Towards Provably Secure Efficiently Searchable Encryption

Saeed Sedghi

Composition of the Graduation Committee:

Prof. Dr. Ir.	A.J. Mouthaan	Universiteit Twente
Prof. Dr.	W. Jonker	Universiteit Twente
Prof. Dr.	P.H. Hartel	Universiteit Twente
Dr.	S. Nikova	Universiteit Twente and Katholieke Universiteit Leuven
Dr.	M. Abdalla	Ecole Normale Superieure
Prof. Dr.	D. Pavlović	Royal Holloway, University of London and Universiteit Twente
Prof. Dr.	M. Petkovic	Technische Universiteit Eindhoven
Prof. Dr.	J.C. van de Pol	Universiteit Twente
Dr. Ir.	B. Schoenmakers	Technische Universiteit Eindhoven

This research is supported by the SEDAN project, funded by the Sentinels program of the Technology Foundation STW, applied science division of NWO and the technology programme of the Ministry of Economic Affairs under project number EIT.7630.

CTIT

CTIT Ph.D. Thesis Series No. 12-219
Centre for Telematics and Information Technology
P.O. Box 217, 7500 AE, Enschede, The Netherlands.

IPA: 2012-5
The work in this thesis has been carried out under the auspices of the research school IPA
(Institute for Programming research and Algorithms).

ISBN: 978-90-365-3333-1
ISSN: 1381-3617 (CTIT Ph.D. thesis Series No. 12-219)
DOI: 10.3990/1.9789036533331
http://dx.doi.org/10.3990/1.9789036533331

Typeset with LaTeX. Printed by IPSKAMP Print Service.

To my daughter, Elena

TOWARDS PROVABLY SECURE EFFICIENTLY SEARCHABLE ENCRYPTION

DISSERTATION

to obtain
the degree of doctor at the University of Twente,
on the authority of the rector magnificus,
prof. dr. H. Brinksma,
on account of the decision of the graduation committee,
to be publicly defended
on Friday, the 17th of February 2012 at 16.45 hrs

by

Saeed Sedghi

born on 2nd of September 1981,
in Mashhad, Iran

The dissertation is approved by:

Prof. Dr.	W. Jonker	(promotor)
Prof. Dr.	P.H. Hartel	(promotor)
Dr.	S. Nikova	(assistant-promotor)

Abstract

Traditional encryption systems are designed in such a way that either the whole data is decrypted, if the encryption and decryption keys match, or nothing is decrypted otherwise. However, there are applications that require a more flexible encryption system which supports decrypting data partially. Searchable encryption is a technique that provides functionalities to decrypt data partially by searching encrypted data.

In searchable encryption, each message of data is associated with a set of keywords. Searchable encryption transforms both, the message and the associated keywords, to an encrypted form, in such a way that the encrypted keywords can be queried later. This allows a client to retrieve or decrypt only the messages of the data that contain a particular keyword without decrypting the data.

Searchable encryption can be based on either symmetric key or public key cryptography. In the symmetric key setting, only the client who stores the data on the server can search the encrypted data. This setting is appropriate for situations where the client stores encrypted data on an honest but curious server, in such a way the the encrypted data can be retrieved selectively. Using symmetric key searchable encryption, the server learns as little information as possible after the storage and the search. In public key searchable encryption, anyone can encrypt data using a public key, while only the owner of the corresponding private key can query encrypted data. Public key searchable encryption allows a client to delegate a decryption key to other users, in such a way that the delegated decryption key can decrypt parts of data only.

The main aspects of searchable encryption are security and efficiency. The efficiency of a scheme is evaluated by the complexity of the scheme. The security of a scheme shows the ability of the scheme in hiding both, the message and the associated keywords, from adversaries. To define the security formally, security models are proposed where each model defines certain computational resources and restrictions for the adversary. Since security is never free, there is a trade off between efficiency and security. Searchable encryption schemes that achieve security in a security model with a more powerful adversary have a higher complexity. The best trade off is achieved when the scheme achieves certain level of security with the lowest possible complexity.

The main contributions of this thesis are the efficient and provably secure searchable encryption schemes which have a lower complexity compared to existing schemes. Our focus in this thesis is the complexity of the search, which is the main

functionality of searchable encryption. In this thesis we propose:

- A searchable encryption scheme in the symmetric key setting which is secure in the symmetric key searchable encryption model. This security model is the only security model proposed in the symmetric key setting. Our scheme, called SES, has a lower complexity for the search compared to existing symmetric key searchable encryption schemes.

- A public key searchable encryption scheme which is secure in the random oracle model. A ransom oracle is a function that maps an input value to a true random output value. In the random oracle model anyone including the adversary has access to a random oracle. Our scheme, called SEPE, allows searching and enforcing an access control policy, while revealing as little as possible about the data and the policy. The SEPE scheme has a lower complexity to perform the search and enforce an access control policy compared to existing schemes.

- A public key searchable encryption scheme which is secure in the selective security model. The adversary in the selective security model must inform in advance of the attack which keyword is intended to be attacked. Our scheme, called SEPS, supports wildcards in the queried keyword. The SEPS scheme is more efficient that existing schemes which allow searching keywords with wildcards on encrypted data.

- A public key searchable encryption scheme which is fully secure. The full security model is the strongest proposed security model. Our scheme, called SEPF, has a lower search complexity compared to existing fully secure schemes.

Acknowledgement

Writing the acknowledgement section is one of the most enjoyable parts of the thesis. This section makes me remember all the great times, experiences, conversations and activities that I had with my friends during the PhD education. It is always a pleasure to thank all the friends who supported me to accomplish the thesis.

I am greatly indebted to my supervisors, Pieter, Willem, Svetla and Jeroen. Without their help completing this thesis would not have been possible. Willem, who was my first promoter, helped me on how to see and approach problems. While Willem's agenda was most of the times overbooked, he always found times to have a weekly meeting, which shows his commitment to the project. Willem is also amazingly fast in grasping the idea of the work from each presentation, which was followed by his nice comments. While Pieter was my second promoter, he has done almost all parts of the work to teach me how to do the research. Pieter is amazingly patient in teaching students on how to perform independent research. I feel so lucky to be his student. Svetla was my daily supervisor after the second year of my work. Svetla is amazingly hard working. She did all the difficult parts of checking the details of my work, my English writing and presentations. Jeroen was my daily supervisor in the first year of my work. Jeroen is amazingly intelligent. Every meeting with him was followed by plenty of ideas and a deeper understanding of the work.

I am grateful to Peter van Liesdonk, my friend and my colleague in TU/e, for all his help. I learned many things about cryptography from Peter. He was always enthusiastic to hear my ideas, which was followed by his useful comments. While traveling from Enschede to Eindhoven is not usually convenient, having a meeting with him always inspired me to have a trip to Eindhoven frequently.

I would like to thank Marlous, Nicole, Thelma and specially Nienke and Bertine, who worked as the secretary of the group in different periods of time. Their help in my financial, business and official work saved me plenty of time. I thank Ruth who helped me improve my English writing.

I was lucky to share my office with Arjan, Ayse, Cristoph, Giorgi, Ileana, Luan, Marcin, Mohsen, Stefan, and Trajce. I had lots of enjoyable times with them. I had also great times with other members of the group, Andre, Begul, Damiano, Dina, Dusko, Emmanuele, Frank, Michael, Richard, Qiang, Sandro, Wolter, and Zheng. I would also like to thank Qiang for his supports and nice comments on my work. I am thankful to the committee members of my defense for their comments to improve

my thesis.

I had great times with all the Iranian friends we met here. We had so enjoyable times together in such a way that we were feeling home always with them. I would like to express my gratitude to my parents for all their support and encouragement. I thank also my sisters, my brother and my family-in-law for their motivation and continual interest in the progress of my studies.

Last and foremost, I owe special thank to my wife Sara. It is hard to express in words my gratitude for All the enjoyable moments living with you.

Contents

Chapter 1

Introduction

In a traditional encryption system, data is encrypted using a predefined encryption key, such that the data can be decrypted with the corresponding decryption key. This property makes these systems coarse-grained, because either the whole data is decrypted, if the encryption key and the decryption key match, or nothing is decrypted otherwise. However, there are applications that require a fine-grained encryption system which supports searching in encrypted data to decrypt or retrieve data selectively. We consider two scenarios. The first is typical for the single user setting, while the second is appropriate when more than one user is involved.

- **Scenario 1 (single-user setting)**. Imagine that Alice wishes to store her medical records digitally such that they are available to her at any time and anywhere. Alice could store all the records in a local memory device and keep the device always with her. However, the device can be damaged, lost or stolen. Hence, it would be more convenient for Alice to store the medical records on a personal health record (PHR) server, such that she can retrieve the records any time and anywhere. To protect the confidentiality of the records, which contain sensitive information, Alice encrypts them prior to storage on the server. However, if Alice uses a traditional *symmetric key* encryption scheme, retrieving parts of the records selectively would be a problem. To search the records, Alice has to either send her decryption key to the server, such that the server decrypts the records to find the desired parts, or she has to retrieve all the records to find the desired parts manually. These solutions are neither secure nor efficient. Alice can also append some metadata, in such a way that she searches the metadata instead of the record directly. However, since the metadata is dependant on the record, some information about the record is revealed to the server.
- **Scenario 2 (multi-user setting)**. Imagine that Bob wants his secretary, Carol, to reply, on his behalf, to his e-mails only if they contain the keyword "Job" in the subject line. If Bob's e-mails are in plaintext, Carol can simply

check the subject line of each e-mail and take action. However, Bob wishes to use an encryption scheme to preserve the confidentiality of his e-mails. Bob could use a traditional *public key* encryption scheme as follows. Either Bob has to reveal his decryption key to Carol, or Bob has to decrypt his e-mails by himself and send only the e-mails with subject "Job" to Carol. The first solution compromises the confidentiality of all e-mails, and the second solution is not efficient.

Scenarios such as those sketched above require an encryption technique that allows searching in encrypted data while making a good compromise between security and efficiency. Therefore, the problem is how can we search in encrypted data with the best trade off between efficiency and security? The focus of this thesis is to provide solutions to this problem using searchable encryption.

1.1 Searchable Encryption

Searchable encryption is a technique that provides functionalities to search encrypted data without requiring the decryption key. Let the data be a set of messages. To support data encryption, such that only particular messages can be decrypted later, each message is associated with a set of keywords. Searchable encryption transforms the message, and the keywords associated with the message to a searchable ciphertext which can be queried using a trapdoor. A trapdoor is a decryption key which is also associated with a set of keywords. The message can be decrypted if and only if the keywords of the trapdoor match the keywords associated with the message.

In Scenario 1, Alice wants to store her medical records on a server in encrypted form in such a way that she can retrieve them selectively. To each record, Alice associates a set of keywords (e.g. the date and the type of the disease). Using searchable encryption, Alice transforms the keywords, which are associated with the record, to a searchable ciphertext. The record itself is encrypted using any standard encryption scheme. Alice then stores the encrypted record and the searchable ciphertext on the server. Now, assume that Alice wants to retrieve only records that contain the keyword "flu". Alice computes a trapdoor using the keyword "flu" and sends the trapdoor to the server. Given the trapdoor, the server checks for each searchable ciphertext, whether it matches the trapdoor. If there is a match, the server sends the encrypted record to Alice who decrypts the record using her secret key. In this case, the server learns which encrypted records Alice requires, but learns nothing about the contents of the records.

In Scenario 2, using searchable encryption, Bob computes a trapdoor which is associated with the keyword "Job", and gives it to Carol. This trapdoor enables Carol to decrypt an e-mail only if its subject line contains the keyword "Job". Now, assume that Dave wants to send an e-mail to Bob. Dave transforms the e-mail and the keywords of the subject line to a searchable ciphertext, and sends it to Bob. Given the searchable ciphertext, Carol can decrypt the e-mail using the trapdoor, but only if the e-mail contains the keyword "Job". In this case, the e-mails that

contain the keyword "Job" are revealed to Carol, as required, but she learns nothing about the other e-mails.

Searchable encryption schemes can be based on either symmetric key or public key cryptography. Table 1.1 summarizes the differences between public key and symmetric key searchable encryption from the perspective of the construction of a searchable ciphertext, the type of application, the type of query, and the performance. In the symmetric key setting, only the owner of the secret key can create the searchable ciphertext. However, public key searchable encryption allows anybody to create the searchable ciphertext using some public parameters. Since sharing a secret key increases the risk of exposure, symmetric key searchable encryption schemes are most suitable for single-user settings. Public key searchable encryption is appropriate for multi-user settings, where any user can encrypt but only one user can search. In the public key setting, the owner of a secret key can query the searchable ciphertext, using a trapdoor, either to decrypt messages selectively, or to search whether a keyword occurs. Whereas, in the symmetric setting, the owner of the secret key can only query to search for a keyword. Symmetric key searchable encryption is, in general, faster than public key searchable encryption.

	Symmetric key searchable encryption	**Public key searchable encryption**
Construction of searchable ciphertext	Created by a secret key	Created by public parameters
Key Management	Single-user settings	Multi-user settings
Functionality	Searching for a keyword	Searching for a keyword and Partially decrypting data
Performance	More efficient	Less efficient

Table 1.1: *Comparison between public key and symmetric key searchable encryption*

Security and efficiency are the main aspects of searchable encryption schemes. To be precise about what we mean by security and efficiency, we discuss them in the following sections.

Security

Informally, security in searchable encryption shows the ability of a scheme to hide a message and its associated keywords from adversaries. For a scheme to be provably secure, it must be formally shown that the message and the keywords are hidden from probabilistic polynomial time (PPT) adversaries who have access to certain computational resources. The computational resources, that the adversary has access to, are defined in a security model. The security model, which also defines how the

adversary interacts with users, shows how powerful the adversary is.

Various security models have been proposed. From a security point of view, models with lower restrictions for the adversary are preferred because these are more realistic. However, the increased security usually causes a loss of efficiency. In fact, one needs to choose a security model based on the application and the cost that one is prepared to pay. There is a variety of security models thus giving flexibility in deciding about the trade-off between the efficiency cost and the level of security.

Here, we briefly explain the security models which we consider in this thesis. We will give a formal definition of each model in Chapter 2.

Symmetric key setting: The most widely used model for searchable encryption in the symmetric key setting is called the *symmetric key searchable encryption model* [38]. For a scheme to be secure in this model, it must be shown that the searchable ciphertext and the trapdoor do not reveal any information to the adversary except the access pattern. The access pattern is the outcome of a search result which shows which searchable ciphertexts match a trapdoor.

Public key setting: Security models in the public key setting allow anyone including the adversary to obtain a trapdoor for each keyword that is queried. In public key settings, for a scheme to be provably secure, it must be shown that a searchable ciphertext which does not match any trapdoor query, reveals nothing to the adversary. The most used sub-models in the public key setting, which we consider in this these, are as follows:

- The *random oracle model*, gives the adversary access to all the functions [6] used in the scheme to construct the searchable ciphertext and the trapdoor. These functions, which are called random oracles, are true random functions that map an input value to a true random output value.
- The *standard model*, where the adversary does not have access to any random oracle. The standard model is thus a stronger model than the random oracle model. There are two prominent sub-models in the standard model:
  - The *selective security model*, where the adversary has to publicly announce which keyword is intended to be attacked [16].
  - The *full security model*, where the adversary is free to attack any keyword. This is a stronger model than the selectively secure model.

The first parameter to choose an appropriate security model for a searchable encryption scheme is the setting in which the scheme is proposed. If the scheme is proposed in the symmetric key setting, the symmetric key security model is used. If the scheme is proposed in the public key setting, one of the sub models mentioned above should be used. After designing a primary construction for the scheme, it must be checked whether the scheme can be proven to be secure in the chosen model. If the proof cannot be accomplished, the construction of the scheme should be adjusted, in such a way that the construction is more randomized. Then, the security proof

must be checked again with the new construction. This cycle must be continued until a security proof is found. Therefore, the weaker the security model requires the less randomization of the construction, which makes schemes in weaker models more efficient.

Searchable encryption schemes in the random oracle model are more efficient than in the standard model. However, random oracles do not exist in practice which makes this model weaker than the standard model. The random oracle is usually deployed in the schemes that make the first step towards addressing a problem (e.g. [11], [25]). In such schemes, to avoid complications, the random oracle is used. In the standard model, we distinguish between selective security and full security. Selective security curb the adversary's flexibility in attacking keywords. Full security has a higher security, which can be used when the keywords are very sensitive. However, a scheme which is secure in this model is more costly than secure schemes in other models.

Efficiency

To evaluate the efficiency of a searchable encryption scheme we consider the following complexity aspects:

- The complexity to create the searchable ciphertext, the trapdoor and to perform the search (Computational complexity).
- The complexity to send the trapdoor and the searchable ciphertext from the client to the server (Communication complexity).
- The complexity to store the public key, secret key, searchable ciphertext and the trapdoor (Storage complexity).

The complexity to send encrypted messages, after performing the search, from the server to the client is not considered as a complexity aspect of searchable encryption, since the size of the results will only depend on the size of the encrypted messages stored, which is independent of the searchable encryption.

In general we are interested in schemes with the lowest complexity possible. However, in practical situations, reducing all complexity aspects is not possible. Indeed, we need to prioritize the complexity aspects with respect to the application. For instance, if searchable encryption is used for retrieving encrypted data from a server that offers cheap storage, the storage complexity is not crucial. However, if the number of searchable ciphertexts stored on the server is large, searching the searchable ciphertexts will be expensive. Therefore, the complexity of the search might be more crucial than the complexity of the memory.

In Scenario 1, Alice is interested in carrying devices with limited memory to store the master secret key. Hence, in case Alice uses a broadband network connection and does not search the records frequently, the storage complexity and the computational searchable ciphertext complexity are more important than the trapdoor complexity. On the other hand, it is not only Alice who stores her records on the PHR server.

Indeed, there are a large number of users who want to use the PHR system. In this case, the server receives a large number of queries at any time, which makes the complexity of the search crucial for the server. Therefore, in this scenario the search complexity as well as the storage and searchable ciphertext complexity are more important than the trapdoor complexity.

In Scenario 2, if a broad band internet connection, and devices with large storage capacity are used, the communication complexity and the storage complexity are therefore not critical. However, creating a searchable ciphertext should be efficient as well as searching. If searching encrypted e-mails also takes a long time, Carol might not be interested in using searchable encryption. Therefore, the computational complexity is more important than the communication and the storage complexity.

1.2 Research question

Our goal in this thesis is to propose searchable encryption schemes, which are provably secure in the appropriate security model, and which have a lower complexity than existing schemes. The research question that this thesis addresses is therefore:

Can we construct provably secure searchable encryption schemes with a complexity as close as possible to plaintext search?

We explore answers to the research question in two settings:

1. The symmetric key setting, which is appropriate for single-user applications. Symmetric key searchable encryption in general has lower complexity than public key searchable encryption.

2. The public key setting, which is used for multi-user applications. Several users may encrypt but only one party creates the trapdoor. In the public key setting we consider searchable encryption in:

    - The random oracle model which is not practical but has less complications compared to the standard model. This model is usually used for the schemes that make the first step to address a problem.
    - The standard model, which offers higher security but at the cost of more complexity.

1.3 Overview of the thesis

The main contributions of this thesis are the efficient and provably secure searchable encryption schemes which are formally analyzed in the security models mentioned earlier. The tree structure showing the contribution of the thesis in relation to previous prominent schemes is illustrated in Figure 1.1. The thesis is organized as follows:

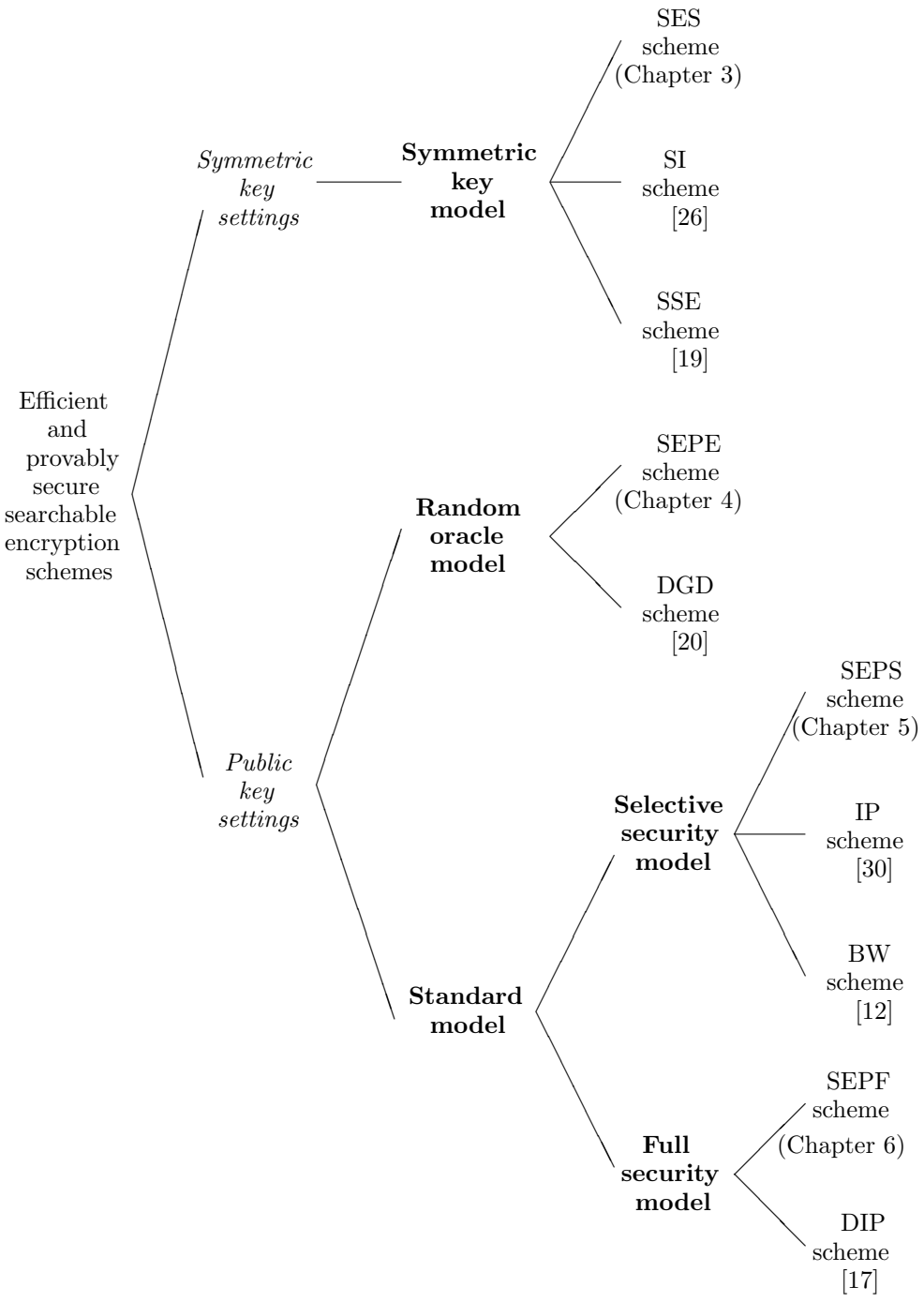

Figure 1.1: *The contributions of the thesis in relation to previous prominent schemes.*

- **Chapter 2**: In this chapter we present formal definitions for searchable encryption, the security models that we have informally introduced in this chapter.

Our solutions to the research question are described in chapters 3, 4, 5, and 6. In each chapter, we propose an efficient searchable encryption scheme which is secure in an appropriate model:

- **Chapter 3**: We first review the state of the art in symmetric key searchable encryption schemes. Then, we propose a searchable encryption scheme, called SES, which is provably secure in the symmetric key model. In existing schemes, the computational complexity of the search is linear in the total number of metadata items stored on the server. In the SES scheme, the computational complexity to search for a keyword is linear in the number of metadata items that contain the query keyword. Two variants of SES are proposed that differ in the computational complexity and the communication complexity of the search. We show how the capability of the SES scheme can be extended such that wildcards in the trapdoor are supported. We conclude the chapter by comparing the complexity of SES with the most prominent symmetric key searchable encryption schemes [19, 26]. We show that the SES scheme has a lower computational complexity for the search than related work.

- **Chapter 4:** We give a comprehensive overview of the public key based techniques that allow searching in encrypted data, and enforcing a role based access control policy by a server. Since a role can leak some information about the message, the role should be stored in a way that the server learns nothing about. The policy should also be enforced in a way that no information about the role is revealed to the server. We propose a unifying framework for searching and enforcing policy by an honest-but-curious server. We then propose our scheme, called SEPE, which permits the server to search and enforce an access control policy without learning much about the policy. We give a formal definition about "learning much". We conclude the chapter by comparing the efficiency of the SEPE scheme with the DGD scheme [20] which is proposed in the random oracle model. We show that the SEPE scheme has the lowest computational complexity for the search and enforcing an access policy.

- **Chapter 5:** We study the problems with existing public key searchable encryption schemes that support wildcards in the trapdoor. We then propose a public key searchable encryption scheme which is provably secure in the selective security model. Our scheme, called SEPS, supports wildcards in the trapdoor. While in existing schemes the computational complexity of the search is linear in the number of non-wildcard characters, the complexity of the SEPS scheme is independent of the number of wildcards. Moreover, SEPS uses more efficient primitives to perform the search, and creates a shorter trapdoor in comparison with existing schemes. We conclude the chapter by comparing the efficiency of SEPS with related work [12, 30]. We show that the SEPS scheme has the lowest computational complexity for the search.

- **Chapter 6:** We first analyze the challenges of achieving full security as well as existing fully secure searchable encryption schemes. We then propose a public key searchable encryption scheme, called SEPF, which is fully secure. The SEPF scheme uses more efficient primitives to perform the search than existing schemes. The SEPF scheme is proven secure using the dual system encryption methodology [42]. We compare the complexity of the SEPF scheme with the existing fully secure scheme in [17]. We show that the SEPF scheme has lower search complexity.

- **Chapter 7:** In the last chapter we draw our conclusions. We analyze the efficiency of the schemes, which we propose in chapters 3, 4, and 5, in the context of their appropriate application.

In this thesis, to answer the research question we propose searchable encryption schemes with lower complexity for the search compared to the existing schemes. Our schemes, presented in chapters 3, 4, 5, and 6 are provably secure in a relevant security model. In addition to the improved search complexity, each of the schemes we propose has a low complexity in some other aspect. Thus, the answer to the main research question is qualified "yes", for certain complexity aspects only. In the concluding chapter we discuss which are the appropriate applications of our schemes with respect to the improved complexity aspects.

Acknowledgement

Chapters 3, 5 and 6 are joint work with Peter van Liesdonk. Both authors contributed equally to each of the chapters.

Chapter 2

Formal Definitions

In this chapter, we present formal definitions for searchable encryption and the security models necessary to analyze searchable encryption schemes. We consider both the symmetric key and public key settings. The security models specify the restrictions on the computational resources of the adversary. Since practical cryptographic primitives are not unconditionally secure, searchable encryption schemes can be proven to be secure in an appropriate security model.

2.1 Introduction

Let $D = (M_1, M_2, ..., M_n)$ be data consisting of n messages $M_1, M_2, ..., M_n$. Each message M_i $(i = 1, ..., n)$ is associated with a metadata item $\mathbb{W}_i = \{W_{i,1}, W_{i,2}, ...\}$ which is actually a set of keywords chosen from a finite set $\mathcal{W}$. Searchable encryption stores the data D on a server such that:

1. A message M_i is retrieved from the server, only in case a particular keyword occurs in its associated metadata $\mathbb{W}_i$, while leaking as little information as possible.

2. The confidentiality of the data is preserved as much as possible.

We now present formal definitions for searchable encryption and the security models, in the symmetric key and public key settings.

2.2 Symmetric Key Searchable Encryption

The goal of the symmetric key searchable encryption is to retrieve encrypted messages from a storage server, when the metadata associated with the message contains a particular keyword. First each message M_i is encrypted, using a standard symmetric

key encryption scheme, and stored on the server. To store the metadata items on the server, in a way that the metadata can be queried later, a symmetric key searchable encryption scheme with the following randomized algorithms is invoked [26].

$\mathsf{Keygen_s}(\sigma)$**:** Given the security parameter σ, outputs a master secret key msk.

$\mathsf{Enc_s}(\mathbb{W}, msk)$**:** Given the metadata $\mathbb{W}$, and the master secret key msk, outputs a searchable ciphertext $S_{\mathbb{W}}$.

$\mathsf{Trapdoor_s}(W, msk)$**:** Given the keyword W, and the master secret key msk, outputs a trapdoor T_W.

$\mathsf{Search}(T_W, S_{\mathbb{W}})$**:** Given the trapdoor T_W, and the searchable ciphertext $S_{\mathbb{W}}$, outputs 1 if $W \in \mathbb{W}$.

The $\mathsf{Keygen_s}$, $\mathsf{Enc_s}$, and $\mathsf{Trapdoor_s}$ algorithms are invoked by the client, and the Search algorithm is invoked by the server. If $\mathsf{Search}(T_W, S_{\mathbb{W}}) = 1$, the server sends back the encrypted message whose associated metadata is $\mathbb{W}$. The message flow of the symmetric key searchable encryption is illustrated in Figure 2.1.

2.2.1 Security

Informally, a searchable encryption scheme in the symmetric key setting is secure if the scheme leaks the access pattern only. The access pattern is the outcome of the Search algorithm which shows whether a searchable ciphertexts matches a trapdoor. Let $L_{\mathbb{W}} = (\mathbb{W}_1, ..., \mathbb{W}_m)$ be a list of m metadata items. Let $L_W = (W_1, ..., W_n)$ be a list of n keywords. The access pattern of $L_{\mathbb{W}}$ and L_W is the matrix $\mathsf{AccessPattern}(L_{\mathbb{W}}, L_W)$ whose i-th row and j-th column are [38]:

$$\mathsf{AccessPattern}(L_{\mathbb{W}}, L_W)_{i,j} = \begin{cases} 1 & \text{if } W_j \in \mathbb{W}_i, \\ 0 & \text{otherwise} \end{cases}$$

where $1 \leq i \leq m$, and $\leq j \leq n$.

The security of symmetric key searchable encryption schemes is defined as the following game between a challenger, who owns the master secret key, and an adversary $\mathcal{A}$ [38].

- **Setup**: The challenger runs the $\mathsf{Keygen_s}(\sigma)$ algorithm to obtain the master secret key msk. The challenger also picks a bit $\beta \in \{0, 1\}$ randomly. The adversary prepares six lists L_S, L_T, $L_{\mathbb{W}_0}, L_{\mathbb{W}_1}, L_{W_0}, L_{W_1}$ which are initially empty.

- **Query**: In this phase, the adversary adaptively makes two types of queries:

  - Searchable ciphertext queries: The adversary sends two metadata items $(\mathbb{W}_0, \mathbb{W}_1)$ to the challenger who picks $\mathbb{W}_\beta$ to compute the searchable ciphertext $S_{\mathbb{W}_\beta} = \mathsf{Enc_s}(\mathbb{W}_\beta, msk)$. The challenger then sends $S_{\mathbb{W}_\beta}$ to the adversary. The adversary appends $S_{\mathbb{W}_\beta}$ to L_S, $\mathbb{W}_0$ to $L_{\mathbb{W}_0}$ and $\mathbb{W}_1$ to $L_{\mathbb{W}_1}$.

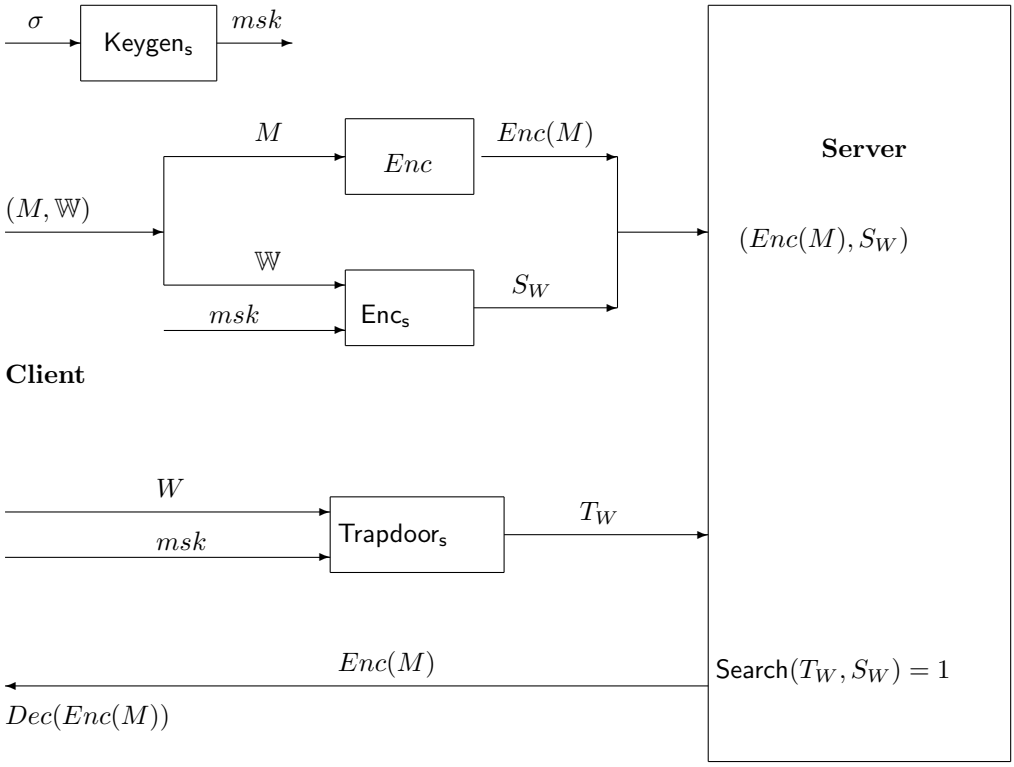

Figure 2.1: *The message flow of the symmetric key searchable encryption for retrieving encrypted messages selectively from a server. Here, Enc is a standard symmetric key encryption scheme.*

– Trapdoor queries: The adversary sends two keywords (W_0, W_1) to the challenger who picks W_β to compute the trapdoor $T_{W_\beta} = \mathsf{Trapdoor_s}(W_\beta, msk)$. The challenger then sends T_{W_β} to the adversary. The adversary appends T_{W_β} to L_T, W_0 to L_{W_0} and W_1 to L_{W_1}.

The only condition for choosing the keywords and the metadata items during the query phase is that

$$\mathsf{AccessPattern}(L_{\mathbb{W}_0}, L_{W_0}) = \mathsf{AccessPattern}(L_{\mathbb{W}_1}, L_{W_1}).$$

- **Response**: After $\mathcal{A}$ decides that the query phase is over, $\mathcal{A}$, using the lists L_T, and L_S outputs a guess β' for β. The adversary $\mathcal{A}$ then sends β' to the challenger.

Intuitively, this game simulates a worst case scenario for the attack. The adversary gathers the maximum possible searchable ciphertexts and trapdoors for the attack. Here, if the adversary can guess the keyword of even one searchable ciphertext or one trapdoor correctly, the attack succeeds. This game also allows the adversary to send his queries for the searchable ciphertexts and the trapdoors adaptively, in a way that each query can be chosen after receiving the response of the previous query. If the scheme leaks even one bit of information from the searchable ciphertext or the trapdoor, the adversary can choose the queries in a way that this flaw is used for guessing β. Here, the metadata items of the searchable ciphertext queries and the keywords of the trapdoor queries must be chosen in a way the adversary cannot learn the bit β trivially. For example, assume that the adversary is allowed to choose two metadata items $(\mathbb{W}_0, \mathbb{W}_1)$ and two keywords (W_0, W_1) for the query, such that $W_0, W_1 \in \mathbb{W}_0$ and $W_0, W_1 \notin \mathbb{W}_1$. In this case, given $S_{\mathbb{W}_\beta}$, and T_{W_β}, the adversary can simply run the Search algorithm to guess β correctly. If $\mathsf{Search}(T_{W_\beta}, S_{\mathbb{W}_\beta}) = 1$, then $\beta = 0$, otherwise, $\beta = 1$. This is the reason why we require that $\mathsf{AccessPattern}(L_{\mathbb{W}_0}, L_{W_0}) = \mathsf{AccessPattern}(L_{\mathbb{W}_1}, L_{W_1})$.

Let $Adv_{\mathcal{A}} = \left| Pr[\beta = \beta'] - \frac{1}{2} \right|$ be the advantage of $\mathcal{A}$ in winning the game.

Definition 1 (Symmetric Key Security)**.** *A symmetric key searchable encryption scheme is secure if for all probabilistic polynomial time (PPT) adversaries $\mathcal{A}$, $Adv_{\mathcal{A}} \leq \varepsilon(\sigma)$, where ε is a negligible function of σ.*

The message flow of the symmetric key security game is illustrated in Figure 2.2

2.3 Public Key Searchable Encryption

Public key searchable encryption schemes create the searchable ciphertext using some public parameters. The goal of the public key searchable encryption is either to decrypt data selectively, or to search for a keyword. A searchable encryption scheme in the public key setting transforms both the message and the metadata associated with the message to a searchable ciphertext. The message can be decrypted using a

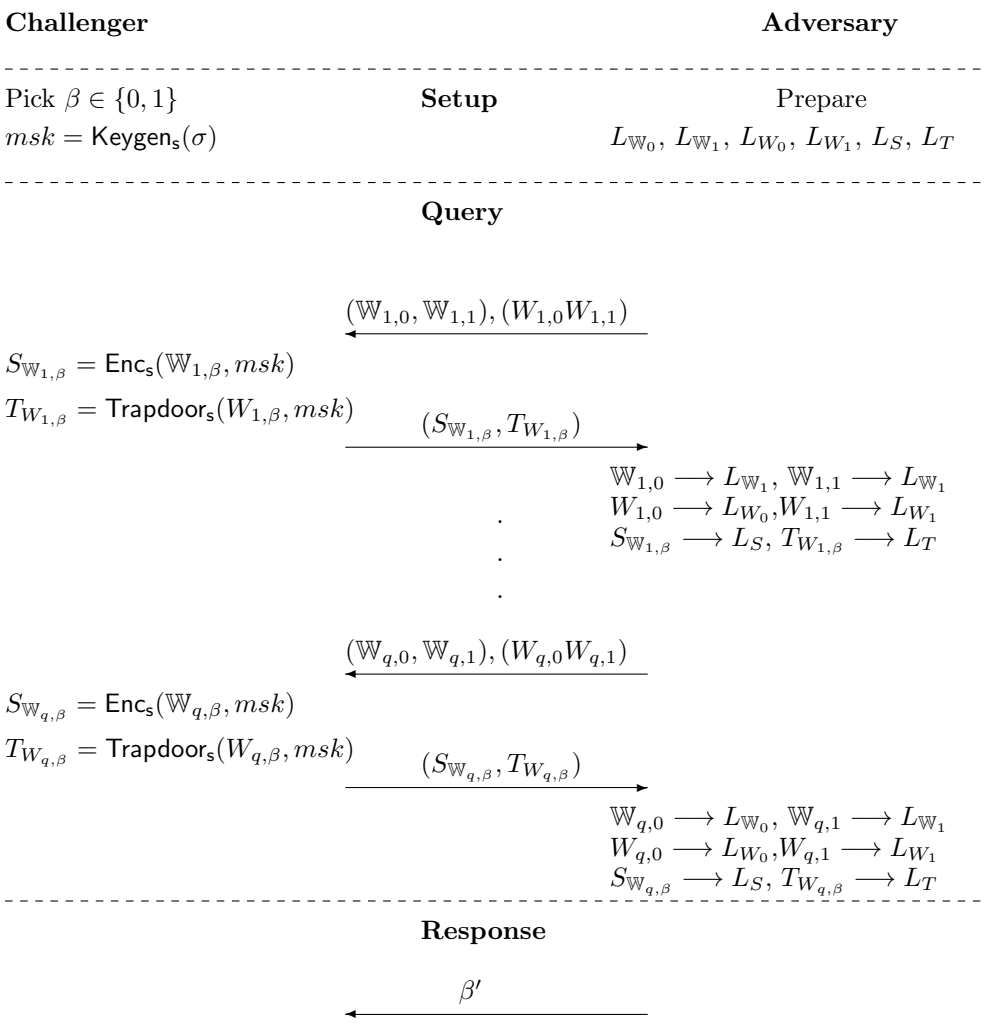

Figure 2.2: *The message flow of the symmetric key security game. Here, q is the number of times that query is performed.*

trapdoor if and only if the keyword of the trapdoor occurs in the metadata associated with the message. Public key searchable encryption schemes consist of the following randomized algorithms [12]:

$\mathsf{Keygen_P}(\sigma)$**:** Given the security parameter σ, outputs a master secret key msk and a set of public parameters $param$.

$\mathsf{Enc_P}(M, \mathbb{W}, param)$**:** Given the message M, the metadata $\mathbb{W}$, and the public parameters $param$, outputs a searchable ciphertext $S_{M,\mathbb{W}}$.

$\mathsf{Trapdoor_P}(W, msk)$**:** Given the keyword W and the master secret key msk, outputs a trapdoor T_W.

$\mathsf{Dec}(T_W, S_{M,\mathbb{W}})$**:** Given the trapdoor T_W, and the searchable ciphertext $S_{M,\mathbb{W}}$, outputs M if and only if $W \in \mathbb{W}$.

Figure 2.3 illustrates the message flow of the public key searchable encryption. Alice generates the master secret key msk and the public parameters $param$. Alice then constructs a trapdoor T_W using the keyword W and delegates T_W to Bob. Assume that Charlie wants to send a message M to Bob. Charlie associates a metadata $\mathbb{W}$ to M and transforms both M and $\mathbb{W}$ to a searchable ciphertext $S_{M,\mathbb{W}}$ using the public parameters $param$. Charlie then sends $S_{M,\mathbb{W}}$ to Bob. Given the searchable ciphertext $S_{M,\mathbb{W}}$, Bob can decrypt M if $W \in \mathbb{W}$.

2.3.1 Security

The security of the public key searchable encryption is defined by a security game between a challenger, who owns the master secret key, and an adversary who tries to learn non-trivial information from the searchable ciphertext. In this game, the adversary is allowed to receive the trapdoor of any keyword that he wants, except for the challenge keyword. This game captures the property that the searchable ciphertext leaks no information on both the message and the metadata. The game proceeds as follows [12]:

- **Setup**. The challenger runs $\mathsf{Keygen_P}(\sigma)$, which outputs the master secret key msk and the public parameters $param$. The challenger then sends $param$ to the adversary $\mathcal{A}$. The adversary prepares two lists L_T and L_W which are initially empty.
- **Query I**. In this phase, $\mathcal{A}$ adaptively issues trapdoor queries. Given a keyword W, the challenger runs $\mathsf{Trapdoor_P}(W, msk)$ which outputs a trapdoor T_W. The challenger then sends T_W to $\mathcal{A}$. The adversary appends T_W to the list L_T, and W to the list L_W.
- **Challenge**. Once $\mathcal{A}$ decides that the query phase is over, $\mathcal{A}$ picks a pair of messages (M_0, M_1) and metadata items $(\mathbb{W}_0, \mathbb{W}_1)$ on which it wishes to be challenged and sends them to the challenger. The only condition is that

$$\mathsf{AccessPattern}(\mathbb{W}_0, L_W) = \mathsf{AccessPattern}(\mathbb{W}_1, L_W)$$

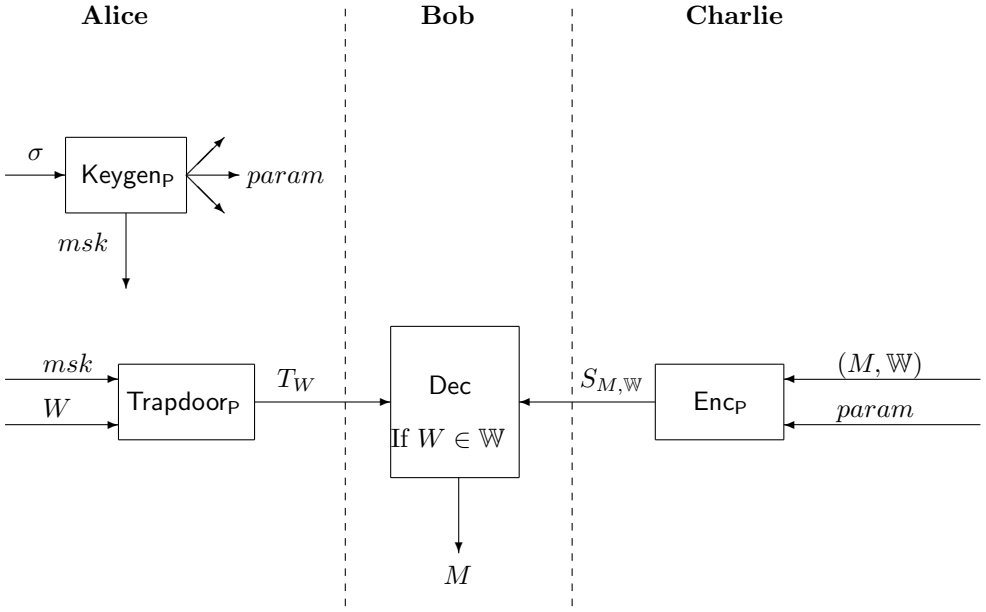

Figure 2.3: *The message flow of public key searchable encryption. Alice, who owns the master secret key, creates a trapdoor and sends it to Bob. Charlie, who wants to send a message to Bob, transforms the message to a searchable ciphertext using the public parameters, and sends it to Bob. Bob decrypts the message if the keyword of the trapdoor and the associated keywords with the message are the same.*

(see Eq. 2.2.1). Given (M_0, M_1) and $(\mathbb{W}_0, \mathbb{W}_1)$, the challenger flips a fair coin $\beta \in \{0,1\}$, and invokes the $\mathsf{Enc_P}(M_\beta, \mathbb{W}_\beta, param)$ algorithm which outputs $S_{M_\beta, \mathbb{W}_\beta}$. The challenger then sends $S_{M_\beta, \mathbb{W}_\beta}$ to $\mathcal{A}$.

- **Query II**. This phase is identical to Query Phase I with the condition that

$$\mathsf{AccessPattern}(\mathbb{W}_0, L_W) = \mathsf{AccessPattern}(\mathbb{W}_1, L_W).$$

- **Output**. Finally, the adversary using L_T outputs a bit β' which represents its guess for β.

Intuitively, this game simulates a worst case situation where the adversary is allowed to gather the maximum possible trapdoors that do not decrypt the challenge. The adversary then tries to learn the message and the associated metadata of the searchable ciphertext, using the trapdoors that have been gathered during the query phases. In contrast to the symmetric key setting, it is not possible to hide the keyword of the trapdoor in the public key setting. Since the adversary has access to the public parameters, given a trapdoor, he can create a searchable ciphertext for any possible keyword to check whether the searchable ciphertext and the trapdoor match. Since in practice the entropy of the keywords is limited, the adversary can learn the keyword of the trapdoor after performing the brute force attack mentioned above. This is the reason why the adversary is allowed to know the keyword of the trapdoor in the query phases. Query phase I allows the adversary to choose the challenge messages based on the trapdoors which are already known. In the public key setting, Query phase II allows the adversary to ask for more trapdoors based on the challenge ciphertext. If the encryption scheme leaks even one bit of information, the adversary can choose the message and the keyword in such a way that this weakness is used for guessing β.

Let $Adv_{\mathcal{A}} = \left| Pr[\beta = \beta'] - \frac{1}{2} \right|$ be the advantage of $\mathcal{A}$ in winning the game.

Definition 2 (Full Security)**.** *A searchable encryption scheme is* fully secure *if for all probabilistic polynomial-time adversaries $\mathcal{A}$ in the full security game, $Adv_{\mathcal{A}} \leq \varepsilon(\sigma)$, where $\varepsilon(\sigma)$ is a negligible function of σ.*

Selective security model. We define a weaker security notion called selective security. The selective security game is the same as the fully secure game except that instead of submitting two keywords (W_0, W_1) in the challenge phase, the adversary commits to the keywords at the beginning of the game [16]. Although the selective security model is a weaker model than the full security model, it has appeared in various constructions in the literature. While the full security model guarantees protecting of all metadata items in the searchable ciphertext, selective security guarantees protecting only one predefined metadata item. However, selective security makes it easier to prove the security of a scheme, which implies less cost for the scheme.

Definition 3 (Selective Security)**.** *A searchable encryption scheme is selectively secure if for all probabilistic polynomial-time adversaries (PPT) $\mathcal{A}$ in the selective security game, $Adv_{\mathcal{A}} \leq \varepsilon(\sigma)$, where $\varepsilon(\sigma)$ is a negligible function of σ.*

The message flow of the full security and selective security games in the public key setting is illustrated in Figure 2.4.

2.4 Primitives and Complexity Assumptions

In this section, the cryptographic primitives and the complexity assumptions that we use in the schemes we propose in the next chapters, are formally defined.

Pseudorandom Function. A pseudorandom function $f : \mathcal{X} \times \mathcal{K} \longrightarrow \mathcal{Y}$ transforms each element $x \in \mathcal{X}$ to an output $y \in \mathcal{Y}$ with a secret key $k_f \in \mathcal{K}$ such that the output is not predictable.

Definition 4 (Secure Pseudorandom Function)**.** *[26] A pseudorandom function $f : \mathcal{X} \times \mathcal{K} \longrightarrow \mathcal{Y}$, is a (t, q, ε_f) secure pseudorandom function if for every algorithm A, which makes at most q oracle queries with a running time of at most t, has advantage:*

$$\Big|Pr[A^{f_{k_f}(.)} = 1 | k_f \in \mathcal{K}] - Pr[A^R = 1 | R \in \{F : \mathcal{X} \to \mathcal{Y}\}]\Big| < \varepsilon_f$$

where R is a true random function chosen uniformly from the set of all maps from $\mathcal{X}$ to $\mathcal{Y}$.

Intuitively, for any PPT algorithm A, the probability of guessing the output of a pseudorandom function correctly, after sending any number of queries, is negligibly larger than the probability of guessing the output of a true random function.

Pseudorandom Permutation Function. $\mathcal{E} : \mathcal{X} \times \mathcal{K} \to \mathcal{X}$ transforms each element $x_1 \in \mathcal{X}$ to an element $x_2 \in \mathcal{X}$ using a secret key $k_e \in \mathcal{K}$ in a way that the output is not predictable.

Definition 5 (Secure Pseudorandom Permutation Function)**.** *[40] A pseudorandom permutation function $\mathcal{E} : \mathcal{X} \times \mathcal{K} \to \mathcal{X}$ is a (t, q, ε_e) secure pseudorandom permutation function if every algorithm A, which makes at most q queries with a running time of at most t, has advantage:*

$$\Big|Pr[A^{\mathcal{E}_{k_e}(.)} = 1 | k_e \in \mathcal{K}] - Pr[A^{\pi} = 1 | \pi \in \{F : \mathcal{X} \to \mathcal{X}\}]\Big| < \varepsilon_e$$

where π is a true random permutation selected uniformly from the set of all bijections on $\mathcal{X}$.

Intuitively, for any PPT algorithm A, the probability of guessing the output of a pseudorandom permutation function correctly, after sending any number of queries, is negligibly larger than the probability of guessing the output of a true random bijection function.

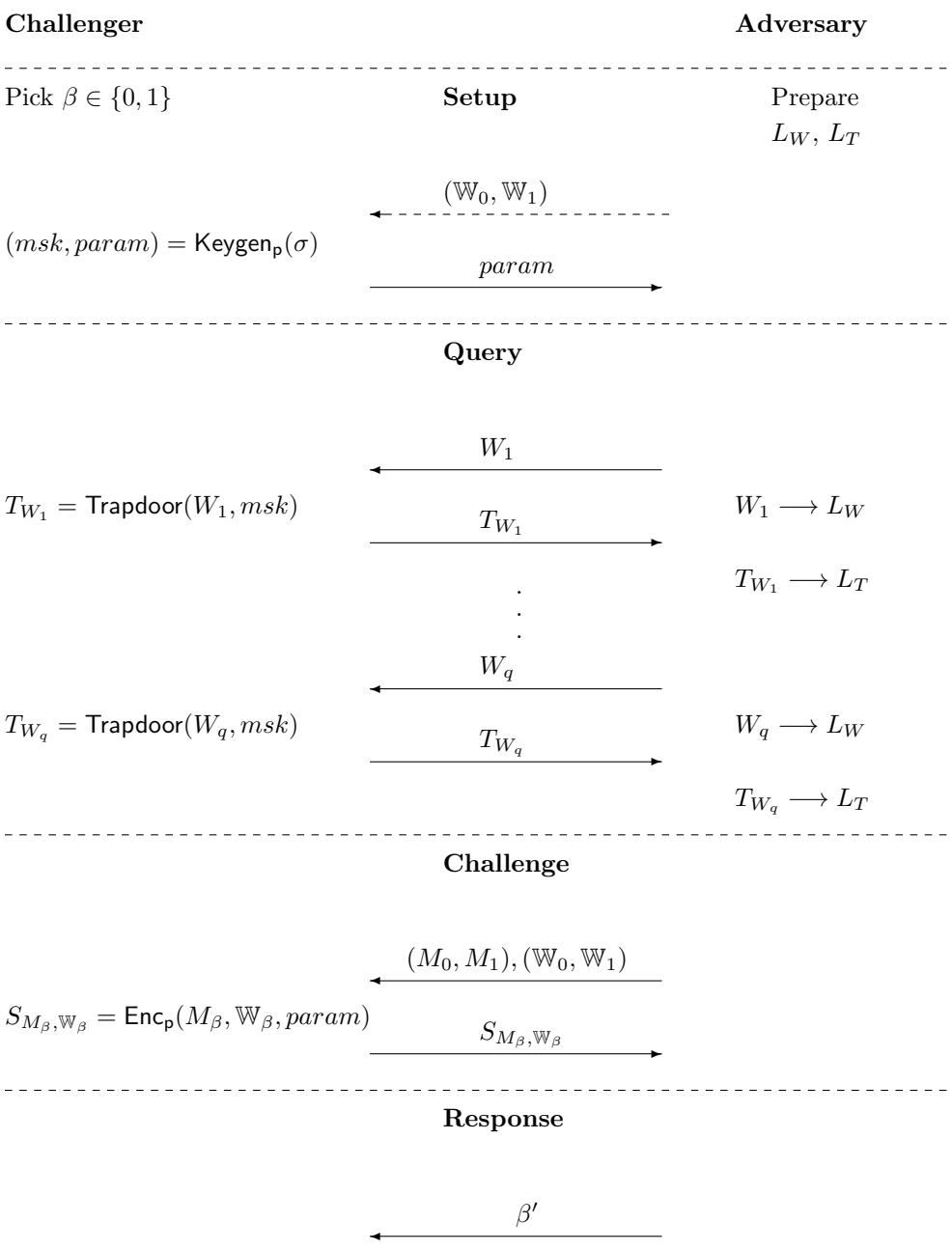

Figure 2.4: *The message flow of the public key security games. The dashed vector belongs to the selective security game only.*

Definition 6 (Bilinear Groups.). *[10] A cyclic group $\mathbb{G}$ of order p with generator g is a* bilinear group *if there exists a group $\mathbb{G}_T$ and a map e such that*

- $(\mathbb{G}_T, \cdot)$ *is also a cyclic group, of prime order p,*
- $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. *In other words, for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p^*$, we have $e(u^a, v^b) = e(u, v)^{ab}$.*
- $e(g, g)$ *is a generator of $\mathbb{G}_T$ (non-degenerate).*

Additionally, for efficiency reasons, we require that the group actions and the bilinear map can be computed in polynomial time. A bilinear map that satisfies these conditions is called *admissible*.

The order P of the bilinear groups can be either a prime number or a composite of prime number. In general, bilinear groups of prime order are more efficient than bilinear groups of composite order because prime order groups are shorter than composite order groups.

Definition 7 (Decision Linear Assumption). *[9] The Decision Linear (DLin) assumption states that there exist bilinear groups $\mathbb{G}$ such that for all probabilistic polynomial-time algorithms $\mathcal{A}$,*

$$\Big| Pr\big[\mathcal{A}(\mathbb{G}, g, g^{z_1}, g^{z_2}, g^{z_1 z_3}, g^{z_4}, g^{z_2(z_3+z_4)}) = 1\big] -$$

$$Pr\big[\mathcal{A}(\mathbb{G}, g, g^{z_1}, g^{z_2}, g^{z_1 z_3}, g^{z_4}, g^{r}) = 1\big] \Big| < \varepsilon(\sigma)$$

for some negligible function $\varepsilon(\sigma)$, where the probabilities are taken over all possible choices of $z_1, z_2, z_3, z_4, r \in \mathbb{Z}_p^*$.

Informally, the DLin assumption states that given a bilinear group $\mathbb{G}$ and elements $g^{z_1}, g^{z_2}, g^{z_1 z_3}, g^{z_4}$ it is hard to distinguish $g^{z_2(z_3+z_4)}$ from a random element in $\mathbb{G}$.

Definition 8 (Decisional Bilinear Diffie-Hellman Assumption). *[10] The Decisional Bilinear Diffie-Hellman (DBDH) assumption states that there exist bilinear groups $\mathbb{G}$ such that for all probabilistic polynomial-time algorithms $\mathcal{A}$,*

$$\Big| Pr\big[\mathcal{A}(\mathbb{G}, g, g^{z_1}, g^{z_2}, g^{z_3}, e(g, g)^{z_1 z_2 z_3}) = 1\big] -$$

$$Pr\big[\mathcal{A}(\mathbb{G}, g, g^{z_1}, g^{z_2}, g^{z_3}, e(g, g)^{r}) = 1\big] \Big| < \varepsilon(\sigma)$$

for some negligible function $\varepsilon(\sigma)$, where the probabilities are taken over all possible choices of $z_1, z_2, z_3, r \in \mathbb{Z}_p^*$.

Informally, the DBDH assumption states that given a bilinear group $\mathbb{G}$ and elements $g^{z_1}, g^{z_2}, g^{z_3}$, it is hard to distinguish the value $e(g, g)^{z_1 z_2 z_3}$ from a random element chosen from $\mathbb{G}_T$.

Chapter 3

Efficient Symmetric Key Searchable Encryption

In existing symmetric key searchable encryption schemes the computational complexity of the search is linear in the total number of the searchable ciphertexts stored on the server. There are a few schemes that search with a lower complexity. However, these schemes cannot update the database efficiently. In this chapter, we propose a novel symmetric key searchable encryption scheme, called SES. The SES scheme has a lower computational complexity for the search compared to the existing schemes that allow efficient update of the database. Two variants of the SES scheme are proposed, which differ in the computational complexity and in the communication complexity of the search. We compare the complexity of the SES scheme with the complexity of the SI [26] and SSE [19] schemes, which are two prominent existing schemes in the symmetric key setting. The SES scheme is proven secure in the symmetric key security model. This chapter is a heavily revised version of the paper published in the proceedings of the 7th Conference on Secure Data Management [3].

3.1 Introduction

Various symmetric key searchable encryption schemes have been proposed [40, 26, 18, 27]. Most of these schemes suffer from the problem that the search complexity is linear in the total number of the searchable ciphertexts stored on the server. Only a few schemes allow more efficient search [19]. However, in those schemes the update of the database is performed inefficiently, in the sense that all the searchable ciphertexts stored on the server should be replaced by new searchable ciphertexts. The problem is thus that existing schemes perform either the search or the update inefficiently. The goal of this chapter is to propose a searchable encryption scheme that allows both, efficient search and update.

Contribution. In this chapter, we propose a novel symmetric key searchable encryption scheme called SES, which is provably secure in the symmetric key security model. The SES scheme searches for a keyword with a lower computational complexity compared to the existing schemes which allow updating the database efficiently. The computational complexity of the search in our scheme is linear in the number of the searchable ciphertexts that match the trapdoor. Since the number of the searchable ciphertexts that match the trapdoor is lower than the total number of the searchable ciphertexts, our scheme has a lower computational complexity for the search compared to existing schemes. The SES scheme allows the client to update the database efficiently and securely. We propose two variants of the SES scheme which differ in the computational complexity and in the communication complexity of the search. The first scheme called SES1, performs the search interactively with the client. The second scheme, called SES2, performs the search non-interactively but at the cost of higher complexity for the trapdoor.

3.2 Related Work

In this section, we first review existing symmetric key searchable encryption schemes. Then, we discuss the efficiency problems in the search algorithm of existing schemes. The problem of searching encrypted data was first studied by Song, Wagner, and Perrig [40], who propose the first symmetric key searchable encryption scheme called SWP. The major drawback of the SWP scheme is that the computational complexity of the search is linear in the number of keywords of the metadata *per* searchable ciphertext. The SI scheme proposed by Eu-Jin Goh [26] uses a Bloom filter to search *each* searchable ciphertext for a keyword with constant computational complexity. In both the SWP and SI schemes, the searchable ciphertext leaks information about the number of the keywords of the metadata. Chang and Mitzenmacher have proposed a symmetric key searchable encryption scheme whose searchable ciphertext hides the number of the keywords of the metadata [18]. In this scheme, the computational complexity to search a searchable ciphertext is constant but the computational complexity of creating a searchable ciphertext is linear in the number of all possible keywords. Golle et al. have proposed a scheme which searches for conjunctive keywords with constant complexity per searchable ciphertext [27].

The schemes mentioned above have a common drawback: the complexity of searching the *database* is linear in the number of searchable ciphertexts stored on the server. To address this issue, Curtmola et al. [19] propose a scheme called SSE, which has a constant computational complexity to perform the search. However, SSE does not allow the database to be updated efficiently. This property makes the scheme suitable for one-time storage only.

3.3 Construction of SES

Let $D = (M_1, ..., M_n)$ be data consisting of n messages. Each message M_i, $(i = 1, ..., n)$ is associated with metadata $\mathbb{W}_i = \{W_{i,1}, W_{i,2}, ...\}$ consisting of a set of keywords. Each message M_i and metadata item $\mathbb{W}_i$, are associated with a unique identifier ID_i.

High Level Intuition. In existing schemes, each metadata item $\mathbb{W}$ is transformed to a searchable ciphertext $S_{\mathbb{W}}$. The searchable ciphertext $S_{\mathbb{W}}$ matches the trapdoor T_W if $W \in \mathbb{W}$. Therefore, to search for a keyword, the server has to check for each searchable ciphertext, whether it matches the trapdoor. This property makes the computational complexity of the search linear in the total number of the searchable ciphertexts stored on the server.

In our scheme, we improve the efficiency of the search by building an index for the keywords. The index maps each unique keyword onto a list of identifiers, which show the desired metadata items. Since the index introduces a new indirection, there will be costs associated with it. We will analyze the costs in section 3.4.

3.3.1 The SES Scheme

Notation We write $x \longleftarrow X$ to represent an element x being sampled uniformly from a set X. We denote string concatenation by $||$.

Here, we first present the construction of the SES scheme and then we give the intuition behind the construction. The SES scheme uses a counter t which is initially 1 and is incremented each time the database is updated. The SES scheme consists of the following algorithms:

KeygenSES(σ)**:** Given the security parameter σ, output a master secret key $msk = (k_f, k_e)$, where $k_f, k_e \longleftarrow \{0,1\}^\sigma$.

EncSES($\{(\mathbb{W}_1, ID_1), ..., (\mathbb{W}_n, ID_n)\}, msk, t$)**:** Given the metadata items and their identifiers, $\{(\mathbb{W}_1, ID_1), ..., (\mathbb{W}_n, ID_n)\}$, the master secret key msk, and the counter t, for each unique keyword $W \in \{\mathbb{W}_1, ..., \mathbb{W}_n\}$, output a searchable ciphertext $S_{W,t}$. After computing a searchable ciphertext for all the unique keywords that occur in $\{\mathbb{W}_1, ..., \mathbb{W}_n\}$, the algorithm increments the counter $t = t + 1$.

TrapdoorSES(W, msk, t)**:** Given the keyword W, the master secret key msk, and the counter t, output a trapdoor $T_{W,t}$.

SearchSES($T_{W,t}, S$)**:** Let S be the set of all the searchable ciphertexts stored on the server. Given the trapdoor $T_{W,t}$ and the searchable ciphertexts S, output the identifiers showing in which metadata items the query keyword W occurs.

The KeygenSES, EncSES, and the TrapdoorSES algorithms are invoked by the client and the SearchSES algorithm is invoked by the server.

The message exchange of the SES scheme is illustrated in Figure 3.1. Here, we summarize the differences between this figure and Figure 2.1, which shows the message exchange of symmetric key searchable encryption:

- The $\mathsf{Keygen_s}$, $\mathsf{Enc_s}$, $\mathsf{Trapdoor_s}$, and Search algorithms in Figure 2.1 are replaced by the KeygenSES EncSES, TrapdoorSES, and SearchSES algorithms in Figure 3.1.
- The EncSES algorithm takes the keyword $W \in \mathbb{W}$ and the metadata identifier ID as input, while the input parameter for $\mathsf{Enc_s}$ is just the metadata item $\mathbb{W}$.
- The Enc algorithm in Figure 2.1 only takes the message M as input, while the the Enc algorithm in Figure 3.1 takes both, the message M and its identifier ID, as input.
- The EncSES and TrapdoorSES algorithms take the number of updates t as input, such that the searchable ciphertext $S_{W,t}$ and the trapdoor $T_{W,t}$ depends on t. In Figure 2.1, the searchable ciphertext $S_{\mathbb{W}}$ and the trapdoor T_W are independent of the number of updates.
- In SES, the identifier ID is stored with the searchable ciphertexts $S_{W,t}$ and the encrypted message $Enc(M)$ on the server. In existing schemes the identifier is not needed.
- The SearchSES algorithm outputs the identifier ID of the metadata item that contains the queried keyword W. In existing schemes, the Search algorithm outputs either "0" or "1".

Intuition for the Construction. To construct the searchable ciphertext, EncSES first associates each unique keyword $W \in \{\mathbb{W}_1, ..., \mathbb{W}_n\}$ with a set of identifiers $I_{W,t} = \{ID_i | W \in \mathbb{W}_i\}$ showing in which metadata items W occurs. The algorithm then transforms both, the keyword W and the associated set of identifiers $I_{W,t}$, to a searchable ciphertext $S_{W,t}$. To update the database in a secure way, the searchable ciphertext $S_{W,t}$ of the current update t, must be indistinguishable from the searchable ciphertexts $S_{W,t-1}$, ..., $S_{W,1}$ of the previous updates. By indistinguishability we mean that the server cannot learn that the searchable ciphertexts $S_{W,t}$, ..., $S_{W,1}$ of different updating time belong to the keyword. Otherwise, the server learns that there is a common keyword between the currently stored metadata items and the previously stored ones. This is the reason why the number of updates t is one of the parameters of the searchable ciphertext.

Each searchable ciphertext $S_{W,t}$ stores the set of identifiers $I_{W,t}$ in an encrypted form to hide it from the server. In SES, the trapdoor $T_{W,t}$, which is computed after t updates, allows the server to decrypt the set of identifiers $I_{W,t}$,....,$I_{W,1}$ which occur in the searchable ciphertexts $S_{W,t}, ..., S_{W,1}$. Here, the trapdoor $T_{W,t}$ must not reveal any information about the identifiers occurring in the searchable ciphertexts of future updates $S_{W,t+1}$, $S_{W,t+2}$, ... Otherwise, the security of future updates is compromised. This is the reason why the number of updates t is also one of the parameters of the trapdoor. Finally, to search for a keyword W, the SearchSES algorithm first searches the database for the searchable ciphertexts $S_{W,t}$,....,$S_{W,1}$ using the trapdoor $T_{W,t}$. Then, the algorithm decrypts the set of identifiers $I_{W,t}$,...,$I_{W,1}$, which point out in which metadata items the queried keyword W occurs.

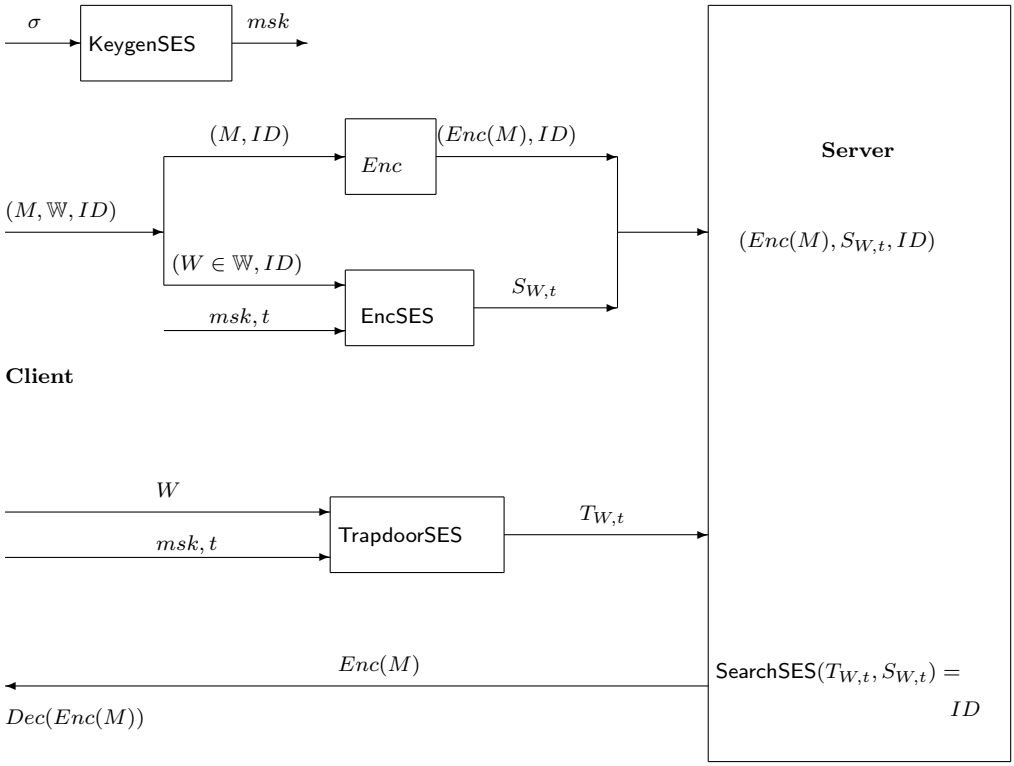

Figure 3.1: *The message flow of the SES scheme. Here, Enc is a standard symmetric key encryption scheme. In SES, the message M and its identifier ID are encrypted using a symmetric key encryption scheme Enc. To compute the searchable ciphertexts, The* EncSES *algorithm transforms every unique keyword of the metadata item $\mathbb{W}$ to a searchable ciphertext $S_{W,t}$ using the master secret key msk, the associated identifer ID, and the number of the updates t. The client then stores the triple $(Enc(M), S_{W,t}, ID)$ on the server. To query for the keyword W, the client transforms the keyword W to a trapdoor $T_{W,t}$ using the master secret key msk and the number of updates t. Given the trapdoor, the server invokes the* SearchSES *algorithm which reveals the identifier ID of the metadata items that contain W. The server then sends the encrypted message $Enc(M)$ which is associated with the metadata $\mathbb{W}$ to the client.*

	SES1	SES2
Revealing identifiers for search	Directly using master secret key	Indirectly using master secret key
Search	Interactive	Non-interactive
Complexity of trapdoor	Lower	Higher

Table 3.1: *Comparison of the search functionality and complexity of SES1 and SES2.*

We present two variants of the SES scheme called SES1 and SES2. The SES1 scheme performs the search interactively with the client. The SES2 searches non-interactively but at a cost of a higher complexity for the trapdoor.

The constructions of SES1 and SES2 differ in the way the set of identifiers can be decrypted for the search. In SES1, to decrypt the set of identifiers, the master secret key should be used directly. Since revealing the master secret key to the server compromises the security of the data, the client has to decrypt the set of identifiers rather than the server. This makes the search interactive between the client and the server. In SES2 the identifiers can be decrypted using a decryption key which is derived from the master secret key, i.e. the identifiers are decrypted using the master secret key indirectly. The decryption keys are computed in the trapdoor. This makes the search non-interactive, because the server can decrypt the set of identifiers using the trapdoor. However, computing the trapdoor has a higher complexity due to computing the decryption keys. Table 3.1 illustrates the main differences between SES1 and SES2. A more detailed comparison of SES1 and SES2 is given in Table 3.2 in Section.3.4

3.3.2 Construction of SES1

The SES1 scheme uses a pseudorandom function f: $\{0,1\}^* \times \{0,1\}^\sigma \longrightarrow \{0,1\}^m$, for some m, and a pseudorandom permutation function $\mathcal{E}$: $\{0,1\}^\sigma \times \{0,1\}^\sigma \longrightarrow \{0,1\}^\sigma$. Here, σ is the security parameter. SES1 consists of the following algorithms:

KeygenSES(σ)**:** Given the security parameter σ, output the master secret key $msk = (k_f, k_e)$, where $k_f, k_e \longleftarrow \{0,1\}^\sigma$.

EncSES1($\{(\mathbb{W}_1, ID_1), ..., (\mathbb{W}_n, ID_n)\}, msk, t$)**:** Given the metadata items and their identifiers, $\{(\mathbb{W}_1, ID_1), ..., (\mathbb{W}_n, ID_n)\}$, the master secret key msk, and the counter t, for each unique keyword $W \in \{\mathbb{W}_1, ..., \mathbb{W}_n\}$,

1. compute the set $I_{W,t} = \{ID_i | W \in \mathbb{W}_i\}$ which shows in which metadata items the keyword W occurs,
2. pick a random $r \longleftarrow \{0,1\}^{\sigma}$,
3. compute the searchable ciphertext:

$$S_{W,t} = (f_{k_f}(W||t), \mathcal{E}_r(I_{W,t}), \mathcal{E}_{k_e}(r)).$$

After computing a searchable ciphertext for all the unique keywords that occur in $\{\mathbb{W}_1, ..., \mathbb{W}_n\}$, the algorithm increments the counter $t = t + 1$.

The first component of $S_{W,t}$ makes a commitment to the keyword W while keeping W hidden. The second component encrypts the set of identifiers $I_{W,t}$ with the encryption key r, and the third component stores the encryption key r in an encrypted form. Here, if $I_{W,t} \nsubseteq \{0,1\}^{\delta}$, the identifiers $I_{W,t}$ are broken into several blocks of δ bits, such that each block is individually encrypted. In case the last block has less than δ bits some zeros must be padded to the last block. If the client can store each $((W,t),r)$ on his memory, the third component can be removed from the searchable ciphertext.

TrapdoorSES1(W, msk, t)**:** Given the keyword W, the master secret key msk, and the counter t, compute the trapdoor

$$T_{W,t} = (f_{k_f}(W||1), ..., f_{k_f}(W||t)).$$

Informally, the trapdoor contains queried keyword W at the update times $1, 2, ..., t$ in an encrypted form.

SearchSES1$(T_{W,t}, S)$**:** Given the trapdoor T_W and the searchable ciphertexts S, for $i = 1, .., t$, the algorithm searches the first component of searchable ciphertexts for $f_{k_f}(W||i) \in T_W$. If $f_{k_f}(W||i)$ occurs,

1. The algorithm sends $\mathcal{E}_{k_e}(r)$ to the client.
2. Given $\mathcal{E}_{k_e}(r)$, the client decrypts r using the decryption key k_e and sends r to the server.
3. Given r, the server decrypts $I_{W,i}$ using the decryption key r, and reads the identifers that occur in $I_{W,i}$.

3.3.3 Proof of Security for SES1

Informally, to prove that SES1 is secure in the symmetric key searchable encryption security model, we must show that the searchable ciphertext and the trapdoor of SES1 are indistinguishable from random. We use a sequence of games, where in each game we show that certain components of the searchable ciphertext and the trapdoor can be replaced by some random values in a way that the adversary cannot detect this replacement. In the last game, we show that the adversary cannot distinguish

the searchable ciphertext and the trapdoor of SES1 from the searchable ciphertext and the trapdoor whose components are fully replaced by some random values.

Let $S_{W,t} = (C_1, C_2, C_3)$ and let $T_{W,t} = (T_1, ..., T_t)$. Consider $\mathsf{SES}^{\mathsf{i}}$ schemes, $i = 1, 2, 3$ whose searchable ciphertext $\mathsf{S}^i_{W,t}$ and trapdoor $\mathsf{T}^i_{W,t}$ are:

$$\mathsf{SES}^1: \quad \mathsf{S}^1_{W,t} = (R, C_2, C_3), \mathsf{T}^1_{W,t} = (R_1, ..., R_t)$$
$$\mathsf{SES}^2: \quad \mathsf{S}^2_{W,t} = (R, C_2, R'), \mathsf{T}^2_{W,t} = (R_1, ..., R_t)$$
$$\mathsf{SES}^3: \quad \mathsf{S}^3_{W,t} = (R, R'', R'), \mathsf{T}^3_{W,t} = (R_1, ..., R_t)$$

where R, R', R'', and $R_1, ..., R_t$ are random values. Each $\mathsf{SES}^{\mathsf{i}}$ has a better security than $\mathsf{SES}^{\mathsf{j}}$ if $i > j$, because more random elements are used to construct $\mathsf{SES}^{\mathsf{i}}$. The SES^3 scheme is perfectly secure since all the elements of its searchable ciphertext $\mathsf{S}^3_{W,t}$ and trapdoor $\mathsf{T}^3_{W,t}$ are random. To prove that SES1 is secure in the symmetric key model defined in 2.2.1, we must show that no PPT adversary can distinguish the searchable ciphertext $S_{W,t}$ and trapdoor $T_{W,t}$ of SES1 from the purely random $\mathsf{S}^3_{W,t}$ and $\mathsf{T}^3_{W,t}$ respectively.

The strategy of the security proof is illustrated in Figure 3.2. To prove that no PPT adversary can distinguish $S_{W,t}$ and $T_{W,t}$ from $\mathsf{S}^3_{W,t}$ and $\mathsf{T}^3_{W,t}$ respectively, we define a sequence of games. Let $Game_{Real}$ be the symmetric key security game of the symmetric key security model (See Figure 2.2). Let $Game_i$ be a symmetric key security game which is the same as $Game_{Real}$, except that the challenger responds to each query with the searchable ciphertext and the trapdoor of the SES^i scheme. Since in $Game_3$, the challenger responds to each query with $\mathsf{S}^3_{W,t}$ and $\mathsf{T}^3_{W,t}$ which are random values, the advantage of the adversary in winning the game is zero. If the advantage of the adversary in distinguishing $Game_{Real}$ from $Game_3$ is negligible, the advantage of the adversary in distinguishing $S_{W,t}$ and $T_{W,t}$ from $\mathsf{S}^3_{W,t}$ and $\mathsf{T}^3_{W,t}$ respectively is negligible.

To prove that $Game_{Real}$ is indistinguishable from $Game_3$, we first prove that $Game_{Real}$ is indistinguishable from $Game_1$, We then prove that $Game_1$ is indistinguishable from $Game_2$ and finally $Game_2$ is indistinguishable from $Game_3$. The advantage of using such a sequence of games is that at each step we can prove the indistinguishability only of certain components of the searchable and the trapdoor from random. This makes the proof less complicated.

Lemma 1. *$Game_{Real}$ is indistinguishable from $Game_1$ in the symmetric key security model assuming that $f(.)$ is a (t, q, ε_f) secure pseudorandom function (see 2.4).*

Proof: Assume that there exists an adversary $\mathcal{A}$ which distinguishes $Game_{Real}$ from $Game_1$ with the advantage ε. We show how to build an algorithm B which uses $\mathcal{A}$ to distinguish the pseudorandom function from the true random function with an advantage ε.

Assume the challenger picks a random $\gamma \longleftarrow \{0,1\}$ before the game starts. If $\gamma = 0$, the challenger uses the pseudorandom function f: $\{0,1\}^* \times \{0,1\} \longrightarrow \{0,1\}^m$ to construct the searchable ciphertext and the trapdoor for each query. Otherwise,

Step 1 (Lemma 1):	Prove $Game_{Real} \equiv Game_1$. Proof shows that: $S_{W,t} \equiv \mathsf{S}^1_{W,t}$, $T_{W,t} \equiv \mathsf{T}^1_{W,t}$.
Step 2 (Lemma 2):	Prove $Game_1 \equiv Game_2$. Proof shows that: $\mathsf{S}^1_{W,t} \equiv \mathsf{S}^2_{W,t}$, $\mathsf{T}^1_{W,t} \equiv \mathsf{T}^2_{W,t}$.
Step 3 (Lemma 3):	Prove $Game_2 \equiv Game_3$. Proof shows that: $\mathsf{S}^2_{W,t} \equiv \mathsf{S}^3_{W,t}$, $\mathsf{T}^2_{W,t} \equiv \mathsf{T}^3_{W,t}$.
Step 4 (Theorem 1):	Prove $S_{W,t} \equiv$ random and $T_{W,t} \equiv$ random.

Figure 3.2: *The strategy of the security proof for SES1.*

the challenger picks some random values to respond the queries. Let Γ_0 be the pseudorandom function $f(.)$ and Γ_1 be a true random function. The algorithm B interacts with the adversary $\mathcal{A}$ in the following game to guess the value γ based on the advantage of $\mathcal{A}$ in breaking the scheme:

- **Setup**: The challenger runs the $\mathsf{Keygen_s}(\sigma)$ algorithm to obtain the master secret key $msk = (k_f, k_e)$, where $k_f, k_e \longleftarrow \{0,1\}^\sigma$. The algorithm B picks a bit $\beta \longleftarrow \{0,1\}$ randomly. The adversary $\mathcal{A}$ prepares six lists $L_{\mathbb{W}_0}, L_{\mathbb{W}_1}, L_{W_0}, L_{W_1}$, L_S and L_T which are initially empty. The algorithm B creates a list L_f which is initially empty.
- **Query I**: In this phase, $\mathcal{A}$ adaptively makes two types of queries. Assume that this is the j-th query.
  - Searchable ciphertext query: $\mathcal{A}$ sends two metadata items with their identifiers:

    $$(\mathbb{W}_0 = \{W_{0,1}, ..., W_{0,n}\}, ID_0), (\mathbb{W}_1 = \{W_{1,1}, ..., W_{1,n}\}, ID_1)$$

    to the algorithm B. The algorithm B picks the tuple $(\mathbb{W}_\beta, ID_\beta)$ and sends the triple $(\mathbb{W}_\beta, ID_\beta, j)$ to the challenger. Given the triple, for each unique keyword $W_{\beta,i} \in \mathbb{W}_\beta$, $(i = 1, .., n)$ the challenger runs the $\mathsf{EncSES1}(\mathbb{W}_\beta, msk, j)$ algorithm which outputs $(\Gamma_\gamma(W_{\beta,i}||j), \mathcal{E}_r(ID_\beta), \mathcal{E}_{k_e}(r))$. The challenger then sends $(\Gamma_\gamma(W_{\beta,i}||j), \mathcal{E}_r(ID_\beta), \mathcal{E}_{k_e}(r))$ as the response to the query to the algorithm B. The algorithm B appends $(W_{\beta,i}||j, \Gamma_\gamma(W_{\beta,i}||j))$

to L_f and then sends $(\Gamma_\gamma(W_{\beta,i}||j), \mathcal{E}_r(ID_\beta), \mathcal{E}_{k_e}(r))$ to $\mathcal{A}$. The adversary $\mathcal{A}$ appends the received triple, the metadata $\mathbb{W}_0$, and $\mathbb{W}_1$ to the lists L_S, $L_{\mathbb{W}_0}$ and $L_{\mathbb{W}_1}$ respectively. In case $\gamma = 0$, the function Γ_0 represents the pseudorandom function $f_{k_f}(.)$ and the response from the challenger is the searchable ciphertext of SES1: $S_{W,j} = (f_{k_f}(W_{\beta,i}||j), \mathcal{E}_r(ID_\beta), \mathcal{E}_{k_e}(r))$. Otherwise, the response is the searchable ciphertext of SES^1: $\mathsf{S}^1(W, j) = (R, \mathcal{E}_r(ID_\beta), \mathcal{E}_{k_e}(r))$, where R is randomly picked from the set $\{0,1\}^m$.

– Trapdoor query: $\mathcal{A}$ sends two keywords (W_0, W_1) to the algorithm B who picks W_β and sends (W_β, j) to the challenger. Given (W_β, j), the challenger runs the $\mathsf{TrapdoorSES1}(W_\beta, msk, j)$ algorithm which outputs

$$(\Gamma_\gamma(W_\beta||1), ..., \Gamma_\gamma(W_\beta||j))$$

. The challenger then sends

$$(\Gamma_\gamma(W_\beta||1), ..., \Gamma_\gamma(W_\beta||j))$$

to the algorithm B. The algorithm B appends

$$((W_\beta||1, \Gamma_\gamma(W_\beta||1)), ..., (W_\beta||j, \Gamma_\gamma(W_\beta||j)))$$

to the list L_f and sends the response to $\mathcal{A}$. The adversary $\mathcal{A}$ then appends the response and the keywords W_0 and W_1 to the lists L_T, L_{W_0}, and L_{W_1} respectively. In case $\gamma = 0$, the response is the trapdoor of SES1:

$$T_{W,j} = (f_{k_f}(W||1), ..., f_{k_f}(W||j))$$

. Otherwise the response is the trapdoor of SES^1: $\mathsf{T}^1_{W_\beta,j} = (R_1, ..., R_j)$, where $R_1, ..., R_j$ are randomly picked from the set $\{0,1\}^m$.

Here the condition on the keywords of the Query phase is that

$$\mathsf{AccessPattern}(\mathsf{L}_{\mathbb{W}_0}, \mathsf{L}_{\mathsf{W}_0}) = \mathsf{AccessPattern}(\mathsf{L}_{\mathbb{W}_1}, \mathsf{L}_{\mathsf{W}_1}).$$

- **Response**: After $\mathcal{A}$ decides that the query phase is over, $\mathcal{A}$ outputs a guess β' for the bit β and sends β' to the algorithm B. The algorithm B then sends β' to the challenger as its guess for γ.

In this game, the algorithm B can make a query $(W||j, \Gamma_\gamma(W||j))$ to the function $\Gamma_\gamma(.)$ for each unique keyword W that the adversary issues in j-th query. If the adversary makes at most $\frac{q}{u}$ queries with the average of u unique keywords in each query, the algorithm B gathers at most q queries to the function Γ_γ. The queries to the function Γ_γ are stored in the list L_f. Since the algorithm B uses $\mathcal{A}$'s guess, β', to represent its guess for the bit γ, the advantage of B in distinguishing the pseudorandom function from a true random function is equal to the advantage of $\mathcal{A}$ in distinguishing $Game_{Real}$ from $Game_1$. However, since $f(.)$ is a (t, q, ε_f) secure pseudorandom function (see 2.4), the probability of guessing γ correctly is at most $\frac{1}{2} + \varepsilon_f$. Therefore, the advantage ε of $\mathcal{A}$ in distinguishing the two games is $\varepsilon = \varepsilon_f$ which is negligible.

Lemma 2. *$Game_1$ and $Game_2$ are indistinguishable in the symmetric key security model, assuming that $\mathcal{E}(.)$ is a (t, q, ε_e) secure pseudorandom permutation function.*

Proof: Assume that there is an adversary $\mathcal{A}$ which distinguishes $Game_1$ from $Game_2$ with advantage ε. We show how to build an algorithm B which uses $\mathcal{A}$ to distinguish the pseudorandom permutation function from a true random bijection.

Assume that the challenger in advance of the game picks a random $\gamma \longleftarrow \{0,1\}$. Let Γ_0 be the pseudorandom permutation function $\mathcal{E}$: $\{0,1\}^v \times \{0,1\}^\delta \longrightarrow \{0,1\}^v$ and Γ_1 be a true random bijection function. The algorithm B uses the adversary $\mathcal{A}$ in the following game to guess the value γ:

- **Setup**: The challenger runs the $\mathsf{Keygen_s}(\sigma)$ algorithm to obtain the master secret key $msk = (k_f, k_e)$, where $k_f, k_e \longleftarrow \{0,1\}^\sigma$. The algorithm B picks a bit $\beta \longleftarrow \{0,1\}$ randomly. The adversary $\mathcal{A}$ prepares six lists $L_{\mathbb{W}_0}, L_{\mathbb{W}_1}, L_{W_0}, L_{W_1}, L_S, L_T$ which are initially empty. The algorithm B creates a list L_e which is initially empty.

- **Query I**: In this phase, $\mathcal{A}$ adaptively makes two types of queries. Assume that this is the jth query.

  – Searchable ciphertext queries: $\mathcal{A}$ sends two metadata items and their identifiers:

  $$(\mathbb{W}_0 = \{W_{0,1}, ..., W_{0,n}\}, ID_0), (\mathbb{W}_1 = \{W_{1,1}, ..., W_{1,n}\}, ID_1)$$

  to the algorithm B. The algorithm B sends the triple $(\mathbb{W}_\beta, ID_\beta, j)$ to the challenger. For each unique keyword $W_{\beta,j} \in \mathbb{W}_\beta$, the challenger runs the $\mathsf{EncSES1}(\mathbb{W}_\beta, msk, j)$ algorithm which outputs $(f_{k_f}(W||j), \mathcal{E}_r(ID_\beta), \Gamma_\gamma(r))$. The challenger then sends the triple $(f_{k_f}(W||j), \mathcal{E}_r(ID_\beta), \Gamma_\gamma(r))$ to the algorithm B. The algorithm B appends each $\Gamma_\gamma(r)$ to the list L_e and picks a random $R \longleftarrow \{0,1\}^m$. The algorithm B then sends $(R, \mathcal{E}_r(ID_\beta), \Gamma_\gamma(r))$ to $\mathcal{A}$ who appends $(R, \mathcal{E}_r(ID_\beta), \Gamma_\gamma(r))$ to the list L_S, and appends the metadata $\mathbb{W}_0$, and $\mathbb{W}_1$ to the lists L_S, $L_{\mathbb{W}_0}$ and $L_{\mathbb{W}_1}$ respectively. In case $\gamma = 0$, the function Γ_0 is a pseudorandom permutation function $\mathcal{E}_{k_e}(.)$ and the response is $\mathsf{S}^1_{W,j} = (R, \mathcal{E}_r(ID_\beta), \mathcal{E}_{k_e}(r))$. Otherwise, the response is $\mathsf{S}^2_{W,j} = (R, \mathcal{E}_r(ID_\beta), R')$, where R' is a random bijection $R' \longleftarrow \{0,1\}^v$.

  – Trapdoor queries: $\mathcal{A}$ sends two keywords (W_0, W_1) to the algorithm B who picks W_β and sends (W_β, j) to the challenger. The challenger runs $\mathsf{Trapdoor}(W_\beta, msk, j)$, which outputs $(f_{k_f}(W||1), ..., f_{k_f}(W||j))$, and sends $(f_{k_f}(W||1), ..., f_{k_f}(W||j))$ to the algorithm B. Given the response, B picks random $R_1, ..., R_j \longleftarrow \{0,1\}^m$ and sends $\mathsf{T}^1_{W_\beta,t} = \mathsf{T}^2_{W_\beta,t} = (R_1, ..., R_j)$ to $\mathcal{A}$. The adversary $\mathcal{A}$ then appends $(R_1, ..., R_j)$ to the list L_T, and appends W_0 and W_1 to L_{W_0}, and L_{W_1} respectively.

  Here the condition on the keywords of the Query phase is that

  $$\mathsf{AccessPattern}(\mathsf{L}_{\mathbb{W}_0}, \mathsf{L}_{\mathsf{W}_0}) = \mathsf{AccessPattern}(\mathsf{L}_{\mathbb{W}_1}, \mathsf{L}_{\mathsf{W}_1}).$$

- **Response**: After $\mathcal{A}$ decides that the query phase is over, $\mathcal{A}$ outputs a guess β' for β and sends β' to the algorithm B. The algorithm B then sends β' to the challenger as his guess for γ.

In this game, the algorithm B can make a query $(W||j, \Gamma_\gamma(W||j))$ to the function $\Gamma_\gamma(.)$ for each unique keyword W of the j-th query. If the adversary makes at most $\frac{q}{u}$ queries with u distinct keywords in each query, the algorithm B gather at most q queries to the function Γ_γ which are stored in the list L_e. If $\mathcal{A}$ can distinguish $Game_1$ from $Game_2$, the algorithm B can use this response to distinguish whether Γ is a pseudorandom permutation function or a true bijection. However, since $\mathcal{E}(.)$ is a (t, q, ε_e) secure pseudorandom permutation, the probability of guessing γ is at most $\frac{1}{2} + \varepsilon_e$. Therefore, the probability of distinguishing the two games is at most $\frac{1}{2} + \varepsilon_e$.

Lemma 3. *$Game_2$ and $Game_3$ are indistinguishable.*

Proof: In $Game_2$ the challenger responds to each searchable ciphertext query with $\mathsf{S}^2_{W,t} = (R, C_2, R')$ and the trapdoor query with $\mathsf{T}^2_{W,t} = (R_1, ..., R_t)$. Since the element $C_2 = \mathcal{E}_r(I_{W,t})$ is encrypted with a random value, and the element C_3 is replaced by a random value, C_2 is a one time pad. Therefore, $Game_2$ and $Game_3$ are unconditionally indistinguishable.

Theorem 1. *SES1 is $(t, \varepsilon_f + \varepsilon_e, \frac{q}{u})$ secure in the symmetric key security model, where u is the average number of unique keywords in each query, if $f(.)$ is a (t, ε_f, q) secure pseudorandom function and $\mathcal{E}(.)$ is a (t, ε_e, q) secure pseudorandom permutation function.*

By Lemmas 1, 2, and 3, the advantage of the adversary $\mathcal{A}$, which is allowed to make at most $\frac{q}{u}$ queries with the average of u distinct keywords in each query, in distinguishing $Game_{Real}$ from $Game_3$ is $\varepsilon_f + \varepsilon_e$. Since the searchable ciphertext and the trapdoor of $Game_3$ are random values, the advantage of the adversary in distinguishing the searchable ciphertext $S_{W,t}$ and the trapdoor $T_{W,t}$ from random is $\varepsilon_f + \varepsilon_e$ which is negligible. This proves Theorem 1.

3.3.4 Construction of SES2

We now present the SES2 scheme which allows searching non-interactively at the cost of higher computational complexity for the trapdoor. In SES1, the key used to encrypt the identifiers $I_{W,t}$ is a random value r, which is stored in the searchable ciphertext in encrypted form, $\mathcal{E}_{k_e}(r)$. Therefore, in order for the client to search for the keyword W, the client should retrieve the encrypted key, $\mathcal{E}_{k_e}(r)$, of each $S_{W,t}$. The client then decrypts the key r and sends r to the server which allows decrypting the identifiers $I_{W,t}$. In SES2 the key used to encrypt $I_{W,t}$ is computed using the pseudorandom function $K_{W,t} = f_{k_e}(W||t)$ which can be computed later for the search. To search for the keyword W, the client computes the key $K_{W,i} = f_{k_e}(W||i)$ of each $I_{W,i}$, $i = 1..., t$ and inserts the keys in the trapdoor $T_{W,t}$. Given the trapdoor $T_{W,t}$, the server can decrypt the identifiers $I_{W,1}$,...,$I_{W,t}$ using the trapdoor non-interactively.

SES2 uses a pseudorandom function $f{:}\{0,1\}^* \times \{0,1\}^\delta \longrightarrow \{0,1\}^m$ for some value m, and a pseudorandom permutation function $\mathcal{E}{:}\{0,1\}^v \times \{0,1\}^m \longrightarrow \{0,1\}^v$ for some v.

$\mathsf{KeygenSES}(\sigma)$**:** Given the security parameter σ, output the master secret key $msk = (k_f, k_e)$, where $k_f, k_e \in \{0,1\}^\sigma$.

$\mathsf{EncSES2}(\{(\mathbb{W}_1, ID_1), ..., (\mathbb{W}_n, ID_n)\}, msk, t)$**:** Given the metadata items and their identifiers, $\{(\mathbb{W}_1, ID_1), ..., (\mathbb{W}_n, ID_n)\}$, the master secret key msk, and the counter t, for each unique keyword $W \in \{\mathbb{W}_1, ..., \mathbb{W}_n\}$:

1. compute the set $I_{W,t} = \{ID_j | W \in \mathbb{W}_j\}$,
2. compute an encryption key $K_{W,t} = f_{k_e}(W||t)$,
3. compute the searchable ciphertext $S_{W,t} = (f_{k_f}(W||t), \mathcal{E}_{K_{W,t}}(I_{W,t}))$.

After computing a searchable ciphertexts for all the unique keywords that occur in the metadata items $\{\mathbb{W}_1, ..., \mathbb{W}_n\}$, the algorithm sets $t = t + 1$.

Intuitively, the first component of the ciphertext makes a commitment to the keyword W, and the second component stores the identifiers $I_{W,t}$ in encryption form.

$\mathsf{TrapdoorSES2}(W, msk, t)$**:** Given the keyword W and the master secret key msk, output the trapdoor

$$T_{W,t} = ((f_{k_f}(W||1), f_{k_e}(W||1)), ..., (f_{k_f}(W||t), f_{k_e}(W||t))).$$

$\mathsf{SearchSES2}(T_{W,t}, S)$**:** Given the trapdoor T_W and the searchable ciphertexts S, for $i = 1, ..., t$ search the first component of the searchable ciphertexts for $f_{k_f}(W||i) \in T_W$. If $f_{k_f}(W||i)$ occurs, decrypt $I_{W,i}$ using $f_{k_e}(W||i) \in T_{W,i}$.

3.3.5 Proof of Security for SES2

The proof strategy is similar to the security proof strategy of SES1 (see Figure 3.2). Let $S_{W,t} = (C_1, C_2)$ be the searchable ciphertext of SES2. Let $T_{W,t} = ((T_1, T_1'), ..., (T_t, T_t'))$ be the trapdoor of SES2. Consider the following $\mathsf{SES^i}$, $(i = 1, 2, 3)$ schemes whose searchable ciphertexts $\mathsf{S}^i_{W,t}$ and trapdoors $\mathsf{T}^i_{W,t}$ are:

$$\begin{aligned}
\mathsf{SES}^1 &: \quad \mathsf{S}^1_{W,t} = (R, C_2), \mathsf{T}^1_{W,t} = ((R_1, T_1') ..., (R_t, T_t')) \\
\mathsf{SES}^2 &: \quad \mathsf{S}^2_{W,t} = (R, C_2), \mathsf{T}^2_{W,t} = ((R_1, R_1') ..., (R_t, R_t')) \\
\mathsf{SES}^3 &: \quad \mathsf{S}^3_{W,t} = (R, R'), \mathsf{T}^3_{W,t} = ((R_1, R_1') ..., (R_t, R_t'))
\end{aligned}$$

where R, R', and $R_1, R_1', ..., R_t, R_t'$ are random values. Assume that in the symmetric key security game, the challenger responds to each query with the searchable ciphertext and the trapdoor of the SES^3 scheme. Since all the elements of the searchable

ciphertext and the trapdoor of SES^3 are random, the advantage of the adversary in winning the game is zero. Let $Game_{Real}$ be the symmetric key security game defined in 2.2.1, where the response to each query is the searchable ciphertext and the trapdoor of the SES2 scheme. Let $Game_i$ be a security game the same as the symmetric key security game, except that the challenger constructs the searchable ciphertext and the trapdoor of the SES^i scheme to respond the queries. We show that these games are indistinguishable for the PPT adversary and therefore $Game_{Real}$ is indistinguishable from $Game_3$.

Lemma 4. *$Game_{Real}$ is indistinguishable from $Game_1$ in the symmetric key security model assuming that the pseudorandom function $f(.)$ is a (t, q, ε_f) secure pseudorandom function.*

Proof: Assume that there exists an adversary $\mathcal{A}$ which distinguishes $Game_{Real}$ from $Game_1$ with the advantage ε. We show how to build an algorithm B that uses $\mathcal{A}$ to distinguish the pseudorandom function from random function.

Assume that the challenger picks a random $\gamma \longleftarrow \{0,1\}$ before the game starts. If $\gamma = 0$, the challenger uses the pseudorandom function $f(.)$ to compute the searchable ciphertext and the trapdoor, otherwise, the challenger picks random values. Let Γ_0 be the pseudorandom function $f(.)$ and Γ_1 be the true random function. The algorithm B uses $\mathcal{A}$ in the following game to guess the value γ:

- **Setup**: The challenger runs the $\mathsf{Keygen_s}(\sigma)$ algorithm to obtain the master secret key $msk = (k_f, k_e)$, where $k_f, k_e \longleftarrow \{0,1\}^\sigma$. The algorithm B picks a bit $\beta \longleftarrow \{0,1\}$ randomly. The adversary $\mathcal{A}$ prepares six lists $L_{\mathbb{W}_0}, L_{\mathbb{W}_1}, L_{W_0}, L_{W_1}$, L_S and L_T which are initially empty. The algorithm B creates a list L_f which is initially empty.

- **Query I**: In this phase, $\mathcal{A}$ adaptively makes two types of queries. Assume that this is the jth query.

  – Searchable ciphertext queries: $\mathcal{A}$ sends two metadata items with their identifiers:

  $$(\mathbb{W}_0 = \{W_{0,1}, ..., W_{0,n}\}, ID_0), (\mathbb{W}_1 = \{W_{1,1}, ..., W_{1,n}\}, ID_1)$$

  to B. The algorithm B picks $(\mathbb{W}_\beta, ID_\beta)$ and sends $(\mathbb{W}_\beta, ID_\beta, j)$ to the challenger. For each unique keyword $W_{\beta,i} \in \mathbb{W}_\beta$, $(i = 1, .., n)$ the challenger runs the

  $$\mathsf{EncSES2}(\mathbb{W}_\beta, ID_\beta, msk, j)$$

  algorithm which outputs where $K_{W,j} = f_{k_e}(W||j)$. The challenger then sends the response

  $$(\Gamma_\gamma(W_{\beta,i}||j), \mathcal{E}_{K_{W,j}}(ID_\beta))$$

  to B. The algorithm B appends

  $$(W_{\beta,i}||j, \Gamma_\gamma(W_{\beta,i}||j))$$

to L_f and then sends

$$(\Gamma_\gamma(W_{\beta,i}||j), \mathcal{E}_{K_{W,j}}(ID_\beta))$$

to $\mathcal{A}$. In case $\gamma = 0$, the function Γ_0 is a pseudorandom function $f_{k_f}(.)$ and the response is the searchable ciphertext of SES2:

$$S_{W,j} = (f_{k_f}(W_{\beta,i}||j), \mathcal{E}_{K_{W,j}}(ID_\beta))$$

. Otherwise, the response is the searchable ciphertext of

$$\mathsf{SES^1} : \mathsf{S}^1(W, j) = (R, \mathcal{E}_{K_{W,j}}(ID_\beta))$$

, where R is a random value.

– Trapdoor queries: $\mathcal{A}$ sends two keywords (W_0, W_1) to B which picks W_β and sends (W_β, j) to the challenger. The challenger then runs the

$$\mathsf{TrapdoorSES2}(W_\beta, msk, j)$$

algorithm which outputs $((\Gamma_\gamma(W_\beta||1), f_{k_e}(W_\beta||1)), ..., (\Gamma_\gamma(W_\beta||j), f_{k_e}(W_\beta||j)))$ as the response. The challenger then sends the response to B. The algorithm B appends $((W_\beta||1, \Gamma_\gamma(W_\beta||1)), ..., (W_\beta||j, \Gamma_\gamma(W_\beta||j)))$ to the list L_f and sends the response to $\mathcal{A}$. In case $\gamma = 0$, the response is the trapdoor of SES2:

$$((f_{k_f}(W_\beta||1), f_{k_e}(W_\beta||1)), ..., (f_{k_f}(W_\beta||j), f_{k_e}(W_\beta||j))),$$

otherwise it is the trapdoor of $\mathsf{SES^1}$.

Here the condition on the keywords of the Query phase is that

$$\mathsf{AccessPattern}(\mathsf{L}_{\mathbb{W}_0}, \mathsf{L}_{\mathsf{W}_0}) = \mathsf{AccessPattern}(\mathsf{L}_{\mathbb{W}_1}, \mathsf{L}_{\mathsf{W}_1}).$$

- **Response**: After $\mathcal{A}$ decides that the query phase is over, $\mathcal{A}$ outputs a guess β' for β and sends β' to the algorithm B. The algorithm B then sends β' to the challenger as its guess for γ.

In this game, the algorithm B can make a query $(W||j, \Gamma_\gamma(W||j))$ to the function $\Gamma_\gamma(.)$ for each unique keyword W that the adversary issues in j-th query. If the adversary makes at most $\frac{q}{u}$ queries with the average of u unique keywords in each query, the algorithm B gathers at most q queries to the function Γ_γ. The queries to the function Γ_γ are stored in the list L_f. Since the algorithm B uses $\mathcal{A}$'s guess, β', to represent its guess for the bit γ, the advantage of B in distinguishing the pseudorandom function from a true random function is equal to the advantage of $\mathcal{A}$ in distinguishing $Game_{Real}$ from $Game_1$. However, since $f(.)$ is a (t, q, ε_f) secure pseudorandom function (see 2.4), the probability of guessing γ correctly is at most $\frac{1}{2} + \varepsilon_f$. Therefore, the advantage ε of $\mathcal{A}$ in distinguishing the two games is $\varepsilon = \varepsilon_f$ which is negligible.

Lemma 5. *$Game_1$ and $Game_2$ are indistinguishable in the symmetric key security model assuming that $\mathcal{E}(.)$ is a (t, q, ε_e) secure pseudorandom permutation function.*

Proof: Assume that there is an adversary $\mathcal{A}$ which distinguishes the two games with the advantage ε. We show how to build an algorithm B that uses $\mathcal{A}$ to distinguish the pseudorandom permutation function from true random bijection. Assume that the challenger picks a random $\gamma \longleftarrow \{0,1\}$. Let Γ_0 be a pseudorandom permutation function $\mathcal{E}(.)$ and Γ_1 be a true random bijection function R'. The algorithm B uses the adversary $\mathcal{A}$ in the following game to guess the value γ:

- **Setup**: The challenger runs the $\mathsf{Keygen_s}(\sigma)$ algorithm to obtain the master secret key $msk = (k_f, k_e)$, where $k_f, k_e \longleftarrow \{0,1\}^\sigma$. The algorithm B picks a bit $\beta \longleftarrow \{0,1\}$ randomly. The dversary $\mathcal{A}$ prepares six lists $L_{\mathbb{W}_0}, L_{\mathbb{W}_1}, L_{W_0}, L_{W_1}$, L_S, and L_T which are initially empty. The algorithm B creates a list L_e which is initially empty.
- **Query I**: In this phase, $\mathcal{A}$ adaptively makes two types of queries. Assume that this is the jth query.
  - Searchable ciphertext queries: $\mathcal{A}$ sends two metadata items and identifiers:

    $$(\mathbb{W}_0 = \{W_{0,1}, ..., W_{0,n}\}, ID_0), (\mathbb{W}_1 = \{W_{1,1}, ..., W_{1,n}\}, ID_1)$$

    to B. The algorithm B picks $(\mathbb{W}_\beta, ID_\beta)$ and sends $(\mathbb{W}_\beta, ID_\beta, j)$ to the challenger. For each unique keyword $W_{\beta,j} \in \{\mathbb{W}_\beta\}$, the challenger runs the

    $$\mathsf{EncSES2}(W_{\beta,j}, ID_\beta, msk, j)$$

    algorithm which outputs

    $$(f_{k_f}(W||j), \mathcal{E}K_{W,i}(ID_\beta),$$

    where $K_{W,i} = f_{k_e}(W||j)$. The challenger then sends $(f_{k_f}(W||j), \mathcal{E}K_{W,i}(ID_\beta)$ to the algorithm B. The algorithm B picks a random $R \in \{0,1\}^m$ and sends $(R, \mathcal{E}K_{W,i}(ID_\beta)$ to $\mathcal{A}$. The response is the searchable ciphertext $\mathsf{S}^1{}_{W,j}$.
  - Trapdoor queries: $\mathcal{A}$ sends two keywords (W_0, W_1) to the algorithm B who picks W_β and sends (W_β, j) to the challenger. The challenger then runs the $\mathsf{TrapdoorSES2}(W_\beta, msk, j)$ which outputs

    $$((f_{k_f}(W||1), \Gamma_\gamma(W||1), ..., (f_{k_f}(W||j), \Gamma_\gamma(W||j))$$

    , The challenger then sends the response to the algorithm B. Given the response, B appends

    $$((W||1, \Gamma_\gamma(W||1)), ..., (W||1, \Gamma_\gamma(W||1)))$$

    to the list L_e. The algorithm B then picks random $R_1, ..., R_j$ and sends

    $$((R_1, \Gamma_\gamma(W||1)), ..., (R_j, \Gamma_\gamma(W||j))$$

to the adversary $\mathcal{A}$. In case $\gamma = 0$, the response from the algorithm B is the trapdoor of SES^1, otherwise it is the trapdoor of SES^2.

Here the condition on the keywords of the Query phase is that

$$\mathsf{AccessPattern}(\mathsf{L}_{\mathbb{W}_0}, \mathsf{L}_{\mathsf{W}_0}) = \mathsf{AccessPattern}(\mathsf{L}_{\mathbb{W}_1}, \mathsf{L}_{\mathsf{W}_1}).$$

- **Response**: After $\mathcal{A}$ decides that the query phase is over, $\mathcal{A}$ outputs a guess β' for β and sends β' to the algorithm B. The algorithm B then sends β' to the challenger as its guess for γ.

In this game, the algorithm B can make a query $(W||j, \Gamma_\gamma(W||j))$ to the function $\Gamma_\gamma(.)$ for each unique keyword W of the j-th query. If the adversary makes at most $\frac{q}{u}$ queries with u distinct keywords in each query, the algorithm B gather at most q queries to the function Γ_γ which are stored in the list L_e. If $\mathcal{A}$ can distinguish $Game_1$ from $Game_2$, the algorithm B can use this response to distinguish whether Γ is a pseudorandom permutation function or a true bijection. However, since $\mathcal{E}(.)$ is a (t, q, ε_e) secure pseudorandom permutation, the probability of guessing γ is at most $\frac{1}{2} + \varepsilon_e$. Therefore, the probability of distinguishing the two games is at most $\frac{1}{2} + \varepsilon_e$.

Lemma 6. *$Game_2$ and $Game_3$ are indistinguishable.*

Proof: In $Game_2$ the element C_2 is a one time pad. Therefore $Game_2$ and $Game_3$ are unconditionally indistinguishable.

Theorem 2. *SES2 is $(t, \varepsilon_f + \varepsilon_e, \frac{q}{u})$ secure, where u is the average number of unique keywords in each metadata query, if $f(.)$ is a (t, ε_f, q) secure pseudorandom function, and $\mathcal{E}(.)$ is a (t, ε_e, q) secure pseudorandom permutation.*

By Lemmas 4,5, and 6, the advantage of the adversary $\mathcal{A}$, which makes at most $\frac{q}{u}$ queries with the average of u distinct keywords in each query, in distinguishing $Game_{Real}$ and $Game_3$ is $\varepsilon_f + \varepsilon_e$. Therefore, the adversary can distinguish the searchable ciphertext and the trapdoor of SES2 from random with the advantage $\varepsilon_f + \varepsilon_e$ which is negligible. This proves the theorem.

3.4 Efficiency Comparison

Table 3.2 compares the complexity of the variants SES1 and SES2 of the SES schemes with the SI and SSE schemes which are the most prominent existing schemes. In this table,

- n is the number of the metadata items stored on the server per update,
- u is the average number of unique keywords per update,
- d is the total number of the keywords per update ($u \leq d$),

		SES1	SES2	SI	SSE
Computational Complexity	Searchable ciphertext	$2u\alpha + u\beta + u\lambda$	$2u\alpha + u\beta$	$2d\alpha$	$2ut\beta + ut\lambda$
	Trapdoor	$t\alpha$	$2t\alpha$	α	α
	Search	$t_u\alpha$	$t_u\alpha$	$2nt\alpha$	2α
Communication Complexity (bits)	Searchable Ciphertxt	$3mu$	$2mu$	mdn	$3mut$
	Trapdoor	mt	$2mt$	m	$2m$
	Search	mt_u	0	0	0
Storage Complexity (bits)	master secret key	2δ	2δ	δ	3δ
	Searchable Ciphertext	$3mut_u$	$2mut_u$	$2mdt$	$3mut_u$

Table 3.2: *Comparison of the complexity of our schemes with the SI and SSE schemes.*

- m is the number of bits in the output of the pseudorandom function,
- σ is the security parameter.
- α is the computational complexity of the pseudorandom function $f(.)$.
- β is the computational complexity of the pseudorandom permutation function $\mathcal{E}(.)$.
- λ is the complexity of the random generation.
- t is the total number of times that the database has been updated,
- t_u is the number of times that the database has been updated for each unique keyword ($t_u \leq t$). For example assume that the tuples $(M_1, \mathbb{W}_1), (M_2, \mathbb{W}_2), (M_3, \mathbb{W}_3)$, including the messages M_1, M_2, M_3 and the associated metadata items $\mathbb{W}_1, \mathbb{W}_2, \mathbb{W}_3$, have been stored on the server. Assume that only $\mathbb{W}_1$ contains the keyword W. Then $t = 3$ and $t_u = 1$ for the keyword W.

The table shows that our schemes, SES1 and SES2, are computationally more efficient than SI for the search. In our schemes, to search for a keyword, a number

of pseudo-random computations (α) is performed which is linear in the number of updates of the keyword (t_u). The number of pseudo-random computations that is performed for the search in SI is linear in the total number of updates (t). Since the number of updates for the keyword is always smaller than the total number of updates ($t_u \leq t$), the search in our scheme is unconditionally more efficient than the search in SI. The table also shows that while SSE is more efficient than SES1 and SES2 for the search, updating the database in SSE has a higher complexity compared to our schemes. In SSE, to store a new metadata item on the server, the client has to retrieve the whole database to alter them. This makes the complexity of the searchable ciphertext linear in t. The complexity of the searchable ciphertext in our scheme is constant. Therefore, SES1 and SES2 can perform both, the search and the update, efficiently, while existing schemes perform either, the search or the update, efficiently. By efficiently we mean that the complexity of the search and the update is not linear in the total number of updates, but it is either constant or linear in the number of updates for the keyword.

The price that we pay for the efficient search and the update is a higher complexity for the trapdoor. While the trapdoor in existing schemes has a constant complexity, the trapdoor in SES1 and SES2 has a complexity linear in t. The table also shows that SES1 has a lower trapdoor complexity compared to SES2. However, due to the interactive search, SES1 has a communication complexity for the search which is linear in the number of updates for the keyword. The search in other schemes is performed non-interactively which has zero bits.

		SES1	SES2	SI	SSE
Computational Complexity	Searchable ciphertext	$4u\alpha$	$3u\alpha$	$2u\alpha$	$3ut\alpha$
	Trapdoor	$t\alpha$	$2t\alpha$	α	α
	Search	$t_u\alpha$	$t_u\alpha$	$2t\alpha$	2α
Communication Complexity (bits)	Searchable Ciphertext	$3mu$	$2mu$	mu	$3mut$
	Trapdoor	mt	$2mt$	m	$2m$
	Search	mt_u	0	0	0

Table 3.3: *Comparison of the complexity of our schemes with the SI and SSE schemes under optimal circumstances for SI.*

In Table 3.3 we compare the computational complexity and the communication complexity of SES with SI and SSE for the case when the client stores one metadata item per update, i.e. $n = 1$ and $u = d$. This is the most efficient case for SI, because we choose the least possible values for n and d, which are the parameters of the complexity. We also assume that the pseudorandom, the pseudorandom permutation and the random generation functions have the same complexity ($\alpha = \beta = \gamma$). These assumptions simplify the comparison of the schemes. In this table we ignore the storage complexity since it is the same as the general case in Table 3.2. The table shows that even in the best case for SI, the computational complexity of our schemes is lower than SI since $t_u \leq t$. The lower the number of updates for the keyword, the lower the search complexity. The computational complexity of the search in SES1 and SES2 is the same as SI only if the keyword occurs in each update, i.e. $t_u = t$. However, in this case, the computational complexity of the searchable ciphertext in SI is lower compared to our schemes. The table also shows that the computational complexity of the search in SES1 and SES2 is lower than SSE if the keyword occurs in each metadata item once. The complexity of the searchable ciphertext in SSE is higher than the complexity of our schemes if the database is updated more than once.

3.5 Conclusion

We propose a novel symmetric key searchable encryption scheme called SES. The SES scheme has a lower computational complexity for the search compared to existing schemes that allow efficient update. Table 3.4 compares the SES scheme with SSE and other existing schemes from the point of view of the search computational complexity and the efficiency of the update. While our scheme can perform both, the search and the update of the database, efficiently, the trapdoor complexity grows linearly with the number of updates of the database. Therefore, the SES scheme is appropriate for the situations where the communication is cheap, the trapdoor is rarely constructed for each client, but the search and the update of the database occurs frequently. We now revisit the PHR scenario we explained in the Introduction Chapter. In this scenario Alice wants to store her medical records on a PHR server in an encrypted form such that the records can be retrieved selectively. In this case, it is a desire for the server to perform the search with the lowest possible cost. On the other hand, Alice is interested in using a scheme that allows updating the records at a low cost. If Alice uses a broadband internet connection and the records are not updated frequently, (e.g. records are updated once per month), the SES scheme is appropriate for this scenario. In this case, since the records are not updated frequently, the computational complexity of the trapdoor, which is linear in the number of updates, increases slowly. The broadband connection also makes the communication cheap. Therefore, using the SES scheme, the server can perform the search at a low cost, Alice can update the records at a low cost and the cost of the query is reasonable. We propose two variants of SES, SES1 and SES2, which differ in the complexity of the search and the trapdoor. SES1 performs the search interactively. SES2 performs the search non-interactively but at a cost of more complexity for the trapdoor. In

case the delay for the search is not critical, the SES1 scheme is more suitable than SES2.

	Existing schemes except SSE	SES	SSE
Computational complexity of Search is	Linear in total number of searchable ciphertexts	Linear in subset of searchable ciphertexts	constant
Update of database	Efficiently	Efficiently	Non-efficiently

Table 3.4: *comparison of SES with existing schemes and SSE.*

Chapter 4

Multi-user Searchable Encryption with Policy Enforcement

A multi-user searchable encryption scheme allows users of a group to store data on a server in an encrypted form in a way that the data can be retrieved selectively. Existing multi-user searchable encryption schemes share the whole database among the users. Therefore, if a user wants to restrict the access of other users to the stored data, an access policy should be defined. Traditional access control systems rely on an *honest* reference monitor to enforce the policy. However, if the data is sensitive, the access control policy might also be sensitive and leaks some information on the data to *honest-but-curious* reference monitor. In this chapter, we first present a high level framework for searching the database and enforcing a role based access control policy. Then, we propose the SEPE scheme which allows querying the encrypted database and enforcing the policy for the messages that satisfy the query. The policy enforcement is performed in such a way that the reference monitor learns as little information as possible about the roles. This chapter is based on a paper published in the proceedings of the 7th Information security Practice and Experience Conference (ISPEC11) [1].

4.1 Introduction

Multi-user searchable encryption allows any user to store encrypted data on the server, such that the data can be retrieved selectively. There are two types of multi-user searchable encryption schemes: the first type allows any user to store data on the server, but only one user can search the database. The second type allows all the

users of a predefined group to store data in encrypted form and search the database. This is the focus of this chapter.

There have been quite some efforts in proposing searchable encryption schemes in the multi-user setting. Existing schemes are applicable when the whole database is shared among the users. However, there are many applications where users define an access policy on the messages of the data to restrict the access of other users to the database. In this case the server should enforce the access control policy for any message that satisfies the query.

The standard implementation of an access control system has a reference monitor to enforce the policy. The reference monitor has to satisfy two strong assumptions. Firstly, the reference monitor is assumed to be honest in the sense that it faithfully enforces the policy. Secondly, the reference monitor is assumed *not* to be curious, in the sense that it does not leak information on the policy. We believe that the second requirement is unnecessarily strong. Normally, the reference monitor would be considered as a part of the trusted computing base. Therefore, even if the reference monitor learns some information about users, this should not be a concern. However, in a distributed system, the reference monitor is not a simple component, so we should be reluctant to trust it fully. This is merely an instance of the principle of the least privilege, which applied here states that the reference monitor should do its job with as little privileges as necessary. To achieve this, we propose a method to enforce the access control policy based on a weaker assumption than normal with the *honest-but-curious* reference monitor.

To illustrate the urgency of the problem, consider the following scenario. Imagine that users (e.g. Alice) store their personal health records (PHR) on a PHR server. The multi-user searchable encryption scheme allows users to store the records in encrypted form, such that the encrypted records can be retrieved selectively using a query. Now, assume that Alice is undergoing treatment for a mental problem, and that she wants to store the medical records of her mental problem on the PHR server, in a way that only a psychiatrist is permitted to access. Then, an access control policy is required which restricts the access of users who do not have the role psychiatrist. However, as the access control policy mentions the role psychiatrist, the reference monitor of the PHR server learns that Alice might have a mental problem. The problem is thus that the policy leaks to the curious PHR server that Alice may have a mental problem. The goal of this chapter is to propose a solution to this problem by blinding the policy and the user credentials from the honest-but-curious server.

Contribution First we propose a unifying framework for searching *and* policy enforcement by an honest-but-curious server. Then, we propose a provably secure scheme called SEPE, for Searchable Encryption with Policy Enforcement. The SEPE scheme permits the server to search on encrypted data and enforce a role based access control policy without learning much about the data and the policy. We will make more precise later what "much" actually means. The SEPE scheme is provably secure in the random oracle model. We describe practical issues for the security of cryptographi-

cally enforced access control policy schemes.

4.2 Related work

Our work is built on searchable encryption and cryptographically enforced access control techniques including attribute based encryption, hidden credentials and predicate encryption techniques. We discuss the most important related work in each of these fields.

Searchable encryption in the multi-user setting can be constructed from symmetric key or public key searchable encryption schemes. Curtmola et al. show how to construct a multi-user searchable encryption scheme from symmetric key searchable encryption. In this case the master secret key must be shared among the whole users of the group [19]. However, since sharing the master secret key increases the risk of disclosure, such schemes are only suitable for small groups of users. Hence in general, multi-user searchable encryption is constructed from public key searchable encryption schemes. Dong et al. [21], Bao et al. [43], and Ho et al. [29] propose multi-user searchable encryption schemes which allow every user of the group to store data and query the database using a unique key. Although searchable encryption allows for secure storage and retrieval of data, access control is not provided. For this other means are necessary, in particular Attribute Based Encryption seems expedient.

Hidden Credentials (HC) and Attribute Based Encryption (ABE) techniques allow a party to specify a policy upon encrypting a message, such that a requestor will be able to decrypt the message if and only if his attributes satisfy the policy associated with the message. HC schemes [23, 14, 34], which were proposed before ABE was introduced, suffer from collusion attacks [7]. A scheme is secure against a collusion attack if it does not allow multiple users who do not have the credential of a role to gain access to the role by colluding. Sahai and Waters were the first to propose ABE [37], which in turn is based on identity based encryption (IBE) [11]. Goyal et al. develop key policy ABE (KP-ABE) [28], where the policy is associated with the decryption key and the message is decrypted if the ciphertext contains sufficient attributes. Bethencourt et al. propose ciphertext policy ABE(CP-ABE) [7] where instead of associating the policy with the decryption key, the former is associated with the ciphertext and the attributes of the user are associated with the decryption key.

Predicate encryption (PE) is a class of attribute based encryption techniques which enforces the policy cryptographically. While in ABE the ciphertext must hide the message only, in PE, the ciphertext must hide both, the message and the policy or the attributes, from the adversary. Boneh and Waters propose the first construction of PE [12]. Katz et al. propose the most expressive predicate encryption scheme [31], called inner product encryption. Following the inner product encryption scheme, some PE schemes with an improvement on the efficiency are proposed [30, 35, 39].

Why do ABE and PE not solve the problem of enforcing policy by an honest but curious reference monitor? Since ABE schemes reveal the policy upon the storage, these schemes cannot address this problem. We now consider PE, which enforces the policy without revealing the policy. In PE, each user has a decryption key which is associated with a set of roles. To store data on the server, each message of the data is encrypted using its associated policy. The user then transforms the metadata item associated with the message to a searchable ciphertext. The user stores the encrypted message and the searchable ciphertext on the server. For a requestor to gain access a message, first a trapdoor is computed and is sent to the server. Given the trapdoor, the server searches for the desired encrypted messages to the requestor. The requestor then has to check for each encrypted message, whether it is decryptable using his decryption key. This approach has two drawbacks: firstly, it has extra communication and computational complexity because the requestor retrieves encrypted messages that are not decryptable. Secondly, the requestor learns that all the retrieved encrypted messages contain the queried keyword, even if some of them are not decryptable. This shows that PE schemes impose complexity and a leakage on the messages. Therefore, to enforce the policy efficiently and securely, the policy should be enforced by the server. To enforce the policy by the server using PE, the decryption key of the user should be revealed to the server which compromises the confidentiality of the messages. The purpose of this chapter to propose a scheme which supports enforcing the policy by the server in a way that the policy does not reveal any information about the roles to the server. To the best of our knowledge, there is no related work on the problem of enforcing access control using an honest but curious reference monitor.

4.3 Blinded server

A blinded server is a server that provides services to the client without knowing the content of the stored messages. A blinded server consists of a blinded database and a blinded reference monitor. A blinded database supports storage and searching of encrypted messages, and the blinded reference monitor enforces role based access control policies. In this section, we provide a high level specification of such a blinded server, which will be refined in subsequent sections. The high level specification of the blinded server provides a framework in which all prominent related work on searching in encrypted data can fit. The specification also shows that there is uncharted terrain in which blinded policy enforcement would fit. We propose the first scheme on blinded policy enforcement that fits into this uncharted terrain.

Consider a finite set of roles $\mathcal{R}$, messages $\mathcal{M}$, identifiers $\mathcal{I}$, and keywords $\mathcal{W}$. If the server, which consists of a database and a reference monitor, is honest and *not* curious, then no blinding is necessary. In the next subsection, we will show as a base line how the database can be queried and how access control can be enforced without blinding.

4.3.1 Unblinded database and reference monitor

The following set of functions determine the interface to an unblinded search and access control scheme:

- The unblinded database consists of a bijection im and a function iw, and the policy consists of a function ir:
  - $\mathsf{im} : \mathcal{I} \to \mathcal{M}$ is a bijection that looks up the message for a given identifier.
  - $\mathsf{iw} : \mathcal{I} \to \mathcal{P}(\mathcal{W})$ associates a (possibly empty) set of keywords, as a metadata item, to each identifier that can be used to query the database.
  - $\mathsf{ir} : \mathcal{I} \to \mathcal{P}(\mathcal{R})$ represents the unblinded access control policy. This function associates each identifier with a (possibly empty) set of authorized roles.
- When a user in a particular role $R \in \mathcal{R}$ queries the unblinded database for a certain keyword $W \in \mathcal{W}$, the query function rw returns all relevant messages to which the user has access as determined by the unblinded policy.

  $$\mathsf{rw} : (\mathcal{R} \times \mathcal{W}) \to (\mathcal{I} \times \mathcal{P}(\mathcal{M}))$$
  $$\mathsf{rw}(R, W) = \{(ID, M) \mid ID \in \mathcal{I} \wedge R \in \mathsf{ir}(ID) \wedge W \in \mathsf{iw}(ID) \wedge M = \mathsf{im}(ID)\}$$

  The term $R \in \mathsf{ir}(ID)$ enforces the unblinded role base access control (RBAC) policy.

4.3.2 Blinded database and reference monitor

To search a curious database we extend the query with the trapdoor. The literature provides a wide selection of trapdoor constructions. In the same vein, to enforce access control with a curious reference monitor, we use some extra data in the form of a blinded credential, which can be used to check access rights of a blinded role.

Consider a finite set of secret keys $\mathcal{K}$, public keys $\mathcal{PK}$, trapdoors $\mathcal{T}$, blinded messages $\overline{\mathcal{M}}$, searchable ciphertexts $\mathcal{S}$, blinded roles $\overline{\mathcal{R}}$, credentials $\mathcal{C}$, and blinded credentials $\overline{\mathcal{C}}$.

The following set of functions determines the interface for a blind query and access control scheme:

- The user is assumed to be able to blind and unblind a message using a pair of functions as follows, where the keys $\mathcal{K}$ are appropriately chosen and kept secret by the user:
  - $\mathsf{km} : (\mathcal{K} \times \mathcal{M}) \to \overline{\mathcal{M}}$
  - $\mathsf{km}^{-1} : (\mathcal{K} \times \overline{\mathcal{M}}) \to \mathcal{M}$
- The user is assumed to be able to transform the metadata to a searchable ciphertext and to generate a trapdoor that can be used to search for the keyword using the functions kw and ws respectively:

- $\mathsf{kw} : (\mathcal{K} \times \mathcal{W}) \to \mathcal{T}$
- $\mathsf{ws} : (\mathcal{PK} \times \mathcal{P}(\mathcal{W})) \to \mathcal{S}$

- The user is assumed to be able to blind a role and to generate a blinded credential that can be used to enforce the RBAC policy using a function:
  - $\mathsf{kr} : (\mathcal{K} \times \mathcal{R}) \to (\overline{\mathcal{C}} \times \overline{\mathcal{R}})$

- The blinded database consists of a bijection $\overline{\mathsf{im}}$ and a function $\overline{\mathsf{iw}}$, and the blinded policy consists of a function $\overline{\mathsf{ir}}$ as follows:
  - $\overline{\mathsf{im}} : \mathcal{I} \to \overline{\mathcal{M}}$ looks up the blinded message for a given message identifier. It must not be possible for the message identifier to leak information on the message.
  - $\overline{\mathsf{iw}} : \mathcal{I} \to \mathcal{S}$ associates a (possibly empty) searchable ciphertext with each message identifier.
  - $\overline{\mathsf{ir}} : \mathcal{I} \to \mathcal{P}(\overline{\mathcal{R}})$ represents the blinded access control policy. This function associates each message identifier with a (possibly empty) set of blinded roles.

- If a user in a particular role $R \in \mathcal{R}$, and with a particular key $K \in \mathcal{K}$ queries the blinded database for a certain keyword $W \in \mathcal{W}$, the query function $\overline{\mathsf{krw}}$ returns all relevant messages to which the user has access as determined by the blinded policy:

$$
\begin{aligned}
&\overline{\mathsf{krw}} : (\mathcal{K} \times \mathcal{R} \times \mathcal{W}) \to \mathcal{P}(\mathcal{I} \times \mathcal{M}) \\
&\overline{\mathsf{krw}}(K, R, W) = \\
&\{(ID, \mathsf{km}^{-1}(K, \overline{M})) \mid \\
&\boxed{ID \in \mathcal{I} \wedge \overline{R} \in_{\overline{C}} \overline{\mathsf{ir}}(ID) \wedge T_W \in_t \overline{\mathsf{iw}}(ID) \wedge \overline{M} = \overline{\mathsf{im}}(ID)}\} \\
&\textsf{where} \\
&(\overline{C}, \overline{R}) = \mathsf{kr}(K, R) \\
&T_W = \mathsf{kw}(K, W)
\end{aligned}
$$

The two terms below the where clause calculate the blinded credential $\overline{C}$ for the role and the trapdoor T_W for the blinded keyword. The set membership operations are adorned with a subscript indicating which credential/trapdoor to use when comparing keywords or roles. The term in the box is calculated by the server. The remaining calculations are performed by the user, hence the server never sees unblinded roles, keywords or messages, nor any keys.

The set of functions $\overline{\mathsf{im}}$, $\overline{\mathsf{iw}}$, and $\overline{\mathsf{ir}}$ will be used throughout the chapter to represent the database and the policy.

4.4 SEPE: Blinding the Server

In this section we present the SEPE scheme which blinds the server by blinding the database and the reference monitor.

4.4.1 Blinding the Database

To blind the database a multi-user searchable encryption scheme is deployed which is constructed from a public key searchable encryption scheme. In order to allow any user of a group to encrypt a message in a way that any user can search the database and decrypt the message without using the master secret key, a central authority (CA) is required. The CA keeps the master secret key and distributes decryption keys among the users, such that any user can encrypt and query the message using a unique key while the message can be decrypted by other users. We discuss the trust issues of the CA in section 4.4.4.

Let $\mathsf{Keygen_P}$, $\mathsf{Enc_P}$, $\mathsf{Trapdoor_P}$, and Dec be the algorithms of a public key searchable encryption scheme. Let $\mathcal{U}$ be the users of the group. A multi-user searchable encryption scheme consists of the following algorithms.

- $\mathsf{Keygen_P}(\sigma)$: given the security parameter σ, generate the master secret key $msk \in \mathcal{K}$, where $\mathcal{K} \in \{0,1\}^\sigma$, and the public parameters $param \in \mathcal{PK}$. This algorithm is invoked by the CA.

- $\mathsf{UserKeyGen}(msk, U)$: given the master secret key and the user identity $U \in \mathcal{U}$, output a unique private/public key pair $(K_U, K'_U) \in (\mathcal{K} \times \mathcal{K})$ for the user U. This algorithm, which is invoked by the CA, sends user's private key K_U to the user and stores user's public key K'_U on the server.

- Blinding a message M consists of the following sub-algorithms which are invoked by the user, say U, who wants to store M on the server:

  - $\mathsf{MessageKeyGen}(\sigma)$: given the security parameter σ output an encryption key $K_M \in \mathcal{K}$.
  - $\mathsf{MessageBlinding}(K_M, M)$: given the encryption key $K_M \in \mathcal{K}$ and the message $M \in \mathcal{M}$ output a blinded message $\overline{M} \in \overline{\mathcal{M}}$.
  - $\mathsf{KeyBlinding}(K_M, K_U)$: given a key pair $(K_M, K_U) \in (\mathcal{K} \times \mathcal{K})$, transform K_M to a blinded form $\overline{K_{M,U}} \in \overline{K}$ using K_U.
  - $\mathsf{Enc_P}(\mathbb{W}, param)$: given the public parameters $param \in \mathcal{PK}$ and the metadata $\mathbb{W} \in \mathcal{P}(\mathcal{W})$ output a searchable ciphertext $S_\mathbb{W} \in \mathcal{S}$.

  Assume that user U' wants to retrieve the encrypted messages that satisfy the query:

- $\mathsf{Query}(K_{U'}, W)$: given the user private key $K_{U'} \in \mathcal{K}$ and the keyword $W \in \mathcal{W}$ output a query $T_{W,U'} \in \mathcal{T}$. This algorithm is invoked by the requestor.

- $\mathsf{QueryTrans}(K'_{U'}, T_{W,U'})$: given the query $T_{W,U'}$ and requestor's public key $K'_{U'}$ output a trapdoor $T_W \in \mathcal{T}$. This algorithm is invoked by the server.
- $\mathsf{Dec}(S_{\mathbb{W}}, T_W)$: given the searchable ciphertext $S_{\mathbb{W}} \in \mathcal{S}$ and the trapdoor $T_W \in \mathcal{T}$ output "1" if $W \in \mathbb{W}$. This algorithm is invoked by the server.
- Retrieval consists of two sub-algorithms:
  - $\mathsf{KeyTrans}(\overline{K_{M,U}}, K'_U, K'_{U'})$: given a blinded key $\overline{K_{M,U}} \in \overline{\mathcal{K}}$, user U's key K'_U, which is stored on the server, and user U''s key $K'_{U'} \in \mathcal{K}$, where $U' \in \mathcal{U}$, transform $\overline{K_{M,U}}$ to a blinded form $\overline{K_{M,U'}} \in \overline{\mathcal{K}}$. This algorithm is invoked by the server.
  - $\mathsf{KeyUnblinding}(K_{U'}, \overline{K_{M,U'}})$: given a key $K_{U'} \in \mathcal{K}$ and the corresponding blinded key $\overline{K_{M,U'}} \in \overline{\mathcal{K}}$ recover the unblinded key $K_M \in \mathcal{K}$. This algorithm is invoked by the user U' who wants to retrieve the message M.
  - $\mathsf{MessageUnblinding}(K_M, \overline{M})$: given a blinded message $\overline{M} \in \overline{\mathcal{M}}$ and the corresponding blinding key $K_M \in \mathcal{K}$ output an unblinded message $M \in \mathcal{M}$. This algorithm is invoked by the user U' who wants to retrieve the message M.

Intuitively, in multi-user searchable encryption, each user has a unique private key. The server also stores a unique public key associated with each user which is generated by the CA. To store a message on the server, the user generates an encryption key and blinds the message using the key. To allow other users to decrypt the message, while the server cannot decrypt the message, the encryption key is blinded using the user unique key. To allow searching the database, the user also transforms the metadata item associated with the message to a searchable ciphertext, using the public parameters of the CA. The user then stores the blinded message, the blinded encryption key, the searchable ciphertext and the identifier of the message on the server. If a requestor wants to retrieve the messages whose associated metadata item contains a particular keyword, the user constructs a query and sends it to the server. Given the query, the server first transforms the query to a trapdoor. Then the server searches the searchable ciphertexts for the ones that match the trapdoor. Having found the desired blinded messages, the server transforms the encryption key of each message to a blinded form that can be unblinded by the requestor only. The server performs the transformation by the requestor's public key stored on the server. The requestor then unblinds the encryption key and unblinds the message.

In this chapter, we do not propose any multi-user searchable encryption construction. However, several multi-user searchable encryption schemes have been proposed [21, 43, 29] which can be used for the database blinding construction of SEPE. We show how the scheme proposed by Dong et al. [21], which is provably secure in the random oracle model, can be used as an instance of multi-user searchable encryption. In this scheme, the $\mathsf{Keygen_P}$, $\mathsf{Enc_P}$, and the Dec algorithms can be borrowed from any public key searchable encryption scheme. The $\mathsf{MessageKeyGen}$, $\mathsf{MessageBlinding}$, and $\mathsf{MessageUnblinding}$ algorithms can be any symmetric key encryption scheme which

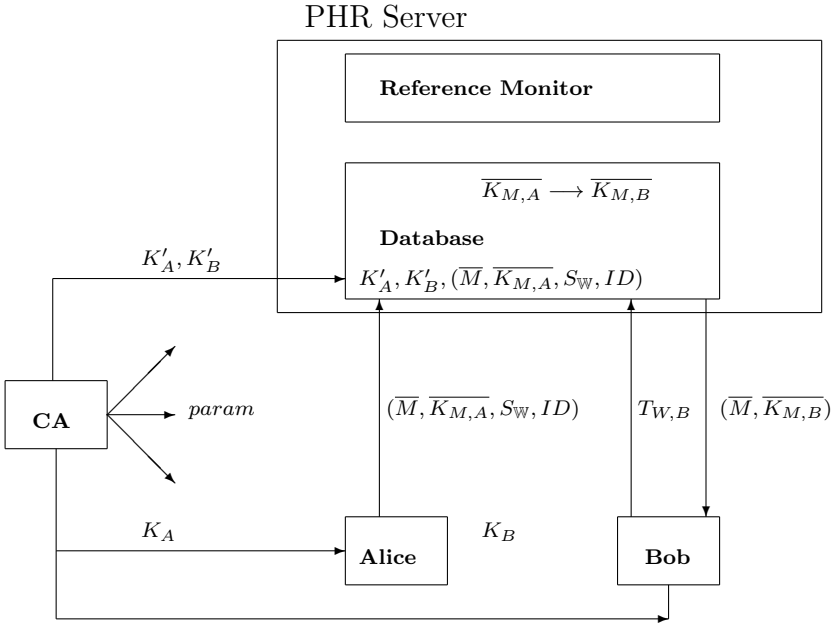

Figure 4.1: *Message exchange of the multi-user searchable encryption*

generates a key K_M using MessageKeyGen to encrypt and decrypt the message M using MessageBlinding and MessageUnblinding respectively. The UserKeyGen algorithm divides the master secret key msk into two parts K_U and K'_U for each user U, such that $K_U K'_U = msk$. This is performed such that

$$\mathsf{QueryTrans}(\mathsf{Query}(W, K_U), K'_U) = \mathsf{Trapdoor}_{\mathsf{P}}(W, msk).$$

The KeyBlinding and KeyTrans algorithms also work in the same way, such that $\mathsf{KeyBlinding}(msk, K_M) = \mathsf{KeyTrans}(\mathsf{KeyBlinding}(K_U, K_M), K'_U)$.

4.4.2 Database Blinding Example

Here we illustrate multi-user searchable encryption using the scenario from the introduction. In this scenario the patient Alice gives the Psychiatrist Bob access to her PHR. Figure 4.1 shows the message exchange of the blinded database using the example.

In this system the CA first generates a master secret key and the public parameters:

1. CA : $(msk, param) = \mathsf{Keygen}_{\mathsf{P}}(\sigma)$,
2. CA $\longrightarrow$ *: $param$.

As soon as Alice and Bob subscribe to the PHR system, the CA generates a unique

key pair for both, Alice and Bob. The CA sends the first element of the key pair to Alice and Bob, and stores the second element of the key pair on the server:

3. CA: $(K_A, K'_A) = \mathsf{UserKeyGen}(msk, A)$;
   $(K_B, K'_B) = \mathsf{UserKeyGen}(msk, B)$,
4. CA $\longrightarrow$ Alice : K_A;
   CA $\longrightarrow$ Bob : K_B;
   CA $\longrightarrow$ PHR : $(K'_B, B), (K'_A, A)$.

Assume that Alice wants to store the message M, which is associated with the metadata $\mathbb{W}$ and the identifier ID, on the server. Alice performs the following steps:

5. Alice : $K_M = \mathsf{MessageKeyGen}(\sigma)$;
   $\overline{M} = \mathsf{MessageBlinding}(M, K_M)$;
   $\overline{K_{M,A}} = \mathsf{KeyBlinding}(K_M, K_A)$;
   $S_{\mathbb{W}} = \mathsf{Enc_P}(\mathbb{W}, param)$;
6. Alice $\longrightarrow$ PHR : $(\overline{M}, \overline{K_{M,A}}, S_{\mathbb{W}}, ID)$.

Now, assume that Bob wants to retrieve M only if its associated metadata $\mathbb{W}$ contains the keyword W. Then:

7. Bob: $T_{W,B} = \mathsf{Query}(K_B, W)$,
8. Bob $\longrightarrow$ PHR: $T_{W,B}$,
9. PHR : $T_W = \mathsf{QueryTrans}(T_{W,B}, K'_B)$;
   $\mathsf{Dec}(S_{\mathbb{W}}, T_W) = 1$.

Since the keyword W occurs in the metadata $\mathbb{W}$, the database can retrieve the message as follows:

10. PHR: $\overline{K_{M,B}} = \mathsf{KeyTrans}(\overline{K_{M,A}}, K'_A, K'_B)$,
11. PHR $\longrightarrow$ Bob: $(\overline{K_{M,B}}, \overline{M})$,
12. Bob: $K_M = \mathsf{KeyUnblinding}(K_B, \overline{K_{M,B}})$;
    $M = \mathsf{MessageUnblinding}(K_M, \overline{M})$.

We now indicate the relation of the refinements explained above to the high level functions explained in 4.3.2 to blind the server. The relation is illustrated in Table. 4.1. The second column of the table shows the refinements for blinding the database. The next column shows the refinements for blinding the reference monitor which is presented in the following section. We describe the refinements of the functions which blind the reference monitor in section 4.4.3.

4.4.3 Blinding the Reference Monitor

We now show how to blind the roles and the credentials, and how to perform access decisions by using a blinded credential.

The refinement consists of introducing a master secret key and functions to ge-

	Refinement blinded database	Refinement blinded reference monitor
km	UserKeyGen MessageKeyGen MessageBlinding KeyBlinding	—
km^{-1}	UserKeyGen KeyTrans KeyUnblinding MessageUnblinding	—
ks	$\mathsf{Keygen}_\mathsf{P}$ Enc_P	—
kw	$\mathsf{Keygen}_\mathsf{P}$ Query QueryTrans	—
kr	—	$\mathsf{Keygen}_\mathsf{P}$ MessageKeyGen $\mathsf{UserKeyGen}_\mathsf{R}$ RoleBlinding CredentialBlinding
$\overline{\mathsf{krw}}$	—	Dec AccessDecision

Table 4.1: *Relation between the high level functions and their refinements for blinding a server*

nerate a blinded credential for each user, to generate a secret key for each user, to transform a role to a blinded form, and to make a decision about a request. However, before we present the scheme we describe its security requirement.

Security Definition. Informally, for a role blinding scheme to be secure, the blind role should not reveal any non-trivial information to the adversary about the role. The adversary also should not be able to learn any non-trivial information about the roles even after receiving blind credentials that cannot directly unblind the role.

The security of the blinding role scheme is defined as the following game between a challenger, who has the master secret key, and the adversary $\mathcal{A}$.

- **Setup**: The challenger sends the public parameters of the role blinding scheme to $\mathcal{A}$.

- **Query I**: In this phase $\mathcal{A}$ adaptively queries for the blinded credential of arbitrary roles. For any role $R \in \mathcal{R}$ that $\mathcal{A}$ queries the challenger creates a blinded credential $\overline{C}_R$ of the role R and sends $\overline{C}_R$ to the adversary $\mathcal{A}$.

- **Challenge**: Once $\mathcal{A}$ decides that the query phase is over, $\mathcal{A}$ picks two challenge roles (R_0, R_1) which have not been used in the query phase and send them to the challenger. Given the challenge roles, the challenger flips a coin $\beta \in \{0, 1\}$ and transforms role R_β to a blinded role $\overline{R}_\beta$. The challenger then sends $\overline{R}_\beta$ to $\mathcal{A}$.
- **Quer II**: This phase is the same as **Query I**. The adversary is still not allowed to query for the blind credentials of challenge roles (R_0, R_1).
- **Output**: Finally, the adversary $\mathcal{A}$ outputs its guess β' for the bit β and sends it to the challenger.

Intuitively, this game simulates an attack situation where the adversary can obtain the blinded credential of any role except the challenge roles. If the adversary has the blinded credential of the challenge roles, he can trivially win the game by checking the access decision between the blinded credentials and the blind roles. If the blinded credential reveals some information about the blinded role, the adversary can use this information to guess the bit β correctly. The adversary in this security model can be any user (including the reference monitor) who does not have the blind credential of the challenge role. Since this game allows the adversary to collect the blinded credential of any role (except for the challenge role), security in this model guarantees resistance against collusion attacks. In the collusion attack, multiple users who do not have a right blinded credential for accessing a message, try to gain access by a collusion.

Let $Adv_{\mathcal{A}} = \left| Pr[\beta = \beta'] - \frac{1}{2} \right|$ be the advantage of $\mathcal{A}$ in winning the game.

Definition 9 (Role Blinding Security)**.** *A role blinding scheme is secure if for all probabilistic polynomial time (PPT) adversaries $\mathcal{A}$, $Adv_{\mathcal{A}} \leq \varepsilon(\sigma)$, where ε is a negligible function of σ.*

4.4.4 Construction

Here, we explain the scheme under the assumption that different users in the same role are indistinguishable. This assumption makes the scheme easier to understand, but less realistic. Therefore, in Section 4.4.6 we present an extension of the scheme which ensures that different users in the same role *can* be distinguished.

We assume that the CA in the system controls the roles and the assignment of users to roles, in the sense that:

1. the CA decides which roles there are
2. the CA decides which user has which role
3. the CA is the only authorized entity to generate master keys, user secret keys and blinded credentials.

For now we assume that at any time a user has only one role, and relies on the CA to enable users to assume different roles (but see Section 4.4.6).

The CA is assumed to be honest and not curious, which we believe is justified by the following argument. After generating a blind credential and a user key for each user, the CA can be kept off-line. Hence, although requiring a CA that is not curious seems at odds with using a curious server, since the CA is kept off-line most of the time, it would have no chance of benefitting from curiosity.

For the construction we make use of a bilinear map $e : \mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{G}_T$ (see 2.4). We also need a random oracle $H : \{0,1\}^* \rightarrow \mathbb{G}$ which maps any arbitrary length value to an element of the group $\mathbb{G}$ randomly. We assume that each user U has a pseudonym usr which is uniquely generated for U by the CA. The pseudonym usr does not reveal any information about U.

A blinded reference monitor consists of the following functions:

- $\mathsf{Keygen_P}(\sigma)$: given the security parameter σ, generate the master secret key $msk \in \mathcal{K}$, where $\mathcal{K} \in \{0,1\}^\sigma$, and the public parameters $param = (p, \mathbb{G}, \mathbb{G}_T, e(.,.), H(.)) \in \mathcal{PK}$. This algorithm is invoked by the CA.

- $\mathsf{UserKeyGen_R}(msk, U)$: given the master secret key $msk \in \mathcal{K}$ and the identity of a user $U \in \mathcal{U}$ with the pseudonym usr, first pick random $a_U \in \mathbb{Z}_p$ and then compute $b_U = \frac{msk}{a_U}$. The user private key is $k_U = g^{a_U} \in \mathbb{G}$. The CA then sends g^{a_U} to the user and stores (b_U, usr) on the server. This function is used exclusively by the CA.

- $\mathsf{CredentialBlinding}(msk, R)$: given the master secret key msk, and a role $R \in \mathcal{R}$ output a blinded credential $\overline{C_R} = H(R)^{msk} \in \overline{\mathcal{C}}$. This function is used exclusively by the CA. Since the identity of the user represented by k_U is not used here, all users in the same role have the same blinded credential. In Section 4.4.6 we show how users can be distinguished.

- $\mathsf{RoleBlinding}(k_U, R)$: given the user private key $k_U = g^{a_U}$, and a role $R \in \mathcal{R}$, generate a blinded role $\overline{R}$ which is executed by the user and the server as follows:

  - **User**: computes $e(g^{a_U}, H(R)^\gamma)$ and sends $(g^\gamma, e(g, H(R))^{a_U \gamma}, usr)$, to the server, where $\gamma \in \mathbb{Z}_p$ is a random value generated by the user.
  - **Server**: Given $(g^\gamma, e(g, H(R))^{a_U \gamma}, usr)$, the server first searches the database for (usr, b_U). After the server finds the tuple (usr, b_U), picks b_U and computes $(e(g, H(R))^{a_U \gamma})^{b_U} = e(g, H(R))^{\gamma msk}$. The server then stores $(\overline{R} = (g^\gamma, e(g, H(R))^{\gamma msk})$ in the database.

- $\mathsf{AccessDecision}(\overline{R}, \overline{C})$: Let $\overline{R} = (x, y)$. Given a blinded credential $\overline{C} \in \overline{\mathcal{C}}$, a blinded role $\overline{R} \in \overline{\mathcal{R}}$, and an identifier $ID \in \mathcal{I}$, output $True \in \{0,1\}$ if $e(\overline{C}, x) = y$ and $\overline{ir}(ID) = \overline{R}$, otherwise outputs $False \in \{0,1\}$. This function is used by the reference monitor.

Here, we have described the construction of our scheme for one role only. The scheme can be extended to support any expressive policy by blinding each role and then specifying the structure of the policy to the blind roles. However, in this case we blind the roles only and not the structure of the policy.

4.4.5 Role Blinding Example

In this section, we illustrate the role blinding construction of the SEPE scheme using the scenario from the introduction. We have the following parties involved. A group of users, including the patient Alice, who stores her medical records on a PHR server, and the psychiatrist Bob. We assume that there is one CA serving the users. Assume that Alice is assigned the role $R_{pnt} \in \mathcal{R}$ and that Bob is assigned role $R_{psy} \in \mathcal{R}$. In this example, for simplicity we do not show the message exchanges related to the blinding database construction explained in 4.3.2.

Let $A \longrightarrow B : M$ denote the event that A sends a message M to B. $A \longrightarrow * : M$ represents broadcasting a message.

The CA first performs step 1 of the database blinding example (see 4.4.2) to generate the master secret key msk. As soon as Alice and Bob subscribe to the PHR system, the CA first performs steps 2 and 3 of the database blinding example and sends a unique key to Alice and Bob for blinding the database. The CA then sends a blinded credential and a user key to Alice and Bob:

2.	CA :	$\overline{C_A} = \mathsf{CredentialBlinding}(msk, R_{pnt}) = H(R_{pnt})^{msk}$;
	:	$k_A = \mathsf{UserKeyGen_R}(msk, A) = (g^{a_A}, b_A)$;
	:	$\overline{C_B} = \mathsf{CredentialBlinding}(msk, R_{psy}) = H(R_{psy})^{msk}$;
	:	$k_B = \mathsf{UserKeyGen_R}(msk, B) = (g^{a_B}, b_B)$;
3.	CA $\longrightarrow$ Alice :	$(\overline{C_A}, k_A)$;
	CA $\longrightarrow$ Server :	(b_A, A);
	CA $\longrightarrow$ Bob :	$(\overline{C_B}, k_B)$;
	CA $\longrightarrow$ Server :	(b_B, B).

Assume that Alice wants to add a blinded message $\overline{M_A} \in \overline{M}$ with the identifier $ID \in \mathcal{I}$ and the searchable ciphertext $S_{\mathbb{W}} \in \mathcal{S}$ to the database, such that only users assigned with the role $R_{psy} \in \mathcal{R}$ are allowed to access the message. To do so, Alice first follows steps 4 and 5 of the blinded database example and stores the blinded message, the blinded key, the searchable ciphertext and the identifer of the message on the database. Alice then performs the following steps to blind the role Psychiatrist:

6.	Alice :	$(g^\gamma, e(g^{a_A}, H(R_{psy})^\gamma)) = \mathsf{RoleBlinding}(k_A, R_{psy})$, where γ is a random value.
7.	Alice $\longrightarrow$ PHR :	$((g^\gamma, e(g^{a_A}, H(R_{psy})^\gamma)), A)$.
8.	PHR :	$\overline{R} = (g^\gamma, (e(g^{a_A}, H(R_{psy})^\gamma))^{b_A}) = (g^\gamma, e(g, H(R_{psy}))^{\gamma msk})$

Now, assume that Bob wants to retrieve the messages whose associated metadata item contain the keyword W. Bob first performs steps 6, 7, and 8 of the blinding database example and sends the trapdoor to the server. Bob then sends his blinded credential to the server:

9.	Bob $\longrightarrow$ PHR:	$\overline{C_B}$

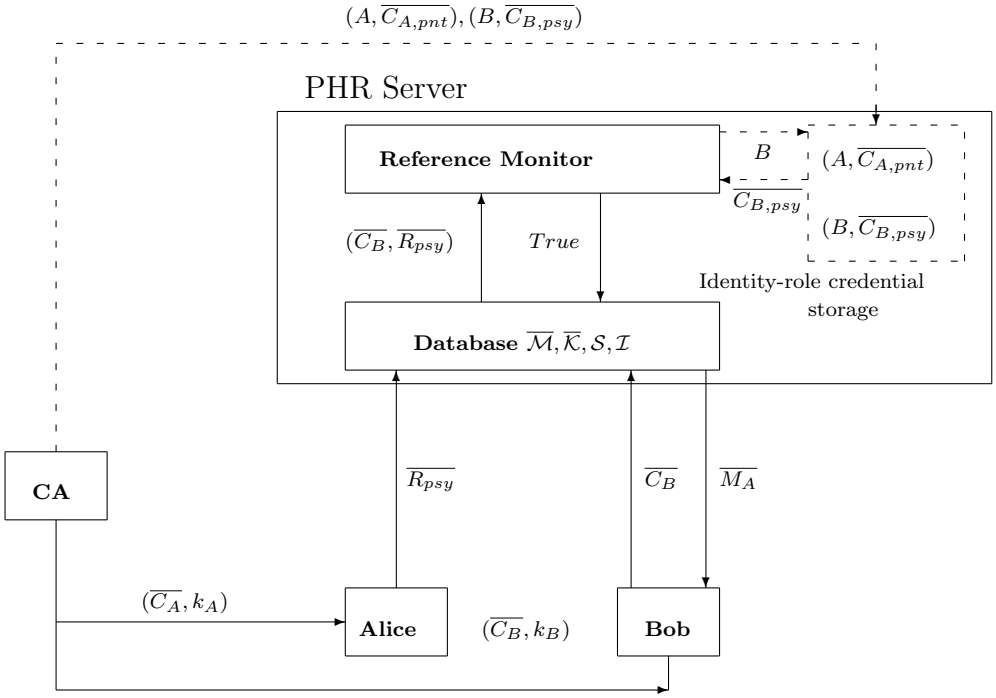

Figure 4.2: *Message exchange of the role blinding example*

Assume that $W \in \mathbb{W}$. Therefore, the search result shows that $\overline{M_A}$ is the blinded message that Bob is requesting. Now, the server has to enforce the policy to check whether Bob is allowed to receive the blinded message. If the policy defines that a psychiatrist can receive the message, the server sends Bob $\overline{M_A}$.

10. PHR $\longrightarrow$ Bob: $\overline{M_A}$, if $\mathsf{AccessDecision}(\overline{C_B}, \overline{R_{psy}}) = True$

The solid lines of Figure 4.2 illustrate the message flow of the example.

4.4.6 Extension

The construction presented in section 4.4.4 issues a blinded credential for each user which is based on the role of the user. This construction imposes the following limitations: i) to revoke a user from the group, the blinded credential of all users with the same role of the revoked user should be changed, ii) the risk of disclosure of blinded credentials is high because all users in the same role share the blinded credential.

Here, we propose a simple extension of the construction which issues a unique blinded, identity based credential for each user. In this extension the $\mathsf{Keygen_P}$,

the $\mathsf{UserKeyGen_R}$ and the $\mathsf{RoleBlinding}$ algorithms remain unaltered. However, the $\mathsf{CredentialBlinding}$ and the $\mathsf{AccessDecision}$ algorithms are modified. The idea is to divide the blinded user agnostic role credential by the factor b_U representing the user identity, and then during the access decision to multiply the user specific credential by b_U again. The refinements of the algorithms $\mathsf{CredentialBlinding}$ and $\mathsf{AccessDesicion}$ are as follows:

- $\mathsf{CredentialBlinding}(msk, R, U)$: Given the master secret key msk, a role R, and the identity of the user $U \in \mathcal{U}$ with the pseudonym usr, the CA picks a random $t_U \in \mathbb{Z}_p$ and computes a blind identity-role credential $\overline{C_{U,R}} = (H(R)^{msk})^{t_U b_U}$. The CA sends $\overline{C_{U,R}} = (H(R)^{msk})^{t_U b_U}$ to the user U and $\overline{C_U} = (t_U b_U, usr)$ to the server for storage.
- $\mathsf{AccessDesicion}(\overline{R}, \overline{C})$: Given the tuple consisting of role-identity credential and a user pseudonym $((H(R)^{msk})^{t_U b_U}, usr)$, the server first searches the identity credential $\overline{C_U}$ for the $t_U b_U$ associated with the name usr. Having found $t_U b_U$, the server then computes

$$((\overline{C_{U,R}})^{\frac{1}{t_U b_U}}) = \overline{C_R} = H(R)^{msk}$$

to obtain the blind role credential $\overline{C_R} = H(R)^{msk}$. Let $\overline{\mathsf{ir}}(ID) = \overline{R}$ and let $\overline{R} = (x, y)$. The reference monitor grants access to the requested message $\overline{M}$ iff $e(H(R)^{msk}, x) = y$, and $\overline{\mathsf{im}}(ID) = \overline{M}$.

The dashed lines of Figure 4.2 shows the message exchange of the extended construction.

Theorem 3. *The blinded policy scheme is semantically secure in the random oracle model assuming that decisional bilinear Diffie-Hellman (DBDH) is intractable.*

Proof: Suppose there exists a PPT adversary $\mathcal{A}$ that can break the security of the blinding policy scheme, i.e. $\mathcal{A}$ has a non-negligible advantage ε in breaking the scheme. We show that we can build an algorithm $\mathcal{B}$ that uses $\mathcal{A}$ to solve the DBDH problem in $\mathbb{G}_T$.

Assume that a challenger selects a bilinear group $\mathbb{G}$ of prime order p and chooses a generator $g \in \mathbb{G}$, the group $\mathbb{G}_T$ and an efficient bilinear map e. Then the challenger picks four random values $z_1, z_2, z_3 \in_R \mathbb{Z}_p^*$, to computes $Z_0 = e(g, g)^{z_1 b z_2 z_3}$. The challenger also picks random $Z_1 \in_R \mathbb{G}_T$. After flipping a fair coin $\omega \in_R \{0, 1\}$, the challenger hands the tuple $(g, g^{z_1}, g^{z_2}, g^{z_3}, Z_\omega)$ to $\mathcal{B}$. Algorithm $\mathcal{B}$'s goal is to guess the value ω correctly. In order to come up with a guess, $\mathcal{B}$ interacts with adversary $\mathcal{A}$ in the following role blinding security game. In this game B provides the adversary simulated public parameters and blind credentials, upon each request from the adversary. We call these provided parameters "simulated" because B generates these parameters without having access to the master secret key. However, the simulated public parameters and blind credentials have the same distribution with the normal scheme. Then, algorithm B generates a blind role using the value Z_ω. If the adversary can break the blind role, algorithm B decides that Z_ω is valid (i.e., $Z_\omega = e(g, g)^{z_1 z_2 z_3}$) and $\omega = 0$. Otherwise, Z_ω is random and $\omega = 1$.

- **Setup**: Algorithm $\mathcal{B}$ sends the adversary $\mathcal{A}$ the public parameters

$$param = (p, \mathbb{G}, e(.,.), H(.))$$

, where $H(.)$ is a random oracle controlled by $\mathcal{B}$, which outputs a response randomly from $\mathbb{G}$, and p is the order of the group $\mathbb{G}$. For the algorithm B to generate blind role using Z_ω, B assigns $g^{z_1} = g^{msk}$. In this case, B does not have access to msk directly.

- **Query I**: In this phase the adversary issues queries on blinded credentials. The adversary sends a role R to the algorithm B. Here we explain a challenge to simulate a blind credential for the role R. The blind credential for the role R is is computed as follow: $\overline{C_R} = H(R)^{msk}$. Assume that the random oracle outputs the value $H(R) = g^b \in \mathbb{G}$ upon the query for the role R. To simulate $H(R)^{msk}$, algorithm B should compute g^{bz_1}, where z_1 is the master secret key, chosen in the setup phase. However, B does not have access to neither z_1 nor b which makes it impossible to compute g^{bz_1}. Here, to address this problem we use the same random oracle proposed in [11] for our proof. Instead of a random oracle that outputs an element from $\mathbb{G}$, we consider a random oracle which is programmed as follows: before any response, the random oracle first flips a coin c which outputs 1 with probability x. Then based on the value of x the random oracle $H(.)$ sends it output as follows:

  - If $c = 0$, output a random $b \in \mathbb{Z}_p$.
  - If $c = 1$, output g^{z_2}.

  Algorithm $\mathcal{B}$ runs the random oracle $H(.)$ to simulate a blind credential. If $c = 1$, algorithm $\mathcal{B}$ aborts because B cannot simulate a blind credential in this case. Otherwise, $\mathcal{B}$ picks b and computes $\overline{C_R} = (g^{z_1})^b \in \mathbb{G}$, and sends $\overline{C_R}$ to $\mathcal{A}$. Since $g^{z_1} = g^{msk}$, the blind credential $\overline{C_R} = g^{z_1 b}$, is a correct blind credential for the role R.

- **Challenge**: When adversary $\mathcal{A}$ decides that the query phase is over, $\mathcal{A}$ generates two roles $R_0, R_1 \in \mathcal{R}$ on which he wishes to be challenged on, and sends the tuple to $\mathcal{B}$. The only condition is that R_0 and R_1 are not queried on the blinded credential query phase.

- **Query II**: This phase is identical to Query I. The adversary is not allowed to query for the roles (R_0, R_1).

- **Response**: Upon receiving (R_0, R_1) the algorithm $\mathcal{B}$ invokes the random oracle H to compute $H(R_0)$ and $H(R_1)$. The response occurs in one of the three folds below:

  1. There is no $H(R_\beta) = g^{z_2}$ for $\beta = 0, 1$ then algorithm $\mathcal{B}$ reports failure and the attack fails.
  2. There is one $H(R_\beta) = g^{z_2}$ for $\beta = 0, 1$, then algorithm $\mathcal{B}$ picks R_β.

3. For both R_0 and R_1, $H(R_0) = H(R_1) = g^{z_2}$, then algorithm $\mathcal{B}$ picks a bit $\beta \in \{0, 1\}$ at random to choose R_β.

Eventually, algorithm $\mathcal{B}$ picks g^{z_3} and sends (g^{z_3}, Z_ω) to the adversary $\mathcal{A}$ as a response to the challenge. Consider that in the case where $\omega = 0$, $Z_0 = e(g, g)^{z_1 z_2 z_3}$, which is a right blind role, otherwise Z_1 is picked randomly from group $\mathbb{G}_T$.

- QueryII: Adversary $\mathcal{A}$ repeats the Query phase. The only condition is that adversary $\mathcal{A}$ is not allowed to query for R_0 or R_1.

- **Guess** Given the tuple (g^{z_3}, Z_ω), adversary $\mathcal{A}$ outputs her guess β' for bit β and submits β' to $\mathcal{B}$. Having received β', algorithm $\mathcal{B}$ outputs his guess β' for β. If adversary $\mathcal{A}$'s output is correct, then algorithm $\mathcal{B}$ outputs $\omega' = 0$. Otherwise $\mathcal{B}$ outputs 1. We now compute the probability that $\mathcal{A}$'s guess is correct ($\beta = \beta'$). We first compute this probability with the condition that algorithm $\mathcal{B}$ does not abort during the query phase.

$$Pr[\beta = \beta'] = Pr[\beta = \beta' | \omega = 0] Pr[\omega = 0] + Pr[\beta = \beta' | \omega = 1] Pr[\omega = 1]$$

According to our assumption the advantage of $\mathcal{A}$ is $\frac{1}{2} + \varepsilon = Pr[\beta = \beta' | \omega = 0]$. In the case where $\omega = 1$, Z_1 is a random value and $\mathcal{A}$ does not get any information to guess β'. Hence, $Pr[\beta = \beta' | \omega = 1] = \frac{1}{2}$. Therefore,

$$Pr[\beta = \beta'] = \frac{1}{2}(\frac{1}{2} + \varepsilon + \frac{1}{2}) = \frac{1}{2}\varepsilon + \frac{1}{2}$$

The probability that algorithm $\mathcal{B}$ does not abort during the query phases is at least $\frac{1}{e}$, where e is Euler's number. The probability that algorithm $\mathcal{B}$ does not abort during the challenge phase is at least $\frac{1}{Q_T}$, where Q_T is the maximum number of queries that $\mathcal{A}$ issues. Hence, the probability that the random oracle $H(.)$ does not abort is at least $\frac{1}{eQ_T}$. Therefore $\mathcal{A}$'s advantage in breaking the scheme is $\frac{\varepsilon}{2eQ_T}$ which is required.

4.4.7 Efficiency

We discuss the complexity of the role blinding construction. Since there is no related work that addresses enforcing blind roles, we compare the efficiency of our construction to the base line provided by the standard reference monitor. Table 4.2 shows a break down of the complexity aspects into four categories. The first two categories describe the complexity incurred by the CA per user per role. The standard reference monitor only has to pick a credential per user per role, where the blinded reference monitor performs a number of computations as indicated. The third category shows that the user and the reference monitor have to perform a number of computations per role, which do not have a counterpart for the standard reference monitor. Finally, the fourth category shows that for each access, while the standard reference monitor

	Blind reference monitor			**Standard reference monitor**
Complexity	**CA**	**User**	**Reference monitor**	**Reference monitor**
Complexity of credential per user	assign a role + 1 exp per role	—	—	assign a role per role
Complexity of user key per user	1 exp + 1 RO per role	—	—	0
Complexity of role blinding	—	2 exp + 1 RO + 1 pairing per role	1 exp per role	0
Complexity of enforcement per access	—	—	1 pairing per role	1 comparison per role

Table 4.2: *Comparison of the complexity of the blinded reference monitor with the standard reference monitor. ("RO" - stands for random oracle computation and "exp" - stands for exponentiation.*

	Complexity	
	SEPE scheme	Predicate Encryption
Communication Complexity	α messages from the server to the user	α_u messages from the server to the user + Blinded credential $\overline{C}$ from the user to the server
Computational Complexity	Decrypt α messages	Decrypt α_u messages

Table 4.3: *Comparison of the complexity of the role blinding construction with predicate encryption constructions for enforcing the policy.*

does one comparison only, the blinded reference monitor has to perform a pairing computation.

We now compare the complexity of our scheme with existing schemes which enforce the policy cryptographically. For existing multi-user searchable encryption schemes (e.g. DGD [20]) the policy enforcement is performed using a predicate encryption scheme (see 4.2). In Table 4.3 we compare the complexity of the role blinding construction of SEPE with the complexity of predicate encryption schemes. In this table, α denotes the number of blinded messages that satisfy the query, and α_u denotes the number of the blinded messages that satisfy the query and the user is permitted to access ($\alpha_u \leq \alpha$). Let $|X|$ denote the number of bits in X. Since $|\alpha_u| \leq |\alpha|$, the computational complexity of our construction is lower compared to predicate encryption schemes. In case $|\alpha_u + \overline{C}| < |\alpha|$, our construction has a lower communication complexity compared to predicate encryption schemes. The table also shows that our scheme has a lower computational complexity than predicate encryption schemes.

4.5 Discussion of Practical Issues

The role blinding construction that we have presented in this chapter allows a database server to enforce the policy in such a manner that the server does not learn which user has which role. While this scheme offers better privacy than any other RBAC scheme, this enhanced privacy comes at a cost: as soon as one user reveals her private key to the server, the latter is able to discover all users who have the same role as the user who leaked her key. In point to point encryption schemes, a user revealing her private key leads to compromising the revealing user's information only. However, in our scheme, since the database is shared among all users, a user revealing her key will compromise other user's information also. Assume that a user in role U reveals her private key to the server. Since the user can blind any role

using her private key, the server can now blind any role too. Hence, upon receiving a blind credential, say $\overline{C}_R$, the server takes a role R from the universe of roles $\mathcal{R}$ and transforms it to a blind form $\overline{C}_R$ using the users private key. The server then checks if $\mathsf{AccessDecision}(\overline{C}_R, \overline{R}) = 1$. If the server learns that $\overline{C}_R$ is the blind credential of role R, the server has discovered that R maps onto $\overline{R}$, otherwise the server keeps trying other roles until the above equation holds.

Therefore, if a user reveals her private key to the server, a blind role remains secure only until the blind credential of the same role is queried. Revealing the private key of a user to the server thus impacts the security of the other users in the same role.

We now argue that the security drawback we described above is a common problem with all cryptographically enforced access control policy schemes (including attribute based encryption). The only countermeasure against the problem explained above is to prevent an adversary, who is the server in our case, from blinding roles. This can be achieved only if all users protect their private key, which in practice is not achievable.

Having to trust all users is a severe practical limitation. However, we can mitigate the risk of a user revealing her user private key in a number of ways:

False positive it would be possible to change the AccessDecision function, such that it generates false positives: Let AccessDecision with a certain probability output 1 for blind roles that do not match with a blind credential. This causes messages to be returned to the user that she will not be able to decrypt. Increasing the false positive rate increases the uncertainty for the server of deciding the role of each blind role. However, the performance of the system will decrease due to an increase in communication overhead and an increase in the complexity on the user side, where the false positives have to be discarded. Hence, introducing false positives creates a security performance trade-off.

CA involvement for blinding The task of the CA in our scheme is to issue a blind credential and a private key for each user when the user first enters the system. However, the CA could also be involved in blinding roles, for example by splitting the user private key in two parts. To blind a role, the user first blinds the role using her part of the key. The CA then completes the blinding using the other part the key. In this case both a user and the CA will have to collude for the server to learn which role belongs to which user. Involving the CA in every access control decision carries a heavy cost, which represents a sever drawback of this mitigating approach.

Restriction on blinding roles In many practical scenarios, each user should be able to blind only a subset of the available roles. If this is the case, a limited number of roles is revealed to the server when a user reveals her private key to the server. However, restricting the roles each user can blind requires an-* access control policy by the CA. Therefore, we are actually bootstrapping the access control policy of the server by the access control policy of the CA, which is at least inelegant.

Forward security To mitigate the effect of key leakage the scheme could be made forward secure in the sense that given a blind credential only the blind roles stored on the server beforehand can be enforced. More technically, each role is blinded with a user key and the time when the blind role is constructed. The user also re-blinds her blind credential, such that the AccessDecision function outputs 1 if (i) the blind credential and the blind roles match and (ii) the time of the storage is before the time of the query. However, this approach requires that the user, who requests a message, constructs the forward secrecy part of the blind credential with the right time.

The ideas presented above just mitigate the risk that a user reveals her private key. However, we believe that reducing the risk to zero is impossible and this is an inherent limitation of using purely cryptographic means to enhance the privacy of access control [38]. As we mentioned earlier this limitation comes from the fact that the security of cryptographic schemes relies on the ability of users to protect their private keys. For applications where a private key is used for a shared database, revealing a private key has a large impact on the security of the entire database.

4.6 Conclusion

We propose the SEPE scheme which allows a group of users to store data in encrypted form, such that the data can be retrieved selectively. In this scheme, users can define a role based access policy on their data, such that the server enforces the policy with learning as little information as possible about the roles. The SEPE scheme hides the roles by blinding the roles and the credentials of users. Our scheme reduces the communication complexity and the computational complexity to enforce the policy compared to PE techniques which allow enforcing policy cryptographically. Our scheme also resists the attacks of colluding users. However, the role blinding construction of SEPE remains secure as long as no user reveals her private key to the server. After a user reveals her private key to the server, the latter will be able to learn the blinded roles that match with a blind credential. We have discussed this drawback and argued that this is an inherent limitation of cryptographic tools on such privacy enhancing access policy schemes.

Chapter 5

Searchable Encryption Supporting Wildcards

A hidden vector encryption scheme is a public key searchable encryption scheme which supports wildcards in the trapdoor. These schemes are useful for a variety of applications to perform range, subset and conjunctive search over encrypted data. In this chapter we construct a hidden vector encryption scheme, called SEPS. Our scheme, which is provably secure in the selective security model, is more efficient than existing schemes for the search, and the trapdoor communication. We compare the efficiency of our scheme with the BW and the IP schemes which are the two prominent selectively secure HVE schemes. This chapter is based on the paper published in the Proceedings of the Seventh international Conference on Security and Cryptography for Networks, SCN 2010 [2].

5.1 Introduction

Most known public key searchable encryption schemes support searching for the exact keyword only. However, in many applications it is convenient to have some flexibility for the search by supporting wildcards in the query. Wildcards or "don't care entries" allow searching multiple keywords with one trapdoor. For instance, if the scheme supports wildcards "$\star$", the trapdoor associated with the keyword "199$\star''$ can search the database for the keywords ranging from "1990" to "1999". Subset and range queries are possible by supporting wildcards in the query.

Existing schemes that address searching keywords with wildcards use a technique called *hidden vector encryption* (HVE) [12]. In HVE, the keyword is considered as a vector of symbols. The search algorithm of HVE outputs the correct message, if all the non-wildcard symbols of both, the keyword associated with the searchable ciphertext and the queried keyword, are the same.

There have been some proposals for HVE schemes, most notably [12, 30, 31, 35, 39]. These schemes have in general three drawbacks: Firstly, most of them use *bilinear groups of composite order* which are (at least 50 times [22]) slower than bilinear groups of prime order (see 2.4). The few schemes that use bilinear groups of prime order, [30, 35] are only capable of working with binary alphabets. Secondly, in all these schemes the *complexity of the searchable ciphertext* is linear in the number of symbols of the keyword. Thirdly, the *complexity of the trapdoor* grows linearly in the number of the non-wildcard symbols of the queried keyword. Therefore, these schemes are not efficient to query for keywords that contain just a few wildcard symbols. The goal of this chapter is to propose an HVE scheme which has a lower complexity to perform the search.

Contribution In this chapter, we propose an HVE scheme, called SEPS, which allows searching keywords with wildcards on encrypted data. SEPS, which is provably secure in the selective security model, has the following advantages:

- SEPS uses bilinear groups of prime order. This makes the construction of SEPS more efficient than existing schemes which use bilinear groups of composite order. Our scheme can also take keywords over any alphabet, unlike [8, 30, 35] that only take binary symbols.
- The complexity of the trapdoor in our scheme is independent of the number of wildcards, while in earlier schemes the complexity of the trapdoor grows linearly in the number of *non*-wildcard symbols of the keyword.
- The storage and the communication complexity of the searchable ciphertext in our scheme is lower compared to existing schemes.

5.2 Related work

The first public key searchable encryption is due to Boneh et. al who propose a scheme called public key encryption with keyword search (PEKS). In this scheme everybody can construct the searchable ciphertext, but only the owner of the master secret key can create a trapdoor, which searches for a keyword. In [4], it is shown that PEKS has a close relation to anonymous identity-based encryption (AIBE), such that any AIBE scheme can be directly used as a searchable encryption scheme. Identity Based Encryption (IBE) is a public key encryption technique which allows for a party to use recipient's identity as a public key to encrypt a message. When an IBE scheme is anonymous, the scheme can hide both, the message and the identity, embedded in the ciphertext. Improved AIBE schemes have been proposed in [13, 24, 5, 15].

The schemes cited above are useful for the equality search only, i.e. a message can be decrypted if the keywords associated with the searchable ciphertext and the trapdoor are the same. Boneh and Waters propose the first construction of HVE [12], which supports wildcards in the trapdoor. Following [12], some more constructions for HVE are proposed with an enhancement either on security or efficiency [31, 39].

Finally, [30] provides a solution that is based purely on bilinear groups of prime order. However, this scheme accepts binary symbols only.

5.3 The SEPS Scheme

In HVE, a common approach to transform a message and its associated keyword to a searchable ciphertext is to blind the message using a random session key. The session key can be recovered for the search based on all the symbols of the keyword, while the trapdoor contains the information to cancel out the effect of the wildcard symbols. The searchable ciphertext and the trapdoor together can thus recover the session key.

In existing schemes, to construct the searchable ciphertext, every symbol of the associated keyword with the message is transformed to a group element. To construct the trapdoor also every non-wildcard symbol of the queried keyword is transformed to a group element. This transformation is carried out in a way that pairing operations between the corresponding elements of the trapdoor and the searchable ciphertext recovers the session key. This way of construction makes the complexity of the searchable ciphertext linear in the number of symbols of the keyword, and the complexity of the trapdoor and the search linear in the number of the non-wildcard symbols of the queried keyword.

In our scheme, we transform every symbol of both, the associated keyword with the message and the queried keyword, to a polynomial. The value of the polynomial at the wildcard symbols is zero, such that the session key can be recovered using the values of the polynomial at the non-wildcard symbols. This polynomial reduces the complexity of the search and the size of the trapdoor. However, a higher computational complexity for the searchable ciphertext and the trapdoor is imposed due to computing the polynomial.

5.4 Construction

Recall that data $D = (M_1, ..., M_n)$ is a sequence of n messages where each message M_i is associated with a keyword W_i. Every symbol of the keyword associated with the message is chosen over a fixed alphabet $\Sigma \subset \mathbb{Z}_p$. Every symbol of the queried keyword is chosen over the alphabet $\Sigma_\star$, where $\Sigma_\star = \Sigma \cup \{\star\}$. The SEPS scheme chooses an upper bound L on the maximum number of symbols in a keyword, and an upper bound Λ on the maximum number of wildcards in a queried keyword. While the upper bound Λ can be equal to L, which supports any number of wildcards, the performance of the scheme increases if $\Lambda \ll L$.

Let $\mathbb{G}$ and $\mathbb{G}_T$ be two multiplicative groups of prime order p. The SEPS scheme uses a bilinear pairing of prime order group $e : \mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{G}_T$ which maps two elements from the group $\mathbb{G}$ to an element from the group $\mathbb{G}_T$. Our scheme consists of the following algorithms:

- KeygenSEPS(σ): given the security parameter σ:

1. Pick $L+1$ random elements $v_0, u_1, ..., u_L \in \mathbb{Z}_p$,
2. Pick random $\alpha, \beta_1, \beta_2, (x_1, \ldots, x_\Lambda) \in \mathbb{Z}_p$,
3. Let $\Omega_1 = e(g, V_0)^{\alpha\beta_1}$ and $\Omega_2 = e(g, V_0)^{\alpha\beta_2}$,
4. Let $V_0 = g^{v_0}$, and $V_j = V_0^{x_j}$ for $j = 1, \ldots, \Lambda$,
5. Let $U_i = g^{u_i}$ for $i = 1, \ldots, L$.

The public parameters are:

$$param = \Big(\big(V_0, V_1, \ldots, V_\Lambda\big), \big(U_1, \ldots, U_L\ \big), g^\alpha, \Omega_1, \Omega_2, p, \mathbb{G}, \mathbb{G}_T, e(\cdot,\cdot) \Big)$$

The master secret key is:

$$msk = \big(p, g, \alpha, \beta_1, \beta_2, v_0, (x_1, \ldots, x_\Lambda), (u_1, \ldots, u_L)\big).$$

- $\mathsf{EncSEPS}(param, W, M)$: Let M be a message associated with the keyword W, and let $W = (w_1, \ldots, w_l) \in \Sigma^l$ be a sequence of l symbols of $w_1, ..., w_l$. Given the message M and the keyword W, the algorithm first picks two random values $r_1, r_2 \in \mathbb{Z}_p$, and then computes the searchable ciphertext:

$$S_{W,M} = \left(\hat{C} = M\Omega_1^{r_1}\Omega_2^{r_2}, \left(\begin{array}{c} C_0 = \big(V_0 \prod_{i=1}^{l} U_i^{w_i}\big)^{r_1+r_2} \\ C_1 = \big(V_1 \prod_{i=1}^{l} U_i^{i\ w_i}\big)^{r_1+r_2} \\ \vdots \\ C_\Lambda = \big(V_\Lambda \prod_{i=1}^{l} U_i^{i^\Lambda\ w_i}\big)^{r_1+r_2} \end{array} \right), \left(\begin{array}{c} g^{\alpha r_1} \\ g^{r_2} \end{array} \right) \right).$$

- $\mathsf{TrapdoorSEPS}(msk, W^\star)$: Let $W^\star = (w^\star{}_1, \ldots, w^\star{}_l) \in \Sigma_\star^l$ be a keyword consisting of l symbols of $w_1^\star, \ldots, w_l^\star$. Assume that $W^\star$ has $\lambda \leq \Lambda$ wildcards occurring at the symbols positions $J = \{j_1, \ldots, j_\lambda\}$. Given the keyword $W^\star$, the algorithm:

  1. picks a random $s \in \mathbb{Z}_p$,
  2. computes $s_1 = \beta_1 + s, s_2 = \beta_2 + s$,
  3. using Viète's formulas computes the coefficients $\{a_k\}$, $(k = 0, ..., \lambda)$ of a polynomial of degree λ which is evaluated to zero at the wildcard positions J,
  4. computes $X = \sum_{k=0}^{\lambda} x_k a_k$, where $x_0 = 1$
  5. computes the trapdoor:

$$T_{W^\star} = \left(\begin{array}{ll} T_0 & = g^{\frac{\alpha s}{X}} \\ T_1 & = V_0^{s_1} \prod_{i=1}^{l} U_i^{\frac{s}{X} \prod_{k=1}^{\lambda}(i-j_k) w_i^\star} \\ T_2 & = V_0^{\alpha s_2} \prod_{i=1}^{l} U_i^{\frac{\alpha s}{X} \prod_{k=1}^{\lambda}(i-j_k) w_i^\star} \\ & \{a_0, a_1, \ldots, a_\lambda\} \end{array} \right).$$

- $\mathsf{DecSEPS}(S_{W,M}, T_{W^\star})$: given a searchable ciphertext $S_{W,M}$ and the trapdoor $T_{W^\star}$, the algorithm decrypts the message M as follow:

$$M = \hat{C} \frac{e(T_0, \prod_{k=0}^{\lambda} C_k^{a_k})}{e(T_1, g^{\alpha r_1}) e(T_2, g^{r_2})}$$

The message exchange of SEPS is illustrated in Figure 5.1. It has the following differences with the message exchange of public key searchable encryption schemes shown in Figure 2.3

- The algorithms $\mathsf{Keygen_P}$, $\mathsf{Enc_P}$, $\mathsf{Trapdoor_P}$, and Dec are replaced by $\mathsf{KeygenSEPS}$, $\mathsf{EncSEPS}$, $\mathsf{TrapdoorSEPS}$, and $\mathsf{DecSEPS}$.
- The metadata in Figure 5.1 is one keyword only while in Figure 2.3 the metadata can take any number of keywords.
- The queried keyword $W^\star$ in Figure 5.1 can take wildcard symbols which is not supported in the queried keyword of Figure 2.3.
- The $\mathsf{DecSEPS}$ algorithm checks whether all the non-wildcard symbols of the ciphertext and queried keywords are the same, while the Dec algorithm checks whether $W \in \mathbb{W}$.

Intuition. In SEPS, the message M is blinded using the session key $\Omega_1^{r_1}\Omega_2^{r_2}$. To decrypt the message M, the $\mathsf{DecSEPS}$ algorithm should recover the session key using the searchable ciphertext and the trapdoor. To do so, the scheme transforms every symbol of the keywords W and $W^\star$ to a polynomial which cancels out the effect of wildcard symbols. The polynomial can recover the session key if all the non-wildcard symbols of keywords W and $W^\star$ are the same. Here, first we explain how this polynomial cancels out the effect of wildcards. Then, we explain how this polynomial is computed in our construction.

Consider the polynomial $\prod_{j\in J}(i-j)$, where i is the variable of the polynomial and J is the position of wildcards (e.g. $J = \{4, 5\}$ for '`04/**/2010`'). This polynomial is equal to zero at the wildcard positions where $i = j$, $j \in J$. Therefore, the following equation holds if all the non-wildcard symbols of both keywords W and $W^\star$ are the same:

$$\sum_{i=1}^{l} w_i \prod_{j\in J}(i-j) = \sum_{\substack{i=1 \\ i\notin J}}^{l} w_i^\star \prod_{j\in J}(i-j), \tag{5.1}$$

In Eq. 5.1, every symbol w_i and $w_i^\star$ of the keywords W and $W^\star$ respectively, is multiplied with the polynomial $\prod_{j\in J}(i-j)$. Since this polynomial is zero at the wildcard symbols ($i \in J$), the equation holds if all the non-wildcard symbols are the same: $w_i = w_i^\star$ for $i \notin J$. Therefore, this polynomial cancels out the effect of symbols at the wildcard positions, such that the comparison can be performed between the non-wildcard symbols only.

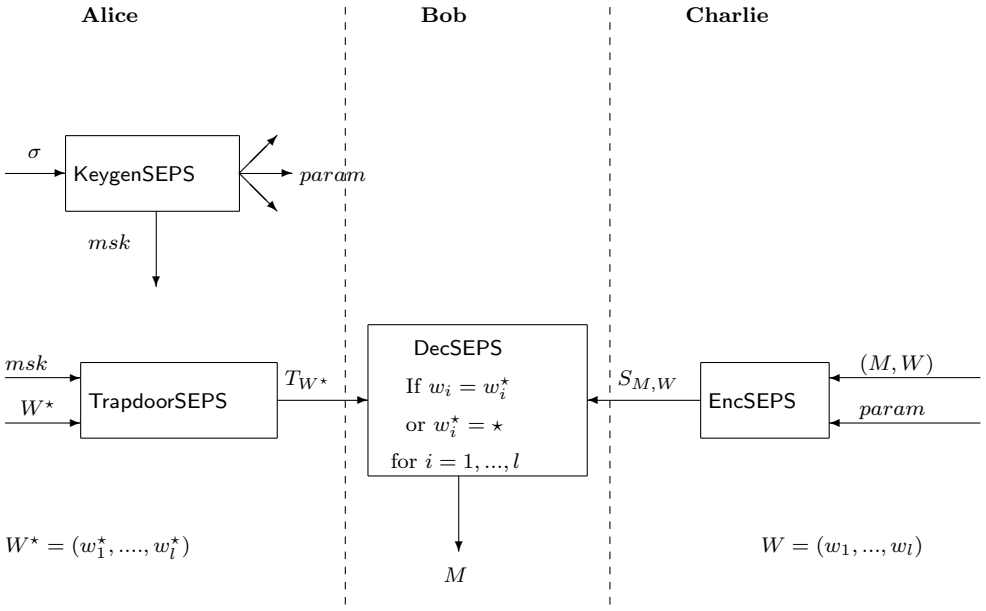

Figure 5.1: *The message exchange of SEPS. Alice first runs the* KeygenSEPS *algorithm which computes the master secret key* msk*, and the public parameters* $param$*. Alice then invokes the* TrapdoorSEPS *algorithm which computes the trapdoor* $T_{W^\star}$ *using* msk*. The trapdoor* $T_{W^\star}$ *is associated with the keyword* $W^\star$*, which contains some wildcard entries. Alice then delegates* $T_{W^\star}$ *to Bob. When Charlie wants to send a message* M *to Bob, Charlie first associates* M *with a keyword* W*. Then Charlie invokes the* EncSEPS *algorithm which transforms both,* M *and* W*, to a searchable ciphertext* $S_{M,W}$ *using* $param$*. Charlie sends* $S_{M,W}$ *to Bob who invokes the* DecSEPS *algorithm to decrypt the message* M*. The message* M *can be decrypted if all the non-wildcard symbols of the keywords* $W^\star$ *and* W *are the same.*

Given that we can expand $\prod_{j\in J}(i-j)=\sum_{k=0}^{\lambda}a_k i^k$, where the set $\{a_0,a_1,...,a_\lambda\}$ is appropriate coefficients dependent on the wildcard positions J, Eq. 5.1 is also equivalent with

$$\sum_{k=0}^{\lambda}a_k\sum_{i=1}^{l}w_i i^k=\sum_{\substack{i=1\\ i\notin J}}^{l}w_i^\star\prod_{j\in J}(i-j). \tag{5.2}$$

We use the same approach in the DecSEPS algorithm to cancel out the effect of wildcards. In our scheme, the polynomial $\sum_{\substack{i=1\\ i\notin J}}^{l}w_i^\star\prod_{j\in J}(i-j)$ (on the right side of Eq. 5.2) occurs in the exponent of the elements, T_1 and T_2, of the trapdoor $T_{W^\star}$. The trapdoor $T_{W^\star}$ also contains the coefficients $\{a_k\}$ of the polynomial. The searchable ciphertext has the value $\sum_{i=1}^{l}w_i i^k$ in the exponent of each group element U_i. For the search, the DecSEPS algorithm first computes the polynomial $\sum_{k=0}^{\lambda}a_k\sum_{i=1}^{l}w_i i^k$ of the left side of Eq. 5.2 using the coefficients $\{a_0,a_1,...,a_\lambda\}$ and the value $\sum_{i=1}^{l}w_i i^k$ from the searchable ciphertext. The DecSEPS algorithm then checks whether this polynomial is equal to the polynomial $\sum_{\substack{i=1\\ i\notin J}}^{l}w_i^\star\prod_{j\in J}(i-j)$ which is in the trapdoor. If the equality holds the message can be decrypted.

For security reasons, we insert all these values and polynomials on the exponents of some group elements $U_1,...,U_L$. We also inject sufficient random values to the ciphertext and the trapdoor such that the scheme can be provably secure.

5.4.1 Correctness

We now show that the DecSEPS algorithm returns the correct message when the non-wildcard symbols of W and $W^\star$ are the same. Let $J=\{j_1,\ldots,j_n\}$. Then

$$\begin{aligned}
e(T_0,\prod_{k=0}^{\lambda}(C_k)^{a_k}) &= e\big(g^{\frac{\alpha s}{\sum_{m=0}^{n}x_m a_m}},\prod_{k=0}^{\lambda}V_k^{a_k(r_1+r_2)}\big)\ e\big(g^{\frac{\alpha s}{\sum_{m=0}^{\lambda}x_m a_m}},\\
&\quad \prod_{k=0}^{\lambda}\prod_{i=1}^{l}U_i^{i^k a_k w_i(r_1+r_2)}\big) \qquad (5.3)\\
&= \prod_{k=0}^{\lambda}\Big(e(g,V_0)^{\frac{\alpha s(r_1+r_2)x_k a_k}{\sum_{m=0}^{\lambda}x_m a_m}}\prod_{i=1}^{l}e(g,U_i)^{\frac{\alpha s(r_1+r_2)w_i i^k a_k}{\sum_{m=0}^{\lambda}x_m a_m}}\Big)\\
&= e(g,V_0)^{\frac{\alpha s(r_1+r_2)\sum_{k=0}^{\lambda}x_k a_k}{\sum_{m=0}^{\lambda}x_m a_m}}\prod_{i=1}^{l}e(g,U_i)^{\frac{\alpha s(r_1+r_2)w_i\sum_{k=0}^{\lambda}i^k a_k}{\sum_{m=0}^{\lambda}a_m x_m}}\\
&= e(g,V_0)^{\alpha s(r_1+r_2)}\prod_{i=1}^{l}e(g,U_i)^{\frac{\alpha s(r_1+r_2)w_i\prod_{k=1}^{\lambda}(i-j_k)}{\sum_{m=0}^{\lambda}a_m x_m}} \qquad (5.4)
\end{aligned}$$

where for (5.4) we use that $\sum_{k=0}^{\lambda} i^k a_k = \prod_{k=1}^{\lambda}(i - j_k)$.

$$\begin{aligned} e(T_1, g^{\alpha r_1}) &= e(V_0, g)^{\alpha r_1 s_1}\ e\Big(\prod_{i=1}^{l} U_i^{\frac{s\prod_{k=1}^{\lambda}(i-j_k)w_i^\star}{\sum_{m=0}^{\lambda} a_m x_m}}, g^{\alpha r_1}\Big) \\ &= \Omega_1^{r_1}\ e(g, V_0)^{\alpha s r_1}\ \prod_{i=1}^{\ell} e(g, U_i)^{\frac{\alpha s r_1 \prod_{k=1}^{\lambda}(i-j_k)W_i^\star}{\sum_{m=0}^{\lambda} a_m x_m}} \end{aligned} \tag{5.5}$$

$$\begin{aligned} e(T_2, g^{r_2}) &= e(V_0, g)^{\alpha r_2 s_2}\ e\Big(\prod_{i=1}^{l} U_i^{\frac{\alpha s\prod_{k=1}^{\lambda}(i-j_k)w_i^\star}{\sum_{m=0}^{\lambda} a_m x_m}}, g^{r_2}\Big) \\ &= \Omega_2^{r_2}\ e(g, V_0)^{\alpha s r_2}\ \prod_{i=1}^{l} e(g, U_i)^{\frac{\alpha s r_2 \prod_{k=1}^{\lambda}(i-j_k)w_i^\star}{\sum_{m=0}^{\lambda} a_m x_m}} \end{aligned} \tag{5.6}$$

$$e(T_{n+1}, g^{\alpha r_1}) e(T_{n+2}, g^{r_2}) = \Omega_1^{r_1}\Omega_2^{r_2}\ e(g, V_0)^{\alpha s(r_1+r_2)} \prod_{i=1}^{l} e(g, U_i)^{\frac{\alpha s(r_1+r_2)w_i^\star \prod_{k=1}^{\lambda}(i-j_k)}{\sum_{m=0}^{\lambda} a_m x_m}} \tag{5.7}$$

If $w_i = W_i^\star$ when $i \notin \{j_1, \ldots, j_\lambda\}$. Thus

$$\hat{C}\frac{e(T_0, \prod_{k=0}^{\lambda} C_k^{a_k})}{e(T_1, g^{\alpha r_1}) e(T_2, g^{r_2})} = \frac{M\Omega_1^{r_1}\Omega_2^{r_2} e(T_0, \prod_{k=0}^{\lambda} C_k^{a_k})}{e(T_{n+1}, g^{\alpha r_1}) e(T_{n+2}, g^{r_2})} = M \tag{5.8}$$

5.4.2 Proof of Security

Theorem 4. *The SEPS scheme is secure in the selective security model (see 2.3.1) assuming that the Decision Linear (DLin) assumption holds in group* $\mathbb{G}$.

Proof: Suppose there exists a PPT adversary $\mathcal{A}$, which can break the selective semantic security, i.e. $\mathcal{A}$, has a non-negligible advantage ε. We build an algorithm $\mathcal{B}$ that uses $\mathcal{A}$ to break the Decision Linear assumption in $\mathbb{G}$.

Assume that there is a challenger who selects a bilinear group $\mathbb{G}$ of prime order p and chooses a generator $g \in \mathbb{G}$, the group $\mathbb{G}_T$ and an efficient bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. Then the challenger picks four random values $z_1, z_2, z_3, z_4 \in_R \mathbb{Z}_p^*$, computes $Z_0 = g^{z_2(z_3+z_4)}$ and chooses $Z_1 \in_R \mathbb{G}$. After flipping a fair coin $\nu \in_R \{0, 1\}$ the challenger sends the tuple $(g, g^{z_1}, g^{z_2}, g^{z_1 z_3}, g^{z_4}, Z_\nu)$ to $\mathcal{B}$. The algorithm $\mathcal{B}$'s goal is to guess ν correctly with a probability non-negligibly larger than $\frac{1}{2}$. In order to come up with a guess, $\mathcal{B}$ interacts with the adversary $\mathcal{A}$ in a selective semantic security game as follows:

Initialization The adversary $\mathcal{A}$ chooses an alphabet $\Sigma \subset \mathbb{Z}_p$, a length L and announces two keywords $W_0^\star \in \Sigma^L$, $W_1^\star \in \Sigma^L$, which will be used for the challenge phase. The algorithm $\mathcal{B}$ flips a coin $\mu \in \{0, 1\}$. Let $W_\mu^\star = \big(w_1^\star, \ldots, w_{L_\mu}^\star\big)$.

Setup The algorithm $\mathcal{B}$ chooses an upper bound $\Lambda \leq L$ to the number of wildcard symbols. Then $\mathcal{B}$ picks random values $v_0, u_1, \ldots, u_L, x_1, \ldots, x_\Lambda \in_R \mathbb{Z}_p$ and sets

$$V_j = (g^{z_2})^{x_j v_0} g^{-\sum_{i=1}^{L_\mu} i^j u_i} \quad \text{for } j = 0, \ldots, \Lambda$$
$$U_i = \begin{cases} g^{\frac{u_i}{W_i^\star}} & \text{for } i = 1 \ldots L_\mu \\ g^{u_i} & \text{for } i = L_\mu + 1, \ldots, L \end{cases}$$

where $x_0 = 1$. $\mathcal{B}$ picks $\sigma_1, \sigma_2, \sigma_3 \in_R \mathbb{Z}_p$ and computes $\Omega_1 = e(g^{z_1}, V_0)^{\sigma_1 - \sigma_2}$ and $\Omega_2 = e\big(g^{\sigma_3}(g^{z_1})^{-\sigma_2}, V_0\big)$. The public parameters are:

$$param = \Big(\big(V_0, V_1, \ldots, V_\Lambda\big), \big(U_1, \ldots, U_L\ \big), g^{z_1}, \Omega_1, \Omega_2, p, \mathbb{G}, \mathbb{G}_T, e(\cdot,\cdot)\Big)$$

The master secret key is implicitly given by

$$msk = \big(\alpha = z_1, t_1 = \sigma_1 - \sigma_2, t_2 = \frac{\sigma_3}{z_1} - \sigma_2, (x_1, \ldots, x_\Lambda)\big).$$

Query I In this phase $\mathcal{A}$ adaptively issues key extraction queries. Each time $\mathcal{A}$ queries for the decryption key of an attribute vector $W^\star = (w_1^\star, \ldots, w_L^\star) \in \Sigma_\star^L$, consisting of L symbols and $\lambda \leq \Lambda$ wildcards at positions $J = \{j_1, \ldots, j_n\}$, algorithm $\mathcal{B}$ responds by computing

$$T_0 = (g^{z_1})^{\frac{\sigma_2}{\sum_{m=0}^{\lambda} x_m a_m}},$$
$$T_1 = V_0^{\sigma_1} \prod_{i=1}^{L} U_i^{\frac{\sigma_2 \prod_{k=1}^{\lambda}(i-j_k) W_i^\star}{\sum_{m=0}^{\lambda} x_m a_m}},$$
$$T_2 = (g^b)^{\sigma_3 v_0} g^{-\sigma_3 \sum_{i=1}^{L_\mu} u_i} (g^{z_1})^{\frac{\sigma_2 \sum_{i=1}^{L_\mu} \frac{u_i}{w_i^\star} \prod_{k=1}^{\lambda}(i-j_k) W_i^\star}{\sum_{m=0}^{\lambda} x_m a_m} + \frac{\sigma_2 \sum_{i=L_\mu+1}^{L} u_i \prod_{k=1}^{\lambda}(i-j_k) W_i^\star}{\sum_{m=0}^{\lambda} x_m a_m}},$$

which is basically a correct trapdoor for $W^\star$ with $s = \sigma_2$. $\mathcal{B}$ returns to $\mathcal{A}$ the decryption key

$$T_{W^\star} = \Big(T_0, T_1, T_2, J\big). \tag{5.9}$$

Challenge Once $\mathcal{A}$ decides that the query phase is over, $\mathcal{A}$ picks a pair of messages $M_0, M_1 \in \mathbb{G}_T$ on which it wishes to be challenged. $\mathcal{B}$ computes $S_{W_\mu^*, M_\mu}$ by first computing

$$\hat{C} = M_\mu \cdot e\big(g^{z_1 z_3}, g^{z_2}\big)^{\sigma_1 v_0} \cdot e\big(g^{z_1 z_3}, g\big)^{(\sigma_1 - \sigma_2)\sum_{i=0}^{L_\mu} u_i}.$$
$$e\big(g^{z_1}, g^{z_4}\big)^{\sigma_2 \sum_{i=0}^{L_\mu} u_i} \cdot e\big(g^{z_2}, g^{z_4}\big)^{\sigma_3 v_0} \cdot e\big(g^{z_4}, g\big)^{\sigma_3 \sum_{i=0}^{L_\mu} u_i} \cdot e\big(g^{z_1}, Z_\nu\big)^{\sigma_2 v_0} \tag{5.10}$$

and then computing $C_0 = Z_\nu^{v_0}$ and $C_k = Z_\nu^{x_k v_0}$ for $k = 1, \ldots, N$. $\mathcal{B}$ sends the challenge ciphertext

$$S_{W_\mu^\star, M_\mu} = \Big(\hat{C}, \{C_k\}_{k=0}^{\Lambda}, \begin{pmatrix} g^{z_1 z_3} \\ g^{z_4} \end{pmatrix}\Big), \tag{5.11}$$

to $\mathcal{A}$. When $\nu = 0$ this is actually a correct encryption of M_μ under $W_\mu^\star$ with $r_1 = z_3$ and $r_2 = z_4$.

Query II In Query Phase II $\mathcal{B}$ behaves exactly the same as in Query Phase I.

Output Eventually, $\mathcal{A}$ outputs a bit μ'.

Finally, $\mathcal{B}$ outputs 1 if $\mu' = \mu$ and 0 if $\mu' \neq \mu$.

We will now analyze the probability of success for algorithm $\mathcal{B}$. First, note that if $\nu = 0$, then $\mathcal{B}$ will behave correctly as a challenger to $\mathcal{A}$. Thus, $\mathcal{A}$ will have probability of $\frac{1}{2} + \epsilon$ of guessing μ. Next note that if $\nu = 1$, then Z_ν is random in $\mathbb{G}$ and $S_{W_\mu^*, M_\mu}$ is independent from μ, thus $\mathcal{A}$ will have a probability of $\frac{1}{2}$ of guessing μ.

To conclude the proof we have

$$
\begin{aligned}
&\Big|\Pr\big[\mathcal{B}(g^{z_1}, g^{z_2}, g^{z_1 z_3}, g^{z_4}, g^{z_2(z_3+z_4)}) = 1\big] - \Pr\big[\mathcal{B}(g^{z_1}, g^{z_2}, g^{z_1 z_3}, g^{z_4}, g^{r}) = 1\big]\Big| \\
\geq &\Big|\Pr\big[\nu = 0 \ \wedge \ \mu' = \mu\big] - \Pr\big[\nu = 1 \ \wedge \ \mu' = \mu\big]\Big| \\
= &\Big|\frac{1}{2}\Pr\big[\mu' = \mu \mid \nu = 0\big] - \frac{1}{2}\Pr\big[\mu' = \mu \mid \nu = 1\big]\Big| \\
= &\frac{1}{2}\Big|\Pr\big[\mathbf{Exp}_{\mathcal{A}}(\kappa) = 1\big] - \frac{1}{2}\Big| \\
\geq &\frac{1}{2}\epsilon,
\end{aligned}
$$

which is non-negligible, contradicting the Decision Linear Assumption.

5.5 Efficiency

In Table 5.1 we compare the efficiency of the SEPS scheme with the BW scheme [12], which uses composite order pairing, and the IP scheme [30] which uses prime order pairing. Since the IP scheme accepts binary alphabets only, each symbol of the keyword should be represented as binary values. Here, we assume that each symbol consists of 7 bits required to represent ACSII codes. In this table also

- L is the maximum number of symbols in the keyword,
- l is the number of symbols in the keyword (for the queried keyword and the keyword associated with the message),
- λ is the number of wildcards in the keyword.

Since BW and IP do not restrict the number of wildcards in the queried keyword, we perform the comparison with the assumption that $\Lambda = L$. In this case, the number of wildcards can be up to the maximum number of symbols of the keyword, which is the same as BW and IP. To make the comparison more convenient, the computational

complexity of each scheme is shown based on the number of exponentiation operations only. Exponentiation has the highest complexity compared to other operations (e.g. multiplication).

Table 5.1 shows that SEPS has a lower complexity for the search and the trapdoor communication compared to BW and IP. The SEPS scheme has a lower complexity compared to BW and IP schemes in the complexity aspects marked with the red color. This table shows that the BW scheme, performs more than 3 composite order pairing for the search. Since each composite order pairing is roughly 50 times slower than the prime order pairing [22], the search in BW is prohibitively slow. Therefore, although this scheme has a lower complexity for the searchable ciphertext and the trapdoor compared to our scheme, BW is not practical particularly for the applications where the number of wildcards is usually small. Comparing the IP scheme with SEPS also shows that the search in SEPS is more efficient than the search in IP. However, the computational complexity of the search ciphertext in IP is more efficient than SEPS if $L \leq 14$. Since keywords in practice can contain more than 14 symbols, the maximum number of supported symbols L should be more than 14. This makes the IP scheme more efficient than our scheme for the searchable ciphertext.

5.6 Conclusion

We present a hidden vector encryption scheme, called SEPS, which searches keywords with wildcards on encrypted data. The SEPS scheme is provably secure in the selective security model. Our scheme is more efficient than existing schemes for the search because SEPS uses bilinear groups of prime order and accepts any alphabet. In existing schemes, the search is performed either using bilinear pairing of composite order, or bilinear pairing of prime order groups but with the binary alphabet only. Our scheme also has a low communication complexity for the trapdoor and a low storage complexity. However, the computational complexity of the searchable ciphertext and the trapdoor can be higher than exiting schemes if the number of wildcards in the queried keyword is large. Therefore, the SEPS scheme is suitable for applications where the efficiency of the search is crucial and either the trapdoor is rarely constructed, or the number of wildcards in the queried keyword is not large. We now revisit Scenario 2 which was explained in the Introduction chapter. In this scenario Bob delegates a trapdoor to Carol. The trapdoor allows Carol to decrypt only Bob's messages that contain the keyword associated with the trapdoor. In this case, Carol should be able to decrypt Bob's e-mails fast which highlights the efficiency of the search. From the other hand, the computational efficiency of the trapdoor is not crucial because the delegated trapdoors are constructed once. Therefore, the SEPS scheme is suitable for this scenario.

		SEPS	IP	BW
Computational Complexity	Searchable ciphertext	$(l+1)L+4$ exp	$14l$ exp	$3l+2$ exp*
	Trapdoor	$2l+3$ exp	$14(l-\lambda)$ exp	$3(l-\lambda)+3$ exp*
	Search	3 pairing	$14(l-\lambda)$ pairing	$2(l-\lambda)+1$ pairing*
Communication Complexity (bits)	Searchable Ciphertext	$L+4$ group elements	$14l$ group elements	$2l+2$ group elements*
	Trapdoor	3 group elements	$14(l-\lambda)$ group elements	$2(l-\lambda)$ group elements*
Storage Complexity	Master secret key (bits)	$(2L+5)\sigma$	$(28L+1)\sigma$	$3(L+1)\sigma$
	Public parameters	$(2L+6)$ group elements	$28L+2$ group elements	$3L+1$ group elements*
	Searchable Ciphertext	$L+4$ group elements	$14l$ group elements	$2l+2$ group elements*

Table 5.1: *Comparison of the complexity of our schemes with the BW and IP schemes. Here "exp" - stands for exponentiation. The sign "*" shows that the operation and the group element belongs to a composite order group. The SEPS scheme has a lower complexity compared to the other schemes in the aspects shown in the red color.*

Chapter 6

Fully Secure Searchable Encryption

Most of existing searchable encryption schemes are proven to be secure in weaker security models than the fully secure model. The few schemes which achieve full security either are inefficient, because of using bilinear groups of composite order, or provide security based on strong assumptions. In this chapter, we propose a novel public key searchable encryption scheme, called SEPF, which is fully secure based on the weak assumptions, DLin and DBDH. The SEPF scheme uses bilinear groups of prime order to perform the search. We compare the complexity of SEPF with the DIP [17] scheme, which is the only scheme that achieves full security based on weak assumptions. We show that SEPF has a higher search efficiency compared to DIP. This efficiency stems from the fact that DIP uses bilinear groups of composite order for the search, which are considerably slower than groups of prime order.

6.1 Introduction

While many searchable encryption schemes have been proposed, there are a few schemes which achieve full security. Most searchable encryption schemes are proposed in the selective security or random oracle models. These models either restrict the power of the adversary, or use theoretical primitives in the construction of schemes, which cannot be implemented.

Existing schemes that are proven to be fully secure either use bilinear groups of composite order [17] (composite of four primes in this case) or use strong assumptions for the security proof [24]. Therefore, these schemes are either prohibitively slow or achieve a lower security compared to schemes that use weak assumptions for the security proof.

In this chapter we propose a fully secure searchable encryption scheme called SEPF. Our scheme:

- is the first fully secure searchable encryption scheme which uses bilinear groups of prime order.

- achieves security based on the DLin and DBDH assumptions which are weak and well established assumptions.

6.2 Related Work

One of the reasons why most of searchable encryption schemes do not achieve full security is the problem with the used security proof methodology. The so called partitioning based method cannot show whether the scheme is fully secure. We explain the problem in more detail in 6.3. The first solution to address this problem is proposed by Gentry [24]. While Gentry's scheme achieves full security, this scheme uses a strong assumption to achieve the security proof. However, it is always preferred to complete the security proof using weak assumptions.

Recently, a new security methodology called dual system encryption (DSE) has been proposed by Waters [42]. Using the DSE method, a fully secure proof can be achieved using weak assumptions such as the DLin or DBDH assumptions. Waters [42] also proposed the first fully secure identity based encryption scheme which uses prime order groups. Following the publication of DSE, several fully secure attribute based encryption and identity based encryption schemes have been proposed [33, 36, 32]. These schemes deploy composite order groups to achieve full security.

The only fully secure anonymous identity based encryption scheme, which can be used as searchable encryption, is proposed in [17]. This scheme uses composite order groups of four primes to achieve security. Using four primes for the security makes the scheme prohibitively slow.

6.3 Challenges of Security Proof

The security of searchable encryption schemes rely on the underlying complexity assumption used in the construction of the scheme. The complexity assumption states that there exists a tuple such that no PPT algorithm can decide whether the tuple is valid, (i.e. the elements of the tuple are related to each other) or whether the tuple is random. Here, we call such a tuple a complexity tuple. To prove that a scheme is secure, it must be shown that breaking the scheme is as hard as breaking the deployed complexity assumption. This type of security proof is formally called a reduction from breaking the complexity assumption to breaking the scheme. Since the complexity assumption is known to be hard to break, we can then prove that breaking the scheme is hard.

The traditional partitioning based method achieves the reduction as follows. It is assumed that a party, called the simulator, wants to decide whether a complexity tuple is valid or random. To make the decision, the simulator uses adversary's power in breaking the scheme by playing the role of the challenger of the security model. The simulator, first provides the adversary simulated public parameters and responds to each trapdoor query of the adversary by simulating and sending a trapdoor to the adversary. Here, the distribution of the simulated public parameters and trapdoors

must be the same as the distribution of the public parameters and trapdoor of the scheme. Then, in the challenge phase, the simulator constructs a challenge searchable ciphertext using the elements of the complexity tuple and sends it to the adversary. Here, if the complexity tuple is random, the challenge searchable ciphertext is random so that the adversary has no advantage to win the game. Therefore, if the adversary wins the game, the simulator learns that the tuple is valid, otherwise the tuple is random. In this case the probability of breaking the scheme is the same as the probability of breaking the complexity assumption. In this way the security reduction is achieved, as required. The message flow of the partitioning based methodology to achieve the security reduction is illustrated in Figure 6.1.

One of the main challenges of completing the security proof using the partitioning based method is that the simulator should be able to simulate the trapdoor of each keyword except the trapdoor of the challenge keywords (T_{W_β} in Figure 6.1). Otherwise, after constructing the challenge searchable ciphertext, the simulator can create the trapdoor of the challenge keywords to check whether decryption is possible. If the tuple is random, the challenge searchable ciphertext is random which is not decryptable. Therefore, if the correct message can be decrypted, the simulator learns that the complexity tuple is valid. In this case the simulator can break the complexity assumption without interacting with the adversary. It shows that the scheme must be constructed in such a way that simulating the trapdoor of some keywords cannot be performed.

To address the problem of simulating most but not all the trapdoor, the selective security and random oracle models have been proposed. These models enable the simulator to simulate most of the trapdoor but not all. The selective security model allows the simulator to receive the challenge keywords in advance of the game. This helps the simulator to embeds the challenge keywords into the public parameters in such a way that simulating the trapdoor of the challenge keywords is impossible. In the random oracle model it is also assumed that the random oracle outputs different types of random values, such that one type is used for simulating the trapdoors and the other type can be used to construct the challenge searchable ciphertext only. While these models solve the problem of simulating the trapdoors, either the adversary of these model is restricted or the scheme uses a random oracle which is impractical.

The first attempt to achieve full security in the standard model is made by Waters [41], where a fully secure identity based encryption scheme is proposed. In [41], the simulator of the security proof would abort if it wants to simulate the trapdoor of the challenge keywords. However, this scheme achieves a loose reduction. In a loose reduction, in contrast to a tight reduction, it is shown that the probability of breaking the scheme is not close to the probability of breaking the complexity assumption. This scheme also is not anonymous and therefore cannot be used as a searchable encryption scheme. Gentry [24] proposed the first fully secure anonymous identity based encryption which achieves a tight reduction. This scheme is constructed in such a way that the number of the trapdoors that can be simulated by the simulator is limited to the number of trapdoor queries made by the attacker. Therefore, in the challenge phase, the simulator cannot simulate the trapdoor of the challenge

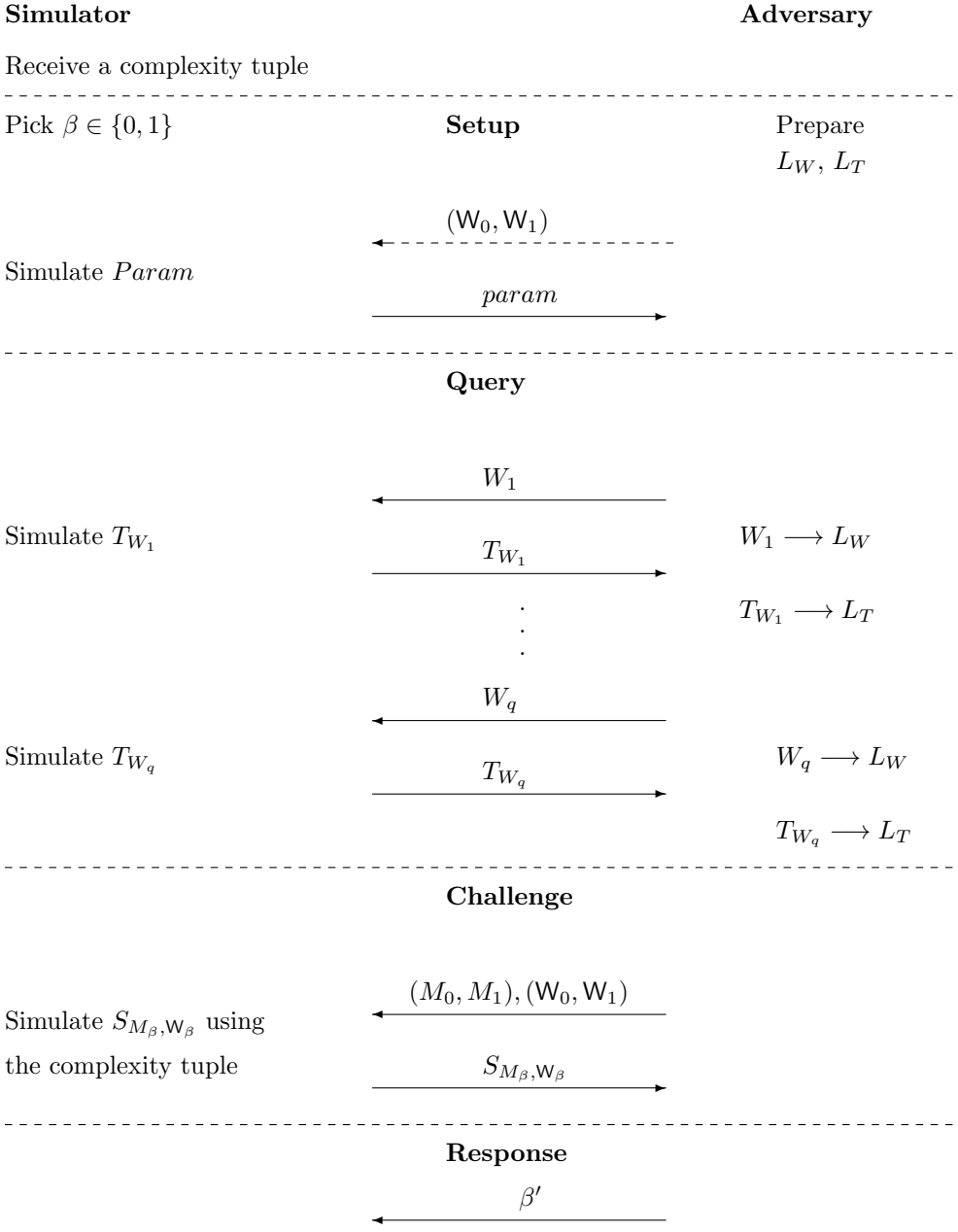

Figure 6.1: *The message flow of the partitioning based methodology. The dashed vector is required for the selective security model, but not the full security.*

keywords. The problem with the Gentry scheme is that a strong assumption, called q-ABDHE, is used for the security proof. However, it is always preferred to complete the proof using weak assumptions.

6.4 Dual System Encryption

Recently, a new security proof methodology has been proposed which is called dual system encryption (DSE) [42]. The DSE method allows completing the security proof without restricting the adversary or assuming theoretical primitives for the scheme. The DSE method also achieves a tight reduction using weak assumptions such as DLin and DBDH.

In dual system encryption (DSE), the searchable ciphertext and the trapdoor can take two distributions: a normal distribution, which is used for the construction of the scheme, and a semi-functional distribution, which is used for the security proof only. The semi-functional searchable ciphertext and the trapdoor are created using the master secret key. These distributions have extra random elements added to the corresponding normal distribution, such that two properties are satisfied:

1. the semi-functional distribution must be indistinguishable from the corresponding normal distribution,

2. the message can be decrypted correctly only if both, the searchable ciphertext and the trapdoor, are not semi-functional.

DSE completes the security proof using the semi-functional distribution of the searchable ciphertext and the trapdoor. Consider a security game which is the same as the semantic security game defined in Section 2.4 except that i) the challenger uses a semi-functional distribution for the trapdoors of the query phase and the challenge searchable ciphertext and ii) the challenger transforms a random message and a random keyword to the challenge searchable ciphertext. We call this game, "game final" as shown in Figure 6.2. We argue that game final is perfectly secure because neither the challenge searchable ciphertext, which embeds random values (e.g. R_1 and R_2 in Figure 6.2) instead of the message and the keyword, nor the semi-functional trapdoors of the query phase, which cannot be used for decryption, reveal any information about the message and the keyword. The goal of DSE is to show that the semantic security game is indistinguishable from game final. To achieve this, DSE defines a sequence of intermediate games between the semantic security game and game final in a such way that the adversary cannot detect the difference between consecutive games. In the first intermediate game, the challenger uses a semi-functional searchable ciphertext in the challenge phase instead of a normal one. We show that the adversary cannot distinguish the semantic security game from the first intermediate game. In each of the next intermediate games, the challenger switches the distribution of one of the trapdoors in the query phase, from normal to semi-functional. Therefore, in the last intermediate game, the searchable ciphertexts and all the trapdoors take a semi-functional distribution. After showing that each

game is indistinguishable from its next game, it results that the semantic security game is indistinguishable from game final. This proves the security of the scheme.

Now, we explain an inherent problem in the proof of the indistinguishability of the semi-functional and normal trapdoors. To show this indistinguishability, it must be shown that the simulator constructs a trapdoor using a complexity tuple, such that the trapdoor is normal only if the tuple is valid, otherwise the trapdoor is semi-functional. The simulator then sends the trapdoor to the adversary who decides whether the trapdoor is normal or semi-functional. In this case it is shown that the probability that the adversary makes a right decision is the same as the probability of deciding whether the tuple is valid, as required. However, the problem is that the simulator can build a semi-functional searchable ciphertext with the same keyword of the trapdoor keyword and check whether the semi-functional searchable ciphertext is decryptable. If the correct message is decrypted, the simulator learns that trapdoor is normal. Therefore, the simulator can break the complexity assumption without interacting with the adversary.

To address this problem, the searchable ciphertext and the trapdoor of a fully secure scheme are associated with a unique tag. The fully secure scheme must be constructed in a way that i) decryption can be performed only if the tag of the searchable ciphertext and the tag of the trapdoor are not the same, and ii) the simulator can construct a semi-functional searchable ciphertext in the intermediate games only if he uses the same tag as the tag used for the trapdoor. Therefore, the simulator cannot check the whether the trapdoor can decrypt the semi-functional searchable ciphertext.

6.5 Intuition

One of the main challenges in dual system encryption is to construct semi-functional distributions for the searchable ciphertext and the trapdoor. Firstly, to create the semi-functional distribution, some extra random elements are added to the normal construction in such a way that adversaries cannot detect these extra randomness. This implies that the normal distribution for both, the searchable ciphertext and the trapdoor, must be highly randomized. Secondly, decryption should be possible if both, the searchable ciphertext and the trapdoor, are not semi-functional. This implies that the extra random elements of the semi-functional searchable ciphertext should be orthogonal to the normal trapdoor. Analogously, the extra elements of the semi-functional trapdoor should also be orthogonal to the normal searchable ciphertext. Here, by orthogonality we mean that the extra random elements of the semi-functional distribution fall out during the decryption.

In bilinear groups of composite order it is easier to find the semi-functional distributions. The reason is that each subgroup is orthogonal to the other subgroup in a way that pairing operations between the elements picked from different subgroups always outputs one. In fully secure schemes that use groups of composite order, the normal trapdoor and searchable ciphertext are constructed from the subgroup. The extra random elements that make the distributions semi-functional are picked

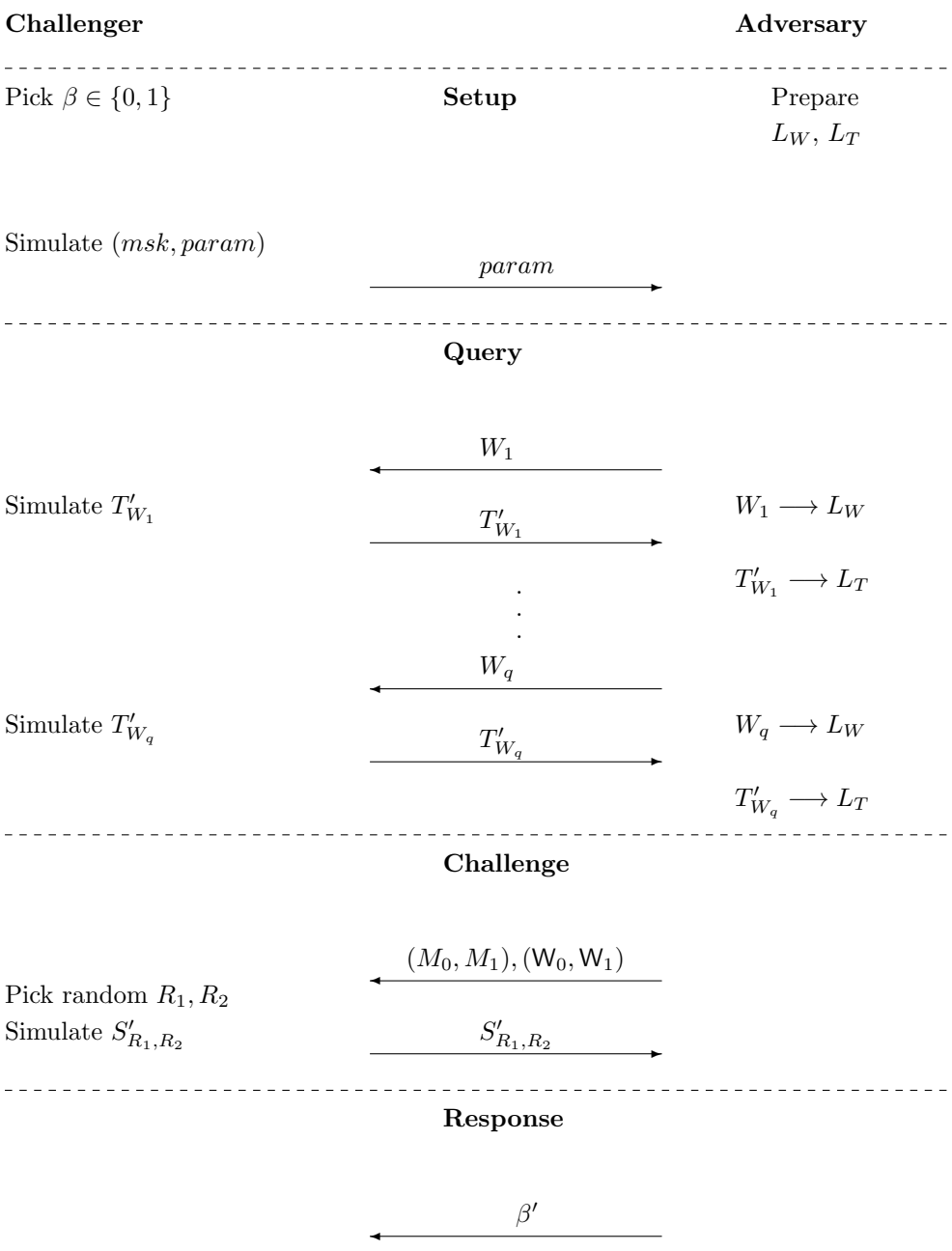

Figure 6.2: *The message flow of game final. Here, T'_{W_1} and T'_{W_q} are semi-functional trapdoors of W_1 and W_q and S'_{R_1,R_2} is the semi-functional searchable ciphertext of message R_1 and keyword R_2.*

from other subgroup. This distribution cancels out the effect of the semi-functional elements during decryption if either the searchable ciphertext or the trapdoor is normal. However, if both, the ciphertext and the trapdoor, are semi-functional, the orthogonality among them does not hold.

In prime order pairings, however, achieving orthogonality is more difficult. In the SEPF scheme we achieve orthogonality in the following way: we introduce two system parameters x and y,which are embedded in the public parameters, a variable in the ciphertext x and a variable in the trapdoor y. We then design the construction in such a way that the search outputs

$$Me(g,g)^{(\mathsf{x}-x)(\mathsf{y}-y)\phi},$$

where ϕ is a random value. In the normal searchable encryption, $\mathsf{x} = x$, and in the normal trapdoor $\mathsf{y} = y$. However, the semi-functional algorithms will choose x or y randomly. It is not hard to see that such a decryption results in M if either the ciphertext or the trapdoor is semi-functional.

6.6 Construction

In this section we present the construction of the SEPF scheme. We explain the intuition behind the scheme after presenting the semi-functional distributions. Our scheme consists of the following algorithms:

- $\mathsf{KeygenSEPF}(\sigma)$: Given the security parameter σ, the algorithm:

  1. Generates a bilinear group $\mathbb{G}$ of a large prime order p and choose a bilinear map $e : \mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{G}_T$.
  2. Picks random $a, b, c, d, l, t, x, y, u, v, h, f, \lambda \in \mathbb{Z}_p$,
  3. Computes $\Omega = e(g, V)^{abc\lambda}$,
  4. Let $U = g^u, V = g^v, H = g^h, F = g^f$,

  The public parameters are:

  $$param = \Big(g^{ab}, g^{abc}, g^{ay}, g^b, g^{bcx}, g^{bd}, g^{byt}, g^y, g^{xy}, U, V, V^{\frac{1}{t}}, H, F, \Omega\Big)$$

  The master secret key is

  $$msk = \big(a, b, c, d, l, t, x, y, u, v, h, f, \lambda\big).$$

- $\mathsf{TrapdoorSEPF}(\mathsf{W}, msk)$: Given the keyword W and the master secret key msk, the algorithm first picks random $s_1, s_2, s_3, s_4, \tau_D \in \mathbb{Z}_p$, and then computes the

trapdoor $T_{\mathsf{W}} =$

$$\left(\left(\begin{array}{ll} T_1 = & g^{ay(s_1+s_2)} g^{-ays_4} \\ T_2 = & g^{-abcs_1} g^{abcs_4} \\ T_3 = & g^{-abcls_1} \\ T_4 = & g^{-acs_2} \\ T_5 = & g^{ds_2} \end{array} \right), \left(\begin{array}{ll} T_6 = & g^{-bc(xs_1+ts_2)} g^{bcs_3} \\ T_7 = & g^{xy(s_1+s_2)} g^{-ys_3} g^{-\lambda} \\ T_8 = & g^{-abc(xs_1+ts_2)} g^{abcs_3} \\ T_9 = & g^{axy(s_1+s_2)} g^{-ays_3} g^{-a\lambda} \end{array} \right), \right.$$

$$\left. \left(\begin{array}{ll} T_{10} = & (U^{\mathsf{W}} V^{\tau_D} H)^{s_1} (F V^{\tau_D})^{s_4} \\ T_{11} = & (U^{\mathsf{W}} V^{\tau_D} H)^{cs_1} \\ T_{12} = & g^{-abcs_1} \\ T_{13} = & g^{-abcs_4} \\ T_{14} = & g^{acs_4} \end{array} \right), \tau_D \right).$$

- $\mathsf{EncSEPF}(W, M, param)$: Given the message M, the keyword W and the public parameters $param$, the algorithm first picks random r_1,r_2 ,r_3, r_4, $r_5, \tau_C \in \mathbb{Z}_p$, and then computes the searchable ciphertext $S_{M,W} =$

$$\left(M\Omega^{r_1+r_2}, \right.$$

$$\left(\begin{array}{ll} C_1 = & (g^{bcx})^{r_1+r_2} \\ C_2 = & (g^{xy})^{r_1+r_2} (V)^{r_3} \\ C_3 = & (V^{\frac{1}{l}})^{r_4} \\ C_4 = & (g^{byt})^{r_1+r_2} (g^{bd})^{r_3} \\ C_5 = & (g^{abc})^{r_3} \end{array} \right), \left(\begin{array}{ll} C_6 = & (g^{ay})^{r_1} \\ C_7 = & (g^{abc})^{r_1} \\ C_8 = & (g^{y})^{r_2} \\ C_9 = & (g^{bc})^{r_2} \end{array} \right),$$

$$\left. \left(\begin{array}{ll} C_{10} = & (g^{abc})^{r_3} \\ C_{11} = & (g^{ab})^{r_4} \\ C_{12} = & (U^{W} V^{\tau_C} H)^{r_3+r_4} \\ C_{13} = & g^{r_5} (F V^{\tau_C})^{r_3} \\ C_{14} = & (g^{b})^{r_5} \end{array} \right), \tau_C \right).$$

- $\mathsf{DecSEPF}(S_{M,W}, T_{\mathsf{W}})$: Given the trapdoor T_{W} and the ciphertext $S_{M,W}$, decrypt the message M as follows: first compute

$$(M\Omega^{r_3+r_4}) \prod_{i=1}^{9} e(T_1, C_i) \Big(\prod_{i=10}^{14} e(T_i, C_i) \Big)^{\frac{1}{\tau_C - \tau_D}}$$

The message exchange of the SEPF scheme is illustrated in Figure 6.3. The message exchange of SEPF has the following differences with the message exchange of public key searchable encryption shown in Figure 2.3:

- The algorithms $\mathsf{Keygen_P}$, $\mathsf{Enc_P}$, $\mathsf{Trapdoor_P}$, and Dec are replaced by $\mathsf{KeygenSEPS}$, $\mathsf{EncSEPF}$, $\mathsf{TrapdoorSEPF}$, and $\mathsf{DecSEPF}$.
- The metadata in Figure 6.3 is one keyword only while in Figure 2.3 the metadata can take any number of keywords.

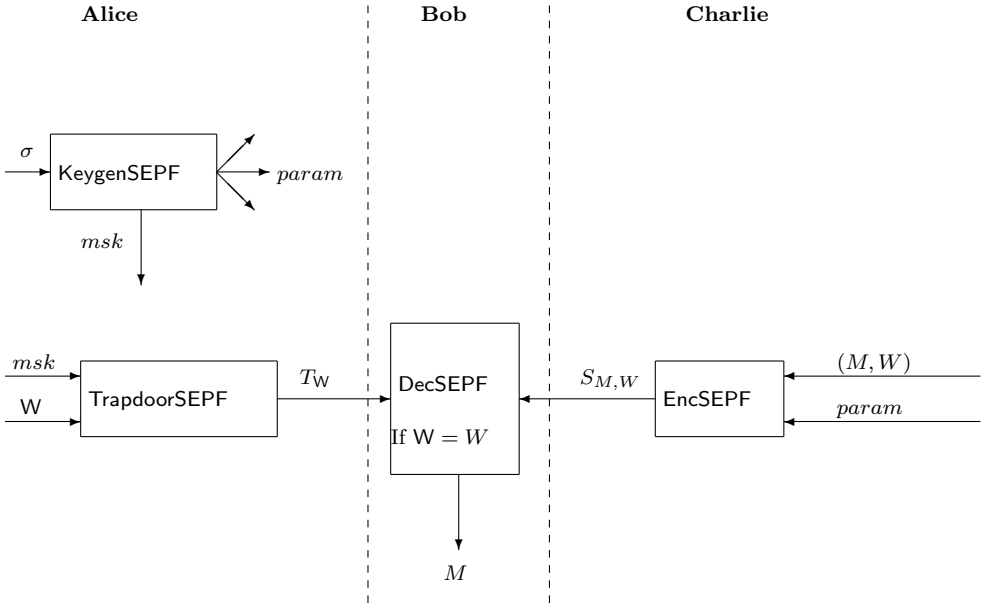

Figure 6.3: *The message flow of the SEPF scheme. Alice, who owns the master secret key, creates a trapdoor and sends it to Bob. Charlie, who wants to send a message to Bob, transforms the message to a searchable ciphertext using the public parameters, and sends it to Bob. Bob decrypts the message if the keyword of the trapdoor and the associated keywords with the message are the same.*

6.6.1 Correctness

We show that the DecSEPF algorithm decrypts the message correctly when the queried keyword W and W are the same.

$$
\begin{aligned}
e(T_1, C_1) =& e(g,g)^{abcxy(r_1+r_2)(s_1+s_2)} e(g,g)^{-abcxy(r_1+r_2)s_4} \\
e(T_2, C_2) =& e(g,g)^{-abcxy(r_1+r_2)s_1} e(g,g)^{abcxy(r_1+r_2)s_4} e(g,V)^{-abcr_3 s_1} e(g,V)^{abcr_3 s_4} \\
e(T_3, C_3) =& e(g,V)^{-abcr_4 s_1} \\
e(T_4, C_4) =& e(g,g)^{-abcyt(r_1+r_2)s_2} e(g,g)^{-abcdr_3 s_2} \\
e(T_5, C_5) =& e(g,g)^{abcdr_3 s_2}
\end{aligned}
$$

Hence:

$$
\begin{aligned}
&\prod_{i=1}^{5} e(T_i, C_i) = \\
&e(g,g)^{abcxy(r_1+r_2)s_1} e(g,g)^{-abcyt(r_1+r_2)s_2} e(g,V)^{-abc(r_3+r_4)s_1} e(g,V)^{abcr_3 s_4}
\end{aligned}
$$

$$
\begin{aligned}
e(T_6, C_6) =& e(g,g)^{-abcxyr_1 s_1} e(g,g)^{-abctyr_1 s_2} e(g,g)^{abcyr_1 s_3} \\
e(T_7, C_7) =& e(g,g)^{abcxyr_1(s_1+s_2)} e(g,g)^{-abcyr_1 s_3} e(g,g)^{-abc\lambda r_1} \\
e(T_8, C_8) =& e(g,g)^{-abcxyr_2 s_1} e(g,g)^{-abctyr_2 s_2} e(g,g)^{abcyr_2 s_3} \\
e(T_9, C_9) =& e(g,g)^{abcxyr_2(s_1+s_2)} e(g,g)^{-abcyr_2 s_3} e(g,g)^{-abc\lambda r_2}
\end{aligned}
$$

Hence:

$$
\prod_{i=6}^{9} e(T_i, C_i) = e(g,g)^{-abcxy(r_1+r_2)s_2} e(g,g)^{-abcty(r_1+r_2)s_2} e(g,g)^{-abc\lambda(r_1+r_2)}
$$

$$
\begin{aligned}
&e(T_{10}, C_{10}) = \\
&e(g,U)^{abcr_3 s_1 \mathsf{W}} e(g,V)^{abcr_3 s_1 \tau_D} e(g,H)^{abcr_3 s_1} e(g,F)^{abcr_3 s_4} e(g,V)^{abcr_3 s_4 \tau_D} \\
&e(T_{11}, C_{11}) = e(g,U)^{abcr_4 s_1 \mathsf{W}} e(g,V)^{abcr_4 s_1 \tau_D} e(g,H)^{abcr_4 s_1} \\
&e(T_{12}, C_{12}) = e(g,U)^{-abc(r_3+r_4)s_1 W} e(g,V)^{-abc(r_3+r_4)s_1 \tau_C} e(g,H)^{-abc(r_3+r_4)s_1} \\
&e(T_{13}, C_{13}) = e(g,g)^{-abcr_5 s_4} e(g,F)^{-abcr_3 s_4} e(g,V)^{-abcr_3 s_4 \tau_C} \\
&e(T_{14}, C_{14}) = e(g,g)^{abcr_5 s_4}
\end{aligned}
$$

Hence:

$$
\begin{aligned}
&\prod_{i=10}^{14} e(T_i, C_i) = \\
&e(g,U)^{abc(r_3+r_4)s_1(\mathsf{W}-W)} e(g,V)^{abc(r_3+r_4)s_1(\tau_D-\tau_C)} e(g,V)^{-abcr_3 s_4(\tau_D-\tau_C)}
\end{aligned}
$$

The decryption algorithm outputs:

$$(M\Omega^{r_1+r_2})\prod_{i=1}^{9} e(T_1, C_i)\Big(\prod_{i=10}^{14} e(T_i, C_i)\Big)^{\frac{1}{\tau_C - \tau_D}} = Me(g,U)^{abc(r_3+r_4)s_1(\mathsf{W}-W)}$$

In case the keywords associated with the trapdoor and the searchable ciphertext are the same, $(\mathsf{W} = W)$ the decryption algorithm outputs the correct message.

6.7 Security Proof

We prove the security of SEPF using the dual system encryption methodology. As mentioned before, DSE uses a semi-functional distribution for the trapdoor and the searchable ciphertext to accomplish the proof. Here, we present the semi-functional distributions of the trapdoor and the searchable ciphertext which are only used for the security proof.

6.7.1 Semi-functional algorithms

Semi-functional trapdoor: First run the $\mathsf{EncSEPF}$ algorithm. Then pick a random $\mathsf{y} \in \mathbb{Z}_p$. The semi-functional trapdoor is:

$$\left(\begin{pmatrix} T'_1 = & T_1 g^{a(\mathsf{y}-y)(s_1+s_2)} \\ T'_2 = & T_2 \\ T'_3 = & T_3 \\ T'_4 = & T_4 \\ T'_5 = & T_5 \end{pmatrix}, \begin{pmatrix} T'_6 = & T_6 \\ T'_7 = & T_7 g^{x(\mathsf{y}-y)(s_1+s_2)} \\ T'_8 = & T_8 \\ T'_9 = & T_9 g^{ax(\mathsf{y}-y)(s_1+s_2)} \end{pmatrix}, \right.$$
$$\left. \begin{pmatrix} T'_{10} = & T_{10} \\ T'_{11} = & T_{11} \\ T'_{12} = & T_{12} \\ T'_{13} = & T_{13} \\ T'_{14} = & T_{14} \end{pmatrix}, \tau_D \right).$$

Observe that the difference between the normal trapdoor and the semi-functional trapdoor is that the value y in the elements T'_2, T'_7, and T'_9 is replaced by the random y.

Semi-functional searchable ciphertext: First run the $\mathsf{EncSEPF}$ algorithm. Then

pick a random $\mathsf{x} \in \mathbb{Z}_p$. The semi-functional searchable ciphertext $S'_{M,W}$ is:

$$\left(\begin{array}{c} M\Omega^{r_1+r_2}, \\ \left(\begin{array}{rc} C'_1 = & C_1 g^{bc(\mathsf{x}-x)(r_1+r_2)} \\ C'_2 = & C_2 g^{y(\mathsf{x}-x)(r_1+r_2)} \\ C'_3 = & C_3 \\ C'_4 = & C_4 g^{by(\mathsf{x}-x)(r_1+r_2)} \\ C'_5 = & C_5 \end{array} \right), \left(\begin{array}{rc} C'_6 = & C_6 \\ C'_7 = & C_7 \\ C'_8 = & C_8 \\ C'_9 = & C_9 \end{array} \right), \\ \left(\begin{array}{rc} C'_{10} = & C_{10} \\ C'_{11} = & C_{11} \\ C'_{12} = & C_{12} \\ C'_{13} = & C_{13} \\ C'_{14} = & C_{14} \end{array} \right), \tau_C \end{array} \right).$$

Observe that the difference between the normal and the semi-functional searchable ciphertext is that the value x in C'_1 and C'_2 is replaced by the random x, and the element $g^{by(\mathsf{x}-x)(r_1+r_2)}$ is multiplied to C_4.

Correctness We show that the DecSEPF algorithm decrypts the message M correctly only if at least one of the distributions of the searchable ciphertext and the trapdoor is normal.

$$
\begin{aligned}
e(T_1', C_1') =& e(T_1, C_1) e(g^{a(\mathsf{y}-y)(s_1+s_2)}, C_1) e(T_1, g^{bc(\mathsf{x}-x)(r_1+r_2)}) \\
& e(g^{a(\mathsf{y}-y)(s_1+s_2)}, g^{bc(\mathsf{x}-x)(r_1+r_2)}) \\
=& e(g^{a(\mathsf{y}-y)(s_1+s_2)}, g^{bcx(r_1+r_2)}) e(g^{ay(s_1+s_2)} g^{-ays_4}, g^{bc(\mathsf{x}-x)(r_1+r_2)}) \\
& e(g^{a(\mathsf{y}-y)(s_1+s_2)}, g^{b(\mathsf{x}-x)(r_1+r_2)}) \\
=& e(g,g)^{abcx(\mathsf{y}-y)(r_1+r_2)(s_1+s_2)} e(g,g)^{abcy(\mathsf{x}-x)(r_1+r_2)(s_1+s_2)} \\
& e(g,g)^{-abc(\mathsf{x}-x)y(r_1+r_2)s_4} e(g,g)^{abc(\mathsf{x}-x)(\mathsf{y}-y)(r_1+r_2)(s_1+s_2)} \\
e(T_2', C_2') =& e(T_2, C_2) e(T_2, g^{y(\mathsf{x}-x)(r_1+r_2)}) \\
=& e(T_2, C_2) e(g^{-abcs_1} g^{abcs_4}, g^{y(\mathsf{x}-x)(r_1+r_2)}) \\
=& e(T_2, C_2) e(g,g)^{-abc(\mathsf{x}-x)y(r_1+r_2)s_1} e(g, g^{abc(\mathsf{x}-x)y(r_1+r_2)} s_4) \\
e(T_4', C_4') =& e(T_4, C_4) e(T_4, g^{by(\mathsf{x}-x)(r_1+r_2)}) \\
=& e(T_4, C_4) e(g^{-acs_2}, g^{by(\mathsf{x}-x)(r_1+r_2)}) \\
=& e(T_4, C_4) e(g,g)^{-abc(\mathsf{x}-x)y(r_1+r_2)s_2} \\
e(T_7', C_7') =& e(T_7 g^{x(\mathsf{y}-y)(s_1+s_2)}, C_7) \\
=& e(T_7, C_7) e(g^{x(\mathsf{y}-y)(s_1+s_2)}, (g^{abc})^{r_1}) \\
=& e(T_7, C_7) e(g,g)^{abcx(\mathsf{y}-y)r_1(s_1+s_2)} \\
e(T_9', C_9') =& e(T_9 g^{ax(\mathsf{y}-y)(s_1+s_2)}, C_9) \\
=& e(T_9, C_9) e(g^{ax(\mathsf{y}-y)(s_1+s_2)}, g^{bc})^{r_2}) \\
=& e(T_9, C_9) e(g,g)^{abcx(\mathsf{y}-y)r_2(s_1+s_2)}
\end{aligned}
$$

Hence,

$$
\begin{aligned}
& \mathsf{DecSEPF}(T_{\mathsf{W}}', S_{W,M}') = \\
& (M\Omega^{r_1+r_2}) \prod_{i=1}^{9} e(T_1', C_i') \Big(\prod_{i=10}^{14} e(T_i', C_i') \Big)^{\frac{1}{\tau_C - \tau_D}} = \\
& (M\Omega^{r_1+r_2}) \prod_{i=1}^{9} e(T_1, C_i) \Big(\prod_{i=10}^{14} e(T_i, C_i) \Big)^{\frac{1}{\tau_C - \tau_D}} e(g,g)^{abc(\mathsf{x}-x)(\mathsf{y}-y)(r_1+r_2)(s_1+s_2)} = \\
& Me(g,U)^{abc(\mathsf{W}-W)(r_3+r_4)s_1} e(g,g)^{abc(\mathsf{x}-x)(\mathsf{y}-y)(r_1+r_2)(s_1+s_2)}
\end{aligned}
$$

In case $\mathsf{W} = W$, the $\mathsf{DecSEPF}$ algorithm outputs $Me(g,g)^{abc(\mathsf{x}-x)(\mathsf{y}-y)(r_1+r_2)(s_1+s_2)}$. Therefore, the correct message is decrypted if at least either the searchable ciphertext or the trapdoor is random, such that $\mathsf{x} = x$ or $\mathsf{y} = y$. Here, $abc(r_1 + r_2)(s_1 + s_2)$ is actually the value ϕ introduced in section 6.5.

6.7.2 Intuition

The searchable ciphertext and the trapdoor should be constructed in such a way that their corresponding semi-functional distribution can be proven to be indistinguishable from the normal distribution. To achieve this indistinguishability for the searchable ciphertext, we use the random values r_1 and r_2. These random values make it hard to distinguish $g^{x(r_1+r_2)}$ from the random $g^{\varkappa(r_1+r_2)}$. In the searchable ciphertext, the elements C_1, C_2, and C_4 are used to make the semi-functional and the normal searchable ciphertext indistinguishable. The element V^{r_3} in C_2 and the element C_3 are used to cancel out the values remained during decryption. The elements $(g^{bd})^{r_3}$ and $(g^{abc})^{r_3}$ in C_4 and C_5 respectively are orthogonal to the trapdoor such that they are used to achieve the security proof. The elements C_6 and C_7 make a commitment to the random r_1 and C_8 and C_9 make a commitment to the random r_2. The element C_{12} is used to make a commitment to the keyword W and the tag τ_C. To provide keyword hiding, C_{12} is randomized using the random values r_3 and r_4. A commitment to the random values r_3 and r_4 is provided by C_{10} and C_{11} respectively. The elements C_{14} and C_{15} are extra random elements which are used for the security proof.

In the trapdoor, T_1, T_7 and T_9 provide indistinguishability of the semi-functional trapdoor using the random values s_1 and s_2. The commitments to s_1 and s_2 are performed by T_3 and T_4. The elements T_{10} and T_{11} provide commitments to the queried keyword W and the tag τ_D. The elements T_6, T_8, T_{12}, T_{13}, and T_{14} are used to cancel out the extra elements upon the decryption.

6.7.3 Sequence of games

We use a sequence of games – according to the DSE method – for the security proof. The first game is the semantic security game, and the last game is a perfectly secure game. We show that each game is indistinguishable from the next game such that we can prove that the first game is indistinguishable from the perfectly secure game. The sequence of game are described below:

$Game_{Real}$: This is the actual semantic security game for searchable encryption schemes show in 2.4.

$Game_0$: This game is the same as $Game_{Real}$ except that in the challenge phase, the challenger sends a semi-functional searchable ciphertext to the adversary.

$Game_k$: In $Game_k$, where $k \in \{1, 2, ...q\}$ and q is the total number of queries made by the adversary, the first k trapdoor queries are responded by semi-functional trapdoors and the rest queries are responded by normal trapdoors. Therefore, $Game_k$ is the same as $Game_{k-1}$ except that the response of the k-th query in $Game_k$ is a semi-functional trapdoor while in $Game_{k-1}$ the response is a normal trapdoor. In $Game_q$, all the trapdoor queries and the challenge searchable ciphertext have a semi-functional form.

$Game_{Keyword-Hiding}$: This game is identical to $Game_q$ except that the challenger creates the challenge searchable ciphertext using the message and but *a random keyword* instead of the challenge keyword.

$Game_{Final}$: This game is the same as $Game_{Keyword-Hiding}$ except that in the challenge phase, the challenger creates a semi-functional searchable ciphertext with a *random keyword* and a *random message*.

We now prove that each game is indistinguishable from the next game from the point of view of a PPT adversary.

Lemma 7. *$Game_{Real}$ is indistinguishable from $Game_0$ assuming that DLin is intractable in group $\mathbb{G}$.*

Proof:
Suppose that there exists an adversary $\mathcal{A}$ which has a non-negligible advantage ε_0 to distinguishing the two games. Then we show that we can build an algorithm B that breaks the DLin assumption with advantage ε_0. Assume that algorithm B receives a tuple $(g^{z_1}, g^{z_2}, g^{z_1 z_3}, g^{z_4}, Z_\gamma)$ from a challenger. Here, $\gamma \in \{0,1\}$ is a fair coin, such that $Z_0 = g^{z_2(z_3+z_4)}$ and Z_1 is picked randomly from $\mathbb{G}$. The goal of algorithm B is to break the DLin assumption by guessing the value γ correctly. To achieve a correct guess, algorithm B interacts with $\mathcal{A}$ in the following game:

Setup In this phase B must provide public parameters for $\mathcal{A}$. The algorithm B:

1. picks random $b, c, d, t', l, u, v, h, f, \lambda \in \mathbb{Z}_p$,
2. defines $g^{z_1} = g^a$, $g^{z_2} = g^x, g^{z_2 t'} = g^t$, $U = g^{uz_1}, V = g^{vz_1}, H = g^{hz_1}$, $F = g^{fz_1}$,
3. computes $\Omega = e(g^{z_1}, g)^{bc\lambda}$

Algorithm B then publishes the public parameters *param* as follow:

$$\left(\begin{array}{c} (g^{z_1})^b, (g^{z_1})^{bc}, (g^{z_1})^y, g^b, (g^{z_2})^{bc}, g^{bd}, (g^{z_2})^{by} g^{byt'}, g^y, (g^{z_2})^y, \\ \\ U, V, V^{\frac{1}{l}}, H, F, \Omega \end{array} \right)$$

From the adversary's point of view all the elements of *param* are chosen randomly from the group $\mathbb{G}$ and $\mathbb{G}_T$. Therefore, *param* has the same distribution as the public parameters of SEPF. The adversary also prepares two lists L_T and L_W which are initially empty.

Query I In this phase $\mathcal{A}$ adaptively makes trapdoor queries. Suppose that $\mathcal{A}$ queries the trapdoor of the keyword W. In this game, each trapdoor query must be responded with a *normal trapdoor*. To respond the query, the algorithm B first picks random $s_1, s_2, s'_3, s_4, \tau_D \in \mathbb{Z}_p$ and then simulates a normal trapdoor T_W in the following way:

$$\left(\begin{array}{l} \left(\begin{array}{ll} T_1 = & (g^{z_1})^{y(s_1+s_2)}(g^{z_1})^{-ys_4} \\ T_2 = & (g^{z_1})^{-bcs_1}(g^{z_1})^{bcs_4} \\ T_3 = & (g^{z_1})^{-bcls_1} \\ T_4 = & (g^{z_1})^{-cs_2} \\ T_5 = & g^{ds_2} \end{array} \right), \left(\begin{array}{ll} T_6 = & g^{-bct's_2}g^{bcs'_3} \\ T_7 = & g^{-ys'_3}g^{-\lambda} \\ T_8 = & (g^{z_1})^{-bct's_2}(g^{z_1})^{bcs'_3} \\ T_9 = & (g^{z_1})^{-ys'_3}(g^{z_1})^{-\lambda} \end{array} \right), \\ \\ \left(\begin{array}{ll} T_{10} = & (g^{z_1})^{(uW+v\tau_D+h)s_1}(g^{z_1})^{(f+v\tau_D)s_4} \\ T_{11} = & (g^{z_1})^{(uW+v\tau_D+h)cs_1} \\ T_{12} = & (g^{z_1})^{-bcs_1} \\ T_{13} = & (g^{z_1})^{-bcs_4} \\ T_{14} = & (g^{z_1})^{cs_4} \end{array} \right), \tau_D \end{array} \right)$$

Let $s_3 = z_2(s_1+s_2)+s'_3$. Then T_W is a correct trapdoor, because T_W is created using a master secret key that has the same distribution with the master secret key of the normal scheme. Also from the adversary's point of view all the values s_1, s_2, s_3, and s_4 are picked randomly, and the trapdoor has the same distribution with the distribution of the normal trapdoor. Algorithm B then sends T_W to $\mathcal{A}$. The adversary appends T_W to the list L_T, and W to the list L_W.

Challenge Once $\mathcal{A}$ decides that the query phase is over, she sends two challenge messages (M_0, M_1) and two challenge keywords (W_0, W_1) to the algorithm B. Here, the condition is that the challenge keywords must not been issued in Query Phase I. Given the challenge messages and keywords, algorithm B first picks a message M_β and a keyword W_β randomly, where $\beta \in \{0, 1\}$. Algorithm B then picks random $r'_3, r_4, r_5, \tau_C \in \mathbb{Z}_p$ and simulates the following searchable ciphertext S_{M_β, W_β}:

$$\left(\begin{array}{l} M_\beta e(g^{z_1z_3}, g)^{bc\lambda}e(g^{z_1}, g^{z_4})^{bc\lambda}, \\ \\ \left(\begin{array}{ll} C_1 = & (Z_\gamma)^{bc} \\ C_2 = & (Z_\gamma)^{y}(g^{z_1z_3})^{\frac{-t'yv}{d}}(g^{z_1})^{vr'_3} \\ C_3 = & ((g^{z_1})^{\frac{v}{l}})^{r_4} \\ C_4 = & (Z_\gamma)^{by}(g^{z_4})^{byt'}g^{bdr'_3} \\ C_5 = & (g^{z_1z_3})^{-\frac{bct'y}{d}}(g^{z_1})^{bcr'_3} \end{array} \right), \left(\begin{array}{ll} C_6 = & (g^{z_1z_3})^{y} \\ C_7 = & (g^{z_1z_3})^{bc} \\ C_8 = & (g^{z_4})^{y} \\ C_9 = & (g^{z_4})^{bc} \end{array} \right), \\ \\ \left(\begin{array}{ll} C_{10} = & (g^{z_1z_3})g^{-\frac{bct'y}{d}}(g^{z_1})^{bcr'_3} \\ C_{11} = & (g^{z_1})^{br_4} \\ C_{12} = & (g^{z_1z_3})^{-(uW_\beta+v\tau_C+h)(\frac{yt'}{d})}(g^{z_1})^{(uW_\beta+v\tau_C+h)(r'_3+r_4)} \\ C_{13} = & g^{r_5}(g^{z_1z_3})^{-(f+v\tau_C)(\frac{yt'}{d})}(g^{z_1})^{(f+v\tau_C)r'_3} \\ C_{14} = & (g^{b})^{r_5} \end{array} \right), \tau_C \end{array} \right).$$

Let $r_1 = z_3$, $r_2 = z_4$, $r_3 = -\frac{yt'}{d}z_3 + r'_3$. Here, S_{M_β, W_β} is a normal searchable

ciphertext if $\gamma = 0$ (i.e. $Z_\gamma = g^{z_2(z_3+z_4)}$), because S_{M_β,W_β} is created using correct public parameters and the values r_1, r_2 , r_3, r_4, and r_5 are picked randomly from the adversary's point of view. Otherwise, Z_γ is random and S_{M_β,W_β} will have a semi-functional distribution. The algorithm B then sends S_{M_β,W_β} to $\mathcal{A}$.

Query II This is the same as Query Phase I except that the adversary cannot query for the challenge keywords.

Output Finally, the adversary using L_T outputs a bit β' which represents its guess for β. The adversary then sends the guess β' to B, which uses this value to represent its guess γ' for the bit γ.

In case ε_0, which is the advantage of $\mathcal{A}$ in distinguishing $Game_0$ and $Game_{Real}$, is non-negligible, the advantage of B in guessing γ will be non-negligible, as required.

Lemma 8. *$Game_{k-1}$ is indistinguishable from $Game_k$, for $k = 1, ..., q$, where q is the maximum number of the trapdoor queries by the attacker, assuming that DLin is intractable in group $\mathbb{G}$.*

Proof:
Similarly to the proof of Lemma 7, suppose that there exists an adversary $\mathcal{A}$ which has a non-negligible advantage ε_k to distinguishing the two games. Then we show that we can build an algorithm B that breaks the DLin assumption with the non-negligible advantage ε_k. Assume that the algorithm B receives a tuple $(g^{z_1}, g^{z_2}, g^{z_1 z_3}, g^{z_4}, Z_\gamma)$ from a challenger. Here, γ is a fair coin such that $Z_0 = g^{z_2(z_3+z_4)}$ and Z_1 is picked randomly from $\mathbb{G}$. The goal of algorithm B is to break the DLin assumption by guessing the value γ correctly. To achieve a correct guess, the algorithm B interacts with $\mathcal{A}$ in the following game to use the power of $\mathcal{A}$ for breaking DLin

Setup Algorithm B provides public parameters for $\mathcal{A}$ using the following steps:

1. picks random $a, c, d, l, \tilde{t}, x, u, v, h, f, A, B, \lambda \in \mathbb{Z}_p$,
2. defines $g^{z_1} = g^b$, $g^{z_2} = g^y$, $U = (g^{z_1})^u (g^{z_2})^{-A}$, $V = (g^{z_1})^v (g^{z_2})$, $H = (g^{z_1})^h (g^{z_2})^{-B}, F = (g^{z_1})^f$.
3. computes $\Omega = e(g^{z_1}, g^{z_1})^{acv\lambda} e(g^{z_1}, g^{z_2})^{ac\lambda}$.

The simulated public parameters *param* are:

$$\left(\begin{array}{c} (g^{z_1})^a, (g^{z_1})^{ac}, (g^{z_2})^a, g^{z_1}, (g^{z_1})^{cx}, (g^{z_1})^d, (g^{z_2})^{\tilde{t}}, g^{z_2}, (g^{z_2})^x, \\ \\ U, V, V^{\frac{1}{l}}, H, F, \Omega \end{array} \right)$$

Let $t = \frac{\tilde{t}}{z_1}$. Since from the point of view of $\mathcal{A}$, all the elements of *param* are picked randomly from the groups $\mathbb{G}$ and $\mathbb{G}_T$, *param* has a correct distribution. The adversary $\mathcal{A}$ also prepares two lists L_T and L_W which are initially empty.

Query I, II Here, for more simplicity we explain Query I and II together. In these phases, $\mathcal{A}$ adaptively makes q trapdoor queries. Suppose that $\mathcal{A}$ queries the trapdoor of the keyword W. Here, based on the number of query, i, the response of B to the query is one of the cases below:

- In case $i \leq k-1$, algorithm B picks $\mathsf{y}, s_1, s_2, s_3, s_4, \tau_D \in \mathbb{Z}_p$ randomly and generates a *semi-functional* trapdoor T'_W using the master secret key and public parameters.
- In case $i = k$, algorithm B picks random $s_3, s_4 \in \mathbb{Z}_p$ and generates the following trapdoor:

$$\left(\begin{array}{c} \left(\begin{array}{cc} T_1 = & (Z_\gamma)^a (g^{z_2})^{-as_4} \\ T_2 = & (g^{z_1 z_3})^{-ac} (g^{z_1})^{acs_4} \\ T_3 = & (g^{z_1 z_3})^{-acl} \\ T_4 = & (g^{z_4})^{-ac} \\ T_5 = & (g^{z_4})^{d} \end{array} \right), \\ \left(\begin{array}{cc} T_6 = & (g^{z_1 z_3})^{-cx} (g^{z_4})^{c\tilde{t}} (g^{z_1})^{cs_3} \\ T_7 = & (Z_\gamma)^x (g^{z_2})^{-s_3} g^{-\lambda} \\ T_8 = & (g^{z_1 z_3})^{-acx} (g^{z_4})^{ac\tilde{t}} (g^{z_1})^{acs_3} \\ T_9 = & (Z_\gamma)^{ax} (g^{z_2})^{-as_3} g^{-a\lambda} \end{array} \right), \\ \left(\begin{array}{cc} T_{10} = & (g^{z_1 z_3})^{(uW+v\tau_D+h)} (g^{z_1})^{fs_4} ((g^{z_1})^v g^{z_2})^{\tau_D s_4} \\ T_{11} = & (g^{z_1 z_3})^{(uW+v\tau_D+h)c} \\ T_{12} = & (g^{z_1 z_3})^{-ac} \\ T_{13} = & (g^{z_1})^{-acs_4} \\ T_{14} = & g^{acs_4} \end{array} \right), \tau_D \end{array} \right)$$

  Let $s_1 = z_3$, $s_2 = z_4$,$\tau_D = AW + B$. If $\gamma = 0$, then B has simulated a normal trapdoor, otherwise B has simulated a semi-functional trapdoor. Here, simulating a correct trapdoor is possible only if we choose $\tau_D = AW + B$. If any other value is chosen for τ_D, algorithm B must compute $g^{z_1 z_2}$ in T_{10} and T_{11}, which is not possible.
- In case $i > k$, given the keyword W, algorithm B picks random $s_1, s_2, s_3, s_4, \tau_D \in \mathbb{Z}_p$ randomly and generates a normal trapdoor T_W.

Algorithm B then sends the simulated trapdoor to $\mathcal{A}$. The adversary $\mathcal{A}$ appends the simulated trapdoor to the list L_T, and W to the list L_W.

Challenge In this phase $\mathcal{A}$ sends two challenge messages (M_0, M_1) and two challenge keywords (W_0, W_1) to B. Given the challenge messages and keywords, B first flips a fair coin $\beta \in \{0, 1\}$ and picks (M_β, W_β). The algorithm B then picks random $r_1, r_2, r'_3, r_4, r'_5, \tilde{x} \in \mathbb{Z}_p$ and simulates the following semi-functional sear-

chable ciphertext S'_{M_β,W_β}:

$$\left(\begin{array}{c} M_\beta e(g^{z_1},g)^{ac\lambda(r_1+r_2)}, \\ \left(\begin{array}{ll} C_1 = & (g^{z_1})^{cx(r_1+r_2)}g^{c\tilde{x}(r_1+r_2)} \\ C_2 = & (g^{z_2})^{x(r_1+r_2)}g^{-v\tilde{x}(r_1+r_2)}(g^{z_2}(g^{z_1})^{v})^{r'_3} \\ C_3 = & (g^{z_2})^{\frac{r_4}{l}}(g^{z_1})^{\frac{vr_4}{l}} \\ C_4 = & (g^{z_2})^{(\tilde{t}+\tilde{x})(r_1+r_2)}g^{-d\tilde{x}(r_1+r_2)}(g^{z_1})^{dr'_3} \\ C_5 = & g^{-ac\tilde{x}(r_1+r_2)}(g^{z_1})^{acr'_3} \end{array}\right), \\ \left(\begin{array}{ll} C_6 = & (g^{z_2})^{ar_1} \\ C_7 = & (g^{z_1})^{acr_1} \\ C_8 = & (g^{z_2})^{r_2} \\ C_9 = & (g^{z_1})^{cr_2} \end{array}\right), \\ \left(\begin{array}{ll} C_{10} = & g^{-ac\tilde{x}(r_1+r_2)}(g^{z_1})^{acr'_3} \\ C_{11} = & (g^{z_1})^{ar_4} \\ C_{12} = & g^{-(uW_\beta+v\tau_C+h)\tilde{x}(r_1+r_2)}(g^{z_1})^{(uW_\beta+v\tau_C+h)(r'_3+r_4)} \\ C_{13} = & g^{r'_5}(g^{-f\tilde{x}(r_1+r_2)})(g^{z_1})^{fr'_3}(g^{z_2})^{\tau_C r'_3}g^{-v\tilde{x}(r_1+r_2)\tau_C}(g^{z_1})^{v\tau_C r'_3} \\ C_{14} = & (g^{z_1})^{r'_5}g^{\tilde{x}(r_1+r_2)\tau_C} \end{array}\right), \tau_C \end{array}\right).$$

Let $\mathsf{x} = x + \frac{\tilde{x}}{z_1}$, $\tau_C = AW_\beta + B$, $r_3 = -\frac{(r_1+r_2)\tilde{x}}{z_1} + r'_3$, $r_5 = \frac{z_2(r_1+r_2)\tau_C\tilde{x}}{z_1} + r'_5$. Then all the elements of S'_{M_β,W_β} are picked randomly from groups $\mathbb{G}$ and $\mathbb{G}_T$. Therefore, S'_{M_β,W_β} is a correct semi-functional searchable ciphertext. Observe that simulating S'_{M_β,W_β} is possible only if we choose $\tau_C = AW_\beta + B$. Otherwise, to simulate the element C_{12}, the algorithm B must compute $g^{\frac{z_2}{z_1}}$ which is not possible.

Output Finally, the adversary using L_T outputs a bit β' which represents its guess for β. The adversary then sends the guess β' to B, which uses this value to represent its guess γ' for the bit γ.

In case ε_k, which is the advantage of $\mathcal{A}$ in distinguishing $Game_{k-1}$ and $Game_k$, is non-negligible, the advantage of B in guessing γ will be non-negligible, as required.

Lemma 9. *$Game_q$ is indistinguishable from $Game_{Keyword-Hiding}$ assuming that DLin is intractable in group $\mathbb{G}$.*

Proof:
Suppose that there exists an adversary $\mathcal{A}$ which has a non-negligible advantage ε_{q+1} in distinguishing the two games. We show how we can build an algorithm B that has a non-negligible advantage ε_{q+1} in breaking the DLin problem. Assume that algorithm B receives a tuple $(g^{z_1}, g^{z_2}, g^{z_1 z_3}, g^{z_4}, Z_\gamma)$ from a challenger, where γ is a fair coin such that $Z_0 = g^{z_2(z_3+z_4)}$ and Z_1 is picked randomly from $\mathbb{G}$. Algorithm B then performs the following game with the adversary $\mathcal{A}$.

Setup Algorithm B provides the public parameters using the following steps:

1. pick random $a, b, \tilde{d}, \tilde{l}, t, x, u, v, h, \tilde{f}, f', \lambda \in \mathbb{Z}_p$,
2. define $g^{z_1} = g^c$, $(g^{z_1})^{\tilde{d}}(g^{z_2})^v = g^d, g^{z_2} = g^y, U = (g^{z_2})^u, V = (g^{z_2})^v, H = (g^{z_2})^h$, $F = (g^{z_2})^{\tilde{f}}(g^{z_1})^{f'}$.
3. compute $\Omega = e(g^{z_2}, g^{z_1})^{abv\lambda}$

The algorithm B then publishes the public parameters *param* as follow:

$$\left(\begin{array}{c} g^{ab}, (g^{z_1})^{ab}, (g^{z_2})^a, g^b, (g^{z_1})^{bx}, (g^{z_1})^{b\tilde{d}}(g^{z_2})^{bv}, (g^{z_2})^{bt}, g^{z_2}, (g^{z_2})^x, \\ \\ U, V, g^{\frac{v}{\tilde{l}}}, H, F, \Omega \end{array} \right)$$

Let $l = \tilde{l}z_2$. Then *param* has a correct distribution because from the adversary's point of view all the elements of *param* are picked randomly from $\mathbb{G}$. The adversary also prepares two lists L_T and L_W which are initially empty.

Query I In this game, each query must be responded with a semi-functional trapdoor. Suppose that $\mathcal{A}$ queries for the trapdoor of the keyword W. Given W, the algorithm B picks random $\tilde{s}_1, s_2, s_3', s_4', \tilde{y}, \tau_D \in \mathbb{Z}_p$ and simulates the following semi-functional trapdoor T_W':

$$\left(\begin{array}{l} \left(\begin{array}{ll} T_1' = & g^{a\tilde{y}\tilde{s}_1}(g^{z_2})^{a\frac{(uW+v\tau_D+h)s_2}{\tilde{f}+v\tau_D}}(g^{z_1})^{a\tilde{y}s_2}(g^{z_2})^{-as_4'} \\ T_2' = & g^{-ab\tilde{s}_1}g^{\frac{-ab(u\bar{W}+v\tau_D+h)\tilde{s}_1}{(\tilde{f}+v\tau_D)}}(g^{z_1})^{abs_4'} \\ T_3' = & (g^{z_2})^{-ab\tilde{l}\tilde{s}_1} \\ T_4' = & (g^{z_1})^{-as_2} \\ T_5' = & (g^{(z_1)\tilde{d}}(g^{z_2})^{vs_2} \end{array} \right), \\ \\ \left(\begin{array}{ll} T_6' = & g^{-bx\tilde{s}_1}(g^{z_1})^{-bxts_2}g^{b(\frac{x(u\bar{W}+v\tau_D+h)\tilde{s}_1}{(\tilde{f}+v\tau_D)})}(g^{z_1})^{bs_3'} \\ T_7' = & g^{-x\tilde{y}\tilde{s}_1}(g^{z_1})^{-x\tilde{y}s_2}(g^{z_2})^{s_3'}(g^{z_2})^{-\frac{(uW+v\tau_D+h)xs_2}{\tilde{f}+v\tau_D}} \\ T_8' = & g^{-abx\tilde{s}_1}(g^{z_1})^{-abxts_2}g^{ab(\frac{x(u\bar{W}+v\tau_D+h)\tilde{s}_1}{(\tilde{f}+v\tau_D)})}(g^{z_1})^{abs_3'} \\ T_9' = & g^{-ax\tilde{y}\tilde{s}_1}(g^{z_1})^{-ax\tilde{y}s_2}(g^{z_2})^{as_3'}(g^{z_2})^{-\frac{(uW+v\tau_D+h)axs_2}{\tilde{f}+v\tau_D}} \end{array} \right), \\ \\ \left(\begin{array}{l} \left(\begin{array}{ll} T_{10}' = & (g^{z_2})^{(\tilde{f}+v\tau_D)s_4'}g^{-f'\tilde{s}_1(\frac{uW+v\tau_D+h}{\tilde{f}+v\tau_D})}(g^{z_1})^{f's_4'} \\ T_{11}' = & (g^{z_2})^{(uW+v\tau_D+h)\tilde{s}_1} \\ T_{12}' = & g^{-ab\tilde{s}_1} \\ T_{13}' = & g^{-ab(\frac{uW+v\tau_D+h}{\tilde{f}+v\tau_D})\tilde{s}_1}(g^{z_1})^{-abs_4'} \\ T_{14}' = & g^{a(\frac{uW+v\tau_D+h}{\tilde{f}+v\tau_D})\tilde{s}_1}(g^{z_1})^{as_4'} \end{array} \right), \tau_D \end{array} \right) \end{array} \right)$$

Let $s_1 = \frac{\tilde{s}_1}{z_1}$, $s_3 = \frac{x(uW+v\tau_D+h)\tilde{s}_1}{(\tilde{f}+v\tau_D)z_1} + s_3'$, $s_4 = -\frac{(uW+v\tau_D+h)\tilde{s}_1}{(\tilde{f}+v\tau_D)z_1} + s_4'$, $\mathsf{y} = \frac{(uW+v\tau_D+h)z_2}{\tilde{f}+v\tau_D} + \frac{\tilde{y}}{z_1}$. Then the simulated T_W' has a correct distribution because from the adversary's point of view s_1, s_2, s_3, s_4, y are picked randomly

and T'_W uses a correct master secret key. The algorithm B sends T'_W to $\mathcal{A}$, who appends T'_W to L_T, and W to L_W.

Challenge Once $\mathcal{A}$ decides that the trapdoor query phase is over, she picks two challenge messages (M_0, M_1) and two challenge keywords (W_0, W_1), and sends them to B. Here, the condition is that the challenge keywords must not be used in Query Phase I. Given the challenge messages and keywords B first flips a fair coin $\gamma \in \{0,1\}$ and picks M_β and W_β. The algorithm B then picks random $\tilde{x}, r_1, r_2, r_5, \tau_C \in \mathbb{Z}_p$, and simulates the following semi-functional searchable ciphertext S'_{M_β, W_β}: $\mathcal{A}$.

$$\left(\begin{array}{c} M_\beta e(g^{z_1}, g)^{ac\lambda(r_1+r_2)}, \\ \left(\begin{array}{cc} C'_1 = & (g^{z_1})^{b\tilde{x}(r_1+r_2)}(g^{z_1 z_3})^{-bv} \\ C'_2 = & (g^{z_2})^{\tilde{x}(r_1+r_2)} \\ C'_3 = & (g^{z_4})^{\frac{v}{l}} \\ C'_4 = & (g^{z_2})^{bt(r_1+r_2)}(g^{z_2})^{b(\tilde{x}-x)(r_1+r_2)}(g^{bz_2\tilde{x}(r_1+r_2)})(g^{z_1 z_3})^{b\tilde{d}} \\ C'_5 = & (g^{z_1 z_3})^{ab} \end{array} \right), \\ \left(\begin{array}{cc} C'_6 = & (g^{z_2})^{ar_1} \\ C'_7 = & (g^{z_1})^{abr_1} \\ C'_8 = & (g^{z_2})^{r_2} \\ C'_9 = & (g^{z_1})^{br_2} \end{array} \right), \\ \left(\begin{array}{cc} C'_{10} = & (g^{z_1 z_3})^{ab} \\ C'_{11} = & (g^{z_4})^{ab} \\ C'_{12} = & (Z_\gamma)^{uW_\beta + \tilde{f} + h} \\ C'_{13} = & g^{r_5}(g^{z_1 z_3})^{f'} \\ C'_{14} = & g^{br_5} \end{array} \right), \tau_C \end{array} \right).$$

Let, $r_3 = z_3$, $r_4 = z_4$, $\tau_C = -\frac{\tilde{f}}{v}$, $\mathsf{x} = \tilde{x} - \frac{vz_3}{r_1+r_2}$. If $\gamma = 0$, then S'_{M_β, W_β} is the semi-functional searchable ciphertext of the message M_β and the keyword W_β. Otherwise, the value Z_γ is random and S'_{M_β, W_β} is the semi-functional searchable ciphertext of a random message and a random keyword.

Query II This phase is identical to Query Phase I except that the adversary is not allowed to query for the challenge keywords.

Output Finally, the adversary using L_T outputs a bit β' which represents its guess for β. The adversary then sends the guess β' to B, which uses this value to represent its guess γ' for the bit γ.

In case ε_{q+1}, which is the advantage of $\mathcal{A}$ in distinguishing $Game_q$ and $Game_{Keyword-Hiding}$, is non-negligible, the advantage of B in guessing γ will be non-negligible, as required.

Lemma 10. *$Game_{Keyword-Hiding}$ is indistinguishable from $Game_{Final}$ assuming that DBDH is intractable in group $\mathbb{G}_T$*

Proof:
Suppose that there exists an adversary $\mathcal{A}$ which has a non-negligible advantage ε_{q+2} in distinguishing the two games. We show how we can build an algorithm B that has a non-negligible advantage ε_{q+2} in breaking the DBDH assumption. Assume that algorithm B receives a tuple $(g^{z_1}, g^{z_2}, g^{z_3}, Z_\gamma)$ from a challenger, where γ is a fair coin such that $Z_0 = e(g,g)^{z_1 z_2 z_3}$ and Z_1 is picked randomly from $\mathbb{G}_T$. Algorithm B then performs the following game with the adversary $\mathcal{A}$.

Setup The algorithm B provides the public parameters as follows:

1. pick random $a, b, c, \tilde{d}, l, t, y, u, v, h, f \in \mathbb{Z}_p$,
2. define $g^{z_1} = g^x, g^g = (g^{z_3})^{\tilde{d}}, \Omega = e(g^{z_1}, g^{z_2})^{abc}$.

The algorithm B then publishes the public parameters *param* as follow: The public parameters are:

$$\left(g^{ab}, g^{abc}, g^{ay}, g^b, (g^{z_1})^{bc}, (g^{z_3})^{b\tilde{d}}, g^{byt}, g^y, (g^{z_1})^y, U, V, V^{\frac{1}{l}}, H, F, \Omega\right)$$

Let $\lambda = z_1 z_2$. Then *param* has a correct distribution because from the adversary's point of view all the elements of *param* are picked randomly from $\mathbb{G}$. The adversary also prepares two lists L_T and L_W which are initially empty.

Query I In this game, each query must be responded with a *semi functional* trapdoor. Suppose that $\mathcal{A}$ queries for the trapdoor of the keyword W. Given W, the algorithm B picks random $\tilde{s}_1, s_2, s_3, s_4, \mathsf{y}, \tau_D \in \mathbb{Z}_p$ and simulates the following semi-functional trapdoor T'_W:

$$\left(\begin{pmatrix} T'_1 = & (g^{z_2})^{-a} g^{-ays_4} \\ T'_2 = & g^{-abcs_1} g^{abcs_4} \\ T'_3 = & g^{-abcls_1} \\ T'_4 = & g^{-acs_2} \\ T'_5 = & (g^{z_3})^{\tilde{d}s_2} \end{pmatrix}, \begin{pmatrix} T'_6 = & (g^{z_1})^{-bcs_1} g^{-bcts_2} g^{bcs_3} \\ T'_7 = & g^{-\mathsf{y}s_3} \\ T'_8 = & (g^{z_1})^{-abcs_1} g^{-abcts_2} g^{abcs_3} \\ T'_9 = & g^{-ays_3} \end{pmatrix}, \right.$$
$$\left. \begin{pmatrix} T'_{10} = & (U^W V^{\tau_D} H)^{s_1} (FV^{\tau_D})^{s_4} \\ T'_{11} = & (U^W V^{\tau_D} H)^{cs_1} \\ T'_{12} = & g^{-abcs_1} \\ T'_{13} = & g^{-abcs_4} \\ T'_{14} = & g^{acs_4} \end{pmatrix}, \tau_D \right).$$

Let $\mathsf{y} = \frac{z_2}{s_1 + s_2}$. Then the simulated T'_W has a correct distribution because it is created using the correct master secret key, and from adversary's point of view all the values s_1, s_2, s_3, s_4, y are picked randomly. The algorithm B sends T'_W to $\mathcal{A}$, who appends T'_W to L_T, and W to L_W.

Challenge Once $\mathcal{A}$ decides that the trapdoor query phase is over, she picks two challenge messages (M_0, M_1) and two challenge keywords (W_0, W_1), and sends them to B. Here, the condition is that the challenge keywords must not be used in Query Phase I. Given the challenge messages and keywords B first flips a fair coin $\gamma \in \{0,1\}$ and picks M_β and W_β. The algorithm B then picks random $\tilde{x}, r_1, r_2, r_5, \tau_C \in \mathbb{Z}_p$, and simulates the following semi-functional searchable ciphertext S'_{M_β, W_β}: $\mathcal{A}$.

$$\left(\begin{array}{c} Me(Z_\gamma)^{abc}, \\ \left(\begin{array}{ll} C'_1 = & (g^{z_3})^{bc\tilde{x}} \\ C'_2 = & (g^{z_3})^{\tilde{x}y}(g^{z_1})^{\frac{vy}{d}} \\ C'_3 = & (V^{\frac{1}{t}})^{r_4} \\ C'_4 = & (g^{z_3})^{byt}(g^{z_3})^{b\tilde{x}y} \\ C'_5 = & (g^{z_1})^{\frac{abcy}{d}} \end{array} \right), \left(\begin{array}{ll} C'_6 = & g^{ayr_1} \\ C'_7 = & g^{abcr_1} \\ C'_8 = & (g^{z_3})^{y} g^{-r1y} \\ C'_9 = & (g^{z_3})^{bc} g^{-bcr_1} \end{array} \right), \\ \left(\begin{array}{ll} C'_{10} = & (g^{z_1})^{\frac{abc}{d}} \\ C'_{11} = & g^{abr_4} \\ C'_{12} = & (g^{z_1})^{(uR_1+v\tau_D+h)\frac{y}{d}} g^{(uR_1+v\tau_C+h)r_4} \\ C'_{13} = & g^{r_5}(g^{z_1})^{\frac{fy+yv\tau_C}{d}}(FV^{\tau_C})^{r_3} \\ C'_{14} = & (g^b)^{r_5} \end{array} \right), \tau_C \end{array} \right).$$

Let, $r_2 = z_3 - r_1$, $r_3 = \frac{y}{d} z_1$. If $\gamma = 0$, then S'_{M_β, R_1} is the semi-functional searchable ciphertext of the message M_β and the random R_1. Otherwise, the value Z_γ is random and S'_{M_β, R_1} is the semi-functional searchable ciphertext of a random message and the random keyword R_1.

Query II This phase is identical to Query Phase I except that the adversary is not allowed to query for the challenge keywords.

Output Finally, the adversary using L_T outputs a bit β' which represents its guess for β. The adversary then sends the guess β' to B, which uses this value to represent its guess γ' for the bit γ.

In case ε_{q+2}, which is the advantage of $\mathcal{A}$ in distinguishing $Game_{Keyword-Hiding}$ and $Game_{Final}$, is non-negligible, the advantage of B in guessing γ will be non-negligible, as required.

Theorem 5. *The SEPF scheme is fully secure assuming that DLin and DBDH assumption are interactable.*

By Lemmas 7, 8, 9, and 10 the advantage of any PPT adversary to distinguish the semantic security game from $Game_{Final}$ is $\varepsilon_0 + \varepsilon_1 + ... + \varepsilon_{q+1} + \varepsilon_{q+2}$, which is negligible. This completes the security proof.

6.8 Efficiency

In Table 6.1 we compare the efficiency of the SEPF scheme with the DIP scheme. In this table we show the computations and elements of composite order group by the sign "*". Before we analyze the table we explain more details about the order of the groups in our scheme and DIP. For a scheme that uses composite order groups to be secure, the order of the group must be large enough, such that factoring the primes of the order be infeasible. If the order of the group, say N, is made up of two primes, say p_1, p_2 (i.e. $N = p_1 p_2$), N must be consists of at least 1024 bits such that factoring p_1 and p_2 is infeasible. In the DIP scheme, where the order of the group is made up of four primes, the order of the group must be at least 2048 bits. However, schemes that use prime order groups can achieve an equivalent level of security by using a smaller order, which is typically 160 bits. This requirement on the size of the groups makes the pairing and other group operations in composite order groups considerably slower than in prime order groups. For example pairing operations in composite order groups are at least 50 times slower than the pairing operations in prime order groups [22].

Table 6.1 shows that the SEPF scheme searches more efficiently than the DIP scheme. As mentioned, the complexity of two pairing operations in DIP, which use composite order groups, is equivalent to the complexity of at least 100 pairing operations in prime order groups. This complexity is higher than the search complexity of SEPF which is 14 prime order pairings.

The communication complexity of our scheme is higher than the communication complexity of DIP. The DIP scheme uses a lower number of groups elements for the searchable ciphertext and the trapdoor compared to SEPF. However, each group element in DIP is larger than each group element in SEPF. We note that each group element in DIP belongs to one of the subgroups of the composite order group. Here, to achieve a more concrete comparison, we use some typical numerical values for the group orders, which is 160 bits for our scheme and 2048 bits for DIP. We assume that each prime of the group order in DIP has 512 bits. In this case, the size of each group element in DIP is 3.2 times more than the size of each group element in SEPF. Therefore, in this example, the size of the searchable ciphertext and the trapdoor in DIP are 1536 and 1024 bits respectively which is lower than SEPF. In SEPF the size of the searchable ciphertext and the trapdoor is 2240 bits. However, the size of the public parameters in SEPF, which is 2400 bits, is lower than the size of the public parameters in DIP, which is 3072 bits.

Table 6.1 shows that the DIP scheme performs fewer group multiplication and exponentiations compared to SEPF. However, group operations in DIP are slower than group operations in SEPF because the size of the elements in DIP is larger than the size of the elements in SEPF. In general, the complexity of the group computation is $O(n)$, where n is the number of bits of the group elements. Therefore, a concrete comparison in this respect can be done if the efficiency of the device performing the computation is known.

		SEPF	DIP10
Computational Complexity	Searchable ciphertext	18 exp	4 exp*
	Trapdoor	16 exp	7 exp*
	Search	14 prime order pairing	2 composite order pairing
Communication Complexity	Searchable Ciphertext	14 group elements	3 group elements*
	Trapdoor	14 group elements	2 group elements*
Storage Complexity	Master secret key (bits)	9σ	σ
	Public parameters	15 group elements	6 group elements*
	Searchable Ciphertext	14 group elements	3 group elements*

Table 6.1: *Comparison of the complexity of our scheme with the DIP scheme. The sign "*" shows that the operation and the group element belongs to a composite order group.*

6.9 Conclusion

We present the SEPF scheme which is fully secure in the public key setting. The SEPF scheme is the first fully secure scheme which has the following features: i) SEPF uses bilinear groups of prime orders, and ii) SEPF provides security based on a weak assumption, which is the DLin assumption. We complete the security proof using the dual system encryption methodology and the DLin assumption. Using bilinear groups of prime order makes the search in SEPF more efficient than the search in DIP, which uses composite order groups to achieve full security. The SEPF scheme also has a lower storage complexity for the public parameters compared to DIP. However, DIP has a lower communication complexity compared to SEPF. Therefore,

SEPF is suitable for encrypting highly sensitive documents, when the communication complexity is not crucial.

Chapter 7

Conclusions

In this section we summarize the contributions of this thesis, in relation to the Research Question described in Chapter 1. We also highlight future research directions in the area of searchable encryption.

In the introductory chapter we formulate the following research question:
Can we construct provably secure searchable encryption schemes with a complexity as close as possible to plaintext search?

Our research question focuses on two aspects of searchable encryption: security and efficiency. The security is evaluated based on the security model of the scheme and the efficiency is evaluated based on the complexity of the construction. Since security is never free, there is a trade off between these aspects. The best trade off is achieved when the scheme achieves certain level of security with the lowest possible complexity.

Before we show how we answer the research question, we explain the difficulties of designing efficiently searchable encryption schemes. Provably secure schemes are designed in a standard pattern; first choose a security model and then create a construction for the scheme and then check whether the construction can be proven to be secure in the chosen security model. In case the security proof cannot be achieved, adjust the construction (by adding more random elements in appropriate positions) and check the security proof again. This cycle is continued until a security proof is found. In case the security cannot be proven in the chosen model, a new security model should be chosen and the construction should be adjusted for the new model. Once the security proof has been completed, the complexity of the scheme is analyzed. The scheme is efficient if the construction has a lower complexity than existing schemes in the same model. Hence, the main challenge is how to adjust the construction in such a way that the security can be proven and in the same time to keep the complexity as low as possible?

Complexity aspects of searchable encryption are not independent – reducing the complexity of one aspect results in increasing the complexity of other complexity aspects. For example, reducing the complexity of the search usually increases the

complexity of the searchable ciphertext or the trapdoor. Therefore, to keep the complexity low during the design process, we need to focus on specific complexity aspects.

In this thesis, we answer the research question by focusing on the complexity of the search, which is the main functionality of searchable encryption. We propose searchable encryption schemes which have a lower search complexity compared to that of existing schemes. Each of our schemes achieves certain level of security. In Chapter 3 we propose the SES scheme, which searches for a single keyword with low complexity. The SES scheme is secure in the symmetric key security model. In Chapter 4 we propose the SEPE scheme, which has a low search complexity and enforces the access policy in the public key setting. The SEPE scheme is secure in the random oracle model. The SEPS scheme, which is proposed in Chapter 5, has a low complexity to search keywords with wildcards in the public key setting. The SEPS scheme is secure in the selective security model. In Chapter 6, we propose the SEPF scheme, which has a low search complexity in the public key setting. The SEPF scheme is proven to be fully secure.

In Figures 7.1 and 7.2 we position our schemes in relation to existing schemes with respect to the following aspects: the search efficiency, which is the main focus of this thesis, the security, and the expressiveness of the query. In Figure 7.1 we compare the symmetric key searchable encryption schemes, and in Figure 7.2 we compare the public key searchable encryption schemes. These figures have the same axes. On the security axis, well-known security models are positioned: the random oracle model, the selective security and full security models. For the query functionality we use the single keyword search and the full SQL query as the simplest and the most expressive query respectively. Between these extreme points we categorize the query functionality based on the level of expressiveness that existing schemes achieve: the keyword with wildcard search, supporting the access policy enforcement, the conjunctive keyword search and the conjunctive-disjunctive query. Supporting wildcards in the query allows searching for several keywords with don't care symbols. Search with access policy enforcement allows searching only in authorized searchable ciphertexts. Conjunctive keyword search supports "AND" among the keywords and the conjunctive-disjunctive query supports both, "AND" and "OR", among the keywords. Unlike the query functionality, the search efficiency does not have any absolute maximum and minimum values. We categorize the search efficiency of the schemes based on the efficiency of the underlying primitives used for the search. These primitives, in increasing order of efficiency, are as follows: pairing in composite order groups, pairing in prime order groups, cryptographic group operations, and search without any group operation, which in the schemes discussed here is most efficient.

Figure 7.1 visualizes the position of the SES scheme in the world of symmetric key searchable encryption. We focus again on the efficiency: for each type of query functionality we show the scheme with the most efficient search. The figure shows that SES has the lowest search complexity in the symmetric key setting. SES also achieves the same level of security as most of the existing schemes (except [38]), which are proposed in the random oracle model. However, the SES scheme can support only the single keyword search. Schemes supporting more complex functionalities, have a

higher complexity. The schemes in [38] and in [27], which are more expressive than our scheme, use less efficient primitives to perform the search compared to SES. While there are some schemes that perform pre-processing for the searchable ciphertexts to enable a wildcard search, there is currently no scheme that supports wildcards in the trapdoor. Since the access policy enforcement is not relevant in the symmetric key setting, there is no scheme which supports such functionalities.

Figure 7.2 visualizes the position of the SEPE, SEPS, and SEPF schemes in the public key searchable encryption world. The SEPF scheme resides on the top of the security axis. The SEPE and SEPS schemes are positioned in the middle of the functionality and security axes. However, these schemes use primitives which are as efficient as the primitives used in the schemes with the low security and simple functionality. The SEPF, SEPS, and SEPS schemes are also more search efficient than the existing schemes which are secure in the same model and support the same functionalities. Therefore, our schemes achieve the best trade off between security and efficiency. Each scheme achieves security, in a certain functionality, with the lowest complexity. The figure shows that the scheme in [31] supports a more expressive functionality compared to our schemes. However, this scheme has a higher search complexity than our schemes due to the use of composite order pairings. Figure 7.2 also shows that there is no public key searchable encryption scheme which uses more efficient primitives than the prime order pairing.

The comparison of Figures 7.1 and 7.2 shows that more effort has been devoted to public key searchable encryption than symmetric key searchable encryption. To explain the reason, we need to look at the applications in these settings. The main application of symmetric key searchable encryption is to retrieve encrypted data selectively from an honest but curious server. In such applications, the efficiency of the search is crucial because the size of the data stored on the server is typically huge. Otherwise, the user can retrieve and search the data by himself. To keep the search complexity low, symmetric key searchable encryption schemes achieve a low security level and support a low functionality. However, public key searchable encryption has a larger variety of applications. In each application, the importance of the security and the efficiency is different. For example, consider Scenario 2 explained in the Introduction chapter, where Bob delegates a trapdoor to Carol in order to decrypt his e-mails selectively. Typically e-mails are not large. Therefore, if Bob's e-mails are sensitive, a searchable encryption scheme in the standard model such as SEPF, which has a higher security, can be used. However, in case Bob's e-mails are not sensitive, Bob would use a scheme in the random oracle model such as [10], which is more efficient but has a weaker security.

In Figure 7.3 we compare the efficiency of our schemes with existing schemes that are secure in the same model and support the same functionalities for the query. The purpose of this figure is to show the impact of the efficient search on other complexity aspects. This figure shows that the SES scheme is more efficient than the SI and SSE schemes to search and update the database simultaneously. The cost of this efficiency is a higher complexity for the trapdoor of SES. With respect to the other complexity aspects, SES stands in the middle. The SEPE scheme is more efficient than the DGD scheme to perform the search and the policy enforcement simultaneously. We assume

that the DGD scheme uses a predicate encryption scheme to enforce the policy. The price that we pay for this efficiency is a higher communication complexity of the trapdoor. The SEPS scheme, is more efficient than the IP and BW schemes for the following aspects: the search computational complexity, the master secret key storage and the trapdoor communication. However, SEPS has a higher computational complexity of the searchable ciphertext. The SEPF scheme is more efficient than the DIP scheme to search and store the public parameters. However, SEPF has a higher communication complexity compared to DIP. That is why these schemes are recommended to be used in applications where the efficiency for the search is crucial and there are enough resources for the other complexity aspects.

Future Work the direction of future work is to improve the functionality, the security and the efficiency of searchable encryption towards supporting full SQL queries, achieving full security, and deploying more efficient primitives than pairings. In this thesis we improved the efficiency of searchable encryption. While many researchers have devoted a significant effort, there are still a large number of open problems. Here, we highlight some of the open problems for future work:

- In the **symmetric key** setting:
  - How to construct a symmetric key searchable encryption scheme which supports wildcards in the trapdoor? Existing schemes that support wildcards, either are not provably secure or not efficient. These schemes transform every possible keyword to a searchable ciphertext, such that the searchable ciphertexts match a trapdoor with wildcards.
  - How to construct an expressive symmetric key searchable encryption scheme which does not use group operations? Existing schemes use group multiplications and exponentiation for the search which makes the scheme costly.
- In the **public key** setting:
  - How to construct a fully secure searchable encryption scheme which uses prime order pairings? While there are a few schemes which achieve full security, these schemes use composite order pairings. Since composite order pairings are prohibitively slow, existing fully secure schemes are not practical.
  - How to construct a scheme which performs fuzzy search? The fuzzy search allows searching for keywords that are the same as the queried keyword except for a few letters. This type of the search functionality is suitable for applications where typos occur frequently.
  - How to construct a searchable encryption scheme which searches for a part of keywords, (e.g. the prefix keyword search, the suffix keyword search, etc)? A trivial solution is to use a scheme which searches keywords with wildcards. In this case, several trapdoors should be sent to the server,

where each trapdoor contains the queried part of the keyword and wildcards. The queried part of the keyword resides in different positions in each trapdoor, such that the rest of the trapdoor contains wildcards. Hence, the refined problem is how to search for a part of keywords using one trapdoor only?

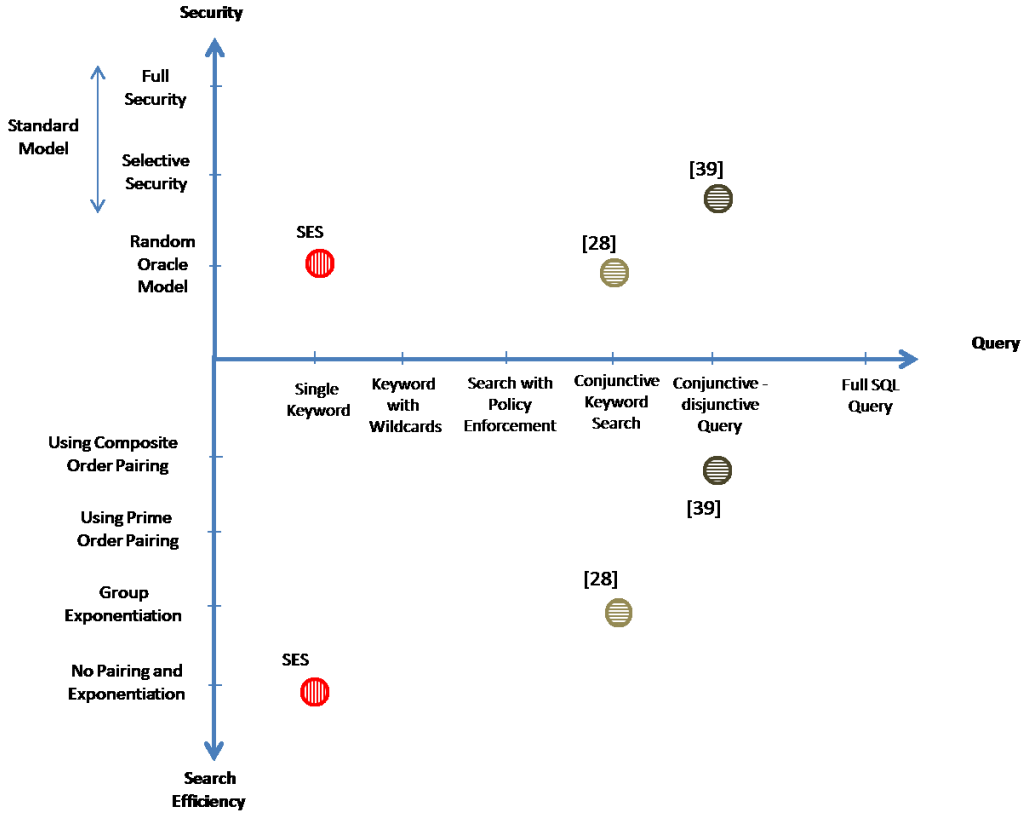

Figure 7.1: *Security and query functionality of our scheme in the symmetric key setting.*

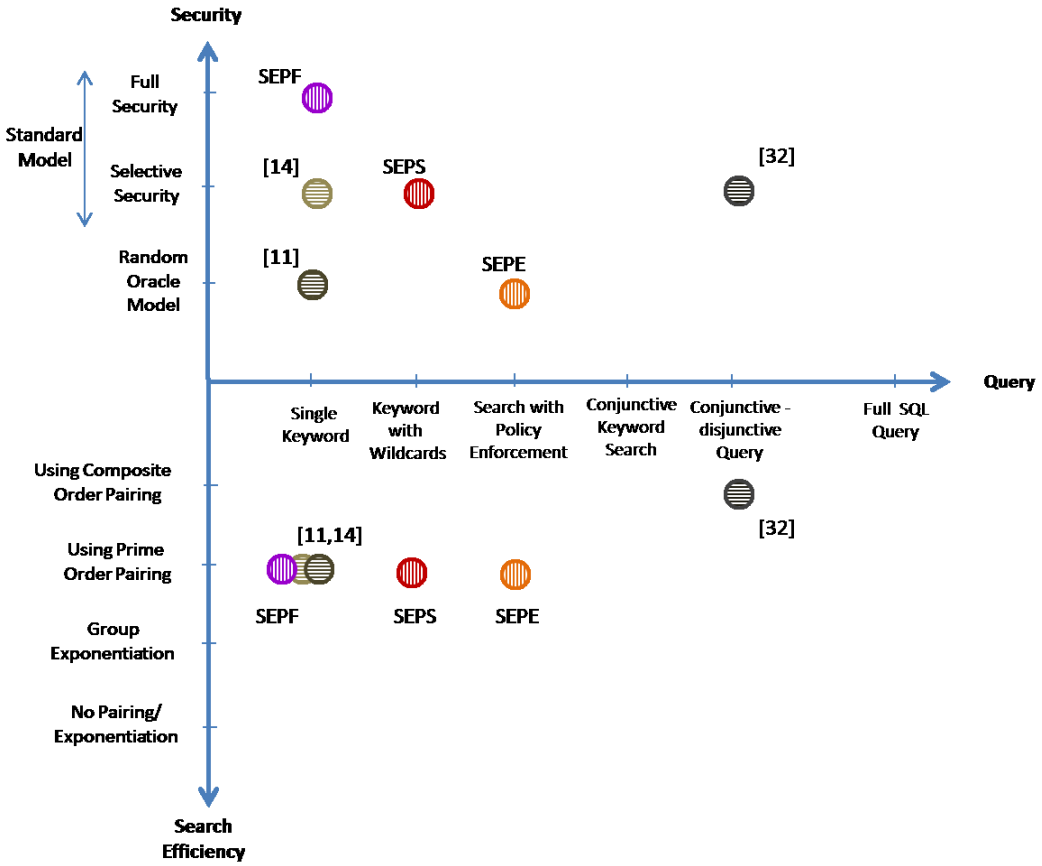

Figure 7.2: *Security and query functionality of our schemes in the public key setting.*

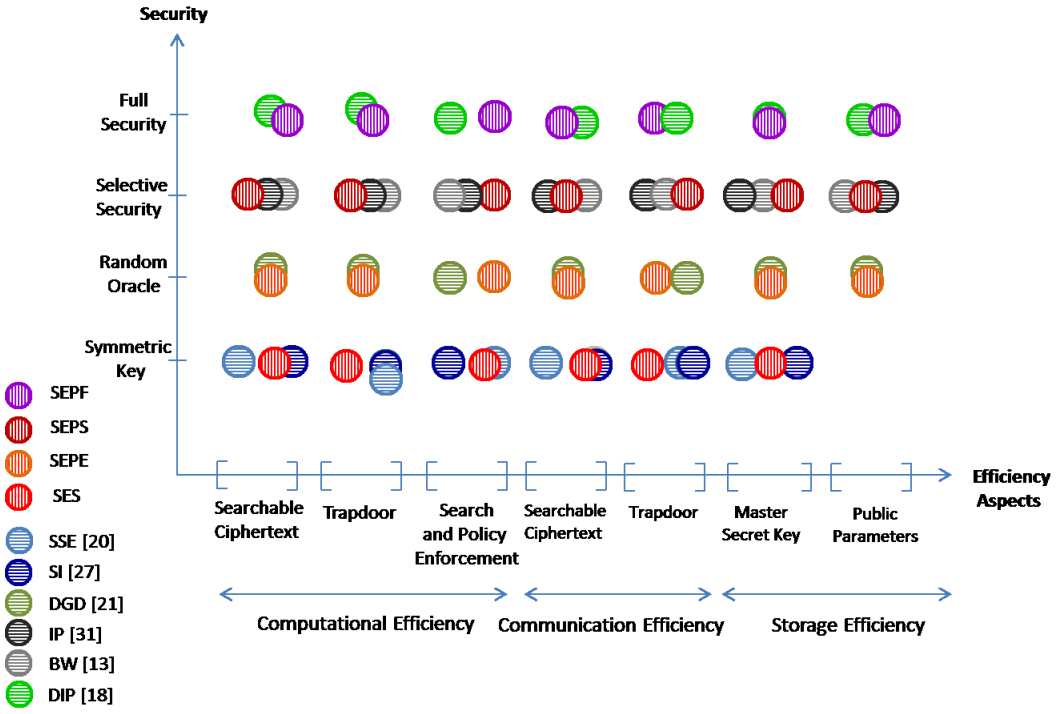

Figure 7.3: *Comparing the efficiency of our schemes with existing schemes.*

Author References

[1] S. Sedghi, H. Hartel, W. Jonker, and S. Nikova. Privacy enhanced access control by means of policy blinding. In *7th Information security Practice and Experience Conference (ISPEC 2011)*. Springer, 2011.

[2] S. Sedghi, P. van Liesdonk, S. Nikova, P. Hartel, and W. Jonker. Searching keywords with wildcards on encrypted data. In *7th Conference on Security and Cryptography for Networks (SCN)*, pages 138–153. Springer, 2010.

[3] P. Van Liesdonk, S. Sedghi, J. Doumen, P. Hartel, and W. Jonker. Computationally efficient searchable symmetric encryption. In *7th VLDB Workshop on Secure Data Management (SDM)*, pages 87–100. Springer, 2010.

Other References

[4] M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi. Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions. *Journal of Cryptology*, 21(3):350–391, 2008.

[5] M. Abdalla, E. Kiltz, and G. Neven. Generalized key delegation for hierarchical identity-based encryption. In *12th European Symposium On Research In Computer Security (ESORICS)*, pages 139–154. Springer, 2007.

[6] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *1st Conference on Computer and Communications Security (CCS)*, pages 62–73. ACM, 1993.

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

[8] C. Blundo, V. Iovino, and G. Persiano. Private-key hidden vector encryption with key confidentiality. In *8th Conference on Cryptology and Network Security (CANS)*, pages 259–277. Springer, 2009.

[9] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *24th Cryptology Coneference (CRYPTO)*, pages 41–55. Springer, 2004.

[10] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. In *23nd Conference on Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pages 506–522. Springer, 2004.

[11] D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. *SIAM Journal of Computing*, 32:586–615, March 2003.

[12] D. Boneh and B. Waters. Conjunctive, subset, and range queries on encrypted data. In *4th Theory of Cryptography Conference (TCC)*, pages 535–554, 2007.

[13] X. Boyen and B. Waters. Anonymous hierarchical identity-based encryption (without random oracles). In *26th Cryptology Conference (CRYPTO)*, pages 290–307. Springer, 2006.

[14] R. W. Bradshaw, J. E. Holt, and K. E. Seamons. Concealing complex policies with hidden credentials. In *12th Conference on Computer and Communications Security (CCS)*, pages 146–157. ACM, 2004.

[15] J. Camenisch, M. Kohlweiss, A. Rial, and C. Sheedy. Blind and anonymous identity-based encryption and authorised private searches on public key encrypted data. In *12th Conference on Practice and Theory in Public-Key Cryptography (PKC)*, pages 196–214, 2009.

[16] R. Canetti, S. Halevi, and J. Katz. A forward-secure public-key encryption scheme. In *22nd Conference on Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pages 255–271. Springer, 2003.

[17] A. De Caro, V. Iovino, and G. Persiano. Fully secure anonymous hibe and secret-key anonymous ibe with short ciphertexts. In *4th Conference on Pairing-Based Cryptography (Pairing)*, pages 347–366. Springer, 2010.

[18] Y. C. Chang and M. Mitzenmacher. Privacy preserving keyword searches on remote encrypted data. In *3rd Conference on Applied Cryptography and Network Security (ACNS)*, pages 442–455, 2005.

[19] R. Curtmola, J. A. Garay, S. Kamara, and R. Ostrovsky. Searchable symmetric encryption: Improved definitions and efficient constructions. In *14th Conference on Computer and Communications Security (CCS)*, pages 79–88. ACM, 2006.

[20] C. Dong, G. Russello, and N. Dulay. Shared and searchable encrypted data for untrusted servers. In *22nd Cnference on Data and Applications Security (DBSec)*, pages 127–143. Springer, 2008.

[21] C. Dong, G. Russello, and N. Dulay. Shared and searchable encrypted data for untrusted servers. In *22nd Workshop on Data and Application Security (DBSec)*, pages 127–143. Springer, 2008.

[22] D. M. Freeman. Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In *29th Conference on Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, volume 6110, page 45, 2010.

[23] K. Frikken, M. Atallah, and J. Li. Attribute-based access control with hidden policies and hidden credentials. *IEEE Transaction on Computer*, 55:1259–1270, 2006.

[24] C. Gentry. Practical identity-based encryption without random oracles. In *25th Conference on Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pages 445–464. Springer, 2006.

[25] C. Gentry and A. Silverberg. Hierarchical id-based cryptography. In *8th Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology (ASIACRYPT)*, pages 548–566. Springer, 2002.

[26] E. J. Goh. Secure indexes. In *Cryptology ePrint Archive, Report 2003/216*, 2004.

[27] P. Golle, Staddon J, and B. R. Waters. Secure conjunctive keyword search over encrypted data. In *2nd Conference on Applied Cryptography and Network Security (ACNS)*, pages 31–45, 2004.

[28] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *14th Conference on Computer and Communications Security (CCS)*, pages 89–98. ACM, 2006.

[29] Y. Hwang and P. Lee. Public key encryption with conjunctive keyword search and its extension to a multi-user system. In *1st Conference on Pairing-Based Cryptography (Pairing)*, pages 2–22. Springer, 2007.

[30] V. Iovino and G. Persiano. Hidden-vector encryption with groups of prime order. In *2nd Conference on Pairing-Based Cryptography (Pairing)*, pages 75–88. Springer, 2008.

[31] J. Katz, A. Sahai, and B. Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *28th Conference on Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pages 146–162. Springer, 2008.

[32] B. Waters L. Allison. New techniques for dual system encryption and fully secure hibe with short ciphertexts. In *7th Theory of Cryptography Conference (TCC)*, pages 445 – 479. Springer, 2010.

[33] A. B. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *29th Conference on Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pages 62–91, 2010.

[34] J. Li and N. Li. Oacerts: Oblivious attribute certificates. In *3rd Conference on Applied Cryptography and Network Security (ACNS)*, pages 301–317. Springer, 2005.

[35] T. Nishide, K. Yoneyama, and K. Ohta. Attribute-based encryption with partially hidden encryptor-specified access structures. In *6th Conference on Applied Cryptography and Network Security (ACNS)*, pages 111–129. Springer, 2008.

[36] T. Okamoto and K. Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In *30th Cryptology Conference (CRYPTO)*, pages 191–208, Berlin, Heidelberg, 2010. Springer-Verlag.

[37] A. Sahai and B. Waters. Fuzzy identity-based encryption. In *25th Conference on Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pages 457–473. Springer, 2005.

[38] E. Shen, E. Shi, and B. Waters. Predicate privacy in encryption systems. In *6th Conference on Theory of Cryptography (TCC)*, pages 457–473. Springer, 2009.

[39] E. Shi and B. Waters. Delegating capabilities in predicate encryption systems. In *35th Conference on Automata, Languages and Programming (ICALP)*, pages 560–578. Springer, 2008.

[40] D. X. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In *21st IEEE Symposium on Security and Privacy*, page 44. IEEE Computer Society, 2000.

[41] B. Waters. Efficient identity-based encryption without random oracles. In *24th Conference on Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pages 114–127. Springer, 2005.

[42] B. Waters. Dual system encryption: Realizing fully secure ibe and hibe under simple assumptions. In *29th Cryptology Conference (CRYPTO)*, pages 619–636. Springer, 2009.

[43] Y. Yang, F. Bao, X. Ding, and R. Deng. Multiuser private queries over encrypted databases. *International Journal of Applied Cryptology*, 1(4):309–319, 2009.

Titles in the IPA Dissertation Series since 2005

E. Dolstra. *The Purely Functional Software Deployment Model*. Faculty of Science, UU. 2006-01

R.J. Corin. *Analysis Models for Security Protocols*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2006-02

P.R.A. Verbaan. *The Computational Complexity of Evolving Systems*. Faculty of Science, UU. 2006-03

K.L. Man and R.R.H. Schiffelers. *Formal Specification and Analysis of Hybrid Systems*. Faculty of Mathematics and Computer Science and Faculty of Mechanical Engineering, TU/e. 2006-04

M. Kyas. *Verifying OCL Specifications of UML Models: Tool Support and Compositionality*. Faculty of Mathematics and Natural Sciences, UL. 2006-05

M. Hendriks. *Model Checking Timed Automata - Techniques and Applications*. Faculty of Science, Mathematics and Computer Science, RU. 2006-06

J. Ketema. *Böhm-Like Trees for Rewriting*. Faculty of Sciences, VUA. 2006-07

C.-B. Breunesse. *On JML: topics in tool-assisted verification of JML programs*. Faculty of Science, Mathematics and Computer Science, RU. 2006-08

B. Markvoort. *Towards Hybrid Molecular Simulations*. Faculty of Biomedical Engineering, TU/e. 2006-09

S.G.R. Nijssen. *Mining Structured Data*. Faculty of Mathematics and Natural Sciences, UL. 2006-10

G. Russello. *Separation and Adaptation of Concerns in a Shared Data Space*. Faculty of Mathematics and Computer Science, TU/e. 2006-11

L. Cheung. *Reconciling Nondeterministic and Probabilistic Choices*. Faculty of Science, Mathematics and Computer Science, RU. 2006-12

B. Badban. *Verification techniques for Extensions of Equality Logic*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2006-13

A.J. Mooij. *Constructive formal methods and protocol standardization*. Faculty of Mathematics and Computer Science, TU/e. 2006-14

T. Krilavicius. *Hybrid Techniques for Hybrid Systems*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2006-15

M.E. Warnier. *Language Based Security for Java and JML*. Faculty of Science, Mathematics and Computer Science, RU. 2006-16

V. Sundramoorthy. *At Home In Service Discovery*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2006-17

B. Gebremichael. *Expressivity of Timed Automata Models*. Faculty of Science, Mathematics and Computer Science, RU. 2006-18

L.C.M. van Gool. *Formalising Interface Specifications*. Faculty of Mathematics and Computer Science, TU/e. 2006-19

C.J.F. Cremers. *Scyther - Semantics and Verification of Security Protocols*. Faculty of Mathematics and Computer Science, TU/e. 2006-20

J.V. Guillen Scholten. *Mobile Channels for Exogenous Coordination of Distributed Systems: Semantics, Implementation and Composition.* Faculty of Mathematics and Natural Sciences, UL. 2006-21

H.A. de Jong. *Flexible Heterogeneous Software Systems.* Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2007-01

N.K. Kavaldjiev. *A run-time reconfigurable Network-on-Chip for streaming DSP applications.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2007-02

M. van Veelen. *Considerations on Modeling for Early Detection of Abnormalities in Locally Autonomous Distributed Systems.* Faculty of Mathematics and Computing Sciences, RUG. 2007-03

T.D. Vu. *Semantics and Applications of Process and Program Algebra.* Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2007-04

L. Brandán Briones. *Theories for Model-based Testing: Real-time and Coverage.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2007-05

I. Loeb. *Natural Deduction: Sharing by Presentation.* Faculty of Science, Mathematics and Computer Science, RU. 2007-06

M.W.A. Streppel. *Multifunctional Geometric Data Structures.* Faculty of Mathematics and Computer Science, TU/e. 2007-07

N. Trčka. *Silent Steps in Transition Systems and Markov Chains.* Faculty of Mathematics and Computer Science, TU/e. 2007-08

R. Brinkman. *Searching in encrypted data.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2007-09

A. van Weelden. *Putting types to good use.* Faculty of Science, Mathematics and Computer Science, RU. 2007-10

J.A.R. Noppen. *Imperfect Information in Software Development Processes.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2007-11

R. Boumen. *Integration and Test plans for Complex Manufacturing Systems.* Faculty of Mechanical Engineering, TU/e. 2007-12

A.J. Wijs. *What to do Next?: Analysing and Optimising System Behaviour in Time.* Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2007-13

C.F.J. Lange. *Assessing and Improving the Quality of Modeling: A Series of Empirical Studies about the UML.* Faculty of Mathematics and Computer Science, TU/e. 2007-14

T. van der Storm. *Component-based Configuration, Integration and Delivery.* Faculty of Natural Sciences, Mathematics, and Computer Science,UvA. 2007-15

B.S. Graaf. *Model-Driven Evolution of Software Architectures.* Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2007-16

A.H.J. Mathijssen. *Logical Calculi for Reasoning with Binding.* Faculty of Mathematics and Computer Science, TU/e. 2007-17

D. Jarnikov. *QoS framework for Video Streaming in Home Networks.* Faculty

of Mathematics and Computer Science, TU/e. 2007-18

M. A. Abam. *New Data Structures and Algorithms for Mobile Data*. Faculty of Mathematics and Computer Science, TU/e. 2007-19

W. Pieters. *La Volonté Machinale: Understanding the Electronic Voting Controversy*. Faculty of Science, Mathematics and Computer Science, RU. 2008-01

A.L. de Groot. *Practical Automaton Proofs in PVS*. Faculty of Science, Mathematics and Computer Science, RU. 2008-02

M. Bruntink. *Renovation of Idiomatic Crosscutting Concerns in Embedded Systems*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2008-03

A.M. Marin. *An Integrated System to Manage Crosscutting Concerns in Source Code*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2008-04

N.C.W.M. Braspenning. *Model-based Integration and Testing of High-tech Multi-disciplinary Systems*. Faculty of Mechanical Engineering, TU/e. 2008-05

M. Bravenboer. *Exercises in Free Syntax: Syntax Definition, Parsing, and Assimilation of Language Conglomerates*. Faculty of Science, UU. 2008-06

M. Torabi Dashti. *Keeping Fairness Alive: Design and Formal Verification of Optimistic Fair Exchange Protocols*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2008-07

I.S.M. de Jong. *Integration and Test Strategies for Complex Manufacturing Machines*. Faculty of Mechanical Engineering, TU/e. 2008-08

I. Hasuo. *Tracing Anonymity with Coalgebras*. Faculty of Science, Mathematics and Computer Science, RU. 2008-09

L.G.W.A. Cleophas. *Tree Algorithms: Two Taxonomies and a Toolkit*. Faculty of Mathematics and Computer Science, TU/e. 2008-10

I.S. Zapreev. *Model Checking Markov Chains: Techniques and Tools*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-11

M. Farshi. *A Theoretical and Experimental Study of Geometric Networks*. Faculty of Mathematics and Computer Science, TU/e. 2008-12

G. Gulesir. *Evolvable Behavior Specifications Using Context-Sensitive Wildcards*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-13

F.D. Garcia. *Formal and Computational Cryptography: Protocols, Hashes and Commitments*. Faculty of Science, Mathematics and Computer Science, RU. 2008-14

P. E. A. Dürr. *Resource-based Verification for Robust Composition of Aspects*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-15

E.M. Bortnik. *Formal Methods in Support of SMC Design*. Faculty of Mechanical Engineering, TU/e. 2008-16

R.H. Mak. *Design and Performance Analysis of Data-Independent*

Stream Processing Systems. Faculty of Mathematics and Computer Science, TU/e. 2008-17

M. van der Horst. *Scalable Block Processing Algorithms.* Faculty of Mathematics and Computer Science, TU/e. 2008-18

C.M. Gray. *Algorithms for Fat Objects: Decompositions and Applications.* Faculty of Mathematics and Computer Science, TU/e. 2008-19

J.R. Calamé. *Testing Reactive Systems with Data - Enumerative Methods and Constraint Solving.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-20

E. Mumford. *Drawing Graphs for Cartographic Applications.* Faculty of Mathematics and Computer Science, TU/e. 2008-21

E.H. de Graaf. *Mining Semi-structured Data, Theoretical and Experimental Aspects of Pattern Evaluation.* Faculty of Mathematics and Natural Sciences, UL. 2008-22

R. Brijder. *Models of Natural Computation: Gene Assembly and Membrane Systems.* Faculty of Mathematics and Natural Sciences, UL. 2008-23

A. Koprowski. *Termination of Rewriting and Its Certification.* Faculty of Mathematics and Computer Science, TU/e. 2008-24

U. Khadim. *Process Algebras for Hybrid Systems: Comparison and Development.* Faculty of Mathematics and Computer Science, TU/e. 2008-25

J. Markovski. *Real and Stochastic Time in Process Algebras for Performance Evaluation.* Faculty of Mathematics and Computer Science, TU/e. 2008-26

H. Kastenberg. *Graph-Based Software Specification and Verification.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-27

I.R. Buhan. *Cryptographic Keys from Noisy Data Theory and Applications.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-28

R.S. Marin-Perianu. *Wireless Sensor Networks in Motion: Clustering Algorithms for Service Discovery and Provisioning.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-29

M.H.G. Verhoef. *Modeling and Validating Distributed Embedded Real-Time Control Systems.* Faculty of Science, Mathematics and Computer Science, RU. 2009-01

M. de Mol. *Reasoning about Functional Programs: Sparkle, a proof assistant for Clean.* Faculty of Science, Mathematics and Computer Science, RU. 2009-02

M. Lormans. *Managing Requirements Evolution.* Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2009-03

M.P.W.J. van Osch. *Automated Model-based Testing of Hybrid Systems.* Faculty of Mathematics and Computer Science, TU/e. 2009-04

H. Sozer. *Architecting Fault-Tolerant Software Systems.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2009-05

M.J. van Weerdenburg. *Efficient Rewriting Techniques.* Faculty

of Mathematics and Computer Science, TU/e. 2009-06

H.H. Hansen. *Coalgebraic Modelling: Applications in Automata Theory and Modal Logic*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2009-07

A. Mesbah. *Analysis and Testing of Ajax-based Single-page Web Applications*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2009-08

A.L. Rodriguez Yakushev. *Towards Getting Generic Programming Ready for Prime Time*. Faculty of Science, UU. 2009-9

K.R. Olmos Joffré. *Strategies for Context Sensitive Program Transformation*. Faculty of Science, UU. 2009-10

J.A.G.M. van den Berg. *Reasoning about Java programs in PVS using JML*. Faculty of Science, Mathematics and Computer Science, RU. 2009-11

M.G. Khatib. *MEMS-Based Storage Devices. Integration in Energy-Constrained Mobile Systems*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2009-12

S.G.M. Cornelissen. *Evaluating Dynamic Analysis Techniques for Program Comprehension*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2009-13

D. Bolzoni. *Revisiting Anomaly-based Network Intrusion Detection Systems*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2009-14

H.L. Jonker. *Security Matters: Privacy in Voting and Fairness in Digital Exchange*. Faculty of Mathematics and Computer Science, TU/e. 2009-15

M.R. Czenko. *TuLiP - Reshaping Trust Management*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2009-16

T. Chen. *Clocks, Dice and Processes*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2009-17

C. Kaliszyk. *Correctness and Availability: Building Computer Algebra on top of Proof Assistants and making Proof Assistants available over the Web*. Faculty of Science, Mathematics and Computer Science, RU. 2009-18

R.S.S. O'Connor. *Incompleteness & Completeness: Formalizing Logic and Analysis in Type Theory*. Faculty of Science, Mathematics and Computer Science, RU. 2009-19

B. Ploeger. *Improved Verification Methods for Concurrent Systems*. Faculty of Mathematics and Computer Science, TU/e. 2009-20

T. Han. *Diagnosis, Synthesis and Analysis of Probabilistic Models*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2009-21

R. Li. *Mixed-Integer Evolution Strategies for Parameter Optimization and Their Applications to Medical Image Analysis*. Faculty of Mathematics and Natural Sciences, UL. 2009-22

J.H.P. Kwisthout. *The Computational Complexity of Probabilistic Networks*. Faculty of Science, UU. 2009-23

T.K. Cocx. *Algorithmic Tools for Data-Oriented Law Enforcement*. Faculty of Mathematics and Natural Sciences, UL. 2009-24

A.I. Baars. *Embedded Compilers*. Faculty of Science, UU. 2009-25

M.A.C. Dekker. *Flexible Access Control for Dynamic Collaborative Environments*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2009-26

J.F.J. Laros. *Metrics and Visualisation for Crime Analysis and Genomics*. Faculty of Mathematics and Natural Sciences, UL. 2009-27

C.J. Boogerd. *Focusing Automatic Code Inspections*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2010-01

M.R. Neuhäußer. *Model Checking Nondeterministic and Randomly Timed Systems*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2010-02

J. Endrullis. *Termination and Productivity*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2010-03

T. Staijen. *Graph-Based Specification and Verification for Aspect-Oriented Languages*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2010-04

Y. Wang. *Epistemic Modelling and Protocol Dynamics*. Faculty of Science, UvA. 2010-05

J.K. Berendsen. *Abstraction, Prices and Probability in Model Checking Timed Automata*. Faculty of Science, Mathematics and Computer Science, RU. 2010-06

A. Nugroho. *The Effects of UML Modeling on the Quality of Software*. Faculty of Mathematics and Natural Sciences, UL. 2010-07

A. Silva. *Kleene Coalgebra*. Faculty of Science, Mathematics and Computer Science, RU. 2010-08

J.S. de Bruin. *Service-Oriented Discovery of Knowledge - Foundations, Implementations and Applications*. Faculty of Mathematics and Natural Sciences, UL. 2010-09

D. Costa. *Formal Models for Component Connectors*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2010-10

M.M. Jaghoori. *Time at Your Service: Schedulability Analysis of Real-Time and Distributed Services*. Faculty of Mathematics and Natural Sciences, UL. 2010-11

R. Bakhshi. *Gossiping Models: Formal Analysis of Epidemic Protocols*. Faculty of Sciences, Department of Computer Science, VUA. 2011-01

B.J. Arnoldus. *An Illumination of the Template Enigma: Software Code Generation with Templates*. Faculty of Mathematics and Computer Science, TU/e. 2011-02

E. Zambon. *Towards Optimal IT Availability Planning: Methods and Tools*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2011-03

L. Astefanoaei. *An Executable Theory of Multi-Agent Systems Refinement*. Faculty of Mathematics and Natural Sciences, UL. 2011-04

J. Proença. *Synchronous coordination of distributed components*. Faculty of Mathematics and Natural Sciences, UL. 2011-05

A. Moralı. *IT Architecture-Based Confidentiality Risk Assessment in Networks of Organizations*. Faculty of

Electrical Engineering, Mathematics & Computer Science, UT. 2011-06

M. van der Bijl. *On changing models in Model-Based Testing*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2011-07

C. Krause. *Reconfigurable Component Connectors*. Faculty of Mathematics and Natural Sciences, UL. 2011-08

M.E. Andrés. *Quantitative Analysis of Information Leakage in Probabilistic and Nondeterministic Systems*. Faculty of Science, Mathematics and Computer Science, RU. 2011-09

M. Atif. *Formal Modeling and Verification of Distributed Failure Detectors*. Faculty of Mathematics and Computer Science, TU/e. 2011-10

P.J.A. van Tilburg. *From Computability to Executability – A process-theoretic view on automata theory*. Faculty of Mathematics and Computer Science, TU/e. 2011-11

Z. Protic. *Configuration management for models: Generic methods for model comparison and model co-evolution*. Faculty of Mathematics and Computer Science, TU/e. 2011-12

S. Georgievska. *Probability and Hiding in Concurrent Processes*. Faculty of Mathematics and Computer Science, TU/e. 2011-13

S. Malakuti. *Event Composition Model: Achieving Naturalness in Runtime Enforcement*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2011-14

M. Raffelsieper. *Cell Libraries and Verification*. Faculty of Mathematics and Computer Science, TU/e. 2011-15

C.P. Tsirogiannis. *Analysis of Flow and Visibility on Triangulated Terrains*. Faculty of Mathematics and Computer Science, TU/e. 2011-16

Y.-J. Moon. *Stochastic Models for Quality of Service of Component Connectors*. Faculty of Mathematics and Natural Sciences, UL. 2011-17

R. Middelkoop. *Capturing and Exploiting Abstract Views of States in OO Verification*. Faculty of Mathematics and Computer Science, TU/e. 2011-18

M.F. van Amstel. *Assessing and Improving the Quality of Model Transformations*. Faculty of Mathematics and Computer Science, TU/e. 2011-19

A.N. Tamalet. *Towards Correct Programs in Practice*. Faculty of Science, Mathematics and Computer Science, RU. 2011-20

H.J.S. Basten. *Ambiguity Detection for Programming Language Grammars*. Faculty of Science, UvA. 2011-21

M. Izadi. *Model Checking of Component Connectors*. Faculty of Mathematics and Natural Sciences, UL. 2011-22

L.C.L. Kats. *Building Blocks for Language Workbenches*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2011-23

S. Kemper. *Modelling and Analysis of Real-Time Coordination Patterns*. Faculty of Mathematics and Natural Sciences, UL. 2011-24

J. Wang. *Spiking Neural P Systems*. Faculty of Mathematics and Natural Sciences, UL. 2011-25

A. Khosravi. *Optimal Geometric Data Structures*. Faculty of Mathematics and Computer Science, TU/e. 2012-01

A. Middelkoop. *Inference of Program Properties with Attribute Grammars, Revisited*. Faculty of Science, UU. 2012-02

Z. Hemel. *Methods and Techniques for the Design and Implementation of Domain-Specific Languages*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2012-03

T. Dimkov. *Alignment of Organizational Security Policies: Theory and Practice*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2012-04

S. Sedghi. *Towards Provably Secure Efficiently Searchable Encryption*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2012-05