CRYPTOGRAPHICALLY ENFORCED SEARCH PATTERN HIDING

CHRISTOPH BÖSCH

GRADUATION COMMITTEE

Chairman and Secretary:

prof.dr.	P.M.G. Apers	University of Twente, The Netherlands

Promoters:

prof.dr.	P.H. Hartel	University of Twente, The Netherlands
prof.dr.	W. Jonker	University of Twente, The Netherlands

Members:

prof.dr.	P.M.G. Apers	University of Twente, The Netherlands
prof.dr.	R.N.J. Veldhuis	University of Twente, The Netherlands
prof.dr.	M. Petković	Eindhoven University of Technology, NL
dr.	H. Wang	Nanyang Technological University, Singapore
prof.dr.	J. Pieprzyk	Macquarie University, Sydney, Australia

CTIT Ph.D. Thesis Series No. 14-340
Centre for Telematics and Information Technology
P.O. Box 217, 7500 AE Enschede, The Netherlands

SIKS Dissertation Series No. 2015-05
The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.

ISBN: 978-90-365-3817-6
ISSN: 1381-3617 (CTIT Ph.D. Thesis Series No. 14-340)
DOI: 10.3990/1.9789036538176

Typeset with LaTeX.
Printed by Ipskamp Drukkers.
Cover design by Malte Hammerbeck.

CRYPTOGRAPHICALLY ENFORCED SEARCH PATTERN HIDING

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 Wednesday, 21 January 2015 at 16.45

by

CHRISTOPH TOBIAS BÖSCH

born on 23 September 1979
in Paderborn, Germany.

This dissertation is approved by:

prof.dr. P.H. Hartel
prof.dr. W. Jonker

Drink water!

This page intentionally left blank.

ABSTRACT

Searchable encryption is a cryptographic primitive that allows a client to outsource encrypted data to an untrusted storage provider, while still being able to query the data without decrypting. To allow the server to perform the search on the encrypted data, a so-called trapdoor is generated by the client and sent to the server. With help of the trapdoor, the server is able to perform the search, on behalf of the client, on the still encrypted data.

All reasonably efficient searchable encryption schemes have a common problem. They leak the search pattern which reveals whether two searches were performed for the same keyword or not. Hence, the search pattern gives information on the occurrence frequency of each query, which can be exploited by statistical analysis, eventually allowing an attacker to gain full knowledge about the underlying plaintext keywords. Thus, attacking the search pattern is a serious problem that renders the encryption less useful.

The goal of this thesis is to construct novel searchable encryption schemes that are efficient and that do not leak the search pattern to mitigate the above attack. In addition, we show the practical applicability of our proposed solutions in real world scenarios by implementing the main building blocks of our constructions in C. Our contributions can be summarized as follows:

- We survey the notion of provably secure searchable encryption by giving a complete and comprehensive overview of the two main SE techniques: Searchable Symmetric Encryption and Public Key Encryption with Keyword Search.
- We propose two constructions that hide the search pattern with reasonable efficiency in practical application scenarios. One scheme is entirely based on efficient XOR and pseudo-random functions, while the other scheme makes use of recent advances in somewhat homomorphic encryption to achieve efficient solutions. To hide the search pattern, we use two different approaches. The first approach processes the whole encrypted database on the server side by calculating the inner product of a query and the database records. In this way, we conceal which of the database records are important per query. The second approach introduces a third party to help with the search. The idea is that the database server randomly shuffles the positions of the database entries, so that the third party performs the actual search on a newly shuffled index per query. In this way, the positions of the processed database entries are different for each (distinct) query.
- We propose a third scheme that illustrates how to use the techniques from our previous schemes, to construct a novel and efficient search scheme for a concrete application scenario. The scheme can be used to perform private/hidden queries on different kinds of unencrypted data, such as RSS feeds.

This page intentionally left blank.

SAMENVATTING

Doorzoekbare encryptie is een cryptografische primitieve die een gebruiker in staat stelt om versleutelde gegevens bij een niet-vertrouwde storage provider op te slaan, terwijl de gebruiker nog steeds in staat is om deze gegevens te doorzoeken zonder deze eerst te decoderen. Om de server in staat te stellen om de zoekopdracht uit te voeren op de versleutelde gegevens, wordt een zogenaamde trapdoor gegenereerd door de gebruiker en naar de server gestuurd. Met behulp van de trapdoor is de server in staat om de zoekopdracht uit te voeren, namens de gebruiker, op de nog versleutelde gegevens.

Alle redelijk efficiënt doorzoekbare encryptie schema's hebben een gemeenschappelijk probleem. Ze lekken het zoekpatroon waaruit blijkt of twee zoekopdrachten werden uitgevoerd voor hetzelfde zoekwoord of niet. Daarom geeft het zoekpatroon informatie over de frequentie van elke zoekwoord. Die informatie kan worden uitgebuit door statistische analyse, waardoor uiteindelijk een aanvaller volledige kennis over de onderliggende gegevens kan krijgen. Het op deze manier aanvallen van het zoekpatroon is een ernstig probleem dat de versleuteling minder bruikbaar maakt.

Het doel van dit proefschrift is om nieuwe schema's voor doerzoekbare encryptie te bouwen die efficient zijn en het zoekpatroon niet lekken om bovengenoemde aanval tegen te gaan. Verder laten we de praktische toepasbaarheid van onze voorgestelde oplossingen zien in realistische scenario's door de belangrijke onderdelen van onze constructies in C te implementeren. Onze bijdragen kunnen als volgt samengevat worden:

- We verkennen de notie van bewijsbare veilige doorzoekbare encryptie door een compleet en begrijpbaar overzicht te geven van de twee belangrijkste doorzoekbare encryptie technieken: doorzoekbare symmetrische encryptie en publieke sleutel encryptie met zoekwoorden.
- We stellen twee constructies voor die het zoekpatroon verbergen met redelijke efficientie in praktische scenario's. Één schema is compleet gebaseerd op efficiente XOR operaties en pseudo-random functies, twerijl het andere schema gebruik maakt van recente doorbraken op het gebied van homomorfe encryptie om efficientie te bereiken. Om het zoekpatroon te verbergen gebruiken we twee verschillende methoden. De eerste methode gebruikt de gehele versleutelde database van de server door de inner product van een zoekopdracht en de database records te berekenen. Op deze manier verbergen we welke database records belangrijk zijn per zoekopdracht. De tweede methode introduceerd een derde partij om met de zoekopdracht te helpen. Het idee is dat de database server de posities in de database records op een gerandomizeerde manier schudt, zodat de derde partij de zoekopdracht op een vers geschudde database index doet. Op deze manier zijn de posities van de records in de database verschillend voor elke (andere) zoekopdracht.
- We stellen een derde schema voor dat illustreerd hoe de technieken van de vorige schema's te gebruiken zijn om een nieuw en efficient zoek schema te bouwen voor concrete applicatie scenario's. Het schema kan gebruikt worden om verborgen zoekopdrachten op verschillende typen van onversleutelde gegevens te doen, zoals bijvoorbeeld RSS feeds.

This page intentionally left blank.

ACKNOWLEDGEMENTS

> .... very little do we have and inclose which we can call our own in the deep sense of the word. We all have to accept and learn, either from our predecessors or from our contemporaries. Even the greatest genius would not have achieved much if he had wished to extract everything from inside himself. But there are many good people, who do not understand this, and spend half their lives wondering in darkness with their dreams of originality. I have known artists who were proud of not having followed any teacher and of owing everything only to their own genius. Such fools!
>
> —Goethe, *Conversations with Eckermann*, 17.02.1832

Numerous people over the last years have helped me to reach the finish line, so there are many people I need to thank for their help, support, and encouragement on finishing this thesis. Thanks to my promoters Pieter and Willem. You taught me how to do research and how to look at things from an academic, as well as an industrial point of view. Thanks for your guidance and the many discussions throughout this project.

I had several daily supervisors. In the beginning, there were Richard and Qiang. Thanks for helping me in the early stage of my PhD. After some time without a supervisor, Andreas took over and did a great job. Thanks for your advice and plenty of fruitful brainstorm sessions. During my PhD I spend several month in Singapore. Thanks to Hoon Wei and Huaxiong for their help during my stay at NTU.

Thanks to all the members of the committee for accepting to be part of the committee and for taking their time to read my thesis.

Of course, I thank all my colleagues from the UT. The secretaries Ida, Nienke, Bertine, and Suse were always helpful. Thanks for your support with all the bureaucracy and your help even in private affairs. Special thanks to Dina, Arjan, and Stefan for sharing most of the PhD time together and for the many evenings, BBQs, road trips, and more. Thanks to Ayse, Trajce, Luan, Saeed, André, Elmer, Eleftheria, Marco, Michael, Jan-Willem, Begül, Jonathan, Wolter, Maarten, Lorena, Eelco, and Steven for the good atmosphere at work, the many coffee breaks and other distractions. Also thanks to the crew from Security Matters, especially Damiano, Emmanuele, Michele, Chris, and Spase.

Thanks to Johanna, Lukas, Rob, Sara, Clara, and Andy for reminding me that there is a life beyond the university and research. Also thanks to the crew from Switzerland, especially Mario and Stefan, for all the distractions. Thanks to Malte for the cover design and André Miede for the classic thesis template.

Special thanks to my parents for giving me life and all the support they have provided me over the years. Thank you for making me who I am. A big HELLO to Marcus, Marie, and Malina. Super special thanks with sugar on top to milady Sylvie, for all the support and freedom one could dream of. It requires a book on its own to express my gratitude for all your help and the many enjoyable moments. Thank you for accompanying me on this journey. I look forward to our next one!

Last and certainly least, I want to mention the crew from Barrio PBC as outstanding non-contributors.

—CTB, 24.12.2014

This page intentionally left blank.

CONTENTS

1 INTRODUCTION 1
- 1.1 Research Question 2
- 1.2 Thesis Outline and Contribution 4

2 SEARCHABLE ENCRYPTION: STATE OF THE ART 7
- 2.1 Motivation and Introduction 7
- 2.2 Preliminaries 11
- 2.3 Single Writer Schemes (S/∗) 15
- 2.4 Multi Writer Schemes (M/∗) 35
- 2.5 Related Work 52
- 2.6 Conclusions and Future Work 55

3 SELECTIVE DOCUMENT RETRIEVAL 61
- 3.1 Introduction 61
- 3.2 Selective Document Retrieval 63
- 3.3 The Proposed SDR Scheme 67
- 3.4 Security Analysis 69
- 3.5 Adaptations of the Proposed SDR Scheme 70
- 3.6 Performance Analysis 73
- 3.7 Conclusion 78

4 DISTRIBUTED SEARCHABLE SYMMETRIC ENCRYPTION 79
- 4.1 Introduction 79
- 4.2 Distributed Searchable Symmetric Encryption 80
- 4.3 The Proposed Distributed Construction 83
- 4.4 Security Analysis 86
- 4.5 Performance Analysis 87
- 4.6 Colluding servers 88
- 4.7 Conclusion 93

5 SECURELY OUTSOURCED FORENSIC IMAGE RECOGNITION 95
- 5.1 Introduction 95
- 5.2 Preliminaries 96
- 5.3 The Proposed SOFIR Construction 97
- 5.4 Security Discussion 98
- 5.5 Performance Analysis 99
- 5.6 Conclusion 102

6 CONCLUDING REMARKS 103
- 6.1 Contributions and Future Work 104

BIBLIOGRAPHY 107

This page intentionally left blank.

1 INTRODUCTION

Web search is the primary way in which we access information and data from everywhere at any time. This new way of information retrieval comes with privacy risks. Internet service providers and search engines for example store all search queries and link them to a specific/unique user. We divulge (private and sensitive) information, e. g., our medical problems and diseases, tax information, our (sexual) preferences, religious views, and political interests. Inter alia, the gathered data can be used to make decisions about us, such as our eligibility for insurance, credit, or employment. Criminals may also use this data, e. g., for identity theft which is a common problem in our society.

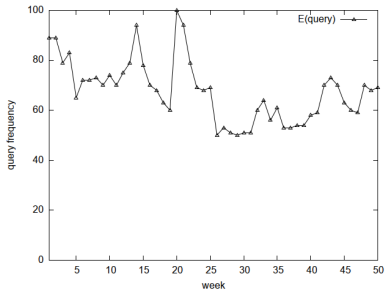
In addition, with the recent development of cloud computing, we outsource more and more (private and sensitive) data to third party storage providers. Storing data in the cloud is practical, since the centrally stored data is accessible from any Internet-capable device, from any location and at any time. But this freedom also has its pitfalls, since storage providers have full access to their servers and consequently to the plaintext data. Not to mention hackers with root access. To store sensitive data in a secure way on an untrusted server the data has to be encrypted. Using a standard encryption scheme, however, makes it impossible to query the encrypted data on the server side without decrypting.

Several lines of research have identified these two problems of private searching and secure data outsourcing, and have proposed solutions to query plaintext and even encrypted data in a privacy-preserving way, by using an encrypted query. The most prevalent technique is called searchable encryption.

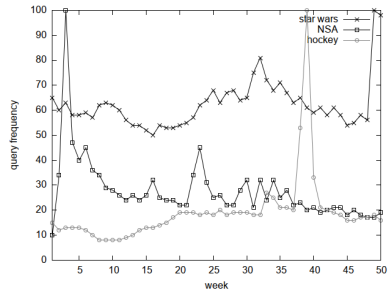
Searchable encryption is a cryptographic primitive that allows a client to outsource encrypted data to an untrusted storage provider (such as a cloud provider) while still being able to query the encrypted data on the server side without decrypting. This can be achieved by either encrypting the data in a special way or by introducing a searchable encrypted index, which is stored together with the encrypted data on the server. To allow the server to query the encrypted data, a so-called token or trapdoor is generated and sent to the server. With help of the trapdoor, the server is able to perform a search on the still encrypted data.

Encryption provides confidentiality for the data and the query separately, but when combined (during the search phase), may leak (sensitive) information. E. g., the database records, which are ultimately represented as memory locations, touched by the server during a search, expose a pattern of the search. This search pattern reveals whether two searches were performed for the same keyword or not. Hence, the search pattern gives information on the occurrence frequency of each query. This is a serious problem, as it allows an attacker to perform statistical analysis on the occurrence frequency, eventually allowing the attacker to gain knowledge about the underlying plaintext keywords.

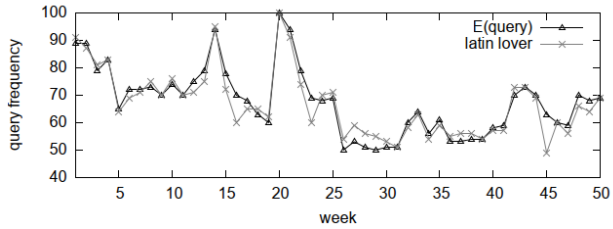
To exploit the search pattern, the attacker records the occurrence frequency of the target queries over a specific period of time, e. g., days or weeks. Figure 1.1a shows an example of an attacker's target query recorded over a timespan of 50 weeks. This graph is based on made-up query frequencies chosen by



(a) An attacker's encrypted target query frequency.



(b) Different query frequencies taken from Google Trends.



(c) The search result with the best match for the encrypted target query using Google Correlate is "latin lover".

Figure 1.1: Query frequencies from Google Trends [98]. The frequencies are normalized and were recorded by Google between 26.05.2013 – 10.05.2014. Accessed on 17.05.2014.

us at random. Afterwards, the attacker can correlate the collected dataset with some ancillary background information from public databases like Google Trends [98] and Google Correlate. Google Trends offers query statistics about all the search terms entered in Google's web search engine in the past. Three example query frequencies are shown in Figure 1.1b. Google Trends even offers statistics under various (sub-)categories (e. g., computer science, finance, medicine), allowing to adjust the attack to a user with specific background knowledge, rendering the attack more efficient. Using Google Correlate, a tool on Google Trends, enables an attacker to upload the query frequency of a target query (cf. Figure 1.1a) to obtain (plaintext) queries with similar patterns. Figure 1.1c shows the best correlation for our random target query: latin lover.

The search pattern or rather the occurrence frequency of queries allows to effectively attack the underlying plaintext keywords of someone's encrypted queries. This attack is also demonstrated by Liu et al. [128] on real-world data. As a result, the search pattern and with it the occurrence frequency of queries should be protected when querying encrypted and also unencrypted data.

1.1 RESEARCH QUESTION

Searchable encryption (SE) can be done in two fundamentally different ways: Searchable Symmetric Encryption (SSE) and Public Key Encryption with Keyword Search (PEKS). As we will discuss in Chapter 2 it is impossible to hide

the search pattern in PEKS schemes due to the use of public key encryption. For symmetric encryption, Shen, Shi, and Waters [167] (SSW) proposed the first search pattern hiding predicate encryption scheme, which is a generalization of SSE. So far, their construction is the only search pattern hiding scheme in the context of searchable encryption. Unfortunately, on commodity hardware SSW's solution is inefficient in practice due to the use of complex building blocks. For example, a query using their scheme, given a dataset of 5000 documents and 250 keywords, takes around 8.4 days to process, as discussed in Chapter 3. This is orders of magnitude away from practical efficiency. Motivated by these limitations and the importance of the topic, we pose the following first research question:

RQ1: How to construct *efficient* search pattern hiding searchable encryption schemes?

To understand, how to hide the search pattern, we must know, how it leaks. Using a deterministic trapdoor for example leaks the search pattern directly, since two queries for the same keyword will always generate the same trapdoor. As a result, the first step to hide the search pattern is to use probabilistic trapdoors. But using a probabilistic trapdoor is not enough, because usually queries for the same keyword touch or process the same database records. In this way, the server knows if two queries were performed for the same keyword or not. This means, that in addition to probabilistic trapdoors, a scheme needs to hide which of the database entries are processed during a search.

In this thesis we focus on the following two methods to hide the processed database entries from a server:

A1) The server needs to process *all* database entries per query. Thus, the server cannot tell, which of the database entries were of importance for the query and as a result the search pattern is hidden.

A2) The positions of the database entries need to be *different* per query, e. g., permuted. Thus, the server processes different database entries if queries are performed for the same keyword and the search pattern remains hidden.

The first approach (A1) is presented as SDR – a selective document retrieval scheme – in Chapter 3. To process all database entries, our SDR scheme calculates the inner product between the trapdoor and the database (in a private manner). In this way, the server touches all database records and does not know, where in the database a match occurs or which of the database records are of importance to a query. This hinders the server from determining the search pattern. For the second approach (A2), we propose DSSE – a distributed searchable symmetric encryption scheme – in Chapter 4, which randomly shuffles the database entries before each query, without the knowledge of the server. This makes the positions of the database entries probabilistic and ensures that two queries, even for the same keyword, access different (random) database entries.

The above approaches are used to create search pattern hiding schemes for encrypted data. But, as stated above, the problem of search pattern hiding also arises when dealing with plaintext data. It is equally important to protect the search pattern for queries on non-encrypted data, since most of the data in the world wide web is in the plain. This is quite different from a query on

encrypted data, because the searching party knows already half of the data, i. e., the plaintext data. Thus, we pose the following second research question:

RQ2: How to construct *efficient* search pattern hiding schemes for unencrypted data?

For this research question we present SOFIR – securely outsourced forensic image recognition – in Chapter 5. SOFIR uses some of the techniques from previous chapters to build a search pattern hiding query scheme for unencrypted data.

We aim for efficient schemes that can be used in practical application scenarios. By efficiency we mean the computational, communication, and space complexity of a scheme. We focus especially on the real running time of the search algorithms and aim for search processes that output results in a reasonable time, e. g., milliseconds to at most several minutes depending on the dataset size. Because the scheme by Shen, Shi, and Waters is the only search pattern hiding scheme so far, we compare the efficiency of our constructions with theirs. Therefore, we implement the main building blocks of all our schemes and SSW in C/C++ to perform a practical comparison of the running times for different datasets.

Primarily, a searchable encryption scheme should be secure to be used in practical applications. Since there is a wide spectrum of different searchable encryption schemes from different communities, we focus only on provable secure searchable encryption schemes in this thesis.

1.2 THESIS OUTLINE AND CONTRIBUTION

Figure 1.2 depicts the outline of this thesis. After this introduction, we start with the state of the art of provable secure searchable encryption. Then we present our own solutions to the problem in the form of our three schemes. Because the three scenarios and solutions are quite different, each chapter uses a slightly modified notation stated in the respective chapter. Finally, we conclude and give directions for further research. The thesis is organized into the following six chapters:

INTRODUCTION: The current chapter provides an introduction and the motivation for our research, as well as the main research question, the contribution and the overall structure of the thesis.

STATE OF THE ART: Chapter 2 surveys the notion of provably secure searchable encryption by giving a complete and comprehensive overview of the two main SE techniques: Searchable Symmetric Encryption and Public Key Encryption with Keyword Search. We present a framework to categorize SE schemes according to their functionality, efficiency, and security. In addition we shed light on the many definitions, notions, and assumptions used in the field. Our results show, that all SE schemes (except for the predicate encryption scheme by Shen, Shi, and Waters [167]) leak the search pattern. Thus, the motivation for our research question. This chapter is based on a refereed journal article [4] in ACM Computing Surveys 2014.

SDR – SELECTIVE DOCUMENT RETRIEVAL: In Chapter 3 we present our first search pattern hiding scheme. This construction hides the search pattern by computing the inner product of the trapdoor and the index,

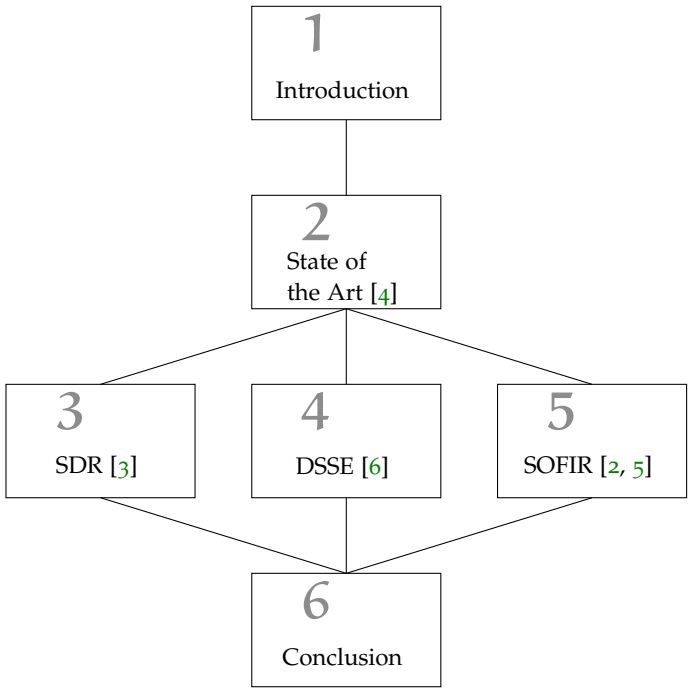


Figure 1.2: Outline of the thesis

thereby processing all database entries per query. In addition, to make the scheme more practical, we separate the search phase from the document retrieval. Therefore, the scheme works like a web search, where the search phase identifies possible results from which the client can decide whether to retrieve documents, and if, which documents. The scheme relies on client interaction in the sense, that the document retrieval takes two rounds of communication. This chapter is based on a refereed conference paper [3] in ISC 2012.

DSSE – DISTRIBUTED SEARCHABLE SYMMETRIC ENCRYPTION: Chapter 4 presents our second construction of a provably secure SE scheme that hides the search pattern. The scheme introduces a third party, the so called query router, which performs the actual search. The scheme hides the search pattern by letting the storage provider randomly shuffle all database entries before each query without the knowledge of the query router. In this way, the query router receives a new index per query and thus, processes different (random) database records even if searching for the same keyword twice. Our results show, that a DSSE scheme can potentially provide more efficiency and better security guarantees than standard SSE. Even if the two untrusted parties collude, the scheme is still secure under Curtmola et al.'s definition for adaptive semantic security for SSE. This chapter is based on a refereed conference paper [6] in PST 2014.

SOFIR – SECURELY OUTSOURCED FORENSIC IMAGE RECOGNITION: In this scheme, presented in Chapter 5, we show how to use the techniques from previous chapters, e. g., somewhat homomorphic encryption, in a

concrete application scenario. The scheme can be used to perform private, i. e., hidden, queries on different kinds of unencrypted data, e. g., RSS feeds. The scheme protects the search pattern by hiding whether the processed database entries per query resulted in a match or not. This chapter is based on a patent application [2] and a refereed conference paper [5] in ICASSP 2014.

CONCLUSION: In Chapter 6 we provide conclusions and suggestions for further research. Compared with the state of the art, our solutions protect the search pattern and can potentially provide a higher level of security and efficiency.

In this thesis we show efficient solutions for the problem of search pattern hiding. We propose three novel search schemes. Two of the schemes are the answer to RQ1, each of which uses one of the approaches A1 and A2. The third scheme answers RQ2. All of the constructions come with their own advantages and drawbacks. We show with a concrete application scenario that our approaches are relevant in practice and work efficiently in the real world. Our three schemes are the first efficient search pattern hiding constructions so far.

Search pattern hiding is an important tool to increase our privacy when querying data, especially to protect personal and sensitive information. In case of encrypted data, the search pattern can be exploited to bypass the encryption and get (full) knowledge on the underlying plaintext data. With our solutions it is possible to securely outsource our data to untrusted parties. At the same time we can query the outsourced encrypted data and also other's unencrypted data in a private manner. We have shown, that practical efficient schemes can be constructed and can now focus on even more efficient and expressive constructions for different application scenarios.

2 SEARCHABLE ENCRYPTION: STATE OF THE ART

In this chapter, we survey the notion of provably secure Searchable Encryption (SE) by giving a complete and comprehensive overview of the two main SE techniques: Searchable Symmetric Encryption (SSE) and Public Key Encryption with Keyword Search (PEKS). Since the pioneering work of Song, Wagner and Perrig (SWP), the field of provably secure SE has expanded to the point where we felt that taking stock would provide benefit to the community.

The survey has been written primarily for the non-specialist who has a basic information security background. Thus, we sacrifice full details and proofs of individual constructions in favor of an overview of the underlying key techniques to give beginners a solid foundation for further research. We categorize and analyze the different provably secure SE schemes in terms of their architecture, security, efficiency, and functionality to provide an easy entry point for non-specialists and to allow researchers to keep up with the many approaches to SE. For an experienced researcher we point out connections between these approaches, identify open research problems, and specify various gaps in the field. Our extensive tables, which reflect our detailed analysis, allow practitioners to find suitable schemes for the many different application scenarios.

Two major conclusions can be drawn from our work. While the so-called IND-CKA2 security notion becomes prevalent in the literature and efficient (sub-linear) SE schemes meeting this notion exist in the symmetric setting, achieving this strong form of security efficiently in the asymmetric setting remains an open problem. We observe that in multi-recipient SE schemes, regardless of their efficiency drawbacks, there is a noticeable lack of query expressiveness which hinders deployment in practice.

2.1 MOTIVATION AND INTRODUCTION

We start with our motivation for writing this survey and introduce the main concepts and challenges of provably secure searchable encryption.

2.1.1 *Motivation*

The wide proliferation of sensitive data in open information and communication infrastructures all around us has fuelled research on secure data management and boosted its relevance. For example, legislation around the world stipulates that electronic health records (EHR) should be encrypted, which immediately raises the question how to search EHR efficiently and securely. After a decade of research in the field of provably secure searchable encryption we felt that the time has come to survey the field by putting the many individual contributions into a comprehensive framework. On the one hand, the framework allows practitioners to select appropriate techniques to address the security requirements of their applications. On the other hand, the framework points out uncharted areas of research since by no means all application requirements are covered by the techniques currently in existence. We hope

that researchers will find inspiration in the survey that is necessary to develop the field further.

2.1.2 *Introduction to Searchable Encryption*

Remote and cloud storage is ubiquitous and widely used for services such as backups or outsourcing data to reduce operational costs. However, these remote servers cannot be trusted, because administrators, or hackers with root rights, have full access to the server and consequently to the plaintext data. Or imagine that your trusted storage provider sells its business to a company that you do not trust, and which will have full access to your data. Thus, to store sensitive data in a secure way on an untrusted server the data has to be encrypted. This reduces security and privacy risks, by hiding all information about the plaintext data. Encryption makes it impossible for both insiders and outsiders to access the data without the keys, but at the same time removes all search capabilities from the data owner. One trivial solution to re-enable searching functionality is to download the whole database, decrypt it locally, and then search for the desired results in the plaintext data. For most applications this approach would be impractical. Another method lets the server decrypt the data, runs the query on the server side, and sends only the results back to the user. This allows the server to learn the plaintext data being queried and hence makes encryption less useful. Instead, it is desirable to support the fullest possible search functionality on the server side, without decrypting the data, and thus, with the smallest possible loss of data confidentiality. This is called *searchable encryption* (SE).

General Model. An SE scheme allows a server to search in encrypted data on behalf of a client without learning information about the plaintext data. Some schemes implement this via a ciphertext that allows searching (e. g., Song et al. [171] (SWP) as discussed in Section 2.3.1), while most other schemes let the client generate a searchable encrypted index. To create a searchable encrypted index I of a database $\mathcal{D} = (M_1, \ldots, M_n)$ consisting of n messages[1] M_i, some data items $\mathcal{W} = (w_1, \ldots, w_m)$, e. g., keywords w_j (which can later be used for queries), are extracted from the document(s) and encrypted (possibly non-decryptable, e. g., via a hash function) under a key K of the client using an algorithm called $\mathsf{BuildIndex}$. M_i may also refer to database records in a relational database, e. g., MySQL. In addition, the document may need to be encrypted with a key K' (often, $K' \neq K$) using an algorithm called Enc. The encrypted index and the encrypted documents can then be stored on a semi-trusted (honest-but-curious [93]) server that can be trusted to adhere to the storage and query protocols, but which tries to learn as much information as possible. As a result the server stores a database of the client in the following form:

$$I = \mathsf{BuildIndex}_K(\mathcal{D} = (M_1, \ldots, M_n), \mathcal{W} = (w_1, \ldots, w_m));$$
$$C = \mathsf{Enc}_{K'}(M_1, \ldots, M_n).$$

To search, the client generates a so-called trapdoor $T = \mathsf{Trapdoor}_K(f)$, where f is a predicate on w_j. With T, the server can search the index using an algorithm called Search and see whether the encrypted keywords satisfy the pred-

1 By messages we mean plaintext data like files, documents or records in a relational database.

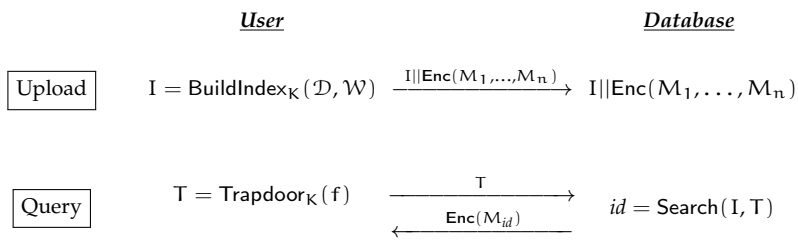


Figure 2.1: General model of an index-based searchable encryption scheme.

icate f, and return the corresponding (encrypted) documents (see Figure 2.1). For example, f could determine whether a specific keyword w is contained in the index [92], and a more sophisticated f could determine whether the inner product of keywords in the index and a target keyword set is o [167].

Figure 2.1 gives a general model of an index-based scheme. Small deviations are possible, e. g., some schemes do not require the entire keyword list $\mathcal{W}$ for building the index.

Single-user vs. Multi-user. SE schemes are built on the client/server model, where the server stores encrypted data on behalf of one or more clients (i. e., the writers). To request content from the server, one or more clients (i. e., readers) are able to generate trapdoors for the server, which then searches on behalf of the client. This results in the following four SE architectures:

- single writer/single reader (S/S)
- multi writer/single reader (M/S)
- single writer/multi reader (S/M)
- multi writer/multi reader (M/M)

Depending on the architecture, the SE scheme is suitable for either data outsourcing (S/S) or data sharing (M/S, S/M, M/M).

Symmetric vs. Asymmetric primitives. Symmetric key primitives allow a single user to read and write data (S/S). The first S/S scheme, proposed by Song et al. [171], uses symmetric key cryptography and allows only the secret key holder to create searchable ciphertexts and trapdoors. In a public key encryption (PKE) scheme, the private key decrypts all messages encrypted under the corresponding public key. Thus, PKE allows multi-user writing, but only the private key holder can perform searches. This requires an M/S architecture. The first M/S scheme is due to Boneh et al. [47] who proposed a public key encryption with keyword search (PEKS) scheme. Meanwhile, PEKS is also used as a name for the class of M/S schemes.

The need for key distribution. Some SE schemes extend the $*$/S setting to allow multi-user reading ($*$/M). This extension introduces the need for distributing the secret key to allow multiple users to search in the encrypted data. Some SE schemes use key sharing; other schemes use key distribution, proxy re-encryption or other techniques to solve the problem.

User revocation. An important requirement that comes with the multi reader schemes is user revocation. Curtmola et al. [75] extend their single-user scheme with broadcast encryption [80] (BE) to a multi-user scheme (S/M). Since only one key is shared among all users, each revocation requires a new key to be distributed to the remaining users, which causes a high revocation overhead.

In other schemes, each user might have its own key, which makes user revocation easier and more efficient.

Research challenges/Trade-offs. There are three main research directions in SE: improve (i) the efficiency, (ii) the security, and (iii) the query expressiveness. Efficiency is measured by the computational and communication complexity of the scheme. To define the security of a scheme formally, a variety of different security models have been proposed. Since security is never free, there is always a trade-off between security on the one hand, and efficiency and query expressiveness on the other. Searchable encryption schemes that use a security model with a more powerful adversary are likely to have a higher complexity.

The query expressiveness of the scheme defines what kind of search queries are supported. In current approaches it is often the case that more expressive queries result in either less efficiency and/or less security. Thus, the trade-offs of SE schemes are threefold: (i) security vs. efficiency, (ii) security vs. query expressiveness, and (iii) efficiency vs. query expressiveness.

2.1.3 *Scope of the Chapter*

The main techniques for provably secure searchable encryption are searchable symmetric encryption (SSE) and public key encryption with keyword search (PEKS). However, techniques such as predicate encryption (PE), inner product encryption (IPE), anonymous identity-based encryption (AIBE), and hidden-vector encryption (HVE) have been brought into relation with searchable encryption [53, 87, 115, 135]. Since the main focus of these techniques is (fine grained) access control (AC) rather than searchable encryption, those AC techniques are mentioned in the related work section but are otherwise not our focus.

2.1.4 *Contributions*

We give a complete and comprehensive overview of the field of SE, which provides an easy entry point for non-specialists and allows researchers to keep up with the many approaches. The survey gives beginners a solid foundation for further research. For researchers, we identify various gaps in the field and indicate open research problems. We also point out connections between the many schemes. With our extensive tables and details about efficiency and security, we allow practitioners to find (narrow down the number of) suitable schemes for the many different application scenarios of SE.

2.1.5 *Reading Guidelines*

We discuss all papers based on the following four aspects. The main features are emphasized in *italics* for easy readability:

GENERAL INFORMATION: The general *idea* of the scheme will be stated.

EFFICIENCY: The efficiency aspect focuses on the computational complexity of the *encryption/index generation* (upload phase) and the *search/test* (query phase) algorithms. For a fair comparison of the schemes, we report the number of operations required in the algorithms. Where appli-

cable, we give information on the update complexity or interactiveness (number of rounds).

SECURITY: To ease the comparison of SE schemes with respect to their security, we briefly outline the major fundamental security definitions in Section 2.2.3 such that the to be discussed SE schemes can be considered as being secure in a certain modification of one of these basic definitions. We provide short and intuitive explanations of these modifications and talk about the underlying security assumptions.[2]

SEE ALSO: We refer to related work within the survey and beyond. For a reference within the survey, we state the original paper reference and the section number in which the scheme is discussed. For references beyond the survey, we give only the paper reference. Otherwise, we omit this aspect.

This reading guideline will act as our framework to compare the different works. Several pioneering schemes (i. e., [47, 75, 171]) will be discussed in more detail, to get a better feeling on how searchable encryption works. Each architecture-section ends with a synthesis and an overview table which summarizes the discussed schemes. The tables (2.1, 2.3, 2.5, 2.7) are, like the sections, arranged by the query expressiveness. The first column gives the paper and section reference. The complexity or efficiency part of the table is split into the encrypt, trapdoor, and search algorithms of the schemes and quantifies the most expensive operations that need to be computed. The security part of the table gives information on the security definitions, assumptions, and if the random oracle model (ROM) is used to prove the scheme secure. The last column highlights some of the outstanding features of the schemes.

2.1.6 *Organization of the Chapter*

The rest of the chapter is organized as follows. Section 2.2 gives background information on indexes and the security definitions used in this survey. The discussion of the schemes can be found in Section 2.3 (S/$*$) and Section 2.4 (M/$*$). We divided these sections into the four architectures: Section 2.3.1 (S/S), Section 2.3.2 (S/M), Section 2.4.1 (M/S), and Section 2.4.2 (M/M). In these sections, the papers are arranged according to their expressiveness. We start with single equality tests, then conjunctive equality tests, followed by extended search queries, like subset, fuzzy or range queries, or queries based on inner products. Inside these subsections, the schemes are ordered chronologically. Section 2.5 discusses the related work, in particular seminal schemes on access control. Section 2.6 concludes and discusses future work.

2.2 PRELIMINARIES

This section gives background information on indexes, privacy issues, and security definitions used in this survey.

2 A detailed security analysis lies outside the scope of this work. We stress that some of the mentioned modifications may have unforeseen security implications that we do not touch upon. The interested reader is recommended to look up the original reference for more details.

document id	keywords
1	w_2, w_5, w_7
2	w_1, w_2, w_4, w_6, w_8
...	...
n	w_2, w_5, w_6

(a) Forward index.

keyword	document ids
w_1	$2, 3, 9$
w_2	$1, 2, 6, 7, n$
...	...
w_m	$1, 3, 8$

(b) Inverted index.

Figure 2.2: Example of an unencrypted forward and inverted index.

2.2.1 *Efficiency in SE Schemes*

As mentioned above, searchable encryption schemes usually come in two classes. Some schemes directly encrypt the plaintext data in a special way, so that the ciphertext can be queried (e. g., for keywords). This results in a search time linear in the length of the data stored on the server. In our example using n documents with w keywords, yields a complexity linear in the number of keywords per document $O(nw)$, since each keyword has to be checked for a match.

To speed up the search process, a common tool used in databases is an index, which is generated over the plaintext data. Introducing an index can significantly decrease the search complexity and thus increases the search performance of a scheme. The increased search performance comes at the cost of a pre-precessing step. Since the index is built over the plaintext data, generating an index is not always possible and highly depends on the data to be encrypted. The two main approaches for building an index are:

- A forward index, is an index per document (see Figure 2.2a) and naturally reduces the search time to the number of documents, i. e., $O(n)$. This is because one index per document has to be processed during a query.
- Currently, the prevalent method for achieving sub-linear search time is to use an inverted index, which is an index per keyword in the database (see Figure 2.2b). Depending on how much information we are willing to leak, the search complexity can be reduced to $O(\log w')$ (e. g.using a hash tree) or $O(|D(w)|)$ in the optimal case, where $|D(w)|$ is the number of documents containing the keyword w.

Note that the client does not have to build an index on the plaintexts. This is the case, e. g., for the scheme by Song, Wagner, and Perrig [171] (SWP) or when deterministic encryption is used. In case of deterministic encryption, it is sometimes reasonable to index the ciphertexts to speed up the search.

All schemes discussed in this survey, except for SWP, make use of a searchable index. Only the SWP scheme encrypts the message in such a way, that the resulting ciphertext is directly searchable and decryptable.

2.2.2 *Privacy Issues in SE Schemes*

An SE scheme will leak information, which can be divided into three groups: index information, search pattern, and access pattern.

- Index information refers to the information about the keywords contained in the index. Index information is leaked from the stored ciphertext/index. This information may include the number of keywords per document/database, the number of documents, the documents length, document ids, and/or document similarity.
- Search pattern refers to the information that can be derived in the following sense: given that two searches return the same results, determine whether the two searches use the same keyword/predicate. Using deterministic trapdoors directly leaks the search pattern. Accessing the search pattern allows the server to use statistical analysis and (possibly) determine (information about) the query keywords.
- Access pattern refers to the information that is implied by the query results. For example, one query can return a document x, while the other query could return x and another 10 documents. This implies that the predicate used in the first query is more restrictive than that in the second query.

Most papers follow the security definition deployed in the traditional searchable encryption [75]. Namely, it is required that nothing should be leaked from the remotely stored files and index, beyond the outcome and the pattern of search queries. SE schemes should not leak the plaintext keywords in either the trapdoor or the index. To capture the concept that neither index information nor the search pattern is leaked, Shen et al. [167] (SSW) formulate the definition of *full security*. All discussed papers (except for SSW) leak at least the search pattern and the access pattern. The two exceptions protect the search pattern and are fully secure.

2.2.3 *A Short History of Security Definitions for (S)SE*

When Song et al. [171] proposed the first SE scheme, there were no formal security definitions for the specific needs of SE. However, the authors proved their scheme to be a secure pseudo-random generator. Their construction is even *indistinguishable against chosen plaintext attacks (IND-CPA)* secure [109]. Informally, an encryption scheme is IND-CPA secure, if an adversary $\mathcal{A}$ cannot distinguish the encryptions of two arbitrary messages (chosen by $\mathcal{A}$), even if $\mathcal{A}$ can adaptively query an encryption oracle. Intuitively, this means that a scheme is IND-CPA secure if the resulting ciphertexts do not even leak partial information about the plaintexts. This IND-CPA definition makes sure, that ciphertexts do not leak information. However, in SE the main information leakage comes from the trapdoor/query, which is not taken into account in the IND-CPA security model. Thus, IND-CPA security is not considered to be the right notion of security for SE.

The first notion of security in the context of SE was introduced by Goh [92] (Section 2.3.1), who defines security for indexes known as *semantic security (indistinguishability) against adaptive chosen keyword attacks (IND1-CKA)*. IND1-CKA makes sure, that $\mathcal{A}$ cannot deduce the document's content from its index. An IND1-CKA secure scheme generates indexes that appear to contain the same number of words for equal size documents (in contrast to unequal size documents). This means, that given two encrypted documents of *equal size* and an index, $\mathcal{A}$ cannot decide which document is encoded in the index. IND1-CKA was proposed for "secure indexes", a secure data structure with many

uses next to SSE. Goh remarks, that IND1-CKA does not require the trapdoors to be secure, since it is not required by all applications of secure indexes.

Chang and Mitzenmacher [64] introduced a new simulation-based IND-CKA definition which is a stronger version of IND1-CKA in the sense that an adversary cannot even distinguish indexes from two *unequal size* documents. This requires, that unequal size documents have indexes that appear to contain the same number of words. In addition, Chang and Mitzenmacher tried to protect the trapdoors with their security definition. Unfortunately, their formalization of the security notion was incorrect, as pointed out by Curtmola et al. [75], and can be satisfied by an insecure SSE scheme.

Later, Goh introduced the *IND2-CKA* security definition which protects the document size like Chang and Mitzenmacher's definition, but still does not provide security for the trapdoors. Both IND1/2-CKA security definitions are considered weak in the context of SE because they do not guarantee the security of the trapdoors, i. e., they do not guarantee that the server cannot recover (information about) the words being queried from the trapdoor.

Curtmola et al. [75] revisited the existing security definitions and pointed out, that previous definitions are not adequate for SSE, and that the security of indexes and the security of trapdoors are inherently linked. They introduce two new adversarial models for searchable encryption, a *non-adaptive (IND-CKA1)* and an *adaptive (IND-CKA2)* one, which are widely used as the standard definitions for SSE to date. Intuitively, the definitions require that nothing should be leaked from the remotely stored files and index beyond the outcome and the search pattern of the queries. The IND-CKA1/2 security definitions include security for trapdoors and guarantee that the trapdoors do not leak information about the keywords (except for what can be inferred from the search and access patterns). Non-adaptive definitions only guarantee the security of a scheme, if the client generates all queries at once. This might not be feasible for certain (practical) scenarios [75]. The adaptive definition allows $\mathcal{A}$ to choose its queries as a function of previously obtained trapdoors and search outcomes. Thus, IND-CKA2 is considered a strong security definition for SSE.

In the asymmetric (public key) setting (see Boneh et al. [47]), schemes do not guarantee security for the trapdoors, since usually the trapdoors are generated using the public key. The definition in this setting guarantees, that no information is learned about a keyword unless the trapdoor for that word is available. An adversary should not be able to distinguish between the encryptions of two challenge keywords of its choice, even if it is allowed to obtain trapdoors for *any* keyword (except the challenge keywords). Following the previous notion, we use *PK-CKA2* to denote indistinguishability against adaptive chosen keyword attacks of public key schemes in the remainder of this survey.

Several schemes adapt the above security definitions to their setting. We will explain these special purpose definitions in the individual sections and mark them in the overview tables.

Other security definitions were introduced and/or adapted for SE as follows:

- *Universal composability (UC)* is a general-purpose model which says, that protocols remain secure even if they are arbitrarily composed with other instances of the same or other protocols. The KO scheme [119] (Section 2.3.1) provides IND-CKA2 security in the UC model (denoted as

UC-CKA2 in the reminder), which is stronger than the standard IND-CKA2.

- *Selectively secure (SEL-CKA)* [61] is similar to PK-CKA2, but the adversary $\mathcal{A}$ has to commit to the search keywords at the beginning of the security game instead of after the first query phase.
- *Fully Secure (FS)* is a security definition in the context of SSE introduced by Shen et al. [167], that allows nothing to be leaked, except for the access pattern.

DETERMINISTIC ENCRYPTION. Deterministic encryption involves no randomness and thus produces always the same ciphertext for a given plaintext and key. In the public key setting, this implies that a deterministic encryption can never be IND-CPA secure, as an attacker can run brute force attacks by trying to construct all possible plaintext-ciphertext pairs using the encryption function. Deterministic encryption allows more efficient schemes, whose security is weaker than using probabilistic encryption. Deterministic SE schemes try to address the problem of searching in encrypted data from a practical perspective where the primary goal is efficiency. An example of an immediate security weakness of this approach is that deterministic encryption inherently leaks message equality. Bellare et al.'s [29] (Section 2.4.2) security definition for deterministic encryption in the public key setting is similar to the standard IND-CPA security definition with the following two exceptions. A scheme that is secure in Bellare et al.'s definition requires plaintexts with large min-entropy and plaintexts that are independent from the public key. This is necessary to circumvent the above stated brute force attack; here large min-entropy ensures that the attacker will have a hard time brute-forcing the correct plaintext-ciphertext pair. The less min-entropy the plaintext has, the less security the scheme achieves. Amanatidis et al. [12] (Section 2.3.1) and Raykova et al. [157] (Section 2.3.2) provide a similar definition for deterministic security in the symmetric setting. Also for their schemes, plaintexts are required to have large min-entropy. Deterministic encryption is not good enough for most practical purposes, since the plaintext data usually has low min-entropy and thus leaks too much information, including document/keyword similarity.

RANDOM ORACLE MODEL VS. STANDARD MODEL. Searchable encryption schemes might be proven secure (according to the above definitions) in the random oracle model [23] (ROM) or the standard model (STM). Other models, e. g., generic group model exist, but are not relevant for the rest of the survey. The STM is a computational model in which an adversary is limited only by the amount of resources available, i. e., time and computational power. This means, that only complexity assumptions are used to prove a scheme secure. The ROM replaces cryptographic primitives by idealized versions, e. g., replacing a cryptographic hash function with a genuinely random function. Solutions in the ROM are often more efficient than solutions in the STM, but have the additional assumption of idealized cryptographic primitives.

2.3 SINGLE WRITER SCHEMES (S/∗)

This section deals with the S/S and S/M schemes.

- $\mathsf{Encrypt}(k', k'', M = \{w_i\})$:
  1. Encrypt w_i with a deterministic encryption algorithm and split $X_i = E_{k''}(w_i)$ into two parts $X_i = \langle L_i, R_i \rangle$.
  2. Generate the pseudo-random value S_i.
  3. Calculate the key $k_i = f_{k'}(L_i)$.
  4. Compute $F_{k_i}(S_i)$, where $F(\cdot)$ is a pseudo-random function, and set $Y_i = \langle S_i, F_{k_i}(S_i) \rangle$.
  5. Output the searchable ciphertext as $C_i = X_i \oplus Y_i$.
- $\mathsf{Trapdoor}(k', k'', w)$:
  1. Encrypt w as $X = E_{k''}(w)$, where X is split into two parts $X = \langle L, R \rangle$.
  2. Compute $k = f_{k'}(L)$.
  3. Output $T_w = \langle X, k \rangle$
- $\mathsf{Search}(T_w = \langle X, k \rangle)$:
  1. Check whether $C_i \oplus X$ is of the form $\langle s, F_k(s) \rangle$ for some s.

Figure 2.3: Algorithmic description of the Song, Wagner and Perrig scheme.

2.3.1 *Single Writer/Single Reader (S/S)*

In a single writer/single reader (S/S) scheme the secret key owner is allowed to create searchable content and to generate trapdoors to search. The secret key should normally be known only by one user, who is the writer and the reader using a symmetric encryption scheme. However, other scenarios, e. g., using a PKE and keeping the public key secret, are also possible, but result in less efficient schemes.

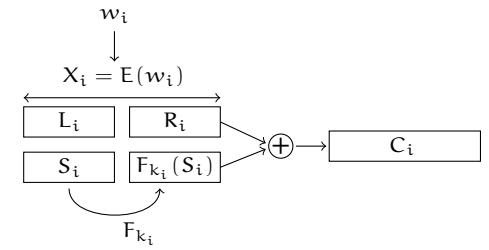
Single Equality Test

With an equality test we mean an exact keyword match for a single search keyword.

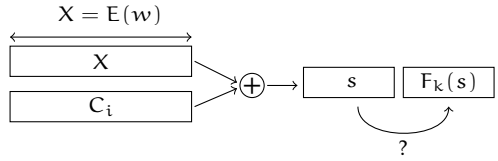
SEQUENTIAL SCAN. Song et al. [171] (SWP) propose the first practical scheme for searching in encrypted data by using a special two-layered encryption construct that allows to search the ciphertexts with a sequential scan. The *idea* is to encrypt each word separately and then embed a hash value (with a special format) inside the ciphertext. To search, the server can extract this hash value and check, if the value is of this special form (which indicates a match).

The disadvantages of SWP are that it has to use fix-sized words, that it is not compatible with existing file encryption standards and that it has to use their specific two-layer encryption method which can be used only for plain text data and not for example on compressed data.

Details: To create searchable ciphertext (cf. Figure 2.4a), the message is split into fixed-size words w_i and encrypted with a deterministic encryption algorithm $E(\cdot)$. Using a deterministic encryption is necessary to generate the correct trapdoor. The encrypted word $X_i = E(w_i)$ is then split into two parts $X_i = \langle L_i, R_i \rangle$. A pseudo-random value S_i is generated, e. g., with help of a stream cipher. A key $k_i = f_{k'}(L_i)$ is calculated (using a pseudo-random function $f(\cdot)$) and used for the keyed hash function $F(\cdot)$ to hash the value S_i. This results in the value $Y_i = \langle S_i, F_{k_i}(S_i) \rangle$ which is used to encrypt X_i as $C_i = X_i \oplus Y_i$, where $\oplus$ denotes the XOR.



(a) SWP: Encryption



(b) SWP: Sequential search

Figure 2.4: Song, Wagner, and Perrig (SWP) [171] scheme.

To search, a trapdoor is required. This trapdoor contains the encrypted keyword to search for $X = E(w) = \langle L, R \rangle$ and the corresponding key $k = f_{k'}(L)$. With this trapdoor, the server is now able to search (cf. Figure 2.4b), by checking for all stored ciphertexts C_i, if $C_i \oplus X$ is of the form $\langle s, F_k(s) \rangle$ for some s. If so, the keyword was found. The detailed algorithm is shown in Figure 2.3.

EFFICIENCY: The complexity of the encryption and search algorithms is linear in the total number of words per document (i. e., worst case). To *encrypt*, one encryption, one XOR, and two pseudo-random functions have to be computed per word per document. The trapdoor requires one encryption and a pseudo-random function. The *search* requires one XOR and one pseudo-random function per word per document.

SECURITY: SWP is the first searchable encryption scheme and uses *no formal security definition for SE*. However, SWP is *IND-CPA* secure under the assumption that the underlying *primitives are proven secure/exist* (e. g., pseudo-random functions). IND-CPA security does not take queries into account and is thus of less interest in the context of SE. SWP *leaks* the potential positions (i. e., positions, where a possible match occurs, taking into account a false positive rate, e. g., due to collisions) of the queried keywords in a document. After several queries it is possible to learn the words inside the documents with statistical analysis.

SEE ALSO: Brinkman et al. [56] show that the scheme can be applied to XML data. SWP is used in CryptDB [156].

SECURE INDEXES PER DOCUMENT. Goh [92] addresses some of the limitations (e. g., use of fixed-size words, special document encryption) of the SWP scheme by adding an index for each document, which is independent of the underlying encryption algorithm. The *idea* is to use a Bloom filter (BF) [35] as a per document index.

A BF is a data structure which is used to answer set membership queries. It is represented as an array of b bits which are initially set to 0. In general

the filter uses r independent hash functions h_t, where $h_t : \{0,1\}^* \to [1,b]$ for $t \in [1,r]$, each of which maps a set element to one of the b array positions. For each element e (e. g., keywords) in the set $\mathcal{S} = \{e_1, \ldots e_m\}$ the bits at positions $h_1(e_i), \ldots, h_r(e_i)$ are set to 1. To check whether an element x belongs to the set $\mathcal{S}$, check if the bits at positions $h_1(x), \ldots, h_r(x)$ are set to 1. If so, x is considered a member of set $\mathcal{S}$.

By using one BF per document, the search time becomes linear in the number of documents. An inherent problem of using Bloom filters is the possibility of *false positives*. With appropriate parameter settings the false positive probability can be reduced to an acceptable level. Goh uses BF, where each distinct word in a document is processed by a pseudo-random function twice and then inserted into the BF. The second run of the pseudo-random function takes as input the output of the first run and, in addition, a unique document identifier, which makes sure that all BF look different, even for documents with the same keyword set. This avoids leaking document similarity upfront.

EFFICIENCY: The *index generation* has to generate one BF per document. Thus the algorithm is linear in the number of distinct words per document. The BF lookup is a constant time operation and has to be done per document. Thus, the time for a *search* is proportional to the number of documents, in contrast to the number of words in the SWP scheme. The size of the document index is proportional to the number of distinct words in the document. Since a Bloom filter is used, the asymptotic constants are small, i. e., several bits.

SECURITY: The scheme is proven *IND1-CKA* secure. In a later version of the paper, Goh proposed a modified version of the scheme which is IND2-CKA secure. Both security definitions do not guarantee the security of the trapdoors, i. e., they do not guarantee that the server cannot recover (information about) the words being queried from the trapdoor.

A disadvantage of BF is, that the number of 1's is dependent on the number of BF entries, in this case the number of distinct keywords per document. As a consequence, the scheme leaks the number of keywords in each document. To avoid this leakage, padding of arbitrary words can be used to make sure that the number of 1's in the BF is nearly the same for different documents. The price to pay is a higher false positive rate or a larger BF compared to the scheme without padding.

INDEX PER DOCUMENT WITH PRE-BUILT DICTIONARIES. Chang and Mitzenmacher [64] develop two index schemes (CM-I, CM-II), similar to Goh [92]. The *idea* is to use a pre-built dictionary of search keywords to build an index per document. The index is an m-bit array, initially set to 0, where each bit position corresponds to a keyword in the dictionary. If the document contains a keyword, its index bit is set to 1. CM-$*$ assume that the user is *mobile* with limited storage space and bandwidth, so the schemes require only a small amount of communication overhead. Both constructions use only pseudo-random permutations and pseudo-random functions. CM-I stores the dictionary at the client and CM-II encrypted at the server. Both constructions can handle secure updates to the document collection in the sense that CM-$*$ ensure the security of the consequent submissions in the presence of previous queries.

E is a semantic secure symmetric encryption scheme, f is a pseudo-random function and π, ψ are two pseudo-random permutations. $\mathcal{D}(w)$ denotes the set of ids of documents that contain keyword w.

- $\mathsf{Keygen}(1^k, 1^l)$: Generate random keys $s, y, z \xleftarrow{R} \{0,1\}^k$ and output $K = (s, y, z, 1^l)$.
- $\mathsf{BuildIndex}(K, \mathcal{D} = \{D_j\})$:
  1. Initialization:
     a) scan $\mathcal{D}$ and build Δ', the set of distinct words in $\mathcal{D}$. For each word $w \in \Delta'$, build $\mathcal{D}(w)$;
     b) initialize a global counter ctr = 1.
  2. Build array A:
     a) for each $w_i \in \Delta'$: (build a linked list L_i with nodes $N_{i,j}$ and store it in array A)
        i. generate $\kappa_{i,0} \xleftarrow{R} \{0,1\}^l$
        ii. for $1 \leqslant j \leqslant |\mathcal{D}(w_i)|$:
            - generate $\kappa_{i,j} \xleftarrow{R} \{0,1\}^l$ and set node $N_{i,j} = \langle \mathsf{id}(D_{i,j}) || \kappa_{i,j} || \psi_s(\mathsf{ctr}+1) \rangle$, where $\mathsf{id}(D_{i,j})$ is the j^{th} identifier in $\mathcal{D}(w_i)$;
            - compute $E_{\kappa_{i,j-1}}(N_{i,j})$ and store it in $A[\psi_s(\mathsf{ctr})]$;
            - ctr = ctr + 1
        iii. for the last node of L_i (i.e., $N_{i,|\mathcal{D}(w_i)|}$), before encryption, set the address of the next node to NULL;
     b) let $m' = \sum_{w_i \in \Delta'} |\mathcal{D}(w_i)|$. If $m' < m$, then set remaining $(m - m')$ entries of A to random values of the same size as the existing m' entries of A.
  3. Build look-up table T:
     a) for each $w_i \in \Delta'$:
        i. value = $\langle \mathsf{addr}(A(N_{i,1})) || \kappa_{i,0} \rangle \oplus f_y(w_i)$;
        ii. set $T[\pi_z(w_i)] =$ value.
     b) if $|\Delta'| < |\Delta|$, then set the remaining $(|\Delta| - |\Delta'|)$ entries of T to random values.
  4. Output $\mathcal{I} = (A, T)$.
- $\mathsf{Trapdoor}(w)$: Output $T_w = (\pi_z(w), f_y(w))$.
- $\mathsf{Search}(\mathcal{I}, T_w)$:
  1. Let $T_w = (\gamma, \eta)$. Retrieve $\theta = T[\gamma]$. Let $\langle \alpha || \kappa \rangle = \theta \oplus \eta$.
  2. Decrypt L starting with the node at address α encrypted under key κ.
  3. Output the list of document identifiers in L.

Figure 2.5: Algorithmic description of the first Curtmola et al. [75] scheme (CGK^+-I). This scheme uses an inverted index and achieves sub-linear (optimal) search time.

EFFICIENCY: The CM-$*$ schemes associate a masked keyword index to each document. The *index generation* is linear in the number of distinct words per document. The time for a *search* is proportional to the total number of documents. CM-II uses a *two-round* retrieval protocol, whereas CM-I only requires one round for searching.

SECURITY: CM introduce a new simulation-based IND-CKA definition which is a stronger version of IND1-CKA. This new security definition has been broken by Curtmola et al. [75]. CM-$*$ still are at least *IND2-CKA* secure.

In contrast to other schemes, which assume only an honest-but-curious server, the authors discuss some security improvements that can deal with a malicious server which sends either incorrect files or incomplete search results back to the user.

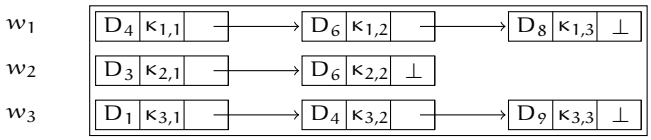
INDEX PER KEYWORD AND IMPROVED DEFINITIONS. Curtmola et al. [75] (CGK$^+$) propose two new constructions (CGK$^+$-I, CGK$^+$-II) where the *idea* is to add an inverted index, which is an index per distinct word in the database instead of per document (cf. Figure 2.2b). This reduces the search time to the number of documents that contain the keyword. This is not only sub-linear, but optimal.

Details (CGK$^+$-I): The index consists of i) an array A made of a linked list L per distinct keyword and ii) a look-up table T to identify the first node in A. To build the array A, we start with a linked list L_i per distinct keyword w_i (cf. Figure 2.6a). Each node $N_{i,j}$ of L_i consists of three fields $\langle a||b||c\rangle$, where a is the document identifier of the document containing the keyword, b is the key $\kappa_{i,j}$ which is used to encrypt the next node and c is a pointer to the next node or $\emptyset$. The nodes in array A are scrambled in a random order and then encrypted. The node $N_{i,j}$ is encrypted with the key $\kappa_{i,j-1}$ which is stored in the previous node $N_{i,j-1}$. The table T is a look-up table which stores per keyword w_i a node $N_{i,0}$ which contains the pointer to the first node $N_{i,1}$ in L_i and the corresponding key $\kappa_{i,0}$ (cf. Figure 2.6b). The node $N_{i,0}$ in the look-up table is encrypted (cf. Figure 2.6c) with $f_y(w_i)$ which is a pseudo-random function dependent on the keyword w_i. Finally, the encrypted $N_{i,0}$ is stored at position $\pi_z(w_i)$, where π is a pseudo-random permutation. Since the decryption key and the storage position per node are both dependent on the keyword, trapdoor generation is simple and outputs a trapdoor as $T_w = (\pi_z(w), f_y(w))$.

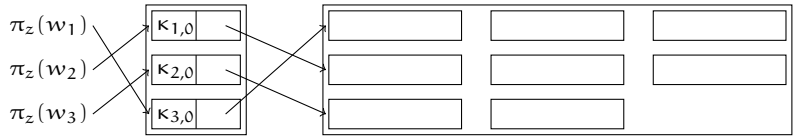
The trapdoor allows the server to identify and decrypt the correct node in T which includes the position of the first node and its decryption key. Due to the nature of the linked list, given the position and the correct decryption key for the first node, the server is able to find and decrypt all relevant nodes to obtain the documents identifiers. The detailed algorithm is shown in Figure 2.5.

EFFICIENCY: CGK$^+$ propose the first sub-linear scheme that achieve *optimal search time*. The *index generation* is linear in the number of distinct words per document. The server computation per *search* is proportional to $|\mathcal{D}(w)|$, which is the number of documents that contain a word w. CGK$^+$-II *search* is proportional to $|\mathcal{D}''(w)|$, which is the maximum number of documents that contain a word w.

Both CKG schemes use a special data structure (FKS dictionary [83]) for a look-up table. This makes the index more compact and reduces the



(a) CGK$^+$: Linked lists L_i



(b) CGK$^+$: Index table T and encrypted linked lists L_i

(c) CGK$^+$: Encrypted index table T and array A consisting of encrypted and scrambled linked lists L_i.

Figure 2.6: BuildIndex algorithm of Curtmola et al. (CGK-I) [75].

look-up time to $O(1)$. *Updates* are expensive due to the representation of the data. Thus, the scheme is more suitable for a static database than a dynamic one.

SECURITY: CGK-I is consistent with the new *IND-CKA1* security definition. CGK-II achieves *IND-CKA2* security, but requires higher communication costs and storage on the server than CGK-I.

EFFICIENTLY-SEARCHABLE AUTHENTICATED ENCRYPTION. Amanatidis et al. [12] (ABO) propose two schemes using *deterministic* message authentication codes (mac) to search. The *idea* of ABO-I (mac-and-encrypt) is to append a deterministic mac to an IND-CPA secure encryption of a keyword. The *idea* of ABO-II (encrypt-with-mac) is to use the mac of the plaintext (as the randomness) inside of the encryption. The schemes can use any IND-CPA secure symmetric encryption scheme in combination with a deterministic mac. ABO also discuss a prefix-preserving search scheme. To search with ABO-I, the client simply generates the mac of a keyword and stores it together with the encrypted keyword on the server. The server searches through the indexed macs to find the correct answer. In ABO-II, the client calculates the mac and embeds it inside the ciphertext for the keyword. The server searches for the queried ciphertexts.

EFFICIENCY: In ABO, the *index generation* per document is linear in the number of words. Both schemes require a mac and an encryption per keyword. The *search* is a simple database search and takes *logarithmic-time* $O(\log v)$ in the database size.

SECURITY: ABO define security for searchable deterministic symmetric encryption like Bellare et al. [29] (Section 2.4.2) which ABO call IND-EASE. Both schemes are proven *IND-EASE* secure. ABO-I is secure under the

assumption that the encryption scheme is IND-CPA secure and the mac is unforgeable against chosen message attacks (uf-cma) and privacy preserving. ABO-II is secure, if the encryption scheme is IND-CPA secure and the mac is a pseudo-random function.

SEE ALSO: Deterministic encryption in the M/M setting [29](Section 2.4.2).

INDEX PER KEYWORD WITH EFFICIENT UPDATES. Van Liesdonk et al. [177] propose two schemes (LSD-I, LSD-II) that offer efficient search and update, which differ in the communication and computation cost. LSD-$*$ use the same *idea* and are closely related to the CGK schemes (one index per keyword) but in contrast the LSD schemes support *efficient updates* of the database.

EFFICIENCY: In LSD-I, the *index generation* per document is linear in the number of distinct words. The algorithm uses only simple primitives like pseudo-random functions. The *search* time is logarithmic in the number of unique keywords stored on the server. LSD-I is an *interactive* scheme and requires *two rounds* of communication for the index generation, update, and search algorithms. LSD-II is non-interactive by deploying a hash chain at the cost of more computation for the search algorithm.

SECURITY: The authors prove their schemes *IND-CKA2* secure.

STRUCTURED ENCRYPTION FOR LABELED DATA. Chase and Kamara [65] (CK) proposed an adaptively secure construction that is based on CGK$^+$-I. The *idea* is to generate an inverted index in form of a padded and permuted dictionary. The dictionary can be implemented using hash tables, resulting in optimal search time.

EFFICIENCY: The *index generation* requires one initial permutation and two pseudo-random functions per distinct keyword in the database. The *search* requires the server to searches for the position of the desired query keyword and to decrypt the stored values, which are the document ids of the matching documents.

SECURITY: CK define a generalization of IND-CKA2 security where the exact leakage (e. g., the access or search pattern) can be influenced through leakage functions. This allows them to also hide the data structure from adversaries. However, their actual construction still leaks the access and search pattern. Conceptually, their scheme is *IND-CKA2* secure and in addition hides the data structure.

SEE ALSO: CK is based on CGK$^+$-I [75] (cf. Section 2.3.1).

VERIFIABLE SSE. Kurosawa and Ohtaki [119] (KO) propose a verifiable SSE scheme that is secure against active adversaries and/or a malicious server. The *idea* is to include a MAC tag inside the index to bind a query to an answer. KO use only PRFs and MACs for building the index. KO define security against active adversaries, which covers keyword privacy as well as reliability of the search results.

EFFICIENCY: *Index generation* requires n PRFs and n MACs per keyword in the database, where n is the number of documents. To *search*, the server performs n table look-ups. *Verification* of the results requires n MACs.

SECURITY: KO is proven *universally composable*(UC) secure. KO's UC security is stronger than IND-CKA2 (cf. Section 2.2.3)

SEE ALSO: KO is based on CGK$^+$-II [75] (cf. Section 2.3.1).

DYNAMIC SSE. Kamara et al. [109] (KPR) propose an extension for the CGK$^+$-I scheme, to allow efficient updates (add, delete, and modify documents) of the database. The *idea* is to add a *deletion array* to keep track of the search array positions that need to be modified in case of an update. In addition KPR use homomorphically encrypted array pointers to modify the pointers without decrypting. To add new documents, the server uses a *free list* to determine the free positions in the search array. KPR uses only PRFs and XORs.

EFFICIENCY: KPR achieves optimal search time, while at the same time handling efficient updates. *Index generation* requires 8 PRFs per keyword. To *search*, the server performs a table look-up for the first node and decrypts the following nodes by performing an XOR operation per node. Each node represents a document that contains the search keyword.

SECURITY: KPR define a variant of IND-CKA2 security that, similar to CK (cf. Section 2.3.1), allows for parametrized leakage and in addition is extended to include dynamic operations (like adding and deleting items). Conceptually, their security definition is a generalization of *IND-CKA2*. Updates leak a small amount of information, i. e., the trapdoors of the keywords contained in an updated document. They prove the security in the *random oracle (RO)* model.

SEE ALSO: KPR is an extension of CGK$^+$-I [75] (cf. Section 2.3.1).

PARALLEL AND DYNAMIC SSE. Kamara and Papamanthou [108] (KP) use the advances in multi-core architectures to propose a new dynamic SSE scheme which is highly parallelizable. KP provide a new way to achieve sublinear search time that is not based on Curtmola et al.'s scheme. The *idea* is to use a tree-based multi-map data structure per keyword which they call *keyword red-black* (KRB) trees. KRB trees are similar to binary trees with pointers to a file as leaves. Each node stores information, if at least one of its following nodes is a path to a file identifier containing the keyword. These KRB trees can be searched in $O(D(v)\log n)$ sequential time or in parallel $O(\frac{D(v)}{p}\log n)$, where p is the number of processors. KP also allows efficient updates, but with 1.5 rounds of interaction.

EFFICIENCY: *Encryption* requires per distinct keyword in the database $2n-1$ (nodes per tree) encryptions, where n is the number of documents. That is each node of a KRB tree per keyword. *Search* requires $(D(v)\log n)$ decryptions.

SECURITY: KP define a variant of CKA2 security, which is slightly stronger than KPR's (cf. Section 2.3.1) CKA2 variant. The difference is that during an update operation (performed before any search operation) no information is leaked. Conceptually, their security definition is a generalization of *IND-CKA2*. KP prove the security in the *RO* model.

Conjunctive Keyword Search

With conjunctive keyword search we mean schemes that allow a client to find documents containing all of several keywords in a single query, i. e., single run over the encrypted data. Building a conjunctive keyword search scheme from

a single keyword scheme in a naïve way provides the server with a trapdoor for each individual keyword. The server performs a search for each of the keywords separately and returns the intersection of all results. This approach leaks which documents contains each individual keyword and may allow the server to run statistical analysis to deduce information about the documents and/or keywords.

FIRST CONJUNCTIVE SEARCH SCHEMES. Golle et al. [97] (GSW) pioneer the construction of conjunctive keyword searches and present two SE schemes (GSW-I, GSW-II). Their *idea* for conjunctive searches is to assume that there are special keyword fields associated with each document. Emails for example could have the keyword fields: "From", "To", "Date", and "Subject". Using keyword fields, the user has to know in advance where (in which keyword field) the match has to occur. The communication and storage cost linearly depend on the number of stored data items (e. g., emails) in the database. Hence, GSW-$*$ are not suitable for large scale databases.

EFFICIENCY: *Encryption* in GSW-I requires $1 + v$ exponentiations per document, where v is the number of keywords per document. GSW-I requires two modular exponentiations per document for each *search*. The size of a trapdoor is linear in the total number of documents. Most of the communication can be done off-line, because the trapdoor is split into two parts and the first part, which is independent of the conjunctive query that the trapdoor allows, can be transmitted long before a query. The second part of the trapdoor is a constant amount of data which depends on the conjunctive query that the trapdoor allows and therefore must be sent online at query time. After receiving a query, the server combines it with the first part to obtain a full trapdoor.

Encryption in GSW-II requires the client to compute $2v + 1$ exponentiations. To *search*, the server has to perform $2k + 1$ symmetric prime order pairings per document (k is the number of keywords to search). The size of a trapdoor is constant in the number of documents, but linear in the number of keyword fields. GSW-II doubles the storage size on the server compared to GSW-I.

SECURITY: GSW extend the IND1-CKA definition to conjunctive keyword searches, meaning that for empty conjunctions (i. e., when querying a single keyword) the definition is the same as IND1-CKA. Therefore, we can say that GSW-I is proven *IND1-CKA* secure in the *RO* model. The security relies on the Decisional Diffie-Hellman *(DDH)* [39] assumption.

The security of GSW-II relies on a new, non-standard, hardness assumption and is also proven to be *IND1-CKA* secure.

SECURE IN THE STANDARD MODEL. Ballard et al. [19] (BKM) propose a construction for conjunctive keyword searches, where the *idea* is to use Shamir's Secret Sharing [165] (SSS). BKM requires keyword fields.

efficiency: BKM requires a trapdoor size that is linear in the number of documents being searched. *Index generation* uses a pseudo-random function per keyword. The trapdoor and *search* algorithms need to perform a standard polynomial interpolation for the SSS per document.

security: BKM is proven secure under the same extended *IND1-CKA* definition as GSW (cf. Section 2.3.1). The security is based on the security of *SSS* in the *standard model (ST)*.

CONSTANT COMMUNICATION AND STORAGE COST. Byun et al. [57] (BLL) construct a conjunctive keyword search scheme with constant communication and storage cost. The *idea* is to improve the communication and storage costs necessary for large databases by using bilinear maps. Communication of BLL is more efficient than both schemes by Golle et al., but encryption is less efficient. BLL requires keyword fields.

EFFICIENCY: BLL uses symmetric prime order bilinear maps. The *encryption* requires one bilinear map per keyword in a document. The *search* requires two bilinear maps per document.

SECURITY: BLL use the same extended IND1-CKA definition for conjunctive queries as GSW (cf. Section 2.3.1). The security of the scheme relies on a new multi decisional bilinear Diffie-Hellman *(MBDH)* assumption, which the authors prove to be equivalent to the decisional Bilinear Diffie-Hellman *(BDH)* assumption [42, 107]. BLL is proven secure under the mentioned extended version of *IND1-CKA* in the *RO* model under the *BDH* assumption.

SMALLER TRAPDOORS. Ryu and Takagi [161] (RT) propose an efficient construction for conjunctive keyword searches where the size of the trapdoors for several keywords is nearly the same as for a single keyword. The *idea* is to use Kiltz an Galindo's work [116] on identity-based key encapsulation. RT requires keyword fields.

EFFICIENCY: RT uses asymmetric pairings [42] in groups of prime order. *Encryption* requires one pairing per document and the server has to perform two pairings per document to *search*. RT achieves better performance than previous schemes (computational and communication costs) and has almost the same communication cost as that of searching for a single keyword.

SECURITY: RT use the extended IND1-CKA definition for conjunctive queries (cf. GSW in Section 2.3.1). RT is proven secure under their extended *IND1-CKA* definition in the *RO* model under their new variant of the External Diffie-Hellman (XDH) assumption, in which the DDH problem is mixed with a random element of G_2. They call this the external co-Diffie-Hellman *(coXDH)* assumption. The XDH assumption was first introduced by Scott [163] and later formalized by Boneh et al. [46] and Ballard et al. [18].

KEYWORD FIELD FREE CONJUNCTIVE KEYWORD SEARCH. Wang et al. [182] (WWP-III) present the first keyword-field free conjunctive keyword search scheme which is proven secure in the ST model. The *idea* is to remove the keyword fields by using a bilinear map per keyword per document index.

EFFICIENCY: WWP-III uses symmetric bilinear pairings of prime order. The *index generation* constructs a v'-degree polynomial per document, where v' is the number of distinct keywords contained in the document. The algorithm requires $v' + 1$ exponentiations per document. A *search* requires a bilinear map per keyword per document index. The size of a

query/trapdoor is linear in the number of keywords contained in the index.

SECURITY: WWP-III is proven secure in the *ST* model under the extended version of *IND1-CKA* from GSW (cf. Section 2.3.1). The security is based on the discrete logarithm *(DL)* assumption [77] and the l-decisional Diffie-Hellman inversion *(l-DDHI)* assumption [60].

SEE ALSO: The authors also extend WWP-III to dynamic groups in the M/M setting (cf. Section 2.4.2). The first keyword-field free conjunctive keyword search scheme in the *RO* model is due to Wang et al. [181] (cf. Section 2.4.2).

SUB-LINEAR CONJUNCTIVE KEYWORD SEARCH. Cash et al. [63] (CJJ$^+$) recently proposed the first sub-linear SSE construction supporting conjunctive queries for arbitrarily-structured data. The construction is based on the inverted index approach of Curtmola et al. [75] (Section 2.3.1). CJJ$^+$ provide a highly scalable implementation. The *idea* is to query for the estimated least frequent keyword first and then filter the search results for the other keywords. The search protocol is *interactive* in the sense, that the server replies to a query with encrypted document ids. The client has to decrypt these ids before retrieving the corresponding documents.

EFFICIENCY: The *index generation* requires for each distinct keyword v'' in the database, that for all $D(v)$ (documents that contain the keyword) six pseudo-random functions, one encryption, and one exponentiation is computed. A *search* requires the server to perform two PRF, one XOR, and $(k-1)$ exponentiation per document that contain the query keyword $D(v)$, where k is the number of keywords in the trapdoor.

SECURITY: CJJ$^+$ define a generalization of IND-CKA2 for conjunctive queries which is parametrized by leakage functions. CJJ$^+$ is proven *IND-CKA2* secure under their generalized definition under the *DDH* assumption.

Extended Queries

In this section we will discuss schemes, that allow more powerful queries, e. g., fuzzy search and inner products.

FUZZY/SIMILARITY SEARCH USING HAMMING DISTANCE. Park et al. [151] (PKL$^+$) propose a method to search for keywords with errors over encrypted data, based on approximate string matching. To search for similar words, the *idea* is to encrypt a word character by character and use the Hamming distance to search for similar keywords. Because character-wise encryption is not secure (domain is too limited) they design a new encryption algorithm. PKL$^+$ comes in two versions. PKL$^+$-I is more secure (i. e., achieves query privacy) and PKL$^+$-II is more efficient.

EFFICIENCY: PKL$^+$-$*$ use only pseudo-random functions, pseudo-random generators, one-way functions, and exponentiations. The *index generation of PKL$^+$-I* requires one PRF, one hash, and one exponentiation per character, per keyword, per document. The trapdoor generation requires a PRF per character of the keyword. To search, the server has to generate a pattern which requires a hash and two exponentiations per character per keyword per stored index. The *search of PKL$^+$-I* is linear in the

number of documents and requires the server to compute the Hamming distance between the pattern and a keyword, per keyword per index.

The *index generation of PKL$^+$-II* requires a PRF and a hash per character per keyword per document. The trapdoor algorithm takes $\mathtt{m}\mathtt{l}$ PRF, where $\mathtt{m}$ is the number of keyword fields and $\mathtt{l}$ the number of characters of the keyword. The pattern generation requires $\mathtt{m}\mathtt{l}$ hash functions and the *search of PKL$^+$-II* has to calculate $\mathtt{m}$ Hamming distances per index stored on the server.

SECURITY: PKL$^+$ redefine IND1-CKA to their setting, by allowing the Hamming distance to leak. The security of PKL$^+$ is based on the *DDH* assumption. Both PKL$^+$ schemes are proven secure under their *IND1-CKA* definition in the *RO* model. PKL$^+$-II does not achieve query privacy, since no random factor in the trapdoor generation is used.

FUZZY SEARCH USING LOCALITY SENSITIVE HASHING. Adjedj et al. [10] (ABC$^+$) propose a fuzzy search scheme for biometric identification. The *idea* is to use locality sensitive hashing (LSH) to make sure, that similar biometric readouts from the same person are hashed to the same value. LSH outputs (with high probability) the same hash value for inputs with *small* Hamming distance. The LSH values are then used in combination with the CGK$^+$-II scheme (Section 2.3.1). After a search, the results have to be decrypted on the client.

EFFICIENCY: *Encryption* requires $\mathtt{b}$ hash functions, PRPs, and Encryptions per document (here: user of the identification system), where $\mathtt{b}$ is the number of hash functions used for the LSH. The *search* consists of $\mathtt{b} \cdot \mathcal{D}''(w)$ database searches, where $\mathcal{D}''(w)$ is the maximum number of user identifiers for a biometric template w.

SECURITY: ABC$^+$ use the standard CGK$^+$-II scheme and is thus *IND-CKA2* secure.

SEE ALSO: Curtmola et al. [75] (Section 2.3.1).

FULLY SECURE SEARCH BASED ON INNER PRODUCTS. Shen et al. [167] (SSW) present a symmetric-key predicate encryption scheme which is based on inner products. The *idea* is to represent the trapdoor and the searchable content as vectors and calculate the inner product of those during the search phase. Thus, SSW does not leak which of the search terms matches the query. SSW introduce the notion of predicate privacy (tokens leak no information about the encoded query predicate). SSW also give a definition for *fully secure* predicate encryption, which means, that nothing should be leaked, except for the access pattern. The dot product enables more complex evaluations on disjunctions, polynomials, and CNF/DNF formulae.

EFFICIENCY: SSW uses composite order symmetric bilinear pairings where the order of the group is the product of four primes. *Encryption* requires $6v + 2$ exponentiations per document, where v is the number of keywords. Trapdoor generation requires $8v$ exponentiations and the *search* algorithm requires $2v + 2$ pairings per document.

SECURITY: The security of SSW relies on three assumptions: (i) the generalized Assumption 1 from Katz et al. [112] *(GKA1)*, (ii) the generalized 3-party Diffie-Hellman *(C3DH)* [43] assumption, and (iii) the decisional

linear *(DLIN)* assumption [46]. SSW is proven *single challenge* (SC) (attacker is limited to a single instance of the security game) fully secure (FS) in the *selective model* (SEL) [61], where an adversary commits to an encryption vector at the beginning of the security game. SSW hides the search pattern.

FUZZY SEARCH USING EDIT DISTANCE. Li et al. [125] (LWW$^+$) propose a search scheme for fuzzy keyword searches based on pre-specified similarity semantics using the Edit distance (number of operations (substitution, deletion, insertion) required to transform one word into another). The *idea* is to pre-compute fuzzy keyword sets $S_{k,d} = \{S'_{k,0}, S'_{k,1}, \cdots, S'_{k,d}\}$ with Edit distance d per keyword k and store them encrypted on the server. The trapdoors are generated in the same manner, so that the server can test for similarity. The set $S_{\mathtt{CAT},1}$ can be constructed as follows, where each $*$ represents an edit operation on that position: $S_{\mathtt{CAT},1} = \{\mathtt{CAT}, *\mathtt{CAT}, *\mathtt{AT}, \mathtt{C}*\mathtt{AT}, \mathtt{C}*\mathtt{T}, \mathtt{CA}*\mathtt{T}, \mathtt{CA}*, \mathtt{CAT}*\}$. The number of set elements is $\sum_{y=0}^{d} \sum_{x=l}^{l+y} \binom{x}{y}$, where d is the distance and l the length of the keyword in characters. The search is *interactive* and requires *two rounds* to retrieve the documents.

EFFICIENCY: *Encryption* requires the client to first construct the fuzzy sets. For each element of the set a pseudo-random function has to be computed. Upon receiving the trapdoor keyword set the *search* consists of a comparison per set element per document.

SECURITY: LWW$^+$ slightly modify the IND-CKA1 definition by allowing the encrypted index to leak the Edit distance between the plaintexts underlying the ciphertexts. They prove their scheme secure in this modified *IND-CKA1* definition.

EFFICIENT SIMILARITY SEARCH. Kuzu et al. [120] (KIK) propose a generic similarity search construction based on locality sensitive hashing (LSH) and Bloom filter (BF) (cf. Adjedj et al. [10] in Section 2.3.1 and Bringer et al. [55] in Section 2.4.1). The *idea* for their keyword search scheme is to represent keywords as n-grams and insert each n-gram into the BF using LSH. To measure the distance for the similarity search, the Jaccard distance is used. The protocol is *interactive* and requires *two rounds* of communication to retrieve the matching documents.

EFFICIENCY: *Index generation* requires a metric space translation for each distinct keyword per document, b LSH functions per keyword and two encryptions per BF bucket. To *search*, the server has to search for b buckets in the first round. The client decrypts the search result and sends some document identifiers to the server. The server replies with the encrypted documents.

SECURITY: KIK adapt the IND-CKA2 security definition of Curtmola et al. [75] (cf. Section 2.3.1) to their setting (allow the leakage of the similarity pattern) and prove their scheme *IND-CKA2* secure under the adapted definition.

Synthesis

The S/S architecture has been the subject of active research for over a decade now and still new schemes are developed. Most of the schemes focus on single

and conjunctive keyword searches, but also more powerful queries are possible. The schemes, that try to achieve a higher level of security or a better query expressiveness are likely to be more complex or use more expensive primitives and are thus less efficient.

In the beginning of SE research with the S/S architecture, there were no formal security definitions for searchable encryption. It took several years until the first definitions were available and still researchers do not use a common security model to prove their schemes secure. Some schemes are based on new assumptions and not proven secure under standard or well known assumptions, which makes it hard to assess the security of a scheme and compare it to others. Also, some authors allow the leakage of the search pattern in their schemes, whereas others want to hide as much information as possible.

26 out of 27 schemes in the S/S setting leak at least the access pattern and the search pattern. Only SSW protects the search pattern by calculating the dot product of the trapdoor and the searchable content. Thus the schemes do not leak which of the keywords match the query, but the search complexity is linear in the number of keywords.

All but eight papers (cf. Table 2.1) propose schemes which achieve at best a search complexity of $O(n)$ which is linear in the number of documents stored in the database. The eight exceptions (cf. gray search fields in Table 2.1) introduce schemes, which achieve sub-linear search times. The schemes achieve at least a search complexity logarithmic in the total number of keywords in the database, since the search consists of a standard database search which can be realized using a binary or hash tree (LSD^+). Some schemes (CGK^+, CK, KPR, KP, CJJ^+) even achieve optimal search time, i. e., the number of documents that contain the query keyword. These schemes require deterministic trapdoors which inherently leak the search pattern, since the server can directly determine whether two searches use the same predicate. Another drawback of some of these schemes is interactiveness, either in the database update (CKG^+) or in the update, search, and encrypt phase (LSD^+). This is due to the fact, that the database consists of an index per keyword (inverted index) instead of an index per document (forward index). The schemes achieve the best search complexity, but since the update operation is expensive, they are best suited for static databases. The implementation of the CJJ^+ scheme is the most scalable, but uses an interactive search protocol.

Table 2.1 gives a detailed overview of the computational complexity and the security of the different algorithms of the discussed schemes. The digest of the table can be found in the reading guidelines in Section 2.1.5 and the legend in Table 2.2.

2.3.2 *Single Writer/Multi Reader (S/M)*

In a single writer/multi reader (S/M) scheme the secret key owner is allowed to create searchable content, whereas a user-defined group is allowed to generate trapdoors.

For historical reasons, we start this section with a non-proven seminal scheme which is worth mentioning. The discussed schemes in this section allow only single equality test queries.

Table 2.1: Comparison of different S/S schemes. The legend is in Table 2.2.

	Efficiency of algorithms			Security			
Scheme	Encrypt	Trapdoor	Search	Def.	Assumption	ROM	Notes
Single Keyword Equality Test							
SWP [171]	$\nu n E$	kE	$\nu n E$	IND-CPA	SP*	-	sequential scan
Goh [92]	$\nu' n f$	kf	$k n f$	IND1-CKA	SP	-	BF index
CM-I [64]	$n(\nu' + \nu'')f$	$2kf$	nf	IND2-CKA	SP	-	dictionary on client
CM-II [64]	$n(\nu' + 2\nu'')f$	$2kf$	nf	IND2-CKA	SP	-	trapdoor interactive
CGK$^+$-I [75]	$3\nu'' n f + n D''(\nu) E$	$2kf$	$D(\nu)D$	IND-CKA1	SP	-	optimal search time, static $\mathcal{D}$
CGK$^+$-II [75]	$n\nu'' D''(\nu) f$	$kD''(\nu)f$	$D''(\nu)TLU$	IND-CKA2	SP	-	static $\mathcal{D}$
ABO-I [12]	$n\nu(h + E)$	$k(h + E)$	ks	deterministic	SP	-	deterministic macs
ABO-II [12]	$n\nu(h + E)$	$k(h + E)$	ks	deterministic	SP	-	deterministic macs
LSD$^+$-I [177]	$n\nu' f + n D(\nu)(D + E)$	kf	$k(D + f)$	IND-CKA2	SP	-	interactive, dynamic $\mathcal{D}$
LSD$^+$-II [177]	$n\nu'(H + E)$	$k(f + H)$	$D(\nu)(h + D)$	IND-CKA2	SP	-	dynamic $\mathcal{D}$
CK [65]	$2\nu'' f + D(\nu)E$	$2kf$	$D(\nu)D$	IND-CKA2^t	SP	-	static $\mathcal{D}$
KO [119]	$2n\nu'' f$	nkf	n TLU	UC-CKA2	SP	-	secure against malicious adv.
KPR [109]	$8n\nu'' f$	$3kf$	$D(\nu)D$	IND-CKA2^t	SP	✓	dynamic $\mathcal{D}$
KP [108]	$(2n - 1)\nu'(3f + E)$	$2kf$	$D(\nu) \log nD$	IND-CKA2^t	SP	✓	dynamic $\mathcal{D}$, parallel

Continued on next page.

Table 2.1: Comparison of different S/S schemes (cont.). The legend is in Table 2.2.

Scheme	Efficiency of algorithms: Encrypt	Trapdoor	Search	Security: Def.	Assumption	ROM	Notes
Conjunctive Keyword Equality Test							
GSW-I [97]	$n(v+1)e$	$ne+kf$	$2ne$	IND1-CKA^t	DDH	✓	split trapdoor
GSW-II [97]	$n(2v+1)e$	$3e+kf$	$n(2k+1)e$	IND1-CKA^t	NS**	-	
BKM [19]	$v'nf$	ki	ni	IND1-CKA^t	SP	-	Shamir's Secret Sharing
BLL [57]	vnp_p^s	$3ke$	$2np_p^s$	IND1-CKA^t	BDH	✓	
RT [161]	$n(v+1)e+np_p^a$	$(m+1)e$	$2np_p^a$	IND1-CKA^t	coXDH	✓	smaller trapdoors
WWP-III [182]	$v'ne$	$2kv'e$	$v'np_p^s$	IND1-CKA^t	DL, 1-DDHI	-	no keyword fields
CJJ$^+$ [63]	$v''D(v)(6f+E+e)$	$D(v)(k-1)e$	$D(v)(k-1)e$	IND-CKA2^t	DDH	-	interactive search (client D)
Single Fuzzy Keyword Test							
PKL$^+$-I [151]	$lvne$	klf	$2lvn(e+H)$	IND1-CKA^t	DDH	✓	character-wise encryption
PKL$^+$-II [151]	$2lvnf$	mlf	$mlf+nmH$	IND1-CKA^t	DDH	✓	character-wise encryption
ABC$^+$ [10]	$nv'b(h+f+E)$	$k(bh+D''(v)f)$	$bD''(v)s$	IND-CKA2	SP	-	LSH, BFS
LWW$^+$ [125]	$n\|S\|f$	$k\|S\|f$	$n\|S\|c$	IND-CKA1^t	SP	-	pre-computed sets, interactive
KIK [120]	$nv'b(h+f+E)$	$kb(h+f)$	bs	IND-CKA2^t	SP	-	LSH, BF, interactive
Keyword Search based on Inner Product							
SSW [167]	$n(6v+2)e$	$8ve$	$n(2v+2)p_{c4}^s$	FS	C3DH, DLIN	-	hides SearchPattern

gray — sub-linear search time (optimal include $D(v)$)

* secure primitives - scheme is secure, if the underlying primitives exist/are secure (generic construction)

** new non-standard hardness assumption

t security definition conceptually as the one stated, but tailored to a specific setting (see respective Section)

Table 2.2: Legend for S/S schemes.

Amount		Primitive	
n	number of documents	p_p^s	symmetric prime order pairing
v	number of keywords per document	p_p^a	asymmetric prime order pairing
v'	number of distinct keywords per document	$p_{c^4}^s$	composite order pairing of degree 4
v''	number of distinct keywords in the database	e	exponentiation
k	number of keywords per trapdoor	f	pseudo-random function, permutation
l	length of keyword in characters	h	hash function, mac
m	number of keyword fields	H	Hash chain
$D(v)$	number of documents containing word v	H	Hamming distance
$D''(v)$	max. number of documents containing word v	m	polynomial multiplication
d	Edit distance	i	polynomial interpolation
$\|S\|$	$\sum_{y=0}^{d}\sum_{x=l}^{l+y}\binom{x}{y}$ (size of set)	E, D	Encryption, Decryption
b	number of LSH functions	s, c	search, comparison
		TLU	table look-up

Single Equality Test

Exact keyword match for a single search keyword.

WORTH MENTIONING. The following scheme does not fit in the structure of the survey by means of our selection criteria, since the authors do not provide a security proof. Nevertheless the idea of the authors is worth mentioning.

Using Bloom filter with group ciphers. Bellovin and Cheswick [30] (BC) present a multi-user scheme based on encrypted Bloom filters and *group ciphers* such as Pohlig-Hellman encryption. They introduce a semi-trusted *third party* which is able to cryptographically transform an encrypted search query for a users database to a query for another users database, without leaking the query to neither the third party nor the database owner. BC, like Goh, uses one Bloom filter per document. Instead of hash functions, a group cipher is used where operations can be done on encrypted data. Due to the use of a Bloom filter per document, BC allows *false positives*.

USING BROADCAST ENCRYPTION WITH SINGLE-USER SSE. Curtmola et al. [75] define SSE in a multi-user setting, where only the data owner is able to write to the document collection, but an arbitrary group of users is allowed to query the data. They propose a *general construction*, where the *idea* is to use *broadcast encryption* (BE) [80] on top of a single-user scheme. BE allows the data owner to distribute the secret key that is used for the SE scheme to a group of users. This allows all users in possession of the key to create trapdoors and thus to search. As an example they use their single-user SSE scheme as described in Section 2.3.1.

EFFICIENCY: The efficiency depends on the underlying SE scheme.

SECURITY: The security depends on the underlying SE scheme. Curtmola et al. provide a proof, that the new multi-user scheme achieves revocation, i. e., revoked users are no longer able to perform searches.

USING RE-ROUTABLE ENCRYPTION. The *idea* of Raykova et al. [157] (RVB$^+$) is to introduce re-routable encryption that allows to transform encryptions under different keys without leaking the encrypted message. They use another entity *(third party)* a so called query router which protects the identity of the clients and checks their authorization on behalf of the server. Thus, their scheme allows to search other users data anonymously.

A client can submit an encrypted query to the query router, who checks the authorisation of the user. If the user is in the group of authorised users, the query router transforms the query and forwards it to the server. The server sends back the search results to the query router, which transforms the results and forwards them to the user. Due to the use of a Bloom filter per document, RVB$^+$ allows for *false positives*.

EFFICIENCY: To achieve more efficiency (*sub-linear* in the size of the data) RVB$^+$ sacrifice the strict definitions of security and privacy by using private key deterministic encryption and a Bloom filter index per document. The *index generation* algorithm has to create a Bloom filter per document. This takes time linear in the number of distinct keywords per document. The trapdoor generation is a single encryption of the search

word for the client and a transformation step for the query router. The *search* operation is a Bloom filter lookup per document.

SECURITY: RVB$^+$ is the second discussed scheme that uses deterministic encryption. RVB$^+$ define DET-CCA security, following the idea of Bellare et al. [29] (Section 2.4.2). The construction is *DET-CCA* secure in the *RO* model under the *DL* hardness assumption. The system *leaks* the search pattern to the query router. The security is based on a trust assumption, which is achieved by splitting the server into several parties.

SEE ALSO: The idea of using deterministic encryption as a trade-off between security and efficiency was first introduced by Bellare et al. [29] who defined deterministic encryption in the public key setting (see Section 2.4.2) and by Amanatidis et al. [12] in the symmetric key setting (Section 2.3.1).

USING BILINEAR MAPS. Yang et al. [186] (YLW) propose a new scheme, which is an adaptation of the M/M scheme from earlier work by Yang et al. [185] which is discussed in Section 2.4.2. In YLW, each authorized user has a distinct query key which allows easy user revocation and accountability. Revoked users lose all their search privileges, also on old data. The search algorithm uses symmetric bilinear maps of prime order. The *idea* is, that with the bilinear map, the users trapdoor (which includes the distinct user key), and a users helper key, the server can calculate a common key to search the index.

YLW requires a *secure channel* to send the query result back to the querying user, since all users share a single record encryption key, which allows also revoked users to decrypt the search result. The authors suggest to use a public key encryption to decrypt the results. The authors also present straightforward *extensions* for conjunctive and wildcard searches.

EFFICIENCY: *Encryption* requires the client to compute a symmetric bilinear map of prime order per distinct keyword per document. The *search* algorithm needs to perform one pairing operation per search.

SECURITY: YLW extend the IND-CKA2 security definition to the multi-user setting. Search patterns leak per user, such that queries from different users are unlinkable. YLW is proven secure in the *RO* model under the *DDH* and the computational Diffie-Hellman *(CDH)* [77] assumptions in their extended *IND-CKA2* definition.

Synthesis

The S/M architecture has not received a lot of research attention, yet. Curtmola et al. [75] proposed a generic combination of broadcast encryption and any S/S scheme. Recently, two provably secure schemes were proposed. Both schemes support only single keyword equality tests and are an example for the trade-off: security vs. efficiency. The more secure a scheme is, the more complex it gets and is thus less efficient. The search algorithm of Raykova et al. [157] is linear in the number of documents, but the scheme uses deterministic encryption and directly leaks the search pattern in addition to the access pattern. Yang et al. [186] achieve a higher level of security, but the search is linear in the number of keywords per document. The schemes in this setting usually introduce a TTP for user authentication or re-encryption of the trapdoors.

Table 2.3 gives a detailed overview of the computational complexity and the security of the different algorithms of the discussed schemes. The digest of the table can be found in the reading guidelines in Section 2.1.5 and the legend in Table 2.4.

2.4 MULTI WRITER SCHEMES (M/∗)

This section deals with the M/S and M/M schemes.

2.4.1 *Multiple Writer/Single Reader (M/S)*

Most of the schemes in this section are variants of PEKS. The main scenarios for PEKS like schemes are: retrieving emails or documents from a server and allowing a server to redirect/route emails. Usually, multiple users (in possession of the public key) can generate searchable ciphertexts, which can be searched by the private key holder.

Single Equality Test

With an equality test we mean an exact keyword match for a single search keyword.

PEKS - PUBLIC KEY ENCRYPTION WITH KEYWORD SEARCH. Boneh et al. [47] (BCO$^+$) propose the first searchable encryption scheme using a public key system. The *idea* for their PEKS scheme is to use identity based encryption (IBE) in which the keyword acts as the identity. Due to the use of a PKE, each user in BCO$^+$ is allowed to create searchable content with the recipients public key. Only the private key holder is able to generate a trapdoor to search inside the encrypted data. The construction is based on Boneh and Franklin's work on IBE [41, 42].

Details: To create a searchable ciphertext, the sender encrypts his message with a standard public key system and appends the PEKS of each keyword (i. e., a publicly known string encrypted under the public key associated with the keyword as identity) (cf. Figure 2.8). The sender then sends the following ciphertext:

$$E_{K_{pub}}(M)||C_1 = \mathsf{PEKS}(K_{pub}, w_1)||\ldots||C_m = \mathsf{PEKS}(K_{pub}, w_m).$$

To search, the receiver uses the master secret key to derive a secret key for a specific keyword it wants to search for (i. e., the keyword is the identity used for the secret key). The resulting secret key is used as the trapdoor and sent to the server (e. g., email server). The server tries to decrypt all the IBE ciphertexts. If the decryption is successful (i. e., results in the publicly known string) the attached encrypted message contains the keyword. The detailed algorithm is shown in Figure 2.7.

EFFICIENCY: BCO$^+$ uses symmetric prime order pairings. The *encryption* requires the server to perform one pairing computation, two exponentiations, and two hashes per keyword. The *search* complexity is linear (one map, one hash) in the number of keywords per document.

SECURITY: BCO$^+$ is proven *PK-CKA2* secure in the *RO* model under the *BDH* assumption. BCO$^+$ requires a *secure channel* to transmit the trapdoors,

Table 2.3: Comparison of different S/M schemes. The legend is in Table 2.4.

	Efficiency			Security			
Scheme	Encrypt	Trapdoor	Search	Def.	Ass.	ROM	Notes
Single Keyword Equality Test							
CGK$^+$ [75]	generic construction: dependent on the underlying SSE and BE schemes						BE
RVB$^+$ [157]	$\nu' n E$	$k E$	$n B$	deterministic	DL	✓	FP
YLW [186]	$\nu' n p_p^s$	$k e$	$1 p_p^s + n \nu' h$	IND-CKA2^t	DDH, CDH	✓	

Table 2.4: Legend for S/M schemes.

Amount		Primitive	
n	number of documents	p_p^s	symmetric prime order pairing
ν'	number of distinct keywords per document	e	exponentiation
k	number of keywords per trapdoor	h	hash function
		E	Encryption
		B	Bloom filter look-up

The size p of G_1, G_2 is determined by the security parameter. The scheme requires two hash functions $H_1 : \{0,1\}^* \to G_1$ and $H_2 : G_2 \to \{0,1\}^{\log p}$ and a bilinear map $e : G_1 \times G_1 \to G_2$.

- Keygen: Pick a random $\alpha \in \mathbb{Z}_p^*$ and a generator g of G_1. It outputs $K_{pub} = (g, h = g^\alpha)$ and $K_{priv} = \alpha$
- PEKS(K_{pub}, w):
  1. Compute $t = e(H_1(w), h^r) \in G_2$ for a random $r \in \mathbb{Z}_p^*$.
  2. Output $C = [g^r, H_2(t)]$.
- Trapdoor(K_{priv}, w): Output $T_w = H_1(w)^\alpha \in G_1$.
- Search(A_{pub}, C, T_w): let $C = [A, B]$. Test if $H_2(e(T_w, A)) = B$, which is equal to testing if $H_2(e(H_1(w)^\alpha, g^r)) = H_2(e(H_1(w), g^{\alpha r}))$. If so, output 'yes'; if not, output 'no'.

Figure 2.7: Public key encryption with keyword search (PEKS) [47] (Section 2.4.1).

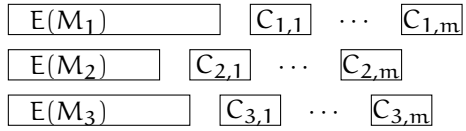


Figure 2.8: PEKS. The ciphertexts $C_{i,j}$ use the keywords as identity for the IBE system and are then appended to the encrypted message $E(M_i)$ (Section 2.4.1).

so that an eavesdropper cannot get hold of a trapdoor. The trapdoors for a keyword are never refreshed. The scheme is vulnerable to an off-line keyword guessing attack [58, 188], as explained in Section 2.4.1. In the current model, the server is able to store a trapdoor and use it for future documents, which means that the current PEKS is a one-time system.

SEE ALSO: Baek et al. [17] (Section 2.4.1) address the problem of the secure channel and the trapdoor refreshing. Abdalla et al. [8] (Section 2.4.1) formally define (A)IBE and present generic transformations from (H/A)IBE to PEKS.

TEMPORARY KEYWORD SEARCH (PETKS). Abdalla et al. [8] (ABC^{++}) formalize anonymous IBE (AIBE) and present a generic SE construction by transforming an AIBE scheme into a searchable encryption scheme. The idea underlying the aibe-2-peks transformation was first given by Boneh et al. [47] (Section 2.4.1). ABC^{++} also give a hierarchical IBE (HIBE) transformation (hibe-2-petks), which allows to transform a HIBE scheme into a PETKS. The *idea* behind PETKS is to generate a trapdoor which is only valid in a specific time interval. With the time interval included in the trapdoor, the server cannot use the trapdoors to search in past or future ciphertexts (outside the time interval).

EFFICIENCY: The efficiency depends on the used HIBE scheme.

SECURITY: The new PETKS scheme that results from the hibe-2-petks transformation is *PK-CKA2* secure in the *RO* model if the HIBE scheme is IND-CPA secure.

SEE ALSO: Boneh et al. [47] (Section 2.4.1)

COMBINING PKE AND PEKS IN A SECURE WAY. Baek et al. [16] (BSS-I) discuss the problems that arise from combining public key encryption (PKE) schemes with PEKS. They give a concrete construction where the *idea* is to combine a variation of the ElGamal cryptosystem [79], PEKS, and the randomness re-use technique of Kurosawa [117]. The authors also give a generic PKE/PEKS construction. We discuss only their ElGamal construction which is proven secure in the paper. The authors give two extensions to their scheme to the multi-receiver setting and the multi-keyword setting.

EFFICIENCY: BSS-I uses symmetric bilinear maps of prime order. The *encryption* algorithm needs to perform three exponentiations and one mapping per keyword per document. The *search* algorithm requires one bilinear map per keyword per document.

SECURITY: BSS-I is proven *PK-CKA2* secure in the *RO* model assuming that the *CDH* problem is intractable.

PEKS BASED ON JACOBI SYMBOLS. Crescenzo and Saraswat [73] (CS) present the first PEKS scheme that is not based on bilinear maps, but on Jacobi symbols. Their *idea* is to use a transformation of Cocks' identity based encryption scheme [71] which is based on the quadratic residuosity problem.

EFFICIENCY: To *encrypt* the data, $4k$ Jacobi symbols per keyword have to be calculated, where k (the length of a keyword in bit) is a scheme's parameter to guarantee the consistency. The authors choose $k = 160$ as an example. The *search* algorithm is linear ($4k$) in the number of ciphertexts. The storage and communication complexity is high.

SECURITY: The security of CS is based on a variant of the well-known quadratic residuosity problem, namely the quadratic indistinguishability problem *(QIP)*. CS is proven *PK-CKA2* secure in the *RO* model.

K-RESILIENT PEKS (KR-PEKS). Khader [113] constructs a scheme based on k-resilient IBE [101, 102]. The *idea* is to use the ability of constructing a PEKS scheme from an IBE. Khader also gives a construction for *multiple keywords* and a *secure-channel-free* PEKS scheme. The main goal of the work was to construct a PEKS scheme that is secure in the standard model.

EFFICIENCY: *Encryption* requires $5 + 3\nu$ exponentiations, where ν is the number of keywords per document. To *search*, 4 exponentiations have to be calculated per keyword, per ciphertext.

SECURITY: Her scheme is proven *PK-CKA2* secure in the *ST* model under the *DDH* assumption.

SECURE CHANNEL FREE PEKS. Baek et al. [17] (BSS-II) remove the need for a secure channel for transmitting the trapdoors in the original PEKS [47] scheme. The *idea* is to add a server public/private key pair to PEKS and use the aggregation technique from Boneh et al. [45]. By adding a server key pair, only the server chosen by the sender *(designated tester)* is able to search. The authors also address the problem of refreshing the trapdoors for the same keyword from PEKS.

EFFICIENCY: BSS-II uses symmetric prime order pairings. *Index generation* requires two pairings and an exponentiation per keyword per document.

To *search*, the server has to compute one pairing per keyword per ciphertext.

SECURITY: BSS-II is proven *PK-CKA2* secure in the *RO* model under the *BDH* assumption.

PEKS WITH DESIGNATED TESTER. Rhee et al. [158] (RPS$^+$-I) enhance the security model of the PEKS construction of Baek et al. [17] (cf. Section 2.4.1) and construct a PEKS scheme which is secure in the enhanced model. The *idea* is to use a new public key structure, where the public key consists of three components. Their enhanced security model allows an attacker to obtain the relation between ciphertexts and a trapdoor. In addition, the attacker publishes only the public key and not the secret key as in Baek et al.'s security model. RPS$^+$-I is proven secure in their enhanced model.

EFFICIENCY: RPS$^+$-I uses symmetric prime order groups. *Encryption* requires initially seven pairings and then one pairing operation and two exponentiations per keyword. To *search*, the server has to perform one pairing and one exponentiation per keyword per document.

SECURITY: RPS$^+$-I is proven *PK-CKA2* secure in the *RO* model under the *BDH* assumption and the bilinear Diffie-Hellman inversion *(1-BDHI)* assumption [40, 133].

SEE ALSO: This is an improved version of Baek et al. [17] (Section 2.4.1)

OUTSOURCE PARTIAL DECRYPTION. Liu et al. [129] (LWW) propose a new scheme where the *idea* is to outsource parts of the decryption process to the service provider and thus reduce the computational decryption overhead of the user.

EFFICIENCY: LWW uses symmetric prime order pairings. *Index generation* requires one pairing and one exponentiation per keyword. To *search*, the server has to compute two pairings.

SECURITY: LWW is proven *PK-CKA2* secure in the *RO* model under the *BDH* assumption.

REGISTERED KEYWORD SEARCH (PERKS). Tang and Chen [175] (TC) propose the concept of public key encryption with registered keyword search (PERKS). The *idea* is to allow a writer to build searchable content only for the keywords that were previously registered by the reader. This makes TC more robust against an off-line keyword guessing attack.

EFFICIENCY: TC uses symmetric prime order pairings. The *encryption* requires two exponentiations and one mapping per distinct keyword. To *search*, the server has to compute one pairing per distinct keyword per index. *Keyword registration* requires to compute one hash value.

SECURITY: TC is proven *PK-CKA2* secure in the *RO* model under the *BDH* assumption.

COMBINING PEKS WITH PKE (PEKS/PKE). Zhang and Imai's [191] (ZI) *idea* is to use a hybrid model to combine PKE and PEKS into a single scheme which uses the same key pair for both primitives. The authors give a generic construction and a concrete instantiation using an anonymous IBE by Gentry [87] and the tag-KEM/DEM (key/data encapsulation mechanism) by Kurosawa-Desmedt [118].

EFFICIENCY: The instantiation of ZI uses symmetric bilinear groups of prime order. The *encryption* requires two pairings and eight exponentiations per keyword and the *search* one pairing and one exponentiation per keyword.

SECURITY: ZI is proven *PK-CKA2/CCA* secure *without ROs* under the assumption that the *Kurosawa-Desmedt tag-KEM/DEM is secure* and the *Gentry IBE is anonymous*. Stand-alone PEKS/PKE may lose data privacy (CCA) [16].

SEE ALSO: KEM/DEM [170], Tag-KEM/DEM [9].

TRAPDOOR SECURITY IN PEKS WITH DESIGNATED TESTER. Rhee et al. [159] (RPS$^+$-II) propose a scheme that is secure against keyword-guessing attacks (only for outside attackers). The *idea* is to make the trapdoors indistinguishable by introducing a random variable in the trapdoor computation.

EFFICIENCY: RPS$^+$-II uses symmetric prime order groups. The *encryption* requires two exponentiations and one pairings per keyword. To *search*, the server has to perform one pairing and two exponentiation per keyword per document.

SECURITY: RPS$^+$-II is proven *PK-CKA2* secure in the *RO* model under the *BDH* assumption and the *1-BDHI* assumption.

DELEGATED SEARCH (PKEDS). Ibraimi et al. [105] (INH$^+$) give a construction for a public key encryption with delegated search (PKEDS) which is an extension of ElGamal [79]. The *idea* of INH$^+$ is to allow the server to search each part of the encrypted data, in contrast to previous schemes where only the metadata is searchable. This can be used for example to let a server scan messages for malware. INH$^+$ allows two different kinds of trapdoors. One allows to search for a keyword inside a trapdoor and the other allows the server to search directly for a keyword.

EFFICIENCY: INH$^+$ uses bilinear groups of prime order. *Encryption* is the same as ElGamal and requires two exponentiations per keyword. Delegation requires five exponentiations. The trapdoor generation requires two asymmetric pairings and two exponentiations per keyword. A *search* for a keyword inside a trapdoor, requires three asymmetric pairings and three exponentiations per keyword per ciphertext. To *search* for a keyword, the server has to perform three asymmetric pairings and two exponentiations per keyword per ciphertext.

SECURITY: INH$^+$ is proven to be ciphertext and trapdoor indistinguishable (i. e., an adversary (except the server) cannot learn any information about the plaintext keyword) under the symmetric external Diffie-Hellman *(SXDH)* [18] assumption. INH$^+$ achieves ciphertext one-wayness under the modified CDH *(mCDH)* assumption which is a stronger variant of the CDH assumption. The mCDH assumption is implied in the BDH problem in Type 3 pairings (BDH-3) [66]. INH$^+$ is proven secure in the *ST* model. The security model is *weaker* than PEKS, since the server can generate any trapdoor.

Conjunctive Equality Search

See Section 2.3.1 for information on conjunctive keyword searches.

PECKS - PUBLIC KEY ENCRYPTION WITH CONJUNCTIVE FIELD KEYWORD SEARCH. Park et al. [149] (PKL) study the problem of public key encryption with conjunctive field keyword search (PECKS). The *idea* is to extend PEKS to allow conjunctive keyword searches (CKS) in the public key setting. PKL present the first two constructions PKL-I and PKL-II with constant trapdoor size that allow CKS.

EFFICIENCY: Both schemes use symmetric prime order pairings. PKL-I requires the user to perform one pairing computation per distinct keyword for *encryption*. To *search* the server has to perform one pairing operation per ciphertext.

In PKL-II a user has to store private keys in proportion to the number of keyword fields. *Encryption* needs one exponentiation per document and the *search* requires two pairings per ciphertext.

SECURITY: PKL adapt the extended version of IND1-CKA from GSW for conjunctive queries to the public key setting, by removing encryption oracle queries (since any user can generate trapdoors with help of the public key). Their adapted definition is basically PK-CKA2. The security of PKL-I is based on the *BDH* assumption. PKL-II is based on the *BDHI* assumptions. Both schemes are proven secure in the *RO* model in their adapted *PK-CKA2* definition. **Remark:** The proofs do not satisfy their model and PKL-I is broken by Hwang and Lee [104], who also showed, that the proof of PKL-II is incomplete.

MORE SECURE SEARCHABLE KEYWORD BASED ENCRYPTION. Park et al. [150] (PCL) propose a new mechanism which is more secure than previous schemes in certain applications like email gateways. The *idea* is to construct a scheme from PECKS (Section 2.4.1) by using a hybrid encryption technique. A user can either create a *decrypt trapdoor* or a *search trapdoor* for specific keywords. The enhanced security is achieved by introducing the decrypt trapdoor, which can decrypt ciphertexts without the need for the user's private decryption key. In case of email routing, each device could have a different decrypt trapdoor for certain keywords. Thus, the user's private decryption key does not need to be on each device which makes the scheme more secure against key compromise. The search trapdoor can test whether a ciphertext contains all of the keywords. PCL requires non-empty *keyword-fields*.

EFFICIENCY: *Encryption* requires two exponentiations per document. PCL requires two symmetric prime order pairing operations per ciphertext to *search*.

SECURITY: PCL adapt the PK-CKA2 security definition to PK-CCA2 (public key - adaptive chosen ciphertext attack) by allowing an adversary to query an decryption oracle next to the normally allowed trapdoor queries. The security of PCL is based on the q-BDHI assumption and the bilinear collusion attack (q-BCA) assumption [68]. The q-BCA assumption is equivalent to the $(q+1)$-BDHI assumption [68]. PCL is proven secure in their tailored *PK-CCA2* definition under the $(q+1)$-*BDHI* assumption in the *RO* model.

PECKS WITH SHORTEST CIPHERTEXT AND PRIVATE KEY. Hwang and Lee [104] (HL) propose a public key encryption with conjunctive keyword search (PECK) and introduce a new concept called multi-user PECKS

(mPECKS) as described in Section 2.4.2. The *idea* of HL is to minimize the communication and storage overhead for the server and also for the user. Hwang and Lee compare the efficiency of their scheme with both PKL schemes [149] (cf. Section 2.4.1).

EFFICIENCY: *Index generation* requires $2 + 2v$ exponentiations, where v is the number of keywords per document. PECK uses three symmetric bilinear maps of prime order per ciphertext to *search*. HL has the *shortest* ciphertext size compared with previous PECKS schemes and requires only *one private key*.

SECURITY: HL prove their scheme secure in the adapted *PK-CKA2* definition from PKL (cf. Section 2.4.1) under the *DLIN* assumption in the *RO* model.

Extended Queries

CONJUNCTIVE, SUBSET, AND RANGE QUERIES. Boneh and Waters [43] (BW) develop a PEKS scheme for conjunctive keyword searches from a generalization of AIBE. The *idea* is to use *hidden vector encryption* (HVE) [53, 169] for searching in encrypted data. BW supports equality, comparison, general subset queries, and arbitrary conjunctions of those. BW also present a general framework for analyzing and constructing SE schemes.

EFFICIENCY: *Encryption* requires $5k + 3$ exponentiations per keyword, per document, where k is the number of characters per keyword. For an equality *search*, the server has to perform $2k - w + 1$ symmetric composite order bilinear pairing operations, where k is the number of characters of the searchable keywords and w the number of wildcard characters in the keyword. The trapdoor size is linear in the number of conjunctive keywords. The ciphertext size is relatively large, due to the use of composite order bilinear groups [49].

SECURITY: BW is proven *SEL-CKA* secure under the *C3DH* assumption and the *BDH* assumption in the *selective model* (SEL) [61], where an adversary commits to an encryption vector at the beginning of the security game. A security advantage of BW is that it does *not leak* the attribute values upon decryption like other schemes.

MULTI-DIMENSIONAL RANGE QUERIES (MRQED). Shi et al. [169] (SBC$^+$) propose a scheme that can create trapdoors for a conjunction of range queries over multiple attributes. The *idea* is, that each tuple that should be encrypted, can be represented as a point in a multi-dimensional space. Then, a multi-dimensional range query is equivalent to testing whether a point falls inside a hyper-rectangle. To represent ranges, the authors use binary interval trees over integers and use one interval tree per dimension.

EFFICIENCY: SBC$^+$ can be constructed using either asymmetric or symmetric bilinear maps of prime order. *Encryption* requires $8DL + 2$ exponentiations, where D is the number of dimensions and L the depth of a node in the corresponding tree. The *search* algorithm requires the server to compute $5D$ pairing operations per ciphertext.

SECURITY: SBC$^+$ is proven *SEL-CKA* secure under the *BDH* assumption and the *DLIN* assumption in the SEL model. SBC$^+$ *leaks* the attribute values

after successful decryption. The authors argue, that this is acceptable for the application of encrypted network audit logs.

ERROR-TOLERANT SEARCHABLE ENCRYPTION. The *idea* of Bringer et al. [55] (BCK) is to use locality-sensitive hashing (LSH) to enable error-tolerant queries. A LSH function outputs the same hash values for similar items, where similarity is measured in the Hamming distance. BCK inserts these LSH values into one Bloom filter with storage [50] (BFS) in encrypted form. If two keywords k, k' are close enough, the LSH outputs the same hash values as input for the BFS, thus allowing error-tolerant queries. The search in BCK is *interactive*. To query the BFS, the scheme uses a PIR protocol to retrieve the encrypted BF positions. The client decrypts all positions and computes the intersection. The result is a set of file identifiers which can be retrieved in a second round.

EFFICIENCY: *Encryption* includes two sets of hash functions (LSH + BFS) and semantically secure PKE per keyword per document. Each modified BFS position will be updated with a private information storage (PIS) protocol. To *search*, a PIR protocol is run to retrieve the content of the BFS positions, which need to be decrypted to obtain the document ids. Document retrieval requires another round of communication.

SECURITY: BCK is proven *PK-CKA2* secure. BCK hides the search pattern using PIR.

SEE ALSO: Adjedj et al. [10] (Section 2.3.1).

WILDCARD PEKS. The *idea* of Sedghi et al. [164] (SLN$^+$) is to construct a new scheme based on HVE which can be used for wildcard searchable encryption. SLN$^+$ allows wildcard searches over any alphabet, in contrast to previous schemes [36, 106, 136] that work only over binary symbols.

EFFICIENCY: SLN$^+$ uses symmetric bilinear pairings of prime order. *Encryption* requires $(N+1)(l+1)+4$ exponentiations per keyword, where N is an upper bound on the number of wildcard symbols in decryption vectors and l the length of the keyword. The *search* requires three bilinear maps and w exponentiations per keyword, where w is the number of wildcard characters.

While in previous works the size of the decryption key and the computational complexity for decryption is linear in the number of non-wildcard symbols, in SLN$^+$ these are constant.

SECURITY: SLN$^+$ is proven *SEL-CKA* secure under the *DLIN* assumption in the SEL model.

Synthesis

Since 2004, research in the M/S architecture has obtained significant attention and is still an active research direction. Like in the S/S architecture, most schemes focus on single and conjunctive keyword searches, but also more powerful queries are possible.

Since there is a wide spectrum of different public key encryption techniques, PEKS schemes can be realized using different primitives, such as IBE, first used by BCO$^+$, AIBE and HIBE used by ABC^{++} or HVE used by BW and SLN$^+$ in the context of SE.

The M/S architecture is a good example for the aforementioned trade-offs, namely, expressiveness vs. efficiency and security vs. efficiency. The M/S architecture focuses mainly on theoretical research which tries to achieve a certain level of security or query expressiveness and is not so much focused on efficiency. All but four (16/20) schemes make heavy use of pairing operations (at least for the search algorithm). Most schemes use at least one pairing per document in the search algorithm and some schemes even use one pairing per keyword per document which is inefficient in practice. Only four schemes (BBO, CS, Khader and BCK) do not use pairings. The search complexity of all (except one) schemes in this section is at best linear in the number of documents stored on the server. The exception (BBO) uses deterministic encryption and achieves sub-linear (logarithmic) search time. If the data is from a small space (low min-entropy), e. g., well known keywords, using deterministic public key encryption is vulnerable to brute force attacks and thus considered insecure for practical purposes.

Seven of the 20 schemes are proven secure in the standard model, whereas 13 schemes are proven secure with random oracles. All of the schemes leak the search pattern and the access pattern. The search pattern is either leaked directly by using a deterministic procedure to generate the trapdoors, or indirectly by an off-line keyword guessing attack as follows.

OFF-LINE KEYWORD GUESSING ATTACK. A problem of the main PEKS concept is, that there is no keyword/predicate privacy. Most PEKS schemes are vulnerable to an off-line keyword guessing attack, which allows an attacker to recover the predicate from a trapdoor. The leakage of the access pattern makes this attack possible. The attack is based on the fact that i) the keyword space is small (and users choose well-known words to search their documents) and ii) the encryption key is public. The attack works as follows:

1. The attacker captures a valid trapdoor T_w.
2. With the user's public key and an appropriate chosen keyword w', the attacker runs the Encrypt algorithm to get a searchable ciphertext.
3. The user's public key, the captured trapdoor and the ciphertext from (2) are then used to check whether the ciphertext satisfies the trapdoor or not. If so, the chosen keyword is a valid keyword. Otherwise the attacker continues with (2).

This allows an attacker to recover the keyword inside a trapdoor. Thus, there is no keyword privacy in the M/S architecture, when using a PKE.

Table 2.5 gives a detailed overview of the computational complexity and the security of the different algorithms of the discussed schemes. The digest of the table can be found in the reading guidelines in Section 2.1.5 and the legend in Table 2.6.

2.4.2 *Multiple Writer/Multi Reader (M/M)*

This section deals with the M/M schemes. The main focus of the discussed schemes in this architecture lies on single and conjunctive keyword searches. More powerful queries are not proposed, yet.

Table 2.5: Comparison of different M/S schemes. The legend is in Table 2.6.

Scheme	Efficiency: Encrypt	Trapdoor	Search	Security: Definition	Assumption	ROM	Notes
Single Keyword Equality Test							
BCO$^+$ [47]	$n\nu'(2e+p_p^s)$	ke	$n\nu'p_p^s$	PK-CKA2	BDH	✓	IBE
BSS-I [16]	$n\nu'(3e+p_p^s)$	ke	$n\nu'p_p^s$	PK-CKA2	CDH	✓	ElGamal
CS [73]	$n\nu'4lJ$	$k4lh$	$n4lJ$	PK-CKA2	QIP	✓	Jacobi symbols
Khader [113]	$5+3n\nu'e$	$6kP$	$4n\nu'e$	PK-CKA2	DDH	-	IBE
BSS-II [17]	$n\nu'(e+2p_p^s)$	kh	$n\nu'p_p^s$	PK-CKA2	BDH	✓	secure ch. free
RPS$^+$-I [158]	$(2n\nu')e+(7+n\nu')p_p^s$	ke	$n\nu'(e+p_p^s)$	PK-CKA2	BDH, 1-BDHI	✓	improve 2.4.1
LWW [129]	$n\nu'(e+p_p^s)$	ke	$2n\nu'p_p^s$	PK-CKA2	BDH	✓	outsource decryption
TC [175]	$n\nu'(2e+p_p^s)$	ke	$n\nu'p_p^s$	PK-CKA2	BDH	✓	registered keywords
ZI [191]	$n\nu'(8e+2p_p^s)$	ke	$n\nu'(e+p_p^s)$	PK-CKA2	SP	-	AIBE
RPS$^+$-II [159]	$2n\nu'e+n\nu'p_p^s$	$3ke$	$n\nu'(2e+p_p^s)$	PK-CKA2	BDH, 1-BDHI	✓	secure trapdoors
INH$^+$ [105]	$2n\nu'e$	$k(2e+2p_p^a)$	$n\nu'(3e+3p_p^a)$	PK-CKA2	SXDH, mCDH	-	ElGamal

Continued on next page.

Table 2.5: Comparison of different M/S schemes (cont.). The legend is in Table 2.6.

Scheme	Efficiency of algorithms: Encrypt	Trapdoor	Search	Security: Def.	Assumption	ROM	Notes
Conjunctive Keyword Equality Test							
PKL-I [149]	$n\nu' p_p^s$	ke	$n p_p^s$	PK-CKA2[t]	BDH	✓	security broken
PKL-II [149]	ne	$2ke$	$n2p_p^s$	PK-CKA2[t]	BDHI	✓	proof incomplete
PCL [150]	$n2e$	$k2e$	$n2p_p^s$	PK-CCA2[t]	q-BDHI	✓	req. non-empty KF
HL [104]	$n(2+2\nu')e$	$k3e$	$n3p_p^s$	PK-CKA2[t]	DLIN	✓	
Extended Keyword Tests							
BW [43]	$n\nu'(5l+3)e$	$k5(l-w)e$	$n\nu'(2l-w+1)p_{c2}^s$	SEL-CKA	C3DH, BDH	-	HVE, subset, range
SBC$^+$ [169]	$n(8DL+2)e$	$k6DLe$	$n5Dp_p^{(s \text{ or } a)}$	SEL-CKA	BDH, DLIN	-	AIBE, range queries
BCK [55]	$n\nu' b(2h+E+PIS)$	$k2bh$	$kb(PIR+D)$	PK-CKA2	SP	-	error-tolerant, interactive
SLN$^+$ [164]	$n((N+1)(l+1)+4)e$	$k(2l+3)e$	$n\nu'(we+3p_p^s)$	SEL-CKA	DLIN	-	HVE, wildcards

[t] security definition conceptually as the one stated, but tailored to a specific setting (see respective Section)

Table 2.6: Legend for M/S schemes.

Amount		Primitive	
n	number of documents	p_p^s	symmetric prime order pairing
v'	number of distinct keywords per document	p_p^a	asymmetric prime order pairing
k	number of keywords per trapdoor	p_{c2}^s	composite order pairing of degree 2
l	length of keyword in characters	e	exponentiation
w	number of wildcard characters	h	hash function
N	upper bound on wildcard symbols	s	search
D	number of dimensions	J, P	Jacobi symbol, polynomial(s)
L	depth of a node in a tree	E, D	Encryption, Decryption
b	number of Bloom filter hash functions	PIS / PIR	private information storage / retrieval

Single Equality Test

Exact keyword match for a single search keyword.

USING DETERMINISTIC ENCRYPTION. The *idea* of Bellare et al. [29] (BBO) is to make SE more efficient by using deterministic encryption, at the cost of a weaker security model. In particular, the encrypted index — in contrast to the tokens in asymmetric SE — is directly vulnerable to dictionary attacks. To make the ciphertext searchable, a deterministic hash of the keyword is appended to the encryption of the keyword.

EFFICIENCY: BBO can use any public key encryption scheme in combination with any (deterministic) hash function. The *encryption* requires one encryption and one hash per keyword. The *search* consists of a database search for the hash value.

SECURITY: BBO provide a semantic-security definition of privacy for deterministic encryption called PRIV secure. The security definition for deterministic encryption is similar to the standard IND-CPA security definition with the following two exceptions. A scheme provides privacy only for plaintexts with large min-entropy (could be no privacy at all) and the plaintexts have to be independent from the public key. BBO's encrypt-and-hash construction is proven *PRIV* secure in the *RO* model under the assumption that the underlying scheme is IND-CPA secure. Due to the use of deterministic encryption, BBO directly leaks the index information and the search pattern.

SEE ALSO: Deterministic encryption in the S/S setting [12] (Section 2.3.1).

PROXY RE-ENCRYPTION. Dong et al. [78] propose two SE schemes (DRD-I, DRD-II), where each user has its own unique key to encrypt, search, and decrypt data. Both schemes require a trusted key management server to manage the keys.

The *idea* of *DRD-I* is to use an RSA-based proxy re-encryption scheme. Proxy re-encryption was introduced by Blaze et al. [34] and can be built on top of different cryptosystems. The proxy re-encryption allows the server to transform an encryption under a user key to an encryption under a different key, e. g.the server's key, without leaking any information on the plaintext. Thus, ciphertexts from different users, can be transformed to ciphertexts under the server key, which allow multiple users to create searchable encrypted content. In the same way, the trapdoors are created. A user creates a trapdoor for a keyword, by encrypting the keyword with the users key. The server re-encrypts the trapdoor, which allows him to search the encrypted database. Also the decryption requires a re-encryption step to transform a ciphertext under the server key to a ciphertext under the recipients key. DRD-I uses a semantically secure symmetric encryption algorithm to encrypt the data, but the searchable part uses only a hash function which makes the data searchable, but is not semantically secure. Thus the authors also present an enhanced version of their scheme.

DRD-II also uses the RSA-based proxy re-encryption scheme for the data. The *idea* is to use optimal asymmetric encryption padding (OAEP) [24] to make the ciphertexts indistinguishable. RSA-OAEP has been proven to be secure under the RSA assumption [86]. The main difference lies in the keyword encryption. The proxy re-encryption used for the keywords is deterministic. DRD propose to use a semantically secure non-interactive zero-knowledge proof

style witness instead of the proxy re-encryption scheme to make the keyword ciphertexts indistinguishable and give a concrete construction.

EFFICIENCY: *DRD-I:* The *index generation* computes $v + 1$ exponentiations, where v is the number of distinct keyword per document. To *search*, the server re-encrypts (one exponentiation) the trapdoor and tests each keyword per index for equality.

DRD-II: The *index generation* computes $4v + 1$ exponentiations, where v is the number of distinct keyword per document. To *search*, the server re-encrypts (one exponentiation) the trapdoor and then has to compute $4v$ exponentiations.

SECURITY: DRD adapt the IND-CKA1 definition to the M/M setting by giving the adversary access to the public parameters. DRD-II is proven secure under their modified *IND-CKA1* definition. DRD-I and the proxy re-encryption (both schemes) are *One-Way (OW) secure* under the *RSA* assumption in the *RO* model. OW guarantees that it is hard for an adversary to invert a ciphertext encrypted under a users encryption key and to learn the keyword, even if the adversary holds the public parameters and all server side key pairs, but without knowing the user key pair.

The main concern with proxy re-encryption schemes comes from a collusion attack, which allows an adversary and a user to recover the master keys, if the adversary knows all server side keys.

BILINEAR MAPS. Bao et al. [20] (BDD$^+$) propose a multi-user scheme, where each user has a distinct secret key to insert his own encrypted data to the database, while each user is allowed to query the whole database. The *idea* is to use a bilinear map to make sure, that users using different query keys still generate the same index for a keyword. The system allows to *dynamically* add and revoke users without the distribution of new keys. The index generation and data encryption are *interactive* algorithms.

EFFICIENCY: BDD$^+$ uses symmetric bilinear maps of prime order. The *index generation* requires the client to calculate two hashes and two exponentiations. The server has to compute a bilinear map per distinct keyword per document. The server needs to perform only one pairing operation per trapdoor per *search*.

SECURITY: BDD$^+$ is proven *IND-CKA2* secure under the *DDH* and *CDH* assumptions. The construction uses the BLS short signature scheme (BL4S) [44] for query unforgability. The BL4S achieves unforgability in the *RO* model.

SEE ALSO: There is also a journal version of the paper [185].

Conjunctive Keyword Search

See Section 2.3.1 for information on conjunctive keyword searches.

MULTI-RECEIVER PUBLIC KEY ENCRYPTION. Hwang and Lee [104] (HLm) study the problem of public key encryption with conjunctive keyword search (PECK) as discussed in Section 2.4.1. They introduce the concept of multi-user PECK (mPECK) and present the first mPECK scheme. The *idea* is to use multi-receiver PKE [22, 26, 28] and randomness re-use [27, 117] to improve

the computation and communication complexity. HLm does not require a third party.

EFFICIENCY: *Index generation* requires $1 + u + 2v$ exponentiations per document, where u is the number of users and v the number of distinct keywords per document. To *search*, HLm requires three pairing operations per trapdoor.

SECURITY: HLm adapt their PK-CKA2 definition for conjunctive keyword searches to the multi-user setting by giving the adversary access to n user public keys. In addition, the keyword sets are encrypted with these n user public keys. During the trapdoor query phase, the adversary has to specify a user index and receives the trapdoor for this specific user. HLm is secure under the *DLIN* assumption in the *RO* model in their adapted *PK-CKA2* definition.

RSA ACCUMULATOR. Wang et al. [179] (WWP-I) are the first to present a searchable encryption scheme in the M/M setting. The *idea* of WWP-I is to use dynamic accumulators [21, 31, 59] (RSA accumulator for membership authentication), Paillier's cryptosystem [148] and blind signatures [67] (mask encrypted data). They propose a new conjunctive keyword scheme, called common secure index for conjunctive keyword-based retrieval, to share encrypted documents in a *dynamic* group without re-encrypting the data.

In contrast to other SE schemes, where the trapdoor generation requires a private key, the trapdoors in WWP-I are generated with public keys. WWP-I uses a *group manager* (GM), which manages the group members, group keys, and user private keys.

The search part of WWP-I is *interactive* in the following way. First, every user encrypts her documents and creates a common index, both with the group public key. To search, a client sends a trapdoor and an authentication code to the server. After retrieving the matched documents, encrypted under the group key, the client uses her blind signature function to encrypt the documents again and sends the encryptions to GM. GM uses the group secret key to re-encrypt the documents under the users blind signature function and sends the data back to the client, who can now decrypt, using its inverse blind signature function.

EFFICIENCY: *Index generation* uses only one pseudo-random function and multiplications, and is linear in the number of distinct words. The *search* requires a division and additions, and is linear in the number of documents.

SECURITY: WWP-I use the extended IND1-CKA definition from GSW (cf. Section 2.3.1). WWP-I is proven secure under the *coDDH* [19, 42] assumption and the strong *RSA* assumption [21, 72, 85] in the *RO* model in the adapted *IND1-CKA* definition.

DYNAMIC ACCUMULATOR. Wang et al. [181] (WWP-II) propose the first keyword-field free conjunctive keyword search (KFF-CKS) scheme in the *RO* model. The *idea* is to combine Wang et al.'s dynamic accumulator [180] (membership authentication), Nyberg's combinatorial accumulator [137] (conjunctive keyword search scheme), and Kiayias et al. public key encryption [114] (data cryptosystem) to a trapdoorless and keyword-field free scheme. WWP-II

is trapdoorless in the sense that no public or private key is required to generate a trapdoor for a list of keywords. They construct a specific three party cryptosystem (TPC), for the security of the data encryption and decryption, using Kiayias et al. public key encryption [114]. The TPC introduces a *third party*, the group manager (GM). The data retrieval is *interactive* like Wang et al.'s RSA accumulator based scheme [179].

EFFICIENCY: *Index generation* uses a hash and a mapping function, and is linear in the upper bound on the number of distinct keywords. The *search* is linear in the number of indexes.

SECURITY: WWP-II use the same IND1-CKA security definition as in WWP-I (cf. Section 2.4.2). WWP-II is proven secure in the *RO* model under the Decisional Composite Residuosity *(DCR)* assumption [148] and the extended strong RSA *(esRSA)* assumption [180] in the adapted *IND1-CKA* definition.

SEE ALSO: Wang et al. [182] (Section 2.3.1) present a KFF-CKS scheme in the ST model.

BILINEAR MAPS. Wang et al. [182] (WWP-IIIm) present the first keyword-field free conjunctive keyword search scheme in the standard model as discussed in Section 2.3.1. The *idea* for their multi-user extension for dynamic groups is to use Boneh and Franklin's IBE system [41, 42] for data decryption and bilinear maps for user authentication and search. The extension to a dynamic group includes *three parties*: a server, the users (members of a group), and a group manager. The data retrieval is *interactive* like Wang et al.'s RSA accumulator based scheme [179] and dynamic accumulator scheme [181].

EFFICIENCY: WWP-IIIm uses symmetric prime order pairings. The *index generation* constructs a l-degree polynomial per document, where l is the number of distinct keywords contained in the document. The algorithm requires l exponentiations per document. A *search* requires a bilinear map per keyword per document index.

SECURITY: WWP-IIIm use the same IND1-CKA security definition as WWP-I (cf. Section 2.4.2). WWP-IIIm is proven secure under the *DL* and l-*DDHI* assumptions in the adapted *IND1-CKA* definition.

SECRET SHARING. Wang et al. [183] (WWP-IV) introduce the notion of threshold privacy preserving keyword search (TPPKS) and construct the first TPPKS scheme based on Shamir's Secret Sharing [165] (SSS) and Boneh and Franklin's ID-based cryptosystem [41, 42]. Using secret sharing, the *idea* is to allow only collaborating users to search the database. To search, every user generates a share of the trapdoor using her own share of the secret. Then, the collaborating users verify their shares and if the verification was successful, they combine their shares to create the trapdoor for the target keywords. To decrypt, each user generates a decryption share from her secret share. If the decryption shares are valid, the users can compute the plaintext. Due to the use of SSS, WWP-IV is *interactive* and works only for a *fixed group* of users, so adding or removing a user is not possible.

EFFICIENCY: WWP-IV uses symmetric prime order pairings for secret share verification. *Index generation* is linear in the number of keywords per document and requires $\nu + 2$ exponentiations, where ν is the number of keywords. The *search* is linear in the number of keywords and indexes.

SECURITY: WWP-IV use the extended IND1-CKA definition from GSW (cf. Section 2.3.1). The secret share verification is secure under the *DL* and the *CDH* assumption. The search process is secure under the *DDH* assumption in the *RO* model in the adapted *IND1-CKA* definition.

Synthesis

Research in the M/M architecture was conducted in the years 2007 and 2008. The schemes focus on single and conjunctive keyword searches. All discussed M/M schemes use PKE in combination with some kind of key distribution or user authentication to allow multiple users to read the encrypted data. All but one scheme introduce a trusted third party (TTP). For example, most of Wang et al.'s schemes discussed in this section are for dynamic groups. These schemes introduce a group manager (GM) as a trusted third party, which has to re-encrypt the query results to allow the client to decrypt. The advantage of this re-encryption is, that revoked users have no access to any of the stored data any more and that new members have access to all data item, even to those items that were previously stored by other users. Only the HLm scheme does not need a TTP.

Only the WWP-IIIm is proven secure in the standard model. All other schemes use random oracles for their security proofs. Half of the schemes base their security on the RSA assumption or a variant of it. The other half of the schemes use bilinear pairings in their constructions and thus base their security on some kind of DH.

Like the M/S schemes, all of the M/M schemes leak the search pattern and the access pattern. The search pattern is either leaked directly by using a deterministic procedure to generate the trapdoors, or indirectly by an off-line keyword guessing attack as discussed in Section 2.4.1. If a TTP is used in the scheme, the attack takes place between the TTP and the storage server.

Table 2.7 gives a detailed overview of the computational complexity and the security of the different algorithms of the discussed schemes. The digest of the table can be found in the reading guidelines in Section 2.1.5 and the legend in Table 2.8.

2.5 RELATED WORK

In theory, searchable encryption can be achieved by using *oblivious RAMs* (ORAM) [94, 143, 144], which hide all information including the access pattern, from a remote server. The schemes are not efficient in practice, because of a high number of communication rounds and large storage costs on the server side. Therefore, more recent searchable encryption schemes try to achieve more efficient solutions by loosening the security requirements and thus leaking some information (e. g., the access pattern).

The work of Ostrovsky and Skeith [145, 147] on *private stream searching* (PSS), followed by the work of Bethencourt et al. [32, 33] are related to searches on encrypted data. It allows a client to send an encrypted search query to an untrusted server, which then uses this query to search in a stream of unencrypted data. The server returns the matching documents without learning anything about the query. PSS can be seen as a generalization of PIR, in the sense, that more general queries are supported and it is applicable for streaming data.

Table 2.7: Comparison of different M/M schemes. The legend is in Table 2.8.

Scheme	Efficiency: Encrypt	Efficiency: Trapdoor	Efficiency: Search	Security: Definition	Security: Assumption	Security: ROM	Notes
Single Keyword Equality Test							
BBO [29]	$n\nu(E+h)$	kh	ks	deterministic	SP	✓	deterministic hash
DRD-I [78]	$n(\nu'+1)e$	ke	$e+R+nc$	OW	RSA	✓	
DRD-II [78]	$n(4\nu'+1)e$	ke	$(1+3\nu)e+nc$	IND-CKA1^t	RSA	✓	
BDD$^+$ [20]	$\nu' n(2e+p_p^s)$	ke	kp_p^s+nc	IND-CKA2	DDH, CDH	✓	interactive encryption
Conjunctive Keyword Equality Test							
HLm [104]	$n(1+u+2\nu')e$	$u3e$	$u3np_p^s$	PK-CKA2^t	DLIN	✓	no TTP
WWP-I [179]	$\nu' nf$	$k(f+2e)$	ne	IND1-CKA^t	coDDH, sRSA	✓	
WWP-II [181]	$\nu'' nh$	kh	nc	IND1-CKA^t	DCR, esRSA	✓	keyword-field free
WWP-IIIm [182]	$\nu' ne$	$k\nu' 2e$	$\nu' np_p^s$	IND1-CKA^t	DL, l-DDHI	-	keyword-field free
WWP-IV [183]	$n(2+\nu')e$	$u(k+1)2p_p^s$	$n(k+1)e$	IND1-CKA^t	DL, CDH, DDH	✓	fixed user group

gray sub-linear search time

^{t} security definition conceptually as the one stated, but tailored to a specific setting (see respective Section)

Table 2.8: Legend for M/M schemes in Table 2.7.

Amount		Primitive	
n	number of documents	p_p^s	symmetric prime order pairing
ν	number of keywords per document	e	exponentiation
ν'	number of distinct keywords per document	f	pseudo-random function, permutation or generator
ν''	max. number of distinct words	h	hash function
k	number of keywords per trapdoor	c	comparison ($=, \leqslant, <, ...$)
u	number of users	R	re-encryption

Agrawal et al. [11] introduce *order-preserving symmetric encryption* (OPE) for allowing efficient range queries on encrypted data. OPE is a symmetric encryption over the integers such that the numerical orders of plaintexts is preserved in the corresponding ciphertexts. OPE was further studied by Boldyreva et al. [37, 38] and Yum et al. [189].

Chase and Kamara [65] (CK) propose *structured encryption* (STE), which is a generalization of index-based SSE. STE allows private queries on arbitrarily-structured data. The authors give concrete constructions for queries on matrix-structured data, labeled data (cf. Section 2.3.1), and (web) graphs, including neighbor, adjacency and subgraph queries. CK also propose the concept of *controlled disclosure*, which reveals as much information as necessary to compute a function.

Tang [172, 173, 174] and Yang et al. [184] proposed the concept of *public key encryption which supports equality tests* between ciphertexts (PKEET). PKEET schemes allow equality tests of plaintexts which are encrypted under different public keys.

The notion of *predicate encryption* (PE) was first presented by Katz, Sahai, and Waters [112]. PE allows fine-grained access control on encrypted data. It is a generalized notion/concept of encryption that covers various cryptographic primitives such as identity-based encryption (IBE) [41, 42, 71, 166], hidden-vector encryption (HVE) [53, 169], and attribute-based encryption (ABE) [162]. PE targets more powerful queries, but the complexity of the query comes at higher computation cost. In PE, secret keys are associated with predicates and ciphertexts are associated with attributes. A user can decrypt the ciphertext, if the private key predicate evaluates to 1 when applied to a ciphertext attribute. PE comes in two versions: (i) with a public index and (ii) attribute hiding. Schemes of (i) are not usable for searchable encryption, because they lack the anonymity property by leaking the set of attributes under which the data is encrypted. The schemes of (ii) can be used for SE but are often based on bilinear pairings and are thus less efficient than schemes based on simpler primitives.

Inner product encryption (IPE) or PE with inner-product relations covers/implies anonymous IBE (AIBE) and hidden-vector encryption (HVE). In IPE, predicates and attributes are represented as vectors. If the inner product of these two vectors is 0, the predicate evaluates to 1 (e. g., attributes correspond to a vector $\overrightarrow{x}$, each predicate $f_{\overrightarrow{v}}$ corresponds to a vector $\overrightarrow{y}$, where $f_{\overrightarrow{v}}(\overrightarrow{x}) = 1$ iff $\overrightarrow{x} \cdot \overrightarrow{y} = 0$). The dot product enables more complex evaluations on disjunctions, polynomials, and CNF/DNF formulae. Katz, Sahai, and Waters [112] proposed a system over Z_N. Okamoto and Takashima [139] and Lewko et al. [124] gave functions over F_p. Then Okamoto and Takashima [140, 141] and Park [152] proposed more advanced schemes.

Anonymous Identity Based Encryption (AIBE) in its standard form can support only equality tests and works in the multiple writer/single reader (M/S) scenario. Boneh et al. [47] were the first who considered searchable encryption in the asymmetric key setting. Their PEKS scheme was enhanced by Baek et al. [17] and Rhee et al. [158]. PEKS has a close connection to AIBE, as pointed out by Boneh et al. [47]. Abdalla et al. [8] formalize AIBE and present a generic SE construction by transforming an anonymous identity-based encryption scheme to a searchable encryption scheme. More improved IBE schemes that are used for searchable encryption were proposed [53, 87, 115, 135]. To allow delegation, *hierarchical* identity-based encryption (HIBE) [48, 90, 103] was

introduced, where the private keys and ciphertexts are associated with ordered lists of identities. Later, *anonymous* HIBE (AHIBE) schemes [53, 122, 168, 169] were proposed. Abdalla et al. [8] also gave a AHIBE to IBE with keyword search (IBEKS) transformation (hibe-2-ibeks).

Hidden vector encryption (HVE), is a public key encryption scheme that supports wildcard characters inside a key. This allows a variety of application scenarios. Boneh and Waters [43] proposed the first HVE scheme for searching in encrypted data in 2007. Their scheme allows conjunctive, subset, and range queries. Katz et al. [112] extended the list with disjunctions, polynomial equations, and inner products. For more information on HVE schemes, we refer the reader to [36, 62, 100, 106, 123, 135, 152, 153, 164, 168]. Delegation in PE, more precisely a primitive called delegateable hidden vector encryption (dHVE) was introduced by Shi and Waters [168]. Iovino and Persiano [106] provide a solution based on prime order groups, but the scheme works only on binary symbols. HVE can be seen as an extreme generalization of AIBE [43]. If the HVE is keyword-hiding, the transformed PEKS does not leak any information about the keyword used in the Encrypt algorithm.

Homomorphic encryption (HE) is a special type of encryption that allows to perform an algebraic operation on ciphertexts without decrypting them. This makes HE an interesting tool for searching over encrypted data, since meaningful computation can be executed on the encrypted data. Most HE schemes support either additions [148] or multiplications [79] on ciphertexts. The pairing based HE scheme proposed by Boneh, Goh, and Nissim [49] is able to perform an arbitrary number of additions and one multiplication. Recently, *fully* homomorphic encryption (FHE) was proposed, which can compute arbitrary functions over encrypted data [88, 89, 176]. It is generally believed, that FHE can solve the problem of querying encrypted data, since *any* meaningful computation can be performed on the encrypted data. However, one issue with FHE is the performance, since current schemes are computationally expensive and have a high storage overhead. Since the first FHE scheme, researchers try to make the schemes more efficient, but still no practical construction has been proposed [134]. For some applications, so called *somewhat* homomorphic encryption schemes can be used. These schemes are more efficient then FHE, but allow only a certain amount of additions and multiplications [54, 91]. The major issue when using somewhat or fully HE as is, is that the resulting search schemes require a search time linear in the length of the dataset. This is too slow for practical applications.

2.6 CONCLUSIONS AND FUTURE WORK

This section gives a summary of our main results, draws conclusions for the theoretically and the practically oriented community and gives a discussion on directions for future research.

2.6.1 *Summary*

Since the early stages of SE, research in the field has been active in all three research directions: improving the query expressiveness, the efficiency, and the security. One can recognise the trade-offs among these three directions: (i) security vs. efficiency, (ii) security vs. query expressiveness, and (iii) efficiency vs. query expressiveness. When a scheme tries to be better in one aspect, usu-

ally it has to sacrifice another. A good example demonstrating the trade-off issue, especially for the case (i) is using deterministic encryption. Deterministic encryption makes a scheme more efficient, but at the same time leaks more information, i. e., the ciphertext itself without any trapdoor leaks information (e. g., document/keyword similarity) and directly leaks the search pattern. In the case of public key deterministic encryption using well known keywords, the server can start a brute force attack, by encrypting all possible keywords with the public key and check the encryptions against the ciphertexts.

Table 2.9 gives an overall view of the field of provably secure searchable encryption. The columns of the table represent the different architectures. In the first eight rows, a check mark denotes that there exists a scheme with the specific query expressiveness. A dash indicates, that we are not aware of a provably secure SE scheme with the respective expressiveness captured by that row. The ninth row gives the number of schemes per architecture, discussed in this chapter. The number of implemented schemes that we know of is stated in the tenth row. The last row denotes the timespan, in which research in the corresponding architecture was conducted.

In total, we analyzed 58 schemes. As indicated in Table 2.9, most of the SE schemes proposed so far fall either in the S/S, or in the M/S architecture. This is due to the use of symmetric or asymmetric encryption primitives, respectively. Although the S/M architecture has not received much research attention in the past, the two existing schemes, that fall into this architecture, were proposed in 2011. The S/M architecture as the natural extension of the S/S architecture is used for data sharing, where a single writer shares data with several readers. This is a common scenario in practice. Nevertheless, research with respect to this architecture is lean. The same applies to the M/M architecture, which was intensively researched between 2007 and 2008, but seems to be currently out of interest.

Note that an S/S scheme can be trivially constructed from an M/S scheme, by keeping the public key of the PKE in use, secret. Since the M/S schemes use PKE, it is likely that those schemes are an order of magnitude less efficient than an S/S scheme which is based on symmetric primitives. S/S schemes can also be trivially constructed from S/M schemes.

Only six papers (ABC$^+$, CJJ$^+$, DRD, KPR, PKL$^+$, and RVB$^+$) provide an implementation of the schemes including performance numbers. Most implementations are not publicly available, which makes it hard to compare the schemes on the same hardware with the same dataset. Moreover, it is hard to provide a direct performance comparison, since existing protocols for SE address different scenarios and threat models.

2.6.2 *Conclusions*

After more than a decade of research, significant progress in the field of provably secure searchable encryption has been made. Research has taken place in all three research directions, mainly focusing on the improvement of query expressiveness and security. In the following, we present our conclusions, classified based on those three research directions.

QUERY EXPRESSIVENESS. Existing SE schemes already achieve a variety of different search features which allow the deployment of a wide range of applications. Looking at Table 2.9 we observe a lack of expressiveness in the

Table 2.9: Overview of known research in the field.

Architecture	S/S	S/M	M/S	M/M
Equality	✓	✓	✓	✓
Conjunction	✓	-	✓	✓
Comparison	-	-	✓	-
Subset	(✓)	-	✓	-
Range	(✓)	-	✓	-
Wildcard	-	-	✓	-
Similar/Fuzzy	✓	-	-	-
Inner Product	✓	-	(✓)	-
# of schemes	27	3	19	9
# of implementations	6	1	-	2
Timespan	2000 - 2013	2009 - 2011	2004 - 2011	2007 - 2008

$*$/M architectures. The widest variety in query expressiveness can be found in the M/S architecture. This is due to the various public key encryption schemes and primitives used in this area. Since there is a wide spectrum of different techniques for PKE, searchable encryption in the M/S architecture can be realized using different primitives, e. g., IBE [40], AIBE, and HIBE [8] or HVE [43]. SE in the M/S setting moves more and more towards (fine-grained) access control (AC). Using AC, a simple search consists of testing whether a certain trapdoor is allowed to decrypt a ciphertext. A successful decryption indicates a match. Since in general, IPE can be used for SE, but no explicit IPE scheme for SE exists, the checkmark in Table 2.9 is in parentheses. Note, that inner products can support conjunctive, disjunctive, subset, and range queries, as well as polynomial evaluations and CNF/DNF formulas.

EFFICIENCY. While research in SE continues in all directions, there is an efficiency issue in the multi-user setting that needs to be solved to allow a widespread use of searchable encryption. Efficient schemes with sub-linear or optimal search time that achieve IND/PK-CKA2 security exist only in the S/S setting. Current SE schemes in the S/M, M/S and M/M settings, achieving IND/PK-CKA2 security, are inefficient (linear in the number of data items/-documents) and do not scale well for large datasets which makes them impractical for real life use. The deployment of the proposed schemes will cause high query latency and will allow a database server to serve a limited number of clients.

However, two reasons make it more and more urgent to construct practical schemes, namely: (i) governments force organizations to use encryption and (ii) the increasing utilization of cloud services:

i) A number of governmental regulations, such as the Health Insurance Portability and Accountability Act (HIPAA) and the Sarbanes-Oxley Act (SOX), stipulate that organizations have to encrypt their data to prevent it from being disclosed to unauthorized parties. At the same time, organizations require to search in and process their encrypted data in a

secure way. Thus, it gets more and more important to come up with solutions, confronting the needs of real-world scenarios.

ii) In the past, companies relied solely on a protected environment and strong access control for their databases. The current use of cloud services leaves companies to rely solely upon encryption to secure their data and confront threats such as business espionage.

SECURITY. All discussed schemes achieve provable security. SE schemes need to be proven secure in some sense. If a scheme is not secure, encryption is redundant. Nonetheless, even if a scheme is proven secure, it is hard to assess its security, since most schemes are proven secure in different security models and under different computational hardness assumptions. Thus, comparing existing schemes in terms of security is not always possible. Some schemes base their proofs on standard or well-known assumptions. Others come up with novel security assumptions, which makes the evaluation of the security difficult. Nevertheless, the IND-CKA2 definition gained widespread acceptance as a strong notion of security in the context of SE.

However, using this definition alone is not enough, since the security of SE schemes also needs to take into account the information leakage during or after a search. Recently, Kamara et al. [109] proposed a framework for describing and comparing the leakage of SE schemes. As a result, the IND-CKA2 definition can be parameterized with additional leakage functions to specify the full information leakage. This is important, since the leaked information can/might be used for statistical analysis (e. g., the search pattern).

The search pattern reveals whether two searches were performed for the same keyword or not. Hence, the search pattern gives information on the occurrence frequency of each query. This is a serious problem, as it allows an attacker to perform statistical analysis on the occurrence frequency, eventually allowing the attacker to gain knowledge about the underlying plaintext keywords.

Most proposed schemes, allow the leakage of the search pattern. Some strive to hide as much information as possible. Revealing the search pattern might not be a problem for certain scenarios, whereas for others it is unacceptable. In a medical database for example, revealing the search pattern might already leak too much information. This information might be used to correlate it with other (anonymised) public databases. In most situations it is reasonable that an SE scheme leaks the access pattern. However, for high security applications the protection of the access pattern might be mandatory. To the best of our knowledge, there is no practical SE scheme that hides the access pattern. However, Boneh et al. [50] propose a theoretical solution for PKE that allows PIR queries and does not reveal any information with respect to the user's search; not even the access pattern. Their solution is computationally expensive and closer to the concept of PIR than SE, which lead us to exclude the aforementioned solution from our analysis.

2.6.3 *Future Work*

Future research should focus on improving the query expressiveness and the efficiency/scalability of SE schemes in the S/M, M/S and M/M setting. For some applications, more secure schemes that protect the search pattern might be of interest.

QUERY EXPRESSIVENESS. Research on query expressiveness needs to move towards closing the gap between existing SE schemes and plaintext searches. This includes, but is not limited to functionalities like phrase search, proximity search or regular expressions. Especially in the $*/M$ settings, which represent typical scenarios of data sharing, more query expressiveness is desirable. An interesting research question for $*/M$ architectures is whether it is possible to create new schemes which do not rely on a TTP.

An interesting approach for future research is certainly a problem-driven approach; identifying the real-world problems, requirements, and needs first and then trying to address them by means of SE would lead to concrete and useful application scenarios, e. g., search in outsourced (personal) databases, secure email routing, search in encrypted emails and electronic health record (EHR) systems. In order to make an important step towards a widespread use of searchable encryption, multi-user schemes need to become more efficient and scalable for large datasets. To assess the real performance of the constructions, more implementations, performance numbers or at least concrete parameters for the used primitives are necessary.

EFFICIENCY. The main focus of future research in the multi-user setting should be the efficiency, since a problem with existing multi-user SE schemes is that they are not practical in real-world applications and do not scale well for large datasets. Only the S/S setting presents reasonable efficient and/or scalable constructions. Consequently, one of the goals of future work should be the reduction of the computational complexity. More efficient provably secure schemes are essential. One possible way to achieve that, seems to be the use of different, more efficient primitives or different data representations (e. g., forward index vs. inverted index vs. trees).

Another promising way, to address the efficiency/scalability problem, might be to explore the possibilities of using two or more collaborating servers to make the search process more efficient. This approach already exists in the context of Secure Multi-Party Computation [187] and Private Information Retrieval [69]. Another approach towards improving the efficiency in SE is outsourcing (heavy) computations to third-party entities. One could explore the possibilities of outsourcing (parts of) the computation to (i) dedicated computing utilities and (ii) peer-to-peer networks. The field of distributed cryptography could be of help towards this direction. Distributed systems have the additional advantages of autonomy and decentralization, fault tolerance, and scalability. Also, distributed designs can be implemented to be secure against malicious participants and/or allow users to remain anonymous.

In the current research stage, most of the schemes are non-interactive. Deploying interactive protocols, can enable the use of simpler primitives and might be computationally more efficient than non-interactive protocols. On the other hand, the communication complexity will most likely increase. This creates a new efficiency trade-off: computational efficiency vs. communication efficiency. However, interactive protocols can achieve a higher level of security [3, 50] or efficiency [63].

SECURITY. First and foremost, a searchable encryption scheme should be secure. The IND-CKA2 security definition is considered strong in the context of searchable encryption, but allows the leakage of the search pattern. The leakage of the search pattern can be exploited to break the encryption, which

Table 2.10: Security assumptions used in this chapter. A * marks assumptions that are generally believed to be well studied.

Security Assumption		Reference
Bilinear DH	(BDH)*	[41, 107]
Bilinear DH Inversion	(BDHI)	[40, 133]
Composite 3-Party DH	(C3DH)	[43]
Computational DH	(CDH)*	[77]
Co-Diffie-Hellman	(coDDH)	[19, 42]
External co-DH	(coXDH)	[161]
Decisional Composite Residuosity	(DCR)*	[148]
Decisional DH	(DDH)*	[39]
Decisional DH Inversion	(DDHI)	[60]
Discrete Logarithm	(DL)*	[77]
Decisional Linear DH	(DLIN)*	[46]
Extended Strong RSA	(es-RSA)	[180]
Gap DH	(GDH)	[138]
Modified CDH	(mCDH)	[105]
Mixed XDH	(MXDH)	[18]
Quadratic Indistinguishability Problem	(QIP)	[95] is QRP
RSA	(RSA)*	[160]
Strong RSA	(s-RSA)*	[21, 72, 85]
Symmetric External DH	(SXDH)	[18]
External DH	(XDH)	[46, 163]

might be fatal in some application scenarios. Since only SSW (predicate encryption) is fully secure, future work should strive to also protect the search pattern.

APPENDIX

Table 2.10 gives an overview of the security assumptions used by the schemes, discussed in this chapter. The first row gives the assumption name and its abbreviation, the second row gives the reference to the assumption.

3 SELECTIVE DOCUMENT RETRIEVAL

As discussed in Chapter 2, existing searchable encryption schemes allow the leakage of the search pattern, which can be used to break the encryption. In this chapter, we construct a provably secure search pattern hiding scheme that is orders of magnitude more efficient than the search pattern hiding predicate encryption scheme by Shen, Shi, and Waters [167] (cf. Section 2.3.1). To hide the search pattern we use approach A1 by privately calculating the inner product between the trapdoor and the database. We propose the concept of selective document retrieval (SDR) from an encrypted database which allows a client to store encrypted data on a third-party server and perform efficient search remotely. The interactive nature of SDR allows to achieve a higher level of security than any existing searchable symmetric encryption (SSE) scheme, although it might be regarded as an efficiency drawback in some application scenarios. We propose a new SDR scheme – based on the recent advances in somewhat homomorphic encryption – and prove its security in our security model. Regarding query expressiveness, our scheme can be extended to support many useful search features, including aggregating search results, supporting conjunctive keyword search queries, advanced keyword search, search with keyword occurrence frequency, and search based on inner product. To evaluate the performance, we implement the search algorithm of our scheme in C. The experiment results show that a search query is three orders of magnitude faster than existing schemes that achieve a similar level of security, i. e., protecting the search pattern.

3.1 INTRODUCTION

Outsourcing data to a third-party server is continuously gaining popularity because it can significantly reduce operational costs for a client. However, to store outsourced data securely on an untrusted server, the data should be encrypted to make it inaccessible to the server and other attackers. The issue is that, if the encryption is done with standard encryption schemes, the client will not be able to search anymore unless it retrieves the whole outsourced database from the server. To solve the problem, we need a special type of encryption primitive which allows the following things.

1. The client can encrypt his data and store the ciphertext on the server. More specifically, we assume the client stores a list of (document, index) pairs on the server, where the index is an encrypted version of the keywords which appear in the document. Note that the document should be encrypted independently. We skip the details of document encryption because it is not relevant for the search.
2. The client can ask the server to search the indexes on his behalf, without leaking information about the keywords in the indexes and what has been searched for. Moreover, the client may even want to hide from the server the fact which documents have been matched by a search query.

3. The client can selectively retrieve (possibly in a private manner, i. e., without leaking the access pattern) the contents identified by a search. The option for the client to selectively retrieve matched documents may be very useful in practice. For example, a search may indicate that 900 out of 1000 documents are matched, but the client may just want to retrieve 10 of them instead of all of them due to various reasons. The option for the client to retrieve matched documents in a private manner may also be useful in practice since which documents are retrieved can already leak some information about the documents.

The first two requirements are motivated by security considerations in an outsourcing scenario, while the last one is motivated by flexibility, efficiency, and security considerations. Note that, an alternative solution would be for the client to store a plaintext copy of the indexes locally so that she can search by herself. This is not a good solution because the client needs to maintain the index storage.

In the setting of SSE, the server can directly return the matched documents after executing the SearchIndex algorithm on all indexes. In contrast, in the setting of SDR, the SearchIndex algorithm only returns the encrypted search results to the client, so that the client and the server need to additionally run a Retrieve algorithm, as explained later, for the client to retrieve the matched documents.

PROBLEM STATEMENT. In the direction of solving the above problem, searchable encryption schemes (SE) have been the most relevant cryptographic primitive. A SE scheme enables a third-party server to search on his behalf directly on encrypted data without knowing the plaintext data. In particular, SE in the symmetric setting can serve as a more suitable solution, where the term *symmetric* means that only the client can generate searchable contents. It is worth noting that there also exist SE schemes in the asymmetric setting, such as PEKS [47], where the concept of a public key encryption scheme is employed and every entity can generate searchable data. As discussed in Chapter 2 it is impossible to hide the search pattern in PEKS schemes due to the use of public key encryption. Thus, asymmetric SE is of lesser interest to our problem.

SSE is meant to achieve the functionalities in the first two requirements mentioned before. By a straightforward extension as discussed in Remark 3.1 in Section 3.2.1, it can achieve the functionality in the third requirement. However, with respect to the desired security guarantees, an SSE scheme leaks a lot of sensitive information to the server, and such information includes (at least) which documents match the client's search request and which documents the client has retrieved.

OUR CONTRIBUTION. Firstly, we propose a new cryptographic primitive, namely selective document retrieval (SDR), and present a security model.

Secondly, based on the recent advances in somewhat homomorphic encryption schemes and the index construction technique by Chang and Mitzenmacher [64], we propose a search pattern hiding SDR scheme to support equality test predicates and prove its security in the proposed security model. The intuition behind the construction is rather straightforward, but interestingly it can serve as a framework to support more flexible search features than single equality tests. We show that the proposed SDR scheme can be easily adapted to support features, including aggregating search results, supporting conjunc-

tive keyword search queries, advanced keyword search, search with keyword occurrence frequency, and search based on inner product.

Thirdly, we set appropriate parameters for the symmetric BV encryption scheme [54] and implement it in C. This is the first publicly-available implementation of the scheme in C with carefully chosen parameters, so that it may be of independent interest for other works[1]. We use the BV scheme to instantiate the encryption component in the proposed SDR scheme, and evaluate the performances. The experiment results show that a search query takes only 47 seconds in an encrypted database with 1000 documents and 100 keywords, while a search query takes around 10 minutes in an encrypted database with 5000 documents and 250 keywords. In contrast, for the SSW scheme by Shen et al. [167], a search query takes around 16 hours in an encrypted database with 1000 documents and 100 keywords on the same server. We did not study the document retrieval performance, because it will be similar for all schemes if they are to achieve a similar level of security. We note that although the performance of the proposed SDR scheme does not say that it is an efficient solution in all application scenarios, it is the most efficient one we have now.

As in the case of SSE, an SDR scheme allows a client to store encrypted data on a third-party server and performs efficient search remotely. The difference is that the server does not directly learn which documents match the client's query, therefore, an additional communication round is required for the client to retrieve the related documents. The interactive nature of SDR allows to achieve a higher level of security than any existing SSE schemes, although it might be regarded as an efficiency drawback in some application scenarios. According to the definition of SSE, the server learns the match results from executing the search query. As a result, a secure SSE scheme cannot be directly extended to achieve all three privacy properties, even if it is fully secure under the definition of Shen et al. [167].

ORGANIZATION. The rest of this chapter is organized as follows. In Section 3.2, we describe SDR and formalize its security property. In Section 3.3, we propose a SDR scheme and prove its security in Section 3.4. In Section 3.5, we describe various search features of the proposed SDR scheme. In Section 3.6, we implement the search algorithm of the proposed SDR scheme and analyze the experimental results. Section 3.7 concludes this chapter.

3.2 SELECTIVE DOCUMENT RETRIEVAL

Throughout this chapter, we use the following notation. Let $\mathcal{D} = \{d_1, \ldots, d_n\}$ be a set of n documents and $\mathcal{W} = \{w_1, \ldots, w_m\}$ be a pre-built dictionary of m keywords. Given a document d, let $u(d)$ denote the set of distinct keywords in d. The keyword used in a query is denoted by $s \in \mathcal{W}$. Given a tuple t, we refer to the i-th entry of t as $t[i]$.

3.2.1 *Algorithmic Definition of SDR*

An SDR scheme comprises five algorithms (Keygen, BuildIndex, Trapdoor, SearchIndex and Retrieve), defined as follows.

1 http://scs.ewi.utwente.nl/other/boesch/bv.zip

- $K \longleftarrow \mathsf{Keygen}(\lambda)$: Run by a client, this algorithm takes a security parameter λ as input, and outputs a secret key K. It may also generate some other public parameters such as a predicate set $\mathcal{F}$.
- $\mathcal{I}_d \longleftarrow \mathsf{BuildIndex}(K, d)$: Run by the client, this algorithm takes the key K and a document $d \in \mathcal{D}$ as input, and outputs an encrypted index $\mathcal{I}_d$.
- $T_f \longleftarrow \mathsf{Trapdoor}(K, f)$: Run by the client, this algorithm takes the key K and a predicate $f \in \mathcal{F}$ as input, and outputs a trapdoor T_f.
- $[\![R_d]\!] \longleftarrow \mathsf{SearchIndex}(T_f, \mathcal{I}_d)$: Run by the server, this algorithm takes a trapdoor T_f and an index $\mathcal{I}_d$ as input and returns an encrypted result $[\![R_d]\!]$ to the client, where R_d implies whether $u(d)$ satisfies the predicate f or not.
- $E(d_i) \longleftarrow \mathsf{Retrieve}(K, \{[\![R_d]\!] \mid d \in \mathcal{D}\}; \mathcal{D})$: Run between the client and the server, the client takes the secret key K and the encrypted search results $\{[\![R_d]\!] \mid d \in \mathcal{D}\}$ as input and the server takes the encrypted database $\mathcal{D}$ as input. At the beginning of the protocol, the client first decrypts $\{[\![R_d]\!] \mid d \in \mathcal{D}\}$ and decides which documents to retrieve, and at the end of the protocol the client retrieves the documents she wants.

A standard work flow of SDR is shown in Figure 3.1. In the setup phase, a client C first runs the Keygen algorithm to generate the key K and parameters. Then, in the upload phase, C runs the BuildIndex algorithm to generate an index for every document she has and finally stores every (document, index) pair on the server. We assume that the documents are encrypted by the client with some standard symmetric encryption algorithm using a key different from K. Later on, in the query phase, when the client wants to retrieve some documents, it first runs the Trapdoor algorithm to generate a trapdoor, then sends the trapdoor to the server which can then run the SearchIndex algorithm to match the trapdoor with every index in the database and send the encrypted match results to the client. Finally, the client runs the Retrieve algorithm with the server to retrieve (some of) the matched documents. Note that the client can selectively retrieve the matched documents, not necessarily all of them.

REMARK 3.1 Referring to the definition of SSE [75], any SSE scheme can be trivially extended to a SDR scheme: by letting the server send the search results (i. e., outputs of SearchIndex executions) back to the client, who then selectively determines which documents to retrieve. If we assume that the server returns all the documents matched by the SearchIndex in SSE, then it is equivalent to a SDR scheme in which the client always retrieves all the matched documents.

Similar to the case in other cryptographic primitives, an SDR scheme should always be sound, namely the following two conditions always hold.

1. If $u(d)$ satisfies f, then $\mathsf{Retrieve}(K, \{[\![R_d]\!] \mid d \in \mathcal{D}\}; \mathcal{D})$ will return all documents d chosen by the client.
2. If $u(d)$ does not satisfy f, then the probability that $\mathsf{Retrieve}(K, \{[\![R_d]\!] \mid d \in \mathcal{D}\}; \mathcal{D})$ returns d is negligible.

3.2.2 *Security Properties for SDR*

Recall that the main objective of SDR schemes is to enable the server to search over the encrypted data and let the client selectively retrieve the matched con-

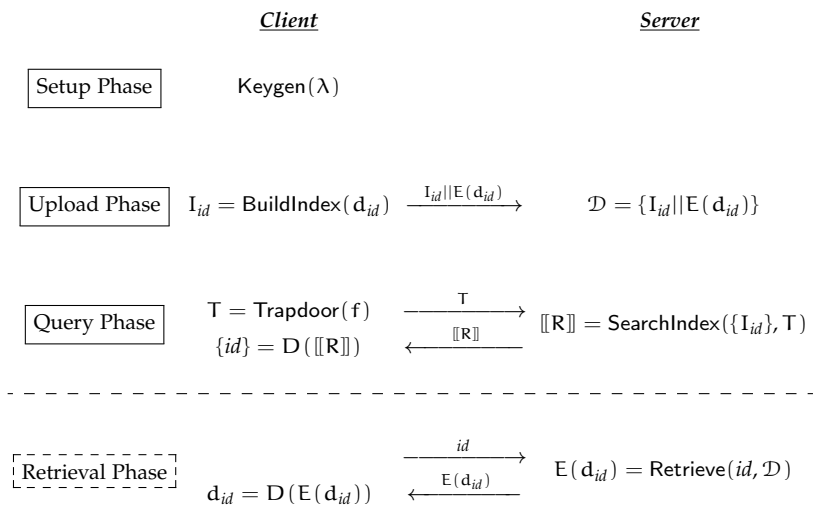


Figure 3.1: General model of selective document retrieval (SDR) schemes. For ease of readability we omit the secret key K.

tents. In this setting, information leakage can come from three sources, namely index, trapdoor, and query results. Correspondingly, there are three types of privacy concerns.

- *Index privacy*, similar to the plaintext privacy in [167], means that indexes should not leak any information about the encoded keywords.
- *Trapdoor privacy*, similar to the predicate privacy in [167], means that trapdoors should not leak any information about the encoded predicates. This inherently includes the protection of the search pattern.
- *Query result privacy* means that if the client retrieves x documents for any integer x in two executions of the Retrieve algorithm, then the server should not know whether the two executions return the same documents or not.

The concerns of index privacy and trapdoor privacy have been considered by existing SSE schemes. Notably, Shen et al. [167] propose a definition of *full security*, which tries to capture the above two privacy concepts. Note that, Shen et al. only give a fully secure SSE scheme which supports inner product queries for vectors of even length, without being able to present a scheme which generally achieves *full security*. To our knowledge, no SSE scheme has been shown to be fully secure in general.

However, query result privacy has not been touched upon in the setting of outsourcing encrypted data, although it is a practical concern for many application scenarios. For example, suppose that Alice stores both her work-related documents and personal documents on a remote server protected by an SSE scheme. Moreover, she only queries her work-related documents in her office, and queries personal documents at home. One day, if the server notices at 10:00 pm that Alice is querying the same document as that she queried at 11:00 am, then the server can guess that Alice is working over the time in her office.

1. The challenger runs the Keygen algorithm and obtains the secret key K and the predicate set $\mathcal{F}$. The challenger publishes $\mathcal{F}$ and picks a random bit b.
2. The attacker $\mathcal{A}$ adaptively makes the following types of queries.
   - *Index oracle query.* On the j-th index query, $\mathcal{A}$ outputs two documents $\mathsf{d}_{\mathsf{j},0}, \mathsf{d}_{\mathsf{j},1} \in \mathcal{D}$. The challenger responds with $\mathsf{BuildIndex}(\mathsf{K}, \mathsf{d}_{\mathsf{j},\mathsf{b}})$.
   - *Trapdoor oracle query.* On the i-th trapdoor query, $\mathcal{A}$ outputs two predicates $\mathsf{f}_{\mathsf{i},0}, \mathsf{f}_{\mathsf{i},1} \in \mathcal{F}$. The challenger responds with $\mathsf{Trapdoor}(\mathsf{K}, \mathsf{f}_{\mathsf{i},\mathsf{b}})$.
   - *Retrieve oracle query.* Suppose that there have been j index queries and i trapdoor queries, the challenger (simulating the client) and the server runs the $\mathsf{Retrieve}$ algorithm. The server's input is the database $\mathcal{D}$, which contains j (index, document) pairs, and the challenger's input is the key K and a set of document identifiers $\mathcal{ID}_{\mathsf{b}}$, where $\mathcal{ID}_0$ and $\mathcal{ID}_1$ are two identifier sets of identical size chosen by the attacker. Basically, $\mathcal{ID}_{\mathsf{b}}$ tells which documents the challenger should retrieve.
3. $\mathcal{A}$ outputs a guess b' of the bit b.

Figure 3.2: Attack Game of SDR

3.2.3 *Game-style Security Definition*

Similar to the security definitions in SSE security models, we consider the attacker to be a semi-honest server (and any other outside attacker). By semi-honest we mean an honest-but-curious [93] database server that can be trusted to adhere to the protocol, but which tries to learn as much information as possible. Formally, the definition is as follows.

DEFINITION 3.1 An SDR scheme is secure if no probabilistic polynomial-time attacker has non-negligible advantage in the attack game defined in Figure 3.2, where the advantage is defined to be $|\Pr[\mathsf{b} = \mathsf{b}'] - \frac{1}{2}|$.

By granting index oracle queries to the attacker, we cover index privacy in the sense that the attacker cannot distinguish the indexes of different documents. By granting trapdoor oracle queries to the attacker, we cover trapdoor privacy in the sense that the attacker cannot distinguish the trapdoors received from the client. Similarly, by granting retrieve oracle queries to the attacker, we cover query result privacy in the sense that the attacker cannot tell apart the retrieved documents by the client. Note that in granting the retrieve oracle queries, we restrict that the identity sets are of the same cardinality; other-

wise the attacker may trivially win the game unless the client always retrieves all the documents. As a consequence, if an SDR scheme is secure under this definition, an attacker only learns how many documents the challenger has retrieved but nothing else.

REMARK 3.2 Compared with the full security definition for SSE [167] the above definition formulates strictly stronger security protection because we removed the restriction on the index and trapdoor queries in the attack game. Not surprisingly, a SDR scheme resulting from a simple extension based on a fully secure scheme as mentioned in Remark 3.1 in Section 3.2.1 will not be secure under Definition 3.1.

3.2.4 *Relaxation of the Security Definition*

As discussed before, query result privacy may be an important concern in many application scenarios for SDR schemes, but it may not be so important in other scenarios. To be secure under Definition 3.1, the Retrieve algorithm of an SDR scheme will use a private information retrieval [69, 142] technique in one way or another so that it will incur significant computational and communication complexities, hence this privacy property may be sacrificed for the efficiency reasons. As a result, it is useful to have a definition covering only index privacy and trapdoor privacy. Formally, we give the following definition.

DEFINITION 3.2 An SDR scheme achieves index privacy and trapdoor privacy, if no probabilistic polynomial-time attacker has non-negligible advantage in the attack game defined in Figure 3.2 with the following exceptions.

1. Retrieve oracle challenge query is disallowed in the game.
2. For any index oracle challenge query $(d_{j,0}, d_{j,1})$ and any trapdoor oracle challenge query $(f_{i,0}, f_{i,1})$, the following is true:
   $u(d_{j,0})$ satisfies $f_{i,0}$ if and only if $u(d_{j,1})$ satisfies $f_{i,1}$.

With the relaxation, the above definition provides the same level of security guarantees to the full security definition [167].

Besides the above relaxation, it is straightforward to give a single challenge version of Definition 3.1 in the same manner as Shen et al. [167], which only allows the attacker to make the challenge on a pair of messages or predicates. As in the case of Shen et al., the new definition will provide weaker privacy guarantee than Definition 3.1.

3.3 THE PROPOSED SDR SCHEME

In this section, we describe a new SDR scheme and prove its security in the security model described in Section 3.2. We describe the scheme for the case of equality test predicates, while the scheme does support other types of predicates which will be elaborated in Section 3.5.

3.3.1 *Preliminary*

An encryption function $E(\cdot)$ is called homomorphic if there exist two (possibly the same) operations ($\otimes$ and $\oplus$), such that $E(a) \otimes E(b) = E(a \oplus b)$. In this

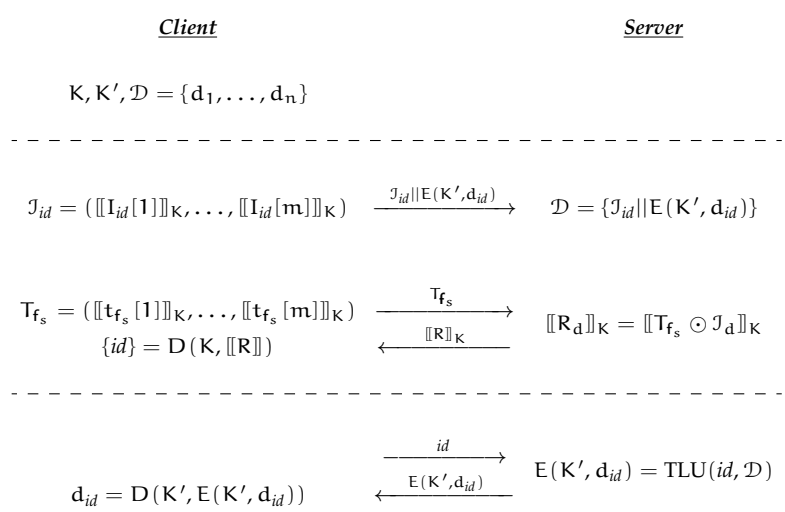


Figure 3.3: Our proposed SDR scheme. TLU denotes a table look-up.

chapter the homomorphic encryption of an element x is written as $\llbracket x \rrbracket$. Thus $\llbracket a \rrbracket \otimes \llbracket b \rrbracket = \llbracket a \oplus b \rrbracket$. In our construction, we use a semantically secure homomorphic encryption scheme that allows one multiplication followed by multiple additions on encrypted values. For example, the lattice-based schemes such as the Gentry-Halevi-Vaikuntanathan (GHV) [91] scheme and Brakerski-Vaikuntanathan (BV) [54] scheme and the pairing-based encryption scheme from Boneh, Goh and Nissim (BGN) [49] satisfy the required property.

3.3.2 *The Proposed Scheme*

The proposed SDR scheme makes use of a symmetric homomorphic encryption scheme satisfying the requirements stated in Section 3.3.1 and the index construction method by Chang and Mitzenmacher [64]. Next, we describe the algorithms of the proposed scheme, namely (Keygen, BuildIndex, Trapdoor, SearchIndex, Retrieve).

- $K \longleftarrow \mathsf{Keygen}(\lambda)$. Given a security parameter λ, generate a key K for a symmetric homomorphic encryption scheme, such as the symmetric version of the BV scheme described in Section 3.6.1, and equality test predicate set $\mathcal{F} = \{f_s \mid s \in \mathcal{W}\}$. For any document d, $u(d)$ satisfies f_s if and only if $s \in u(d)$.
- $\mathcal{I}_d \longleftarrow \mathsf{BuildIndex}(K, d)$. With the key K and a document d, the algorithm does the following:
  1. Generate the list of distinct keywords, namely $u(d)$.
  2. Construct a plaintext index for d, denoted as $I_d = (I_d[1], I_d[2], \ldots, I_d[m])$. Note that m is the size of the possible keyword set. The bit $I_d[i]$ is set to be 1 if $s \in u(d) = w_i \in \mathcal{W}$; otherwise, the $I_d[i]$ is set to be 0.
  3. Generate $\llbracket I_d \rrbracket = (\llbracket I_d[1] \rrbracket, \llbracket I_d[2] \rrbracket, \ldots, \llbracket I_d[m] \rrbracket)$, which means that the plaintext version index is encrypted bit by bit.

4. Output the index $\mathcal{I}_d = [\![I_d]\!]$.

- $T_{f_s} \longleftarrow \mathsf{Trapdoor}(K, f_s)$. With the key K and a predicate f_s, the algorithm does the following:
  1. Construct $t_{f_s} = (t_{f_s}[1], t_{f_s}[2], \ldots, t_{f_s}[m])$. For every $1 \leqslant i \leqslant m$, the value of $t_{f_s}[i]$ is set to be 1 if $s = w_i$ and 0 otherwise.
  2. Output the trapdoor $T_{f_s} = ([\![t_{f_s}[1]]\!], [\![t_{f_s}[2]]\!], \ldots, [\![t_{f_s}[m]]\!])$.
- $[\![R_d]\!] \longleftarrow \mathsf{SearchIndex}(T_{f_s}, \mathcal{I}_d)$. With a trapdoor T_{f_s} and an index $\mathcal{I}_d$, the algorithm outputs $[\![R_d]\!] = [\![t_{f_s} \odot I_d]\!]$, where the notation $\odot$ represents an inner product. Note that the computation is based on T_{f_s} and $\mathcal{I}_d$ using the homomorphic properties stated in Section 3.3.1. The server sends $[\![R_d]\!]$ to the client.
- $E(d_i) \longleftarrow \mathsf{Retrieve}(K, \{[\![R_d]\!] \mid d \in \mathcal{D}\}; \mathcal{D})$. Here, $\mathcal{D}$ is the database which contains all the (document, index) pairs the client has stored at the server. The client and the server interact as follows:
  1. The client first decrypts the encrypted search results $\{[\![R_d]\!] \mid d \in \mathcal{D}\}$, and gets to know which are the matched documents.
  2. The client decides a subset of the matched documents, and runs a private information retrieval (PIR) protocol (e. g. [69, 142, 146]) with the server to retrieve the documents.

For efficiency reasons, in the Retrieve algorithm, the client can select the desired documents and directly tell the server which documents she wants.

3.4 SECURITY ANALYSIS

With respect to the proposed SDR scheme, it is clear that the SearchIndex algorithm always returns 1 if $u(d)$ satisfies f_s and 0 otherwise. Hence, the soundness property is achieved given that the PIR protocol used in the Retrieve algorithm is also sound. Next, we summarize the security of the SDR scheme.

THEOREM 3.1 The proposed SDR scheme in Section 3.3.2 is secure under Definition 3.1 given that the adopted symmetric homomorphic encryption scheme is IND-CPA secure [96] and the PIR protocol in the Retrieve algorithm is secure [69].

Sketch of Proof. Suppose that an attacker has advantage $\epsilon_0 = |\Pr[b = b'] - \frac{1}{2}|$ in the attack game, defined in Figure 3.2. More precisely, we let $\epsilon_0 = |\epsilon_{0,0} + \epsilon_{0,1} - \frac{1}{2}|$, where $\epsilon_{0,0} = \Pr[b = b'|b = 0]$ and $\epsilon_{0,1} = \Pr[b = b'|b = 1]$. Let the faithful execution of the attack game be denoted as Game_0.

Now, we consider a new game, denoted as Game_1. In this game, we assume that the attacker performs the same as in Game_0 except that it always sets $\mathcal{ID}_1$ to be identical to $\mathcal{ID}_0$ in the Retrieve algorithm execution. In this game, let the attacker's advantage be $\epsilon_1 = |\epsilon_{1,0} + \epsilon_{1,1} - \frac{1}{2}|$, where $\epsilon_{1,0} = \Pr[b = b'|b = 0]$ and $\epsilon_{1,1} = \Pr[b = b'|b = 1]$. Due to the difference between Game_1 and Game_0, we have the following: $\epsilon_{0,0} = \epsilon_{1,0}$ and $|\epsilon_0 - \epsilon_1| \leqslant |\epsilon_{0,1} - \epsilon_{1,1}|$. Note that $\epsilon_{0,1}$ and $\epsilon_{1,1}$ are the probabilities that the attacker outputs 1 respectively, after adaptively executing a sequence of the Retrieve algorithm. For each execution of the Retrieve algorithm, the attacker sets $\mathcal{ID}_1$ to run the PIR protocol in the case of $\epsilon_{0,1}$ and sets $\mathcal{ID}_0$ to run the PIR protocol in the case of $\epsilon_{1,1}$. Different from the security definition of PIR protocol, in our setting the attacker outputs

a guess after executing a sequence of algorithm executions. However, with a standard reduction technique, it is straightforward to prove that the difference between $|\epsilon_{0,1} - \epsilon_{1,1}|$ and the advantage of winning a PIR security game is negligible. This leads to the conclusion that $|\epsilon_{0,1} - \epsilon_{1,1}|$ is negligible given that the PIR protocol used in the Retrieve algorithm is secure.

Next, we need to evaluate the probability ϵ_1 in Game_1. In this game, the challenger will retrieve the same set of documents regardless of the value of b. Therefore, the attacker's guess will be independent from the executions of the Retrieve algorithm. Having made this clear, then Game_1 is equivalent to a indistinguishability game for the encryption scheme, in which the attacker is asked to distinguish the ciphertexts of two vectors of plaintexts which can be adaptively chosen. It is well known that if an encryption scheme is IND-CPA secure then the advantage is negligible in the above case. In fact, this can be proven by a simple reduction on the vector length, but we omit the details here. As a conclusion, the probability ϵ_1 is negligible given that the encryption scheme is IND-CPA secure.

To sum up, we have informally shown that both ϵ_1 and $|\epsilon_0 - \epsilon_1|$ are negligible. As a result, ϵ_0 is negligible, and the theorem follows. □

In the proposed SDR scheme, if the client directly retrieves the matched documents without using a PIR protocol in the Retrieve algorithm, then the scheme achieves the relaxed security under Definition 3.2 given that the encryption scheme is IND-CPA secure. The intuition is very straightforward based on the fact that all operations in the search are carried out in the ciphertext domain using the homomorphic properties of the encryption scheme.

THEOREM 3.2 The proposed SDR scheme without using PIR protocol in the Retrieve algorithm achieves index privacy and trapdoor privacy under Definition 3.2 given that the adopted symmetric homomorphic encryption scheme is IND-CPA secure [96].

3.5 ADAPTATIONS OF THE PROPOSED SDR SCHEME

In the previous section, we described an SDR scheme and analyzed its security. Besides supporting equality test predicates, the scheme can be adapted to support a number of useful search features, including aggregating search results, supporting conjunctive keyword search queries, advanced keyword search, search with keyword occurrence frequency, and search based on inner product. Moreover, based on the same analysis in Section 3.4, all variants in this section are still secure in our security model. We also show that it is straightforward to adapt the proposed SDR scheme to the asymmetric setting or multi-user setting.

3.5.1 *Aggregating Search Results*

In the proposed scheme, the server has to send back an $[\![R_d]\!]$ for each document. If the symmetric BV scheme [54] is used in the scheme, to reduce the communication complexity, we can transform (depending on the degree α of the polynomials) up to α ciphertexts that encode α bits separately, into a single ciphertext C_p [121] (intuitively, it is shown in Figure 3.4). For a detailed

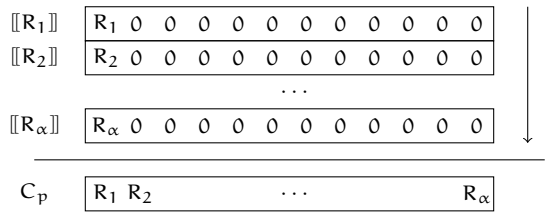


Figure 3.4: Aggregating Search Results

description of the BV-scheme and the used variables, we refer the reader to Section 3.6.1. The *packed* ciphertext is calculated by:

$$C_p = \left(\sum_i c_{0,i} x_i, \sum_i c_{1,i} x_i \right).$$

This means, for a collection of 1000 documents and using a 1024 degree polynomial, the server has to send back only one ciphertext instead of 1000.

3.5.2 *Conjunctive Keyword Search*

To support conjunctive keyword search queries for any number of keywords, we propose a variant of our SDR scheme. The Trapdoor algorithm needs to be changed slightly, while other algorithms stay basically the same. For conjunctive keyword search, the predicate set can be denoted as $\mathcal{F} = \{f_{\mathcal{W}'} \mid \mathcal{W}' \subseteq \mathcal{W}\}$. For any document d, $u(d)$ satisfies $f_{\mathcal{W}'}$ if and only if $u(d) \subseteq \mathcal{W}'$.

- $\mathsf{Trapdoor}(K, f_{\mathcal{W}'})$. With the key K and a predicate $f_{\mathcal{W}'}$, it does the following:
  1. Construct $t_{f_{\mathcal{W}'}} = (t_{f_{\mathcal{W}'}}[1], t_{f_{\mathcal{W}'}}[2], \ldots, t_{f_{\mathcal{W}'}}[m])$. For every keyword $s_i \in \mathcal{W}$, the value of $t_{f_{\mathcal{W}'}}[i]$ is set to be 1 if $s_i \in \mathcal{W}'$ and 0 otherwise.
  2. Output the trapdoor $T_{f_{\mathcal{W}'}} = ([\![t_{f_{\mathcal{W}'}}[1]]\!], [\![t_{f_{\mathcal{W}'}}[2]]\!], \ldots, [\![t_{f_{\mathcal{W}'}}[m]]\!])$.

As a result of the modification, the output of a $\mathsf{SearchIndex}(T_{f_{\mathcal{W}'}}, \mathcal{I}_d)$ query tells the client how many keywords in the trapdoor appear in the index $\mathcal{I}_d$.

3.5.3 *Advanced Keyword Search*

In some application scenarios, the client may care about some keywords more than others, which implies that it is desirable to allow the client to put a weight on each keyword in the trapdoor. To do so, we propose another variant of our SDR scheme. The Trapdoor and Retrieve algorithms need to be changed slightly, while other algorithms stay basically the same. For this variant, the predicate set can be denoted as $\mathcal{F} = \{f_{\mathcal{W}'} \mid \mathcal{W}' \subseteq \mathcal{W}\}$, as specified in Section 3.5.2.

- $\mathsf{Trapdoor}(K, f_{\mathcal{W}'})$. With the key K and a predicate $f_{\mathcal{W}'}$, it does the following:
  1. Construct $t_{f_{\mathcal{W}'}} = (t_{f_{\mathcal{W}'}}[1], t_{f_{\mathcal{W}'}}[2], \ldots, t_{f_{\mathcal{W}'}}[m])$. For every keyword $s_i \in \mathcal{W}$, the value of $t_{f_{\mathcal{W}'}}[i]$ is set to be 2^{i-1} if $s_i \in \mathcal{W}'$ and 0 otherwise.
  2. Output the trapdoor $T_{f_{\mathcal{W}'}} = ([\![t_{f_{\mathcal{W}'}}[1]]\!], [\![t_{f_{\mathcal{W}'}}[2]]\!], \ldots, [\![t_{f_{\mathcal{W}'}}[m]]\!])$.

- $\mathsf{Retrieve}(K, \{[\![R_d]\!] \mid d \in \mathcal{D}\}; \mathcal{D})$. Here, $\mathcal{D}$ is the database which contains all the (index, document) pairs the client has stored at the server. The client and the server interact as follows:
  1. The client first decrypts the encrypted search results $\{[\![R_d]\!] \mid d \in \mathcal{D}\}$. For every document d, the client can recover which keywords are contained in the index (by writing R_d in a binary form, if the i-th bit is 1 then s_i is contained in the index). The client can then add weights on the keywords and decide which documents to retrieve.
  2. The client and the server run a PIR protocol for the client to retrieve the documents.

By letting the client know exactly, which of several keywords satisfy the search, the client is able to run multiple queries at once using only one trapdoor.

3.5.4 *Search with Keyword Occurrence Frequency*

In practice, a search query may rank the relevance of a document based on not only whether some keywords are contained but also the occurrence frequency of these keywords in the documents. The proposed scheme can be modified to support such a requirement. To do so, we proposed another variant of the proposed SDR scheme. The $\mathsf{BuildIndex}$ algorithm needs to be changed slightly, while other algorithms stay basically the same. For this variant, the predicate set is still the equality test one.

- $\mathsf{BuildIndex}(K, d)$. With the key K and a document d, it does the following:
  1. Generate the list of distinct keywords, namely $u(d)$.
  2. Construct a plaintext index for d, denoted as $I_d = (I_d[1], I_d[2], \ldots, I_d[m])$. The bit $I_d[i]$ is set to be the occurrence frequency of s if $s \in u(d) = w_i \in \mathcal{W}$; otherwise, the $I_d[i]$ is set to be 0.
  3. Generate $[\![I_d]\!] = ([\![I_d[1]]\!], [\![I_d[2]]\!], \ldots, [\![I_d[m]]\!])$.
  4. Output the index $\mathcal{I}_d = [\![I_d]\!]$.

In this variant, the value of a $\mathsf{SearchIndex}(T_{f_s}, \mathcal{I}_d)$ query tells the client the occurrence of the keyword s in the document d, and then the client can decide which documents to retrieve accordingly.

3.5.5 *Search based on Inner Product*

Inner product predicates can allow complex evaluations on conjunctions, subsets and ranges [43] as well as disjunctions, polynomial evaluations and CNF/DNF formulas [112]. To support search based on inner product, we propose another variant of our SDR scheme. The $\mathsf{BuildIndex}$ and $\mathsf{Trapdoor}$ algorithms need to be changed slightly, while other algorithms stay basically the same. For this variant, the predicate set can be denoted as $\mathcal{F} = \{f = (f[1], f[2], \cdots, f[m]) | f[i] (1 \leqslant i \leqslant m) \in \mathbb{N}\}$.

- $\mathsf{BuildIndex}(K, d)$. With the key K and a document d, it does the following:
  1. Generate the list of distinct keywords, namely $u(d)$.
  2. Construct a plaintext index for d, denoted as $I_d = (I_d[1], I_d[2], \ldots, I_d[m])$. The value $I_d[i]$ is set to be s if $s = w_i$; otherwise, the $I_d[i]$ is set to be 0.

3. Generate $[\![I_d]\!] = ([\![I_d[1]]\!], [\![I_d[2]]\!], \ldots, [\![I_d[m]]\!])$.
4. Output the index $\mathcal{I}_d = [\![I_d]\!]$.

- $\mathsf{Trapdoor}(K, f)$. With the key K and a predicate f, the algorithm outputs the trapdoor $T_f = ([\![f[1]]\!], [\![f[2]]\!], \ldots, [\![f[m]]\!])$.

As a result of the modification, the output of a $\mathsf{SearchIndex}(T_f, \mathcal{I}_d)$ query tells the client the inner product of f and the keyword vector in the index $\mathcal{I}_d$.

3.5.6 *Multi-User Variant (adaption to asymmetric setting)*

In some application scenarios, it may be desirable that multiple users are able to write new data to an existing database as in the case of PEKS [47]. The proposed SDR scheme can be extended straightforwardly to meet the requirement. In the Keygen algorithm, the client generates a public/private key pair for a homomorphic public key encryption scheme, such as the public key version of the BV scheme [54]. In the algorithms BuildIndex, Trapdoor, and SearchIndex, the encryptions are done with the client's public key. The Retrieve algorithm stays the same. In the extended scheme, everyone can generate searchable indexes based on the client's public key. However, only the client with the private key is able to decrypt the search results which are always encrypted under the client's public key. Thus, compared with other similar schemes in the asymmetric setting such as PEKS [47], the extended scheme does not suffer from the inherent offline keyword recovery attacks [58, 188] (cf. Section 2.4.1). Without the client's secret key, the server cannot get the output of a search.

3.5.7 *Updates*

The proposed SDR scheme allows efficient updates, like most SSE schemes. A user can update the indexes in the sense that she can add and delete documents without revealing information. Only the number of documents processed is leaked. To add a document, the BuildIndex algorithm is run, and the index and encrypted document are sent to the server. To delete documents from the server, the index $[\![I_d]\!]$ and the related encrypted document can be removed from the server. The scheme also allows updating of the supported search words. To add search support for a new keyword, we simply add the keyword(s) to the pre-built keyword dictionary. All newly added documents can use the new keyword(s). To support new keywords for existing documents, their indexes have to be rebuilt.

3.6 PERFORMANCE ANALYSIS

In this section, we adapt the lattice-based symmetric encryption scheme by Brakerski and Vaikuntanathan (BV) [54] to our proposed solution and explain our choice of parameters. We then show our implementation results and discuss some optimizations for the implementation. Note that our implementation focuses on the SearchIndex algorithm, in an attempt to demonstrate the efficiency differences between the proposed SDR scheme and existing SSE schemes.

The performance of the Retrieve algorithm is also an efficiency concern for SDR schemes depending on the document size and database size, because it will require PIR protocols. The performance of PIR protocols is currently an

ongoing research topic for the community, and researchers have shown that such protocols can actually be practical [142]. Nonetheless, in order to achieve query result privacy, such an interactive algorithm is inevitable. We leave the comprehensive performance investigation of the proposed SDR scheme to be a future work.

3.6.1 *Adaption of the Symmetric BV Scheme*

In this subsection we denote scalars in plain and vectors in bold. We write $x \xleftarrow{R} X$ when we mean that x is chosen at random from the distribution X. The scheme uses the following parameters:

- the dimension α, which is a power of 2,
- the modulus q, which is a prime such that $q \equiv 1 \pmod{2\alpha}$
- the cyclotomic polynomial $f(x) = x^{\alpha} + 1$,
- the error distribution χ over the ring $R_q = \mathbb{Z}_q[x]/\langle f(x)\rangle$
- ciphertext degree D (supports $D-1$ multiplications),
- number of supported additions A,
- message space $t < q$, which is prime,
- error parameter σ (standard deviation of the discrete Gaussian error distribution).

All parameters are chosen in such a way to guarantee correctness and security of the scheme. For correctness the BV scheme requires:

$$q \geqslant 4 \cdot (2t\sigma^2\sqrt{\alpha})^D \cdot (2\alpha)^{(D-1)/2} \cdot \sqrt{A}.$$

Note that D is the ciphertext degree and not the number of supported multiplications [121]. The encryption scheme consists of the following algorithms. We simplified the Mul and Add algorithms to support one multiplication followed by several additions:

- $\mathsf{SH.Keygen}(1^{\kappa})$: Sample a ring element $s \xleftarrow{R} \chi$ and set the secret key $sk := s$. (If we only care about homomorphism, sampling $s \xleftarrow{R} R_q$ is sufficient.)
- $\mathsf{SH.Enc}(sk, m)$: We encode our message as a degree α polynomial with coefficients in $\mathbb{Z}_t$. To encrypt, sample $a \xleftarrow{R} R_q$ and $e \xleftarrow{R} \chi$ and output the ciphertext $\mathbf{c} = (c_0, c_1) \in R_q^2$ where $c_1 = -a$ and $c_0 = as + te + m$.
- $\mathsf{SH.Mul}(\mathbf{c}, \mathbf{c}')$: Given the two ciphertexts $\mathbf{c} = (c_0, c_1)$ and $\mathbf{c}' = (c_0', c_1')$ output the ciphertext vector $\mathbf{c}_{\text{mul}} = \mathbf{c} \cdot \mathbf{c}' = (c_0c_0', c_0c_1' + c_0'c_1, c_1c_1')$ using polynomial multiplication.
- $\mathsf{SH.Add}(\mathbf{c}, \mathbf{c}')$: Given two ciphertexts $\mathbf{c} = (c_0, c_1, c_2)$ and $\mathbf{c}' = (c_0', c_1', c_2')$ output the ciphertext vector $\mathbf{c}_{\text{add}} = \mathbf{c} + \mathbf{c}' = (c_0 + c_0', c_1 + c_1', c_2 + c_2') \in R_q^3$ which is calculated by coordinate-wise vector addition of the ciphertext vectors.
- $\mathsf{SH.Dec}(sk, \mathbf{c})$: To decrypt a ciphertext, first define the secret key vector $\mathbf{s} = (1, s, s^2, \ldots, s^D) \in R_q^{D+1}$, compute $\langle \mathbf{c}, \mathbf{s}\rangle = \sum_{i=0}^{D} c_i s^i \in R_q$, and output the message $m = \langle \mathbf{c}, \mathbf{s}\rangle \pmod{t}$.

α	$\lceil \lg(q) \rceil$	$\lg(T)$	WC $\lvert c \rvert$	MUL	ADD
256	14	64	896 B	410 E-06	11 E-06
512	20	107	2.5 kB	454 E-06	21 E-06
1024	33	134	8.25 kB	2.8 E-03	72 E-06

Table 3.1: Implementation results for the parameters mentioned in Section 3.6.2. The degree of the polynomials is denoted by α, $\lceil \lg(q) \rceil$ is the bit size of q, and $\lg(T)$ is the logarithm of the runtime of the distinguishing attack from [126]. WC $\lvert c \rvert$ is the worst case ciphertext size and the last two columns describe the time in seconds, that is required for a single multiplication or addition, respectively.

3.6.2 *Choice of BV Parameters and Implementation*

We choose our parameters for the symmetric BV scheme based on our needs, and also take into account the work of Lauter et al. [121] which assessed the security against the decoding attack [126] and the distinguishing attack [132]. We use the following parameters: $D = 2, A = 100, t = 2, \sigma = 8$. With these fixed parameters, we calculate the flexible parameters as seen in Table 3.1. We made experiments with smaller q and larger A (up to 1000) and still ended up with correct results.

We implemented the scheme in C/C++ using FLINT, namely Fast Library for Number Theory [99]. We tested the code on an Intel Xeon CPU X5677@3.47 GHz running linux 2.6.37-sabayon x86_64. In this situation, our results for degree 512 polynomials show that an addition (after a multiplication) takes 21×10^{-6} seconds and a multiplication takes 454×10^{-6} seconds.

At this moment, we only have a single threaded implementation of our scheme. The homomorphic multiplication operation has to calculate four independent polynomial multiplications, which can be done in parallel. This will decrease the computation time significantly. The same is applicable for the addition operation, which uses three independent polynomial additions. These additions can also be easily done in parallel. Another optimization, which is mentioned by Lauter et al. [121] is to use the Fast Fourier Transformation (FFT) to speed up computations. This has already been considered in SWIFFT [131]. Due to the choice of parameters ($\mathbb{Z}_q \mod x^\alpha + 1$, where α is a power of 2 and $q = 1 \pmod{2\alpha}$) the FFT can be computed more efficiently.

To compare our scheme with others, we also implemented a type A symmetric prime order pairing, using the PBC [130] library. On the same machine, a single pairing operation takes 5.8×10^{-3} seconds.

3.6.3 *Performance of the Proposed SDR Scheme*

We now consider the efficiency of the proposed SDR scheme, where the efficiency is measured in terms of the computation, communication and space complexities.

In Table 3.2, the first column shows the number of supported search keywords. The second column shows the number of documents stored on the server. The third and fourth columns show the number of required additions

Keywords	Docs	Add.	Mul.	WC $\lvert T_{f_s}^{256}\rvert$	WC $\lvert T_{f_s}^{512}\rvert$
100	1000	99,000	100,000	87 kB	250 kB
100	5000	495,000	500,000		
250	1000	249,000	250,000	218.75 kB	625 kB
250	5000	1,245,000	1,250,000		

Table 3.2: Number example of the computational complexity. The last two columns describe the worst case trapdoor size considering the use of 256 or 512 degree polynomials. This is also the size of an encrypted index for a single document.

Keywords	Docs	Pairings	GE(CT)	GE(T)
100	1000	202,000	202	202
100	5000	1,010,000		
250	1000	502,000	502	502
250	5000	2,510,000		

Table 3.3: Number example of the computational complexity of the SSW scheme. GE(CT) and GE(T) shows the number of group elements per ciphertext and trapdoor, respectively.

and multiplications for a search over the database. The last two columns show the worst case trapdoor size, which has to be transmitted, depending on the degree of the polynomial. Based on the performances of the symmetric BV scheme, for a document set of size 1000 with a keyword set of size 100, a search takes 47 seconds. For a document set of size 5000 with a keyword set of size 250, a search takes around 10 minutes. After applying techniques such as parallel computation and optimization mentioned in Section 3.6.2, we expect the search speed of the second scenario can be improved into less than 1 minute. The result of a query is of size $\left\lceil \frac{\text{Docs}}{\alpha} \right\rceil \cdot |c|$, where α is the degree of the polynomial and $|c|$ the size of a single ciphertext according to Table 3.1. Note that the worst case trapdoor size is also the worst case index size, that has to be stored on the server for a single document. Table 3.3 shows the computational complexity of the SSW [167] scheme in terms of pairings that have to be computed per search. The last two columns show the number of group elements per ciphertext and trapdoor, respectively.

In Table 3.4, we compare our scheme to other schemes. The first three rows describe the asymptotic comparison from the perspective of computational complexity of the algorithms. Our Trapdoor algorithm is a constant time operation, since it requires only a table lookup which can be done using a trivial hash function as index. Our BuildIndex algorithm has to process each distinct keyword per document. Thus the complexity is $O(n|\Delta|)$. To search, the server has to perform a constant number (namely, m) of operations for all n documents. Thus the server load is $O(n)$. The server has to store one index per document, so the index size is $O(n)$. The fourth and fifth rows of the table

Properties	SWP [171]	Goh [92]	SSE [75]	SSW [167]	Our [3]
Compute Trapdoor	$O(1)$	$O(1)$	$O(1)$	$O(v)$	$O(a)$
Compute Indexes	$O(nv)$	$O(n\|\Delta\|)$	$O(n\|\Delta\|)$	$O(nv)$	$O(n\|\Delta\|)$
Search Indexes	$O(nv)$	$O(n)$	$O(\|D(s)\|)^*$	$O(nv)$	$O(n)$
Conjunctive Search	No	No	No	Yes	Yes
Advanced Search Features	No	No	No	Yes	Yes
Full Security	No	No	No	No	Yes

Table 3.4: Computational performance of different search schemes, where n is the number of documents in the database, v the number of words per document, and a is the number of keywords in the trapdoor. The number of distinct words per document is denoted by $|\Delta|$ and $|D(s)|$ denotes the number of documents containing the keyword w. The asterisk * refers to the use of a so-called *FKS dictionary* introduced by Fredman et al. [83], which reduces the lookup time to $O(1)$.

Scheme	small (100/1000)	large (250/5000)
Our ($\alpha = 512$)	47 s	9.9 m
Our ($\alpha = 1024$)	4.8 m	59.8 m
SSW (prime)	19.5 m	4.0 h
SSW (composite)	16.3 h	8.4 d

Table 3.5: Comparison of the search times of our scheme and Shen et al. scheme. The SSW (prime) column shows the SSW scheme under the assumption that it uses prime order pairing. The SSW (composite) shows a calculated value.

compare the expressiveness of search queries, and the last row compares the security of the schemes.

It is worth noting that the above computational complexity comparison is asymptotic. In practice, different operations make a great difference for the real speed number. In our case, the operations are polynomial additions and multiplications, which are much more efficient than other operations such as pairings. For example, for the SSW scheme [167], a search query in the database needs $n(2m+2)$ composite order pairings[2]. As shown in Table 3.5, for the proposed scheme, given a database set of size 1000 with a keyword set of size 100, a search takes 47 seconds. However, for the same setting, a search takes 58,580 (i. e., $1000 \times 202 \times 0.0058 \times 50$) seconds ($\approx$ 16.3 hours) for the SSW scheme on the same machine, which is 1247 $\times$ slower than our proposed scheme. These numbers are based on the performance of a type A symmetric prime order pairing using the PBC [130] library and the fact that a pairing on a 1024-bit composite order elliptic curve can be 50 times slower than in a prime order group [84]. For our comparison this is a conservative estimate since the SSW scheme uses composite order groups, where the order is the product of four primes.

2 For a fair comparison of the schemes we assume a pre-build dictionary of m keywords as in our construction.

3.7 CONCLUSION

We have proposed the concept of selective document retrieval (SDR) as a cryptographic primitive for outsourcing encrypted data. Compared with symmetric searchable encryption (SSE), an SDR scheme can potentially provide more flexible services and better security guarantees. We described a security model to cover three types of privacy properties, including index privacy, trapdoor privacy, and query result privacy. We show that a secure SSE scheme cannot be trivially extended to provide query result privacy. Therefore, we have proposed a construction for SDR based on homomorphic encryption and the index construction method by Chang and Mitzenmacher [64]. The construction offers a very flexible framework, and can be adapted very easily to support many useful search features. To evaluate the performance, we have implemented the search algorithm in C based on the symmetric Brakerski-Vaikuntanathan (BV) scheme [54], and the results show that it is more efficient than a solution based on existing SSE schemes. In Section 3.6, we have evaluated only the search algorithm of the proposed SDR scheme, but a comprehensive performance study is still needed, in particular for the Retrieve algorithm. The performance of PIR protocols is currently an ongoing research topic for the community, and researchers have shown that such protocols can actually be practical [142]. We leave a full discussion of the issue as a future work.

In this chapter we answered our first research question by showing how to construct an efficient search pattern hiding scheme using approach A1. Our proposed scheme relies on client interaction which is regarded as an efficiency drawback in some applications. It is still an open question whether we can construct reasonably efficient search pattern hiding schemes without client interaction.

4 DISTRIBUTED SEARCHABLE SYMMETRIC ENCRYPTION

In Chapter 3 we proposed a search pattern hiding scheme, using approach A1, that relies on client interaction. In some application scenarios client interaction might be considered as a drawback. In this chapter, we introduce the concept of *distributed* SSE (DSSE), which uses a query proxy in addition to the storage provider to distribute the search on the encrypted data. The search pattern is hidden using approach A2. We give a probable secure construction that combines an inverted index approach (for efficiency) with scrambling functions used in private information retrieval (PIR) (for security). The proposed scheme, which is entirely based on XOR operations and pseudo-random functions, is efficient and does not leak the search pattern. For instance, a secure search in an index over one million documents and 500 keywords is executed in less than 1 second.

4.1 INTRODUCTION

Searchable Symmetric Encryption (SSE) allows a client to outsource data in encrypted form to a semi-honest server/storage provider (like a cloud provider), such that the encrypted data remains searchable without decrypting and without the server learning the contents of the data or the words being searched for. In most cases this is achieved by introducing a searchable encrypted index, which is stored together with the encrypted data (e. g., documents) on a server. To enable the server to query the data, the client creates a trapdoor which allows the server to do the search on behalf of the client. Practical SSE schemes try to make this search process as efficient as possible, which usually comes at the cost of leaking (sensitive) information, such as the search pattern, i. e., the information if two trapdoors were generated for the same keyword [76].

Over the last decade there has been active research in SSE [3, 51, 64, 76, 92, 108, 109, 119, 167, 171, 177]. The majority of the schemes has a search complexity which is linear in the number of documents stored on the server, since one index per document has to be searched. Some schemes allow a more efficient search, e. g., by using an *inverted* index [76, 108, 109, 177], which is an index per distinct keyword in the database. This reduces the search complexity to (at least) the number of distinct keywords in the document collection. However, the reduced complexity usually comes at the cost of reduced security.

A limitation of most previous SSE schemes is the leakage of the search pattern [109]. Revealing the search pattern in SSE schemes is a serious problem, as it allows an attacker to perform statistical analysis on the occurrence frequency of each query. This allows an attacker to gain knowledge on the underlying plaintext keywords, rendering the encryption scheme less useful (as is convincingly demonstrated by Liu et al. [128]). The problem of leaking the search pattern is not only recognized in the setting of SSE, but also in the setting of predicate encryption [167].

Kantarcioğlu and Clifton [110] were the first to prove that in a *single* server setting, a cryptographically secure SSE scheme needs to process the whole database per query to protect sensitive information (including the search pat-

tern), thus being inefficient in practice. The authors propose the use of a fully trusted party (a trusted hardware module [13, 14] in their case) to make a cryptographically secure SSE scheme efficient and sketch a construction for relational databases.

To obtain more efficient SSE schemes or to realize more complex queries than just keyword queries, a common approach is to split the server into a semi-honest storage provider and a semi-honest (query) proxy [30, 51, 74, 155, 157, 181, 182]. Unfortunately, all of these schemes leak the search pattern.

In this chapter we propose an efficient construction of an SSE scheme in the above setting that hides the search pattern. The main idea behind our construction is to distribute the search on the encrypted data to the storage provider and the query proxy. Therefore, we call our new scheme a *distributed* Searchable Symmetric Encryption (DSSE) scheme. Our DSSE scheme achieves its efficiency due to distributed computation and the use of efficient primitives like XOR and pseudo-random functions only. We use an inverted index, which is a common approach to reduce the search complexity in databases. The ordinary use of an inverted index directly leaks the search pattern. To hide the search pattern, we make use of techniques used in oblivious RAM [94, 143, 144] (ORAM) and private information retrieval [15, 70] (PIR), which solve this problem by continuously re-shuffling the index as it is being accessed. In this way, neither the storage provider nor the query proxy can tell which record was accessed and thus the search pattern of the scheme remains hidden.

We make the following contributions:

1. We formally define the concept of DSSE and its security (Section 4.2).
2. We propose a simple and efficient search pattern hiding DSSE construction (Section 4.3).
3. We prove the security of our DSSE scheme in the semi-honest model and show that it only leaks the access pattern and nothing more (Section 4.4).
4. We implement the core components of our scheme and analyse its performance (Section 4.5).
5. We discuss the security implications of colluding servers, and propose a highly efficient SSE construction resulting from such a collusion (Section 4.6).
6. We prove adaptive semantic security for the SSE scheme, as defined by Curtmola et al. [76] (Section 4.6.2).
7. We give an analysis of theoretical performance of the SSE scheme.

4.2 DISTRIBUTED SEARCHABLE SYMMETRIC ENCRYPTION

In this section, we formally define the new notion of *distributed* searchable symmetric encryption (DSSE) and its security. As such a scheme involves a client C, a storage provider (*SP*) and a query proxy (*QP*), we formulate it as a protocol between these three parties.

NOTATION. Throughout this chapter, we use the following notation. Let $\mathcal{D} = \{d_1, \ldots, d_n\}$ be a set of n files (e. g., documents). Let $\mathcal{W} = \{w_1, \ldots, w_m\}$ be a pre-built dictionary of m keywords. Given a document d, let $u(d)$ denote the set of distinct keywords in d. Given a tuple t, we refer to the i-th entry of

t as t[i]. The encrypted index is denoted by $\mathcal{I} = \{I_{w_1}, \ldots, I_{w_m}\}$. $\mathcal{J}$ denotes a re-encrypted index and $\mathcal{I}'$ a re-encrypted and permuted index. The keyword used in a query we denote by $s \in \mathcal{W}$. The set of document identifiers of all documents in $\mathcal{D}$ containing the query keyword s is written as $\mathcal{D}(s)$. An element a randomly chosen from a set A is denoted by $a \xleftarrow{\$} A$. For two distribution ensembles X and Y, we denote computational indistinguishability by $X \equiv_c Y$.

DEFINITION 4.1 (Distributed Searchable Symmetric Encryption Scheme) A *Distributed Searchable Symmetric Encryption (DSSE)* scheme for a set of keywords $\mathcal{W} = \{w_1, \ldots, w_m\}$ is a protocol between three parties: a client C, a storage provider *SP* and a query proxy *QP*, and consists of the following four probabilistic polynomial time (PPT) algorithms:

- $(K_C, K_1, K_2) \longleftarrow \mathsf{Keygen}(\lambda)$: This algorithm is run by the client C, takes a security parameter λ as input, and outputs a secret key K_C to the client C and secret keys K_1 and K_2 to *SP* and *QP*, respectively.

- $\mathcal{I} = (\mathcal{I}_1, \mathcal{I}_2) \longleftarrow \mathsf{BuildIndex}(K_C, \mathcal{D})$: This algorithm is run by the client C, takes a key K_C and a set of documents $\mathcal{D}$ as input, and outputs an encrypted index $\mathcal{I}_1$ to *SP* and $\mathcal{I}_2$ to *QP*.

- $T^s = (T_1^s, T_2^s) \longleftarrow \mathsf{Trapdoor}(K_C, s)$: This algorithm is run by the client C, takes a key K_C and a query keyword $s \in \mathcal{W}$ as input, and outputs a trapdoor T_1^s to *SP* and trapdoor T_2^s to *QP*.

- $X \longleftarrow \mathsf{SearchIndex}(T^s, \mathcal{I} = (\mathcal{I}_1, \mathcal{I}_2), K_1, K_2)$: This algorithm is a protocol between *SP* and *QP*. *SP* provides $T_1^s, \mathcal{I}_1, K_1$ and *QP* provides $T_2^s, \mathcal{I}_2, K_2$ as input. The algorithm has a set of document identifiers X of documents in $\mathcal{D}$ as (public) output.

Additionally, we require a DSSE scheme to be *correct*, i. e., that for the set of keywords $\mathcal{W}$, any set of documents $\mathcal{D}$, all security parameter λ, all outputs $(K_C, K_1, K_2) \longleftarrow \mathsf{Keygen}(\lambda)$, $\mathcal{I} \longleftarrow \mathsf{BuildIndex}(K_C, \mathcal{D})$, $T^s \longleftarrow \mathsf{Trapdoor}(K_C, s)$ and all keywords $s, w \in \mathcal{W}$, it holds that:

$$\mathsf{SearchIndex}(T^s, \mathcal{I}, K_1, K_2) = \mathcal{D}(s),$$

where $\mathcal{D}(s)$ denotes the set of identifiers of all documents in $\mathcal{D}$ containing the keyword s. The sequence of document identifiers $\mathcal{D}(s)$ for consecutive keywords s is called the *access pattern*.

Suppose a client makes Q queries, while the i-th query queries for keyword $s_i \in \mathcal{W}$; so in total the client queries for $s_1, \ldots, s_Q \in \mathcal{W}$. To distinguish between the different trapdoors associated with these Q queries, we write T^{s_i} to denote a trapdoor for the i-th query (i. e., the client queries for the keyword s_i). We denote an admissible protocol run of a DSSE scheme, where the client performs Q queries, by Π^Q_{DSSE}. Formally, an admissible Q-query protocol run Π^Q_{DSSE} is defined as follows:

DEFINITION 4.2 (Admissible Q-query protocol run Π^Q_{DSSE}) Consider a DSSE scheme with keyword set $\mathcal{W}$, output (K_C, K_1, K_2) of $\mathsf{Keygen}(\lambda)$, and a document set $\mathcal{D}$. For a given $Q \in \mathbb{N}$, an *admissible* Q*-query protocol run* consists of one call of algorithm $\mathcal{I} \longleftarrow \mathsf{BuildIndex}(K_C, \mathcal{D})$, followed by Q calls of algorithm $T^{s_i} \longleftarrow \mathsf{Trapdoor}(K_C, s_i)$ for (possibly different) keywords $s_i \in \mathcal{W}$

for $i \in [1, Q]$, and another Q calls of algorithm $\mathsf{SearchIndex}(T^{s_i}, \mathcal{I}, K_1, K_2)$. We denote such a protocol run by Π^{Q}_{DSSE}.

4.2.1 *Security Model*

Following all previous works on SSE, we treat the client as a trusted party. Concerning *SP* and *QP*, we approach the security of a DSSE scheme by following the real-vs-ideal paradigm of secure multiparty computation [93, Ch. 7] in the semi-honest model. This means that we assume *SP* and *QP* to act honest-but-curious, i. e., they will follow all protocol steps honestly but may try to infer all kinds of information on other parties inputs or intermediate results beyond what the output of the DSSE scheme reveals. Moreover, we assume secure channels between any of the parties and that *SP* and *QP* do not collude.

In particular, this implies that only admissible Q-query protocol runs Π^{Q}_{DSSE} are performed (for $Q \in \mathbb{N}$). Now intuitively, since the protocol Π^{Q}_{DSSE} only has the access pattern $\mathcal{D}(s_1), \ldots, \mathcal{D}(s_Q)$ as public output to all participants, if a DSSE scheme is secure in the semi-honest real-vs-ideal paradigm, it leaks no information (including the search pattern) other than the access pattern. Following this paradigm [93, Ch. 7], we first define the ideal functionality of a DSSE scheme as follows:

DEFINITION 4.3 (Functionality $\mathcal{F}^{Q}_{\mathsf{DSSE}}$) Consider a DSSE scheme with keyword set $\mathcal{W}$, output (K_C, K_1, K_2) of $\mathsf{Keygen}(\lambda)$, and a document set $\mathcal{D}$. For $Q \in \mathbb{N}$, $\mathcal{F}^{Q}_{\mathsf{DSSE}}$ is the functionality that takes as input

- K_C and keywords $s_1, \ldots, s_Q$ from the client C,
- K_1 from the storage provider *SP*, and
- K_2 from the query proxy *QP*.

and outputs $\mathcal{D}(Q) := (\mathcal{D}(s_1), \ldots, \mathcal{D}(s_Q)$ to *all* the parties C, *SP* and *QP*.

Then, we say that a DSSE scheme is *secure* if any admissible Q-query protocol run Π^{Q}_{DSSE} (for any $Q \in \mathbb{N}$) privately computes the functionality $\mathcal{F}^{Q}_{\mathsf{DSSE}}$. Formally, this means:

DEFINITION 4.4 (Security) We say that a DSSE scheme is *secure*, if for any $Q \in \mathbb{N}$, the protocol Π^{Q}_{DSSE} privately computes the functionality $\mathcal{F}^{Q}_{\mathsf{DSSE}}$ between the three parties C, *SP* and *QP*, i. e., there exists a (PPT) simulator $\mathcal{S}$ such that

$$\{\mathcal{S}(K_1, \mathcal{D}(Q))\}_{K_C, s_1, \ldots, s_Q, K_1, K_2}$$
$$\equiv_c \{\mathsf{View}_{SP}(K_C, s_1, \ldots, s_Q, K_1, K_2)\}_{K_C, s_1, \ldots, s_Q, K_1, K_2}$$

and

$$\{\mathcal{S}(K_2, \mathcal{D}(Q))\}_{K_C, s_1, \ldots, s_Q, K_1, K_2}$$
$$\equiv_c \{\mathsf{View}_{QP}(K_C, s_1, \ldots, s_Q, K_1, K_2)\}_{K_C, s_1, \ldots, s_Q, K_1, K_2}$$

Note that it is sufficient to simulate the views of *SP* and *QP* separately as we do not consider any form of collusion between them. Recall that the client is treated as a trusted party who only provides inputs and so the security definition does not need to take the client's view into account.

4.3 THE PROPOSED DISTRIBUTED CONSTRUCTION

Recall that a DSSE scheme consists of three parties: a client C, a storage provider *SP* and a query proxy *QP*. Our proposed scheme uses an inverted index, that is, an index per distinct keyword in the database. Each index consists of a single bit per keyword per document. A plaintext index ι_w for keyword w is a bit string of length n, where n is the number of documents in the database. Each position $\iota_w[j]$ corresponds to a unique document, where j is a unique document identifier. If a document d_j contains the keyword w, then the j-th bit of ι_w is set to 1. Otherwise the bit is set to 0. To protect the plaintext index ι_w, it is encrypted, by a bitwise XOR operation (denoted as $\oplus$) with several keyed pseudo-random functions described below. Concerning the output of $\mathsf{Keygen}(\lambda)$, the key $K_C = (K_f, K_p)$ is only known by the client C, the key K_1 is a shared key and known by C and *SP*. Formally, K_1 is contained in K_C as a second component which we omit here for reasons of readability and just say that C knows both K_C and K_1. The second key K_2 for *QP* is empty in our proposed solution. We assume that the documents are encrypted by the client with some standard symmetric encryption algorithm using a key different from K_C. Since the document encryption is independent from our scheme it is not considered further. Our construction makes use of the following cryptographic primitives:

- $f(K_C, w)$: The function $f(K_C, w)$ takes a key K_C and a keyword w as input. It outputs a pseudo-random bit-string of length n.
- $g(K_1, w, r_1)$: The function takes as input a key K_1, a keyword w and a random value r_1. It outputs a pseudo-random bit-string of length n.
- $h(K_1, r_1)$: The function takes a key K_1, and a random value r_1 as input. The output is an n-bit pseudo-random string.
- σ_k: The keyed pseudo-random permutation σ_k describes a permutation on the set $[1, m]$. The evaluation of the permutation σ_k takes as input an element $x \in [1, m]$ and outputs its permuted position $\sigma_k(x) \in [1, m]$.
- $\pi(\mathcal{X}, \sigma_x)$: The function takes as input a set $\mathcal{X}$ of size $|\mathcal{W}|$ and a random permutation σ_x. It outputs a permuted set according to σ_x.

For ease of readability we will omit the keys K_C, K_1 and use $f_w, g_w(r_1)$ and $h(r_1)$ in the rest of this chapter to denote $f(K_C, w)$, $g(K_1, w, r_1)$ and $h(K_1, r_1)$, respectively.

4.3.1 *Our Construction*

Next, we describe the four algorithms of our proposed scheme, namely Keygen, BuildIndex, Trapdoor and SearchIndex. The key K_2, as well as the index $\mathcal{I}_2$ are empty in our construction and are thus omitted in the description.

- $(K_C, K_1, K_2) \longleftarrow \mathsf{Keygen}(\lambda)$: Given a security parameter λ, generate a key $K = (K_C = (K_f, K_p), K_1)$ for the pseudo-random functions. The key K_C is only known by C, the key K_1 is known by C and *SP*.
- $\mathcal{I} = (\mathcal{I}_1, \mathcal{I}_2) \longleftarrow \mathsf{BuildIndex}(K_C, \mathcal{D})$: With the key K_C, and a document collection $\mathcal{D}$, the algorithm does the following:

1. For all search keywords $w_i \in \mathcal{W}$:
   a) $\forall d_j \in \mathcal{D}$: set $\iota_{w_i}[j] = 1$, if $w_i \in u(d_j)$; otherwise $\iota_{w_i}[j]$ is set to 0.
   b) Encrypt the index ι_{w_i} as follows: $I_{w_i} = \iota_{w_i} \oplus f_{w_i}$.
2. Permute the index $\mathcal{I} = \pi(\{I_{w_i}\}, \sigma_{K_p})$ based on the client's key K_p.
3. Output the index $\mathcal{I}$ and send to *SP*.

- $T^s = (T_1^s, T_2^s) \longleftarrow \mathsf{Trapdoor}(K_C, s)$: With the key K, and a query keyword $s \in \mathcal{W}$, the algorithm selects three random values r_1, r_2, r_3 and sets $T_1^s = (r_1, r_2, r_3)$. Then, the algorithm generates the client's dictionary as $\mathcal{W}^c = \pi(\mathcal{W}, \sigma_{K_p})$. Next, the algorithm calculates the query dictionary $\mathcal{W}^q = \pi(\mathcal{W}^c, \sigma_{r_2})$ and looks up the current position $q_s(r_2)$ for the desired keyword s in the permuted keyword list $\mathcal{W}^q$. Generate the trapdoor $T_2^s = (q_s(r_2), k = f_s \oplus g_s(r_1) \oplus r_3)$. Output
$$T^s = (T_1^s = (r_1, r_2, r_3), T_2^s = (q_s(r_2), k = f_s \oplus g_s(r_1) \oplus r_3))$$
- $X \longleftarrow \mathsf{SearchIndex}(T^s, \mathcal{I} = (\mathcal{I}_1, \mathcal{I}_2), K_1, K_2)$: (*SP* provides T_1^s and *QP* provides T_2^s) The storage provider *SP* re-encrypts and permutes the index $\mathcal{I}$ for all $i \in [1, m]$ as follows:
$$\mathcal{J} = \{J_{w_i}\} = \{I_{w_i} \oplus g_{w_i}(r_1) \oplus h(r_1)\},$$
$$\mathcal{I}' = \pi(\mathcal{J}, \sigma_{r_2})$$
and sends $\mathcal{I}'$ to *QP*. *QP* stores $\mathcal{I}'$ as its current index and performs a table lookup for $q_s(r_2)$ on $\mathcal{I}'$ to obtain the right I'_s. *QP* then re-encrypts as follows:
$$\begin{aligned} I''_s &= I'_s \oplus k \\ &= (\iota_s \oplus f_s \oplus g_s(r_1) \oplus h(r_1)) \oplus (f_s \oplus g_s(r_1) \oplus r_3) \\ &= \iota_s \oplus h(r_1) \oplus r_3. \end{aligned}$$
I''_s is sent to *SP*, which can now decrypt $\iota_s = I''_s \oplus h(r_1) \oplus r_3$. The result ι_s encodes, whether a document satisfies the query keyword s or not. Depending on the client, *SP* sends either the matching document ids or directly the matching encrypted documents to C.

A standard work flow is as follows. A client C first runs the Keygen algorithm to generate the key K. To create a searchable index, C runs the BuildIndex algorithm which outputs the inverted index $\mathcal{I}$. Finally C stores the index $\mathcal{I}$ together with the encrypted documents on the storage provider *SP*.

Later on, when the client wants to retrieve some documents containing a search keyword $s \in \mathcal{W}$, it first runs the Trapdoor algorithm to generate the trapdoor $T^s = (T_1^s, T_2^s)$. C sends T_1^s to *SP* and T_2^s to *QP*. Then, *SP* and *QP* can run the SearchIndex algorithm. *SP* re-encrypts and permutes the index $\mathcal{I}$ with help of T_1^s and sends the new $\mathcal{I}'$ to *QP*. *QP* performs a table look-up and then re-encrypts the result using the key k inside T_2^s. The temporary result I''_s is sent to *SP*, which can now decrypt using T_1^s to obtain the plaintext index ι_s for the search keyword s. Finally, *SP* either sends the matching ids or the matching encrypted documents to the client.

By letting *SP* perform the re-encryption and permutation, *QP* receives a fresh index before each query. These indexes are indistinguishable from each

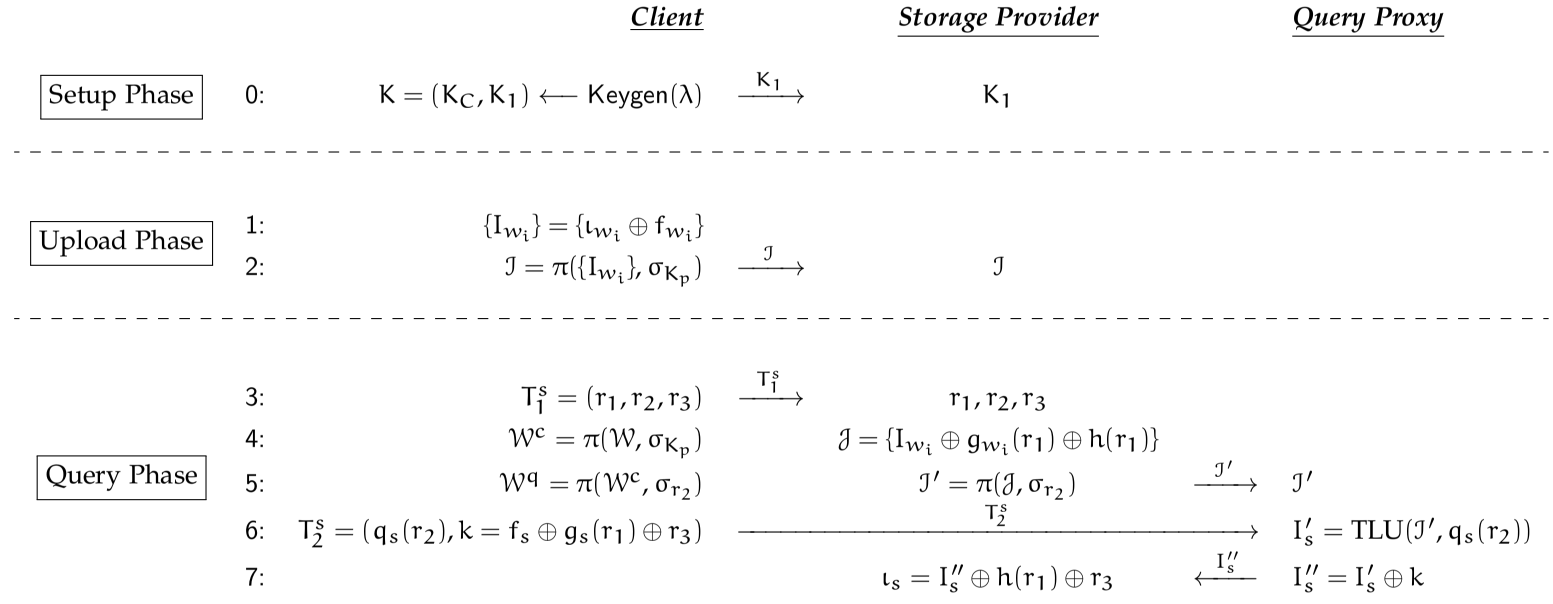


Figure 4.1: Simplified upload and search processes of our DSSE scheme. TLU denotes a table look-up. The document up- and download is omitted.

other and also from random. Thus the next query will not leak any information. To make the scheme more efficient, the client can choose another re-encryption policy, e. g., to trigger the re-encryption before he queries the same keyword twice. In this way, *SP* and *QP* can reduce the computational and communication complexity.

4.3.2 *Updates*

The proposed DSSE scheme allows efficient updates of the document collection, like most of the SSE schemes. A user can update the index by adding and deleting documents without revealing information. Only the number of documents processed is leaked. To add a document $j+1$, the BuildIndex algorithm is run and the new indexes $\iota_{w_i}[j+1]$ are encrypted and appended to the existing indexes. To delete a document d_x from the collection, the client sets the indexes $\iota_{w_i}[x]$ to 0, encrypts and sent them to *SP*.

4.4 SECURITY ANALYSIS

THEOREM 4.1 (Security) Our proposed DSSE scheme from Section 4.3 is secure with respect to Definition 4.4.

Proof. Let $Q \in \mathbb{N}$. By the Composition Theorem in the semi-honest model [93, Theorem 7.5.7], we can treat each step in protocol Π^{Q}_{DSSE} separately. We start by constructing a simulator $\mathcal{S}$ of *SP*'s view in each step of protocol Π^{Q}_{DSSE}. We then construct a simulator $\mathcal{S}$ of *QP*'s view of protocol Π^{Q}_{DSSE}.

STORAGE PROVIDER SP. In line 2 of Figure 4.1, *SP* learns the values I_w for all keywords $w \in \mathcal{W} = \{w_1, \ldots, w_m\}$. Since this value is computed as an XOR of the plaintext index ι_w and the n-bit output of the pseudo-random function f with key K_C and keyword w, the value I_w is computationally indistinguishable from a random n-bit string (recall that $\mathcal{D}$ contains $|\mathcal{D}| = n$ documents). Therefore, $\mathcal{S}$ can simulate these values with random n-bit strings.

Now, let $s_1, \ldots, s_Q$ denote the keywords that the client queries for. In line 4, for each of these keywords s_j ($j = 1, \ldots, Q$), the storage provider *SP* learns the three random bit-strings r_1, r_2, and r_3. These can be trivially simulated by $\mathcal{S}$ by choosing random strings.

Finally, in line 7, *SP* receives the value I''_{s_j} which equals $\iota_{s_j} \oplus h(K_1, r_1) \oplus r_3$. But the simulator $\mathcal{S}$ knows the key K_1 and the overall output $\mathcal{D}(Q) = (\mathcal{D}(s_1), \ldots, \mathcal{D}(s_Q))$ of functionality $\mathcal{F}^{Q}_{\mathsf{DSSE}}$ by definition, and since he created the random values r_1 and r_3 himself, he can simulate I''_s by simply computing $\iota_s \oplus h(K_1, r_1) \oplus r_3$. This can be done for each keyword s_j and so $\mathcal{S}$ successfully simulated the view of the storage provider *SP*.

QUERY PROXY QP. In line 5 of Figure 4.1, for each keyword w_i ($i = 1, \ldots, m$), *QP* learns the value/index $\mathcal{I}'$. But this index is computed as a pseudo-random permutation of the re-encrypted index $\mathcal{I} = \{I_{w_i} \oplus g_{w_i}(r_1) \oplus h(r_1)\}$, while every entry in $\mathcal{I}$ is indistinguishable from a random n-bit string. Therefore, for each keyword w_j, the index $\mathcal{I}'$ is indistinguishable from a random $(m \times n)$-bit matrix, which can be simulated by $\mathcal{S}$ as such.

Let $s_1, \ldots, s_Q$ denote the Q keywords that the client queries for. In line 6, for each of the keywords s_j for $j \in [1, Q]$, the query proxy *QP* learns the values $q_{s_j}(r_2)$ and k. Since $q_{s_j}(r_2)$ is an index position for keyword s_j after a pseudo-random permutation with function π with input $\mathcal{J}$ and the pseudo-random permutation based on the random value r_2, the value can be simulated, by choosing a random value between 1 and m. The value k is computed as an XOR of the n-bit outputs of the pseudo-random functions $f(K_C, s_j)$ and $g(K_1, s_j, r_1)$ and the random n-bit string r_3. The value k is thus indistinguishable from random and can be simulated by $\mathcal{S}$ with a random n-bit string. In total, this shows that $\mathcal{S}$ successfully simulates the view of the query proxy *QP*. □

4.5 PERFORMANCE ANALYSIS

In this section, we consider the efficiency of our proposed DSSE scheme, where the efficiency is measured in terms of the computation and communication complexities.

COMPUTATION. The BuildIndex algorithm generates for all keywords $w \in \mathcal{W}$ an n-bit string (f_w). The resulting index is an $m \times n$-matrix, where m is the number of keywords and n the number of documents. The algorithm has to generate m times an n-bit string and calculate mn bitwise XOR. Thus, the index size, as well as the computation complexity is $O(mn)$.

The Trapdoor algorithm chooses two random values r_1, r_2 and a random n-bit string r_3, evaluates the permutation $\pi(\mathcal{W}, \sigma_{r_2})$ at keyword s to find position $q_s(r_2)$, generate two n-bit strings ($f_s, g_s(r_1)$) and finally computes the two bitwise XORs on the n-bit strings. The trapdoor size and the computation complexity is $O(n)$.

In the SearchIndex algorithm, *SP* generates $(m + 1)$ n-bit strings and computes two XORs per keyword for the re-encryption of the index. Then, *SP* generates and performs a random permutation on m index positions. Thus the computational complexity for *SP* is $O(mn)$. *QP* performs a simple table-lookup and calculates one XOR on a n-bit string, resulting in a complexity of $O(n)$.

COMMUNICATION. Our scheme requires the index to be transferred per query. Since our index uses one bit per keyword per document (cf. Table 4.1), the communication complexity is $O(mn)$.

The trapdoor T_1^s consists of two random values and a n-bit random string. The trapdoor T_2^s consists of an index position, i. e., a number between 1 and m, and the n-bit string k. The intermediate result I_s'' of the query proxy *QP* that has to be transferred to *SP* is of size n bit.

REMARK 4.1 Note, that the above asymptotic complexities are similar to previous schemes with the same security guarantee [3, 167]. In practice, however, various operations make a difference for the real performance numbers. In particular, our scheme is based entirely on XOR operations and pseudo-random functions, which are orders of magnitude more efficient than other operations such as pairings. As an example, the scheme by Shen et al. [167] needs to compute $n(2m + 2)$ composite order pairings per search query. For a document set of 5000 documents and 250 keywords, a search query requires 8.4 days [3]. In

Table 4.1: Example index sizes for different document and keyword sets.

	10,000	50,000	100,000	1,000,000
100	122 kB	610 kB	1.2 MB	12 MB
250	305 kB	1.5 MB	3 MB	30 MB
500	610 kB	3 MB	6 MB	60 MB
...		...		
100,000	119 MB	596 MB	1.2 GB	11.6 GB

Table 4.2: Estimated search times for a keyword search in different document/keyword sets assuming a 1 Gb/s network connection between *SP* and *QP*.

	10,000	50,000	100,000	1,000,000
100	1.6 ms	7.8 ms	16 ms	161 ms
250	3.9 ms	20 ms	39 ms	393 ms
500	7.8 ms	39 ms	79 ms	786 ms
...		...		
100,000	1.56 s	2.68 s	15.6 s	156 s

comparison, our scheme requires $n(2m+3)$ XOR operations and performs a search on the same dataset in less than 2 ms, assuming a 1 Gb/s network connection between *SP* and *QP*. See the example below and Table 4.2 for estimated performance numbers of different document/keyword sets.

EXAMPLE. For the following example, we use a data collection of 1 million documents and a keyword list of 500 (which we consider practical in many scenarios). Then, the encrypted index is of size $500 * 1M$ bit $= 500$ M bit or 60 MB. Using a 1 Gb/s network connection between *SP* and *QP* results in a theoretical max. transmission rate of 120 MB/s. The real max. is around 80 MB/s. To transmit an index of 60 MB takes 0.75 s at a rate of 80 MB/s. The computation on *SP* requires $2m+2$ XOR on n-bit strings. The query proxy *QP* performs one XOR on n-bit strings. An bitwise XOR on 500 million bits, takes less than 18 ms on an Intel i5 CPU M460@2.53 GHz. Per search, we require $n(2m+3)$ XORs. In our example, this results in 1,003,000,000 XOR, taking 36 ms. In total, the search takes 786 ms. Even for a huge keyword list of 100,000 keywords and one million documents, a query takes around 2.6 minutes.

4.6 COLLUDING SERVERS

Recall that our security analysis assumes that the storage provider and the query proxy do not collude. In this section, we discuss the implication of *SP* and *QP* colluding.

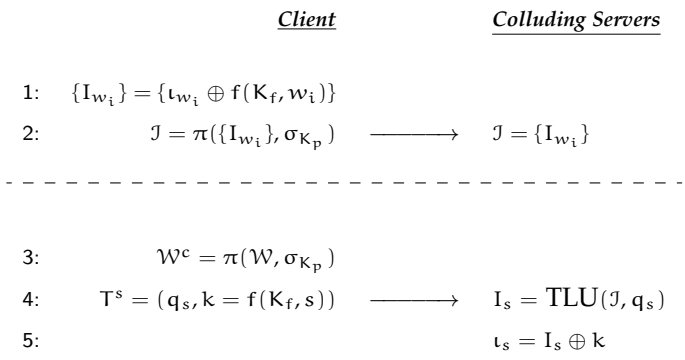


Figure 4.2: SSE scheme with colluding servers.

If *SP* and *QP* collude, they can invert the permutation and encryption performed on a per-query basis (lines 4 and 5 in Figure 4.1). In this case, we can omit the re-encryption and permutation without further sacrificing data confidentiality. Figure 4.2 shows the resulting scheme, which treats *SP* and *QP* as a single server.

The original distributed scheme is reduced to a centralized scheme consisting of a client and a server. In the reduced scheme, the client sends an encrypted and permuted index to the server, and queries the server directly by sending trapdoors. Hence, the reduced scheme is in fact a "standard" SSE scheme. It is easy to see that it leaks the search and access pattern. However, we show in the next section that this reduced scheme still satisfies Curtmola et al.'s [76] definition for adaptive security.

4.6.1 *The reduced scheme*

As mentioned before, the reduced scheme is a plain SSE scheme which does not fall under the DSSE definition given in Section 4.2. Therefore, we redefine this scheme in the standard SSE terminology as introduced by Curtmola et al. [76].

- $K \longleftarrow \mathsf{Gen}(1^\lambda)$: the client generates a set of three secret keys $K = (K_d, K_f, K_p)$, for the document encryption, the row encryption and the table permutation respectively. Remark that this construction does not share keys with the search provider.
- $(I, \mathbf{c}) \longleftarrow \mathsf{Enc}(K, \mathbf{D})$: the client encrypts every document in $\mathcal{D}$ with the key K_d using a PCPA secure symmetric encryption scheme, e. g.AES in CTR mode [127]. An encrypted and permuted index I is calculated as follows:
  1. For every keyword $w_i \in \mathcal{W}$:
     a) For all documents $d_j \in \mathbf{D}$; let $\mathcal{I}[i]_j = 1$ if w_i matches d_j, otherwise set $\mathcal{I}[i]_j = 0$.
     b) Reassign $\mathcal{I}[i] \longleftarrow \mathcal{I}[i] \oplus f(K_f, w_i)$, which encrypts the row $\mathcal{I}[i]$.
  2. Generate a permuted index I by applying σ to the encrypted rows, such that $I = \pi(I, \sigma_{K_p})$. Thus, for all $1 \leqslant i \leqslant m$: $I[\sigma_{K_p}(i)] = \mathcal{I}[i]$.

  Output the encrypted and permuted index I.

- $t \longleftarrow \mathsf{Trpdr}(K, w_i)$: using the key K_p, the client can calculate the trapdoor $t = (\sigma_{K_p}(i), f(K_f, w_i))$. The trapdoor contains the position of the row in the permuted index corresponding to w_i and the encryption/decryption key for the row.
- $X \longleftarrow \mathsf{Search}(I, t)$: given an index and a trapdoor, the algorithm does the following:
  1. Find and decrypt the row $r = f(K_f, w_i) \oplus I[K_p(i)]$.
  2. From the decrypted row, deduce the set of document identifiers $\{id(d_i) | d_i \in \mathbf{D} \wedge r[i] = 1\}$. Note that the server only has to know what document identifier corresponds to the i-th bit.

4.6.2 *Security Analysis*

In this section, we prove our reduced SSE scheme to be semantically secure against an adaptive adversary. We use the simulation-based definition for adaptive semantic security as provided by Curtmola et al. [76] (Definition 4.13).

Recall that a *history* $H = (\mathcal{D}, \mathbf{s})$ over q queries, is a tuple including the document collection and the queried keywords. We denote the keywords queried for by $\mathbf{s} = (s_1, \ldots, s_q)$ where s_i is the keyword asked for in the i-th query, and every $s_i \in \mathcal{W}$. Note there may exist a pair s_i, s_j where $i \neq j$ but $s_i = s_j$.

An *access pattern* $\alpha(H)$ from a history $H = (\mathcal{D}, \mathbf{s})$ contains the results of each query in H. Thus $\alpha(H) = (\mathcal{D}(s_1), \ldots, \mathcal{D}(s_q))$ is a vector containing the sets of document identifiers of the matched documents.

The *search pattern* $\sigma(H)$ induced from a q-query history is a $q \times q$ binary matrix such that $s_i = s_j \Leftrightarrow \sigma(H)[i][j] = 1$. If the setting is unambiguous, we write α (resp. σ) for $\alpha(H)$ (resp. $\sigma(H)$).

A *trace* $\tau(H) = (|d_1|, \ldots, |d_n|, \alpha(H), \sigma(H))$ contains the lengths of all documents in $\mathbf{D}$ and the access and search pattern induced by the input history H.

In the simulation-based definition of adaptive security by Curtmola et al. [76], the basic idea is to build a simulator which is given only the trace, and can simulate an index, ciphertexts and trapdoors that are indistinguishable from the real index, ciphertexts and trapdoors. We allow the adversary to build the history linked to the trace adaptively; the adversary can query for a keyword, receive a trapdoor and query again polynomially many times.

THEOREM 4.2 (Security) Our reduced SSE scheme from Section 4.6.1 is secure with respect to Curtmola et al.'s [76] definition for adaptive semantic security for SSE.

Proof. We will first define the q-query simulator $\mathcal{S} = (\mathcal{S}_0, \ldots, \mathcal{S}_q)$ that, given a trace $\tau(H)$, generates $v^* = (I^*, c^*, t^*)$ and a state $st_{\mathcal{A}}$. The simulator $\mathcal{S}_0$ only creates an index and document ciphertexts, as no keywords have been queried for at this stage. The i-th simulator $\mathcal{S}_i$ returns trapdoors up until the i-th query. We will then prove that no polynomial-size distinguisher D can distinguish between the distributions of v^* and the outputs of an adaptive adversary that runs the real algorithm.

- $\mathcal{S}_0(1^k, \tau(H))$: given $(|d_1|, \ldots, |d_n|)$, choose $I^* \xleftarrow{\$} \{0,1\}^{m \times n}$. Recall that m is public as it is the size of the dictionary $\mathcal{W}$, and that n is included in the trace as the number of $|d_i|$'s.

  The ciphertexts are simulated by creating random strings of the same lengths as the documents; $c_i^* \xleftarrow{\$} \{0,1\}^{|d_i|}$, where $|d_i|$ is included in the trace. Also, a random permutation $p^* : [1, m] \to [1, m]$ is generated.

  The simulator stores I^*, a counter $c = 0$ and a random permutation $\sigma^* : [1, m] \to [1, m]$ in the state $st_{\mathcal{S}}$, and outputs $v^* = (I^*, c^*, st_{\mathcal{S}})$.

- $\mathcal{S}_i(st_{\mathcal{S}}, \tau(H, s_1, \ldots, s_i))$ for any $1 \leqslant i \leqslant q$: given the state (which includes I^* and any previous trapdoors) and the access pattern α, the simulator can generate a trapdoor t_i^* as follows:

  Check if the keyword has been queried before; if there is a $j \neq i$ such that $\sigma[i][j] = 1$, set $t_i^* = t_j^*$. Otherwise:

  - Increase the counter by one and generate a unique row index $\sigma^*(c)$, using the counter and the random permutation. Note that the counter will never exceed m, as there are only m unique keywords.
  - Calculate a bit string $r \in \{0,1\}^n$ such that for $1 \leqslant j \leqslant n$: $r[j] = 1 \Leftrightarrow id(d_j) \in \alpha[i]$. We now have what should be the unencrypted row of the index corresponding to the keyword queried for.
  - Calculate $k^* = r \oplus I^*[\sigma^*(c)]$. We now have a dummy key which satisfies the property $k^* \oplus I^*[\sigma^*(c)] = r$.
  - Let $t_i^* = (\sigma^*(c), k^*)$

  Include t_i^* in $st_{\mathcal{S}}$, and output $(t_i^*, st_{\mathcal{S}})$.

We will now show that the outputs of $\mathbf{Real}_{SSE,\mathcal{A}}$ and $\mathbf{Sim}_{SSE,\mathcal{A},\mathcal{S}}$, being v and v^*, can not be distinguished by a distinguisher D that is given $st_{\mathcal{A}}$. Recall that $v = (I, \mathbf{c}, t_1, \ldots, t_q)$ and $v^* = (I^*, \mathbf{c}^*, t_1^*, \ldots, t_q^*)$.

- (Indistinguishability of I and I^*) The output of $f(K_f, w_i)$ is indistinguishable from random by definition of f. Therefore, the XOR of entries of the index and the output of f is indistinguishable from random bit strings of the same length [25]. Since I^* is a random bit string of the same length as I, and with all but negligible probability $st_{\mathcal{A}}$ does not contain the key, we conclude that I and I^* are indistinguishable.
- (Indistinguishability of c_i and c_i^*) Since c_i is PCPA-secure encrypted, c_i cannot be distinguished from a random string. Since every c_i^* is random and of the same length as c_i, and with all but negligible probability $st_{\mathcal{A}}$ does not contain the encryption key, c_i is distinguishable from c_i^*.
- (Indistinguishability of $t_i = (K_p(i), k = f(K_f, w_i))$ and $t_i^* = (\sigma^*(c), k^*)$) With all but negligible probability, $st_{\mathcal{A}}$ will not contain the key K_p, so the pseudo-randomness of π guarantees that each $\sigma^*(c)$ is computationally indistinguishable from $\pi(K_p, i)$.

  As stated above, I and I^* are indistinguishable, thus $I[i]$ and $I^*[i]$ are indistinguishable, thus $I[i] \oplus r = k$ and $I^*[i] \oplus r = k^*$ are indistinguishable. Thus, t_i and t_i^* are indistinguishable.

This indistinguishability shows that $\mathcal{S}$ successfully simulates the view of the adversary, which concludes the proof. □

4.6.3 *Performance Analysis*

COMPUTATION AND STORAGE. The Enc algorithm of this scheme is similar to the BuildIndex algorithm of our scheme in Section 4.3: it generates an inverted index of the dictionary $\mathcal{W}$ over the documents $\mathbf{D}$. Thus, the index size and the computation complexity are $O(m \cdot n)$. Table 4.1 shows example index sizes for various document and keyword sets.

The Trapdoor algorithm calculates the position of a row and its decryption key by evaluating σ_{K_p} and f. Since the decryption key is as long as a row, the trapdoor size is $O(n + \log(m))$. The computational complexity depends on the chosen pseudo-random function f.

Given a trapdoor, the server evaluates the SearchIndex algorithm by doing a table lookup and XOR'ing the resulting row with the given decryption key. The computational complexity is $O(n)$.

COMMUNICATION. The trapdoor contains a row id ($O(\log m)$ bits) and the row decryption key ($O(n)$ bits). Thus, the communication complexity is $O(n + \log(m))$.

REMARK 4.2 As with the DSSE scheme, the above functions are based entirely on XOR operations and pseudo-random functions.

COMPARISON. To demonstrate the efficiency of our scheme, we will compare it to Curtmola et al.'s [76] adaptively secure SSE scheme. Since both schemes use negligibly little computational resources (lookups and XOR's only), we focus on the sizes of trapdoors instead.

For details we refer to Curtmola et al.'s [76] scheme and only state here that their scheme stores document identifiers of matching documents, rather than a single bit encoding of whether a document matches a keyword. To hide the actual number of matches a document has, every document id is stored once for every possible keyword/document match. The number of possible matches equals the number of keywords a document can contain. This value, referred to as $\mathtt{max}$, is limited by two factors: the number of distinct keywords in $\mathcal{W}$ and the size of a document. The following algorithm can be used to determine $\mathtt{max}$:

- Let $i = 0$, $\mathtt{max} = 0$ and S be the document size in bytes.
- While $S > 0$:
  - If $2^{8 \cdot i} \cdot i \leqslant S$, set $i = i + 1$, $\mathtt{max} = \mathtt{max} + 2^{8 \cdot i}$ and $S = S - 2^{8 \cdot i} \cdot i$
  - Otherwise, set $\mathtt{max} = \mathtt{max} + \frac{S}{i}$ and $S = 0$.
- Let $\mathtt{max} = \min(\mathtt{max}, |\mathcal{W}|)$: if there are not enough keywords to fill the entire document, use the size of the dictionary as $\mathtt{max}$ value.

In [76], a trapdoor is $n \cdot \log(\mathtt{max} \cdot n)$ bit. Our scheme uses trapdoors of size $\log(m) + n$ bit; a $\log(m)$ bit row id and an n bit key to decrypt it. Notice that the number of keywords hardly affects the size of a trapdoor.

We compare document sets with documents of 25 kB (i. e., $\mathtt{max} \leqslant 12628$), to demonstrate the effect of the document size on the performance of the schemes.

The comparison in Table 4.3 indicates that our scheme outperforms Curtmola et al.'s scheme in terms of trapdoor sizes in the given setting. We believe that our scheme is of interest even outside the context of this chapter due to its conceptual simplicity and high efficiency.

Table 4.3: Comparison of trapdoor sizes.

Doc. size		Curtmola et al.			Our scheme
25 kB	$m = 100$	1000	15,000	100,000	100,000
$n = 1000$	2.1 kB	2.5 kB	2.9 kB	2.9 kB	141 B
10,000	24.9 kB	29.0 kB	33.6 kB	33.6 kB	1.25 kB
100,000	290 kB	332 kB	378 kB	378 kB	12.5 kB
10,000,000	37.3 MB	41.5 MB	46.1 MB	46.1 MB	1.25 MB

4.7 CONCLUSION

In this chapter, we have explored SSE in a distributed setting and proposed the concept of distributed searchable symmetric encryption (DSSE) for outsourcing encrypted data. Compared with standard SSE, a DSSE scheme can potentially provide more efficiency and better security guarantees. We described a security model that in addition to previous models protects the search pattern. We proposed a construction for DSSE (based entirely on binary XOR operations and pseudo-random functions) which is highly efficient, despite the additional security. The scheme uses an inverted index approach and borrows re-shuffling techniques from private information retrieval. The main idea is, that the query proxy gets a fresh (i. e., re-encrypted and shuffled) index per query. Thus, the query can be realized by a simple table look-up, without revealing the search pattern.

We have also shown that even if the storage provider and query proxy collude, the scheme is still secure under Curtmola et al.'s definition for adaptive semantic security for SSE. The resulting SSE scheme when the two servers collude is very efficient and outperforms Curtmola et al.'s scheme in terms of trapdoor sizes.

The scheme hides the search pattern, using approach A2. The proposed scheme relies on interaction between the query proxy and the storage provider. Whether we can construct efficient search pattern hiding schemes without any interaction is still an open question.

This page intentionally left blank.

5

SECURELY OUTSOURCED FORENSIC IMAGE RECOGNITION

In Chapters 3 and 4 we present two search pattern hiding constructions to query encrypted data, based on A1 and A2, respectively. In this chapter, answering RQ2, we propose a novel scheme, for a concrete application scenario, i. e., to securely outsource forensic image recognition. The scheme uses the primitives from previous chapters to construct a search pattern hiding scheme for unencrypted data.

Forensic image recognition tools are used by law enforcement agencies all over the world to automatically detect illegal images on confiscated equipment. This detection is commonly done with the help of a strictly confidential database consisting of hash values of known illegal images. To detect and mitigate the distribution of illegal images, for instance in network traffic of companies or Internet service providers, it is desirable to outsource the recognition of illegal images to these companies. However, law enforcement agencies want to keep their hash databases secret at all costs as an unwanted release may result in misuse which could ultimately render these databases useless.

In this chapter, we present SOFIR, a provably secure tool for the Secure Outsourcing of Forensic Image Recognition allowing companies and law enforcement agencies to jointly detect illegal network traffic at its source, thus facilitating immediate regulatory actions. SOFIR cryptographically hides the hash database from the involved companies. At fixed intervals, SOFIR sends out an encrypted report to the law enforcement agency that only contains the number of found illegal images in the given interval, while otherwise keeping the company's legal network traffic private. Our experimental results show the effectiveness and practicality of our approach in the real-world.

5.1 INTRODUCTION

Forensic Image Recognition (FIR) tools are being used by Law Enforcement Agencies (LEAs) worldwide in order to detect illegal images on confiscated equipment. The Dutch police, for example, owns a database consisting of hash values of so-called *PIPs* (Picture that Interests the Police), such as images showing glorification or overexposure of violence, indignity or pornographic content, like zoophilia and pedophilia. When the police confiscates equipment with data storage, the hash of each picture found in the storage is computed and looked up in the PIP database. If there are many matches, the police knows that the confiscated equipment contains PIPs with high probability and the investigation is continued manually to crosscheck.

Next to LEAs, companies like Internet service providers (ISPs) or hosting providers, and especially public funded institutions also have an interest in filtering PIPs from their own network traffic. In many countries, ISPs and hosting providers are already filtering out illegal content, either voluntarily [81] or are forced by law (e. g., the Communications Assistance for Law Enforcement Act, CALEA, in the USA). Several companies are even specialized in image filtering techniques for network traffic.

To facilitate the fight of the distribution of PIPs on network traffic, access to the existing police's PIP database would be beneficial. But a major concern of the police when outsourcing the filtering to third parties is the leakage of the PIP database. An even partially disclosed PIP database would allow perpetrators to misuse the database, e. g., by matching their data against the PIP database (before distribution) to check if their images are detectable by the system or not.

A problem with current filter technologies is that they instantly block access to known PIPs. This inherently reveals that the blocked image is in the database, eventually causing the disclosure of the database. Next to the commercial solutions, Peter et al. [154] propose a privacy-preserving architecture to outsource FIR. While preserving the privacy of the owner of the confiscated equipment, their approach unfortunately leaks the PIP database, so its security relies only on legally binding license agreements.

On the other hand, ISPs and especially companies, do not want to expose information on their own network traffic for privacy reasons. Thus the police should learn only the least amount of necessary information to take further legal actions, i. e., the number of actual PIPs detected.

In this chapter, we propose SOFIR, a patent-pending [2] Securely Outsourced Forensic Image Recognition tool that inspects network traffic to detect known PIPs. SOFIR allows third parties to scan their network traffic for PIPs, without ever having access to the PIP database. At the same time, the third party reveals only the number of PIPs detected in a certain interval in their network traffic.

The rest of this chapter is organized as follows: Section 5.2 introduces the building blocks used in our construction, which in turn is described in Section 5.3. Our implementation parameters and results are presented in Section 5.5, while Section 5.6 concludes with a summary.

5.2 PRELIMINARIES

We use the following notation and building blocks. Let $\mathcal{D} = \{d_1, \ldots, d_n\}$ be a database, consisting of n known PIPs.

BLOOM FILTER. A Bloom filter (BF) [35] is a data structure which is used to answer set membership queries. It is represented as an array of m bits which are initially set to 0. We write $B[i]$ to denote the i-th position of the BF. In general the filter uses k independent hash functions h_j $(1 \leqslant j \leqslant k)$, where $h_j : \{0,1\}^* \to [1, m]$ maps a set element to one of the m array positions. For each element e in a set $\mathcal{S} = \{e_1, \ldots, e_n\}$ the bits at positions $B[h_j(e)]$ are set to 1. To check whether an element x belongs to the set $\mathcal{S}$, we check if the bits at all positions $B[h_j(x)]$ are set to 1, i. e., if $\prod_{j=1}^{k} B[h_j(x)] \stackrel{?}{=} 1$. If so, x is considered a member of set $\mathcal{S}$. BFs do not produce false negatives, but inherently have a possibility of false positives (FPs), since the positions of an element may have been set by one or more other elements. With appropriate parameters m, n and k, the false positive probability P can be set to a desired low level [52]. The bit size of the BF can be approximated as:

$$m = \frac{1}{1 - \left(1 - P^{\frac{1}{k}}\right)^{\frac{1}{kn}}}.$$

SOMEWHAT HOMOMORPHIC ENCRYPTION. Somewhat homomorphic encryption (SHE) allows to perform a limited number of different algebraic operations on plaintexts but in the encrypted domain without knowing the decryption key. We use the private-key lattice-based SHE scheme by Brakerski and Vaikuntanathan (BV) [54], which allows for multiplications and additions. Any other probabilistic semantically secure SHE scheme that allows at least one multiplication followed by multiple additions on encrypted values can also be used (e. g., Gentry-Halevi-Vaikuntanathan [91] or Boneh-Goh-Nissim [49]). The homomorphic encryption of an element x is written as $[\![x]\!]$. For the BV scheme we write: $[\![x]\!] \otimes [\![x']\!] = [\![x \otimes x']\!]$, where $\otimes \in \{+, \cdot\}$.

The BV scheme works over polynomials and uses the following parameters: a polynomial degree α (which is a power of 2), a modulus q (which is a prime such that $q \equiv 1 \pmod{2\alpha}$), the cyclotomic polynomial $f(x) = x^\alpha + 1$, the discrete Gaussian error distribution χ with standard deviation σ, the ring $R_q = \mathbb{Z}_q[x]/\langle f(x)\rangle$, the number of supported additions A and multiplications M and a prime $t < q$, which defines the message space as $R_t = \mathbb{Z}_t[x]/\langle f(x)\rangle$.

A freshly generated ciphertext $ct = (c_0, c_1)$ consists of two elements in R_q (i. e., polynomials). We say that ct has *ciphertext degree* $C = 2$. Multiplying two ciphertexts increases the degree of the resulting ciphertext: $(c_0, \ldots, c_a) \cdot (c_0, \ldots, c_b) = (c_0, \ldots, c_{a+b})$. Since each polynomial coefficient is at most of size $q-1$, the ciphertext size $|c| = C \cdot \alpha \cdot \lceil \lg(q) \rceil$ is an upper bound and denoted by WC $|c|$. The security of the scheme is measured by the runtime T of the distinguishing attack [126]. Thus, $\lg(T)$ denotes the bit security of the scheme. An algorithmic definition of the BV scheme is described in Section 3.6.1.

5.3 THE PROPOSED SOFIR CONSTRUCTION

A Law Enforcement Agency (LEA) encrypts its PIP database and gives it to the ISP (or some other company or hosting provider). The ISP uses the encrypted database to find PIPs in its network traffic and regularly sends an encrypted report on the number of detected PIPs back to the LEA. The LEA can decrypt the report to check the results of the matching and, if necessary, starts an investigation.

SECURITY REQUIREMENTS. To securely outsource FIR, we require that the hash values of the database do not leak to anybody. Note that this also includes the protection of the matching result, since this inherently leaks information on the database. To protect the privacy of the ISP, the LEA should learn only the least amount of necessary information possible, i. e., the total number of PIPs found.

OUR CONSTRUCTION. We present SOFIR, which consists of three phases: the *initialization phase* (run at the LEA), the *recognition phase* (run at the ISP) and the *revelation phase* (run at the LEA).

During the *initialization phase*, the LEA first generates a secret key K for the BV scheme and initializes a BF. Moreover, an inner hash function h^{in} (to compute the hash value of an image) and several outer hash functions h_j^{out}, for $j \in [1, k]$ (to calculate the BF positions) are chosen.

To insert all PIPs $d \in \mathcal{D}$ into the BF, first an inner hash value $x = h^{in}(d)$ is computed. Then, for all x, the positions $p_j = h_j^{out}(x)$ for $j \in [1, k]$ are calculated, using the outer hash functions. The BF positions $B[p_j]$ are set to 1. After all PIPs

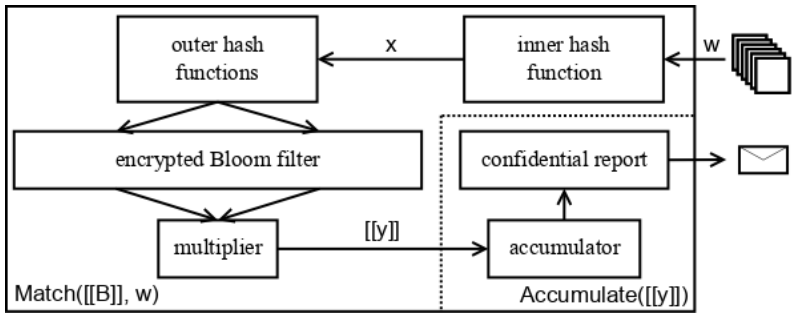


Figure 5.1: SOFIR architecture (simplified).

have been inserted into the BF, it is encrypted bit-by-bit using the BV scheme and the secret key K. The encrypted BF $[\![B]\!] = ([\![B[1]]\!], \ldots, [\![B[m]]\!])$ can then be used in the SOFIR recognition phase by the ISP as we explain momentarily.

The *recognition phase* (cf. Figure 5.1) is split into two algorithms: Match (which identifies PIPs in the encrypted domain) and Accumulate (which adds up all the matching results and sends a confidential report to the LEA). To check the network traffic for PIPs, each image file img is processed in the following way. First, Match uses the inner hash function to calculate $x = h^{in}(\mathsf{img})$. The outer hash function is then used to calculate the BF positions $p_j = h_j^{out}(x)$ for all $j \in [1, k]$. Note that h^{in} and h_j^{out} are the same hash functions as used by the LEA in the initialization phase. The encrypted BF positions $[\![B[p_j]]\!]$ are processed by the *multiplier*, which uses the multiplicative homomorphic property of the BV scheme to privately compute the matching result $[\![y]\!] = \prod_{j=1}^{k} [\![B[p_j]]\!]$. The value $[\![y]\!]$ will be $[\![1]\!]$ in case of a match and $[\![0]\!]$ otherwise. The Accumulate algorithm takes $[\![y]\!]$ and adds it (using the additive homomorphic property of BV) to the final accumulated result $[\![R]\!]$, which is the total number of PIPs matching the database. After a certain time (e. g., one hour or day) or threshold (e. g., 50,000 queries), $[\![R]\!]$ is sent to the LEA and the internal $[\![R]\!]$ is reset to $[\![0]\!]$.

During the *revalation phase*, given $[\![R]\!]$ and the secret key K, the LEA decrypts $[\![R]\!]$ and outputs the number of possible matches. If $R > \tau$, where τ is a certain threshold, an alarm is raised for further investigation.

5.4 SECURITY DISCUSSION

The security of our construction can be analyzed as follows. In our scenario, the Bloom filter acts as a (long term) query on the unencrypted data stream. Due to the use of a probabilistic semantically secure SHE scheme, it is impossible to distinguish between $[\![0]\!]$ and $[\![1]\!]$. Therefore, the BF itself does not leak any information on the contents and thus serves as a probabilistic trapdoor. Since all operations for the matching (homomorphic multiplication of the encrypted BF values), as well as the accumulation (homomorphic addition of encrypted values) are performed in the encrypted domain, the ISP cannot gain any information on the encrypted database, the computations or the results. Thus, the search pattern remains hidden. The LEA receives only the accumulated result, which is the number of found PIPs. Thus, the ISP does not reveal information on its network traffic, except the number of detected image files.

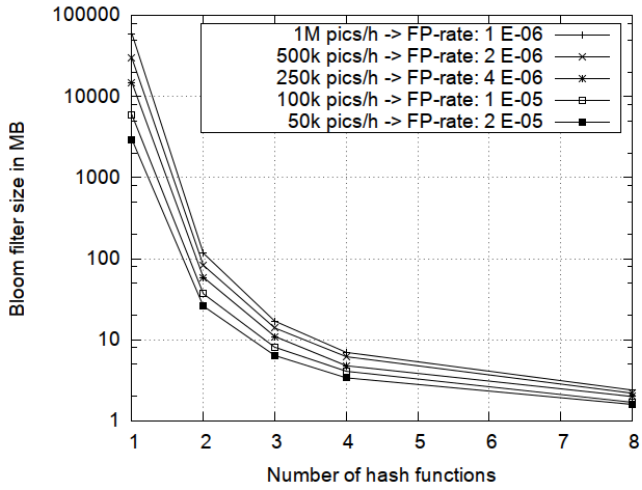


Figure 5.2: Bloom filter sizes in MB using $n = 500{,}000$.

5.5 PERFORMANCE ANALYSIS

This section gives implementation details and a feasibility study, where we explore the parameter space to get realistic numbers for implementing SOFIR.

To setup our system we have to set (i) the system parameters, (ii) BF parameters and finally (iii) BV parameters. We will show experimental results based on a real-world setting.

We start by estimating N, the number of image files that need to be scanned per hour. To get a realistic value, we monitored the network traffic of our University homepage for two weeks and (on average) registered access to around 50,000 image files (i. e., .jpg, .jpeg, .png, and .gif) per hour. This is our starting point to determine our parameters.

(I) SYSTEM PARAMETERS. Having 50,000 images per hour allows for 71.4 ms (60 min/50,000) of maximal processing time per image and a false positive rate of 1/50,000 = 2 E-05 (one false positive per hour)[1]. Since there is no publicly available information on PIP database sizes, we assume a PIP database consisting of $n = 500{,}000$ PIPs. The size n has an effect only on the BF size m and not on our timing results.

(II) BF PARAMETERS. For $n = 500{,}000$ we calculate m (cf. Section 5.2), the bit-size of the BF for different k and FP-rates ($P \leqslant$ 2.0 E-05) as shown in Figure 5.2.

Increasing the number of hash functions, significantly decreases the Bloom filter size. We realize the BF lookup by multiplying the corresponding BF values per image ($[\![y]\!] = \prod_{j=1}^{k} [\![B[p_j]]\!]$). The number of hash functions also determines M, the number of multiplications the BV scheme needs to support ($M = k - 1$). Therefore, we will look at the influence of M on the efficiency of the BV scheme. Recall that we only have 71.4 ms per image.

1 Note that the LEA will post-filter the results to remove false positives in case of an investigation to avoid accusing innocent.

Table 5.1: Details of the used BV parameters (cf. Section 5.2).

	M	α	$\lceil \lg(q) \rceil$	$\lg(T)$	WC $\lvert c \rvert$
(a)	3	4096	140	107	140 kB
(b)	2	4096	100	196	100 kB
(c)	1	2048	77	91	38.5 kB

(III) BV PARAMETERS. We choose our parameters for the symmetric BV scheme based on the number of images scanned per interval. The accumulator has to perform $A = 50{,}000$ additions. Recall, that the accumulator is adding either an encrypted 0 or 1, implying that 50,000 is the biggest value our encryption scheme needs to be able to handle. Thus, we set the size of the message space $t = 50{,}021$ (next prime $> 50{,}000$).

We also take into account the work of Lauter et al. [121] ($\sigma = 8$) which assessed the security against the decoding attack [126] and the distinguishing attack [132]. With these fixed parameters, we calculate the flexible parameters for different M as seen in Table 5.1.

Organizing the multiplications in form of a binary tree (cf. Figure 5.3) allows us to perform the multiplications in *layers*. Supporting M multiplications (one per layer) in the BV scheme, allows us to use up to $k = 2^M$ hash functions in our BF. We verified the results by experiments.

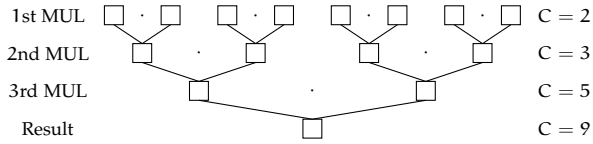


Figure 5.3: Multiplication tree for $k = 8$.

We implemented the symmetric BV scheme in C/C++ using FLINT, the Fast Library for Number Theory [99]. We tested the code on an Intel Xeon CPU X5677 with 3.47 GHz running linux 3.11.0-sabayon x86_64. Our timing results are shown in Table 5.2. By using the multiplication tree (cf. Figure 5.3), we always multiply ciphertexts with the same degree C. To compute the total processing time, we have to add up the times for all used operations (per layer). For instance, for (a), we have to compute 4, 2 and 1 multiplication in layers 1, 2 and 3, respectively for the matching, plus the final addition for the accumulator. Thus, the total precessing time per image is 1021 ms ($4 \cdot 63.6 + 2 \cdot 154 + 453 + 5.58$). Looking at Table 5.2 we see, that (c) is the only setting, that achieves our required processing time of max. 71.4 ms.

OPTIMIZATIONS AND FINAL RESULTS. At this moment, we have only a single threaded implementation of the BV scheme. The BV scheme itself is highly parallelizable and offers several optimization options as mentioned by Lauter et al. [121]. This reduces the times for the homomorphic multiplications and additions from Table 5.2.

Another possible optimization for SOFIR is to parallelize the image processing and use a single CPU core per image. Modern CPUs consist of 2–48 cores

Table 5.2: Implementation results for the BV scheme. Times for a single operation (MUL, ADD) dependent on the ciphertext degree.

BV	Operation	C = 2	C = 3	C = 5	C = 9	Total
(a)	ADD	0.19 ms	1.56 ms	2.94 ms	5.58 ms	1021 ms
	MUL	63.6 ms	154 ms	453 ms	—	
(b)	ADD	0.82 ms	1.39 ms	2.6 ms	—	211 ms
	MUL	48.9 ms	111 ms	—	—	
(c)	ADD	0.41 ms	0.65 ms	—	—	20.85 ms
	MUL	20.2 ms	—	—	—	

Table 5.3: Final implementation results. Encrypted BF sizes depending on k and BV. Single-core and optimized parallel timings.

M	k	BV	sec.	FP-rate	$\|[\![B]\!]\|$	time	time (parallel)
3	8	(a)	107 b	1.5 E-05	1.8 TB	1021 ms	16 cores
				1.2 E-06	2.6 TB		63.8 ms
2	4	(b)	196 b	1.7 E-05	2.8 TB	211 ms	4 cores
				1.6 E-06	5.1 TB		52.75 ms
1	2	(c)	91 b	1.5 E-05	9 TB	20.2 ms	1 core
				1.6 E-06	28 TB		20.2 ms

(e. g., AMD Opteron, Intel Xeon). For instance, using a usual 16 core CPU outputs 16 results in 1021 ms for (a), reducing the average processing time to 63.8 ms per image[2]. In this way we achieve our goal of having a processing time per image of less than 71.4 ms. Using the BV parameters (b), a Quad-Core CPU brings the average processing time down to 52.75 ms as shown in Table 5.3. The final BF size $|[\![B]\!]|$ is computed as $m \cdot$ WC $|c|$, since each of the m encrypted BF positions is of size WC $|c|$. Recall that $|[\![B]\!]|$ is an upper bound as explained in Section 5.2.

Table 5.3 shows, that depending on the available cores, we have several options to implement the system at our University. Having a CPU with 16 cores, allows us to use the BV parameters (a), resulting in 1.8 TB of storage and a maximum of 56,426 images per hour. A graphical representation of the trade-offs on security, time and storage using different BV parameters for our single-core results from Table 5.3 is shown in Figure 5.4.

Note that the timing results do not take the hash functions into account. However, compared to the homomorphic operations the times for hashing are negligible.

LIMITATIONS. Like all FIR tools, SOFIR is not able to detect encrypted PIPs. Encryption makes it impossible to access and process the plaintext im-

2 The parallel timings do not take the overhead for creating the threads for parallelization into account. Since the systems is designed to run continuously the times for the initial threading are negligible.

age without the decryption key. In practice however, most network traffic in companies or at hosting providers is unencrypted. SOFIR is designed to detect illegal images in such (unencrypted) settings. It gives companies more insight into their own network traffic by utilizing the confidential PIP databases held by law enforcement agencies. This is beneficial for both companies and LEAs to detect and mitigate the distribution of PIPs.

Another limitation of our current system is the inability to detect manipulated PIPs. To detect small image manipulations (e. g., cropping, rotating, scaling, shifting, JPEG compression, median filtering), a perceptual/robust hash function should be used in place of the inner hash function h^{in}. Such a perceptual hash function is a compression function that outputs very similar values (in terms of some metric, e. g., the Hamming metric) for perceptually similar pictures. Numerous instantiations using different techniques are known [178, 190]. In its current form, SOFIR is not able to deal with the fuzziness or error-proneness of perceptual hash functions. We consider this as interesting future work.

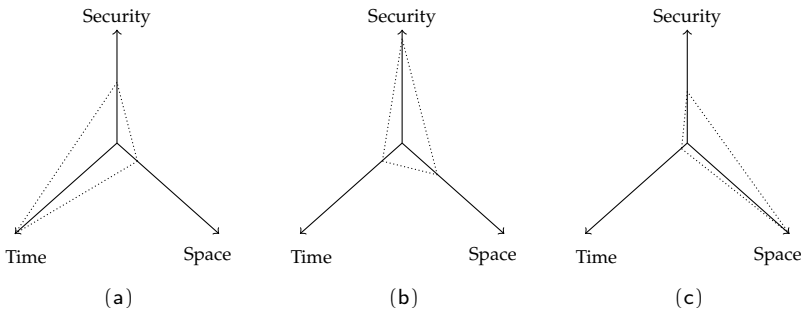


Figure 5.4: Graphical representation of the single-core implementation results from Table 5.3 for different BV parameters.

5.6 CONCLUSION

We have presented SOFIR, a system to detect known illegal images in network traffic in a privacy-preserving manner. Our mechanism is not limited to images but can also detect all other file formats, e. g., documents or videos. SOFIR cryptographically hides the hash database from the involved companies. The encrypted reports to the LEA only contain the number of found illegal images in a given interval, thereby keeping the company's legal network traffic private. We instantiated our proposed system using the somewhat homomorphic encryption scheme from Brakerski and Vaikuntanathan [54] and showed, that it is efficient to be used in real world application scenarios.

As future work we plan to replace the inner hash function with (i) a perceptual hash function to detect small image manipulations and (ii) different feature extraction algorithms, e. g., digital camera identifiers or watermarks, that can identify a camera model or images, respectively, in a unique way.

6 CONCLUDING REMARKS

In this thesis, we have shown, that search pattern hiding is feasible, even with reasonable efficiency. We discussed three ways for hiding the search pattern, two for encrypted and one for unencrypted data, and presented the first provably secure *efficient* search pattern hiding schemes.

Approaches A1 and A2 are based upon hiding the processed database entries from the server and are presented as SDR in Chapter 3 and DSSE in Chapter 4, respectively. For unencrypted data, we answer RQ2 and propose SOFIR in Chapter 5. To evaluate the performance of our schemes, we have implemented the building blocks of all our proposed search algorithms and SSW in C/C++. Our experimental results show the practical efficiency of our schemes.

Table 6.1 shows the running times of our search algorithms for different data sets compared to the search pattern hiding predicate encryption scheme by Shen, Shi, and Waters [167]. The numbers for SSW are based on the performance of a type A symmetric prime order pairing using the PBC [130] (Pairing-Based Cryptography) library and the fact that a pairing on a 1024-bit composite order elliptic curve can be 50 times slower than in a prime order group [84]. This is a conservative estimate since the SSW scheme uses composite order groups, where the order is the product of four primes. For the SDR and SOFIR scheme we used the FLINT library, namely Fast Library for Number Theory [99] to implement the Brakerski-Vaikuntanathan (BV) scheme [54]. The code for the BV scheme was co-developed by Arjan Jeckmans. This is the first publicly-available implementation of the scheme in C with carefully chosen parameters, so that it may be of independent interest for other works[1]. Our DSSE scheme is entirely based on XOR operations and pseudo-random functions. Note that these numbers are based on implementations on commodity hardware. Using specialized hardware will decrease the running times even further. Our solutions are orders of magnitude more efficient than SSW and show the practical applicability of our proposed solutions.

The implementation results show that a search query, using SDR, takes 47 seconds in an encrypted database with 1000 documents and 100 keywords, while a search query takes around 10 minutes in an encrypted database with 5000 documents and 250 keywords. In contrast, for the SSW scheme, a search query takes around 16 hours in an encrypted database with 1000 documents and 100 keywords on the same server, which is far away from efficient. We note that although the performance of the proposed SDR scheme does not say that it is an efficient solution in all application scenarios, it is reasonably efficient for small data sets like for example private emails or documents. Using the proposed DSSE scheme requires only 2.2 seconds to search through an encrypted dataset of one million documents and 1000 keywords. This shows, that search on encrypted data can be done efficiently, even for larger datasets. Our SOFIR construction can search for 50,000 query items through half a million plaintext items in 17.4 minutes. Thus, a search for one item requires 20.2

1 http://scs.ewi.utwente.nl/other/boesch/bv.zip

Scheme	Times for (keyword/document) sets (100/1000)	(250/5000)	(1000/1,000,000)
SSW [167]	16.3 h	8.4 d	18.4 y
SDR (1024) – Ch. 3	4.8 m	59.8 m	33.2 d
SDR (512) – Ch. 3	47 s	9.9 m	5.5 d
DSSE – Ch. 4	7 µs	0.09 ms	2.2 s
SSE – Ch. 4	36 ns	180 ns	36 µs
SOFIR – Ch. 5		(50,000/500,000) 17.4 m	

Table 6.1: Comparison of the search times of our proposed schemes with Shen et al.'s scheme.

ms. The running times of our algorithms show the practical relevance of our theoretical approaches.

6.1 CONTRIBUTIONS AND FUTURE WORK

Table 6.2 shows the results of the thesis per chapter. In particular:

CHAPTER 2. We survey the notion of provably secure searchable encryption by giving a complete and comprehensive overview of the two main SE techniques: Searchable Symmetric Encryption and Public Key Encryption with Keyword Search. Three major conclusions can be drawn from our work regarding efficiency, query expressiveness, and security. While the so-called IND-CKA2 security notion becomes prevalent in the literature and efficient (sublinear) SE schemes meeting this notion exist in the symmetric setting, achieving this strong form of security efficiently in the asymmetric setting remains an open problem. We observe that in multi-recipient SE schemes, regardless of their efficiency drawbacks, there is a noticeable lack of query expressiveness which hinders deployment in practice. Almost all searchable encryption schemes have a common problem. They leak the search pattern which reveals whether two searches were performed for the same keyword or not. Hence, the search pattern gives information on the occurrence frequency of each query, which can be exploited by statistical analysis, eventually allowing an attacker to gain full knowledge about the underlying plaintexts.

As a result, more research is required in all three research directions. In order to make an important step towards a widespread use of searchable encryption, schemes need to improve the *practical* efficiency as well as scalability for large datasets. To move towards closing the gap to plaintext searches, SE schemes have to improve the query expressiveness to include, for example, functionalities like phrase search, proximity search or regular expressions. The IND-CKA2 security definition is considered strong in the context of searchable encryption, but allows the leakage of the search pattern which can be fatal in certain applications. Thus, more search pattern hiding schemes for different scenarios are of importance.

Interaction	SP and AP	Leakage only AP	nothing
None	most [Ch. 2]	SSW [167]	Franz et al.* [82]
Client	several [Ch. 2]	SDR [Ch. 3]	ORAM SOFIR [Ch. 5] SDR** [Ch. 3]
Server	several [Ch. 2]	DSSE [Ch. 4]	Karvelas et al. [111]

Table 6.2: Our contributions in the field. * This scheme is non-interactive with all the clients, except for the data-owner. ** Using the extended PIR version of the scheme.

CHAPTER 3. We propose the concept of selective document retrieval (SDR) as a cryptographic primitive for outsourcing encrypted data. Compared with symmetric searchable encryption (SSE), an SDR scheme can potentially provide more flexible services and better security guarantees, including the protection of the search pattern (using approach A1). We describe a security model to cover three types of privacy properties, including index privacy, trapdoor privacy, and query result privacy. SDR offers a very flexible framework, and can be adapted very easily to support many useful search features. SDR's interactive nature makes the search process more practical, comparable to a web search. Our implementation results show the practical applicability of our scheme for small datasets. Due to the practical efficiency and increased security of our scheme, SDR has a strong impact on the field of secure data outsourcing. We propose the first efficient search pattern hiding construction (see Table 6.2). For our experiments we set appropriate parameters for the symmetric BV encryption scheme [54] and implement it in C. This is the first implementation of the scheme in C, so that it may be of independent interest. The results show, that SDR is orders of magnitude more efficient than the search pattern hiding predicate encryption scheme by Shen, Shi, and Waters [167].

SDR relies on client interaction which is regarded as a drawback in some application scenarios. How to construct efficient search pattern hiding schemes without client interaction is an interesting open question.

CHAPTER 4. We explore SSE in a distributed setting and propose the concept of distributed searchable symmetric encryption (DSSE) for outsourcing encrypted data. Compared with standard SSE, a DSSE scheme can potentially provide more efficiency and better security guarantees. We describe a security model that in addition to previous models also protects the search pattern (using approach A2). We propose a construction for DSSE (based entirely on binary XOR operations and pseudo-random functions) which is highly efficient, despite the additional security. Our DSSE scheme can perform a query by a simple table look-up, without revealing the search pattern. Our implementation results show the practical applicability of our DSSE scheme even for large datasets. DSSE can be used for practical applications where client interaction

is not appropriate. Due to its simplicity, efficiency and security improvements, DSSE has a strong impact on the field of provably secure searchable encryption. We propose the first efficient search pattern hiding construction without client interaction (see Table 6.2). If the storage provider and query proxy collude, the scheme is still IND-CKA2 secure. The resulting colluding SSE scheme is even more efficient than DSSE and outperforms Curtmola et al.'s [75] scheme in terms of trapdoor sizes and simplicity, which shows the practical importance of the colluding scheme. Both schemes bring us closer to a practical deployment of searchable encryption.

Constructing an efficient search pattern hiding scheme without interaction remains an open problem.

CHAPTER 5. We propose SOFIR, to answer RQ2. SOFIR illustrates how the techniques from our previous schemes can be used to construct a novel search scheme for a concrete real world application. The experimental results, which demonstrate SOFIR's efficiency, and our patent application [2] show the practical applicability and importance of our construction. Our application scenario has a strong impact on society and outsourced private search. SOFIR can effectively and efficiently detect the distribution of illegal content via the world wide web. SOFIR can also be used to privately search in all kinds of unencrypted data, e. g., RSS feeds.

In its current form, our SOFIR scheme is not able to deal with the fuzziness or error-proneness of perceptual hash functions. We consider this lack in query expressiveness as interesting future work.

SUMMARY. We present the first efficient search pattern hiding schemes so far. Our proposed solutions are more secure than previous constructions, due to the protection of the search pattern. This fills the gap between (search pattern leaking) searchable encryption and (nothing leaking) ORAM (see Table 6.2). At the same time, our constructions are orders of magnitude more efficient than SSW, which is the only search pattern hiding scheme in the context of searchable encryption. Our implementation results show the practical applicability of all our solutions.

We have learned, that building efficient and provably secure search pattern hiding schemes is possible. Further research is needed to increase the expressiveness of schemes for a widespread deployment of searchable encryption.

BIBLIOGRAPHY

AUTHOR REFERENCES

[1] Christoph Bösch, Richard Brinkman, Pieter Hartel, and Willem Jonker. Conjunctive Wildcard Search over Encrypted Data. In *Secure Data Management - 8th VLDB Workshop, SDM 2011, Seattle, WA, USA, September 2, 2011, Proceedings*, volume 6933 of *Lecture Notes in Computer Science*, pages 114–127. Springer, 2011. ISBN 978-3-642-23555-9.

[2] Christoph Bösch, Richard Brinkman, Pieter Hartel, and Willem Jonker. Method and System of Monitoring a Data Stream. Patent NL2009845 (Application), 2012.

[3] Christoph Bösch, Qiang Tang, Pieter Hartel, and Willem Jonker. Selective Document Retrieval from Encrypted Database. In *Information Security Conference - 15th Information Security Conference, ISC 2012, Passau, Germany, September 19-21, Proceedings*, volume 7483 of *Lecture Notes in Computer Science*, pages 224–241. Springer, 2012.

[4] Christoph Bösch, Pieter Hartel, Willem Jonker, and Andreas Peter. A Survey of Provably Secure Searchable Encryption. *ACM Computing Surveys*, 47(2):18:1–18:51, Aug 2014. ISSN 0360-0300.

[5] Christoph Bösch, Andreas Peter, Pieter Hartel, and Willem Jonker. SOFIR: Securely Outsourced Forensic Image Recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2014, Florence, Italy, May 04-09, 2014*, pages 2694–2698. IEEE, 2014.

[6] Christoph Bösch, Andreas Peter, Bram Leenders, Hoon Wei Lim, Qiang Tang, Huaxiong Wang, Pieter Hartel, and Willem Jonker. Distributed Searchable Symmetric Encryption. In *Twelfth Annual International Conference on Privacy, Security and Trust, PST 2014, Toronto, ON, Canada, July 23-24, 2014*, pages 330–337. IEEE, 2014.

[7] Jonathan Petit, Christoph Bösch, Michael Feiri, and Frank Kargl. On the Potential of PUF for Pseudonym Generation in Vehicular Networks. In *2012 IEEE Vehicular Networking Conference, VNC 2012, Seoul, Korea (South), November 14-16, 2012*, pages 94–100. IEEE, 2012. ISBN 978-1-4673-4995-6.

OTHER REFERENCES

[8] Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. Searchable Encryption Revisited: Consistency Properties, Relation to Anonymous IBE, and Extensions. *J. Cryptology*, 21(3): 350–391, 2008.

[9] Masayuki Abe, Rosario Gennaro, Kaoru Kurosawa, and Victor Shoup. Tag-KEM/DEM: A New Framework for Hybrid Encryption and A New

Analysis of Kurosawa-Desmedt KEM. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, volume 3494 of *Lecture Notes in Computer Science*, pages 128–146. Springer, 2005. ISBN 3-540-25910-4.

[10] Michael Adjedj, Julien Bringer, Hervé Chabanne, and Bruno Kindarji. Biometric Identification over Encrypted Data Made Feasible. In Atul Prakash and Indranil Gupta, editors, *Information Systems Security, 5th International Conference, ICISS 2009, Kolkata, India, December 14-18, 2009, Proceedings*, volume 5905 of *Lecture Notes in Computer Science*, pages 86–100. Springer, 2009. ISBN 978-3-642-10771-9.

[11] Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, and Yirong Xu. Order-Preserving Encryption for Numeric Data. In Gerhard Weikum, Arnd Christian König, and Stefan Deßloch, editors, *Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris, France, June 13-18, 2004*, pages 563–574. ACM, 2004. ISBN 1-58113-859-8.

[12] Georgios Amanatidis, Alexandra Boldyreva, and Adam O'Neill. Provably-Secure Schemes for Basic Query Support in Outsourced Databases. In Steve Barker and Gail-Joon Ahn, editors, *Data and Applications Security XXI, 21st Annual IFIP WG 11.3 Working Conference on Data and Applications Security, Redondo Beach, CA, USA, July 8-11, 2007, Proceedings*, volume 4602 of *Lecture Notes in Computer Science*, pages 14–30. Springer, 2007. ISBN 978-3-540-73533-5.

[13] Todd W. Arnold and Leendert van Doorn. The IBM PCIXCC: A New Cryptographic Coprocessor for the IBM eServer. *IBM Journal of Research and Development*, 48(3-4):475–488, 2004.

[14] Todd W. Arnold, Carl U. Buscaglia, F. Chan, Vincenzo Condorelli, John C. Dayka, W. Santiago-Fernandez, Nihad Hadzic, Michael D. Hocker, M. Jordan, T. E. Morris, and Klaus Werner. IBM 4765 Cryptographic Coprocessor. *IBM Journal of Research and Development*, 56(1):10, 2012.

[15] Dmitri Asonov. *Querying Databases Privately: A New Approach to Private Information Retrieval*, volume 3128 of *Lecture Notes in Computer Science*. Springer, 2004. ISBN 3-540-22441-6.

[16] Joonsang Baek, Reihaneh Safavi-Naini, and Willy Susilo. On the Integration of Public Key Data Encryption and Public Key Encryption with Keyword Search. In Sokratis K. Katsikas, Javier Lopez, Michael Backes, Stefanos Gritzalis, and Bart Preneel, editors, *Information Security, 9th International Conference, ISC 2006, Samos Island, Greece, August 30 - September 2, 2006, Proceedings*, volume 4176 of *Lecture Notes in Computer Science*, pages 217–232. Springer, 2006. ISBN 3-540-38341-7.

[17] Joonsang Baek, Reihaneh Safavi-Naini, and Willy Susilo. Public Key Encryption with Keyword Search Revisited. In Osvaldo Gervasi, Beniamino Murgante, Antonio Laganà, David Taniar, Youngsong Mun, and Marina L. Gavrilova, editors, *Computational Science and Its Applications - ICCSA 2008, International Conference, Perugia, Italy, June 30 - July 3,*

2008, Proceedings, Part I, volume 5072 of *Lecture Notes in Computer Science*, pages 1249–1259. Springer, 2008. ISBN 978-3-540-69838-8.

[18] Lucas Ballard, Matthew Green, Breno de Medeiros, and Fabian Monrose. Correlation-Resistant Storage via Keyword-Searchable Encryption. *IACR Cryptology ePrint Archive*, 2005:417, 2005.

[19] Lucas Ballard, Seny Kamara, and Fabian Monrose. Achieving Efficient Conjunctive Keyword Searches over Encrypted Data. In Sihan Qing, Wenbo Mao, Javier Lopez, and Guilin Wang, editors, *Information and Communications Security, 7th International Conference, ICICS 2005, Beijing, China, December 10-13, 2005, Proceedings*, volume 3783 of *Lecture Notes in Computer Science*, pages 414–426. Springer, 2005. ISBN 3-540-30934-9.

[20] Feng Bao, Robert H. Deng, Xuhua Ding, and Yanjiang Yang. Private Query on Encrypted Data in Multi-user Settings. In Liqun Chen, Yi Mu, and Willy Susilo, editors, *Information Security Practice and Experience, 4th International Conference, ISPEC 2008, Sydney, Australia, April 21-23, 2008, Proceedings*, volume 4991 of *Lecture Notes in Computer Science*, pages 71–85. Springer, 2008. ISBN 978-3-540-79103-4.

[21] Niko Bari and Birgit Pfitzmann. Collision-Free Accumulators and Fail-Stop Signature Schemes Without Trees. In Walter Fumy, editor, *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding*, volume 1233 of *Lecture Notes in Computer Science*, pages 480–494. Springer, 1997. ISBN 3-540-62975-0.

[22] Olivier Baudron, David Pointcheval, and Jacques Stern. Extended Notions of Security for Multicast Public Key Cryptosystems. In Ugo Montanari, José D. P. Rolim, and Emo Welzl, editors, *Automata, Languages and Programming, 27th International Colloquium, ICALP 2000, Geneva, Switzerland, July 9-15, 2000, Proceedings*, volume 1853 of *Lecture Notes in Computer Science*, pages 499–511. Springer, 2000. ISBN 3-540-67715-1.

[23] Mihir Bellare and Phillip Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security, November 3-5, 1993, Fairfax, Virginia, USA. ACM, 1993*, pages 62–73, 1993.

[24] Mihir Bellare and Phillip Rogaway. Optimal Asymmetric Encryption. In Alfredo De Santis, editor, *Advances in Cryptology - EUROCRYPT '94, Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, May 9-12, 1994, Proceedings*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111. Springer, 1994. ISBN 3-540-60176-7.

[25] Mihir Bellare, Anand Desai, Eron Jokipii, and Phillip Rogaway. A Concrete Security Treatment of Symmetric Encryption. In *Foundations of Computer Science*, pages 394–403. IEEE, 1997.

[26] Mihir Bellare, Alexandra Boldyreva, and Silvio Micali. Public-Key Encryption in a Multi-user Setting: Security Proofs and Improvements. In Bart Preneel, editor, *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques,*

Bruges, Belgium, May 14-18, 2000, Proceeding, volume 1807 of *Lecture Notes in Computer Science*, pages 259–274. Springer, 2000. ISBN 3-540-67517-5.

[27] Mihir Bellare, Alexandra Boldyreva, and Jessica Staddon. Randomness Re-use in Multi-recipient Encryption Schemeas. In Yvo Desmedt, editor, *Public Key Cryptography - PKC 2003, 6th International Workshop on Theory and Practice in Public Key Cryptography, Miami, FL, USA, January 6-8, 2003, Proceedings*, volume 2567 of *Lecture Notes in Computer Science*, pages 85–99. Springer, 2003. ISBN 3-540-00324-X.

[28] Mihir Bellare, Alexandra Boldyreva, K. Kurosawa, and Jessica Staddon. Multirecipient Encryption Schemes: How to Save on Bandwidth and Computation Without Sacrificing Security. *IEEE Transactions on Information Theory*, 53(11):3927–3943, 2007.

[29] Mihir Bellare, Alexandra Boldyreva, and Adam O'Neill. Deterministic and Efficiently Searchable Encryption. In Alfred Menezes, editor, *Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007, Proceedings*, volume 4622 of *Lecture Notes in Computer Science*, pages 535–552. Springer, 2007. ISBN 978-3-540-74142-8.

[30] Steven M. Bellovin and William R. Cheswick. Privacy-Enhanced Searches Using Encrypted Bloom Filters. *IACR Cryptology ePrint Archive*, 2004:22, 2004.

[31] Josh Cohen Benaloh and Michael de Mare. One-Way Accumulators: A Decentralized Alternative to Digital Sinatures (Extended Abstract). In Tor Helleseth, editor, *Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings*, volume 765 of *Lecture Notes in Computer Science*, pages 274–285. Springer, 1993. ISBN 3-540-57600-2.

[32] John Bethencourt, Dawn Xiaodong Song, and Brent Waters. New Constructions and Practical Applications for Private Stream Searching (Extended Abstract). In *2006 IEEE Symposium on Security and Privacy (S&P 2006), 21-24 May 2006, Berkeley, California, USA*, pages 132–139. IEEE Computer Society, 2006. ISBN 0-7695-2574-1.

[33] John Bethencourt, Dawn Xiaodong Song, and Brent Waters. New Techniques for Private Stream Searching. *ACM Transactions on Information and System Security (TISSEC)*, 12(3), 2009.

[34] Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible Protocols and Atomic Proxy Cryptography. In Kaisa Nyberg, editor, *Advances in Cryptology - EUROCRYPT '98, International Conference on the Theory and Application of Cryptographic Techniques, Espoo, Finland, May 31 - June 4, 1998, Proceeding*, volume 1403 of *Lecture Notes in Computer Science*, pages 127–144. Springer, 1998. ISBN 3-540-64518-7.

[35] Burton H. Bloom. Space/Time Trade-Offs in Hash Coding with Allowable Errors. *Commun. ACM*, 13(7):422–426, 1970.

[36] Carlo Blundo, Vincenzo Iovino, and Giuseppe Persiano. Private-Key Hidden Vector Encryption with Key Confidentiality. In Juan A. Garay, Atsuko Miyaji, and Akira Otsuka, editors, *Cryptology and Network Security, 8th International Conference, CANS 2009, Kanazawa, Japan, December 12-14, 2009. Proceedings*, volume 5888 of *Lecture Notes in Computer Science*, pages 259–277. Springer, 2009. ISBN 978-3-642-10432-9.

[37] Alexandra Boldyreva, Nathan Chenette, Younho Lee, and Adam O'Neill. Order-Preserving Symmetric Encryption. In Antoine Joux, editor, *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, volume 5479 of *Lecture Notes in Computer Science*, pages 224–241. Springer, 2009. ISBN 978-3-642-01000-2.

[38] Alexandra Boldyreva, Nathan Chenette, and Adam O'Neill. Order-Preserving Encryption Revisited: Improved Security Analysis and Alternative Solutions. In Phillip Rogaway, editor, *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, volume 6841 of *Lecture Notes in Computer Science*, pages 578–595. Springer, 2011. ISBN 978-3-642-22791-2.

[39] Dan Boneh. The Decision Diffie-Hellman Problem. In Joe Buhler, editor, *Algorithmic Number Theory, Third International Symposium, ANTS-III, Portland, Oregon, USA, June 21-25, 1998, Proceedings*, volume 1423 of *Lecture Notes in Computer Science*, pages 48–63. Springer, 1998. ISBN 3-540-64657-4.

[40] Dan Boneh and Xavier Boyen. Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, volume 3027 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 2004. ISBN 3-540-21935-8.

[41] Dan Boneh and Matthew K. Franklin. Identity-Based Encryption from the Weil Pairing. In Joe Kilian, editor, *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 2001. ISBN 3-540-42456-3.

[42] Dan Boneh and Matthew K. Franklin. Identity-Based Encryption from the Weil Pairing. *SIAM J. Comput.*, 32(3):586–615, 2003.

[43] Dan Boneh and Brent Waters. Conjunctive, Subset, and Range Queries on Encrypted Data. In Salil P. Vadhan, editor, *Theory of Cryptography, 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, February 21-24, 2007, Proceedings*, volume 4392 of *Lecture Notes in Computer Science*, pages 535–554. Springer, 2007. ISBN 3-540-70935-5.

[44] Dan Boneh, Ben Lynn, and Hovav Shacham. Short Signatures from the Weil Pairing. In Colin Boyd, editor, *Advances in Cryptology - ASIACRYPT*

2001, 7th International Conference on the Theory and Application of Cryptology and Information Security, Gold Coast, Australia, December 9-13, 2001, Proceedings, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532. Springer, 2001. ISBN 3-540-42987-5.

[45] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and Verifiably Encrypted Signatures from Bilinear Maps. In Eli Biham, editor, *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings*, volume 2656 of *Lecture Notes in Computer Science*, pages 416–432. Springer, 2003. ISBN 3-540-14039-5.

[46] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short Group Signatures. In Matthew K. Franklin, editor, *Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer, 2004. ISBN 3-540-22668-0.

[47] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public Key Encryption with Keyword Search. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, volume 3027 of *Lecture Notes in Computer Science*, pages 506–522. Springer, 2004. ISBN 3-540-21935-8.

[48] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical Identity Based Encryption with Constant Size Ciphertext. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, volume 3494 of *Lecture Notes in Computer Science*, pages 440–456. Springer, 2005. ISBN 3-540-25910-4.

[49] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF Formulas on Ciphertexts. In Joe Kilian, editor, *Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005, Proceedings*, volume 3378 of *Lecture Notes in Computer Science*, pages 325–341. Springer, 2005. ISBN 3-540-24573-1.

[50] Dan Boneh, Eyal Kushilevitz, Rafail Ostrovsky, and William E. Skeith III. Public Key Encryption that Allows PIR Queries. In Alfred Menezes, editor, *Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007, Proceedings*, volume 4622 of *Lecture Notes in Computer Science*, pages 50–67. Springer, 2007. ISBN 978-3-540-74142-8.

[51] Dan Boneh, Craig Gentry, Shai Halevi, Frank Wang, and David J. Wu. Private Database Queries Using Somewhat Homomorphic Encryption. In *ACNS*, pages 102–118, 2013.

[52] Prosenjit Bose, Hua Guo, Evangelos Kranakis, Anil Maheshwari, Pat Morin, Jason Morrison, Michiel H. M. Smid, and Yihui Tang. On the False-Positive Rate of Bloom Filters. *Inf. Process. Lett.*, 108(4):210–213, 2008.

[53] Xavier Boyen and Brent Waters. Anonymous Hierarchical Identity-Based Encryption (Without Random Oracles). In Cynthia Dwork, editor, *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings*, volume 4117 of *Lecture Notes in Computer Science*, pages 290–307. Springer, 2006. ISBN 3-540-37432-9.

[54] Zvika Brakerski and Vinod Vaikuntanathan. Fully Homomorphic Encryption from Ring-LWE and Security for Key Dependent Messages. In Phillip Rogaway, editor, *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, volume 6841 of *Lecture Notes in Computer Science*, pages 505–524. Springer, 2011. ISBN 978-3-642-22791-2.

[55] Julien Bringer, Hervé Chabanne, and Bruno Kindarji. Error-Tolerant Searchable Encryption. In *Proceedings of IEEE International Conference on Communications, ICC 2009, Dresden, Germany, 14-18 June 2009*, pages 1–6. IEEE, 2009.

[56] Richard Brinkman, Ling Feng, Jeroen Doumen, Pieter H. Hartel, and Willem Jonker. Efficient Tree Search in Encrypted Data. *Information Systems Security*, 13(3):14–21, 2004.

[57] Jin Wook Byun, Dong Hoon Lee, and Jongin Lim. Efficient Conjunctive Keyword Search on Encrypted Data Storage System. In Andrea S. Atzeni and Antonio Lioy, editors, *Public Key Infrastructure, Third European PKI Workshop: Theory and Practice, EuroPKI 2006, Turin, Italy, June 19-20, 2006, Proceedings*, volume 4043 of *Lecture Notes in Computer Science*, pages 184–196. Springer, 2006. ISBN 3-540-35151-5.

[58] Jin Wook Byun, Hyun Suk Rhee, Hyun-A Park, and Dong Hoon Lee. Off-Line Keyword Guessing Attacks on Recent Keyword Search Schemes over Encrypted Data. In Willem Jonker and Milan Petkovic, editors, *Secure Data Management, Third VLDB Workshop, SDM 2006, Seoul, Korea, September 10-11, 2006, Proceedings*, volume 4165 of *Lecture Notes in Computer Science*, pages 75–83. Springer, 2006. ISBN 978-3-540-38984-2.

[59] Jan Camenisch and Anna Lysyanskaya. Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials. In Moti Yung, editor, *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings*, volume 2442 of *Lecture Notes in Computer Science*, pages 61–76. Springer, 2002. ISBN 3-540-44050-X.

[60] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Compact E-Cash. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, volume 3494 of *Lecture Notes in Computer Science*, pages 302–321. Springer, 2005. ISBN 3-540-25910-4.

[61] Ran Canetti, Shai Halevi, and Jonathan Katz. A Forward-Secure Public-Key Encryption Scheme. In Eli Biham, editor, *Advances in Cryptology*

- *EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings*, volume 2656 of *Lecture Notes in Computer Science*, pages 255–271. Springer, 2003. ISBN 3-540-14039-5.

[62] Angelo De Caro, Vincenzo Iovino, and Giuseppe Persiano. Hidden Vector Encryption Fully Secure Against Unrestricted Queries. *IACR Cryptology ePrint Archive*, 2011:546, 2011.

[63] David Cash, Stanislaw Jarecki, Charanjit S. Jutla, Hugo Krawczyk, Marcel-Catalin Rosu, and Michael Steiner. Highly-Scalable Searchable Symmetric Encryption with Support for Boolean Queries. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 353–373. Springer, 2013. ISBN 978-3-642-40040-7.

[64] Yan-Cheng Chang and Michael Mitzenmacher. Privacy Preserving Keyword Searches on Remote Encrypted Data. In *Proceedings of 3rd Applied Cryptography and Network Security Conference (ACNS)*, pages 442–455, 2005.

[65] Melissa Chase and Seny Kamara. Structured Encryption and Controlled Disclosure. In Masayuki Abe, editor, *ASIACRYPT*, volume 6477 of *LNCS*, pages 577–594. Springer, 2010. ISBN 978-3-642-17372-1.

[66] Sanjit Chatterjee and Alfred Menezes. On cryptographic protocols employing asymmetric pairings - The role of revisited. *Discrete Applied Mathematics*, 159(13):1311–1322, 2011.

[67] David Chaum. Blind Signatures for Untraceable Payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *Advances in Cryptology: Proceedings of CRYPTO '82, Santa Barbara, California, USA, August 23-25, 1982*, pages 199–203. Plenum Press, New York, 1982.

[68] Liqun Chen and Zhaohui Cheng. Security Proof of Sakai-Kasahara's Identity-Based Encryption Scheme. In Nigel P. Smart, editor, *Cryptography and Coding, 10th IMA International Conference, Cirencester, UK, December 19-21, 2005, Proceedings*, volume 3796 of *Lecture Notes in Computer Science*, pages 442–459. Springer, 2005. ISBN 3-540-30276-X.

[69] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private Information Retrieval. In *FOCS '95: Proceedings of the 36th Annu. IEEE Symposium on Foundations of Computer Science*, pages 41–50, 1995.

[70] Benny Chor, Eyal Kushilevitz, Oded Goldreich, and Madhu Sudan. Private Information Retrieval. *Journal of the ACM*, 45(6):965–981, November 1998.

[71] Clifford Cocks. An Identity Based Encryption Scheme Based on Quadratic Residues. In Bahram Honary, editor, *Cryptography and Coding, 8th IMA International Conference, Cirencester, UK, December 17-19, 2001, Proceedings*, volume 2260 of *Lecture Notes in Computer Science*, pages 360–363. Springer, 2001. ISBN 3-540-43026-1.

[72] Ronald Cramer and Victor Shoup. Signature schemes based on the strong RSA assumption. *ACM Trans. Inf. Syst. Secur.*, 3(3):161–185, 2000.

[73] Giovanni Di Crescenzo and Vishal Saraswat. Public Key Encryption with Searchable Keywords Based on Jacobi Symbols. In K. Srinathan, C. Pandu Rangan, and Moti Yung, editors, *Progress in Cryptology - INDOCRYPT 2007, 8th International Conference on Cryptology in India, Chennai, India, December 9-13, 2007, Proceedings*, volume 4859 of *Lecture Notes in Computer Science*, pages 282–296. Springer, 2007. ISBN 978-3-540-77025-1.

[74] Emiliano De Cristofaro, Yanbin Lu, and Gene Tsudik. Efficient Techniques for Privacy-Preserving Sharing of Sensitive Information. In *TRUST*, pages 239–253, 2011.

[75] Reza Curtmola, Juan Garay, Seny Kamara, and Rafail Ostrovsky. Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions. In *CCS '06: Proceedings of the 13th ACM Conference on Computer and Communications Security*, pages 79–88, New York, NY, USA, 2006. ACM. ISBN 1-59593-518-5.

[76] Reza Curtmola, Juan A. Garay, Seny Kamara, and Rafail Ostrovsky. Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions. *Journal of Computer Security*, 19(5):895–934, 2011.

[77] Whitfield Diffie and Martin E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, November 1976.

[78] Changyu Dong, Giovanni Russello, and Naranker Dulay. Shared and Searchable Encrypted Data for Untrusted Servers. In *Proceeedings of the 22nd annual IFIP WG 11.3 working conference on Data and Applications Security*, pages 127–143, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 978-3-540-70566-6.

[79] Taher ElGamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[80] Amos Fiat and Moni Naor. Broadcast Encryption. In Douglas R. Stinson, editor, *Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology Conference, Santa Barbara, California, USA, August 22-26, 1993, Proceedings*, volume 773 of *Lecture Notes in Computer Science*, pages 480–491. Springer, 1994. ISBN 3-540-57766-1.

[81] Internet Watch Foundation. URL List Recipients. https://www.iwf.org.uk/members/member-policies/url-list/iwf-list-recipients, October 2013.

[82] Martin Franz, Peter Williams, Bogdan Carbunar, Stefan Katzenbeisser, Andreas Peter, Radu Sion, and Miroslava Sotáková. Oblivious Outsourced Storage with Delegation. In George Danezis, editor, *Financial Cryptography and Data Security - 15th International Conference, FC 2011, Gros Islet, St. Lucia, February 28 - March 4, 2011, Revised Selected Papers*, volume 7035 of *Lecture Notes in Computer Science*, pages 127–140. Springer, 2011. ISBN 978-3-642-27575-3.

[83] Michael L. Fredman, János Komlós, and Endre Szemerédi. Storing a Sparse Table with o(1) Worst Case Access Time. *J. ACM*, 31(3):538–544, 1984.

[84] D. M. Freeman. Converting Pairing-Based Cryptosystems from Composite-Order Groups to Prime-Order Groups. In H. Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 44–61. Springer, 2010.

[85] Eiichiro Fujisaki and Tatsuaki Okamoto. Statistical Zero Knowledge Protocols to Prove Modular Polynomial Relations. In Burton S. Kaliski Jr., editor, *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings*, volume 1294 of *Lecture Notes in Computer Science*, pages 16–30. Springer, 1997. ISBN 3-540-63384-7.

[86] Eiichiro Fujisaki, Tatsuaki Okamoto, David Pointcheval, and Jacques Stern. RSA-OAEP Is Secure under the RSA Assumption. In Joe Kilian, editor, *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, volume 2139 of *Lecture Notes in Computer Science*, pages 260–274. Springer, 2001. ISBN 3-540-42456-3.

[87] Craig Gentry. Practical Identity-Based Encryption Without Random Oracles. In Serge Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, volume 4004 of *Lecture Notes in Computer Science*, pages 445–464. Springer, 2006. ISBN 3-540-34546-9.

[88] Craig Gentry. Fully Homomorphic Encryption Using Ideal Lattices. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2*, pages 169–178, Bethesda, MD, USA, May 2009. ACM. ISBN 978-1-60558-506-2.

[89] Craig Gentry. Computing Arbitrary Functions of Encrypted Data. *Commun. ACM*, 53(3):97–105, 2010.

[90] Craig Gentry and Alice Silverberg. Hierarchical ID-Based Cryptography. In Yuliang Zheng, editor, *Advances in Cryptology - ASIACRYPT 2002, 8th International Conference on the Theory and Application of Cryptology and Information Security, Queenstown, New Zealand, December 1-5, 2002, Proceedings*, volume 2501 of *Lecture Notes in Computer Science*, pages 548–566. Springer, 2002. ISBN 3-540-00171-9.

[91] Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. A Simple BGN-Type Cryptosystem from LWE. In Henri Gilbert, editor, *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30 - June 3, 2010. Proceedings*, volume 6110 of *Lecture Notes in Computer Science*, pages 506–522. Springer, 2010. ISBN 978-3-642-13189-9.

[92] Eu-Jin Goh. Secure Indexes. Cryptology ePrint Archive, Report 2003/216, 2003. URL http://eprint.iacr.org/2003/216/.

[93] Oded Goldreich. *The Foundations of Cryptography - Volume 2, Basic Applications*. Cambridge University Press, 2004. ISBN 0-521-83084-2.

[94] Oded Goldreich and Rafail Ostrovsky. Software Protection and Simulation on Oblivious RAMs. *J. ACM*, 43(3):431–473, 1996.

[95] Shafi Goldwasser and Silvio Micali. Probabilistic Encryption and How to Play Mental Poker Keeping Secret All Partial Information. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing, 5-7 May 1982, San Francisco, California, USA*, pages 365–377. ACM, 1982.

[96] Shafi Goldwasser and Silvio Micali. Probabilistic Encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

[97] Philippe Golle, Jessica Staddon, and Brent Waters. Secure Conjunctive Keyword Search Over Encrypted Data. In M. Jakobsson, M. Yung, and J. Zhou, editors, *Proceedings of the 2004 Applied Cryptography and Network Security Conference*, pages 31–45. LNCS 3089, 2004.

[98] Google. Google Trends. https://www.google.com/trends/.

[99] David Harvey and William Hart. FLINT: Fast Library for Number Theory. http://www.flintlib.org.

[100] Mitsuhiro Hattori, Takato Hirano, Takashi Ito, Nori Matsuda, Takumi Mori, Yusuke Sakai, and Kazuo Ohta. Ciphertext-Policy Delegatable Hidden Vector Encryption and Its Application to Searchable Encryption in Multi-user Setting. In Liqun Chen, editor, *Cryptography and Coding - 13th IMA International Conference, IMACC 2011, Oxford, UK, December 12-15, 2011. Proceedings*, volume 7089 of *Lecture Notes in Computer Science*, pages 190–209. Springer, 2011. ISBN 978-3-642-25515-1.

[101] Swee-Huay Heng and Kaoru Kurosawa. k-Resilient Identity-Based Encryption in the Standard Model. In Tatsuaki Okamoto, editor, *Topics in Cryptology - CT-RSA 2004, The Cryptographers' Track at the RSA Conference 2004, San Francisco, CA, USA, February 23-27, 2004, Proceedings*, volume 2964 of *Lecture Notes in Computer Science*, pages 67–80. Springer, 2004. ISBN 3-540-20996-4.

[102] Swee-Huay Heng and Kaoru Kurosawa. *k*-Resilient Identity-Based Encryption in the Standard Model. *IEICE Transactions*, 89-A(1):39–46, 2006.

[103] Jeremy Horwitz and Ben Lynn. Toward Hierarchical Identity-Based Encryption. In Lars R. Knudsen, editor, *Advances in Cryptology - EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April 28 - May 2, 2002, Proceedings*, volume 2332 of *Lecture Notes in Computer Science*, pages 466–481. Springer, 2002. ISBN 3-540-43553-0.

[104] Yong Ho Hwang and Pil Joong Lee. Public Key Encryption with Conjunctive Keyword Search and Its Extension to a Multi-user System. In Tsuyoshi Takagi, Tatsuaki Okamoto, Eiji Okamoto, and Takeshi Okamoto, editors, *Pairing-Based Cryptography - Pairing 2007, First International Conference, Tokyo, Japan, July 2-4, 2007, Proceedings*, volume 4575 of *Lecture Notes in Computer Science*, pages 2–22. Springer, 2007. ISBN 978-3-540-73488-8.

[105] Luan Ibraimi, Svetla Nikova, Pieter H. Hartel, and Willem Jonker. Public-Key Encryption with Delegated Search. In Javier Lopez and Gene Tsudik, editors, *Applied Cryptography and Network Security - 9th International Conference, ACNS 2011, Nerja, Spain, June 7-10, 2011. Proceedings*, volume 6715 of *Lecture Notes in Computer Science*, pages 532–549, 2011. ISBN 978-3-642-21553-7.

[106] Vincenzo Iovino and Giuseppe Persiano. Hidden-Vector Encryption with Groups of Prime Order. In Steven D. Galbraith and Kenneth G. Paterson, editors, *Pairing-Based Cryptography - Pairing 2008, Second International Conference, Egham, UK, September 1-3, 2008. Proceedings*, volume 5209 of *Lecture Notes in Computer Science*, pages 75–88. Springer, 2008. ISBN 978-3-540-85503-3.

[107] Antoine Joux. The Weil and Tate Pairings as Building Blocks for Public Key Cryptosystems. In Claus Fieker and David R. Kohel, editors, *Algorithmic Number Theory, 5th International Symposium, ANTS-V, Sydney, Australia, July 7-12, 2002, Proceedings*, volume 2369 of *Lecture Notes in Computer Science*, pages 20–32. Springer, 2002. ISBN 3-540-43863-7.

[108] Seny Kamara and Charalampos Papamanthou. Parallel and Dynamic Searchable Symmetric Encryption. In *Financial Cryptography and Data Security 2013 - 17th International Conference, FC 2013, Okinawa, Japan, April 1-5, 2013*, 2013.

[109] Seny Kamara, Charalampos Papamanthou, and Tom Roeder. Dynamic Searchable Symmetric Encryption. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *the ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16-18, 2012*, pages 965–976. ACM, 2012. ISBN 978-1-4503-1651-4.

[110] Murat Kantarcioglu and Chris Clifton. Security Issues in Querying Encrypted Data. In Sushil Jajodia and Duminda Wijesekera, editors, *Data and Applications Security XIX, 19th Annual IFIP WG 11.3 Working Conference on Data and Applications Security, Storrs, CT, USA, August 7-10, 2005, Proceedings*, volume 3654 of *Lecture Notes in Computer Science*, pages 325–337. Springer, 2005. ISBN 3-540-28138-X.

[111] Nikolaos P. Karvelas, Andreas Peter, Stefan Katzenbeisser, and Sebastian Biedermann. Efficient Privacy-Preserving Big Data Processing through Proxy-Assisted ORAM. Cryptology ePrint Archive, Report 2014/072, 2014. http://eprint.iacr.org/.

[112] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products. In Nigel P. Smart, editor, *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*, volume 4965 of *Lecture Notes in Computer Science*, pages 146–162. Springer, 2008. ISBN 978-3-540-78966-6.

[113] Dalia Khader. Public Key Encryption with Keyword Search Based on K-Resilient IBE. In Osvaldo Gervasi and Marina L. Gavrilova, editors, *Computational Science and Its Applications - ICCSA 2007, International Confer-*

ence, Kuala Lumpur, Malaysia, August 26-29, 2007. Proceedings. Part III, volume 4707 of *Lecture Notes in Computer Science*, pages 1086–1095. Springer, 2007. ISBN 978-3-540-74482-5.

[114] Aggelos Kiayias, Yiannis Tsiounis, and Moti Yung. Group Encryption. In Kaoru Kurosawa, editor, *Advances in Cryptology - ASIACRYPT 2007, 13th International Conference on the Theory and Application of Cryptology and Information Security, Kuching, Malaysia, December 2-6, 2007, Proceedings*, volume 4833 of *Lecture Notes in Computer Science*, pages 181–199. Springer, 2007. ISBN 978-3-540-76899-9.

[115] Eike Kiltz. From Selective-ID to Full Security: The Case of the Inversion-Based Boneh-Boyen IBE Scheme. Cryptology ePrint Archive, Report 2007/033, 2007. URL http://eprint.iacr.org/.

[116] Eike Kiltz and David Galindo. Direct Chosen-Ciphertext Secure Identity-Based Key Encapsulation Without Random Oracles. In Lynn Margaret Batten and Reihaneh Safavi-Naini, editors, *ACISP*, volume 4058 of *LNCS*, pages 336–347. Springer, 2006. ISBN 3-540-35458-1.

[117] Kaoru Kurosawa. Multi-recipient Public-Key Encryption with Shortened Ciphertext. In David Naccache and Pascal Paillier, editors, *Public Key Cryptography, 5th International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2002, Paris, France, February 12-14, 2002, Proceedings*, volume 2274 of *Lecture Notes in Computer Science*, pages 48–63. Springer, 2002. ISBN 3-540-43168-3.

[118] Kaoru Kurosawa and Yvo Desmedt. A New Paradigm of Hybrid Encryption Scheme. In Matthew K. Franklin, editor, *Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, volume 3152 of *Lecture Notes in Computer Science*, pages 426–442. Springer, 2004. ISBN 3-540-22668-0.

[119] Kaoru Kurosawa and Yasuhiro Ohtaki. UC-Secure Searchable Symmetric Encryption. In Angelos D. Keromytis, editor, *Financial Cryptography and Data Security - 16th International Conference, FC 2012, Kralendijk, Bonaire, Februray 27-March 2, 2012, Revised Selected Papers*, volume 7397 of *Lecture Notes in Computer Science*, pages 285–298. Springer, 2012. ISBN 978-3-642-32945-6.

[120] Mehmet Kuzu, Mohammad Saiful Islam, and Murat Kantarcioglu. Efficient Similarity Search over Encrypted Data. In Anastasios Kementsietsidis and Marcos Antonio Vaz Salles, editors, *IEEE 28th International Conference on Data Engineering (ICDE 2012), Washington, DC, USA (Arlington, Virginia), 1-5 April, 2012*, pages 1156–1167. IEEE Computer Society, 2012. ISBN 978-0-7695-4747-3.

[121] Kristin Lauter, Michael Naehrig, and Vinod Vaikuntanathan. Can Homomorphic Encryption be Practical? In *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, CCSW '11, pages 113–124, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-1004-8.

[122] Kwangsu Lee and Dong Hoon Lee. New Techniques for Anonymous HIBE with Short Ciphertexts in Prime Order Groups. *TIIS*, 4(5):968–988, 2010.

[123] Kwangsu Lee and Dong Hoon Lee. Improved hidden vector encryption with short ciphertexts and tokens. *Des. Codes Cryptography*, 58(3):297–319, 2011.

[124] Allison B. Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully Secure Functional Encryption: Attribute-Based Encryption and (Hierarchical) Inner Product Encryption. In Henri Gilbert, editor, *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30 - June 3, 2010. Proceedings*, volume 6110 of *Lecture Notes in Computer Science*, pages 62–91. Springer, 2010. ISBN 978-3-642-13189-9.

[125] Jin Li, Qian Wang, Cong Wang, Ning Cao, Kui Ren, and Wenjing Lou. Fuzzy Keyword Search over Encrypted Data in Cloud Computing. In *INFOCOM 2010. 29th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 15-19 March 2010, San Diego, CA, USA*, pages 441–445. IEEE, 2010. ISBN 978-1-4244-5838-7.

[126] Richard Lindner and Chris Peikert. Better Key Sizes (and Attacks) for LWE-Based Encryption. In Aggelos Kiayias, editor, *Topics in Cryptology - CT-RSA 2011 - The Cryptographers' Track at the RSA Conference 2011, San Francisco, CA, USA, February 14-18, 2011. Proceedings*, volume 6558 of *Lecture Notes in Computer Science*, pages 319–339. Springer, 2011. ISBN 978-3-642-19073-5.

[127] Helger Lipmaa, David Wagner, and Phillip Rogaway. Comments to NIST Concerning AES Modes of Operation: CTR-Mode Encryption. 2000.

[128] Chang Liu, Liehuang Zhu, Mingzhong Wang, and Yu an Tan. Search Pattern Leakage in Searchable Encryption: Attacks and New Construction. *Inf. Sci.*, 265:176–188, 2014.

[129] Qin Liu, Guojun Wang, and Jie Wu. An Efficient Privacy Preserving Keyword Search Scheme in Cloud Computing. In *Proceedings IEEE CSE'09, 12th IEEE International Conference on Computational Science and Engineering, August 29-31, 2009, Vancouver, BC, Canada*, pages 715–720. IEEE Computer Society, 2009.

[130] Ben Lynn. The Pairing-Based Cryptography (PBC) library. http://crypto.stanford.edu/pbc.

[131] Vadim Lyubashevsky, Daniele Micciancio, Chris Peikert, and Alon Rosen. SWIFFT: A Modest Proposal for FFT Hashing. In Kaisa Nyberg, editor, *Fast Software Encryption, 15th International Workshop, FSE 2008, Lausanne, Switzerland, February 10-13, 2008, Revised Selected Papers*, volume 5086 of *Lecture Notes in Computer Science*, pages 54–72. Springer, 2008. ISBN 978-3-540-71038-7.

[132] Daniele Micciancio and Oded Regev. *Lattice-Based Cryptography*, pages 147–191. Springer-Verlag, February 2009.

[133] S. Mitsunari, R. Sakai, and M. Kasahara. A New Traitor Tracing. *EICE Trans Fundam Electron Commun Comput Sci (Inst Electron Inf Commun Eng)*, E85-A:481–484, 2002. ISSN 0916-8508.

[134] Michael Naehrig, Kristin Lauter, and Vinod Vaikuntanathan. Can homomorphic encryption be practical? In Christian Cachin and Thomas Ristenpart, editors, *Proceedings of the 3rd ACM Cloud Computing Security Workshop, CCSW 2011, Chicago, IL, USA, October 21, 2011*, pages 113–124. ACM, 2011. ISBN 978-1-4503-1004-8.

[135] Takashi Nishide, Kazuki Yoneyama, and Kazuo Ohta. Attribute-Based Encryption with Partially Hidden Encryptor-Specified Access Structures. In Steven M. Bellovin, Rosario Gennaro, Angelos D. Keromytis, and Moti Yung, editors, *Applied Cryptography and Network Security, 6th International Conference, ACNS 2008, New York, NY, USA, June 3-6, 2008. Proceedings*, volume 5037 of *Lecture Notes in Computer Science*, pages 111–129, 2008. ISBN 978-3-540-68913-3.

[136] Takashi Nishide, Kazuki Yoneyama, and Kazuo Ohta. Attribute-Based Encryption with Partially Hidden Ciphertext Policies. *IEICE Transactions*, 92-A(1):22–32, 2009.

[137] Kaisa Nyberg. Fast Accumulated Hashing. In Dieter Gollmann, editor, *Fast Software Encryption, Third International Workshop, Cambridge, UK, February 21-23, 1996, Proceedings*, volume 1039 of *Lecture Notes in Computer Science*, pages 83–87. Springer, 1996. ISBN 3-540-60865-6.

[138] Tatsuaki Okamoto and David Pointcheval. The Gap-Problems: A New Class of Problems for the Security of Cryptographic Schemes. In Kwangjo Kim, editor, *Public Key Cryptography, 4th International Workshop on Practice and Theory in Public Key Cryptography, PKC 2001, Cheju Island, Korea, February 13-15, 2001, Proceedings*, volume 1992 of *Lecture Notes in Computer Science*, pages 104–118. Springer, 2001. ISBN 3-540-41658-7.

[139] Tatsuaki Okamoto and Katsuyuki Takashima. Hierarchical Predicate Encryption for Inner-Products. In Mitsuru Matsui, editor, *Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings*, volume 5912 of *Lecture Notes in Computer Science*, pages 214–231. Springer, 2009. ISBN 978-3-642-10365-0.

[140] Tatsuaki Okamoto and Katsuyuki Takashima. Fully Secure Functional Encryption with General Relations from the Decisional Linear Assumption. In Tal Rabin, editor, *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, volume 6223 of *Lecture Notes in Computer Science*, pages 191–208. Springer, 2010. ISBN 978-3-642-14622-0.

[141] Tatsuaki Okamoto and Katsuyuki Takashima. Adaptively Attribute-Hiding (Hierarchical) Inner Product Encryption. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology - EUROCRYPT*

2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings, volume 7237 of *Lecture Notes in Computer Science*, pages 591–608. Springer, 2012. ISBN 978-3-642-29010-7.

[142] Femi Olumofin and Ian Goldberg. Revisiting the Computational Practicality of Private Information Retrieval. In *Financial Cryptography and Data Security*, 2011.

[143] Rafail Ostrovsky. Efficient Computation on Oblivious RAMs. In *Proceedings of the Twenty Second Annual ACM Symposium on Theory of Computing, 14-16 May 1990, Baltimore, Maryland, USA*, pages 514–523. ACM, 1990.

[144] Rafail Ostrovsky. *Software Protection and Simulations on Oblivious RAMs*. PhD thesis, MIT, 1992.

[145] Rafail Ostrovsky and William E. Skeith III. Private Searching on Streaming Data. In Victor Shoup, editor, *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, volume 3621 of *Lecture Notes in Computer Science*, pages 223–240. Springer, 2005. ISBN 3-540-28114-2.

[146] Rafail Ostrovsky and William Skeith. A Survey of Single-Database Private Information Retrieval: Techniques and Applications. In *Public Key Cryptography â PKC 2007: 10th International Conference on Practice and Theory in Public-Key Cryptography, Beijing, China, April 16-20, 2007, Proceedings*, volume 4450, pages 393–411. Springer Berlin / Heidelberg, April 2007.

[147] Rafail Ostrovsky and William E. Skeith. Private Searching on Streaming Data. *Journal of Cryptology*, 20(4):397–430, October 2007.

[148] Pascal Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In Jacques Stern, editor, *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 1999. ISBN 3-540-65889-0.

[149] Dong Jin Park, Kihyun Kim, and Pil Joong Lee. Public Key Encryption with Conjunctive Field Keyword Search. In Chae Hoon Lim and Moti Yung, editors, *Information Security Applications, 5th International Workshop, WISA 2004, Jeju Island, Korea, August 23-25, 2004, Revised Selected Papers*, volume 3325 of *Lecture Notes in Computer Science*, pages 73–86. Springer, 2004. ISBN 3-540-24015-2.

[150] Dong Jin Park, Juyoung Cha, and Pil Joong Lee. Searchable Keyword-Based Encryption. *IACR Cryptology ePrint Archive*, 2005:367, 2005.

[151] Hyun-A Park, Bum Han Kim, Dong Hoon Lee, Yon Dohn Chung, and Justin Zhan. Secure Similarity Search. In *2007 IEEE International Conference on Granular Computing, GrC 2007, San Jose, California, USA, 2-4 November 2007*, pages 598–604. IEEE, November 2007.

[152] Jong Hwan Park. Inner-product encryption under standard assumptions. *Des. Codes Cryptography*, 58(3):235–257, 2011.

[153] Jong Hwan Park and Dong Hoon Lee. A Hidden Vector Encryption Scheme with Constant-Size Tokens and Pairing Computations. *IEICE Transactions*, 93-A(9):1620–1631, 2010.

[154] Andreas Peter, Thomas Hartmann, Sascha Müller, and Stefan Katzenbeisser. Privacy-Preserving Architecture for Forensic Image Recognition. In *2012 IEEE International Workshop on Information Forensics and Security, WIFS 2012, Costa Adeje, Tenerife, Spain, December 2-5, 2012*, pages 79–84. IEEE, 2012. ISBN 978-1-4673-2285-0.

[155] Andreas Peter, Erik Tews, and Stefan Katzenbeisser. Efficiently Outsourcing Multiparty Computation Under Multiple Keys. *IEEE Transactions on Information Forensics and Security*, 8(12):2046–2058, 2013.

[156] Raluca A. Popa, Catherine M. S. Redfield, Nickolai Zeldovich, and Hari Balakrishnan. CryptDB: protecting confidentiality with encrypted query processing. In *SOSP*, pages 85–100. ACM, 2011. ISBN 978-1-4503-0977-6.

[157] Mariana Raykova, Binh Vo, Steven M. Bellovin, and Tal Malkin. Secure Anonymous Database Search. In Radu Sion and Dawn Song, editors, *Proceedings of the first ACM Cloud Computing Security Workshop, CCSW 2009, Chicago, IL, USA, November 13, 2009*, pages 115–126. ACM, 2009. ISBN 978-1-60558-784-4.

[158] Hyun Sook Rhee, Jong Hwan Park, Willy Susilo, and Dong Hoon Lee. Improved Searchable Public Key Encryption with Designated Tester. In Wanqing Li, Willy Susilo, Udaya Kiran Tupakula, Reihaneh Safavi-Naini, and Vijay Varadharajan, editors, *Proceedings of the 2009 ACM Symposium on Information, Computer and Communications Security, ASIACCS 2009, Sydney, Australia, March 10-12, 2009*, pages 376–379. ACM, 2009. ISBN 978-1-60558-394-5.

[159] Hyun Sook Rhee, Jong Hwan Park, Willy Susilo, and Dong Hoon Lee. Trapdoor Security in a Searchable Public-Key Encryption Scheme with a Designated Tester. *Journal of Systems and Software*, 83(5):763–771, 2010.

[160] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[161] Eun-Kyung Ryu and Tsuyoshi Takagi. Efficient Conjunctive Keyword-Searchable Encryption. In *AINAW '07: Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops*, pages 409–414, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-2847-3.

[162] Amit Sahai and Brent Waters. Fuzzy Identity-Based Encryption. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, volume 3494 of *Lecture Notes in Computer Science*, pages 457–473. Springer, 2005. ISBN 3-540-25910-4.

[163] Mike Scott. Authenticated ID-based Key Exchange and remote log-in with simple token and PIN number. *IACR Cryptology ePrint Archive*, 2002:164, 2002.

[164] Saeed Sedghi, Peter van Liesdonk, Svetla Nikova, Pieter H. Hartel, and Willem Jonker. Searching Keywords with Wildcards on Encrypted Data. In Juan A. Garay and Roberto De Prisco, editors, *Security and Cryptography for Networks, 7th International Conference, SCN 2010, Amalfi, Italy, September 13-15, 2010. Proceedings*, volume 6280 of *Lecture Notes in Computer Science*, pages 138–153. Springer, 2010. ISBN 978-3-642-15316-7.

[165] Adi Shamir. How to Share a Secret. *Communications of the ACM*, 22(11): 612–613, November 1979.

[166] Adi Shamir. Identity-Based Cryptosystems and Signature Schemes. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer, 1985. ISBN 3-540-15658-5.

[167] Emily Shen, Elaine Shi, and Brent Waters. Predicate Privacy in Encryption Systems. In Omer Reingold, editor, *Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009, San Francisco, CA, USA, March 15-17, 2009. Proceedings*, volume 5444 of *Lecture Notes in Computer Science*, pages 457–473. Springer, 2009. ISBN 978-3-642-00456-8.

[168] Elaine Shi and Brent Waters. Delegating Capabilities in Predicate Encryption Systems. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part II - Track B: Logic, Semantics, and Theory of Programming & Track C: Security and Cryptography Foundations*, volume 5126 of *Lecture Notes in Computer Science*, pages 560–578. Springer, 2008. ISBN 978-3-540-70582-6.

[169] Elaine Shi, John Bethencourt, Hubert T.-H. Chan, Dawn Xiaodong Song, and Adrian Perrig. Multi-Dimensional Range Query over Encrypted Data. In *2007 IEEE Symposium on Security and Privacy (S&P 2007), 20-23 May 2007, Oakland, California, USA*, pages 350–364. IEEE Computer Society, 2007.

[170] V. Shoup. ISO 18033-2: An Emerging Standard for Public-Key Encryption (committee draft). Available at http://shoup.net/iso/, June 2004.

[171] Dawn Xiaodong Song, David Wagner, and Adrian Perrig. Practical Techniques for Searches on Encrypted Data. In *SP '00: Proceedings of the 2000 IEEE Symposium on Security and Privacy*, pages 44–55, Washington, DC, USA, 2000. IEEE Computer Society. ISBN 0-7695-0665-8.

[172] Qiang Tang. Towards Public Key Encryption Scheme Supporting Equality Test with Fine-Grained Authorization. In Udaya Parampalli and Philip Hawkes, editors, *ACISP*, volume 6812 of *LNCS*, pages 389–406. Springer, 2011. ISBN 978-3-642-22496-6.

[173] Qiang Tang. Public Key Encryption Supporting Plaintext Equality Test and User-Specified Authorization. *Security and Communication Networks*, 5(12):1351–1362, 2012.

[174] Qiang Tang. Public Key Encryption Schemes Supporting Equality Test with Authorisation of Different Granularity. *IJACT*, 2(4):304–321, 2012.

[175] Qiang Tang and Liqun Chen. Public-Key Encryption with Registered Keyword Search. In Fabio Martinelli and Bart Preneel, editors, *Public Key Infrastructures, Services and Applications - 6th European Workshop, EuroPKI 2009, Pisa, Italy, September 10-11, 2009, Revised Selected Papers*, volume 6391 of *Lecture Notes in Computer Science*, pages 163–178. Springer, 2009. ISBN 978-3-642-16440-8.

[176] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully Homomorphic Encryption over the Integers. In Henri Gilbert, editor, *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30 - June 3, 2010. Proceedings*, volume 6110 of *Lecture Notes in Computer Science*, pages 24–43. Springer, 2010. ISBN 978-3-642-13189-9.

[177] Peter van Liesdonk, Saeed Sedghi, Jeroen Doumen, Pieter H. Hartel, and Willem Jonker. Computationally Efficient Searchable Symmetric Encryption. In Willem Jonker and Milan Petkovic, editors, *Secure Data Management, 7th VLDB Workshop, SDM 2010, Singapore, September 17, 2010. Proceedings*, volume 6358 of *Lecture Notes in Computer Science*, pages 87–100. Springer, 2010. ISBN 978-3-642-15545-1.

[178] Ramarathnam Venkatesan, S.-M. Koon, Mariusz H. Jakubowski, and Pierre Moulin. Robust Image Hashing. In *Proceedings of the International Conference on Image Processing, ICIP 2000, September 10-13, Vancouver, BC, Canada*, 2000.

[179] Peishun Wang, Huaxiong Wang, and Josef Pieprzyk. Common Secure Index for Conjunctive Keyword-Based Retrieval over Encrypted Data. In Willem Jonker and Milan Petkovic, editors, *Secure Data Management, 4th VLDB Workshop, SDM 2007, Vienna, Austria, September 23-24, 2007, Proceedings*, volume 4721 of *Lecture Notes in Computer Science*, pages 108–123. Springer, September 2007. ISBN 978-3-540-75247-9.

[180] Peishun Wang, Huaxiong Wang, and Josef Pieprzyk. A New Dynamic Accumulator for Batch Updates. In Sihan Qing, Hideki Imai, and Guilin Wang, editors, *Information and Communications Security, 9th International Conference, ICICS 2007, Zhengzhou, China, December 12-15, 2007, Proceedings*, volume 4861 of *Lecture Notes in Computer Science*, pages 98–112. Springer, 2007. ISBN 978-3-540-77047-3.

[181] Peishun Wang, Huaxiong Wang, and Josef Pieprzyk. An Efficient Scheme of Common Secure Indices for Conjunctive Keyword-Based Retrieval on Encrypted Data. In Kyo-Il Chung, Kiwook Sohn, and Moti Yung, editors, *Information Security Applications, 9th International Workshop, WISA 2008, Jeju Island, Korea, September 23-25, 2008, Revised Selected Papers*, volume 5379 of *Lecture Notes in Computer Science*, pages 145–159. Springer, 2008. ISBN 978-3-642-00305-9.

[182] Peishun Wang, Huaxiong Wang, and Josef Pieprzyk. Keyword Field-Free Conjunctive Keyword Searches on Encrypted Data and Extension for Dynamic Groups. In Matthew K. Franklin, Lucas Chi Kwong Hui, and Duncan S. Wong, editors, *Cryptology and Network Security, 7th International Conference, CANS 2008, Hong-Kong, China, December 2-4, 2008. Proceedings*, volume 5339 of *Lecture Notes in Computer Science*, pages 178–195. Springer, December 2008. ISBN 978-3-540-89640-1.

[183] Peishun Wang, Huaxiong Wang, and Josef Pieprzyk. Threshold Privacy Preserving Keyword Searches. In Viliam Geffert, Juhani Karhumäki, Alberto Bertoni, Bart Preneel, Pavol Návrat, and Mária Bieliková, editors, *SOFSEM 2008: Theory and Practice of Computer Science, 34th Conference on Current Trends in Theory and Practice of Computer Science, Nový Smokovec, Slovakia, January 19-25, 2008, Proceedings*, volume 4910 of *Lecture Notes in Computer Science*, pages 646–658. Springer, January 2008. ISBN 978-3-540-77565-2.

[184] Guomin Yang, Chik How Tan, Qiong Huang, and Duncan S. Wong. Probabilistic Public Key Encryption with Equality Test. In Josef Pieprzyk, editor, *CT-RSA*, volume 5985 of *LNCS*, pages 119–131. Springer, 2010. ISBN 978-3-642-11924-8.

[185] Yanjiang Yang, Feng Bao, Xuhua Ding, and Robert H. Deng. Multiuser private queries over encrypted databases. *IJACT*, 1(4):309–319, 2009.

[186] Yanjiang Yang, Haibing Lu, and Jian Weng. Multi-User Private Keyword Search for Cloud Computing. In Costas Lambrinoudakis, Panagiotis Rizomiliotis, and Tomasz Wiktor Wlodarczyk, editors, *IEEE 3rd International Conference on Cloud Computing Technology and Science, CloudCom 2011, Athens, Greece, November 29 - December 1, 2011*, pages 264–271. IEEE, 2011. ISBN 978-1-4673-0090-2.

[187] Andrew Chi-Chih Yao. Protocols for Secure Computations (Extended Abstract). In *IEEE Symposium on Foundations of Computer Science (FOCS '82)*, pages 160–164. IEEE Computer Society, 1982.

[188] Wei-Chuen Yau, Swee-Huay Heng, and Bok-Min Goi. Off-Line Keyword Guessing Attacks on Recent Public Key Encryption with Keyword Search Schemes. In Chunming Rong, Martin Gilje Jaatun, Frode Eika Sandnes, Laurence Tianruo Yang, and Jianhua Ma, editors, *Autonomic and Trusted Computing, 5th International Conference, ATC 2008, Oslo, Norway, June 23-25, 2008, Proceedings*, volume 5060 of *Lecture Notes in Computer Science*, pages 100–105. Springer, 2008. ISBN 978-3-540-69294-2.

[189] Dae Hyun Yum, Duk Soo Kim, Jin Seok Kim, Pil Joong Lee, and Sung Je Hong. Order-Preserving Encryption for Non-uniformly Distributed Plaintexts. In *Information Security Applications - 12th International Workshop, WISA 2011, Jeju Island, Korea, August 22-24, 2011. Revised Selected Papers*, volume 7115 of *Lecture Notes in Computer Science*, pages 84–97. Springer, 2011. ISBN 978-3-642-27889-1.

[190] Christoph Zauner. Implementation and Benchmarking of Perceptual Image Hash Functions. Master's thesis, Upper Austria University of Applied Sciences, 2010. http://www.phash.org/docs/pubs/thesis_zauner.pdf.

[191] Rui Zhang and Hideki Imai. Combining Public Key Encryption with Keyword Search and Public Key Encryption. *IEICE Transactions*, 92-D(5): 888–896, 2009.

This page intentionally left blank.

2009

2009-01 Rasa Jurgelenaite (RUN)
Symmetric Causal Independence Models

2009-02 Willem Robert van Hage (VU)
Evaluating Ontology-Alignment Techniques

2009-01 Rasa Jurgelenaite (RUN)
Symmetric Causal Independence Models

2009-02 Willem Robert van Hage (VU)
Evaluating Ontology-Alignment Techniques

2009-03 Hans Stol (UvT)
A Framework for Evidence-based Policy Making Using IT

2009-04 Josephine Nabukenya (RUN)
Improving the Quality of Organisational Policy Making using Collaboration Engineering

2009-05 Sietse Overbeek (RUN)
Bridging Supply and Demand for Knowledge Intensive Tasks - Based on Knowledge, Cognition, and Quality

2009-06 Muhammad Subianto (UU)
Understanding Classification

2009-07 Ronald Poppe (UT)
Discriminative Vision-Based Recovery and Recognition of Human Motion

2009-08 Volker Nannen (VU)
Evolutionary Agent-Based Policy Analysis in Dynamic Environments

2009-09 Benjamin Kanagwa (RUN)
Design, Discovery and Construction of Service-oriented Systems

2009-10 Jan Wielemaker (UVA)
Logic programming for knowledge-intensive interactive applications

2009-11 Alexander Boer (UVA)
Legal Theory, Sources of Law & the Semantic Web

2009-12 Peter Massuthe (TUE, Humboldt-Universitaet zu Berlin)
Operating Guidelines for Services

2009-13 Steven de Jong (UM)
Fairness in Multi-Agent Systems

2009-14 Maksym Korotkiy (VU)
From ontology-enabled services to service-enabled ontologies (making ontologies work in e-science with ONTO-SOA)

2009-15 Rinke Hoekstra (UVA)
Ontology Representation - Design Patterns and Ontologies that Make Sense

2009-16 Fritz Reul (UvT)
New Architectures in Computer Chess

2009-17 Laurens van der Maaten (UvT)
Feature Extraction from Visual Data

2009-18 Fabian Groffen (CWI)
Armada, An Evolving Database System

2009-19 Valentin Robu (CWI)
Modeling Preferences, Strategic Reasoning and Collaboration in Agent-Mediated Electronic Markets

2009-20 Bob van der Vecht (UU)
Adjustable Autonomy: Controling Influences on Decision Making

2009-21 Stijn Vanderlooy (UM)
Ranking and Reliable Classification

2009-22 Pavel Serdyukov (UT)
Search For Expertise: Going beyond direct evidence

2009-23 Peter Hofgesang (VU)
Modelling Web Usage in a Changing Environment

2009-24 Annerieke Heuvelink (VUA)
Cognitive Models for Training Simulations

2009-25 Alex van Ballegooij (CWI)
"RAM: Array Database Management through Relational Mapping"

2009-26 Fernando Koch (UU)
An Agent-Based Model for the Development of Intelligent Mobile Services

2009-27 Christian Glahn (OU)
Contextual Support of social Engagement and Reflection on the Web

2009-28 Sander Evers (UT)
Sensor Data Management with Probabilistic Models

2009-29 Stanislav Pokraev (UT)
Model-Driven Semantic Integration of Service-Oriented Applications

2009-30 Marcin Zukowski (CWI)
Balancing vectorized query execution with bandwidth-optimized storage

2009-31 Sofiya Katrenko (UVA)
A Closer Look at Learning Relations from Text

2009-32 Rik Farenhorst (VU) and Remco de Boer (VU)
Architectural Knowledge Management: Supporting Architects and Auditors

2009-33 Khiet Truong (UT)
How Does Real Affect Affect Affect Recognition In Speech?

2009-34 Inge van de Weerd (UU)
Advancing in Software Product Management: An Incremental Method Engineering Approach

2009-35 Wouter Koelewijn (UL)
Privacy en Politiegegevens; Over geautomatiseerde normatieve informatie-uitwisseling

2009-36 Marco Kalz (OUN)
Placement Support for Learners in Learning Networks

2009-37 Hendrik Drachsler (OUN)
Navigation Support for Learners in Informal Learning Networks

2009-38 Riina Vuorikari (OU)
Tags and self-organisation: a metadata ecology for learning resources in a multilingual context

2009-39 Christian Stahl (TUE, Humboldt-Universitaet zu Berlin)
Service Substitution – A Behavioral Approach Based on Petri Nets

2009-40 Stephan Raaijmakers (UvT)
Multinomial Language Learning: Investigations into the Geometry of Language

2009-41 Igor Berezhnyy (UvT)
Digital Analysis of Paintings

2009-42 Toine Bogers (UvT)
Recommender Systems for Social Bookmarking

2009-43 Virginia Nunes Leal Franqueira (UT)
Finding Multi-step Attacks in Computer Networks using Heuristic Search and Mobile Ambients

2009-44 Roberto Santana Tapia (UT)
Assessing Business-IT Alignment in Networked Organizations

2009-45 Jilles Vreeken (UU)
Making Pattern Mining Useful

2009-46 Loredana Afanasiev (UvA)
Querying XML: Benchmarks and Recursion

2010

2010-01 Matthijs van Leeuwen (UU)
Patterns that Matter

2010-02 Ingo Wassink (UT)
Work flows in Life Science

2010-03 Joost Geurts (CWI)
A Document Engineering Model and Processing Framework for Multimedia documents

2010-04 Olga Kulyk (UT)
Do You Know What I Know? Situational Awareness of Co-located Teams in Multidisplay Environments

2010-05 Claudia Hauff (UT)
Predicting the Effectiveness of Queries and Retrieval Systems

2010-06 Sander Bakkes (UvT)
Rapid Adaptation of Video Game AI

2010-07 Wim Fikkert (UT)
Gesture interaction at a Distance

2010-08 Krzysztof Siewicz (UL)
Towards an Improved Regulatory Framework of Free Software. Protecting user freedoms in a world of software communities and eGovernments

2010-09 Hugo Kielman (UL)
A Politiele gegevensverwerking en Privacy, Naar een effectieve waarborging

2010-10 Rebecca Ong (UL)
Mobile Communication and Protection of Children

2010-11 Adriaan Ter Mors (TUD)
The world according to MARP: Multi-Agent Route Planning

2010-12 Susan van den Braak (UU)
Sensemaking software for crime analysis

2010-13 Gianluigi Folino (RUN)
High Performance Data Mining using Bio-inspired techniques

2010-14 Sander van Splunter (VU)
Automated Web Service Reconfiguration

2010-15 Lianne Bodenstaff (UT)
Managing Dependency Relations in Inter-Organizational Models

2010-16 Sicco Verwer (TUD)
Efficient Identification of Timed Automata, theory and practice

2010-17 Spyros Kotoulas (VU)
Scalable Discovery of Networked Resources: Algorithms, Infrastructure, Applications

2010-18 Charlotte Gerritsen (VU)
Caught in the Act: Investigating Crime by Agent-Based Simulation

2010-19 Henriette Cramer (UvA)
People's Responses to Autonomous and Adaptive Systems

2010-20 Ivo Swartjes (UT)
Whose Story Is It Anyway? How Improv Informs Agency and Authorship of Emergent Narrative

2010-21 Harold van Heerde (UT)
Privacy-aware data management by means of data degradation

2010-22 Michiel Hildebrand (CWI)
End-user Support for Access to Heterogeneous Linked Data

2010-23 Bas Steunebrink (UU)
The Logical Structure of Emotions

2010-24 Dmytro Tykhonov
Designing Generic and Efficient Negotiation Strategies

2010-25 Zulfiqar Ali Memon (VU)
Modelling Human-Awareness for Ambient Agents: A Human Mindreading Perspective

2010-26 Ying Zhang (CWI)
XRPC: Efficient Distributed Query Processing on Heterogeneous XQuery Engines

2010-27 Marten Voulon (UL)
Automatisch contracteren

2010-28 Arne Koopman (UU)
Characteristic Relational Patterns

2010-29 Stratos Idreos(CWI)
Database Cracking: Towards Auto-tuning Database Kernels

2010-30 Marieke van Erp (UvT)
Accessing Natural History - Discoveries in data cleaning, structuring, and retrieval

2010-31 Victor de Boer (UVA)
Ontology Enrichment from Heterogeneous Sources on the Web

2010-32 Marcel Hiel (UvT)
An Adaptive Service Oriented Architecture: Automatically solving Interoperability Problems

2010-33 Robin Aly (UT)
Modeling Representation Uncertainty in Concept-Based Multimedia Retrieval

2010-34 Teduh Dirgahayu (UT)
Interaction Design in Service Compositions

2010-35 Dolf Trieschnigg (UT)
Proof of Concept: Concept-based Biomedical Information Retrieval

2010-36 Jose Janssen (OU)
Paving the Way for Lifelong Learning; Facilitating competence development through a learning path specification

2010-37 Niels Lohmann (TUE)
Correctness of services and their composition

2010-38 Dirk Fahland (TUE)
From Scenarios to components

2010-39 Ghazanfar Farooq Siddiqui (VU)
Integrative modeling of emotions in virtual agents

2010-40 Mark van Assem (VU)
Converting and Integrating Vocabularies for the Semantic Web

2010-41 Guillaume Chaslot (UM)
Monte-Carlo Tree Search

2010-42 Sybren de Kinderen (VU)
Needs-driven service bundling in a multi-supplier setting - the computational e3-service approach

2010-43 Peter van Kranenburg (UU)
A Computational Approach to Content-Based Retrieval of Folk Song Melodies

2010-44 Pieter Bellekens (TUE)
An Approach towards Context-sensitive and User-adapted Access to Heterogeneous Data Sources, Illustrated in the Television Domain

2010-45 Vasilios Andrikopoulos (UvT)
A theory and model for the evolution of software services

2010-46 Vincent Pijpers (VU)
e3alignment: Exploring Inter-Organizational Business-ICT Alignment

2010-47 Chen Li (UT)
Mining Process Model Variants: Challenges, Techniques, Examples

2010-48 Withdrawn

2010-49 Jahn-Takeshi Saito (UM)
Solving difficult game positions

2010-50 Bouke Huurnink (UVA)
Search in Audiovisual Broadcast Archives

2010-51 Alia Khairia Amin (CWI)
Understanding and supporting information seeking tasks in multiple sources

2010-52 Peter-Paul van Maanen (VU)
Adaptive Support for Human-Computer Teams: Exploring the Use of Cognitive Models of Trust and Attention

2010-53 Edgar Meij (UVA)
Combining Concepts and Language Models for Information Access

2011

2011-01 Botond Cseke (RUN)
Variational Algorithms for Bayesian Inference in Latent Gaussian Models

2011-02 Nick Tinnemeier(UU)
Organizing Agent Organizations. Syntax and Operational Semantics of an Organization-Oriented Programming Language

2011-03 Jan Martijn van der Werf (TUE)
Compositional Design and Verification of Component-Based Information Systems

2011-04 Hado van Hasselt (UU)
Insights in Reinforcement Learning; Formal analysis and empirical evaluation of temporal-difference learning algorithms

2011-05 Base van der Raadt (VU)
Enterprise Architecture Coming of Age - Increasing the Performance of an Emerging Discipline.

2011-06 Yiwen Wang (TUE)
Semantically-Enhanced Recommendations in Cultural Heritage

2011-07 Yujia Cao (UT)
Multimodal Information Presentation for High Load Human Computer Interaction

2011-08 Nieske Vergunst (UU)
BDI-based Generation of Robust Task-Oriented Dialogues

2011-09 Tim de Jong (OU)
Contextualised Mobile Media for Learning

2011-10 Bart Bogaert (UvT)
Cloud Content Contention

2011-11 Dhaval Vyas (UT)
Designing for Awareness: An Experience-focused HCI Perspective

2011-12 Carmen Bratosin (TUE)
Grid Architecture for Distributed Process Mining

2011-13 Xiaoyu Mao (UvT)
Airport under Control. Multiagent Scheduling for Airport Ground Handling

2011-14 Milan Lovric (EUR)
Behavioral Finance and Agent-Based Artificial Markets

2011-15 Marijn Koolen (UvA)
The Meaning of Structure: the Value of Link Evidence for Information Retrieval

2011-16 Maarten Schadd (UM)
Selective Search in Games of Different Complexity

2011-17 Jiyin He (UVA)
Exploring Topic Structure: Coherence, Diversity and Relatedness

2011-18 Mark Ponsen (UM)
Strategic Decision-Making in complex games

2011-19 Ellen Rusman (OU)
The Mind's Eye on Personal Profiles

2011-20 Qing Gu (VU)
Guiding service-oriented software engineering - A view-based approach

2011-21 Linda Terlouw (TUD)
Modularization and Specification of Service-Oriented Systems

2011-22 Junte Zhang (UVA)
System Evaluation of Archival Description and Access

2011-23 Wouter Weerkamp (UVA)
Finding People and their Utterances in Social Media

2011-24 Herwin van Welbergen (UT)
Behavior Generation for Interpersonal Coordination with Virtual Humans On Specifying, Scheduling and Realizing Multimodal Virtual Human Behavior

2011-25 Syed Waqar ul Qounain Jaffry (VU)
Analysis and Validation of Models for Trust Dynamics

2011-26 Matthijs Aart Pontier (VU)
Virtual Agents for Human Communication – Emotion Regulation and Involvement-Distance Trade-Offs in Embodied Conversational Agents and Robots

2011-27 Aniel Bhulai (VU)
Dynamic website optimization through autonomous management of design patterns

2011-28 Rianne Kaptein(UVA)
Effective Focused Retrieval by Exploiting Query Context and Document Structure

2011-29 Faisal Kamiran (TUE)
Discrimination-aware Classification

2011-30 Egon van den Broek (UT)
Affective Signal Processing (ASP): Unraveling the mystery of emotions

2011-31 Ludo Waltman (EUR)
Computational and Game-Theoretic Approaches for Modeling Bounded Rationality

2011-32 Nees-Jan van Eck (EUR)
Methodological Advances in Bibliometric Mapping of Science

2011-33 Tom van der Weide (UU)
Arguing to Motivate Decisions

2011-34 Paolo Turrini (UU)
Strategic Reasoning in Interdependence: Logical and Game-theoretical Investigations

2011-35 Maaike Harbers (UU)
Explaining Agent Behavior in Virtual Training

2011-36 Erik van der Spek (UU)
Experiments in serious game design: a cognitive approach

2011-37 Adriana Burlutiu (RUN)
Machine Learning for Pairwise Data, Applications for Preference Learning and Supervised Network Inference

2011-38 Nyree Lemmens (UM)
Bee-inspired Distributed Optimization

2011-39 Joost Westra (UU)
Organizing Adaptation using Agents in Serious Games

2011-40 Viktor Clerc (VU)
Architectural Knowledge Management in Global Software Development

2011-41 Luan Ibraimi (UT)
Cryptographically Enforced Distributed Data Access Control

2011-42 Michal Sindlar (UU)
Explaining Behavior through Mental State Attribution

2011-43 Henk van der Schuur (UU)
Process Improvement through Software Operation Knowledge

2011-44 Boris Reuderink (UT)
Robust Brain-Computer Interfaces

2011-45 Herman Stehouwer (UvT)
Statistical Language Models for Alternative Sequence Selection

2011-46 Beibei Hu (TUD)
Towards Contextualized Information Delivery: A Rule-based Architecture for the Domain of Mobile Police Work

2011-47 Azizi Bin Ab Aziz (VU)
Exploring Computational Models for Intelligent Support of Persons with Depression

2011-48 Mark Ter Maat (UT)
Response Selection and Turn-taking for a Sensitive Artificial Listening Agent

2011-49 Andreea Niculescu (UT)
Conversational interfaces for task-oriented spoken dialogues: design aspects influencing interaction quality

2012

2012-01 Terry Kakeeto (UvT)
Relationship Marketing for SMEs in Uganda

2012-02 Muhammad Umair(VU)
Adaptivity, emotion, and Rationality in Human and Ambient Agent Models

2012-03 Adam Vanya (VU)
Supporting Architecture Evolution by Mining Software Repositories

2012-04 Jurriaan Souer (UU)
Development of Content Management System-based Web Applications

2012-05 Marijn Plomp (UU)
Maturing Interorganisational Information Systems

2012-06 Wolfgang Reinhardt (OU)
Awareness Support for Knowledge Workers in Research Networks

2012-07 Rianne van Lambalgen (VU)
When the Going Gets Tough: Exploring Agent-based Models of Human Performance under Demanding Conditions

2012-08 Gerben de Vries (UVA)
Kernel Methods for Vessel Trajectories

2012-09 Ricardo Neisse (UT)
Trust and Privacy Management Support for Context-Aware Service Platforms

2012-10 David Smits (TUE)
Towards a Generic Distributed Adaptive Hypermedia Environment

2012-11 J.C.B. Rantham Prabhakara (TUE)
Process Mining in the Large: Preprocessing, Discovery, and Diagnostics

2012-12 Kees van der Sluijs (TUE)
Model Driven Design and Data Integration in Semantic Web Information Systems

2012-13 Suleman Shahid (UvT)
Fun and Face: Exploring non-verbal expressions of emotion during playful interactions

2012-14 Evgeny Knutov(TUE)
Generic Adaptation Framework for Unifying Adaptive Web-based Systems

2012-15 Natalie van der Wal (VU)
Social Agents. Agent-Based Modelling of Integrated Internal and Social Dynamics of Cognitive and Affective Processes.

2012-16 Fiemke Both (VU)
Helping people by understanding them - Ambient Agents supporting task execution and depression treatment

2012-17 Amal Elgammal (UvT)
Towards a Comprehensive Framework for Business Process Compliance

2012-18 Eltjo Poort (VU)
Improving Solution Architecting Practices

2012-19 Helen Schonenberg (TUE)
What's Next? Operational Support for Business Process Execution

2012-20 Ali Bahramisharif (RUN)
Covert Visual Spatial Attention, a Robust Paradigm for Brain-Computer Interfacing

2012-21 Roberto Cornacchia (TUD)
Querying Sparse Matrices for Information Retrieval

2012-22 Thijs Vis (UvT)
Intelligence, politie en veiligheidsdienst: verenigbare grootheden?

2012-23 Christian Muehl (UT)
Toward Affective Brain-Computer Interfaces: Exploring the Neurophysiology of Affect during Human Media Interaction

2012-24 Laurens van der Werff (UT)
Evaluation of Noisy Transcripts for Spoken Document Retrieval

2012-25 Silja Eckartz (UT)
Managing the Business Case Development in Inter-Organizational IT Projects: A Methodology and its Application

2012-26 Emile de Maat (UVA)
Making Sense of Legal Text

2012-27 Hayrettin Gurkok (UT)
Mind the Sheep! User Experience Evaluation & Brain-Computer Interface Games

2012-28 Nancy Pascall (UvT)
Engendering Technology Empowering Women

2012-29 Almer Tigelaar (UT)
Peer-to-Peer Information Retrieval

2012-30 Alina Pommeranz (TUD)
Designing Human-Centered Systems for Reflective Decision Making

2012-31 Emily Bagarukayo (RUN)
A Learning by Construction Approach for Higher Order Cognitive Skills Improvement, Building Capacity and Infrastructure

2012-32 Wietske Visser (TUD)
Qualitative multi-criteria preference representation and reasoning

2012-33 Rory Sie (OUN)
Coalitions in Cooperation Networks (COCOON)

2012-34 Pavol Jancura (RUN)
Evolutionary analysis in PPI networks and applications

2012-35 Evert Haasdijk (VU)
Never Too Old To Learn – On-line Evolution of Controllers in Swarm- and Modular Robotics

2012-36 Denis Ssebugwawo (RUN)
Analysis and Evaluation of Collaborative Modeling Processes

2012-37 Agnes Nakakawa (RUN)
A Collaboration Process for Enterprise Architecture Creation

2012-38 Selmar Smit (VU)
Parameter Tuning and Scientific Testing in Evolutionary Algorithms

2012-39 Hassan Fatemi (UT)
Risk-aware design of value and coordination networks

2012-40 Agus Gunawan (UvT)
Information Access for SMEs in Indonesia

2012-41 Sebastian Kelle (OU)
Game Design Patterns for Learning

2012-42 Dominique Verpoorten (OU)
Reflection Amplifiers in self-regulated Learning

2012-43 Withdrawn

2012-44 Anna Tordai (VU)
On Combining Alignment Techniques

2012-45 Benedikt Kratz (UvT)
A Model and Language for Business-aware Transactions

2012-46 Simon Carter (UVA)
Exploration and Exploitation of Multilingual Data for Statistical Machine Translation

2012-47 Manos Tsagkias (UVA)
Mining Social Media: Tracking Content and Predicting Behavior

2012-48 Jorn Bakker (TUE)
Handling Abrupt Changes in Evolving Time-series Data

2012-49 Michael Kaisers (UM)
Learning against Learning - Evolutionary dynamics of reinforcement learning algorithms in strategic interactions

2012-50 Steven van Kervel (TUD)
Ontologogy driven Enterprise Information Systems Engineering

2012-51 Jeroen de Jong (TUD)
Heuristics in Dynamic Sceduling; a practical framework with a case study in elevator dispatching

2013

2013-01 Viorel Milea (EUR)
News Analytics for Financial Decision Support

2013-02 Erietta Liarou (CWI)
MonetDB/DataCell: Leveraging the Column-store Database Technology for Efficient and Scalable Stream Processing

2013-03 Szymon Klarman (VU)
Reasoning with Contexts in Description Logics

2013-04 Chetan Yadati(TUD)
Coordinating autonomous planning and scheduling

2013-05 Dulce Pumareja (UT)
Groupware Requirements Evolutions Patterns

2013-06 Romulo Goncalves(CWI)
The Data Cyclotron: Juggling Data and Queries for a Data Warehouse Audience

2013-07 Giel van Lankveld (UvT)
Quantifying Individual Player Differences

2013-08 Robbert-Jan Merk(VU)
Making enemies: cognitive modeling for opponent agents in fighter pilot simulators

2013-09 Fabio Gori (RUN)
Metagenomic Data Analysis: Computational Methods and Applications

2013-10 Jeewanie Jayasinghe Arachchige(UvT)
A Unified Modeling Framework for Service Design.

2013-11 Evangelos Pournaras(TUD)
Multi-level Reconfigurable Self-organization in Overlay Services

2013-12 Marian Razavian(VU)
Knowledge-driven Migration to Services

2013-13 Mohammad Safiri(UT)
Service Tailoring: User-centric creation of integrated IT-based homecare services to support independent living of elderly

2013-14 Jafar Tanha (UVA)
Ensemble Approaches to Semi-Supervised Learning Learning

2013-15 Daniel Hennes (UM)
Multiagent Learning - Dynamic Games and Applications

2013-16 Eric Kok (UU)
Exploring the practical benefits of argumentation in multi-agent deliberation

2013-17 Koen Kok (VU)
The PowerMatcher: Smart Coordination for the Smart Electricity Grid

2013-18 Jeroen Janssens (UvT)
Outlier Selection and One-Class Classification

2013-19 Renze Steenhuizen (TUD)
Coordinated Multi-Agent Planning and Scheduling

2013-20 Katja Hofmann (UvA)
Fast and Reliable Online Learning to Rank for Information Retrieval

2013-21 Sander Wubben (UvT)
Text-to-text generation by monolingual machine translation

2013-22 Tom Claassen (RUN)
Causal Discovery and Logic

2013-23 Patricio de Alencar Silva(UvT)
Value Activity Monitoring

2013-24 Haitham Bou Ammar (UM)
Automated Transfer in Reinforcement Learning

2013-25 Agnieszka Anna Latoszek-Berendsen (UM)
Intention-based Decision Support. A new way of representing and implementing clinical guidelines in a Decision Support System

2013-26 Alireza Zarghami (UT)
Architectural Support for Dynamic Homecare Service Provisioning

2013-27 Mohammad Huq (UT)
Inference-based Framework Managing Data Provenance

2013-28 Frans van der Sluis (UT)
When Complexity becomes Interesting: An Inquiry into the Information eXperience

2013-29 Iwan de Kok (UT)
Listening Heads

2013-30 Joyce Nakatumba (TUE)
Resource-Aware Business Process Management: Analysis and Support

2013-31 Dinh Khoa Nguyen (UvT)
Blueprint Model and Language for Engineering Cloud Applications

2013-32 Kamakshi Rajagopal (OUN)
Networking For Learning; The role of Networking in a Lifelong Learner's Professional Development

2013-33 Qi Gao (TUD)
User Modeling and Personalization in the Microblogging Sphere

2013-34 Kien Tjin-Kam-Jet (UT)
Distributed Deep Web Search

2013-35 Abdallah El Ali (UvA)
Minimal Mobile Human Computer Interaction

2013-36 Than Lam Hoang (TUe)
Pattern Mining in Data Streams

2013-37 Dirk Börner (OUN)
Ambient Learning Displays

2013-38 Eelco den Heijer (VU)
Autonomous Evolutionary Art

2013-39 Joop de Jong (TUD)
A Method for Enterprise Ontology based Design of Enterprise Information Systems

2013-40 Pim Nijssen (UM)
Monte-Carlo Tree Search for Multi-Player Games

2013-41 Jochem Liem (UVA)
Supporting the Conceptual Modelling of Dynamic Systems: A Knowledge Engineering Perspective on Qualitative Reasoning

2013-42 Léon Planken (TUD)
Algorithms for Simple Temporal Reasoning

2013-43 Marc Bron (UVA)
Exploration and Contextualization through Interaction and Concepts

2014

2014-01 Nicola Barile (UU)
Studies in Learning Monotone Models from Data

2014-02 Fiona Tuliyano (RUN)
Combining System Dynamics with a Domain Modeling Method

2014-03 Sergio Raul Duarte Torres (UT)
Information Retrieval for Children: Search Behavior and Solutions

2014-04 Hanna Jochmann-Mannak (UT)
Websites for children: search strategies and interface design - Three studies on children's search performance and evaluation

2014-05 Jurriaan van Reijsen (UU)
Knowledge Perspectives on Advancing Dynamic Capability

2014-06 Damian Tamburri (VU)
Supporting Networked Software Development

2014-07 Arya Adriansyah (TUE)
Aligning Observed and Modeled Behavior

2014-08 Samur Araujo (TUD)
Data Integration over Distributed and Heterogeneous Data Endpoints

2014-09 Philip Jackson (UvT)
Toward Human-Level Artificial Intelligence: Representation and Computation of Meaning in Natural Language

2014-10 Ivan Salvador Razo Zapata (VU)
Service Value Networks

2014-11 Janneke van der Zwaan (TUD)
An Empathic Virtual Buddy for Social Support

2014-12 Willem van Willigen (VU)
Look Ma, No Hands: Aspects of Autonomous Vehicle Control

2014-13 Arlette van Wissen (VU)
Agent-Based Support for Behavior Change: Models and Applications in Health and Safety Domains

2014-14 Yangyang Shi (TUD)
Language Models With Meta-information

2014-15 Natalya Mogles (VU)
Agent-Based Analysis and Support of Human Functioning in Complex Socio-Technical Systems: Applications in Safety and Healthcare

2014-16 Krystyna Milian (VU)
Supporting trial recruitment and design by automatically interpreting eligibility criteria

2014-17 Kathrin Dentler (VU)
Computing healthcare quality indicators automatically: Secondary Use of Patient Data and Semantic Interoperability

2014-18 Mattijs Ghijsen (VU)
Methods and Models for the Design and Study of Dynamic Agent Organizations

2014-19 Vincius Ramos (TUE)
Adaptive Hypermedia Courses: Qualitative and Quantitative Evaluation and Tool Support

2014-20 Mena Habib (UT)
Named Entity Extraction and Disambiguation for Informal Text: The Missing Link

2014-21 Kassidy Clark (TUD)
Negotiation and Monitoring in Open Environments

2014-22 Marieke Peeters (UU)
Personalized Educational Games - Developing agent-supported scenario-based training

2014-23 Eleftherios Sidirourgos (UvA/CWI)
Space Efficient Indexes for the Big Data Era

2014-24 Davide Ceolin (VU)
Trusting Semi-structured Web Data

2014-25 Martijn Lappenschaar (RUN)
New network models for the analysis of disease interaction

2014-26 Tim Baarslag (TUD)
What to Bid and When to Stop

2014-27 Rui Jorge Almeida (EUR)
Conditional Density Models Integrating Fuzzy and Probabilistic Representations of Uncertainty

2014-28 Anna Chmielowiec (VU)
Decentralized k-Clique Matching

2014-29 Jaap Kabbedijk (UU)
Variability in Multi-Tenant Enterprise Software

2014-30 Peter de Kock Berenschot (UvT)
Anticipating Criminal Behaviour

2014-31 Leo van Moergestel (UU)
Agent Technology in Agile Multiparallel Manufacturing and Product Support

2014-32 Naser Ayat (UVA)
On Entity Resolution in Probabilistic Data

2014-33 Tesfa Tegegne Asfaw (RUN)
Service Discovery in eHealth

2014-34 Christina Manteli (VU)
The Effect of Governance in Global Software Development: Analyzing Transactive Memory Systems

2014-35 Joost van Oijen (UU)
Cognitive Agents in Virtual Worlds: A Middleware Design Approach

2014-36 Joos Buijs (TUE)
Flexible Evolutionary Algorithms for Mining Structured Process Models

2014-37 Maral Dadvar (UT)
Experts and Machines United Against Cyberbullying

2014-38 Danny Plass-Oude Bos (UT)
Making brain-computer interfaces better: improving usability through post-processing

2014-39 Jasmina Maric (UvT)
Web Communities, Immigration and Social Capital

2014-40 Walter Oboma (RUN)
A Framework for Knowledge Management Using ICT in Higher Education

2014-41 Frederik Hogenboom (EUR)
Automated Detection of Financial Events in News Text

2014-42 Carsten Eickhoff (CWI/TUD)
Contextual Multidimensional Relevance Models

2014-43 Kevin Vlaanderen (UU)
Supporting Process Improvement using Method Increments

2014-44 Paulien Meesters (UvT)
Intelligent Blauw. Intelligence-gestuurde politiezorg in gebiedsgebonden eenheden

2014-45 Birgit Schmitz (OU)
Mobile Games for Learning: A Pattern-Based Approach

2014-46 Ke Tao (TUD)
Social Web Data Analytics: Relevance, Redundancy, Diversity

2014-47 Shangsong Liang (UVA)
Fusion and Diversification in Information Retrieval

2015

2015-01 Niels Netten (UVA)
Machine Learning for Relevance of Information in Crisis Response

2015-02 Faiza Bukhsh (UvT)
Smart auditing: Innovative Compliance Checking in Customs Controls

2015-03 Twan van Laarhoven (RUN)
Machine learning for network data

2015-04 ??? (OU)
???

2015-05 Christoph Bösch (UT)
Cryptographically Enforced Search Pattern Hiding