\ell_t]} \Pi_{i2}^{a_i}\Big) \prod_{i\in[\ell_t]} e\Big(\Pi_{i3} \cdot \varepsilon_i^{-1}, I'_{\psi_t^{\circ}(i)6}\Big)^{a_i} \\ Y_1 &= \breve{Q}_1 Q_2 \end{aligned}$$

Since $S' \models \Gamma'_e, B'_j \subseteq S'$ for some $j \in [t]$,

$$\begin{aligned} Q_3 &= \frac{e(C'_{1,j}, D_2)}{e(C'_{2,j}, \prod_{y \in B'_j} D_y)} \\ Q_4 &= e(\breve{D}, D_1) \\ Y_2 &= Q_3^{-1} \cdot Q_4 \end{aligned}$$

The transformed ciphertext $CT_{tr} = (Y_1, Y_2, C_0, C'_0, \breve{B}, \breve{D}, C_2, T, \mathsf{tag1})$.

Verify-and-Decrypt(PP, CT_{tr}, sk_{tr}). Given PP, CT_{tr}, and sk_{tr}, it validates the accuracy of the search outcomes provided by PCS and then outputs the message m or an error symbol $\perp$.

Case 1. If CT_{tr} is a transformed original ciphertext, then compute $\Omega = R_1^{-\phi_1} \cdot R_2^{\phi_2}$ and $\delta = H_4(\Omega)$. Check whether

$$H_5(\delta || C_0) \stackrel{?}{=} \mathsf{tag}$$

If this condition is false, the error symbol $\perp$ is returned to confirm that the PCS deceptively returns a false search result. Else, the delegator computes

$$C_0 \oplus H_2(\Omega) = m || \gamma$$

and outputs m if $\breve{B} = g^{H_1(m||\gamma)}$ and $C_2 = g_1^{H_1(m||\gamma)}$. Otherwise, return $\perp$.

Case 2. If CT_{tr} is a transformed re-encrypted ciphertext, then compute $\Omega' = Y_1^{-\phi_1} \cdot Y_2^{\phi_2}$ and $\delta' = H_4(\Omega')$. Check whether

$$H_5(\delta' || C'_0) \stackrel{?}{=} \mathsf{tag1}$$

If this condition is false, the error symbol $\perp$ is returned to indicate that the PCS deceptively returns a false search result. Else, the delegatee computes

$$C'_0 \oplus H_2(\Omega') = \phi || \gamma_1$$

and returns ϕ if $\breve{D} = g^{H_1(\phi||\gamma_1)}$. Next, it computes

$$C_0 \oplus H_2(T^{1/H_9(\phi)}) = m || \gamma$$

and outputs m if $\breve{B} = g^{H_1(m||\gamma)}$ and $C_2 = g_1^{H_1(m||\gamma)}$.
Otherwise, it outputs $\perp$.

Correctness of ABPRE-BKS

Theorem 11. *If $S \models \Gamma_e$ and $W \models \Gamma_t$, then the delegator can verify the accuracy of search results and recover the original message m from the original ciphertext CT.*

Proof. Let $\varphi_i = b_1 b_2 \breve{q}_i + b_3 b_4 \breve{q}'_i$. Since $W \models \Gamma_t$, we have $\sum_{i\in[\ell_t]} a_i\zeta_i = \gamma' H_{10}(\mathsf{pp})$. Then

$$
\begin{aligned}
J_1 &= e\Big(\breve{B}, \prod_{i\in[\ell_t]} \Pi_{i1}^{a_i}\Big) \times \prod_{i\in[\ell_t]} e\Big(\Pi_{i3}\cdot {\varepsilon_i}^{-1}, I_{\psi_t^\circ(i)5}\Big)^{a_i} \\
&= e\Big(C_1^{1/H_6(e(\sigma',\sigma'')^{csk})}, \prod_{i\in[\ell_t]} g_6^{\zeta_i}\cdot g_4^{\varphi_i}\Big)^{a_i} \cdot \prod_{i\in[\ell_t]} e\Big(g^{\varphi_i}, \big(g_2^{w_{\psi_t^\circ(i)}} g_3\big)^{\mu_{\psi_t^\circ(i)}} g_4^{-s}\Big)^{a_i} \\
&= \prod_{i\in[\ell_t]} \Big\{ e\Big(g^{\frac{sH_6(e(\sigma',\hat{X})^{\delta''})}{H_6(e(\sigma',\sigma'')^{csk})}}, g_6^{\zeta_i} g_4^{\varphi_i}\Big)^{a_i} \cdot e\big(g, g_4\big)^{-s\varphi_i a_i} \cdot e\big(g, g_2^{w_{\psi_t^\circ(i)}} g_3\big)^{\mu_{\psi_t^\circ(i)}\varphi_i a_i} \Big\} \\
&= \prod_{i\in[\ell_t]} \Big\{ e\big(g, g_4\big)^{s\varphi_i a_i} \cdot e\big(g, g_6\big)^{a_i\zeta_i s} \cdot e\big(g, g_4\big)^{-s\varphi_i a_i} e\big(g, g_2^{w_{\psi_t^\circ(i)}} g_3\big)^{\mu_{\psi_t^\circ(i)}\varphi_i a_i} \Big\} \\
&= e\big(g, g_6\big)^{s\sum\limits_{i\in[\ell_t]} a_i\zeta_i} \cdot \prod_{i\in[\ell_t]} e\big(g, g_2^{w_{\psi_t^\circ(i)}} g_3\big)^{\mu_{\psi_t^\circ(i)}\cdot\varphi_i a_i} \\
&= e\big(g, g_6\big)^{s\cdot\gamma'\cdot H_{10}(\mathsf{pp})} \cdot \prod_{i\in[\ell_t]} e\big(g, g_2^{w_{\psi_t^\circ(i)}} g_3\big)^{\mu_{\psi_t^\circ(i)}\cdot\varphi_i a_i}
\end{aligned}
$$

$$
\begin{aligned}
J_2 &= \prod_{i\in[\ell_t]} \Big\{ e\big(I_{\psi_t^\circ(i)1}, U_{i2}\big)\cdot e\big(I_{\psi_t^\circ(i)2}, U_{i1}\big)\cdot e\big(I_{\psi_t^\circ(i)3}, U_{i4}\big)\cdot e\big(I_{\psi_t^\circ(i)4}, U_{i3}\big)\Big\}^{a_i} \\
&= \prod_{i\in[\ell_t]} \Big\{ e\Big(h_1^{\mu_{\psi_t^\circ(i)}-\tau_{\psi_t^\circ(i)1}}, \big(g_2^{w_{\psi_t^\circ(i)}} g_3\big)^{-\breve{q}_i b_2}\Big)\cdot e\Big(h_2^{\tau_{\psi_t^\circ(i)1}}, \big(g_2^{w_{\psi_t^\circ(i)}} g_3\big)^{-\breve{q}_i b_1}\Big) \\
&\qquad \times e\Big(h_3^{\mu_{\psi_t^\circ(i)}-\tau_{\psi_t^\circ(i)2}}, \big(g_2^{w_{\psi_t^\circ(i)}} g_3\big)^{-\breve{q}'_i b_4}\Big)\cdot e\Big(h_4^{\tau_{\psi_t^\circ(i)2}}, \big(g_2^{w_{\psi_t^\circ(i)}} g_3\big)^{-\breve{q}'_i b_3}\Big)\Big\}^{a_i} \\
&= \prod_{i\in[\ell_t]} \Big\{ e\big(g, g_2^{w_{\psi_t^\circ(i)}} g_3\big)^{-\breve{q}_i b_1 b_2(\mu_{\psi_t^\circ(i)}-\tau_{\psi_t^\circ(i)1})}\cdot e\big(g, g_2^{w_{\psi_t^\circ(i)}} g_3\big)^{-\breve{q}_i b_1 b_2 \tau_{\psi_t^\circ(i)1}} \\
&\qquad \times e\big(g, g_2^{w_{\psi_t^\circ(i)}} g_3\big)^{-\breve{q}'_i b_3 b_4(\mu_{\psi_t^\circ(i)}-\tau_{\psi_t^\circ(i)2})}\cdot e\big(g, g_2^{w_{\psi_t^\circ(i)}} g_3\big)^{-\breve{q}_i b_3 b_4 \tau_{\psi_t^\circ(i)1}}\Big\}^{a_i}
\end{aligned}
$$

$$
= \prod_{i\in[\ell_t]} e\big(g, g_2^{w_{\psi_t^\circ(i)}} g_3\big)^{-\mu_{\psi_t^\circ(i)}(b_1 b_2\breve{q}_i + b_3 b_4\breve{q}'_i)a_i} = \prod_{i\in[\ell_t]} e\big(g, g_2^{w_{\psi_t^\circ(i)}} g_3\big)^{-\mu_{\psi_t^\circ(i)}\varphi_i a_i}
$$

Hence,

$$J_1 J_2 = e\big(g, g_6\big)^{s\cdot\gamma'\cdot H_{10}(\mathsf{pp})} = h_T^{\,s\cdot H_{10}\left(e(\Pi_1,\Pi_2)^{csk}\right)} = {k_T}^{H_{10}\left(e(\Pi_1,\Pi_2)^{csk}\right)}$$

Now,

$$\begin{aligned}
\breve{J}_1 &= e\Big(\breve{B}, \prod_{i\in[\ell_t]} \Pi_{i2}^{a_i}\Big) \prod_{i\in[\ell_t]} e\Big(\Pi_{i3}\cdot\varepsilon_i^{-1}, I_{\psi_t^\circ(i)6}\Big)^{a_i} \\
&= \prod_{i\in[\ell_t]} \Big\{ e\Big(g^s, g^{\beta\nu_i} g_5^{\varphi_i}\Big)^{a_i} e\Big(g^{\varphi_i}, \big(g_2^{w_{\psi_t^\circ(i)}} g_3\big)^{\mu_{\psi_t^\circ(i)}} g_5^{-s}\Big)^{a_i} \Big\} \\
&= \prod_{i\in[\ell_t]} \Big\{ e\big(g^s, g^{\beta\nu_i}\big)^{a_i} e\big(g^s, g_5^{\varphi_i}\big)^{a_i} e\big(g, g_2^{w_{\psi_t^\circ(i)}} g_3\big)^{\mu_{\psi_t^\circ(i)}\varphi_i a_i} \cdot e\big(g, g_5\big)^{-s\varphi_i a_i} \Big\} \\
&= \prod_{i\in[\ell_t]} \Big\{ e\big(g, g\big)^{s\beta a_i\nu_i} \cdot e\big(g, g_2^{w_{\psi_t^\circ(i)}} g_3\big)^{\mu_{\psi_t^\circ(i)}\varphi_i a_i} \Big\} \\
&= e\big(g, g\big)^{\beta s \sum_{i\in[\ell_t]} a_i\nu_i} \cdot \prod_{i\in[\ell_t]} \Big\{ e\big(g, g_2^{w_{\psi_t^\circ(i)}} g_3\big)^{\mu_{\psi_t^\circ(i)}\varphi_i a_i} \Big\} \\
&= e\big(g, g\big)^{\beta s\cdot\phi_3/\phi_1} \cdot \prod_{i\in[\ell_t]} \Big\{ e\big(g, g_2^{w_{\psi_t^\circ(i)}} g_3\big)^{\mu_{\psi_t^\circ(i)}\varphi_i a_i} \Big\} \\
R_1 &= \breve{J}_1 J_2 = e\big(g, g\big)^{\beta s\cdot\phi_3/\phi_1}
\end{aligned}$$

Since $S \models \Gamma_e$, we get

$$\begin{aligned}
J_3 &= \frac{e\big(C_{1,j}, D_2\big)}{e\big(C_{2,j}, \prod_{y\in B_j} D_y\big)} \\
&= \frac{e\big(g^{\beta s} \prod_{y\in B_j} H_3(y)^{s_j}, g^{r/\phi_2}\big)}{e\big(g^{s_j}, \prod_{y\in B_j} H_3(y)^{r/\phi_2}\big)} \\
&= e\big(g, g\big)^{\beta s\cdot r/\phi_2}
\end{aligned}$$

$$\begin{aligned}
J_4 &= e\big(\breve{B}, D_1\big) \\
&= e\big(g^s, g^{\alpha+\beta r} g^{\beta\phi_3}\big)^{1/\phi_2} \\
&= e\big(g, g\big)^{\alpha\cdot s/\phi_2} e\big(g, g\big)^{\beta r s/\phi_2} e\big(g, g\big)^{\beta s\phi_3/\phi_2} \\
R_2 &= J_3^{-1}\cdot J_4 \\
&= e\big(g, g\big)^{-\beta r s/\phi_2} e\big(g, g\big)^{\alpha\cdot s/\phi_2} e\big(g, g\big)^{\beta r s/\phi_2} e\big(g, g\big)^{\beta s\phi_3/\phi_2} \\
&= e\big(g, g\big)^{\alpha s/\phi_2} e(g, g)^{\beta s\phi_3/\phi_2}
\end{aligned}$$

Now,

$$\begin{aligned}\Omega &= R_1^{-\phi_1} \cdot R_2^{\phi_2} \\ &= \Big(e\big(g,g\big)^{\beta s\phi_3/\phi_1}\Big)^{-\phi_1}\Big(e\big(g,g\big)^{s\alpha/\phi_2}\Big)^{\phi_2}\Big(e\big(g,g\big)^{\beta s\phi_3/\phi_2}\Big)^{\phi_2} \\ &= e\big(g,g\big)^{\alpha s} \\ &= Y^s\end{aligned}$$

$$\delta = H_4(\Omega) = H_4(Y^s)$$

Hence, $H_5\big(\delta||C_0\big) = \mathsf{tag}$

$$C_0 \oplus H_2(\Omega) = (m||\gamma) \oplus H_2(Y^s) \oplus H_2(\Omega) = m||\gamma$$

Now, $g^{H_1(m||\gamma)} = g^s = \breve{B}$ and $g_1^{H_1(m||\gamma)} = g_1^s = C_2$

Therefore, the delegator accepts m as valid message. □

Theorem 12. *If $S \models \Gamma_e$, $S' \models \Gamma_e'$ and $W' \models \Gamma_t'$. then the delegatee can verify the accuracy of search results and recover the original message m from the re-encrypted ciphertext CT.*

Proof. Since $S \models \Gamma_e$, we have

$$\begin{aligned}T &= \frac{e(C_1^{1/H_6(e(\sigma',\sigma'')^{csk})}, rk_0)}{e(C_2, rk_1)\cdot e(C_{1i}, rk_2)\cdot e\big(C_{2i}^{-1}, \prod\limits_{x\in B_i} rk_x\big)} \\ &= \frac{e(g^s, g^{(\alpha+\beta r)H_9(\phi)}g_1^{\eta})}{e\big(g_1^s, g^{\eta}\big)\cdot e\big(g^{\beta s}\prod\limits_{x\in B_i} H_3(x)^{s_i}, g^{rH_9(\phi)}\big)\cdot e\big(g^{-s_i}, \prod\limits_{x\in B_i} H_3(x)^{rH_9(\phi)}\big)} \\ &= \frac{e(g,g)^{\alpha s H_9(\phi)}\cdot e(g,g)^{\beta r s H_9(\phi)}}{e(g,g)^{\beta r s H_9(\phi)}} \\ &= e(g,g)^{\alpha s H_9(\phi)}\end{aligned}$$

In a similar way, as shown in proof of Theorem 11, Q_1 and Q_2 can be computed, and it can be shown that $Q_1 Q_2 = {k_T'}^{H_{10}\left(e(\Pi_1,\Pi_2)^{csk}\right)}$.

In this case, we get

$$Y_1 = e\big(g,g\big)^{\beta s'\phi_3/\phi_1},\ Y_2 = e\big(g,g\big)^{\alpha s'/\phi_2}\cdot e(g,g)^{\beta s'\phi_3/\phi_2}$$

Now, the delegatee can recover the message m as shown below.

$$\begin{aligned}\Omega' &= Y_1^{-\phi_1} \cdot Y_2^{\phi_2} \\ &= \Big(e\big(g,g\big)^{\beta s' \phi_3/\phi_1}\Big)^{-\phi_1} \Big(e\big(g,g\big)^{s'\alpha/\phi_2}\Big)^{\phi_2} \cdot \Big(e\big(g,g\big)^{\beta s' \phi_3/\phi_2}\Big)^{\phi_2} \\ &= e\big(g,g\big)^{\alpha s'}\end{aligned}$$

$\delta' = H_4(\Omega') = H_4\Big(e\big(g,g\big)^{\alpha s'}\Big) = H_4(Y^{s'})$
Hence, $H_5\big(\delta' || C_0'\big) = \mathsf{tag1}$.
Now,

$C_0' \oplus H_2(\Omega') = (\phi || \gamma_1) \oplus H_2\Big(e\big(g,g\big)^{\alpha s'}\Big) \oplus H_2(\Omega') = \phi || \gamma_1.$

$C_0 \oplus H_2(T^{1/H_9(\phi)}) = (m||\gamma) \oplus H_2\Big(e\big(g,g\big)^{\alpha s}\Big) \oplus H_2\Big(e\big(g,g\big)^{\alpha s}\Big) = m||\gamma.$ □

5.4 Security Proof

In the following theorems, we demonstrate the security of our scheme.

Theorem 13. *Our ABPRE-BKS scheme demonstrates IND-CCA2-Or security against a Type-1 adversary, with the DBDH problem assumed to be hard.*

Proof. Let a PPT Type-1 adversary $\mathcal{A}$ breaks IND-CCA2-Or security with non-negligible advantage, then a challenger $\mathcal{B}$ can solve the DBDH problem by interacting with $\mathcal{A}$ as given in IND-CCA2-Or-Type-1 game. Given the DBDH problem instance $(\Delta, A = g^a, B = g^b, C = g^c, Z)$, $\mathcal{B}$ has to ascertain if Z is equal to $e(g,g)^{abc}$ or Z has been randomly selected from $\mathbb{G}_T$.

- sk_{list}: It stores tuples of the form (S, sk_S).
- rk_{list}: It stores tuples of the form $(S, \Gamma_e', W', rk, flag)$ where $flag \in \{1, 0\}$. Here, $flag = 1$ indicates rk is a valid re-encryption key, and $flag = 0$ indicates rk is randomly chosen.

Init. $\mathcal{A}$ sends a challenge attribute $y^\star \in \mathcal{U}$ to $\mathcal{B}$.
Setup. $\mathcal{B}$ generates the system public parameters as follows. It picks $\alpha' \xleftarrow{u} \mathbb{Z}_p^*$ and implicitly sets $\alpha = \alpha' + ab$. It calculates

$$Y = e(g,g)^{\alpha'} \cdot e(A,B)$$

It also chooses $\breve{\beta} \xleftarrow{u} \mathbb{Z}_p^*$ and implicitly sets $\beta = \breve{\beta} + b$. It computes $X = g^{\breve{\beta}} \cdot B$. Next $\mathcal{B}$ samples $z_1, z_2, z_3, z_4, z_5, z_6, z_7, z_8, z_9, z_8', z_9' \xleftarrow{u} \mathbb{Z}_p^*$, sets $g_i = g^{z_i}$ for all $i = 1, 2, 3, 4, 5, 6$, and computes

$$g_7 = g^{z_7} A, g_8 = g^{z_8} A^{z_8'}, g_9 = g^{z_9} A^{z_9'}$$

It chooses ten collision-resistant hash functions $H_1 : \{0,1\}^{2\lambda} \to \mathbb{Z}_p^*$, $H_2 : \mathbb{G}_T \to \{0,1\}^{2\lambda}$, $H_3 : \{0,1\}^* \to \mathbb{G}$, $H_4 : \mathbb{G}_T \to \mathbb{Z}_p^*$, $H_5 : \{0,1\}^* \to \{0,1\}^{\ell_{H_5}}$, $H_6 : \mathbb{G}_T \to \mathbb{Z}_p^*$, $H_7 : \{0,1\}^* \to \mathbb{Z}_p^*$, $H_8 : \{0,1\}^* \to \mathbb{G}$, $H_9 : \{0,1\}^\lambda \to \mathbb{Z}_p^*$, $H_{10} : \mathbb{G}_T \to \mathbb{Z}_p^*$, in which H_3 is calculated as follows.

H_3^{list} : Let $y \in \mathcal{U}$ be an attribute with $y \neq y^\star$. $\mathcal{A}$ queries H_3 for $H_3(y)$, if (y, v_y, g^{v_y}) exists in H_3^{list}, returns g^{v_y}. Otherwise, choose $v_y \xleftarrow{u} \mathbb{Z}_p^*$ and returns g^{v_y}. Adds (y, v_y, g^{v_y}) to H_3^{list}. If $y = y^\star$, choose $v_{y^\star} \xleftarrow{u} \mathbb{Z}_p^*$ and returns $g^{v_{y^\star}} C$. Adds $(y^\star, v_{y^\star}, g^{v_{y^\star}} C)$ to H_3^{list}.

$\mathcal{B}$ sets $mpk = (\Delta, X, Y, \{g_i\}_{i=1}^9, M, \{H_i\}_{i=1}^{10})$.

Next $\mathcal{B}$ chooses $\gamma', \beta', b_1, b_2, b_3, b_4 \xleftarrow{u} \mathbb{Z}_p^*$, computes $h_1 = g^{b_1}, h_2 = g^{b_2}$, $h_3 = g^{b_3}, h_4 = g^{b_4}$, $h_T = e(g, g_6)^{\gamma'}$, $\hat{X} = X^{\beta'}$ and sets $tpk = (h_T, h_1, h_2, h_3, h_4)$, $tsk = (\gamma', \{b_i\}_{i=1}^4)$, $cpk = \hat{X}$ and $csk = \beta'$.

$\mathcal{B}$ sends $PP = (mpk, tpk, cpk)$ and csk to $\mathcal{A}$.

Phase I. The following set of queries is posed by $\mathcal{A}$.

- $\mathcal{O}_{sk}(S)$: Here $y^\star \notin S$. $\mathcal{B}$ searches sk_{list}. If (S, sk_S) already present, output sk_S. Else, it picks $\breve{r} \xleftarrow{u} \mathbb{Z}_p^*$ and implicitly define $r = \breve{r} - a$. Compute

$$K_1 = g^{\alpha'} g^{\breve{\beta}\breve{r}} A^{-\breve{\beta}} B^{\breve{r}}, K_2 = g^{\breve{r}} A^{-1}, K_y = g^{v_y \breve{r}} A^{-v_y}, \ \forall y \in S$$

  $\mathcal{B}$ adds (S, sk_S) to sk_{list}.

- $\mathcal{O}_{rk}(S, W', \Gamma_e')$: $\mathcal{B}$ searches rk_{list}. If $(S, \Gamma_e', W', rk, *)$ exists, it returns rk. Otherwise it does the following.

  - If $y^\star \in S$ but sk_{list} does not contain any pair $(S', sk_{S'})$, where $S' \models \Gamma_e'$, $\mathcal{B}$ chooses each components of rk randomly. Adds $(S, \Gamma_e', W', rk, 0)$ to rk_{list}.
  - Otherwise, if $y^\star \notin S$, $\mathcal{B}$ queries $\mathcal{O}_{sk}(S)$ and generates rk using the obtained sk_S. Adds (S, sk_S) and $(S, \Gamma_e', W', rk, 1)$ to sk_{list} and rk_{list}, respectively.

- $\mathcal{O}_{re}(CT, S, W', \Gamma'_e)$: If equations (5.1),(5.2) and (5.3) do not hold simultaneously or $y^\star \in S$ and $(S', sk_{S'})$ in sk_{list} with $S' \models \Gamma'_e$, output $\perp$. Else, if there exists a tuple $(S, \Gamma'_e, W', rk, *)$ in rk_{list}, then CT will be re-encrypted using rk. Else, $\mathcal{B}$ issues $\mathcal{O}_{rk}(S, W', \Gamma'_e)$ to get rk, where $S \not\models \Gamma'_e$. Next, CT is re-encrypted by $\mathcal{B}$ using rk. The final result will be given to $\mathcal{A}$.

- $\mathcal{O}_{token}(\Gamma_t, S)$: $\mathcal{B}$ performs the following cases.

  - If $y^\star \in S$, $\mathcal{B}$ chooses $\xi, \xi', d, d', d'', \ddot{r}, \breve{q}_i, \breve{q}'_i \xleftarrow{u} \mathbb{Z}_p^*$, and implicitly sets $\phi_1 = \alpha d''/(dd'), \phi_2 = \alpha/d, \phi_3 = \alpha d''/d$. After this, $\mathcal{B}$ computes the token $tok = (\Gamma_t^\circ, \Pi_1, \Pi_2, \{\Pi_{i1}, \Pi_{i2}, \Pi_{i3}, U_{i1}, U_{i2}, U_{i3}, U_{i4}\}_{i\in[\ell_t]}, D_1, D_2, \{D_y\}_{y\in S})$ corresponding to the keyword policy $\Gamma_t = (\mathcal{K}_t, \psi_t^\circ, \{w_{\psi_t^\circ(i)}\}_{i\in[\ell_t]})$ as follows. Set $\ddot{r} = r/\phi_2$,

$$D_1 = g^d X^{\ddot{r}+d''}, D_2 = g^{\ddot{r}}, D_y = \begin{cases} g^{v_y \ddot{r}}, & \text{if } y \neq y^\star \\ (g^{v_{y^\star}} C)^{\ddot{r}}, & \text{if } y = y^\star \end{cases}$$

    $\Pi_1 = g^\xi, \mathsf{pp} = e(\Pi_1, \hat{X})^{\xi'}$, $\vec{\eta} = \big(\gamma' H_{10}(\mathsf{pp}), y_2, y_3, \ldots, y_{\ell_t}\big)$, where $y_2, y_3, \ldots, y_{\ell_t} \xleftarrow{u} \mathbb{Z}_p^*$, and $\zeta_i = \mathcal{K}_t^{(i)} \cdot \vec{\eta}$, for all $i \in [\ell_t]$.
    Also, set $\vec{\eta}' = \big(\phi_3/\phi_1 = d', y'_2, y'_3, \ldots, y'_{\ell_t}\big)$, where $y'_2, y'_3, \ldots, y'_{\ell_t} \xleftarrow{u} \mathbb{Z}_p^*$, and $\nu_i = \mathcal{K}_t^{(i)} \cdot \vec{\eta}'$, for all $i \in [\ell_t]$.
    Now, $\mathcal{B}$ calculates the rest of the token components as shown in construction and sends the token tok to $\mathcal{A}$. Note that, in this case, $\mathcal{B}$ does not know $sk_{tr} = (\phi_1, \phi_2)$; however, the simulation of tok is similar to the original construction. It can be seen from Remark 10.

  - If $y^\star \notin S$, it computes $sk_S \leftarrow \mathcal{O}_{sk}(S)$. Then, it returns $(tok, sk_{tr}) \leftarrow \mathsf{TokenGen}(PP, \Gamma_t, tsk, sk_S)$ to $\mathcal{A}$.

- $\mathcal{O}_{search}(\Gamma_t, S, CT)$: Taking a ciphertext CT, a Boolean keyword formula Γ_t, and an attribute set S, it outputs the search result $1/0 \leftarrow \mathsf{Search}(PP, CT, csk, tok)$, where $tok \leftarrow \mathcal{O}_{token}(\Gamma_t, S)$.

- $\mathcal{O}_{decrypt}(\Gamma_t, S, CT)$: $\mathcal{B}$ responds as follows,

  (i) Suppose CT is an original ciphertext. If equations (5.1),(5.2) and (5.3) do not hold simultaneously or $S \not\models \Gamma_e$ or $\mathcal{O}_{search}(\Gamma_t, S, CT) \to 0$, return $\perp$. Else, $\mathcal{B}$ does the following.

– If $y^\star \in S$, $\mathcal{B}$ does not have the transformed secret key sk_{tr} according to $\mathcal{O}_{token}(\Gamma_t, S)$ and hence it proceeds in the following way. It calculates

$$\Omega = e(\breve{B}, g^{\alpha'}) \cdot e\Big(\bar{C}(\breve{B})^{-(\rho z_7 + \epsilon z_8 + z_9)}, B^{(\rho + z'_8\epsilon + z'_9)^{-1}}\Big) = Y^s$$

and $\delta = H_4(\Omega)$. Note that $\rho + z'_8\epsilon + z'_9 = 0$ happens with probability at most $1/p$ and hence $\rho + z'_8\epsilon + z'_9 \neq 0$ since p is very large prime number. Next, $\mathcal{B}$ checks whether $H_5(\delta||C_0) \stackrel{?}{=} \mathsf{tag}$. If this is not true, it returns $\perp$. Otherwise, it computes $m||\gamma = C_0 \oplus H_2(\Omega)$, and returns m to $\mathcal{A}$ if $\breve{B} = g^{H_1(m||\gamma)}$ and $C_2 = g_1^{H_1(m||\gamma)}$.

– If $y^\star \notin S$, $\mathcal{B}$ knows sk_{tr}. Hence, it computes the transformed ciphertext $CT_{tr} \leftarrow \mathsf{Transform}(PP, CT_m, tok)$ and returns to $\mathcal{A}$ the output of the algorithm $\mathsf{Verify\text{-}and\text{-}Decrypt}(PP, CT_{tr}, sk_{tr})$.

(ii) Suppose CT is a re-encrypted ciphertext. If equations (5.5) and (5.6) do not hold simultaneously or $S \not\models \Gamma'_e$ or $\mathcal{O}_{search}(\Gamma_t, S, CT) \to 0$, return $\perp$. Otherwise, $\mathcal{B}$ does the following.

– If $y^\star \in S$, $\mathcal{B}$ does not have the transformed secret key sk_{tr} according to $\mathcal{O}_{token}(\Gamma_t, S)$ and hence it calculates

$$\Omega' = e(\breve{D}, g^{\alpha'}) \cdot e\Big(\bar{C}'(\breve{D})^{-(\rho' z_7 + \epsilon' z_8 + z_9)}, B^{(\rho' + z'_8\epsilon' + z'_9)^{-1}}\Big) = Y^{s'}$$

and $\delta' = H_4(\Omega')$. Note that $\rho' + z'_8\epsilon' + z'_9 = 0$ happens with probability at most $1/p$ and hence $\rho' + z'_8\epsilon' + z'_9 \neq 0$ since p is very large prime number. Next, $\mathcal{B}$ checks whether $H_5(\delta'||C'_0) \stackrel{?}{=} \mathsf{tag1}$. If this is not true, it returns $\perp$. Otherwise, it computes $\phi||\gamma_1 = C'_0 \oplus H_2(\Omega')$ and obtains ϕ if $\breve{D} = g^{H_1(\phi||\gamma_1)}$.
Next, $\mathcal{B}$ computes $m||\gamma = C_0 \oplus H_2(T^{1/H_9(\phi)})$ and returns m to $\mathcal{A}$ if $\breve{B} = g^{H_1(m||\gamma)}$ and $C_2 = g_1^{H_1(m||\gamma)}$.

– If $y^\star \notin S$, $\mathcal{B}$ knows sk_{tr}. Hence, it computes the transformed ciphertext $CT_{tr} \leftarrow \mathsf{Transform}(PP, CT_m, tok)$ and returns to $\mathcal{A}$ the output of the algorithm $\mathsf{Verify\text{-}and\text{-}Decrypt}(PP, CT_{tr}, sk_{tr})$.

Challenge. $\mathcal{A}$ selects two equal length messages $m_0^\star$ and $m_1^\star$, a challenge encryption policy $\Gamma_e^\star = B_1^\star \vee B_2^\star \vee \ldots \vee B_n^\star$, and a challenge keyword set $W^\star$, and sends to $\mathcal{B}$. Now, $\mathcal{B}$ selects a bit $i \xleftarrow{u} \{0,1\}$, outputs $CT^\star \leftarrow \mathsf{Encrypt}(PP, m_i^\star, \Gamma_e^\star \wedge y^\star, W^\star)$, where $\Gamma_e^\star \wedge y^\star = (B_1^\star \cup \{y^\star\}) \vee (B_2^\star \cup \{y^\star\}) \vee \cdots \vee (B_n^\star \cup \{y^\star\})$, in the following way.

- Pick $\gamma^\star \in \{0,1\}^\lambda$ and implicitly define $H_1(m_i^\star||\gamma^\star) = c$.
- Select $\delta', \delta'' \xleftarrow{\text{u}} \mathbb{Z}_p^*$, and set $\sigma' = g^{\delta'}, \sigma'' = X^{\delta''}$.
- Implicitly define $s_i = \breve{s}_i - b$, for all $i \in [n]$ and compute

$$C_0^\star = (m_i^\star||\gamma^\star) \oplus H_2\big(Z \cdot e(g^{\alpha'}, C)\big), C_1^\star = C^{H_6(e(\sigma', \hat{X})^{\delta''})}, C_2^\star = C^{z_1},$$

$$C_{1,i}^\star = C^{\breve{\beta}} g^{v_{y^\star}\breve{s}_i} B^{-v_{y^\star}} C^{\breve{s}_i} \prod_{\substack{y \in B_i^\star \\ y \neq y^\star}} \big(g^{v_y \breve{s}_i} B^{-v_y}\big), C_{2,i}^\star = g^{\breve{s}_i} B^{-1},$$

$$\delta^\star = H_4\big(Z \cdot e(g^{\alpha'}, C)\big), \mathsf{tag}^\star = H_5(\delta^\star||C_0^\star).$$

  Note that $C_{1,i}^\star$ is corresponding to $B_i^\star \cup \{y^\star\}$ in the encryption policy $\Gamma_e^\star \wedge y^\star$. Set $ct_e^\star = (\Gamma_e^\star \wedge y^\star, \sigma', \sigma'', C_0^\star, C_1^\star, C_2^\star, \{C_{1,i}^\star, C_{2,i}^\star\}_{i \in [n]})$.
- Choose $\mu_{\mathcal{W}}, \tau_{\mathcal{W}1}, \tau_{\mathcal{W}2} \xleftarrow{\text{u}} \mathbb{Z}_p^*$, for all $[\mathcal{W} : w^\star] \in W^\star$, where $W^{\star\circ} = \{\mathcal{W}\}$ and compute

$$I_{\mathcal{W}1}^\star = h_1^{\mu_{\mathcal{W}} - \tau_{\mathcal{W}1}}, I_{\mathcal{W}2}^\star = h_2^{\tau_{\mathcal{W}1}}, I_{\mathcal{W}3}^\star = h_3^{\mu_{\mathcal{W}} - \tau_{\mathcal{W}2}}, I_{\mathcal{W}4}^\star = h_4^{\tau_{\mathcal{W}2}},$$

$$I_{\mathcal{W}5}^\star = (g_2^{w^\star} g_3)^{\mu_{\mathcal{W}}} C^{-z_4}, I_{\mathcal{W}6}^\star = (g_2^{w^\star} g_3)^{\mu_{\mathcal{W}}} C^{-z_5}, k_T^\star = e(C, g_6)^{\gamma'}$$

  Set $ct_k^\star = (W^{\star\circ}, k_T^\star, \{I_{\mathcal{W}1}^\star, I_{\mathcal{W}2}^\star, I_{\mathcal{W}3}^\star, I_{\mathcal{W}4}^\star, I_{\mathcal{W}5}^\star, I_{\mathcal{W}6}^\star\}_{\mathcal{W} \in W^{\star\circ}})$
- Choose $\varepsilon^\star = \frac{-\varrho^\star - z_9'}{z_8'}$ and compute $\bar{C}^\star = C^{(z_7\varrho^\star + z_8\varepsilon^\star + z_9)}$, where $\varrho^\star = H_7(ct_e^\star||ct_k^\star||\mathsf{tag}^\star)$. Finally, output the challenge original ciphertext $CT^\star = (ct_e^\star, ct_k^\star, \mathsf{tag}^\star, \bar{C}^\star, \varepsilon^\star)$.

Phase II. $\mathcal{A}$ continues to query similar to **Phase I**, with the exception of the limitations imposed by the IND-CCA2-Or-Type-1 game.

Guess. $\mathcal{A}$ outputs a bit $i' \in \{0,1\}$. If $i' = i$, $\mathcal{A}$ wins.

If $Z = e(g,g)^{abc}$, the challenge ciphertext $CT^\star$ is valid. But, if $Z \xleftarrow{\text{u}} \mathbb{G}_T$, the challenge ciphertext $CT^\star$ is independent of the bit i in the view of $\mathcal{A}$. Hence, if $\mathcal{A}$ has a non-negligible advantage in winning the game, $\mathcal{B}$ can solve DBDH problem with non-negligible advantage. □

Theorem 14. *Our ABPRE-BKS scheme demonstrates IND-CCA2-Or security against a Type-2 adversary, with the DBDH problem assumed to be hard.*

Proof. Let a PPT Type-2 adversary $\mathcal{A}$ breaks IND-CCA2-Or security with non-negligible advantage, then a challenger $\mathcal{B}$ can solve the DBDH problem by interacting with $\mathcal{A}$ as given in IND-CCA2-Or-Type-2 game. Given the DBDH problem

instance $(\Delta, A = g^a, B = g^b, C = g^c, Z)$, $\mathcal{B}$ has to ascertain if Z is equal to $e(g,g)^{abc}$ or Z has been randomly selected from $\mathbb{G}_T$.

Setup. $\mathcal{B}$ generates the system public parameters as follows. It picks $\alpha, \beta \xleftarrow{u} \mathbb{Z}_p^*$ and sets $Y = e(g,g)^\alpha, X = g^\beta$. Next, $\mathcal{B}$ samples $\{g_i\}_{i=1}^9 \xleftarrow{u} \mathbb{G}$, and chooses ten collision resistant hash functions $\{H_i\}_{i=1}^{10}$ same as given in construction, in which H_6 is calculated as follows.
H_6^{list} : $\mathcal{A}$ queries H_6 with g^s. If $\left(C_1 = g^{sH_6(e(\sigma', \hat{X})^{\delta''})}, g^s, H_6(e(\sigma', \hat{X})^{\delta''})\right)$ exists in H_6^{list}, returns C_1. Otherwise, it picks $\delta', \delta'' \xleftarrow{u} \mathbb{Z}_p^*$, computes $\sigma' = g^{\delta'}$,
$C_1 = g^{sH_6(e(\sigma', \hat{X})^{\delta''})}$ and returns C_1. Next, it adds the tuple $\left(g^s, H_6(e(\sigma', \hat{X})^{\delta''}), C_1\right)$ to H_6^{list}.
$\mathcal{B}$ sets $mpk = (\Delta, X, Y, \{g_i\}_{i=1}^9, M, \{H_i\}_{i=1}^{10})$ and $msk = g^\alpha$.

Next $\mathcal{B}$ chooses $\gamma', b_1, b_2, b_3, b_4 \xleftarrow{u} \mathbb{Z}_p^*$, computes $h_1 = g^{b_1}, h_2 = g^{b_2}, h_3 = g^{b_3}$, $h_4 = g^{b_4}$, $h_T = e(g, g_6)^{\gamma'}$, and sets $tpk = (h_T, h_1, h_2, h_3, h_4)$, $tsk = (\gamma', \{b_i\}_{i=1}^4)$. It implicitly sets $csk = a$ and $cpk = A^\beta$.

$\mathcal{B}$ sends $PP = (mpk, tpk, cpk)$ to $\mathcal{A}$.

Phase I. $\mathcal{A}$ queries for the oracles $\mathcal{O}_{sk}, \mathcal{O}_{rk}, \mathcal{O}_{re}, \mathcal{O}'_{token}, \mathcal{O}'_{search}$ and $\mathcal{O}_{decrypt}$. Since $\mathcal{B}$ knows msk and tsk, it runs suitable algorithms to answer $\mathcal{A}$'s queries to $\mathcal{O}_{sk}, \mathcal{O}_{rk}, \mathcal{O}_{re}$, $\mathcal{O}'_{token}$ and $\mathcal{O}_{decrypt}$.

- $\mathcal{O}'_{search}(\Gamma_t, S, CT)$: Suppose CT is an original (resp. re-encrypted) ciphertext. If the entry corresponding to C_1 (resp. C_1') is not found in H_6 list, $\mathcal{B}$ returns $\perp$. Else, it obtains $tok \leftarrow \mathcal{O}'_{token}(\Gamma_t, S)$ and returns $1/0 \leftarrow \mathsf{Search}(PP, CT, \star, tok)$ to $\mathcal{A}$.

Challenge. $\mathcal{A}$ selects two equal length messages $m_0^\star$ and $m_1^\star$, a challenge keyword set $W^\star$ and a challenge encryption policy $\Gamma_e^\star = B_1^\star \vee B_2^\star \vee \ldots \vee B_n^\star$, and sends them to $\mathcal{B}$. Now, $\mathcal{B}$ selects a bit $i \xleftarrow{u} \{0,1\}$, returns $CT^\star$ in the following way.
Pick $\gamma^\star \in \{0,1\}^\lambda$ and compute $H_1(m_i^\star || \gamma^\star) = s$. Choose $s_i \xleftarrow{u} \mathbb{Z}_p^*$, for all $i \in [n]$ and also pick $\varepsilon, \mu_{\mathcal{W}}, \tau_{\mathcal{W}1}, \tau_{\mathcal{W}2} \xleftarrow{u} \mathbb{Z}_p^*$, for all $\mathcal{W} \in W^\circ$. Compute

$$\sigma'^\star = B, \sigma''^\star = C^\beta, C_1^\star = g^{sH_6(Z^\beta)}$$

The remaining elements of $CT^\star$ will be calculated in a similar manner as the construction. Finally, $\mathcal{A}$ receives the challenge ciphertext $CT^\star$ from $\mathcal{B}$.

Phase II. $\mathcal{A}$ continues to query similar to **Phase I**, with the exception of the limitations imposed by the IND-CCA2-Or-Type-2 game.

Guess. $\mathcal{A}$ outputs a bit $i' \in \{0,1\}$. If $i' = i$, $\mathcal{A}$ wins.

If Z is equal to $e(g,g)^{abc}$, the challenge ciphertext $CT^\star$ is accuratately derived. However, if Z is randomly chosen from $\mathbb{G}_T$, the $CT^\star$ is not dependent on i in the view of $\mathcal{A}$. Hence, if $\mathcal{A}$ has a non-negligible advantage in winning the game, $\mathcal{B}$ can solve DBDH problem with non-negligible advantage. □

Theorem 15. *Our ABPRE-BKS scheme demonstrates IND-CCA2-Re security against a Type-1 adversary, with the DBDH problem assumed to be hard.*

Proof. Let a PPT Type-1 adversary $\mathcal{A}$ breaks IND-CCA2-Re security with non-negligible advantage, then a challenger $\mathcal{B}$ can solve the DBDH problem by interacting with $\mathcal{A}$ as given in IND-CCA2-Re-Type-1 game. Given the DBDH problem tuple $(\Delta, A = g^a, B = g^b, C = g^c, Z)$, $\mathcal{B}$ has to ascertain if Z is equal to $e(g,g)^{abc}$ or Z has been randomly selected from $\mathbb{G}_T$.

Init, **Setup**, and **Phase I** are similar to that of Theorem 13.

Challenge. $\mathcal{A}$ sends to $\mathcal{B}$ two equal length messages $m_0^\star$ and $m_1^\star$, a challenge encryption policy $\Gamma_e^\star$ and a challenge keyword set $W^\star$. Now, $\mathcal{B}$ selects $i \xleftarrow{u} \{0,1\}$, computes a re-encrypted challenge ciphertext
$CT^\star \leftarrow \mathsf{Re\text{-}Encrypt}\big(PP, csk, \mathsf{Encrypt}(PP, m_i^\star, \Gamma_e, W), rk^\star\big)$, where
$rk^\star \leftarrow \mathsf{Re\text{-}KeyGen}(PP, sk_S, \Gamma_e^\star \wedge y^\star, W^\star), S \models \Gamma_e$,
$\Gamma_e^\star \wedge y^\star = (B_1^\star \cup \{y^\star\}) \vee (B_2^\star \cup \{y^\star\}) \vee \cdots \vee (B_n^\star \cup \{y^\star\})$, in the following way.

- Pick $\gamma^\star, \phi^\star, \gamma_1^\star \in \{0,1\}^\lambda$ and set $H_1(m_i^\star||\gamma^\star) = s$.
- Select $\delta'^\star, \delta''^\star \xleftarrow{u} \mathbb{Z}_p^*$ and set $\sigma'^\star = g^{\delta'^\star}, \sigma''^\star = X^{\delta''^\star}$.
- Compute

$$C_0^\star = (m_i^\star||\gamma^\star) \oplus H_2\big(e(g,g)^{\alpha' s} e(A,B)^s\big), C_1^\star = g^{sH_6(e(\sigma'^\star, \hat{X})^{\delta''^\star})},$$

$$C_2^\star = g_1^s, T^\star = e(g,g)^{\alpha' s H_9(\phi^\star)} e(A,B)^{sH_9(\phi^\star)}$$

- Implicitly define $H_1(\phi^\star||\gamma_1^\star) = c$.
- Select $\delta_1^\star, \delta_2^\star \xleftarrow{u} \mathbb{Z}_p^*$, and set $\sigma_1^\star = g^{\delta_1^\star}, \sigma_2^\star = X^{\delta_2^\star}$.
- Implicitly define $s_i' = \breve{s}_i' - b$, for all $i \in [n]$ and compute

$$C_0'^\star = (\phi^\star||\gamma_1^\star) \oplus H_2\big(Z \cdot e(g^{\alpha'}, C)\big), C_1'^\star = C^{H_6\left(e(\sigma_1^\star, \hat{X})^{\delta_2^\star}\right)},$$

$$C'^{\star}_{1,i} = C^{\breve{\beta}} g^{v_{y^\star}\breve{s}'_i} B^{-v_{y^\star}} C^{\breve{s}'_i} \prod_{\substack{y \in B^\star_i \\ y \neq y^\star}} \left(g^{v_y \breve{s}'_i} B^{-v_y}\right), C'^{\star}_{2,i} = g^{\breve{s}'_i} B^{-1},$$

$$\delta'^{\star} = H_4\big(Z \cdot e(g^{\alpha'}, C)\big), \mathsf{tag}^\star_1 = H_5\big(\delta'^{\star} || C'^{\star}_0\big).$$

Note that $C'^{\star}_{1,i}$ is corresponding to $B^\star_i \cup \{y^\star\}$ in the encryption policy $\Gamma^\star_e \wedge y^\star$. Set $ct'^{\star}_e = (\Gamma^\star_e \wedge y^\star, \sigma^\star_1, \sigma^\star_2, C'^{\star}_0, C'^{\star}_1, \{C'^{\star}_{1,i}, C'^{\star}_{2,i}\}_{i\in[n]})$.

– Choose $\mu'_{\mathcal{W}}, \tau'_{\mathcal{W}1}, \tau'_{\mathcal{W}2} \xleftarrow{\text{u}} \mathbb{Z}^*_p$, for all $[\mathcal{W} : w^\star] \in W^\star$, where $W^{\star\circ} = \{\mathcal{W}\}$ and compute

$$I'^{\star}_{\mathcal{W}1} = h_1^{\mu'_{\mathcal{W}} - \tau'_{\mathcal{W}1}}, I'^{\star}_{\mathcal{W}2} = h_2^{\tau'_{\mathcal{W}1}}, I'^{\star}_{\mathcal{W}3} = h_3^{\mu'_{\mathcal{W}} - \tau'_{\mathcal{W}2}}, I'^{\star}_{\mathcal{W}4} = h_4^{\tau'_{\mathcal{W}2}},$$

$$I'^{\star}_{\mathcal{W}5} = (g_2^{w^\star} g_3)^{\mu'_{\mathcal{W}}} C^{-z_4}, I'^{\star}_{\mathcal{W}6} = (g_2^{w^\star} g_3)^{\mu'_{\mathcal{W}}} C^{-z_5}, k'^{\star}_T = e(C, g_6)^{\gamma'}$$

Set $ct'^{\star}_k = (W^{\star\circ}, k'^{\star}_T, \{I'^{\star}_{\mathcal{W}1}, I'^{\star}_{\mathcal{W}2}, I'^{\star}_{\mathcal{W}3}, I'^{\star}_{\mathcal{W}4}, I'^{\star}_{\mathcal{W}5}, I'^{\star}_{\mathcal{W}6}\}_{\mathcal{W}\in W^{\star\circ}})$

– Choose $\varepsilon'^{\star} = \frac{-\varrho'^{\star} - z'_9}{z'_8}$ and compute $\bar{C}'^{\star} = C^{(z_7\varrho'^{\star} + z_8\varepsilon'^{\star} + z_9)}$, where $\varrho'^{\star} = H_7(ct'^{\star}_e || ct'^{\star}_k || \mathsf{tag}^\star_1 || S)$. Define $rk^\star_3 = (ct'^{\star}_e, ct'^{\star}_k, \mathsf{tag}^\star_1, \bar{C}'^{\star}, \varepsilon'^{\star})$. Finally, generate the challenge re-encrypted ciphertext $CT^\star = (S, C^\star_0, C^\star_1, C^\star_2, T^\star, rk^\star_3)$.

Phase II. $\mathcal{A}$ continues to query similar to **Phase I**, with the exception of the limitations imposed by the IND-CCA2-Re-Type-1 game.
Guess. $\mathcal{A}$ outputs a bit $i' \in \{0, 1\}$. If $i' = i$, $\mathcal{A}$ wins.

If Z is equal to $e(g, g)^{abc}$, the challenge ciphertext $CT^\star$ is accuratately derived. However, if Z is randomly chosen from $\mathbb{G}_T$, the $CT^\star$ is not dependent on i in the view of $\mathcal{A}$. Hence, if $\mathcal{A}$ has a non-negligible advantage in winning the game, $\mathcal{B}$ can solve DBDH problem with non-negligible advantage. □

Theorem 16. *Our ABPRE-BKS scheme demonstrates IND-CCA2-Re security against a Type-2 adversary, with the DBDH problem assumed to be hard.*

Proof. Consider, a PPT Type-2 adversary $\mathcal{A}$ breaks IND-CCA2-Re security with non-negligible advantage, and a challenger $\mathcal{B}$ solves the DBDH problem by communicating with $\mathcal{A}$ as given in IND-CCA2-Re-Type-2 game. Given the DBDH problem tuple $(\Delta, A = g^a, B = g^b, C = g^c, Z)$, $\mathcal{B}$ has to ascertain if Z is equal to $e(g, g)^{abc}$ or Z has been randomly selected from $\mathbb{G}_T$.
Setup. This phase's simulation follows the same pattern as Theorem 14.
Phase I. The following are $\mathcal{B}$'s responses to $\mathcal{A}$'s queries.

- $\mathcal{O}'_{search}(\Gamma_t, S, CT)$: Here, CT is a re-encrypted ciphertext. If the entry corresponding to C'_1 is not available in H_6^{list}, output $\perp$. Otherwise, $\mathcal{B}$ gets $tok \leftarrow \mathcal{O}'_{token}(\Gamma_t, S)$ and returns $1/0 \leftarrow \mathsf{Search}(PP, CT, \star, tok)$ to $\mathcal{A}$.

The simulation of other oracles is same as that of Theorem 16.

Challenge. After Phase I is over, $\mathcal{A}$ selects two equal length messages $m_0^\star$ and $m_1^\star$, a challenge keyword set $W^\star$ and a challenge encryption policy $\Gamma_e^\star$, and sends them to $\mathcal{B}$. Now, $\mathcal{B}$ picks $i \xleftarrow{u} \{0,1\}$ and produces the challenge ciphertext $CT^\star$ in the following way.

Pick $\phi^\star, \gamma_1^\star \in \{0,1\}^\lambda$ and compute $H_1(\phi^\star || \gamma_1^\star) = s'$. Choose $s'_i \xleftarrow{u} \mathbb{Z}_p^*$, for all $i \in [n]$ and also pick $\varepsilon', \mu'_{\mathcal{W}}, \tau'_{\mathcal{W}1}, \tau'_{\mathcal{W}2} \xleftarrow{u} \mathbb{Z}_p^*$, for all $\mathcal{W} \in W^{\star\circ}$. Compute the components of $rk_3^\star$ as given below.

$$\sigma_1^\star = B, \sigma_2^\star = C^\beta, C_1'^\star = g^{s' H_6(Z^\beta)}$$

The other components of the challenge re-encrypted ciphertext $CT^\star$ can be computed as given in the construction.

Phase II. $\mathcal{A}$ continues to query similar to **Phase I**, with the exception of the limitations imposed by the IND-CCA2-Re-Type-2 game.

Guess. $\mathcal{A}$ outputs a guess $i' \in \{0,1\}$. If $i' = i$, $\mathcal{A}$ wins.

If Z is equal to $e(g,g)^{abc}$, the challenge ciphertext $CT^\star$ is accuratately derived. However, if Z is randomly chosen from $\mathbb{G}_T$, the $CT^\star$ is not dependent on i in the view of $\mathcal{A}$. Hence, if $\mathcal{A}$ has a non-negligible advantage in winning the game, $\mathcal{B}$ can solve DBDH problem with non-negligible advantage. □

Theorem 17. *Our ABPRE-BKS ensures IND-CKA$_{ct}$ security, assuming the hardness of the DBDH problem.*

Proof. Let a PPT adversary $\mathcal{A}$ without having the cloud secret key *csk* breaks the IND-CKA$_{ct}$ security with non-negligible advantage, then a challenger $\mathcal{B}$ can solve the DBDH problem by interacting with $\mathcal{A}$ as given in IND-CKA$_{ct}$ game. Given the DBDH problem instance $(\Delta, A = g^a, B = g^b, C = g^c, Z)$, $\mathcal{B}$ has to ascertain if Z is equal to $e(g,g)^{abc}$ or Z has been randomly selected from $\mathbb{G}_T$.

Setup. Same as described in Theorem 14.

Phase I. $\mathcal{A}$ queries for the oracles $\mathcal{O}'_{sk}, \mathcal{O}'_{rk}, \mathcal{O}''_{token}, \mathcal{O}'_{search}$ and $\mathcal{O}_{decrypt}$. Since $\mathcal{B}$ has *msk* and *tsk*, it runs suitable algorithms to answer $\mathcal{A}$'s queries.

Challenge. $\mathcal{A}$ sends to $\mathcal{B}$ two equal size keyword sets $W_0^\star$ and $W_1^\star$ (where either $W_0^\star \models \Gamma_t \wedge W_1^\star \models \Gamma_t$ or $W_0^\star \not\models \Gamma_t \wedge W_1^\star \not\models \Gamma_t$ for all Γ_t submitted to

$\mathcal{O}''_{token}, \mathcal{O}'_{search}, \mathcal{O}_{decrypt}$ in Phase I), a challenge message $m^\star$ and a challenge encryption policy $\Gamma_e^\star$. Now $\mathcal{B}$ selects a bit $i \xleftarrow{u} \{0,1\}$, outputs an original ciphertext $CT^\star \leftarrow \mathsf{Encrypt}(PP, m^\star, \Gamma_e^\star, W_i^\star)$, or a re-encrypted ciphertext $CT^\star \leftarrow \mathsf{Re\text{-}Encrypt}\big(PP, csk, \mathsf{Encrypt}(PP, m^\star, \Gamma_e, W), rk^\star\big)$, where $rk^\star \leftarrow \mathsf{Re\text{-}KeyGen}(PP, sk_S, \Gamma_e^\star, W_i^\star), S \models \Gamma_e$, in the following way.

- If $CT^\star$ is an original ciphertext, pick $\gamma^\star \in \{0,1\}^\lambda$ and compute $H_1(m^\star||\gamma^\star) = s$. Choose $s_i \xleftarrow{u} \mathbb{Z}_p^*$, for all $i \in [n]$ and also pick $\varepsilon, \mu_{\mathcal{W}}, \tau_{\mathcal{W}1}, \tau_{\mathcal{W}2} \xleftarrow{u} \mathbb{Z}_p^*$ for all $\mathcal{W} \in W^\circ$. Compute

$$\sigma'^\star = B, \sigma''^\star = C^\beta, C_1^\star = g^{sH_6(Z^\beta)}$$

  The other components of the challenge original ciphertext $CT^\star$ can be computed as given in the construction.

- If $CT^\star$ is a re-encrypted ciphertext, pick $\phi^\star, \gamma_1^\star \in \{0,1\}^\lambda$ and calculate $H_1(\phi^\star||\gamma_1^\star) = s'$. Choose $s'_i \xleftarrow{u} \mathbb{Z}_p^*$, for all $i \in [n]$ and also pick $\varepsilon', \mu'_{\mathcal{W}}, \tau'_{\mathcal{W}1}, \tau'_{\mathcal{W}2} \xleftarrow{u} \mathbb{Z}_p^*$, for all $\mathcal{W} \in W_i^{\star\circ}$. Compute the components of $rk_3^\star$ as given below.

$$\sigma_1^\star = B, \sigma_2^\star = C^\beta, C_1'^\star = g^{s'H_6(Z^\beta)}$$

  The other components of the challenge re-encrypted ciphertext $CT^\star = (S, C_0, C_1, C_2, T, rk_3^\star)$ can be computed as given in the construction.

Finally, $\mathcal{A}$ receives the challenge ciphertext $CT^\star$ from $\mathcal{B}$.

Phase II. $\mathcal{A}$ continues to query similar to **Phase I**, with the exception of the limitations imposed by the $\mathsf{IND\text{-}CKA_{ct}}$ game.

Guess. $\mathcal{A}$ outputs a guess $i' \in \{0,1\}$. If $i' = i$, $\mathcal{A}$ wins.

If Z is equal to $e(g,g)^{abc}$, the challenge ciphertext $CT^\star$ is accuratately derived. However, if Z is randomly chosen from $\mathbb{G}_T$, the $CT^\star$ is not dependent on i in the view of $\mathcal{A}$. Hence, if $\mathcal{A}$ has a non-negligible advantage in winning the game, $\mathcal{B}$ can solve DBDH problem with non-negligible advantage. □

Theorem 18. *The proposed ABPRE-BKS scheme ensures IND-CKA$_{tok}$ security, assuming the hardness of the DBDH problem.*

Proof. Suppose a PPT adversary $\mathcal{A}$ without having the cloud secret key *csk* breaks $\mathsf{IND\text{-}CKA_{tok}}$ security with non-negligible advantage, then a challenger $\mathcal{B}$ can solve the DBDH problem by interacting with $\mathcal{A}$ as given in $\mathsf{IND\text{-}CKA_{tok}}$ game. Given the DBDH problem instance $(\Delta, A = g^a, B = g^b, C = g^c, Z)$, $\mathcal{B}$ has to ascertain if Z is

equal to $e(g,g)^{abc}$ or Z has been randomly selected from $\mathbb{G}_T$.

Setup. Same as explained in Theorem 14.

Phase I. $\mathcal{A}$ queries for the oracles $\mathcal{O}'_{sk}, \mathcal{O}'_{rk}, \mathcal{O}''_{token}, \mathcal{O}'_{search}$ and $\mathcal{O}_{decrypt}$. Since $\mathcal{B}$ has the access of msk and tsk, it runs suitable algorithms to answer $\mathcal{A}$'s queries.

Challenge. $\mathcal{A}$ selects two equal size Boolean keyword formulas $\Gamma^{\star}_{t(0)}$ and $\Gamma^{\star}_{t(1)}$ $\big($where either $W \models \Gamma^{\star}_{t(0)} \wedge W \models \Gamma^{\star}_{t(1)}$ or $W \not\models \Gamma^{\star}_{t(0)} \wedge W \not\models \Gamma^{\star}_{t(1)}$ for all W submitted to $\mathcal{O}'_{rk}$ and for all W attached in CT, which is input to $\mathcal{O}'_{search}$ and $\mathcal{O}_{decrypt}$ in Phase I$\big)$, a challenge attribute set $S^{\star}$, and sends them to $\mathcal{B}$. Now, $\mathcal{B}$ selects a bit $i \xleftarrow{u} \{0,1\}$, outputs $(tok^{\star}, sk^{\star}_{tr}) \leftarrow \mathsf{TokenGen}(PP, \Gamma^{\star}_{t(i)}, tsk, sk_{S^{\star}})$ in the following way.

Let $\Gamma^{\star}_{t(i)} = (\mathcal{K}_{t(i)}, \rho^{\circ}_{t(i)}, \{w_{\rho^{\circ}_{t(i)}}\}_{i \in \ell_t})$. Choose $\vec{\eta} = \big(\gamma' H_{10}(\mathsf{pp}), y_2, y_3, \ldots, y_{\ell_t}\big)$, where pp is computed below and $y_2, y_3, \ldots, y_{\ell_t} \in \mathbb{Z}_p^*$. Compute $\zeta_j = \mathcal{K}^{(j)}_{t(i)} \cdot \vec{\eta}$, for all $j \in [\ell_t]$. Also, choose $\phi_1, \phi_2, \phi_3, \breve{q}_j, \breve{q}'_j \xleftarrow{u} \mathbb{Z}_p^*$, set $\vec{\eta}' = \big(\phi_3/\phi_1, y'_2, y'_3, \ldots, y'_{\ell_t}\big)$, where $y'_2, y'_3, \ldots, y'_{\ell_t} \in \mathbb{Z}_p^*$ and compute $\nu_j = \mathcal{K}^{(j)}_{t(i)} \cdot \vec{\eta}'$, for all $j \in [\ell_t]$. Compute the following token components.

$$\Pi_1 = B, \Pi_2 = C^{\beta}, \mathsf{pp} = Z^{\beta}, \Pi_{j1} = g_6^{\zeta_j} \cdot g_4^{b_1 b_2 \breve{q}_j + b_3 b_4 \breve{q}'_j},$$

$$\Pi_{j2} = X^{\nu_j} \cdot g_5^{b_1 b_2 \breve{q}_j + b_3 b_4 \breve{q}'_j}, \Pi_{j3} = H_8(\mathsf{pp} || \Gamma^{\star\circ}_{t(i)} || \mathcal{K}^{(j)}_{t(i)}) \cdot g^{b_1 b_2 \breve{q}_j + b_3 b_4 \breve{q}'_j},$$

$$U_{j1} = (g_2^{w_{\rho^{\circ}_{t(i)}(j)}} g_3)^{-\breve{q}_j b_1}, U_{j2} = (g_2^{w_{\rho^{\circ}_{t(i)}(j)}} g_3)^{-\breve{q}_j b_2},$$

$$U_{j3} = (g_2^{w_{\rho^{\circ}_{t(i)}(j)}} g_3)^{-\breve{q}'_j b_3}, U_{j4} = (g_2^{w_{\rho^{\circ}_{t(i)}(j)}} g_3)^{-\breve{q}'_j b_4}$$

Set $sk^{\star}_{tr} = (\phi_1, \phi_2)$. The other components of token
$tok^{\star} = (\Gamma^{\star\circ}_{t(i)}, \Pi_1, \Pi_2, \{\Pi_{j1}, \Pi_{j2}, \Pi_{j3}, U_{j1}, U_{j2}, U_{j3}, U_{j4}\}_{j \in [\ell_t]}, D_1, D_2, \{D_y\}_{y \in S^{\star}})$ can be computed as given in the construction. Finally, $tok^{\star}$ and $sk^{\star}_{tr}$ are sent to $\mathcal{A}$.

Phase II. $\mathcal{A}$ continues to query similar to **Phase I**, with the exception of the limitations imposed by the $\mathsf{IND\text{-}CKA}_{\mathsf{tok}}$ game.

Guess. $\mathcal{A}$ outputs a guess $i' \in \{0,1\}$. If $i' = i$, $\mathcal{A}$ wins.

If $Z = e(g,g)^{abc}$, the token $tok^{\star}$ is valid. However, if $Z \xleftarrow{u} \mathbb{G}_T$, the challenge token $tok^{\star}$ is independent of the bit i in the view of $\mathcal{A}$. Hence, if $\mathcal{A}$ has a non-negligible advantage in winning the game, $\mathcal{B}$ can solve DBDH problem with non-negligible advantage. □

Theorem 19. *Our ABPRE-BKS scheme is verifiable if the collision-resistance assumption of H_4 and H_5 holds.*

Proof. Suppose a PPT adversary $\mathcal{A}$ having the cloud secret key csk breaks the

Verifiability security with non-negligible advantage, then, by communicating with $\mathcal{A}$, a challenger $\mathcal{B}$ may identify a collision for hash functions H_4 or H_5 as given in Verifiability game.

Setup. $\mathcal{B}$ computes $(mpk, msk) \leftarrow \mathsf{PKG.Setup}(1^\lambda, U)$, $(tpk, tsk) \leftarrow \mathsf{TGC.Setup}(1^\lambda, U)$, $(cpk, csk) \leftarrow \mathsf{PCS.Setup}(1^\lambda, U)$, and sends (PP, csk) to $\mathcal{A}$, where $PP = (mkp, cpk, tpk)$.

Phase I. $\mathcal{A}$ queries the oracles $\mathcal{O}'_{sk}, \mathcal{O}''_{rk}, \mathcal{O}'''_{token}, \mathcal{O}''_{search}, \mathcal{O}_{transfom}$ and $\mathcal{O}_{decrypt}$. Since $\mathcal{B}$ has msk, csk and tsk, it runs appropriate algorithms to answer $\mathcal{A}$'s queries.

Challenge. $\mathcal{A}$ selects a message $m^\star$, a keyword set $W^\star$ and an encryption policy $\Gamma_e^\star$, and forwards them to $\mathcal{B}$. Now, $\mathcal{B}$ outputs an original ciphertext
$CT^\star \leftarrow \mathsf{Encrypt}(PP, m^\star, \Gamma_e^\star, W^\star)$, or a re-encrypted ciphertext
$CT^\star \leftarrow \mathsf{Re\text{-}Encrypt}\big(PP, csk, \mathsf{Encrypt}(PP, m^\star, \Gamma_e, W), rk^\star\big)$, where
$rk^\star \leftarrow \mathsf{Re\text{-}KeyGen}(PP, sk_S, \Gamma_e^\star, W^\star), S \models \Gamma_e$, and sends $CT^\star$ to $\mathcal{A}$.
Here $CT^\star = (ct_e^\star, ct_k^\star, \mathsf{tag}^\star, \bar{C}^\star, \varepsilon^\star)$ or $CT^\star = (S, C_0^\star, C_1^\star, C_2^\star, T^\star, rk_3^\star)$

Phase II. $\mathcal{A}$ continues to query similar to **Phase I**, with the exception of the limitations imposed by the Verifiability game.

Output. $\mathcal{A}$ outputs a keyword policy $\Gamma_t^\star$, an attribute set $S^\star$ and a transformed original ciphertext $CT_{tr}^\star = (R_1, R_2, C_0, \breve{B}, C_2, \mathsf{tag}^\star)$ or a transformed re-encrypted ciphertext $CT_{tr}^\star = (Y_1, Y_2, C_0, C_0', \breve{B}, \breve{D}, C_2, T, \mathsf{tag1}^\star)$.

If $CT_{tr}^\star$ is a transformed original ciphertext, as described in the Verifiability game, $\mathcal{B}$ possesses the tuple $(\Gamma_t^\star, S^\star, tok^\star, sk_{tr}^\star)$, where $sk_{tr}^\star = (\phi_1^\star, \phi_2^\star)$. If $\mathcal{A}$ is able to break the security game, $\mathcal{B}$ can get back a message $m \leftarrow \mathsf{Verify\text{-}and\text{-}Decrypt}(\mathsf{PP}, CT_{tr}^\star, sk_{tr}^\star)$, where $m \notin \{m^\star, \perp\}$, as follows.
(i) Compute $\Omega = R_1^{-\phi_1^\star} \cdot R_2^{\phi_2^\star}$, (ii) observe $H_5(H_4(\Omega)||C_0) = \mathsf{tag}^\star$, and (iii) obtain $m||\gamma = C_0 \oplus H_2(\Omega)$.

If $\Omega^\star$ corresponds to Ω utilized in creation of $CT^\star$, then there are two possibilities $\Omega \neq \Omega^\star$ or $\Omega = \Omega^\star$.
From (ii), we have that $H_5(H_4(\Omega)||C_0) = \mathsf{tag}^\star = H_5(H_4(\Omega^\star)||C_0^\star)$.

If $\Omega \neq \Omega^\star$, then $H_4(\Omega) \neq H_4(\Omega^\star)$; otherwise, the pair $(\Omega, \Omega^\star)$ forms a collision for H_4. Hence, $H_4(\Omega)||C_0 \neq H_4(\Omega^\star)||C_0^\star$. This shows that the pair $(H_4(\Omega)||C_0, H_4(\Omega^\star)||C_0^\star)$ forms a collision for H_5.

Suppose $\Omega = \Omega^\star$. Then, $C_0 \oplus H_2(\Omega) = (m||\gamma) \neq (m^\star||\gamma^\star) = C_0^\star \oplus H_2(\Omega^\star)$ and hence $C_0 \neq C_0^\star$. So, the pair $(H_4(\Omega)||C_0, H_4(\Omega)||C_0^\star)$ causes H_5 to collide.

$\mathcal{B}$ detects a collision for H_5 in each scenario. Due to the collision-resistance nature of hash function H_5 , it is impossible for $\mathcal{A}$ to gain a non-negligible advantage and win the verifiability game. Therefore, we prove our scheme to be verifiable.

It should be noted that if $CT_{tr}^\star$ represents a transformed re-encrypted ciphertext,

we can demonstrate the verifiability of our scheme in a similar manner. □

5.5 Performance

The following symbols and notations have been utilized within this chapter.

n	:	total number of clauses in DNF encryption policy
ς	:	size of a keyword set ascribed to a ciphertext
$m_{\mathbb{G}}$ (resp. $x_{\mathbb{G}}$)	:	one multiplication (resp. exponentiation) execution time in $\mathbb{G}$
$\|S\|$	:	number of DUs attributes
x_T	:	one exponentiation execution time on $\mathbb{G}_T$ element
$I_{\mathbb{G}}$	:	one inversion execution time on $\mathbb{G}$ element
ℓ	:	total number of attributes in an encryption policy
I_T	:	one inversion execution time on $\mathbb{G}_T$ element
$\|\mathcal{U}\|$	:	number of attributes in encryption attribute universe
t_P	:	one pairing computation execution time
ℓ_t	:	total number of keywords within a Boolean query formula Γ_t
t_H	:	one hash function calculation execution time
$\|\mathbb{G}\|$	:	size of an element of $\mathbb{G}$
$\|\mathbb{G}_T\|$	:	size of an element of $\mathbb{G}_T$
ℓ_H	:	hash function's output length
$\|msg\|$	:	message size
Dec(Or)	:	an original ciphertext decryption
Dec(Re)	:	a re-encrypted ciphertext decryption
Transformed (Or) ciphertext size	:	size of a transformed original ciphertext
Transformed (Re) ciphertext size	:	size of a transformed re-encrypted ciphertext
O	:	big-O

Table 5.2 presents a comparison of the functionality between the proposed method and the works that are most closely comparable, as referenced by [48], [37], and [25]. Table 5.2 shows that our suggested method, along with [25], is based on CP-ABE, whereas [48] and [37] are KP-ABE. Note that CP-ABE encryption is preferred over KP-ABE [48, 37] for data sharing in cloud environments due to the ability for DOs

Table 5.2: Functionality Comparison

Scheme	Enc. type	KeyGen/ Encryption policy	Attribute universe	Search query expressivity	Security	Secure against KGAs on		Non-interactive verifiability	Constant decryption cost
						Token	CT		
[48]	KP	BF	small	single keyword	IND-CCA2 IND-CKA	✓	✓	✓	×
[37]	KP	BF	small	single keyword	IND-CPA IND-CKA	×	✓	×	×
[25]	CP	BF	small	single keyword	IND-CCA2 IND-CKA	×	✓	✓	×
ABPRE-BKS	CP	DNF policy	large	BF	IND-CCA2 IND-CKA Verifiability	✓	✓	✓	✓

✓ (resp. ×): the functionality is attained (resp. not attained) by the scheme; BF: Boolean formula

Table 5.3: Comparison of computation cost

Scheme	KeyGen	Encrypt	TokenGen	Re-Encrypt	Search Results Verify	Dec(Or)	Dec(Re)
[48]	$O(\ell \cdot \lvert\mathcal{U}\rvert)x_{\mathbb{G}}$ $+O(\ell)m_{\mathbb{G}}$	$O(\lvert S\rvert)x_{\mathbb{G}} + O(1)x_T$ $+3t_H$	$O(\ell^2)x_{\mathbb{G}} + t_H$	$9t_P + 5t_H$	$2t_P + t_H$ $+m_{\mathbb{G}}$	$2t_P + 3t_H$ $+O(\lvert S\rvert)x_{\mathbb{G}}$	$2t_P + 6t_H$ $+O(\lvert S\rvert)x_{\mathbb{G}} + x_T$
[37]	$O(\ell)x_{\mathbb{G}} + O(\ell)t_H$ $+O(\ell)m_{\mathbb{G}}$	$O(\lvert S\rvert)x_{\mathbb{G}} + x_T$ $+O(\lvert S\rvert)t_H$	$O(\ell)x_{\mathbb{G}} + t_H$	$O(\ell)P + O(\ell)I_T$	–	–	$O(\ell)t_P + x_T$ $+O(\ell)I_T + t_H$
[25]	$(2\lvert S\rvert + 6)x_{\mathbb{G}}$ $+I_G$	$O(\ell)x_{\mathbb{G}} + x_T$ $+O(\ell)t_H$	$(\lvert S\rvert + 6)x_{\mathbb{G}}$	$O(\ell)x_{\mathbb{G}} + O(\ell)t_P$	$O(\ell)x_{\mathbb{G}} + O(\ell)t_P$ $+O(\ell)I_T$	$O(\ell)x_{\mathbb{G}} + O(\ell)t_P$ $+I_T + 3t_H$	$O(\ell)x_{\mathbb{G}} + O(\ell)t_P$ $+2I_T + 4t_H$
ABPRE-BKS	$(\lvert S\rvert + 2)x_{\mathbb{G}}$	$O(n+\varsigma)x_{\mathbb{G}} + 2x_T$ $+O(n+\varsigma)m_{\mathbb{G}}$ $+t_P + 5t_H$	$O(\ell_t + \lvert S\rvert)x_{\mathbb{G}}$ $+x_T + \ell_t t_H$	$O(n)t_P + 4x_{\mathbb{G}}$	$2x_T + I_T + 2t_H$	$2x_{\mathbb{G}} + 2t_H$	$3x_{\mathbb{G}} + x_T + 4t_H$

to define their own access controls. Our scheme utilises a large attribute universe framework, whereas the other schemes [25, 48, 37] employ a small attribute universe. The KP-ABE schemes [48, 37] and the CP-ABE scheme [25] use Boolean formula-based access policy for key generation and encryption, respectively. Our scheme, on the other hand, uses an explicit DNF policy for encryption. Unlike the schemes [25, 48, 37], which rely on an inefficient and less expressive single keyword-based search mechanism, our approach achievesan efficient and expressive Boolean keyword search framework. In contrast to the schemes [25, 37], our ABPRE-BKS and the scheme [48] can resist KGAs on both the ciphertext and token. Other than the scheme [48], the schemes [25, 37] including our suggested scheme offer non-interactiveverifiability. However, the approaches employed in [25, 48, 37] fail to offer constant decryption cost on the DU side. By outsourcing decryption rights to the cloud server, our scheme, in contrast, offers constant decryption cost on the DU side.

Table 5.3 presents a comparison of computation costs of our proposed scheme ABPRE-BKS with S-ABPRE-KU [48], ABDR-PRE [37], and CPAB-KSDS [25]. Our technique is most efficient in KeyGen, Dec(Or), and Dec(Re), as demonstrated in Table 5.3. Specially, our scheme requires only two exponentiations in $\mathbb{G}$ and

Table 5.4: Comparison of communication cost

Scheme	Key size	Ciphertext Size (Or)	Token Size	Transformed (Or) Ciphertext Size	Transformed(Re) Ciphertext Size
[48]	$O(\ell^2)\|\mathbb{G}\|$	$O(\|S\|)\|\mathbb{G}\| + \|\mathbb{G}_T\| + O(1)\|msg\|$	$O(\ell^2)\|\mathbb{G}\|$	$O(\|S\|)\|\mathbb{G}\| + \|\mathbb{G}_T\| + O(1)\|msg\|$	$O(\|S\|)\|\mathbb{G}\| + 2\|\mathbb{G}_T\| + O(1)\|msg\|$
[37]	$O(\ell)\|\mathbb{G}\|$	$O(\|S\|)\|\mathbb{G}\| + 2\|\mathbb{G}_T\|$	$O(\ell)\|\mathbb{G}\|$	–	$O(\|S\|)\|\mathbb{G}\| + 3\|\mathbb{G}_T\|$
[25]	$(2\|S\| + 5)\|\mathbb{G}\|$	$O(\ell)\|\mathbb{G}\| + \|\mathbb{G}_T\| + \|msg\|$	$O(\|S\|)\|\mathbb{G}\|$	$O(\ell)\|\mathbb{G}\| + \|\mathbb{G}_T\| + \|msg\|$	$O(\ell)\|\mathbb{G}\| + \|\mathbb{G}_T\| + \|msg\|$
ABPRE-BKS	$(\|S\| + 2)\|\mathbb{G}\|$	$O(n + \varsigma)\|\mathbb{G}\| + O(1)\|msg\|$	$O(\ell_t + \|S\|)\|\mathbb{G}\|$	$2\|\mathbb{G}\| + 2\|\mathbb{G}_T\| + \ell_H + O(1)\|msg\|$	$3\|\mathbb{G}\| + 3\|\mathbb{G}_T\| + \ell_H + O(1)\|msg\|$

two hash value computations to execute Dec(Or) algoritm, while S-ABPRE-KU [48] and CPAB-KSDS [25] incur linearly increasing cost, i.e., $2t_P + O(|S|)x_{\mathbb{G}} + 3t_H$ and $O(\ell)t_P + O(\ell)x_{\mathbb{G}} + O(\ell)I_T$, respectively. Also, in Dec(Re) algorithm, our scheme enjoys constant decryption cost whereas the other schemes [25, 48, 37] suffer from linear decryption cost. As $n < \ell$, compared with [25, 37], our scheme requires less pairing cost in Re-Encrypt algorithm. In our schme, the search result verification cost on DU side is constant whereas this cost depends on the number of attributes in encryption policy in CPAB-KSDS [25]. In our proposed scheme, the encryption process exhibits a longer duration when compared to the other schemes [25, 48, 37]. This is due to the fact that while other schemes [25, 48, 37] rely on less effective single keyword search framework, our scheme offers efficient Boolean keyword search mechanism. Although our scheme takes longer than the schemes [37] and [25] to generate a token, it exhibits enhanced efficiencyin key generation, re-encryption, search results verification, and decryption processes.

Table 5.4 compares the communication costs of our proposed scheme with [48, 37, 25]. Table 5.4 shows that our scheme provides smaller size of decryption key, transformed original ciphertext and transformed re-encrypted ciphertext compared to those of [25, 48, 37]. In particular, the size of the transformed original/re-encrypted ciphertext is constant in our scheme whereas these sizes increase with the number of attributes in the encryption policy or the cardinality of the attribute set in [25, 48, 37]. Even though our scheme produces larger size of token and ciphertext, it supports expressive and efficient Boolean search framework whereas [25, 48, 37] provide less efficient single keyword search framework.

5.6 Chapter Summary

In this chapter, we present a ciphertext-policy attribute-based mechanism supporting Boolean keyword search and data sharing. We provide a concrete construction of our searchable ABPRE scheme with keyword set update mechanism. Our scheme enables an outsourced decryption mechanism, resulting in a constant decryption cost on the DU side. This characteristic makes our scheme more practical. In addi-

tion, our scheme empowers a DU to independently validate the accuracy of search outcomes provided by a cloud server. We prove that our scheme is verifiable, IND-CCA2 secure at both original and re-encrypted ciphertext, IND-CKA secure on both ciphertext and token . The performance and functionality comparison demonstrate that our suggested scheme is both efficient and practical. The proposed design outperforms the existing schemes in terms of decryption cost while supporting expressive access policies. To be specific, only 2 exponentiations in $\mathbb{G}$ and 2 hash functions computations are required to complete our original ciphertext decryption process, and 3 exponentiation in $\mathbb{G}$, one exponentiations in $\mathbb{G}_T$ and 4 hash functions calculations are required to complete our re-encrypted ciphertext decryption process irrespective of the number of required attributes.

Chapter 6

Conclusions and Scope for Future Work

Cloud computing has emerged as a solution to the issue of managing and maintaining personal data with the rise in popularity of personal electronic devices. Data storage, data sharing, and data retrieval are the three main features of cloud computing. Cloud computing technology offers many advantages, but it also raises new problems, such as data privacy and data access control.

Attribute-based framework has recently emerged as a powerful cryptographic platform to achieve fine-grained access control and data confidentiality for outsourced encrypted data. There is a rich variety of cryptographic primitives build on ABE. Starting from the seminal work of Sahai and Waters [75], attribute-based mechanism has been used to construct KP-ABE [76, 29], CP-ABE [4, 85], ABS [57], ABSE [92, 81], ABSC [18], searchable ABSC [53], ABPRE [50], searchable ABPRE [80] etc. The enhanced functionality and flexibility provided by attribute-based cryptosystems are appealing for many different practical applications such as cloud-based PHR management system. Since the PHR includes the sensitive information like disease, it is important to ensure DO anonymity and keyword privacy while exchanging PHRs with DUs. How to check the precision of the search results acquired from the cloud is another challenge in ABSE framework. Providing DUs a better search experience by getting the search results in a single query using Boolean keyword serach is also very much desirable. Along with Boolean keyword-based data searching, how to allow a DU to share the encrypted data with another DU without decrypting it? Making constant decryption cost on DU side is another important aspect. All these issues motivate us to design the schemes contained in Chapter 3, Chapter 4 and Chapter 5.

The contributions of the thesis are briefly summarized below.

- In Chapter 3, we introduce an online-offline attribute-based searchable signcryption scheme with verifiable data storage and retrieval. The scheme simultaneously provides data and DO authenticity, fine-grained data access control, DO anonymity, outsourced unsigncryption, non-interactive search results verification, keyword policy search, keyword privacy, and KGAs security on both ciphertext and token. The scheme is proven to be IND-CCA2 secure in the random oracle model under the hardness assumption of q-1 and DBDH problems. Its EUF-CMA security is provided in the random oracle model assuming the hardness of q-DHE problem. Assuming q-2, DLin and DBDH problems are hard, the scheme is proven to be IND-CKA secure. Also, it is proven that the scheme is non-interactive verifiable and it maintains DO privacy.

- The main contribution of Chapter 4 is the construction of an attribute-based searchable signcryption scheme for cloud-based EMR management system, which allows EMR owners to securely store and distribute their EMRs to specific groups of healthcare professionals. Using DNF policy in signcryption algorithm, the scheme reduces the ciphertext size compared to the scheme presented in Chapter 3. Also, it offers EMR confidentiality, EO anonymity, EMR and EO authenticity, keyword policy search over encrypted EMR, search results verification by EUs, constant decryption cost on EU side, keyword privacy. In the random oracle model, the scheme's IND-CCA2 security is demonstrated under the hardness of DBDH assumption. Assuming the q-DHE problem's hardness, its EUF-CMA security is implemented in a random oracle model. The scheme's search results verifiability and preservation of EO privacy have been demonstrated. And, the IND-CKA security of the scheme has been proven under q-2, DLin and DBDH hardness assumptions.

- In Chapter 5, we propose the first Boolean searchable attribute-based proxy re-encryption scheme with data storage, data sharing, and data retrieval mechanisms simultaneously. Our scheme facilitates expressive Boolean keyword search, keyword privacy, keyword set updating, data sharing, outsourced decryption, security against KGAs on both token and ciphertext, and non-interactive verifiability. The scheme makes use of a large attribute universe framework, and as a result, the public parameter size becomes constant. We ensure IND-CCA2 security at both the original and re-encrypted ciphertexts in the random oracle model under the DBDH hardness assumption. The

IND-CKA$_{ct}$ and IND-CKA$_{tok}$ securities of the scheme have been proven under the hardness assumption of DBDH. Also, our scheme is proven to be verifiable.

Future Directions

We would like to extend our work in the following directions.

- One limitation of the proposed schemes in all the chapters is that the schemes do not consider user revocation. A DU shouldn't be allowed to access original data stored on the cloud server if it exits the system. Moreover, data storage on the cloud server should be precluded for a revoked DO. Hence, one promising future work direction is extending our schemes to support user revocation functionality.

- Another necessary functionality is traitor tracing. Some DUs may sell their secret keys for financial gain. Therefore, it is critical that EMR management systems enable the tracing of the identity of any user who sells its secret key fraudulently. Therefore, another future research approach is to design a searchable signcryption and searchable proxy re-encryption with traitor tracing.

- Our schemes in all the chapters also motivate us to solve a interesting open problem, which is to reduce the search token size. The smaller size tokens reduces communication cost on DUs' side.

- Independent token generation is an another essential feature in ABSE framework. How to enable a DU to create search token independently in Boolean searchable ABPRE framework, we left as open problem in Chapter 5.

- Furthermore, integrating blockchain technology into our schemes across all chapters to secure personal data would be an appealing addition.

Bibliography

[1] Ali, Mohammad, Sadeghi, Mohammad-Reza, Liu, Ximeng, Miao, Yinbin, and Vasilakos, Athanasios V, Verifiable online/offline multi-keyword search for cloud-assisted industrial internet of things, *Journal of Information Security and Applications*, 65:103101, 2022.

[2] Belguith, Sana, Kaaniche, Nesrine, Hammoudeh, Mohammad, and Dargahi, Tooska, Proud: Verifiable privacy-preserving outsourced attribute based signcryption supporting access policy update for cloud assisted iot applications, *Future Generation Computer Systems*, 111:899–918, 2020.

[3] Bera, Sourav, Prasad, Suryakant, Rao, Y Sreenivasa, Das, Ashok Kumar, and Park, Youngho, Designing attribute-based verifiable data storage and retrieval scheme in cloud computing environment, *Journal of Information Security and Applications*, 75:103482, 2023.

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

[5] Bethencourt, John, Sahai, Amit, and Waters, Brent, Ciphertext-policy attribute-based encryption, in 2007 IEEE symposium on security and privacy (SP'07), pages 321–334, IEEE, 2007.

[6] Boneh, Dan, Di Crescenzo, Giovanni, Ostrovsky, Rafail, and Persiano, Giuseppe, Public key encryption with keyword search, in Christian Cachin and Jan L. Camenisch, editors, Advances in Cryptology - EUROCRYPT 2004, pages 506–522, Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[7] Boneh, Dan and Franklin, Matt, Identity-based encryption from the weil pairing, in Annual international cryptology conference, pages 213–229, Springer, 2001.

[8] Boyen, Xavier and Waters, Brent, Anonymous hierarchical identity-based encryption (without random oracles), in Cynthia Dwork, editor, Advances in Cryptology - CRYPTO 2006, pages 290–307, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[9] Byun, Jin Wook, Rhee, Hyun Suk, Park, Hyun-A, and Lee, Dong Hoon, Offline keyword guessing attacks on recent keyword search schemes over encrypted data, in Willem Jonker and Milan Petković, editors, Secure Data Management, pages 75–83, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[10] Chen, Cheng, Chen, Jie, Lim, HoonWei, Zhang, Zhenfeng, and Feng, Dengguo, Combined public-key schemes: The case of abe and abs, in Tsuyoshi Takagi, Guilin Wang, Zhiguang Qin, Shaoquan Jiang, and Yong Yu, editors, Provable Security, volume 7496 of *Lecture Notes in Computer Science*, pages 53–69, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[11] Chen, Ningyu, Li, Jiguo, Zhang, Yichen, and Guo, Yuyan, Efficient cp-abe scheme with shared decryption in cloud storage, *IEEE Transactions on Computers*, 71(1):175–184, 2020.

[12] Cui, Hui, Deng, Robert H., Liu, Joseph K., and Li, Yingjiu, Attribute-based encryption with expressive and authorized keyword search, in Josef Pieprzyk and Suriadi Suriadi, editors, Information Security and Privacy, pages 106–126, Springer International Publishing, Cham, 2017.

[13] Cui, Hui, Wan, Zhiguo, Deng, Robert H., Wang, Guilin, and Li, Yingjiu, Efficient and expressive keyword search over encrypted data in cloud, *IEEE Transactions on Dependable and Secure Computing*, 15(3):409–422, 2018.

[14] Deng, Fuhu, Wang, Yali, Peng, Li, Xiong, Hu, Geng, Ji, and Qin, Zhiguang, Ciphertext-policy attribute-based signcryption with verifiable outsourced designcryption for sharing personal health records, *IEEE Access*, 6:39473–39486, 2018.

[15] Emura, Keita, Miyaji, Atsuko, and Rahman, Mohammad Shahriar, Dynamic attribute-based signcryption without random oracles, *Int. J. Appl. Cryptol.*, 2(3):199–211, 2012.

[16] Fang, Liming, Susilo, Willy, Ge, Chunpeng, and Wang, Jiandong, Chosen-ciphertext secure anonymous conditional proxy re-encryption with keyword search, *Theoretical Computer Science*, 462:39–58, 2012.

[17] Frey, Gerhard, Muller, Michael, and Ruck, H-G, The tate pairing and the discrete logarithm applied to elliptic curve cryptosystems, *IEEE Transactions on Information Theory*, 45(5):1717–1719, 1999.

[18] Gagné, Martin, Narayan, Shivaramakrishnan, and Safavi-Naini, Reihaneh, Threshold attribute-based signcryption, in JuanA. Garay and Roberto De Prisco, editors, Security and Cryptography for Networks, volume 6280 of *LNCS*, pages 154–171, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[19] Galbraith, Steven D, Harrison, Keith, and Soldera, David, Implementing the tate pairing, in International Algorithmic Number Theory Symposium, pages 324–337, Springer, 2002.

[20] Ge, A-J, Ma, C-G, and Zhang, Z-F, Attribute-based signature scheme with constant size signature in the standard model, *IET Information Security*, 6(2):47–54, 2012.

[21] Ge, Aijun, Zhang, Rui, Chen, Cheng, Ma, Chuangui, and Zhang, Zhenfeng, Threshold ciphertext policy attribute-based encryption with constant size ciphertexts, in Information Security and Privacy: 17th Australasian Conference, ACISP 2012, Wollongong, NSW, Australia, July 9-11, 2012. Proceedings 17, pages 336–349, Springer, Berlin, Heidelberg, 2012.

[22] Ge, Chunpeng, Susilo, Willy, Baek, Joonsang, Liu, Zhe, Xia, Jinyue, and Fang, Liming, A verifiable and fair attribute-based proxy re-encryption scheme for data sharing in clouds, *IEEE Transactions on Dependable and Secure Computing*, 19(5):2907–2919, 2021.

[23] Ge, Chunpeng, Susilo, Willy, Fang, Liming, Wang, Jiandong, and Shi, Yunqing, A cca-secure key-policy attribute-based proxy re-encryption in the adaptive corruption model for dropbox data sharing system, *Designs, Codes and Cryptography*, 86:2587–2603, 2018.

[24] Ge, Chunpeng, Susilo, Willy, Liu, Zhe, Baek, Joonsang, Luo, Xiapu, and Fang, Liming, Attribute-based proxy re-encryption with direct revocation mechanism for data sharing in clouds, *IEEE Transactions on Dependable and Secure Computing*, 2023.

[25] Ge, Chunpeng, Susilo, Willy, Liu, Zhe, Xia, Jinyue, Szalachowski, Pawel, and Fang, Liming, Secure keyword search and data sharing mechanism for

cloud computing, *IEEE Transactions on Dependable and Secure Computing*, 18(6):2787–2800, 2020.

[26] Ge, Chunpeng, Susilo, Willy, Liu, Zhe, Xia, Jinyue, Szalachowski, Pawel, and Fang, Liming, Secure keyword search and data sharing mechanism for cloud computing, *IEEE Transactions on Dependable and Secure Computing*, 18(6):2787–2800, 2021.

[27] Ge, Chunpeng, Susilo, Willy, Wang, Jiandong, Huang, Zhiqiu, Fang, Liming, and Ren, Yongjun, A key-policy attribute-based proxy re-encryption without random oracles, *The Computer Journal*, 59(7):970–982, 2016.

[28] Goyal, Vipul, Pandey, Omkant, Sahai, Amit, and Waters, Brent, Attribute-based encryption for fine-grained access control of encrypted data, in Proceedings of the 13th ACM conference on Computer and communications security, pages 89–98, 2006.

[29] Goyal, Vipul, Pandey, Omkant, Sahai, Amit, and Waters, Brent, Attribute-based encryption for fine-grained access control of encrypted data, in Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS '06, pages 89–98, ACM, Alexandria, Virginia, USA, 2006.

[30] Green, Matthew, Hohenberger, Susan, and Waters, Brent, Outsourcing the decryption of abe ciphertexts, in In Proceedings of the USENIX Security Symposium, 2011.

[31] Green, Matthew, Hohenberger, Susan, and Waters, Brent, Outsourcing the decryption of {ABE} ciphertexts, 20th USENIX Security Symposium (USENIX Security 11), 2011.

[32] Guo, Fuchun, Mu, Yi, Susilo, W., Wong, D.S., and Varadharajan, V., Cp-abe with constant-size keys for lightweight devices, *Information Forensics and Security, IEEE Transactions on*, 9(5):763–771, 2014.

[33] Hohenberger, Susan and Waters, Brent, Attribute-based encryption with fast decryption, in Kaoru Kurosawa and Goichiro Hanaoka, editors, Public-Key Cryptography – PKC 2013, pages 162–179, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[34] Hohenberger, Susan and Waters, Brent, Attribute-based encryption with fast decryption, International workshop on public key cryptography, pages 162–179, Springer, 2013.

[35] Hohenberger, Susan and Waters, Brent, Online/offline attribute-based encryption, in Public-Key Cryptography–PKC 2014: 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26-28, 2014. Proceedings 17, pages 293–310, Springer, 2014.

[36] Hohenberger, Susan and Waters, Brent, Public-Key Cryptography (PKC 2014), chapter Online/Offline Attribute-Based Encryption, pages 293–310, Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[37] Hong, Hanshu, Liu, Ximeng, and Sun, Zhixin, A fine-grained attribute based data retrieval with proxy re-encryption scheme for data outsourcing systems, *Mobile Networks and Applications*, pages 1–6, 2021.

[38] Krawczyk, Hugo, Cryptographic Extraction and Key Derivation: The HKDF Scheme, pages 631–648, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[39] L. Yang, K. Zheng, Q. Mei and Hanauer, D. A., Query log analysis of an electronic health record search engine, *AMIA Annu Symp Proc. 2011*, pages 915–924, 2011.

[40] Lai, J., Deng, R. H., Guan, C., and Weng, J., Attribute-based encryption with verifiable outsourced decryption, *IEEE Transactions on Information Forensics and Security*, 8(8):1343–1354, 2013.

[41] Lai, Junzuo, Deng, Robert H, Guan, Chaowen, and Weng, Jian, Attribute-based encryption with verifiable outsourced decryption, *IEEE Transactions on information forensics and security*, 8(8):1343–1354, 2013.

[42] Lai, Junzuo, Deng, Robert H, and Li, Yingjiu, Expressive cp-abe with partially hidden access structures, in Proceedings of the 7th ACM symposium on information, computer and communications security, pages 18–19, 2012.

[43] Lewko, Allison, Okamoto, Tatsuaki, Sahai, Amit, Takashima, Katsuyuki, and Waters, Brent, Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption, in Advances in Cryptology–EUROCRYPT 2010: 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, pages 62–91, Springer, 2010.

[44] Li, J., Huang, X., Li, J., Chen, X., and Xiang, Y., Securely outsourcing attribute-based encryption with checkability, *IEEE Transactions on Parallel and Distributed Systems*, 25(8):2201–2210, 2014.

[45] Li, J., Wang, Y., Zhang, Y., and Han, J., Full verifiability for outsourced decryption in attribute based encryption, *IEEE Transactions on Services Computing*, pages 1–1, 2017.

[46] Liang, Kaitai, Au, Man Ho, Susilo, Willy, Wong, Duncan S, Yang, Guomin, and Yu, Yong, An adaptively cca-secure ciphertext-policy attribute-based proxy re-encryption for cloud data sharing, in Information Security Practice and Experience: 10th International Conference, ISPEC 2014, Fuzhou, China, May 5-8, 2014. Proceedings 10, pages 448–461, Springer, 2014.

[47] Liang, Kaitai, Fang, Liming, Susilo, Willy, and Wong, Duncan S, A ciphertext-policy attribute-based proxy re-encryption with chosen-ciphertext security, in 2013 5th International Conference on Intelligent Networking and Collaborative Systems, pages 552–559, IEEE, 2013.

[48] Liang, Kaitai and Susilo, Willy, Searchable attribute-based mechanism with efficient data sharing for secure cloud storage, *IEEE Transactions on Information Forensics and Security*, 10(9):1981–1992, 2015.

[49] Liang, Kaitai and Susilo, Willy, Searchable attribute-based mechanism with efficient data sharing for secure cloud storage, *IEEE Transactions on Information Forensics and Security*, 10(9):1981–1992, 2015.

[50] Liang, Xiaohui, Cao, Zhenfu, Lin, Huang, and Shao, Jun, Attribute based proxy re-encryption with delegating capabilities, in Proceedings of the 4th international symposium on information, computer, and communications security, pages 276–286, 2009.

[51] Lin, S., Zhang, R., Ma, H., and Wang, M., Revisiting attribute-based encryption with verifiable outsourced decryption, *IEEE Transactions on Information Forensics and Security*, 10(10):2119–2130, 2015.

[52] Liu, Zhenhua and Fan, Yaqing, Provably secure searchable attribute-based authenticated encryption scheme., *Int. J. Netw. Secur.*, 21(2):177–190, 2019.

[53] Liu, Zhenhua, Liu, Yaohui, and Fan, Yaqing, Searchable attribute-based signcryption scheme for electronic personal health record, *IEEE Access*, 6:76381–76394, 2018.

[54] Lu, Zhenhua, Guo, Yuyan, Li, Jiguo, Jia, Weina, Lv, Liping, and Shen, Jie, Novel searchable attribute-based encryption for the internet of things, *Wireless Communications and Mobile Computing*, Article ID 8350006,, 2022.

[55] Luo, Song, Hu, Jianbin, and Chen, Zhong, Ciphertext policy attribute-based proxy re-encryption, in Information and Communications Security: 12th International Conference, ICICS 2010, Barcelona, Spain, December 15-17, 2010. Proceedings 12, pages 401–415, 2010.

[56] Maas, Martijn, Pairing-based cryptography, *Master's Thesis, Technische Universiteit Eindhoven*, 2004.

[57] Maji, HemantaK., Prabhakaran, Manoj, and Rosulek, Mike, Attribute-based signatures, in Aggelos Kiayias, editor, Topics in Cryptology 17CT-RSA 2011, volume 6558 of *LNCS*, pages 376–392, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[58] Mambo, Masahiro and Okamoto, Eiji, Proxy cryptosystems: Delegation of the power to decrypt ciphertexts, *IEICE transactions on fundamentals of electronics, Communications and computer sciences*, 80(1):54–63, 1997.

[59] Mao, X., Lai, J., Mei, Q., Chen, K., and Weng, J., Generic and efficient constructions of attribute-based encryption with verifiable outsourced decryption, *IEEE Transactions on Dependable and Secure Computing*, 13(5):533–546, 2016.

[60] Menezes, Alfred, Vanstone, Scott, and Okamoto, Tatsuaki, Reducing elliptic curve logarithms to logarithms in a finite field, in Proceedings of the twenty-third annual ACM symposium on Theory of computing, pages 80–89, 1991.

[61] Menezes, Alfred J, Elliptic curve public key cryptosystems, volume 234, Springer Science & Business Media, 1993.

[62] Miao, Yinbin, Deng, Robert H., Choo, Kim-Kwang Raymond, Liu, Ximeng, Ning, Jianting, and Li, Hongwei, Optimized verifiable fine-grained keyword search in dynamic multi-owner settings, *IEEE Transactions on Dependable and Secure Computing*, 18(4):1804–1820, 2021.

[63] Miao, Yinbin, Deng, Robert H., Liu, Ximeng, Choo, Kim-Kwang Raymond, Wu, Hongjun, and Li, Hongwei, Multi-authority attribute-based keyword search over encrypted cloud data, *IEEE Transactions on Dependable and Secure Computing*, 18(4):1667–1680, 2021.

[64] Miao, Yinbin, Liu, Ximeng, Choo, Kim-Kwang Raymond, Deng, Robert H., Li, Jiguo, Li, Hongwei, and Ma, Jianfeng, Privacy-preserving attribute-based keyword search in shared multi-owner setting, *IEEE Transactions on Dependable and Secure Computing*, 18(3):1080–1094, 2021.

[65] Ning, Jianting, Cao, Zhenfu, Dong, Xiaolei, Liang, Kaitai, Ma, Hui, and Wei, Lifei, Auditable σ -time outsourced attribute-based encryption for access control in cloud computing, *IEEE Transactions on Information Forensics and Security*, 13(1):94–105, 2018.

[66] Obiri, Isaac Amankona, Xia, Qi, Xia, Hu, Affum, Eric, Abla, Smahi, and Gao, Jianbin, Personal health records sharing scheme based on attribute based signcryption with data integrity verifiable, *Journal of Computer Security*, 30(2):291–324, 2022.

[67] Pandit, Tapas, Pandey, SumitKumar, and Barua, Rana, Attribute-based signcryption : Signer privacy, strong unforgeability and ind-cca2 security in adaptive-predicates attack, in ShermanS.M. Chow, JosephK. Liu, LucasC.K. Hui, and SiuMing Yiu, editors, Provable Security, volume 8782 of *Lecture Notes in Computer Science*, pages 274–290, Springer International Publishing, Switzerland, 2014.

[68] Qin, B., Deng, R. H., Liu, S., and Ma, S., Attribute-based encryption with efficient verifiable outsourced decryption, *IEEE Transactions on Information Forensics and Security*, 10(7):1384–1393, 2015.

[69] Rao, Y. Sreenivasa, Attribute-based online/offline signcryption scheme, *International Journal of Communication Systems*, 30(16):e3322, 2017, e3322 dac.3322.

[70] Rao, Y. Sreenivasa, A secure and efficient ciphertext-policy attribute-based signcryption for personal health records sharing in cloud computing, *Future Generation Computer Systems*, 67:133 – 151, 2017.

[71] Rao, Y Sreenivasa and Dutta, Ratna, Expressive bandwidth-efficient attribute based signature and signcryption in standard model, in Australasian Conference on Information Security and Privacy, pages 209–225, Springer, 2014.

[72] Rao, Y. Sreenivasa and Dutta, Ratna, Efficient attribute-based signature and signcryption realizing expressive access structures, *International Journal of Information Security*, 15(1):81–109, 2016.

[73] Rao, Y. Sreenivasa and Dutta, Ratna, Computational friendly attribute-based encryptions with short ciphertext, *Theoretical Computer Science*, 668:1 – 26, 2017.

[74] Rouselakis, Yannis and Waters, Brent, Practical constructions and new proof methods for large universe attribute-based encryption, in Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, CCS '13, pages 463–474, ACM, New York, NY, USA, 2013.

[75] Sahai, Amit and Waters, Brent, Fuzzy identity-based encryption, Annual international conference on the theory and applications of cryptographic techniques, pages 457–473, Springer, 2005.

[76] Sahai, Amit and Waters, Brent, Fuzzy identity-based encryption, in Advances in Cryptology EUROCRYPT 2005, volume 3494 of *LNCS*, pages 457–473, 2005.

[77] Sangeetha, D, Chakkaravarthy, S Sibi, Satapathy, Suresh Chandra, Vaidehi, V, and Cruz, Meenalosini Vimal, Multi keyword searchable attribute based encryption for efficient retrieval of health records in cloud, *Multimedia Tools and Applications*, 81(16):22065–22085, 2022.

[78] Shahandashti, Siamak F and Safavi-Naini, Reihaneh, Threshold attribute-based signatures and their application to anonymous credential systems, in Progress in Cryptology–AFRICACRYPT 2009: Second International Conference on Cryptology in Africa, Gammarth, Tunisia, June 21-25, 2009. Proceedings 2, pages 198–216, Springer, 2009.

[79] Shao, Jun, Cao, Zhenfu, Liang, Xiaohui, and Lin, Huang, Proxy re-encryption with keyword search, *Information Sciences*, 180(13):2576–2587, 2010.

[80] Shi, Yanfeng, Liu, Jiqiang, Han, Zhen, Zheng, Qingji, Zhang, Rui, and Qiu, Shuo, Attribute-based proxy re-encryption with keyword search, *PloS one*, 9(12):e116325, 2014.

[81] Sun, W., Yu, S., Lou, W., Hou, Y. T., and Li, H., Protecting your right: Verifiable attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud, *IEEE Transactions on Parallel and Distributed Systems*, 27(4):1187–1198, 2016.

[82] Varri, Uma Sankararao, Pasupuleti, Syam Kumar, and Kadambari, KV, Practical verifiable multi-keyword attribute-based searchable signcryption in cloud storage, *Journal of Ambient Intelligence and Humanized Computing*, 14:1–13, 2022.

[83] Wang, Shangping, Jia, Shasha, and Zhang, Yaling, Verifiable and multi-keyword searchable attribute-based encryption scheme for cloud storage, *IEEE Access*, 7:50136–50147, 2019.

[84] Wang, Xu An, Huang, Xinyi, Yang, Xiaoyuan, Liu, Longfei, and Wu, Xuguang, Further observation on proxy re-encryption with keyword search, *Journal of Systems and Software*, 85(3):643–654, 2012.

[85] Waters, Brent, Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization, in Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, Public Key Cryptography – PKC 2011, pages 53–70, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[86] Waters, Brent, Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization, in International workshop on public key cryptography, pages 53–70, Springer, 2011.

[87] Xiang, Xinyin and Zhao, Xingwen, Blockchain-assisted searchable attribute-based encryption for e-health systems, *Journal of Systems Architecture*, 124:102417, 2022.

[88] Yang, Yang, Liu, Ximeng, Deng, Robert H., and Li, Yingjiu, Lightweight sharable and traceable secure mobile health system, *IEEE Transactions on Dependable and Secure Computing*, 17(1):78–91, 2020.

[89] Yau, Wei-Chuen, Phan, Raphael C W, Heng, Swee-Huay, and Goi, Bok-Min, Proxy re-encryption with keyword search: new definitions and algorithms, in Security Technology, Disaster Recovery and Business Continuity: International Conferences, SecTech and DRBC 2010, Held as Part of the Future Generation Information Technology Conference, FGIT 2010, Jeju Island, Korea, December 13-15, 2010. Proceedings, pages 149–160, Springer, 2010.

[90] Zhang, Jindan, Wang, Baocang, and Wang, Xu An, Improved online/offline attribute based encryption and more, in Leonard Barolli, Mingwu Zhang, and Xu An Wang, editors, Advances in Internetworking, Data & Web Technologies, pages 603–610, Springer International Publishing, Cham, 2018.

[91] Zhang, Kai, Wen, Mi, Lu, Rongxing, and Chen, Kefei, Multi-client sub-linear boolean keyword searching for encrypted cloud storage with owner-enforced authorization, *IEEE Transactions on Dependable and Secure Computing*, pages 1–1, 2020.

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

Appendix A

Appendix

Proof of Lemma 5

Proof. Suppose there is a PPT Type-1 adversary $\mathscr{A}$ that breaks the IND-CCA2 security (modeled as a game $\mathsf{Game}^{\mathsf{IND\text{-}CCA2}}_{\mathsf{Type\text{-}1}}$ in Section 4.2.2) of our MediCare with non-negligible advantage in the random oracle model. Then we can build a challenger $\mathscr{C}$ that is able to solve DBDH problem with non-negligible advantage, by interacting with $\mathscr{A}$ as in $\mathsf{Game}^{\mathsf{IND\text{-}CCA2}}_{\mathsf{Type\text{-}1}}$.

$\mathscr{C}$ is given the DBDH problem instance $\langle \Sigma, g, G_1, G_2, G_3, Z \rangle$, where g is a random generator of $\mathbb{G}, G_1 := g^{\phi_1}, G_2 := g^{\phi_2}, G_3 := g^{\phi_3}$,
(unknown) $\phi_1, \phi_2, \phi_3 \xleftarrow{\text{u}} \{2, 3, \ldots, p-1\}$, and $Z \in \mathbb{G}_T$. To determine whether $Z = e(g,g)^{\phi_1\phi_2\phi_3}$ or Z is a random element of $\mathbb{G}_T$, $\mathscr{C}$ interacts with $\mathscr{A}$ as described below.

(1) $\mathscr{A}$ sends the challenge attribute $y^\star \in U_e$ to $\mathscr{C}$.

(2) $\mathscr{C}$ samples $\alpha' \xleftarrow{\text{u}} \mathbb{Z}_p^*$ and sets $g_T := e(g,g)^{\alpha'} \cdot e(G_1, G_2)$ (i.e., the system master secret $\mathcal{MK}$ is defined implicitly as g^α where $\alpha := \alpha' + \phi_1\phi_2$). Next $\mathscr{C}$ samples $z, z'_4, z'_5, z_1, z_2, \ldots, z_{10} \xleftarrow{\text{u}} \mathbb{Z}_p^*$ and defines

$$h := g^z G_2, g_3 := g^{z_3} G_1, g_4 := g^{z_4} G_1^{z'_4}, g_5 := g^{z_5} G_1^{z'_5}, g_i := g^{z_i}, i = 1, 2, 6, 7, 8, 9, 10.$$

$\mathscr{C}$ chooses seven collision-resistant hash functions $H_1 : \{0,1\}^* \to \mathbb{G}$,
$H_2 : \{0,1\}^* \to \mathbb{G}, H_3 : \mathbb{G}_T \to \mathbb{Z}_p^*, H_4 : \mathbb{G}_T \to \{0,1\}^{\ell_{H_4}}, H_5 : \{0,1\}^* \to \{0,1\}^{\ell_{H_5}}$,
$H_6 : \{0,1\}^* \to \mathbb{Z}_p^*, H_7 : \{0,1\}^* \to \mathbb{Z}_p^*$, in which $\mathscr{C}$ simulates H_1 as follows.

To answer H_1 hash queries, $\mathscr{C}$ maintains a table Tab_{H_1}. If one submits an attribute $\mathsf{at} \in U_s \cup U_e$ and $\mathsf{at} \neq y^\star$, $\mathscr{C}$ answers in the following way. If the

tuple $[\![\mathsf{at}, \upsilon_{\mathsf{at}}, H_1(\mathsf{at}) := g^{\upsilon_{\mathsf{at}}}]\!]$ exists in Tab_{H_1}, it returns $g^{\upsilon_{\mathsf{at}}}$. Otherwise, $\mathscr{C}$ picks $\upsilon_{\mathsf{at}} \xleftarrow{\text{u}} \mathbb{Z}_p^*$, returns $g^{\upsilon_{\mathsf{at}}}$ and inserts the new tuple $[\![\mathsf{at}, \upsilon_{\mathsf{at}}, H_1(\mathsf{at}) := g^{\upsilon_{\mathsf{at}}}]\!]$ into Tab_{H_1}. For $y^\star \in U_e$, it selects $\upsilon_{y^\star} \xleftarrow{\text{u}} \mathbb{Z}_p^*$, returns $g^{\upsilon_{y^\star}} G_3$ and inserts the tuple $[\![y^\star, \upsilon_{y^\star}, H_1(y^\star) := g^{\upsilon_{y^\star}} G_3]\!]$ into Tab_{H_1}.

$\mathscr{C}$ sets $\mathcal{KPK} := \langle \Sigma, g_T, g, h, \{g_i\}_{i=1}^{10}, \mathcal{M}, \mathsf{KDF}, \{H_i\}_{i=1}^{7} \rangle$, where $\mathcal{M} := \{0,1\}^{\ell_{pt}}$ is the plaintext space, and KDF is the key derivation function with output length ℓ_{pt} and keying source $\mathbb{G}_T$. Lastly, $\mathscr{C}$ selects $\beta, \gamma, \varpi_1, \varpi_2, \varpi_3, \varpi_4 \xleftarrow{\text{u}} \mathbb{Z}_p^*$, computes $h_1 := g^{\varpi_1}, h_2 := g^{\varpi_2}, h_3 := g^{\varpi_3}, h_4 := g^{\varpi_4}, h_T := e(g, g_{10})^{\gamma}, Y := h^{\beta}$ and sets $\mathcal{TPK} := \langle h_T, h_1, h_2, h_3, h_4 \rangle$,$\mathcal{TSK} := \langle \gamma, \varpi_1, \varpi_2, \varpi_3, \varpi_4 \rangle$,
$\mathcal{CPK} := Y, \mathcal{CSK} := \beta$. $\mathscr{C}$ sends the tuple $[\mathcal{PP}, \mathcal{CSK}]$ to $\mathscr{A}$,
where $\mathcal{PP} := \langle \mathcal{KPK}, \mathcal{CPK}, \mathcal{TPK} \rangle$.

(3) $\mathscr{A}$ queries signing key generation oracle $\mathcal{O}_{\mathcal{SKG}}(A_s)$, token generation oracle $\mathcal{O}_{\mathcal{TG}}(A_d, \Gamma_t)$, ciphertext generation oracle $\mathcal{O}_{\mathcal{CG}}(emr, \Gamma_s, \Gamma_e, W)$ and EMR retrieval oracle $\mathcal{O}_{\mathcal{ER}}(\mathcal{CT}, A_d, \Gamma_t)$, with the respective inputs. Then $\mathscr{C}$ answers these queries as described below.

- $\mathcal{O}_{\mathcal{SKG}}(A_s)$: $\mathscr{C}$ chooses $\tilde{r} \xleftarrow{\text{u}} \mathbb{Z}_p^*$, implicitly defines $r' := \tilde{r} - \phi_1$ and returns the signing key
  $\mathcal{SK}_{A_s} := \left\langle A_s, S := g^{\alpha'} G_2^{\tilde{r}} G_1^{-z} g^{z\tilde{r}}, S_0 := g^{\tilde{r}} G_1^{-1}, \{S_x := G_1^{-\upsilon_x} g^{\tilde{r}\upsilon_x}\}_{x \in A_s} \right\rangle$ to $\mathscr{A}$.
- $\mathcal{O}_{\mathcal{TG}}(A_d, \Gamma_t)$: $\mathscr{C}$'s response is one of the following two types.
  (i) If $y^\star \in A_d$, then $\mathscr{C}$ selects $f, f', \breve{r}_i, \breve{r}'_i, \bar{\bar{r}}, d, d', d'' \xleftarrow{\text{u}} \mathbb{Z}_p^*$, and implicitly sets $\tau_1 := \alpha d''/(dd'), \tau_2 := \alpha/d, \tau_3 := \alpha d''/d$. Next, $\mathscr{C}$ computes the EMR retrieval request token

$$token := \begin{pmatrix} \mathcal{TD}_{\Gamma_t^\circ} := \langle \Gamma_t^\circ, T_1, T_2, \{T_{i1}, T_{i2}, T_{i3}, V_{i1}, V_{i2}, V_{i3}, V_{i4}\}_{i \in [\ell_t]} \rangle \\ \mathcal{TK}_{A_d} := \langle A_d, D', D'_0, \{D'_y\}_{y \in A_d} \rangle \end{pmatrix}$$

  of the keyword policy $\Gamma_t := (\mathbf{M}_t, \rho_t^\circ, \{w_{\rho_t^\circ(i)}\}_{i \in [\ell_t]})$ and the decryption attribute set A_d as follows. $D' := g^d Y^{\bar{\bar{r}}} h^{d''}, D'_0 := g^{\bar{\bar{r}}}$

$$D'_y := \begin{cases} g^{\bar{\bar{r}}\upsilon_y}, & \text{if } y \in A_d \setminus \{y^\star\}; \\ G_3^{\bar{\bar{r}}} \cdot g^{\bar{\bar{r}}\upsilon_{y^\star}}, & \text{if } y = y^\star. \end{cases}$$

  ϑ_i (resp. $\breve{\vartheta}_i$) is the ith share of $\gamma \cdot H_3(e(T_1, Y)^{f'})$ (resp. $\tau_3/\tau_1 = d'$) with respect to the policy $(\mathbf{M}_t, \rho_t^\circ)$, and calculate the other components

of *token* as in Equation (5.4). Note that, in this case, $\mathscr{C}$ does not know $\mathcal{TDK} := \langle \tau_1, \tau_2 \rangle$. However, since $\mathscr{C}$ knows TGA's secret key $\langle \gamma, \varpi_1, \varpi_2, \varpi_3, \varpi_4 \rangle$, the above *token* is properly distributed according to Remark 10.

(ii) If $y^\star \notin A_d$, then $\mathscr{C}$ selects $\check{r} \xleftarrow{u} \mathbb{Z}_p^*$, implicitly defines $r := \check{r} - \beta^{-1}\phi_1$ and calculates the decryption key $\mathcal{DK}_{A_d} := \langle A_d, D, D_0, \{D_y\}_{y \in A_d} \rangle$ as given below.

$$D := g^{\alpha'} G_1^{-z} G_2^{\beta \check{r}} g^{\beta z \check{r}}, D_0 := G_1^{-\beta^{-1}} g^{\check{r}}, D_y := G_1^{-\beta^{-1} \upsilon_y} g^{\check{r} \upsilon_y}$$

Since $\mathscr{C}$ knows TGA's secret key $\mathcal{TSK} := \langle \gamma, \varpi_1, \varpi_2, \varpi_3, \varpi_4 \rangle$, it computes the trapdoor $\widetilde{\mathcal{TD}}_{\Gamma_t} \leftarrow \mathsf{TrapGen}(\mathcal{PP}, \mathcal{TSK}, \Gamma_t)$. Finally, $\mathscr{C}$ sends $[\mathcal{DK}_{A_d}, \widetilde{\mathcal{TD}}_{\Gamma_t}]$ to $\mathscr{A}$. In this case, both $\mathscr{A}$ and $\mathscr{C}$ can generate $token := \langle \mathcal{TD}_{\Gamma_t^\circ}, \mathcal{TK}_{A_d} \rangle$ and $\mathcal{TDK}$ by using $[\mathcal{DK}_{A_d}, \widetilde{\mathcal{TD}}_{\Gamma_t}]$.

- $\mathcal{O}_{\mathcal{CG}}(emr, \Gamma_s, \Gamma_e, W)$: $\mathscr{C}$ selects a signing attribute set A_s such that $\Gamma_s(A_s) = 1$, computes $\mathcal{SK}_{A_s} \leftarrow \mathcal{O}_{\mathcal{SKG}}(A_s)$ and returns the ciphertext $\mathcal{CT} \leftarrow \mathsf{Signcrypt}(\mathcal{PP}, \mathcal{SK}_{A_s}, \Gamma_s, \Gamma_e, W, emr)$.

- $\mathcal{O}_{\mathcal{ER}}(\mathcal{CT}, A_d, \Gamma_t)$: Firstly, $\mathscr{C}$ performs the EMR storage phase (given in Figure 4.4). If the output is $\perp$, then $\mathscr{C}$'s response is $\perp$ as well. If the output is $\mathcal{CT}_u := \langle \Delta_e, \Delta_k, \mathsf{tag2}, E_0, \eta, \breve{E}, \mathsf{tag} \rangle$, then it carries out the following steps. Note that $\Delta_e := \langle \Gamma_e, ct, E, \{E_{i1}, E_{i2}\}_{i \in [m]} \rangle$. First, it calculates $token \leftarrow \mathcal{O}_{\mathcal{TG}}(A_d, \Gamma_t)$. If $\mathsf{Search}(\mathcal{PP}, \mathcal{CT}_u, token, \mathcal{CSK}) \rightarrow \perp$ or $\Gamma_e(A_d) = 0$, it returns $\perp$. Otherwise, $\mathscr{C}$'s response can be one of the two types given subsequently.

(i) In case $y^\star \in A_d$, $\mathscr{C}$ does not have the knowledge of the secret transformation decryption key $\mathcal{TDK}$ according to the simulation of $\mathcal{O}_{\mathcal{TG}}(A_d, \Gamma_t)$ and hence it proceeds in the following way. It calculates

$$\Lambda := e(\breve{E}, g^{\alpha'}) \cdot e\Big(E_0(\breve{E})^{-(\xi z_3 + \eta z_4 + z_5)}, G_2^{(\xi + z_4' \eta + z_5')^{-1}}\Big)$$

and $\delta := H_3(\Lambda)$. Note that $\xi + z_4'\eta + z_5' = 0$ happens with probability at most $1/p$ and hence $\xi + z_4'\eta + z_5' \neq 0$ since p is very large prime number. Next, $\mathscr{C}$ checks whether $H_4(g_T^{1/\delta}) \stackrel{?}{=} \mathsf{tag}$ and $H_5(\delta || ct) \stackrel{?}{=} \mathsf{tag2}$. If any one of these is not true, it returns $\perp$. Otherwise, it returns $emr = ct \oplus \mathsf{KDF}(\Lambda)$ to $\mathscr{A}$.

(ii) In case $y^\star \notin A_d$, $\mathscr{C}$ knows $\mathcal{TDK}$. Hence, it computes the transformed

ciphertext $\mathcal{CT}_{tr} \leftarrow \mathsf{Transform}(\mathcal{PP}, \mathcal{CT}_u, token, \mathcal{CSK})$ and returns to $\mathscr{A}$ the output of the algorithm $\mathsf{Verify\text{-}Retrieve}(\mathcal{PP}, \mathcal{CT}_{tr}, \mathcal{TDK})$.

When $\mathscr{A}$ decides that this query phase is completed, it outputs two messages $emr_0^\star, emr_1^\star \in \mathcal{M}$, an encryption policy $\Gamma_e^\star$, a signing policy $\Gamma_s^\star$ and a keyword set $W^\star$.

(4) Let $\Gamma_s^\star := (\mathbf{M}_s^\star, \rho_s^\star)$, where $\mathbf{M}_s^\star$ is an $\ell_s^\star \times n_s^\star$ matrix. First $\mathscr{C}$ selects a signing attribute set A_s such that $\Gamma_s^\star(A_s) = 1$, calculates $\vec{a} := (a_1, a_2, \ldots, a_{\ell_s^\star}) \leftarrow \mathsf{Reconstruct}(\mathbf{M}_s^\star, \rho_s^\star, A_s)$ satisfying $\sum_{i\in[\ell_s^\star]} a_i \cdot \vec{M}_s^{\star(i)} = \vec{1}_{n_s^\star}$ and $a_i = 0$ for all $i \in \{i|\rho_s^\star(i) \notin A_s\}$, samples $(b_1, b_2, \ldots, b_{\ell_s^\star}) \xleftarrow{\mathrm{u}} \big\{(b_1, b_2, \ldots, b_{\ell_s^\star}) \in \mathbb{Z}_p^{\ell_s^\star} | \sum_{i\in[\ell_s^\star]} b_i \cdot \vec{M}_s^{\star(i)} = \vec{0}_{n_s^\star}\big\}$. Next, it picks $i \xleftarrow{\mathrm{u}} \{0, 1\}$ and computes the challenge ciphertext $\mathcal{CT}^\star$ of the message $emr_i^\star$ for encryption policy $\Gamma_e^\star \wedge y^\star := (B_1^\star \cup \{y^\star\}) \vee (B_2^\star \cup \{y^\star\}) \vee \cdots \vee (B_m^\star \cup \{y^\star\})$, signing policy $\Gamma_s^\star := (\mathbf{M}_s^\star, \rho_s^\star)$, and keyword set $W^\star := \{[\mathcal{W} : w^\star]\}$, in the following way.

(a) Implicitly define $\theta := \phi_3$,
compute $\delta := H_3(Z \cdot e(g, G_3)^{\alpha'}), \mathsf{key} := \mathsf{KDF}(Z \cdot e(g, G_3)^{\alpha'}), k_T := e(G_3, g_{10})^\gamma$,
$ct := emr_i^\star \oplus \mathsf{key}, \mathsf{tag2}^\star := H_5(\delta||ct)$,

(b) pick $\delta', \delta'', o_2 \xleftarrow{\mathrm{u}} \mathbb{Z}_p^*$,
$\sigma' := g^{\delta'}, \sigma'' := h^{\delta''}, \mathsf{tag1} := H_4(g_T^{1/\delta} \cdot e(\sigma', Y)^{\delta''})$,
$\mathcal{SK}_{A_s} := \big\langle A_s, S, S_0, \{S_x\}_{x\in A_s}\big\rangle \leftarrow \mathcal{O}_{\mathcal{SKG}}(A_s)$,
$\sigma := S^{1/\delta} G_3^{o_1 z_1 + z_2} \prod_{i\in[\ell_s^\star]} \big(S_{\rho_s^\star(i)}^{a_i/\delta} \cdot H_1(\rho_s^\star(i))^{o_2 b_i}\big)$,
where $o_1 := H_6(ct||\mathsf{tag1}||\Gamma_e^\star \wedge y^\star||\Gamma_s^\star||W^{\star\circ})$,
$\sigma_i := S_0^{a_i/\delta} g^{o_2 b_i}$, for each $i \in [\ell_s^\star]$,
The signature components $\Delta_s^\star := \langle \Gamma_s^\star, \sigma', \sigma'', \mathsf{tag1}, \sigma, \{\sigma_i\}_{i\in[\ell_s^\star]}\rangle$.

(c) calculate $E := G_3^{H_3\left(e(\sigma', Y)^{\delta''}\right)}$,
pick $\check{\theta}_i \xleftarrow{\mathrm{u}} \mathbb{Z}_p^*$ and implicitly define $\theta_i := -\phi_2 + \check{\theta}_i$, for each $i \in [m]$,
compute $E_{i1} := G_2^{-1} g^{\check{\theta}_i}, E_{i2} := G_3^z G_3^{\theta\check{\theta}_i} G_2^{-\upsilon_{y^\star}} g^{\upsilon_{y^\star}\check{\theta}_i} \prod_{y\in B_i^\star} \big(G_2^{-\upsilon_y} g^{\upsilon_y \check{\theta}_i}\big)$,
note that E_{i2} is corresponding to $B_i^\star \cup \{y^\star\}$ in the encryption policy $\Gamma_e^\star \wedge y^\star$,
The encryption components $\Delta_e^\star := \langle \Gamma_e^\star \wedge y^\star, ct, E, \{E_{i1}, E_{i2}\}_{i\in[m]}\rangle$.

(d) $t_\mathcal{W}, \pi_{\mathcal{W}1}, \pi_{\mathcal{W}2} \xleftarrow{\mathrm{u}} \mathbb{Z}_p^*$, for each $[\mathcal{W} : w^\star] \in W^\star$, where $W^{\star\circ} := \{\mathcal{W}\}$,
$K_{\mathcal{W}1} := h_1^{t_\mathcal{W} - \pi_{\mathcal{W}1}}, K_{\mathcal{W}2} := h_2^{\pi_{\mathcal{W}1}}, K_{\mathcal{W}3} := h_3^{t_\mathcal{W} - \pi_{\mathcal{W}2}}, K_{\mathcal{W}4} := h_4^{\pi_{\mathcal{W}2}}$,
$L_{\mathcal{W}1} := (g_6^{w^\star} g_7)^{t_\mathcal{W}} G_3^{-z_8}, L_{\mathcal{W}2} := (g_6^{w^\star} g_7)^{t_\mathcal{W}} G_3^{-z_9}$,
The keyword components,
$\Delta_k^\star := \langle W^{\star\circ}, k_T, \{K_{\mathcal{W}1}, K_{\mathcal{W}2}, K_{\mathcal{W}3}, K_{\mathcal{W}4}, L_{\mathcal{W}1}, L_{\mathcal{W}2}\}_{\mathcal{W}\in W^{\star\circ}}\rangle$.

(e) choose $\eta^\star = (-\xi - z_5')/z_4'$, where $\xi := H_7(\Delta_s^\star||\Delta_e^\star||\Delta_k^\star||\mathsf{tag2}^\star)$,
calculate $E_0^\star := G_3^{z_3\xi + z_4\eta^\star + z_5}$,
The challenge ciphertext is $\mathcal{CT}^\star := \langle \Delta_s^\star, \Delta_e^\star, \Delta_k^\star, \mathsf{tag2}^\star, E_0^\star, \eta^\star \rangle$.

Lastly, $\mathscr{C}$ sends the challenge ciphertext $\mathcal{CT}^\star$ to $\mathscr{A}$.

(5) Now, $\mathscr{A}$ issues a second series of additional queries like step (3), with the obvious restriction that it cannot query EMR retrieval oracle with the input $(\mathcal{CT}^\star, A_d, \Gamma_t)$ satisfying $y^\star \in A_d$ and $\Gamma_t(W^\star) = 1$, here $W^\star$ is the keyword set of $\mathcal{CT}^\star$. Once this query phase is over, $\mathscr{A}$ outputs a guess i' of i.

$\mathscr{A}$ wins the game if $i' = i$. Therefore, if $\mathscr{A}$ wins, $\mathscr{C}$ will claim that $Z = e(g,g)^{\phi_1\phi_2\phi_3}$; otherwise, $\mathscr{C}$ claims that Z is a random element of $\mathbb{G}_T$.

When $Z = e(g,g)^{\phi_1\phi_2\phi_3}$, it can be seen that the challenge ciphertext $\mathcal{CT}^\star$ is a properly simulated ciphertext as in original construction. Hence, $\mathscr{C}$ simulates the game correctly. Note that if Z is randomly chosen from $\mathbb{G}_T$, then $\mathcal{CT}^\star$ is independent of i in $\mathscr{A}$'s view. In this case, $\mathscr{A}$'s guess is random and its advantage is 0. Thus, if the advantage of $\mathscr{A}$ in the game $\mathsf{Game}^{\mathsf{IND\text{-}CCA2}}_{\mathsf{Type\text{-}1}}$ is non-negligible, then $\mathscr{C}$ can solve the DBDH problem with non-negligible advantage. □

Proof of Lemma 6

Proof. Suppose there is a PPT Type-2 adversary $\mathscr{A}$ that breaks the IND-CCA2 security (modeled as a game $\mathsf{Game}^{\mathsf{IND\text{-}CCA2}}_{\mathsf{Type\text{-}2}}$ in Section 4.2.2) of our MediCare with non-negligible advantage. Then we can build a challenger $\mathscr{C}$ that is able to solve DBDH problem with non-negligible advantage, by interacting with $\mathscr{A}$ as in $\mathsf{Game}^{\mathsf{IND\text{-}CCA2}}_{\mathsf{Type\text{-}2}}$. $\mathscr{C}$ is given the DBDH problem instance $\langle \Sigma, g, G_1, G_2, G_3, Z \rangle$, where $G_1 := g^{\phi_1}$, $G_2 := g^{\phi_2}, G_3 := g^{\phi_3}$ (note that ϕ_1, ϕ_2, ϕ_3 are unknown to $\mathscr{C}$). To determine whether $Z = e(g,g)^{\phi_1\phi_2\phi_3}$ or Z is a random element of $\mathbb{G}_T$, $\mathscr{C}$ interacts with $\mathscr{A}$ as described below.

(1) $\mathscr{C}$ picks $\alpha, z \xleftarrow{\text{u}} \mathbb{Z}_p^*$ and sets $g_T := e(g,g)^\alpha, h := g^z$. Next, it chooses $g_1, g_2, \ldots, g_{10} \xleftarrow{\text{u}} \mathbb{G}$, and seven collision-resistant hash functions $\{H_i\}_{i=1}^7$ (as described in the construction). Now, $\mathscr{C}$ sets $\mathcal{KPK} := \langle \Sigma, g_T, g, h, \{g_i\}_{i=1}^{10}, \mathcal{M}, \mathsf{KDF}, \{H_i\}_{i=1}^7 \rangle$, where $\mathcal{M} := \{0,1\}^{\ell_{pt}}$ is the message space and KDF is the key derivation function, and $\mathcal{MK} := g^\alpha$. Next, $\mathscr{C}$ selects $\gamma, \varpi_1, \varpi_2, \varpi_3, \varpi_4 \xleftarrow{\text{u}} \mathbb{Z}_p^*$, computes $h_1 := g^{\varpi_1}, h_2 := g^{\varpi_2}, h_3 := g^{\varpi_3}, h_4 := g^{\varpi_4}, h_T := e(g, g_{10})^\gamma, Y := G_1^z$ and sets $\mathcal{TPK} := \langle h_T, h_1, h_2, h_3, h_4 \rangle, \mathcal{TSK} := \langle \gamma, \varpi_1, \varpi_2, \varpi_3, \varpi_4 \rangle$, $\mathcal{CPK} := Y$, and implicitly sets $\mathcal{CSK} := \phi_1$. $\mathscr{C}$ sends the tuple $\mathcal{PP} := \langle \mathcal{KPK}, \mathcal{CPK}, \mathcal{TPK} \rangle$ to $\mathscr{A}$.

(2) $\mathscr{A}$ queries signing key generation oracle $\mathcal{O}_{\mathcal{SKG}}(A_s)$, token generation oracle $\mathcal{O}'_{\mathcal{TG}}(A_d, \Gamma_t)$, ciphertext generation oracle $\mathcal{O}_{\mathcal{CG}}(emr, \Gamma_s, \Gamma_e, W)$, EMR retrieval oracle $\mathcal{O}_{\mathcal{ER}}(\mathcal{CT}, A_d, \Gamma_t)$. Since $\mathscr{C}$ knows the system master secret $\mathcal{MK}$ and TGA secret key $\mathcal{TSK}$, it can answer the $\mathscr{A}$'s queries by running suitable algorithms of MediCare. Once this query phase is over, $\mathscr{A}$ sends to $\mathscr{C}$ two messages $emr_0^\star, emr_1^\star \in \mathcal{M}$, an encryption policy $\Gamma_e^\star$, a signing policy $\Gamma_s^\star$ and a keyword set $W^\star$.

(3) Let $\Gamma_s^\star := (\mathbf{M}_s^\star, \rho_s^\star)$ and $\Gamma_e^\star := B_1 \vee B_2 \vee \cdots \vee B_m$, where $\mathbf{M}_s^\star$ is an $\ell_s^\star \times n_s^\star$ matrix. $\mathscr{C}$ formulates a signing attribute set A_s such that $\Gamma_s^\star(A_s) = 1$, calculates $\vec{a} := (a_1, a_2, \ldots, a_{\ell_s^\star}) \leftarrow \mathsf{Reconstruct}(\mathbf{M}_s^\star, \rho_s^\star, A_s)$ satisfying $\sum_{i \in [\ell_s^\star]} a_i \cdot \vec{M}_s^{\star(i)} = \vec{1}_{n_s^\star}$ and $a_i = 0$ for all $i \in \{i | \rho_s^\star(i) \notin A_s\}$, picks $(b_1, b_2, \ldots, b_{\ell_s^\star}) \xleftarrow{\mathrm{u}} \{(b_1, b_2, \ldots, b_{\ell_s^\star}) \in \mathbb{Z}_p^{\ell_s^\star} | \sum_{i \in [\ell_s^\star]} b_i \cdot \vec{M}_s^{\star(i)} = \vec{0}_{n_s^\star}\}$. Next, $\mathscr{C}$ samples $i \xleftarrow{\mathrm{u}} \{0, 1\}$ and computes the challenge ciphertext $\mathcal{CT}^\star$ of the message $emr_i^\star$ for the signing policy $\Gamma_s^\star$, encryption policy $\Gamma_e^\star$ and keyword set $W^\star := \{[\mathcal{W} : w]\}$, in the following way.
Choose $\theta, o_2, \theta_i, t_{\mathcal{W}}, \pi_{\mathcal{W}1}, \pi_{\mathcal{W}2}, \eta \xleftarrow{\mathrm{u}} \mathbb{Z}_p^*$, for each $i \in [m]$ and $[\mathcal{W} : w] \in W$, set $\sigma' := G_2, \sigma'' := G_3^z, E := g^\theta H_3(Z^z)$, and the other components of $\mathcal{CT}^\star$ can be computed as in the construction of MediCare. Lastly, $\mathscr{C}$ sends $\mathcal{CT}^\star$ to $\mathscr{A}$.

(4) Now, $\mathscr{A}$ issues a second series of additional queries as in step (2), with the restriction that it cannot query EMR retrieval oracle with the input $(\mathcal{CT}^\star, A_d, \Gamma_t)$ satisfying $\Gamma_e^\star(A_d) = 1 \wedge \Gamma_t(W^\star) = 1$. When this query phase is over, $\mathscr{A}$ announces a guess i' of i.

$\mathscr{A}$ wins the game if $i' = i$. Therefore, if $\mathscr{A}$ wins, $\mathscr{C}$ will claim that $Z = e(g, g)^{\phi_1 \phi_2 \phi_3}$; otherwise, $\mathscr{C}$ claims that Z is a random element of $\mathbb{G}_T$.

When $Z = e(g, g)^{\phi_1 \phi_2 \phi_3}$, it can be seen that the challenge ciphertext $\mathcal{CT}^\star$ is a properly simulated ciphertext as in original construction of MediCare. Hence, $\mathscr{C}$ simulates the game $\mathsf{Game}_{\mathsf{Type\text{-}2}}^{\mathsf{IND\text{-}CCA2}}$ correctly. If Z is randomly chosen from $\mathbb{G}_T$, then $\mathcal{CT}^\star$ is independent of i in $\mathscr{A}$'s view; resulting in $\mathscr{A}$'s guess is random and its advantage is 0. Thus, if the advantage of $\mathscr{A}$ in $\mathsf{Game}_{\mathsf{Type\text{-}2}}^{\mathsf{IND\text{-}CCA2}}$ is non-negligible, then $\mathscr{C}$ can solve the DBDH problem with non-negligible advantage. □

Proof of Theorem 7

Proof. Suppose there exists a PPT adversary $\mathscr{A}$ that can break the EUF-CMA security (modeled as a game $\mathsf{Game}_{\mathscr{A}}^{\mathsf{EUF\text{-}CMA}}$ in Section 4.2.2) of our MediCare with

non-negligible advantage in the random oracle model. Then we can build a challenger $\mathscr{C}$ that is able to solve q-DHE problem with non-negligible advantage, by interacting with $\mathscr{A}$. Given the q-DHE problem instance $\langle \Sigma, g, \{g^{\phi^i}\}_{i\in[2q],i\neq q+1}\rangle$, the task of $\mathscr{C}$ is to compute $g^{\phi^{q+1}}$. We show below how this can be done.

(1) $\mathscr{A}$ sends the challenge signing policy $\Gamma_s^\star := (\mathbf{M}_s^\star, \rho_s^\star)$ to $\mathscr{C}$, where $\mathbf{M}_s^\star$ is an $\ell_s^\star \times n_s^\star$ matrix with $n_s^\star \leq q$. Let $\vec{M}_s^{\star(i)} := (M_{s1}^{\star(i)}, M_{s2}^{\star(i)}, \ldots, M_{sn_s^\star}^{\star(i)})$ be the ith row of $\mathbf{M}_s^\star$.

(2) $\mathscr{C}$ chooses $\alpha' \xleftarrow{\text{u}} \mathbb{Z}_p^*$, and sets $g_T := e(g,g)^{\alpha'} e(g^\phi, g^{\phi^q})$ by implicitly defining $\alpha := \alpha' + \phi^{q+1}$ and $h := g^\phi$. Next, $\mathscr{C}$ picks $z_1, z_2, \ldots, z_{10} \xleftarrow{\text{u}} \mathbb{Z}_p^*$ and defines $g_i := g^{z_i}$ for $i = 1, 2, \ldots, 10$. $\mathscr{C}$ selects seven collision-resistant hash functions $\{H_i\}_{i=1}^7$ (as mentioned in the construction), and simulates H_1 and H_6 as explained below.

H_1 Hash Queries: To answer H_1 hash queries, $\mathscr{C}$ maintains a table Tab_{H_1}. If one submits an attribute $\mathsf{at} \in U_e \cup U_s$, $\mathscr{C}$ answers in the following way. If the tuple $[\![\mathsf{at}, \upsilon_{\mathsf{at}}, H_1(\mathsf{at})]\!]$ exists in Tab_{H_1}, $\mathscr{C}$ returns $H_1(\mathsf{at})$. Otherwise, $\mathscr{C}$ picks $\upsilon_{\mathsf{at}} \xleftarrow{\text{u}} \mathbb{Z}_p^*$, returns

$$H_1(\mathsf{at}) := \begin{cases} g^{\upsilon_{\mathsf{at}}} \prod_{j\in[n_s^\star]} \left(g^{\phi^j}\right)^{-M_{sj}^{\star(i)}}, & \text{if } \rho_s^\star(i) = \mathsf{at} \text{ for some row } i \text{ of } M_s^\star, \\ g^{\upsilon_{\mathsf{at}}}, \text{ otherwise} \end{cases}$$

and inserts the new tuple $[\![\mathsf{at}, \upsilon_{\mathsf{at}}, H_1(\mathsf{at})]\!]$ into Tab_{H_1}.

H_6 Hash Queries: To answer H_6 hash queries, $\mathscr{C}$ maintains a table Tab_{H_6}. These queries are of two types. (i) Queries being submitted by $\mathscr{A}$. When $\mathscr{A}$ submits the input $(ct, \mathsf{tag1}, \Gamma_e, \Gamma_s, W^\circ)$, $\mathscr{C}$ responds as follows. If the tuple $[\![(ct, \mathsf{tag1}, \Gamma_e, \Gamma_s, W^\circ), \check{o}_1]\!]$ exists in Tab_{H_6}, $\mathscr{C}$ returns $\check{o}_1$ as $H_6(ct||\mathsf{tag1}||\Gamma_e||\Gamma_s||W^\circ) := \check{o}_1$. Else, $\mathscr{C}$ selects $\check{o}_1 \xleftarrow{\text{u}} \mathbb{Z}_p^*$, returns $\check{o}_1$ and inserts the new tuple $[\![(ct, \mathsf{tag1}, \Gamma_e, \Gamma_s, W^\circ), \check{o}_1]\!]$ into Tab_{H_6}. (ii) Queries being conducted by $\mathscr{C}$ during ciphertext generation oracle simulation (which will be discussed in ciphertext generation oracle execution given below).

Next, $\mathscr{C}$ sets $\mathcal{KPK} := \langle \Sigma, g_T, g, h, \{g_i\}_{i=1}^{10}, \mathcal{M}, \mathsf{KDF}, \{H_i\}_{i=1}^7\rangle$, where $\mathcal{M} := \{0,1\}^{\ell_{pt}}$ is the plaintext space, and KDF is the key derivation function with output length ℓ_{pt} and keying source $\mathbb{G}_T$. Lastly, $\mathscr{C}$ selects $\beta, \gamma, \varpi_1, \varpi_2, \varpi_3, \varpi_4 \xleftarrow{\text{u}} \mathbb{Z}_p^*$, computes $h_1 := g^{\varpi_1}, h_2 := g^{\varpi_2}, h_3 := g^{\varpi_3}$, $h_4 := g^{\varpi_4}, h_T := e(g, g_{10})^\gamma, Y := h^\beta$ and sets $\mathcal{TPK} := \langle h_T, h_1, h_2, h_3, h_4\rangle$, $\mathcal{TSK} := \langle \gamma, \varpi_1, \varpi_2, \varpi_3, \varpi_4\rangle, \mathcal{CPK} := Y, \mathcal{CSK} := \beta$. $\mathscr{C}$ sends the tuple $[\mathcal{PP}, \mathcal{CSK}]$ to $\mathscr{A}$, where $\mathcal{PP} := \langle \mathcal{KPK}, \mathcal{CPK}, \mathcal{TPK}\rangle$.

(3) Now, $\mathscr{A}$ queries signing key generation oracle $\mathcal{O}'_{\mathcal{SKG}}(A_s)$, token generation oracle $\mathcal{O}'_{\mathcal{TG}}(A_d, \Gamma_t)$, ciphertext generation oracle $\mathcal{O}_{\mathcal{CG}}(emr, \Gamma_s, \Gamma_e, W)$ and EMR retrieval oracle $\mathcal{O}_{\mathcal{ER}}(\mathcal{CT}, A_d, \Gamma_t)$. Then $\mathscr{C}$ responds to these queries as explained below.

- $\mathcal{O}'_{\mathcal{SKG}}(A_s)$: $\mathscr{A}$ submits a signing attribute set A_s such that $\Gamma^\star_s(A_s) = 0$. Then, $\mathscr{C}$ calculates $\vec{\varepsilon} := (\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_{n^\star_s}) \in \mathbb{Z}_p^{n^\star_s}$ such that $\varepsilon_1 = -1$ and $\vec{\varepsilon} \cdot \vec{M}^{\star(i)}_s = 0$, $\forall i \in \{i | \rho^\star_s(i) \in A_s\}$. Now, $\mathscr{C}$ picks $r'_0 \xleftarrow{\text{u}} \mathbb{Z}^*_p$, implicitly defines $r' := r'_0 + \sum_{\iota \in [n^\star_s]} \varepsilon_\iota \phi^{q-\iota+1}$ and returns the signing key $\mathcal{SK}_{A_s} := \langle A_s, S, S_0, \{S_x\}_{x \in A_s} \rangle$, where

$$S := g^{\alpha'} h^{r'_0} \prod_{\iota=2}^{n^\star_s} \left(g^{\phi^{q-\iota+2}}\right)^{\varepsilon_\iota}, \; S_0 := g^{r'_0} \prod_{\iota \in [n^\star_s]} \left(g^{\phi^{q-\iota+1}}\right)^{\varepsilon_\iota},$$

$$S_x := \begin{cases} S_0^{\upsilon_x} \prod_{j \in [n^\star_s]} \left(g^{\phi^j}\right)^{-r'_0 M^{\star(i)}_{sj}} \prod_{(\iota,j) \in [n^\star_s, n^\star_s], \iota \neq j} \left(g^{\phi^{q-\iota+1+j}}\right)^{-\varepsilon_\iota M^{\star(i)}_{sj}}, \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{if } \rho^\star_s(i) = x, \\ S_0^{\upsilon_x}, \text{ otherwise.} \end{cases}$$

- $\mathcal{O}'_{\mathcal{TG}}(A_d, \Gamma_t)$: $\mathscr{A}$ submits a decryption attribute set A_d and a keyword policy Γ_t. $\mathscr{C}$ picks $r_0 \xleftarrow{\text{u}} \mathbb{Z}^*_p$, implicitly sets $r := r_0 - \beta^{-1}\phi^q$ and calculates the decryption key $\mathcal{DK}_{A_d} := \langle A_d, D, D_0, \{D_y\}_{y \in A_d} \rangle$, where

$$D := g^{\alpha'} h^{\beta r_0}, D_0 := g^{r_0} \left(g^{\phi^q}\right)^{-\beta^{-1}}, D_y := g^{\upsilon_y r_0} \left(g^{\phi^q}\right)^{-\beta^{-1}\upsilon_y}$$

  Since $\mathscr{C}$ knows TGA's secret key $\mathcal{TSK} := \langle \gamma, \varpi_1, \varpi_2, \varpi_3, \varpi_4 \rangle$, it computes the trapdoor $\widetilde{\mathcal{TD}_{\Gamma_t}} \leftarrow \mathsf{TrapGen}(\mathcal{PP}, \mathcal{TSK}, \Gamma_t)$. Finally, $\mathscr{C}$ sends $[\mathcal{DK}_{A_d}, \widetilde{\mathcal{TD}_{\Gamma_t}}]$ to $\mathscr{A}$.

- $\mathcal{O}_{\mathcal{CG}}(emr, \Gamma_s, \Gamma_e, W)$: Let $\Gamma_s := (\mathbf{M}_s, \rho_s)$, $\Gamma_e := B_1 \vee B_2 \vee \cdots \vee B_m$, where $\mathbf{M}_s$ is a matrix of size $\ell_s \times n_s$, and $W := \{[\mathcal{W} : w]\}$. $\mathscr{C}$ chooses a signing attribute set A_s such that $\Gamma_s(A_s) = 1$, calculates $\vec{a} := (a_1, a_2, \ldots, a_{\ell_s}) \leftarrow \mathsf{Reconstruct}(\mathbf{M}_s, \rho_s, A_s)$ satisfying $\sum_{i \in [\ell_s]} a_i \cdot \vec{M}^{(i)}_s = \vec{1}_{n_s}$ and $a_i = 0$ for all $i \in \{i | \rho_s(i) \notin A_s\}$, samples $(b_1, b_2, \ldots, b_{\ell_s}) \xleftarrow{\text{u}} \left\{(b_1, b_2, \ldots, b_{\ell_s}) \in \mathbb{Z}_p^{\ell_s} | \sum_{i \in [\ell_s]} b_i \cdot \vec{M}^{(i)}_s = \vec{0}_{n_s}\right\}$. To generate a ciphertext $\mathcal{CT}$ of the message emr for the signing policy Γ_s, encryption policy Γ_e and keyword set W, $\mathscr{C}$ picks $\theta, \delta', \delta'', o_2, r', \check{o}_1, \theta_i, t_{\mathcal{W}}, \pi_{\mathcal{W}1}, \pi_{\mathcal{W}2}, \eta \xleftarrow{\text{u}} \mathbb{Z}^*_p$, for each $i \in [m]$ and $[\mathcal{W} : w] \in W$, and computes $\delta := H_3(g_T^\theta), \mathsf{key} := \mathsf{KDF}(g_T^\theta)$,

$ct := emr \oplus \mathsf{key}, \sigma' := g^{\delta'}, \mathsf{tag1} := H_4(g_T^{1/\delta} \cdot e(\sigma', Y)^{\delta''})$. Next, it implicitly defines $H_6(ct||\mathsf{tag1}||\Gamma_e||\Gamma_s||W^{\circ}) := \check{o}_1 - (\theta z_1)^{-1}\phi^{q+1}\delta^{-1}$ (this is of type (ii) query mentioned above) and sets

$$\sigma := g^{\alpha'/\delta} h^{r'/\delta} g^{z_1\theta\check{o}_1} g^{z_2\theta} \prod_{i\in[\ell_s]} H_1(\rho_s(i))^{\frac{r'a_i}{\delta}+o_2b_i}, \quad \sigma_i := (g^{r'})^{a_i/\delta} g^{o_2b_i}$$

The other components of $\mathcal{CT}$ are computes as in the construction of Medi-Care. Lastly, $\mathscr{C}$ sends this $\mathcal{CT}$ to $\mathscr{A}$.

- $\mathcal{O}_{\mathcal{ER}}(\mathcal{CT}, A_d, \Gamma_t)$: First, $\mathscr{C}$ performs the EMR storage phase (given in Figure 4.4), and if its output is $\perp$, then $\mathscr{C}$ returns $\perp$ as the response of $\mathcal{O}_{\mathcal{ER}}$ oracle. Otherwise, it computes $\mathcal{CT}_u := \langle \Delta_e, \Delta_k, \mathsf{tag2}, E_0, \eta, \breve{E}, \mathsf{tag}\rangle$, and carries out the steps given below.
  Obtain $[\mathcal{DK}_{A_d}, \widetilde{\mathcal{TD}}_{\Gamma_t}] \leftarrow \mathcal{O}'_{\mathcal{TG}}(A_d, \Gamma_t)$ and
  $[token, \mathcal{TDK}] \leftarrow \mathsf{TokenGen}(\mathcal{PP}, \widetilde{\mathcal{TD}}_{\Gamma_t}, \mathcal{DK}_{A_d})$.
  Execute the algorithms $\mathsf{Search}(\mathcal{PP}, \mathcal{CT}_u, token, \mathcal{CSK})$,
  $\mathsf{Transform}(\mathcal{PP}, \mathcal{CT}_u, token, \mathcal{CSK})$, $\mathsf{Verify\text{-}Retrieve}(\mathcal{PP}, \mathcal{CT}_{tr}, \mathcal{TDK})$, in the order. The final output will be sent to $\mathscr{A}$.

When this query phase is over, $\mathscr{A}$ outputs a forgery ciphertext $\mathcal{CT}^{\star}$ of the message $emr^{\star}$ for the signing policy $\Gamma_s^{\star}$, encryption policy $\Gamma_e^{\star}$ and keyword set $W^{\star}$.

$\mathscr{A}$ wins the game if all the following conditions are true. (i) There exist A_d, Γ_t such that $\Gamma_e^{\star}(A_d) = 1, \Gamma_t(W^{\star}) = 1$, $\mathcal{O}_{\mathcal{ER}}(\mathcal{CT}^{\star}, A_d, \Gamma_t) = emr^{\star} \neq \perp$, and (ii) $\mathscr{A}$ was never queried to $\mathcal{O}_{\mathcal{CG}}$ with the input $(emr^{\star}, \Gamma_s^{\star}, \Gamma_e^{\star}, W^{\star})$. The ciphertext $\mathcal{CT}^{\star}$ is parsed as $\mathcal{CT}^{\star} := \langle \Delta_s^{\star}, \Delta_e^{\star}, \Delta_k^{\star}, \mathsf{tag2}^{\star}, E_0^{\star}, \eta^{\star}\rangle$, where $\Delta_s^{\star} := \langle \Gamma_s^{\star}, \sigma'^{\star}, \sigma''^{\star}, \mathsf{tag1}^{\star}, \sigma^{\star}, \{\sigma_i^{\star}\}_{i\in[\ell_s]}\rangle$, $\Delta_e^{\star} := \langle \Gamma_e^{\star}, ct^{\star}, E^{\star}, \{E_{i1}^{\star}, E_{i2}^{\star}\}_{i\in[m]}\rangle$.

The condition (i) implies that $\sigma^{\star} = g^{\alpha/\delta} h^{r'/\delta} \big(g_1^{o_1} g_2\big)^{\theta} \prod_{i\in[\ell_s^{\star}]} H_1\big(\rho_s^{\star}(i)\big)^{\frac{r'a_i}{\delta}+o_2b_i}$, $\sigma_i^{\star} = g^{\frac{r'a_i}{\delta}+o_2b_i}$, $E^{\star} = g^{\theta H_3\left(e(\sigma'^{\star}, \sigma''^{\star})^{\beta}\right)}$, where r', θ, o_2 are random exponents, $\beta = \mathcal{CSK}, \delta = H_3(g_T^{\theta}), o_1 = H_6(ct^{\star}||\mathsf{tag1}^{\star}||\Gamma_e^{\star}||\Gamma_s^{\star}||W^{\star\circ})$, $(a_1, a_2, \ldots, a_{\ell_s^{\star}})$ and $(b_1, b_2, \ldots, b_{\ell_s^{\star}})$ are vectors satisfying respectively $\sum_{i\in[\ell_s^{\star}]} a_i \cdot \vec{M}_s^{\star(i)} = \vec{1}_{n_s^{\star}}$ and $\sum_{i\in[\ell_s^{\star}]} b_i \cdot \vec{M}_s^{\star(i)} = \vec{0}_{n_s^{\star}}$.

Condition (ii) implies that $o_1 = H_6(ct^{\star}||\mathsf{tag1}^{\star}||\Gamma_e^{\star}||\Gamma_s^{\star}||W^{\star\circ}) = \check{o}_1$ (this is of type (i) query, $\mathscr{C}$ can obtain this value from Tab_{H_6}). Now, $\mathscr{C}$ computes $[token, \mathcal{TDK}] \leftarrow \mathcal{O}'_{\mathcal{TG}}(A_d, \Gamma_t)$ and then obtains $\delta^{\star} = H_3(\Lambda)$ by executing the algorithms $\mathsf{Search}(\mathcal{PP}, \mathcal{CT}_u^{\star}, token, \mathcal{CSK})$, $\mathsf{Transform}(\mathcal{PP}, \mathcal{CT}_u^{\star}, token, \mathcal{CSK})$ and $\mathsf{Verify\text{-}Retrieve}(\mathcal{PP}, \mathcal{CT}_{tr}^{\star}, \mathcal{TDK})$. Condition (i) implies that $\delta^{\star} = \delta$.

Since $\sigma^\star = g^{\alpha/\delta} h^{r'/\delta} \big(g_1^{o_1} g_2\big)^{\theta} \prod_{i\in[\ell_s^\star]} H_1\big(\rho_s^\star(i)\big)^{\frac{r' a_i}{\delta}+o_2 b_i}$, from the simulation of H_1 hash function, one can see that $\sigma^\star = g^{\alpha/\delta} \big(g_1^{o_1} g_2\big)^{\theta} \prod_{i\in[\ell_s^\star]} \big(g^{\upsilon_{\rho_s^\star(i)}}\big)^{\frac{r' a_i}{\delta}+o_2 b_i}$. This is due to the fact that

$$\begin{aligned}
\sigma^\star &= g^{\alpha/\delta} h^{r'/\delta} \big(g_1^{o_1} g_2\big)^{\theta} \prod_{i\in[\ell_s^\star]} H_1\big(\rho_s^\star(i)\big)^{\frac{r' a_i}{\delta}+o_2 b_i} \\
&= g^{\alpha/\delta} h^{r'/\delta} \big(g_1^{o_1} g_2\big)^{\theta} \prod_{i\in[\ell_s^\star]} \big(g^{\upsilon_{\rho_s^\star(i)}} \prod_{j\in[n_s^\star]} \big(g^{\phi^j}\big)^{-M_{sj}^{\star(i)}}\big)^{\frac{r' a_i}{\delta}+o_2 b_i} \\
&= g^{\alpha/\delta} h^{r'/\delta} \big(g_1^{o_1} g_2\big)^{\theta} \prod_{i\in[\ell_s^\star]} \big(g^{\upsilon_{\rho_s^\star(i)}}\big)^{\frac{r' a_i}{\delta}+o_2 b_i} \big(\prod_{i\in[\ell_s^\star]} \prod_{j\in[n_s^\star]} \big(g^{\phi^j}\big)^{-M_{sj}^{\star(i)}}\big)^{\frac{r' a_i}{\delta}+o_2 b_i} \\
&= g^{\alpha/\delta} h^{r'/\delta} \big(g_1^{o_1} g_2\big)^{\theta} \prod_{i\in[\ell_s^\star]} \big(g^{\upsilon_{\rho_s^\star(i)}}\big)^{\frac{r' a_i}{\delta}+o_2 b_i} \Big(g^{-\sum_{i\in[\ell_s^\star]}\sum_{j\in[n_s^\star]} \phi^j M_{sj}^{\star(i)}(\frac{r' a_i}{\delta}+o_2 b_i)}\Big)
\end{aligned}$$

Since $\sum_{i\in[\ell_s^\star]} a_i \cdot \vec{M}_s^{\star(i)} = \vec{1}_{n_s^\star}$ and $\sum_{i\in[\ell_s^\star]} b_i \cdot \vec{M}_s^{\star(i)} = \vec{0}_{n_s^\star}$, we can see that

$$-\sum_{i\in[\ell_s^\star]}\sum_{j\in[n_s^\star]} \phi^j M_{sj}^{\star(i)}\big(\frac{r' a_i}{\delta}+o_2 b_i\big) = -\phi\frac{r'}{\delta}$$

and hence

$$g^{-\sum_{i\in[\ell_s^\star]}\sum_{j\in[n_s^\star]} \phi^j M_{sj}^{\star(i)}(\frac{r' a_i}{\delta}+o_2 b_i)} = h^{-r'/\delta}$$

Now, $\mathscr{C}$ calculates $\breve{E}^\star := (E^\star)^{1/H_3\left(e(\sigma'^\star,\sigma''^\star)^\beta\right)} = g^\theta$ and then computes the unknown value $g^{\phi^{q+1}}$ as

$$g^{\phi^{q+1}} = \left(\frac{\sigma^\star}{g^{\alpha'/\delta^\star}\big(\breve{E}^\star\big)^{z_1 o_1+z_2}\prod_{i\in[\ell_s^\star]}(\sigma_i^\star)^{\upsilon_{\rho_s^\star(i)}}}\right)^{\delta^\star}$$

Therefore, if the advantage of $\mathscr{A}$ in the game $\mathsf{Game}_{\mathscr{A}}^{\mathsf{EUF\text{-}CMA}}$ is non-negligible, then $\mathscr{C}$ can solve the q-DHE problem with non-negligible advantage. □

Proof of Theorem 8

Proof. The challenger $\mathscr{C}$ interacts with an adversary $\mathscr{A}$ as in $\mathsf{Game}_{\mathscr{A}}^{\mathsf{EO\text{–}Anonymity}}$ (formulated in Section 4.2.2).

(1) $\mathscr{C}$ computes $[\mathcal{KPK}, \mathcal{MK}] \leftarrow \mathsf{KGA\text{-}Setup}(1^\kappa)$, $[\mathcal{TPK}, \mathcal{TSK}] \leftarrow \mathsf{TGA\text{-}Setup}(\mathcal{KPK})$, $[\mathcal{CPK}, \mathcal{CSK}] \leftarrow \mathsf{HCS\text{-}Setup}(\mathcal{KPK})$ and sends $[\mathcal{PP}, \mathcal{MK}, \mathcal{TSK}, \mathcal{CSK}]$ to $\mathscr{A}$, where $\mathcal{PP} := \langle \mathcal{KPK}, \mathcal{CPK}, \mathcal{TPK}\rangle$.

(2) In this step, $\mathscr{A}$ does not need to query any oracle and it can compute required components by itself because $\mathscr{A}$ has the knowledge of system master secret, cloud secret key and trapdoor secret key. $\mathscr{A}$ sends to $\mathscr{C}$ a message emr, a signing policy Γ_s, an encryption policy Γ_e, a keyword set W and two signing attribute sets $A_s^{(0)}, A_s^{(1)}$ obeying the condition $\Gamma_s(A_s^{(0)}) = 1 = \Gamma_s(A_s^{(1)})$.

(3) $\mathscr{C}$ samples $i \xleftarrow{\text{u}} \{0,1\}$, computes the signing key $\mathcal{SK}_{A_s^{(i)}} \leftarrow \mathsf{sKeyGen}(\mathcal{PP}, \mathcal{MK}, A_s^{(i)})$, the challenge ciphertext $\mathcal{CT}^\star \leftarrow \mathsf{Signcrypt}(\mathcal{PP}, A_s^{(i)}, \Gamma_s, \Gamma_e, W, emr)$, and sends $\mathcal{CT}^\star$ to $\mathscr{A}$.

(4) $\mathscr{A}$ outputs its guess i' of i.

From the challenge ciphertext $\mathcal{CT}^\star$, the signature of the message emr for the signing policy $\Gamma_s := (\mathbf{M}_s, \rho_s)$ is $\Delta_s := \langle \Gamma_s, \sigma', \sigma'', \mathsf{tag1}, \sigma, \{\sigma_i\}_{i\in[\ell_s]} \rangle$. Note that σ and σ_i are the only signing attribute components, and the distribution of the signature is as follows.

$$\sigma' := g^{\delta'}, \sigma'' := h^{\delta''}, \mathsf{tag1} := H_4(g_T^{1/\delta} \cdot e(\sigma', Y)^{\delta''}),$$

$$\sigma = g^{\alpha/\delta} h^{r'/\delta} \big(g_1^{o_1} g_2\big)^{\theta} \prod_{i\in[\ell_s]} H_1\big(\rho_s(i)\big)^{\frac{r' a_i}{\delta} + o_2 b_i} \text{ and } \sigma_i = g^{\frac{r' a_i}{\delta} + o_2 b_i}$$

where $\delta', \delta'', \theta, o_2, b_i$ are random elements of $\mathbb{Z}_p^*$ selected during signing. Hence, all the components of the signature are random elements from adversary's point of view. That is, the signature is independent of the signing key (and hence of the signing attribute set) being used to generate it. Therefore, the challenge ciphertext gives no information about i in $\mathsf{Game}_{\mathscr{A}}^{\mathsf{EO\text{-}Anonymity}}$ to the adversary $\mathscr{A}$. Due to this, $\mathscr{A}$ can output just random guess i'. In this case, $\text{Prob}[i' = i] = 1/2$. Hence, $\mathscr{A}$'s advantage $\mathsf{Adv}_{\mathscr{A}}^{\mathsf{EO\text{-}Anonymity}}(1^\kappa) \stackrel{\text{def}}{=} \text{Prob}[i' = i] = 1/2$. Thus, MediCare provides EO anonymity. □

Proof of Theorem 9

Proof. If there exists an adversary $\mathscr{A}$ that wins the verifiability game $\mathsf{Game}_{\mathscr{A}}^{\mathsf{verifiability}}$ (presented in Section 4.2.2) of MediCare, then an authorized EU $\mathscr{C}$ can find a collision for the hash function H_4 or H_5 by interacting with $\mathscr{A}$ as described below.

(1) $\mathscr{C}$ computes $[\mathcal{KPK}, \mathcal{MK}] \leftarrow \mathsf{KGA\text{-}Setup}(1^\kappa)$, $[\mathcal{TPK}, \mathcal{TSK}] \leftarrow \mathsf{TGA\text{-}Setup}(\mathcal{KPK})$, $[\mathcal{CPK}, \mathcal{CSK}] \leftarrow \mathsf{HCS\text{-}Setup}(\mathcal{KPK})$, and sends $[\mathcal{PP}, \mathcal{CSK}]$ to $\mathscr{A}$, where $\mathcal{PP} := \langle \mathcal{KPK}, \mathcal{CPK}, \mathcal{TPK} \rangle$. The secret keys $\mathcal{MK}$ and $\mathcal{TSK}$ are kept secret by $\mathscr{C}$.

(2) $\mathscr{A}$ adaptively queries signing key generation oracle $\mathcal{O}_{\mathcal{SKG}}(A_s)$, token generation oracle $\mathcal{O}''_{\mathcal{TG}}(A_d, \Gamma_t)$, ciphertext generation oracle $\mathcal{O}_{\mathcal{CG}}(emr, \Gamma_s, \Gamma_e, W)$ and EMR retrieval oracle $\mathcal{O}_{\mathcal{ER}}(\mathcal{CT}, A_d, \Gamma_t)$. Since $\mathscr{C}$ knows the system master secret $\mathcal{MK}$ and TGA's secret key $\mathcal{TSK}$, it can simulate $\mathscr{A}$'s queries properly. At the end of this phase, $\mathscr{A}$ announces a message $emr^\star$, an encryption policy $\Gamma_e^\star$, a signing policy $\Gamma_s^\star$, a keyword set $W^\star$, and sends $[emr^\star, \Gamma_e^\star, \Gamma_s^\star, W^\star]$ to $\mathscr{C}$.

(3) $\mathscr{C}$ selects a signing attribute set A_s such that $\Gamma_s^\star(A_s) = 1$ and obtains the signing key $\mathcal{SK}_{A_s} \leftarrow \mathsf{sKeyGen}(\mathcal{PP}, \mathcal{MK}, A_s)$. Next, it computes $\mathcal{CT}^\star \leftarrow \mathsf{Signcrypt}(\mathcal{PP}, \mathcal{SK}_{A_s}, \Gamma_s^\star, \Gamma_e^\star, W^\star, emr^\star)$. Then, the ciphertext $\mathcal{CT}^\star$ will be given to $\mathscr{A}$. Here $\mathcal{CT}^\star := \langle \Delta_s^\star, \Delta_e^\star, \Delta_k^\star, \mathsf{tag2}^\star, E_0^\star, \eta^\star \rangle$ and $\Delta_e^\star := \langle \Gamma_e^\star, ct^\star, E^\star, \{E_{i1}^\star, E_{i2}^\star\}_{i \in [m]} \rangle$.

(4) Again $\mathscr{A}$ queries the oracles $\mathcal{O}_{\mathcal{SKG}}, \mathcal{O}''_{\mathcal{TG}}, \mathcal{O}_{\mathcal{CG}}, \mathcal{O}_{\mathcal{ER}}$ and obtains the respective responses as in step (2). At the end of this phase, $\mathscr{A}$ outputs a decryption attribute set A_d, a keyword policy Γ_t and a transformed ciphertext $\mathcal{CT}_{tr} := \langle \mathcal{Y}_1, \mathcal{Y}_2, ct, \mathsf{tag2}^\star, \mathsf{tag} \rangle$ such that $\Gamma_e^\star(A_d) = 1 \wedge \Gamma_t(W^\star) = 1$.

Suppose that the tuple $[\![A_d, \Gamma_t, token, \mathcal{TDK}]\!]$ is in table $\mathsf{Tab}''_{\mathcal{ER}}$, where $\mathcal{TDK} := \langle \tau_1, \tau_2 \rangle$. If not, it can be generated by querying the oracle $\mathcal{O}''_{\mathcal{TG}}$ with the input (A_d, Γ_t). If $\mathscr{A}$ breaks the verifiability of MediCare, $\mathscr{C}$ will recover a plaintext $emr \leftarrow \mathsf{Verify\text{-}Retrieve}(\mathcal{PP}, \mathcal{CT}_{tr}, \mathcal{TDK})$, where $emr \notin \{emr^\star, \perp\}$, as follows. $\mathscr{C}$ (i) computes $\Lambda := \mathcal{Y}_1^{-\tau_1} \cdot \mathcal{Y}_2^{\tau_2}, \delta := H_3(\Lambda)$, (ii) observes $H_4(g_T^{1/\delta}) = \mathsf{tag}$, (iii) observes $H_5(\delta || ct) = \mathsf{tag2}^\star$ and (iv) obtains $emr = ct \oplus \mathsf{KDF}(\Lambda)$.

If $\Lambda^\star$ and $ct^\star$ are the respective components of Λ and ct being used in the generation of $\mathcal{CT}^\star$, then there are two possibilities $\Lambda \neq \Lambda^\star$ or $\Lambda = \Lambda^\star$. Note that (iii) implies that $H_5\big(H_3(\Lambda)||ct\big) = \mathsf{tag2}^\star = H_5\big(H_3(\Lambda^\star)||ct^\star\big)$.

If $\Lambda \neq \Lambda^\star$, $H_3(\Lambda)||ct \neq H_3(\Lambda^\star)||ct^\star$ and hence the pair $\big(H_3(\Lambda)||ct, H_3(\Lambda^\star)||ct^\star\big)$ forms a collision for the hash function H_5.

Suppose $\Lambda = \Lambda^\star$. Then, $ct \oplus \mathsf{KDF}(\Lambda) = emr \neq emr^\star = ct^\star \oplus \mathsf{KDF}(\Lambda^\star)$ implies $ct \neq ct^\star$. So, $H_3(\Lambda)||ct \neq H_3(\Lambda)||ct^\star$ and hence the pair $\big(H_3(\Lambda)||ct, H_3(\Lambda)||ct^\star\big)$ forms a collision for H_5.

In both the cases, $\mathscr{C}$ finds a collision for H_5. Since H_5 is a collision-resistant hash function, $\mathscr{A}$ cannot win the verifiability game $\mathsf{Game}_{\mathscr{A}}^{\mathsf{verifiability}}$ with non-negligible advantage and hence MediCare is verifiable. □

Table A.1: The sequence of $2\varsigma + 2$ games $\mathsf{G}_{real}, \mathsf{G}_0, \mathsf{G}_1, \ldots, \mathsf{G}_{2\varsigma}$.

Game	*The Δ_k of the Challenge Ciphertext $\mathcal{CT}^\star$*
G_{real}	$W^\circ, k_T, \{L_{\mathcal{W}_j 1}, L_{\mathcal{W}_j 2}\}_{j\in[\varsigma]}$, $\{K_{\mathcal{W}_1 1}, K_{\mathcal{W}_1 2}, K_{\mathcal{W}_1 3}, K_{\mathcal{W}_1 4}\}, \{K_{\mathcal{W}_2 1}, K_{\mathcal{W}_2 2}, K_{\mathcal{W}_2 3}, K_{\mathcal{W}_2 4}\}, \ldots, \{K_{\mathcal{W}_\varsigma 1}, K_{\mathcal{W}_\varsigma 2}, K_{\mathcal{W}_\varsigma 3}, K_{\mathcal{W}_\varsigma 4}\}$
G_0	$W^\circ, \boxed{r_T}, \{L_{\mathcal{W}_j 1}, L_{\mathcal{W}_j 2}\}_{j\in[\varsigma]}$, $\{K_{\mathcal{W}_1 1}, K_{\mathcal{W}_1 2}, K_{\mathcal{W}_1 3}, K_{\mathcal{W}_1 4}\}, \{K_{\mathcal{W}_2 1}, K_{\mathcal{W}_2 2}, K_{\mathcal{W}_2 3}, K_{\mathcal{W}_2 4}\}, \ldots, \{K_{\mathcal{W}_\varsigma 1}, K_{\mathcal{W}_\varsigma 2}, K_{\mathcal{W}_\varsigma 3}, K_{\mathcal{W}_\varsigma 4}\}$
G_1	$W^\circ, \boxed{r_T}, \{L_{\mathcal{W}_j 1}, L_{\mathcal{W}_j 2}\}_{j\in[\varsigma]}$, $\{\boxed{R_{\mathcal{W}_1 1}}, K_{\mathcal{W}_1 2}, K_{\mathcal{W}_1 3}, K_{\mathcal{W}_1 4}\}, \{K_{\mathcal{W}_2 1}, K_{\mathcal{W}_2 2}, K_{\mathcal{W}_2 3}, K_{\mathcal{W}_2 4}\}, \ldots, \{K_{\mathcal{W}_\varsigma 1}, K_{\mathcal{W}_\varsigma 2}, K_{\mathcal{W}_\varsigma 3}, K_{\mathcal{W}_\varsigma 4}\}$
G_2	$W^\circ, \boxed{r_T}, \{L_{\mathcal{W}_j 1}, L_{\mathcal{W}_j 2}\}_{j\in[\varsigma]}$, $\{\boxed{R_{\mathcal{W}_1 1}}, K_{\mathcal{W}_1 2}, K_{\mathcal{W}_1 3}, K_{\mathcal{W}_1 4}\}, \{\boxed{R_{\mathcal{W}_2 1}}, K_{\mathcal{W}_2 2}, K_{\mathcal{W}_2 3}, K_{\mathcal{W}_2 4}\}, \ldots, \{K_{\mathcal{W}_\varsigma 1}, K_{\mathcal{W}_\varsigma 2}, K_{\mathcal{W}_\varsigma 3}, K_{\mathcal{W}_\varsigma 4}\}$
$\vdots$	
G_ς	$W^\circ, \boxed{r_T}, \{L_{\mathcal{W}_j 1}, L_{\mathcal{W}_j 2}\}_{j\in[\varsigma]}$, $\{\boxed{R_{\mathcal{W}_1 1}}, K_{\mathcal{W}_1 2}, K_{\mathcal{W}_1 3}, K_{\mathcal{W}_1 4}\}, \{\boxed{R_{\mathcal{W}_2 1}}, K_{\mathcal{W}_2 2}, K_{\mathcal{W}_2 3}, K_{\mathcal{W}_2 4}\}, \ldots, \{\boxed{R_{\mathcal{W}_\varsigma 1}}, K_{\mathcal{W}_\varsigma 2}, K_{\mathcal{W}_\varsigma 3}, K_{\mathcal{W}_\varsigma 4}\}$
$\mathsf{G}_{\varsigma+1}$	$W^\circ, \boxed{r_T}, \{L_{\mathcal{W}_j 1}, L_{\mathcal{W}_j 2}\}_{j\in[\varsigma]}$, $\{\boxed{R_{\mathcal{W}_1 1}}, K_{\mathcal{W}_1 2}, \boxed{R_{\mathcal{W}_1 3}}, K_{\mathcal{W}_1 4}\}, \{\boxed{R_{\mathcal{W}_2 1}}, K_{\mathcal{W}_2 2}, K_{\mathcal{W}_2 3}, K_{\mathcal{W}_2 4}\}, \ldots, \{\boxed{R_{\mathcal{W}_\varsigma 1}}, K_{\mathcal{W}_\varsigma 2}, K_{\mathcal{W}_\varsigma 3}, K_{\mathcal{W}_\varsigma 4}\}$
$\mathsf{G}_{\varsigma+2}$	$W^\circ, \boxed{r_T}, \{L_{\mathcal{W}_j 1}, L_{\mathcal{W}_j 2}\}_{j\in[\varsigma]}$, $\{\boxed{R_{\mathcal{W}_1 1}}, K_{\mathcal{W}_1 2}, \boxed{R_{\mathcal{W}_1 3}}, K_{\mathcal{W}_1 4}\}, \{\boxed{R_{\mathcal{W}_2 1}}, K_{\mathcal{W}_2 2}, \boxed{R_{\mathcal{W}_2 3}}, K_{\mathcal{W}_2 4}\}, \ldots, \{\boxed{R_{\mathcal{W}_\varsigma 1}}, K_{\mathcal{W}_\varsigma 2}, K_{\mathcal{W}_\varsigma 3}, K_{\mathcal{W}_\varsigma 4}\}$
$\vdots$	
$\mathsf{G}_{2\varsigma}$	$W^\circ, \boxed{r_T}, \{L_{\mathcal{W}_j 1}, L_{\mathcal{W}_j 2}\}_{j\in[\varsigma]}$, $\{\boxed{R_{\mathcal{W}_1 1}}, K_{\mathcal{W}_1 2}, \boxed{R_{\mathcal{W}_1 3}}, K_{\mathcal{W}_1 4}\}, \{\boxed{R_{\mathcal{W}_2 1}}, K_{\mathcal{W}_2 2}, \boxed{R_{\mathcal{W}_2 3}}, K_{\mathcal{W}_2 4}\}, \ldots, \{\boxed{R_{\mathcal{W}_\varsigma 1}}, K_{\mathcal{W}_\varsigma 2}, \boxed{R_{\mathcal{W}_\varsigma 3}}, K_{\mathcal{W}_\varsigma 4}\}$

Proof of Lemma 7

Proof. We prove this security notion (modeled as $\mathsf{Game}^{\mathsf{IND\text{-}CKA}}_{\mathsf{Type\text{-}1}}$ in Section 4.2.2) employing a hybrid experiment which consisting of a sequence of games between a challenger $\mathscr{C}$ and a Type-1 adversary $\mathscr{A}$. The individual games differ in how $\mathscr{C}$ constructs the challenge ciphertext given to $\mathscr{A}$. Let $\mathcal{CT}^\star := \langle \Delta_s, \Delta_e, \Delta_k, \mathsf{tag2}, E_0, \eta \rangle$ be the challenge ciphertext given to $\mathscr{A}$ during IND-CKA game. Let ς be the size of the keyword set used in computation of $\mathcal{CT}^\star$ such that $\varsigma \leq q$. The sequence of $2\varsigma + 2$ games $\mathsf{G}_{real}, \mathsf{G}_0, \mathsf{G}_1, \ldots, \mathsf{G}_{2\varsigma}$ is defined in Table A.1. In these games, only the components in Δ_k are modified (precisely, some components are replaced by random elements) and all other elements of $\mathcal{CT}^\star$ remain unchanged. For brevity, we omit the components $\Delta_s, \Delta_e, \mathsf{tag2}, E_0, \eta$ from $\mathcal{CT}^\star$ and present only the components of $\Delta_k := \langle W^\circ, k_T, \{K_{\mathcal{W}1}, K_{\mathcal{W}2}, K_{\mathcal{W}3}, K_{\mathcal{W}4}, L_{\mathcal{W}1}, L_{\mathcal{W}2}\}_{\mathcal{W}\in W^\circ} \rangle$ in Table A.1. Let r_T be the random element of $\mathbb{G}_T$ and $R_{\mathcal{W}_1 1}, R_{\mathcal{W}_1 3}, R_{\mathcal{W}_2 1}, R_{\mathcal{W}_2 3}, \ldots, R_{\mathcal{W}_\varsigma 1}, R_{\mathcal{W}_\varsigma 3}$ be random elements of $\mathbb{G}$.

Note that all the components of Δ_k in the challenge ciphertext computed in $\mathsf{G}_{2\varsigma}$ are random elements and hence the challenge ciphertext is independent of the two keyword sets submitted by $\mathscr{A}$. But, the challenge ciphertext of G_{real} is well formed. The challenge ciphertext in $\mathsf{G}_{2\varsigma}$ reveals nothing about its keyword set. To complete

the proof, we show that the transitions from G_{real} to G_0 to G_1 to ... to $\mathsf{G}_{2\varsigma}$ are all computationally indistinguishable (in Claims 4, 5 and 6).

Claim 4. *Under the assumption that q-2 problem is hard, there is no PPT Type-1 adversary that distinguishes between the games G_{real} and G_0 with non-negligible advantage.*

Proof. Assume there is a Type-1 adversary $\mathscr{A}$ that distinguishes between the games G_{real} and G_0 with non-negligible advantage. Then, we construct a challenger $\mathscr{C}$ that solves q-2 problem with non-negligible advantage by interacting with $\mathscr{A}$ as follows. Given the q-2 problem instance $\big\langle \Sigma, g, g^{\phi_1}, g^{\phi_2}, g^{\phi_3}, g^{(\phi_1\phi_3)^2}, \{g^{\psi_i}, g^{\phi_1\phi_3\psi_i}, g^{\phi_1\phi_3/\psi_i}, g^{\phi_1^2\phi_3\psi_i}, g^{\phi_2/\psi_i^2}, g^{\phi_2^2/\psi_i^2}\}_{i\in[q]}, \{g^{\phi_1\phi_3\psi_i/\psi_j}, g^{\phi_2\psi_i/\psi_j^2}, g^{\phi_1\phi_2\phi_3\psi_i/\psi_j^2}, g^{(\phi_1\phi_3)^2\psi_i/\psi_j}\}_{(i,j)\in[q,q],i\neq j}, Z\big\rangle$, the task for $\mathscr{C}$ is to determine whether $Z = e(g,g)^{\phi_1\phi_2\phi_3}$ or Z is a random element of $\mathbb{G}_T$.

(1) $\mathscr{A}$ announces two challenge keyword sets $W_0^\star := \{[\mathcal{W}_1 : w_1^{(0)}], \ldots, [\mathcal{W}_\varsigma : w_\varsigma^{(0)}]\}$ and $W_1^\star := \{[\mathcal{W}_1 : w_1^{(1)}], \ldots, [\mathcal{W}_\varsigma : w_\varsigma^{(1)}]\}$. Note that these two sets satisfy the conditions $|W_0^\star| = |W_1^\star| = \varsigma$ and $W_0^{\star\circ} = W_1^{\star\circ} = \{\mathcal{W}_1, \ldots, \mathcal{W}_\varsigma\}$ of $\mathsf{Game}^{\mathsf{IND\text{-}CKA}}_{\mathsf{Type\text{-}1}}$.

(2) Firstly, $\mathscr{C}$ samples $\mu \xleftarrow{\text{u}} \{0,1\}$ and sets $W_\mu^\star := \{[\mathcal{W}_1 : w_1^{(\mu)}], \ldots, [\mathcal{W}_\varsigma : w_\varsigma^{(\mu)}]\}$. Then, $\mathscr{C}$ selects $\alpha', z, z_i \xleftarrow{\text{u}} \mathbb{Z}_p^*, i \in \{1,2,\ldots,7,9,10\}$, implicitly defines $\alpha := \alpha'\phi_2$ and sets

$$g_T := e(g, g^{\phi_2})^{\alpha'}, h := g^z, g_i := g^{z_i} \text{ for } i \in \{1,2,3,4,5,10\},$$
$$g_6 := g^{z_6} \prod_{j\in[\varsigma]} g^{\phi_2/\psi_j^2}, g_7 := g^{z_7} \prod_{j\in[\varsigma]} (g^{\phi_1\phi_3/\psi_j})\big(g^{\phi_2/\psi_j^2}\big)^{-w_j^{(\mu)}},$$
$$g_8 := g^{\phi_1}, g_9 := g^{\phi_1} g^{z_9}$$

Next, it selects seven collision-resistant hash functions $\{H_i\}_{i=1}^7$ (as mentioned in the construction) and sets $\mathcal{KPK} := \langle \Sigma, g_T, g, h, \{g_i\}_{i=1}^{10}, \mathcal{M}, \mathsf{KDF}, \{H_i\}_{i=1}^7\rangle$, where $\mathcal{M} := \{0,1\}^{\ell_{pt}}$ is the plaintext space, and KDF is the key derivation function with output length ℓ_{pt} and keying source $\mathbb{G}_T$. Here $\mathcal{MK} := (g^{\phi_2})^{\alpha'}$. Finally, $\mathscr{C}$ selects $\beta, \gamma_0, \varpi_1, \varpi_2, \varpi_3, \varpi_4 \xleftarrow{\text{u}} \mathbb{Z}_p^*$, computes $h_1 := g^{\varpi_1}, h_2 := g^{\varpi_2}$, $h_3 := g^{\varpi_3}, h_4 := g^{\varpi_4}, h_T := e(g, g_{10})^{\gamma_0} \cdot e(g^{\phi_1}, g^{\phi_2})^{z_{10}}, Y := h^\beta$ and sets $\mathcal{TPK} := \langle h_T, h_1, h_2, h_3, h_4\rangle, \mathcal{TSK} := \langle \gamma := \gamma_0 + \phi_1\phi_2, \varpi_1, \varpi_2, \varpi_3, \varpi_4\rangle$, $\mathcal{CPK} := Y, \mathcal{CSK} := \beta$. Note that $\gamma := \gamma_0 + \phi_1\phi_2$ is implicitly defined. Lastly, $\mathscr{C}$ sends the tuple $[\mathcal{PP}, \mathcal{CSK}]$ to $\mathscr{A}$, where $\mathcal{PP} := \langle \mathcal{KPK}, \mathcal{CPK}, \mathcal{TPK}\rangle$.

(3) $\mathscr{A}$ queries signing key generation oracle $\mathcal{O}_{\mathcal{SKG}}(A_s)$, token generation oracle $\mathcal{O}'''_{\mathcal{TG}}(A_d, \Gamma_t)$ and search oracle $\mathcal{O}_{search}(\mathcal{CT}, \Gamma_t)$. Then $\mathscr{C}$ responds as follows.

- $\mathcal{O}_{\mathcal{SKG}}(A_s) : \mathscr{C}$ chooses $r' \xleftarrow{u} \mathbb{Z}_p^*$, sets $S := (g^{\phi_2})^{\alpha'} h^{r'}, S_0 := g^{r'}$, $S_x := (H_1(x))^{r'}, \forall x \in A_s$, and returns $\mathcal{SK}_{A_s} := \langle A_s, S, S_0, \{S_x\}_{x \in A_s} \rangle$.
- $\mathcal{O}'''_{\mathcal{TG}}(A_d, \Gamma_t)$: $\mathscr{A}$ submits a decryption attribute set A_d and a keyword policy Γ_t with the condition that $\Gamma_t(W_0^\star) = 0 \wedge \Gamma_t(W_1^\star) = 0$. Hence $\Gamma_t(W_\mu^\star) = 0$. Let $\Gamma_t := (\mathbf{M}_t, \rho_t^\circ, \{w_{\rho_t^\circ(i)}\}_{i \in [\ell_t]})$, where $\mathbf{M}_t$ is a matrix of size $\ell_t \times n_t$. Since $\Gamma_t(W_\mu^\star) = 0$, $\mathscr{C}$ can compute a vector $\vec{\varepsilon} := (\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_{n_t}) \in \mathbb{Z}_p^{n_t}$ such that $\varepsilon_1 = -1$ and $\vec{M}_t^{(i)} \cdot \vec{\varepsilon} = 0$, $\forall i \in \{i | \rho_t^\circ(i) \in W_\mu^{\star\circ}\}$. It selects $\breve{\varepsilon}_2, \ldots, \breve{\varepsilon}_{n_t}, f, f' \xleftarrow{u} \mathbb{Z}_p^*$, sets $T_1 := g^f$, $T_2 := h^{f'}, \partial := H_3(e(T_1, Y)^{f'})$, and implicitly sets $\vec{\vartheta} := (0, \breve{\varepsilon}_2, \ldots, \breve{\varepsilon}_{n_t}) - \partial(\gamma_0 + \phi_1\phi_2)\vec{\varepsilon}$. For each row $i \in [\ell_t]$, its share is

$$\vartheta_i = \vec{M}_t^{(i)} \cdot \vec{\vartheta} = \vec{M}_t^{(i)} \cdot (0, \breve{\varepsilon}_2, \ldots, \breve{\varepsilon}_{n_t}) - \partial\gamma_0 \vec{M}_t^{(i)} \cdot \vec{\varepsilon} - \partial\phi_1\phi_2 \vec{M}_t^{(i)} \cdot \vec{\varepsilon}$$

It samples $\tau'_2, \tau'_3, \varepsilon'_2, \ldots, \varepsilon'_{n_t} \xleftarrow{u} \mathbb{Z}_p^*$, implicitly defines $\tau_3 := \tau'_3 \partial\phi_1\phi_2 z_{10}/z$, $\tau_1 := \tau'_3, \tau_2 := \tau'_2\phi_2$, $\vec{\breve{\vartheta}} := (0, \varepsilon'_2, \ldots, \varepsilon'_{n_t}) - (\partial\phi_1\phi_2 z_{10}/z)\vec{\varepsilon}$. In this case, the share of the row $i \in [\ell_t]$ is

$$\breve{\vartheta}_i := \vec{M}_t^{(i)} \cdot \vec{\breve{\vartheta}} = \vec{M}_t^{(i)} \cdot (0, \varepsilon'_2, \ldots, \varepsilon'_{n_t}) - (\partial\phi_1\phi_2 z_{10}/z)\vec{M}_t^{(i)} \cdot \vec{\varepsilon}$$

$\mathscr{C}$ calculates the transform trapdoor $\mathcal{TD}_{\Gamma_t^\circ} := \langle \Gamma_t^\circ, T_1, T_2, \{T_{i1}, T_{i2}, T_{i3}, V_{i1}, V_{i2}, V_{i3}, V_{i4}\}_{i \in [\ell_t]} \rangle$ (mentioned in Equation (5.4)) in the following way. Let $\mathsf{vt}_i := \vec{M}_t^{(i)} \cdot (0, \breve{\varepsilon}_2, \ldots, \breve{\varepsilon}_{n_t})$ and $\mathsf{ut}_i := \vec{M}_t^{(i)} \cdot (0, \varepsilon'_2, \ldots, \varepsilon'_{n_t})$.

Case-1: For the row $i \in [\ell_t]$ where $\rho_t^\circ(i) \in W_\mu^{\star\circ}$. In this case, $\vec{M}_t^{(i)} \cdot \vec{\varepsilon} = 0$ and therefore $\vartheta_i = \vec{M}_t^{(i)} \cdot (0, \breve{\varepsilon}_2, \ldots, \breve{\varepsilon}_{n_t}) = \mathsf{vt}_i$ and $\breve{\vartheta}_i = \vec{M}_t^{(i)} \cdot (0, \varepsilon'_2, \ldots, \varepsilon'_{n_t}) = \mathsf{ut}_i$. Choose $\breve{r}_i, \breve{r}'_i \xleftarrow{u} \mathbb{Z}_p^*$, set

$$\begin{aligned} T_{i1} &:= g_{10}^{\mathsf{vt}_i} \cdot g_8^{\varpi_1\varpi_2\breve{r}_i + \varpi_3\varpi_4\breve{r}'_i}, \\ T_{i2} &:= h^{\mathsf{ut}_i} \cdot H_2(e(T_1, Y)^{f'} || \Gamma_t^\circ || \vec{M}_t^{(i)}) \cdot g_9^{\varpi_1\varpi_2\breve{r}_i + \varpi_3\varpi_4\breve{r}'_i}, \\ T_{i3} &:= g^{\varpi_1\varpi_2\breve{r}_i + \varpi_3\varpi_4\breve{r}'_i}, \end{aligned}$$

and compute $V_{i1}, V_{i2}, V_{i3}, V_{i4}$ as in Equation (4.1).

Case-2: For the row $i \in [\ell_t]$ where $\rho_t^\circ(i) \notin W_\mu^{\star\circ}$ (i.e., $w_{\rho_t^\circ(i)} \neq w_j^{(\mu)}, \forall j \in [\varsigma]$). In this case, $\vec{M}_t^{(i)} \cdot \vec{\varepsilon} \neq 0$, and $\mathscr{C}$ can-

not compute $\partial\phi_1\phi_2\vec{M}_t^{(i)}\cdot\vec{\varepsilon}$ and $(\partial\phi_1\phi_2 z_{10}/z)\vec{M}_t^{(i)}\cdot\vec{\varepsilon}$. However, by properly defining $\breve{r}_i$ and $\breve{r}'_i$, $\mathscr{C}$ is able to calculate $T_{i1}, T_{i2}, T_{i3}, V_{i1}, V_{i2}, V_{i3}, V_{i4}$. Choose $r_i, r'_i \xleftarrow{\text{u}} \mathbb{Z}_p^*$, implicitly define

$$\begin{aligned}\breve{r}_i &:= r_i + \frac{\partial\phi_2 z_{10}(\vec{M}_t^{(i)}\cdot\vec{\varepsilon})}{2\varpi_1\varpi_2} - \sum_{j\in[\varsigma]}\frac{\partial\phi_1\phi_3\psi_j z_{10}(\vec{M}_t^{(i)}\cdot\vec{\varepsilon})}{2\varpi_1\varpi_2(w_{\rho_t^\circ(i)}-w_j^{(\mu)})}\\ \breve{r}'_i &:= r'_i + \frac{\partial\phi_2 z_{10}(\vec{M}_t^{(i)}\cdot\vec{\varepsilon})}{2\varpi_3\varpi_4} - \sum_{j\in[\varsigma]}\frac{\partial\phi_1\phi_3\psi_j z_{10}(\vec{M}_t^{(i)}\cdot\vec{\varepsilon})}{2\varpi_3\varpi_4(w_{\rho_t^\circ(i)}-w_j^{(\mu)})}\end{aligned}$$

and compute

$$\begin{aligned}T_{i3} &:= g^{\varpi_1\varpi_2 r_i+\varpi_3\varpi_4 r'_i}\cdot(g^{\phi_2})^{\partial z_{10}(\vec{M}_t^{(i)}\cdot\vec{\varepsilon})}\cdot\prod_{j\in[\varsigma]}(g^{\phi_1\phi_3\psi_j})^{\frac{-\partial z_{10}(\vec{M}_t^{(i)}\cdot\vec{\varepsilon})}{w_{\rho_t^\circ(i)}-w_j^{(\mu)}}}\\ T_{i1} &:= g_{10}^{\mathsf{vt}_i-\partial\gamma_0(\vec{M}_t^{(i)}\cdot\vec{\varepsilon})}\cdot g_8^{\varpi_1\varpi_2 r_i+\varpi_3\varpi_4 r'_i}\cdot\prod_{j\in[\varsigma]}(g^{\phi_1^2\phi_3\psi_j})^{\frac{-\partial z_{10}(\vec{M}_t^{(i)}\cdot\vec{\varepsilon})}{w_{\rho_t^\circ(i)}-w_j^{(\mu)}}}\end{aligned}$$

$$\begin{aligned}T_{i2} &:= h^{\mathsf{ut}_i}\cdot H_2\big(e(T_1,Y)^{f'}||\Gamma_t^\circ||\vec{M}_t^{(i)}\big)\cdot T_{i3}^{z_9}\cdot(g^{\phi_1})^{\varpi_1\varpi_2 r_i+\varpi_3\varpi_4 r'_i}\\ &\quad\times\textstyle\prod_{j\in[\varsigma]}(g^{\phi_1^2\phi_3\psi_j})^{\frac{-\partial z_{10}(\vec{M}_t^{(i)}\cdot\vec{\varepsilon})}{w_{\rho_t^\circ(i)}-w_j^{(\mu)}}}\\ V_{i1} &:= \big(g_6^{w_{\rho_t^\circ(i)}}\cdot g_7\big)^{-r_i\varpi_1}\cdot(g^{\phi_2})^{\frac{-(z_6 w_{\rho_t^\circ(i)}+z_7)\partial z_{10}(\vec{M}_t^{(i)}\cdot\vec{\varepsilon})}{2\varpi_2}}\\ &\quad\times\textstyle\prod_{j\in[\varsigma]}\big(g^{\phi_1\phi_3\psi_j}\big)^{\frac{(z_6 w_{\rho_t^\circ(i)}+z_7)\partial z_{10}(\vec{M}_t^{(i)}\cdot\vec{\varepsilon})}{2\varpi_2(w_{\rho_t^\circ(i)}-w_j^{(\mu)})}}\\ &\quad\times\textstyle\prod_{(j,\iota)\in[\varsigma,\varsigma]}\big(g^{(\phi_1\phi_3)^2\psi_\iota/\psi_j}\big)^{\frac{\partial z_{10}(\vec{M}_t^{(i)}\cdot\vec{\varepsilon})}{2\varpi_2(w_{\rho_t^\circ(i)}-w_j^{(\mu)})}}\\ &\quad\times\textstyle\prod_{j\in[\varsigma]}\big(g^{\phi_2^2/\psi_j^2}\big)^{\frac{-\partial z_{10}(\vec{M}_t^{(i)}\cdot\vec{\varepsilon})(w_{\rho_t^\circ(i)}-w_j^{(\mu)})}{2\varpi_2}}\\ &\quad\times\textstyle\prod_{(j,\iota)\in[\varsigma,\varsigma],j\neq\iota}\big(g^{\phi_1\phi_2\phi_3\psi_\iota/\psi_j^2}\big)^{\frac{\partial z_{10}(\vec{M}_t^{(i)}\cdot\vec{\varepsilon})(w_{\rho_t^\circ(i)}-w_j^{(\mu)})}{2\varpi_2(w_{\rho_t^\circ(i)}-w_\iota^{(\mu)})}}\end{aligned}$$

The components V_{i2}, V_{i3}, V_{i4} can be computed as V_{i1} since these components have the term $(g_6^{w_{\rho_t^\circ(i)}}\cdot g_7)$ in common and $\breve{r}_i, \breve{r}'_i$ have the same structure. Now, $\mathscr{C}$ generates the transform key $\mathcal{TK}_{A_d} := \langle A_d, D', D'_0, \{D'_y\}_{y\in A_d}\rangle$, where $\bar{\bar{r}} \xleftarrow{\text{u}} \mathbb{Z}_p^*$ and

$$D' := g^{\alpha'/\tau'_2}\cdot Y^{\bar{\bar{r}}}\cdot(g^{\phi_1})^{(\tau'_3 z_{10}\partial)/\tau'_2},\ D'_0 := g^{\bar{\bar{r}}},\ D'_y := (H_1(y))^{\bar{\bar{r}}}$$

Lastly, $\mathscr{C}$ hands over $token := \langle \mathcal{TD}_{\Gamma_t^\circ}, \mathcal{TK}_{A_d} \rangle$ to $\mathscr{A}$. Note that from Remark 10, it can be seen that it is a properly simulated EMR retrieval request token for (A_d, Γ_t).

- $\mathcal{O}_{search}(\mathcal{CT}, \Gamma_t)$: $\mathscr{A}$ submits a ciphertext $\mathcal{CT}$ and a keyword policy Γ_t obeying the condition $\Gamma_t(W_0^\star) = 0 \wedge \Gamma_t(W_1^\star) = 0$. First, $\mathscr{C}$ performs the EMR storage phase presented in Figure 4.4, and if its output is $\perp$, then $\mathscr{C}$ returns $\perp$ as the response of $\mathcal{O}_{search}$ oracle. Otherwise, it computes $\mathcal{CT}_u := \langle \Delta_e, \Delta_k, \mathsf{tag2}, E_0, \eta, \breve{E}, \mathsf{tag} \rangle$, and carries out the steps: (i) $\mathscr{C}$ chooses a decryption attribute set A_d, (ii) obtains $token := \langle \mathcal{TD}_{\Gamma_t^\circ}, \mathcal{TK}_{A_d} \rangle \leftarrow \mathcal{O}'''_{\mathcal{TG}}(A_d, \Gamma_t)$, (iii) returns the output of $\mathsf{Search}(\mathcal{PP}, \mathcal{CT}_u, token, \mathcal{CSK})$ to $\mathscr{A}$. Note that the output of the Search algorithm is independent the choice of A_d because it uses only $\mathcal{TD}_{\Gamma_t^\circ}$ from $token$.

When this query phase is over, $\mathscr{A}$ submits a message $emr^\star$, an encryption policy $\Gamma_e^\star$ and a signing policy $\Gamma_s^\star$.

(4) Let $\Gamma_s^\star := (\mathbf{M}_s^\star, \rho_s^\star)$, where $\mathbf{M}_s^\star$ is an $\ell_s^\star \times n_s^\star$ matrix, and $\Gamma_e^\star := B_1^\star \vee B_2^\star \vee \cdots \vee B_m^\star$. First $\mathscr{C}$ selects a signing attribute set A_s such that $\Gamma_s^\star(A_s) = 1$, calculates $\vec{a} := (a_1, a_2, \ldots, a_{\ell_s^\star}) \leftarrow \mathsf{Reconstruct}(\mathbf{M}_s^\star, \rho_s^\star, A_s)$ satisfying $\sum_{i \in [\ell_s^\star]} a_i \cdot \vec{M}_s^{\star(i)} = \vec{1}_{n_s^\star}$ and $a_i = 0$ for all $i \in \{i | \rho_s^\star(i) \notin A_s\}$, and samples $(b_1, b_2, \ldots, b_{\ell_s^\star}) \xleftarrow{\mathrm{u}} \{(b_1, b_2, \ldots, b_{\ell_s^\star}) \in \mathbb{Z}_p^{\ell_s^\star} | \sum_{i \in [\ell_s^\star]} b_i \cdot \vec{M}_s^{\star(i)} = \vec{0}_{n_s^\star}\}$. The keyword set is $W_\mu^\star := \{[\mathcal{W}_1 : w_1^{(\mu)}], \ldots, [\mathcal{W}_\varsigma : w_\varsigma^{(\mu)}]\}$. Now, the challenge ciphertext $\mathcal{CT}^\star$ is computed in the following way.

(a) Implicitly define $\theta := \phi_3$,
compute $\delta := H_3\big(e(g^{\phi_3}, g^{\phi_2})^{\alpha'}\big)$, $\mathsf{key} := \mathsf{KDF}\big(e(g^{\phi_3}, g^{\phi_2})^{\alpha'}\big)$,
$k_T := Z^{z_{10}} \cdot e(g^{\phi_3}, g^{z_{10}})^{\gamma_0}$,
$ct := emr^\star \oplus \mathsf{key}$, $\mathsf{tag2}^\star := H_5(\delta || ct)$,

(b) pick $\delta', \delta'', o_2 \xleftarrow{\mathrm{u}} \mathbb{Z}_p^*$,
$\sigma' := g^{\delta'}$, $\sigma'' := h^{\delta''}$, $\mathsf{tag1} := H_4(g_T^{1/\delta} \cdot e(\sigma', Y)^{\delta''})$,
$\mathcal{SK}_{A_s} := \big\langle A_s, S, S_0, \{S_x\}_{x \in A_s} \big\rangle \leftarrow \mathcal{O}_{\mathcal{SKG}}(A_s)$,
$\sigma := S^{1/\delta}(g^{\phi_3})^{o_1 z_1 + z_2} \prod_{i \in [\ell_s^\star]} \big(S_{\rho_s^\star(i)}^{a_i/\delta} \cdot H_1(\rho_s^\star(i))^{o_2 b_i}\big)$,
where $o_1 := H_6(ct || \mathsf{tag1} || \Gamma_e^\star || \Gamma_s^\star || W_\mu^{\star\circ})$,
$\sigma_i := S_0^{a_i/\delta} g^{o_2 b_i}$, for each $i \in [\ell_s^\star]$,
The signature components $\Delta_s^\star := \langle \Gamma_s^\star, \sigma', \sigma'', \mathsf{tag1}, \sigma, \{\sigma_i\}_{i \in [\ell_s^\star]} \rangle$.

(c) pick $\theta_i \xleftarrow{\mathrm{u}} \mathbb{Z}_p^*$, for each $i \in [m]$,
calculate $E := (g^{\phi_3})^{H_3(e(\sigma', Y)^{\delta''})}$, $E_{i1} := g^{\theta_i}$,

$E_{i2} := (g^{\phi_3})^{z}\big(\prod_{y\in B_i^\star} H_1(y)\big)^{\theta_i}$,
The encryption components $\Delta_e^\star := \langle \Gamma_e^\star, ct, E, \{E_{i1}, E_{i2}\}_{i\in[m]}\rangle$.

(d) choose $\pi_{\mathcal{W}_j 1}, \pi_{\mathcal{W}_j 2} \xleftarrow{\text{u}} \mathbb{Z}_p^*$, implicitly define $t_{\mathcal{W}_j} := \psi_j$, for each $j \in [\varsigma]$, and compute
$K_{\mathcal{W}_j 1} := (g^{\psi_j})^{\varpi_1} h_1^{-\pi_{\mathcal{W}_j 1}}, K_{\mathcal{W}_j 2} := h_2^{\pi_{\mathcal{W}_j 1}}$,
$K_{\mathcal{W}_j 3} := (g^{\psi_j})^{\varpi_3} h_3^{-\pi_{\mathcal{W}_j 2}}, K_{\mathcal{W}_j 4} := h_4^{\pi_{\mathcal{W}_j 2}}$,
$L_{\mathcal{W}_j 1} := (g^{\psi_j})^{z_6 w_j^{(\mu)} + z_7} \cdot \prod_{\iota\in[\varsigma]} \big(g^{\phi_2\psi_j/\psi_\iota^2}\big)^{(w_j^{(\mu)} - w_\iota^{(\mu)})} \cdot \prod_{\iota\in[\varsigma],\iota\neq j} g^{\phi_1\phi_3\psi_j/\psi_\iota}$,
$L_{\mathcal{W}_j 2} := L_{\mathcal{W}_j 1} \cdot (g^{\phi_3})^{-z_9}$,
The keyword components,
$\Delta_k^\star := \langle W_\mu^{\star\circ}, k_T, \{K_{\mathcal{W}_j 1}, K_{\mathcal{W}_j 2}, K_{\mathcal{W}_j 3}, K_{\mathcal{W}_j 4}, L_{\mathcal{W}_j 1}, L_{\mathcal{W}_j 2}\}_{j\in[\varsigma]}\rangle$.

(e) pick $\eta \xleftarrow{\text{u}} \mathbb{Z}_p^*$,
set $E_0^\star := (g^{\phi_3})^{\xi z_3 + \eta z_4 + z_5}$, where $\xi := H_7(\Delta_s^\star || \Delta_e^\star || \Delta_k^\star || \mathsf{tag2}^\star)$,
The challenge ciphertext is $\mathcal{CT}^\star := \langle \Delta_s^\star, \Delta_e^\star, \Delta_k^\star, \mathsf{tag2}^\star, E_0^\star, \eta\rangle$.

Then, $\mathscr{C}$ hands over the challenge ciphertext to $\mathscr{A}$.

(5) In this step, $\mathscr{A}$ issues a second series of additional queries as in step (3). Once this query phase is over, $\mathscr{A}$ outputs a guess μ' of μ.

$\mathscr{A}$ wins if $\mu' = \mu$. Therefore, if $\mathscr{A}$ wins, $\mathscr{C}$ will claim that $Z = e(g,g)^{\phi_1\phi_2\phi_3}$; otherwise, $\mathscr{C}$ claims that Z is a random element of $\mathbb{G}_T$.

If $Z = e(g,g)^{\phi_1\phi_2\phi_3}$, then $k_T := Z^{z_{10}} \cdot e(g^{\phi_3}, g^{z_{10}})^{\gamma_0} = h_T^\theta$ and hence $\mathscr{A}$'s view is identical to the original game G_{real}. On the other hand, if Z is a random element then k_T is a random element as well and hence $\mathscr{A}$'s view is identical to the game G_0. Therefore, if $\mathscr{A}$ can distinguish between G_{real} and G_0 with non-negligible advantage, $\mathscr{C}$ has a non-negligible advantage in solving q-2 problem. *(of Claim 4)*

Claim 5. *Under the assumption that DLin problem is hard, there is no PPT Type-1 adversary that can distinguish between the games G_l and G_{l+1} with non-negligible advantage, $l \in \{0, 1, \ldots, \varsigma - 1\}$.*

Proof. Suppose there is a Type-1 adversary $\mathscr{A}$ that distinguishes between the games G_l and G_{l+1} with non-negligible advantage. Then, we construct a challenger $\mathscr{C}$ that solves DLin problem with non-negligible advantage by interacting with $\mathscr{A}$ as follows. Given the DLin problem instance $\langle \Sigma, g, g^{\phi_1}, g^{\phi_2}, g^{\phi_1\phi_3}, g^{\phi_2\phi_4}, Z\rangle$, the task for $\mathscr{C}$ is to determine whether $Z = g^{\phi_3+\phi_4}$ or Z is a random element of $\mathbb{G}$. As in [8], we write DLin problem as $\langle \Sigma, g, g^{\phi_1}, g^{\phi_2}, g^{\phi_1\phi_3}, \mathcal{D}, g^{\psi}\rangle$ for ψ such that $g^\psi = Z$, and consider the task of deciding whether $\mathcal{D} = g^{\phi_2(\psi-\phi_3)}$.

(1) $\mathscr{A}$ submits two challenge keyword sets $W_0^\star := \{[\mathcal{W}_1 : w_1^{(0)}], \ldots, [\mathcal{W}_\varsigma : w_\varsigma^{(0)}]\}$ and $W_1^\star := \{[\mathcal{W}_1 : w_1^{(1)}], \ldots, [\mathcal{W}_\varsigma : w_\varsigma^{(1)}]\}$. These two sets satisfy the conditions $|W_0^\star| = |W_1^\star| = \varsigma$ and $W_0^{\star\circ} = W_1^{\star\circ} = \{\mathcal{W}_1, \ldots, \mathcal{W}_\varsigma\}$ of $\mathsf{Game}^{\mathsf{IND\text{-}CKA}}_{\mathsf{Type\text{-}1}}$.

(2) $\mathscr{C}$ samples $\mu \xleftarrow{\text{u}} \{0,1\}$ and sets $W_\mu^\star := \{[\mathcal{W}_1 : w_1^{(\mu)}], \ldots, [\mathcal{W}_\varsigma : w_\varsigma^{(\mu)}]\}$. Then, it selects $\alpha, \beta, \gamma, \varpi_3, \varpi_4, z, z_i \xleftarrow{\text{u}} \mathbb{Z}_p^*, i \in \{1,2,\ldots,10\} \setminus \{6\}$, sets $g_T := e(g,g)^\alpha$, $h := g^z, g_i := g^{z_i}$ for $i \in \{1,2,\ldots,10\} \setminus \{6,7\}$, $g_6 := (g^{\phi_2})^\gamma, g_7 := g^{z_7}(g^{\phi_2})^{-\gamma w_l^{(\mu)}}$, $h_1 := g^{\phi_2}, h_2 := g^{\phi_1}, h_3 := g^{\varpi_3}, h_4 := g^{\varpi_4}, h_T := e(g, g_{10})^\gamma, Y := h^\beta$. Next, it chooses seven collision-resistant hash functions $\{H_i\}_{i=1}^7$ as mentioned in the construction and sets $\mathcal{KPK} := \langle \Sigma, g_T, g, h, \{g_i\}_{i=1}^{10}, \mathcal{M}, \mathsf{KDF}, \{H_i\}_{i=1}^7 \rangle$, $\mathcal{MK} := g^\alpha, \mathcal{TPK} := \langle h_T, h_1, h_2, h_3, h_4 \rangle, \mathcal{TSK} := \langle \gamma, \varpi_1 := \phi_2, \varpi_2 := \phi_1, \varpi_3, \varpi_4 \rangle$, $\mathcal{CPK} := Y, \mathcal{CSK} := \beta$. Note that $\varpi_1 := \phi_2, \varpi_2 := \phi_1$ are implicitly defined. $\mathscr{C}$ sends the tuple $[\mathcal{PP}, \mathcal{CSK}]$ to $\mathscr{A}$, where $\mathcal{PP} := \langle \mathcal{KPK}, \mathcal{CPK}, \mathcal{TPK} \rangle$.

(3) $\mathscr{C}$ simulates $\mathscr{A}$'s queries as follows.

- $\mathcal{O}_{\mathcal{SKG}}(A_s) : \mathscr{C}$ returns $\mathcal{SK}_{A_s} \leftarrow \mathsf{sKeyGen}(\mathcal{PP}, \mathcal{MK}, A_s)$ since $\mathscr{C}$ knows $\mathcal{MK}$.
- $\mathcal{O}'''_{\mathcal{TG}}(A_d, \Gamma_t)$: $\mathscr{A}$ submits a decryption attribute set A_d and a keyword policy Γ_t with the condition that $\Gamma_t(W_0^\star) = 0 \wedge \Gamma_t(W_1^\star) = 0$. Hence $\Gamma_t(W_\mu^\star) = 0$. Let $\Gamma_t := (\mathbf{M}_t, \rho_t^\circ, \{w_{\rho_t^\circ(i)}\}_{i\in[\ell_t]})$, where $\mathbf{M}_t$ is a matrix of size $\ell_t \times n_t$. firstly, $\mathscr{C}$ calculates $\mathcal{TD}_{\Gamma_t^\circ} := \langle \Gamma_t^\circ, T_1, T_2, \{T_{i1}, T_{i2}, T_{i3}, V_{i1}, V_{i2}, V_{i3}, V_{i4}\}_{i\in[\ell_t]} \rangle$ as described below. Choose $f, f', \tau_1, \tau_3, r_i, r_i', \varepsilon_2, \ldots, \varepsilon_{n_t}, \varepsilon_2', \ldots, \varepsilon_{n_t}' \xleftarrow{\text{u}} \mathbb{Z}_p^*, i \in [\ell_t]$, implicitly define

$$\breve{r}_i := \frac{r_i\gamma(w_{\rho_t^\circ(i)} - w_l^{(\mu)})}{\phi_2\gamma(w_{\rho_t^\circ(i)} - w_l^{(\mu)}) + z_7}, \; \breve{r}_i' := r_i' + \frac{z_7\phi_1 r_i}{\varpi_3\varpi_4\big(\phi_2\gamma(w_{\rho_t^\circ(i)} - w_l^{(\mu)}) + z_7\big)},$$

and set $\mathsf{tt} := e(T_1, Y)^{f'}$, $\vartheta_i := \vec{M}_t^{(i)} \cdot (\gamma \cdot H_3(\mathsf{tt}), \varepsilon_2, \ldots, \varepsilon_{n_t})$, and $\breve{\vartheta}_i := \vec{M}_t^{(i)} \cdot (\tau_3/\tau_1, \varepsilon_2', \ldots, \varepsilon_{n_t}')$,

$$T_1 := g^f, T_2 := h^{f'}, T_{i3} := (g^{\phi_1})^{r_i} g^{\varpi_3\varpi_4 r_i'}, T_{i1} := g_{10}^{\vartheta_i} \cdot T_{i3}^{z_8},$$

$$T_{i2} := h^{\breve{\vartheta}_i} \cdot H_2(e(T_1, Y)^{f'} || \Gamma_t^\circ || \vec{M}_t^{(i)}) \cdot T_{i3}^{z_9},$$

$$V_{i1} := (g^{\phi_2})^{-r_i\gamma(w_{\rho_t^\circ(i)} - w_l^{(\mu)})}, V_{i2} := (g^{\phi_1})^{-r_i\gamma(w_{\rho_t^\circ(i)} - w_l^{(\mu)})},$$

$$V_{i3} := \left(g_6^{w_{\rho_t^\circ(i)}} \cdot g_7\right)^{-r_i'\varpi_3} (g^{\phi_1})^{-z_7 r_i/\varpi_4},$$

$$V_{i4} := \left(g_6^{w_{\rho_t^\circ(i)}} \cdot g_7\right)^{-r_i'\varpi_4} (g^{\phi_1})^{-z_7 r_i/\varpi_3},$$

Next, $\mathscr{C}$ generates the transform key $\mathcal{TK}_{A_d} := \left\langle A_d, D', D_0', \{D_y'\}_{y\in A_d}\right\rangle$, where $\bar{\bar{r}}, \tau_2 \xleftarrow{u} \mathbb{Z}_p^*$ and $D' := g^{\alpha/\tau_2} Y^{\bar{\bar{r}}} h^{\tau_3/\tau_2}, D_0' := g^{\bar{\bar{r}}}, D_y' := (H_1(y))^{\bar{\bar{r}}}$. Finally, $\mathscr{C}$ sends $token := \langle \mathcal{TD}_{\Gamma_t^\circ}, \mathcal{TK}_{A_d}\rangle$ to $\mathscr{A}$. Note that from Remark 10, it can be seen that it is a properly simulated EMR retrieval request token for (A_d, Γ_t).

- $\mathcal{O}_{search}(\mathcal{CT}, \Gamma_t)$: The simulation is similar to that of Claim 4.

When $\mathscr{A}$ decides that this query phase is completed, it outputs a message $emr^\star$, an encryption policy $\Gamma_e^\star$ and a signing policy $\Gamma_s^\star$.

(4) To compute the challenge ciphertext $\mathcal{CT}^\star$ of the message $emr^\star$ for $\Gamma_s^\star := (\mathbf{M}_s^\star, \rho_s^\star)$, where $\mathbf{M}_s^\star$ is an $\ell_s^\star \times n_s^\star$ matrix, $\Gamma_e^\star := B_1^\star \vee B_2^\star \vee \cdots \vee B_m^\star$, and the keyword set $W_\mu^\star := \{[\mathcal{W}_1 : w_1^{(\mu)}], \ldots, [\mathcal{W}_\varsigma : w_\varsigma^{(\mu)}]\}$, $\mathscr{C}$ calculates the components of $\Delta_k := \langle W_\mu^{\star\circ}, k_T, \{K_{\mathcal{W}_j 1}, K_{\mathcal{W}_j 2}, K_{\mathcal{W}_j 3}, K_{\mathcal{W}_j 4}, L_{\mathcal{W}_j 1}, L_{\mathcal{W}_j 2}\}_{j\in[\varsigma]}\rangle$ in the following manner.

For $j \in [\varsigma]$ and $j \neq l$, $\mathscr{C}$ picks $\theta, t_{\mathcal{W}_j}, \pi_{\mathcal{W}_j 1}, \pi_{\mathcal{W}_j 2} \xleftarrow{u} \mathbb{Z}_p^*$, and sets $k_T := h_T^\theta$,

$$K_{\mathcal{W}_j 1} := h_1^{t_{\mathcal{W}_j} - \pi_{\mathcal{W}_j 1}}, K_{\mathcal{W}_j 2} := h_2^{\pi_{\mathcal{W}_j 1}}, K_{\mathcal{W}_j 3} := h_3^{t_{\mathcal{W}_j} - \pi_{\mathcal{W}_j 2}}, K_{\mathcal{W}_j 4} := h_4^{\pi_{\mathcal{W}_j 2}},$$

$$L_{\mathcal{W}_j 1} := (g_6^{w_j^{(\mu)}} g_7)^{t_{\mathcal{W}_j}} g_8^{-\theta}, L_{\mathcal{W}_j 2} := (g_6^{w_j^{(\mu)}} g_7)^{t_{\mathcal{W}_j}} g_9^{-\theta}.$$

For $j = l$, $\mathscr{C}$ implicitly defines $t_{\mathcal{W}_l} := \psi, \pi_{\mathcal{W}_l 1} := \phi_3$, chooses $\pi_{\mathcal{W}_l 2} \xleftarrow{u} \mathbb{Z}_p^*$, and sets

$$K_{\mathcal{W}_l 1} := \mathcal{D}, K_{\mathcal{W}_l 2} := g^{\phi_1\phi_3}, K_{\mathcal{W}_l 3} := (g^\psi)^{\varpi_3} h_3^{-\pi_{\mathcal{W}_l 2}}, K_{\mathcal{W}_l 4} := h_4^{\pi_{\mathcal{W}_l 2}},$$

$$L_{\mathcal{W}_l 1} := (g^\psi)^{z_7} g_8^{-\theta}, L_{\mathcal{W}_l 2} := (g^\psi)^{z_7} g_9^{-\theta}.$$

$\mathscr{C}$ computes the other components of $\mathcal{CT}^\star$ as in the construction of MediCare. $\mathscr{A}$ receives this challenge ciphertext $\mathcal{CT}^\star$.

(5) Again, $\mathscr{A}$ issues a second series of additional queries as in step (3). Once this query phase is over, $\mathscr{A}$ outputs a guess μ' of μ.

$\mathscr{A}$ wins if $\mu' = \mu$. Hence, if $\mathscr{A}$ wins, $\mathscr{C}$ will claim that $\mathcal{D} = g^{\phi_2(\psi-\phi_3)}$; otherwise, $\mathscr{C}$ claims that $\mathcal{D}$ is a random element of $\mathbb{G}$.

If $\mathcal{D} = g^{\phi_2(\psi-\phi_3)}$, then $K_{\mathcal{W}_l 1} = \mathcal{D} = g^{\phi_2(\psi-\phi_3)} = h_1^{t_{\mathcal{W}_l} - \pi_{\mathcal{W}_l 1}}, K_{\mathcal{W}_l 2} = g^{\phi_1\phi_3} = h_2^{\pi_{\mathcal{W}_l 1}}$ and hence $\mathscr{A}$'s view is identical to G_l. On the other hand, if $\mathcal{D}$ is a random element of $\mathbb{G}$, then $\mathscr{A}$'s view is identical to G_{l+1}. Therefore, if $\mathscr{A}$ can distinguish between G_l and G_{l+1} with non-negligible advantage, $\mathscr{C}$ has a non-negligible advantage in solving DLin problem. (*of Claim 5*)

Claim 6. *There is no PPT Type-1 adversary that can distinguish between the games* G_l *and* G_{l+1}, $l \in \{\varsigma, \varsigma+1, \ldots, 2\varsigma-1\}$, *with non-negligible advantage, under the assumption that DLin problem is hard.*

Proof. The proof is almost identically to that of Claim 5, except where the simulation is done over h_3 and h_4 instead of h_1 and h_2. (*of Claim 6*)

This completes the proof of Lemma 7. □

Proof of Lemma 8

Proof. Assume there is a PPT Type-2 adversary $\mathscr{A}$ that breaks the IND-CKA security (modeled as a game $\mathsf{Game}^{\mathsf{IND\text{-}CKA}}_{\mathsf{Type\text{-}2}}$ in Section 4.2.2) of our MediCare with non-negligible advantage. Then we build a challenger $\mathscr{C}$ that can solve DBDH problem with non-negligible advantage, by interacting with $\mathscr{A}$. Given the DBDH problem instance $\langle \Sigma, g, G_1, G_2, G_3, Z \rangle$, where $G_1 := g^{\phi_1}, G_2 := g^{\phi_2}, G_3 := g^{\phi_3}$ (note that ϕ_1, ϕ_2, ϕ_3 are unknown to $\mathscr{C}$), the task for $\mathscr{C}$ is to determine whether $Z = e(g,g)^{\phi_1\phi_2\phi_3}$ or Z is a random element of $\mathbb{G}_T$.

(1) $\mathscr{C}$ picks $\alpha, z \xleftarrow{\text{u}} \mathbb{Z}_p^*$ and sets $g_T := e(g,g)^\alpha, h := g^z$. Next, it chooses $g_1, g_2, \ldots, g_{10} \xleftarrow{\text{u}} \mathbb{G}$, and seven collision-resistant hash functions $\{H_i\}_{i=1}^7$ (as described in the construction). Now, $\mathscr{C}$ sets $\mathcal{KPK} := \langle \Sigma, g_T, g, h, \{g_i\}_{i=1}^{10}, \mathcal{M}, \mathsf{KDF}, \{H_i\}_{i=1}^7 \rangle$, where $\mathcal{M} := \{0,1\}^{\ell_{pt}}$ is the message space and KDF is the key derivation function, and $\mathcal{MK} := g^\alpha$. Next, $\mathscr{C}$ selects $\gamma, \varpi_1, \varpi_2, \varpi_3, \varpi_4 \xleftarrow{\text{u}} \mathbb{Z}_p^*$, computes $h_1 := g^{\varpi_1}, h_2 := g^{\varpi_2}, h_3 := g^{\varpi_3}, h_4 := g^{\varpi_4}, h_T := e(g, g_{10})^\gamma, Y := G_1^z$ and sets $\mathcal{TPK} := \langle h_T, h_1, h_2, h_3, h_4 \rangle, \mathcal{TSK} := \langle \gamma, \varpi_1, \varpi_2, \varpi_3, \varpi_4 \rangle$, $\mathcal{CPK} := Y$, and implicitly sets $\mathcal{CSK} := \phi_1$. $\mathscr{C}$ sends the tuple $\mathcal{PP} := \langle \mathcal{KPK}, \mathcal{CPK}, \mathcal{TPK} \rangle$ to $\mathscr{A}$.

(2) In this phase, firstly $\mathscr{A}$ queries signing key generation oracle $\mathcal{O}_{\mathcal{SKG}}(A_s)$, token generation oracle $\mathcal{O}'_{\mathcal{TG}}(A_d, \Gamma_t)$, and search oracle $\mathcal{O}'_{search}(\mathcal{CT}, \Gamma_t)$. $\mathscr{C}$ answers as described subsequently.

- $\mathcal{O}_{\mathcal{SKG}}(A_s)$: $\mathscr{C}$ returns $\mathcal{SK}_{A_s} \leftarrow \mathsf{sKeyGen}(\mathcal{PP}, \mathcal{MK}, A_s)$ since $\mathscr{C}$ knows $\mathcal{MK}$.

- $\mathcal{O}'_{\mathcal{TG}}(A_d, \Gamma_t) : \mathscr{C}$ generates the trapdoor $\widetilde{\mathcal{TD}}_{\Gamma_t} := \langle \Gamma_t, T_1, T_2, \{T_{i1}, T'_{i2}, T_{i3}, V_{i1}, V_{i2}, V_{i3}, V_{i4}\}_{i\in[\ell_t]}\rangle$ as follows. Here $\Gamma_t := (\mathbf{M}_t, \rho_t^\circ, \{w_{\rho_t^\circ(i)}\}_{i\in[\ell_t]})$. Choose $\breve{r}_i, \breve{r}'_i, \varepsilon_2, \ldots, \varepsilon_{n_t} \xleftarrow{u} \mathbb{Z}_p^*$ and compute $T_1 := G_2, T_2 := G_3^z, \mathsf{tt} := Z^z$, $\vartheta_i := \vec{M}_t^{(i)} \cdot (\gamma H_3(\mathsf{tt}), \varepsilon_2, \ldots, \varepsilon_{n_t})$, $T_{i1} := g_{10}^{\vartheta_i} \cdot g_8^{\varpi_1\varpi_2\breve{r}_i + \varpi_3\varpi_4\breve{r}'_i}$, $T'_{i2} := H_2(\mathsf{tt}||\Gamma_t^\circ||\vec{M}_t^{(i)}) \cdot g_9^{\varpi_1\varpi_2\breve{r}_i + \varpi_3\varpi_4\breve{r}'_i}$, $T_{i3} := g^{\varpi_1\varpi_2\breve{r}_i + \varpi_3\varpi_4\breve{r}'_i}$, $V_{i1} := (g_6^{w_{\rho_t^\circ(i)}} g_7)^{-\breve{r}_i\varpi_1}$, $V_{i2} := (g_6^{w_{\rho_t^\circ(i)}} g_7)^{-\breve{r}_i\varpi_2}, V_{i3} := (g_6^{w_{\rho_t^\circ(i)}} g_7)^{-\breve{r}'_i\varpi_3}, V_{i4} := (g_6^{w_{\rho_t^\circ(i)}} g_7)^{-\breve{r}'_i\varpi_4}$, Since $\mathscr{C}$ knows $\mathcal{MK}$, it computes the decryption key $\mathcal{DK}_{A_d} \leftarrow \mathsf{dKeyGen}(\mathcal{PP}, \mathcal{MK}, A_d)$ and sends the tuple $[\mathcal{DK}_{A_d}, \widetilde{\mathcal{TD}}_{\Gamma_t}]$ to $\mathscr{A}$.
- $\mathcal{O}'_{search}(\mathcal{CT}, \Gamma_t) : \mathscr{C}$ performs the EMR storage phase (given in Figure 4.4), and if its output is $\perp$, then $\mathscr{C}$ returns $\perp$ as the response of $\mathcal{O}'_{search}$ oracle. Otherwise, it computes $\mathcal{CT}_u := \langle \Delta_e, \Delta_k, \mathsf{tag2}, E_0, \eta, \breve{E}, \mathsf{tag}\rangle$. Next, $\mathscr{C}$ obtains the trapdoor $\widetilde{\mathcal{TD}}_{\Gamma_t}$ as in the simulation of $\mathcal{O}'_{\mathcal{TG}}$ and then computes $\mathcal{TD}_{\Gamma_t^\circ}$, and finally returns the output of Search algorithm to $\mathscr{A}$.

When this query phase is over, $\mathscr{A}$ sends to $\mathscr{C}$ a message $emr^\star$, an encryption policy $\Gamma_e^\star$, a signing policy $\Gamma_s^\star$ and two keyword sets $W_0^\star := \{[\mathcal{W}_1 : w_1^{(0)}], \ldots, [\mathcal{W}_\varsigma : w_\varsigma^{(0)}]\}$ and $W_1^\star := \{[\mathcal{W}_1 : w_1^{(1)}], \ldots, [\mathcal{W}_\varsigma : w_\varsigma^{(1)}]\}$.

(3) $\mathscr{C}$ selects $\mu \xleftarrow{u} \{0,1\}$, and a singing attribute set A_s such that $\Gamma_s^\star(A_s) = 1$. Next, it obtains $\mathcal{CT}^\star \leftarrow \mathsf{Signcrypt}(\mathcal{PP}, \mathcal{SK}_{A_s}, \Gamma_s^\star, \Gamma_e^\star, W_\mu^\star, emr^\star)$, where $\mathcal{SK}_{A_s} \leftarrow \mathsf{sKeyGen}(\mathcal{PP}, \mathcal{MK}, A_s)$, and sends the challenge ciphertext $\mathcal{CT}^\star$ to $\mathscr{A}$. This is possible because $\mathscr{C}$ has the knowledge of the system master secret $\mathcal{MK}$ and TGA's secret key $\mathcal{TSK}$.

(4) $\mathscr{A}$ now issues additional queries like step (2) with the obvious restriction that $\mathscr{A}$ cannot query search oracle with the input $(\mathcal{CT}^\star, \Gamma_t)$ such that $\Gamma_t(W_\mu^\star) = 1$. $\mathscr{C}$ responds as in step (2). When this query phase is over, $\mathscr{A}$ outputs a guess μ' of μ.

$\mathscr{A}$ wins the game if $\mu' = \mu$. Hence, if $\mathscr{A}$ wins, $\mathscr{C}$ will claim that $Z = e(g,g)^{\phi_1\phi_2\phi_3}$; otherwise, $\mathscr{C}$ claims that Z is a random element of $\mathbb{G}_T$.

If $Z = e(g,g)^{\phi_1\phi_2\phi_3}$, the challenge ciphertext $\mathcal{CT}^\star$ is properly simulated. If Z is randomly chosen from $\mathbb{G}_T$, then $\mathcal{CT}^\star$ is independent of μ in $\mathscr{A}$'s view; resulting in $\mathscr{A}$'s advantage is 0. As a result, if $\mathscr{A}$ has a non-negligible advantage in winning the game, $\mathscr{C}$ has a non-negligible advantage in solving DBDH problem. □