

Master Thesis
Secure Data Storage Outsourcing with Conjunctive Keyword Search
Author: A.H.P. (Arjan) van Vliet 1286544
Exam committee Prof. Dr. Ir. R.L. Lagendijk Dr. Ir. J.C.A. van der Lubbe Dr. P. Cimiano Ir. P. Kornelisse RE CISA ...
Delft, University of Technology Department of Mediamatics Faculty of Electrical Engineering, Mathematics and Computer Science (EEMCS) Information and Communication Theory Group

Abstract

This thesis in the field of cryptography considers secure data outsourcing with keyword search capabilities. To ensure data confidentiality the data is stored in encrypted format. An efficient solution has been developed to enable conjunctive keyword search. Also, a protocol is applied to enable efficient and secure sharing of encrypted data. It is possible to efficiently enrol and revoke users for both the searching and the decryption part of the system. The newly developed search technique is implemented as well to test its practical performance.

Preface

This thesis is written to graduate as a Master of Science in Media and Knowledge Engineering at the TU Delft. The thesis is written in the field of cryptography which is a topic of the Information and Communication Technology group at the Electrical Engineering, Mathematics and Computer Science faculty. All work has been done in the Information Security and Control business unit at KPMG IT Advisory from July 2008 till March 2009.

Preceding this thesis a literature report was written about Data Leakage and its Prevention. It showed the importance of data confidentiality. In this report a short introduction to relevant research topics was given as well. It discussed some examples, found in literature, of private data storage to ensure data confidentiality while offering the possibility to search on encrypted data. Other literature described an approach for secure multi user data storage. By combining both research directions and the fact that server management is an expensive task a thesis research topic was found. The topic of this thesis is secure data storage outsourcing with the possibility to perform conjunctive keywords searches.

Acknowledgements

First of all I would like to thank my parents, Henk and Ria, for their endless support and keeping me on the track at times I lacked motivation. I would like to thank my girlfriend Maartje for always having a listening ear at times things did not go the way I wanted and her useful comments on my English language.

I would like to thank all people at the Information Security and Control (ISC) group at KPMG IT Advisory. In the first place I would like to thank John and Bram, who were the first of the group I had contact with, for providing me the opportunity to graduate in the ISC group. I would like to thank all my supervisors at KPMG for making time in their busy schedules. Thanks to Hans for supervising my literature research, finding my way within KPMG and pointing me to the available resources. Many thanks go to Erwin who always had time for me, reread my work several times and gave many useful tips to improve the structure which really pushed this thesis to the next level. I would also like to thank Peter for being the second supervisor for both my literature research and my thesis, making time for me in his very busy schedule and his tough questions that made me rethink. Finally, I would like to thank all other employees at ISC for their nice discussions, friendliness and good atmosphere to graduate.

I would also like to thank everyone at the TU Delft that contributed to this thesis. First of all, my supervisor Jan for his critical questions making me look at things differently and providing me the freedom to give research the direction I liked most. Secondly, I would like all members from the crypto reading group for the many good and useful discussions we had. Thanks to Thijs for supervising the reading groups. I would like to thank Zeki for his discussions on early versions of my algorithm at the time I felt stuck. It is because of him that I decided to implement my solution which gave me new energy. I would like to thank Ilyaz for reviewing an early version of this thesis and giving my valuable feedback. Thanks to Michael for his clear explanations during the reading group sessions and thanks to Katherina. Finally, I would like all members of the students meetings and Emile for supervising these meetings.

Table of Contents

Preface	i
Acknowledgements	i
1. Introduction	1
1.1. Motivation	1
1.2. Research Goal	2
1.3. Research Methodology	3
1.4. Report Outline	3
2. Existing Solutions and Related Work	5
2.1. Keyword Searchable Encryption	5
2.1.1. Overview of Keyword Searchable Encryption Schemes	6
2.1.2. Range Queries and Other Secure Search Techniques	7
2.2. Proxy Re-Encryption	8
2.3. Summary	9
3. Setting the Scene	11
3.1. Scene Description	11
3.2. Assumptions on Functionality and the Actors	13
3.3. Requirements	15
3.4. Summary	16
4. High Level System Construction	17
4.1. Search Technique	17
4.2. Keyword Index Construction and Storage	18
4.2.1. One Index for All Documents	19
4.2.2. A Separate Index for Every Document	20
4.3. Query Construction and Matching	20
4.4. Multi User Data Encryption	22
4.5. Authentication and Authorisation	23
4.5.1. Query and Index Authentication and Authorisation	24
4.5.2. Decryption Authentication and Authorisation	24
4.6. Putting Together	25
4.6.1. DSSP Data and Key Storage Representation	25
4.6.2. System Overview	26
4.7. Summary	27
5. Preliminaries	29
5.1. Cryptographic Preliminaries	29
5.1.1. Bilinear Map (Pairing)	29
5.1.2. Proxy Re-encryption	31
5.2. System Definition	32
5.3. Security Definitions	33
5.3.1. Security Definition	34
5.3.2. Revocability	35
5.4. Summary	35
6. Constructing a Searchable Encryption Scheme	37
6.1. Attempt 1 – A Keyword Searchable Scheme	37

6.1.1.	Evaluation	40
6.2.	Attempt 2 – A Conjunctive Keyword Searchable Scheme	41
6.2.1.	Security Proof	45
6.2.2.	Evaluation	46
6.3.	Extensions	46
6.3.1.	Role Based Access to Subgroups of Documents	47
6.3.2.	Dynamic Dictionary	47
6.4.	Threat Analysis and Measures	48
6.4.1.	Threat Analysis	48
6.4.2.	Additional Security Measures	49
6.5.	Summary	50
7.	Implementation	51
7.1.	Programming Details and Cryptographic Libraries	51
7.2.	Pairing Parameters	52
7.3.	Implementation Overview	52
7.4.	Experimental Results	56
7.5.	Pointers to Further Improvements and Optimisation	61
7.6.	Summary	61
8.	Conclusion and Future Work	63
8.1.	Scientific Contribution	63
8.2.	Other Applications of Searchable Encryption	64
8.3.	Conclusion	65
8.4.	Future work	66
	Bibliography	69
	Appendix A – Terms, Symbols and Notations	71

Table of Figures

Figure 1:	Services, actions and actors overview	12
Figure 2:	Actor overview in an outsourced data storage setting	13
Figure 3:	One index for all documents	19
Figure 4:	A separate index for every document	20
Figure 5:	Documents with their lockboxes	23
Figure 6:	Query authorisation	24
Figure 7:	Decryption authorization	25
Figure 8:	Data representation on the data server	26
Figure 9:	From query submission to decryption (schematic overview)	27
Figure 10:	Miller's Algorithm	30
Figure 11:	System definition	33
Figure 12:	Attempt 1 - A keyword searchable encryption scheme	39
Figure 13:	Attempt 1 - Query submission, searching and document decryption	40
Figure 14:	Attempt 2 – A conjunctive keyword searchable encryption scheme	44
Figure 15:	Attempt 2 - Query submission, searching and decryption	44
Figure 16:	Document group storage	47
Figure 17:	Loading a parameter file and initialising a pairing	52
Figure 18:	System initialisation	53
Figure 19:	Index computation function call	53
Figure 20:	Index generation	54

Figure 21: Authentication and authorisation and document storage	54
Figure 22: Authentication and authorisation	55
Figure 23: Query execution function call	55
Figure 24: The search process	56
Figure 25: Query performance counter	57
Figure 26: Test program	57
Figure 27: Search speed	59
Figure 28: Output correctness test	60

Table of Tables

Table 1: Comparison of searchable encryption schemes	7
Table 2: Threat analysis	49
Table 3: Index generation duration	58
Table 4: Query generation duration	58
Table 5: Search speed with two pre-processed elements	59

1.

Introduction

In this report a realistic scenario is defined in which a company outsources its data storage management. An important aspect for data storage outsourcing to become successful is to ensure data confidentiality. A preceding literature study showed a growing number of data leakage incidents [32] and stated that consequences when private and other sensitive information becomes public can be severe in terms of both corporate image and financial costs. Data leakage incidents are a result of the increased amount of digitally stored information and people not constantly being aware about the data's value. This digital information is often more easy to find, access and exchange and inevitably includes sensitive information like credit card numbers, financial information or technical designs. Since the consequences of a data leakage incident can be enormous it is no wonder that companies are taking measures and purchase Data Leakage Prevention (DLP) products to ensure confidentiality of their data.

To ensure data confidentiality when data storage is outsourced, data should be stored in an encrypted format since the storage provider can not be fully trusted. Like most security measures this has an influence on the usability and functionality. When data is encrypted it is no longer possible to search and selectively retrieve documents. Most encryption schemes have the scenario where information is encrypted in such a way that only one intended recipient can recover the original content, unless they share a common secret. In practical situations often many employees should have access to the data and should be able to perform searches, preferably without sharing a common secret.

This report gives an analysis of existing solutions to enable search operations on encrypted data and to share encrypted data among a group of existing users. A new, improved, system is proposed which offers both keyword search capabilities and the ability to share encrypted data with all authorised users at the cost of a reasonable overhead. This will help companies to save costs on data storage management due to location and scale advantages of the data storage company. To test the practical performance the newly developed search algorithm is actually implemented and several tests are performed.

1.1. Motivation

There are several reasons to write this report related to IT and security. Both IT and security are of personal interest. The increased amount of data leakage incidents including personal information, as shown in the preceding literature study [32], which can have enormous impact are a motivation to a

subject related to data protection as well. Data leakage incidents often involve social security numbers, credit card numbers and other personal identifiable information. This information can be used to commit identity fraud where people try to gain personal advantage with somebody else his credits or achievements.

Another reason is to reduce the high IT costs in organisations. Nowadays there are almost no companies that do not rely on IT solutions for their day to day business activities. In a lot of those companies every employee has a computer or laptop for their daily work activities. All those employees are using many business applications and IT services like e-mail, intranet and network or remote storage. An increased amount of digital data leads to more storage space and as a result more data storage servers. These applications and services cannot run if the infrastructure and servers running in the background are not functioning properly. The servers have to be kept in optimal conditions, by protecting them against existing and newly rising computer security threats and by regularly upgrading them with increased demands due to more users, or new more demanding software. Other server management tasks are related to system and business continuity. In case of a power failure additional power supplies have to be in place. Also, regularly creating system backups is an important task related to server management. Outsourcing of server management operations can help to reduce high IT costs.

1.2. Research Goal

As already mentioned at the beginning of the introduction and in the motivation, a practical and secure system which makes it possible to securely outsource data storage management with search capabilities has to be developed. On the one hand it will help to reduce the high IT costs and on the other hand it will add to a higher level of data confidentiality.

Recent literature has already described several solutions to search on encrypted data. The problem with most of these schemes is that they focus on a single user setting, the scenario where one user has its private data storage. In this report a multi user scenario is considered, where multiple users can access and search the same data storage. Some of these solutions, however, describe a multi user extension to their scheme, but since they are not intended as a true multi user system they face limitations in a multi user setting. Other literature describes a multi user encrypted file storage system to securely outsource data storage, but has no search capabilities.

The main goal of this MSc thesis assignment is to develop an extension to the multi user secure data storage system as described above that adds keyword search capabilities. Part of this goal is to develop an improvement or optimisation of the current searchable encryption schemes. Improvement can be additional functionality, a higher level of security or a more efficient implementation in terms of storage space and computation.

To support the main goal a set of sub goals have been posed. Three sub goals have been defined covering the system's security and confidentiality, the system's manageability, the outsourcing aspect of the system and the system's efficiency for practical applications. The sub goals are listed below:

1. Since confidentiality is important the data is stored in encrypted format and the amount of information the server is able to learn has to be minimized.
2. It should be possible to enrol and revoke users from the system. All authorised users should be able to search for keywords, add documents and be able to decrypt documents.
3. Since IT service management is an expensive operation as much tasks as securely and conveniently possible have to be outsourced.
4. For a system to succeed in practical applications its efficiency is important and for this reason the system has to be as efficient as possible.

1.3. Research Methodology

Research for this thesis has started with gathering relevant scientific literature. In order to find relevant literature the references in interesting literature was used to find other relevant literature as well. After reading scientific literature, the scene setting introduced in this introduction was defined and elaborated on. Also, several assumptions were posed on the system as a whole and the different parties involved. The assumptions were made to make it feasible to construct a satisfactory system.

From the insights gained while reading scientific literature a high level design of the system was developed. During the design, different alternatives found in literature or thought of were analysed and discussed based on the three sub goals.

After finishing the high level design the required preliminaries and definitions were introduced for the detailed system and algorithm construction. At first a more basic system was developed not completely satisfying the research goals. The second construction does satisfy the research goals and its security is proven. Also, some extensions to the system were developed.

Finally, the keyword search part of the system was implemented to test its practical performance. Search engines were used to find cryptographic libraries supporting the necessary mathematical operations and algorithms. For the evaluation several tests were performed to evaluate its computational speed.

1.4. Report Outline

At first some scientific literature is discussed about encryption methods that support keyword searches and proxy re-encryption mechanisms to support multi user data encryption. An overview of the scientific literature which is read to find out what is already done and to find useful insights for the development of the system in this work is given in the next chapter. A comparison is made between the existing schemes to get an idea in which areas the system developed in this work can improve and to evaluate the final system. From chapter three and onwards most chapters cover own work descriptions.

In chapter three the research goal is further elaborated upon and a scene description of a practical situation is given. This section also describes the assumptions posed on the different parties involved and on the system as a whole. Chapter four describes the high level design for the system that meets the scene description as given in chapter three. In this chapter the different design options are mentioned, analysed and explained.

Chapter five starts with a description of the required preliminaries from literature needed to understand the system's construction in chapter six. Hereafter a formal definition of the system constructed for this thesis is given. Chapter five also gives a formal security definition based on the ones found in literature. Once the preliminaries are described, detailed constructions of the developed system are given in chapter six. The technical descriptions given in this chapter are analysed to see if they fulfil the goals posed earlier in the report. Also some extensions to the system, including one to support role based access to documents, are given.

The implementation of the search algorithm is discussed in chapter seven. It gives an overview of the implementation details and discusses a number of results from the implementation tests. Also, it gives some pointers to further improve the computational speed obtained from the tests. Finally, this thesis ends with a conclusion which is given in chapter eight. It also mentions the scientific contribution, some other applications of searchable encryption and future work.

In the descriptions throughout this thesis report a lot of terminology and symbols are used. For an overview of the used terms, symbols and notations used in this thesis the reader is referred to appendix A. The overview given in appendix A can be flipped out such that the reader can view the appendix while reading the report.

2.

Existing Solutions and Related Work

In this chapter an overview is given of the scientific literature that is studied to find existing solutions that already solve parts of the goals defined in the introduction. The solution for the system consists of two parts. First, a construction that offers keyword search capabilities to all authorised users and secondly a construction that enables sharing of encrypted data between all authorised users.

The first section of this chapter gives an overview of the different searchable encryption schemes that can be found in literature. Once the schemes are shortly described and analysed, a comparison is made which enables comparing the scheme developed in this work with existing solutions. Also, some papers related to keyword searchable encryption are described.

In section three an overview is given of different proxy re-encryption schemes. It describes which properties are useful for the system constructed in this work and which properties are not. The best solution is chosen.

2.1. Keyword Searchable Encryption

The term searchable encryption might be a bit confusing. One might expect that the actual ciphertexts are searched, but this is not the case. Instead, separate indexes are encrypted in a special way to make searching possible. Usually a standard encryption scheme is used in combination with special index generation and query mechanisms. A system in which every file has one or more keywords assigned to it which can be matched in a special encrypted way is called a searchable encryption scheme.

Searchable encryption has numerous applications and therefore different solutions exist. A distinction can be made between simple or conjunctive keyword searchable encryption and range queries on encrypted data. Keyword searchable encryption is mostly demonstrated in a secure file storage system or in a secure database setting. Range queries on encrypted data are mostly used to test a certain predicate. For example, if the data has a value in a specific range such as if the IP address falls within a certain range in network audit logs.

First an overview of different keyword searchable encryption schemes is given. The most important details are summarised in a table to compare the different solutions and to evaluate the scheme constructed in this report. The final section gives an overview of some range query techniques studied

to get an idea of their approaches and to see if it can be valuable in keyword searchable encryption schemes.

2.1.1. Overview of Keyword Searchable Encryption Schemes

Keyword searchable encryption schemes have been proposed in both the symmetric and asymmetric setting. Most keyword searchable encryption schemes focus on the scenario where a single user has private data storage and wants to perform keyword searches to selectively retrieve his or her data. A single user approach has limited practical applications since these mostly require multi user solutions. Some of the newer schemes describe multi user extensions, but are often faced with limited functionality. A limitation is for example, that only one user can add documents. User management is often problematic as well. Enrolling or revoking users often requires redistribution of keys.

The notion of searchable encryption was introduced by Song, Wagner and Perrig [30] who have proposed a solution in the symmetric setting. Every document is seen as a stream of words and every word is masked with a stream cipher. Only the user which knows the seed can reproduce the stream cipher and perform search operations. Their solution is different from others since it is the only one that encrypts all words instead of using separately constructed keyword indexes.

Most index construction schemes are in the symmetric setting [2, 3, 9, 11, 28]. Chang and Mitzenmacher [9] proposed a scheme where only one keyword can be assigned to each document. Besides this limitation their security definition is less stringent compared to others. It only guarantees non adaptive security and thus does not guarantee security when previous query results are taken into account constructing new queries. The security definition used by Ballard et al [2] does not require secure trapdoors or queries. Curtmolla et al [11] have introduced the general adaptive security definitions for searchable encryption. Their scheme is also the only one that has one global index for all documents.

Asymmetric solutions have been proposed by Byun et al [7] and Hwang et al [18], but are less efficient than symmetric solutions. Both schemes support conjunctive keyword search, but only Hwang et al described a multi user extension. In order to support multiple users, additional information for each individual user is attached to the indexes. This approach is not very practical since all indexes have to be changed upon revocation or enrolment.

Only one true multi user keyword searchable encryption scheme is found in literature. In the system constructed by F. Bao et al [3] all users can search, add and decrypt documents. Efficient enrolment and revocation is possible without the need of redistributing keys or re-computing indexes. It is, however, not possible to revoke decryption rights due to the use of one symmetric key to encrypt all documents which is shared by all users. Also, only one keyword can be attached to every document. This approach thus lacks functionality for practical applications.

The table below gives an overview and a comparison of the different symmetric and asymmetric keyword searchable encryption schemes. Since the solution from Song, Wagner and Perrig does not use indexes it is not included in the table.

	Chang et al [9]	Ballard et al [2]*	Curtmolla et al [11]	Byun et al [7]	Hwang et al [18]	Ryu et al [28]	Bao et al [3]
Index Size	$O(w)$	$O(n(w+1))$	$O(w)$	$O(n(w+2))$	$O(n(wu+1))$	$O(n(w+2))$	$O(2n)$
Query Size	$O(2)$	$O(2)$	$O(w)$	$O(3)$	$O(n+3)$	$O(2)$	$O(1)$
Search Complexity	$O(n)$	$O(n)$ $(2n)$ pr	$O(w)$ 0 pr	$O(n)$ $(2n+1)$ pr	$O(n)$ $(3n)$ pr	$O(n)$	$O(n)$

	Chang et al [9]	Ballard et al [2]*	Curtmola et al [11]	Byun et al [7]	Hwang et al [18]	Ryu et al [28]	Bao et al [3]
Single conjunctive keyword search	Single kws	Conjunctive kws	Single kws	Conjunctive kws	Conjunctive kws	Conjunctive kws	Single kws
Security	Non-adaptive	Adaptive*	Adaptive	Adaptive	Adaptive	Adaptive	Non-adaptive
MU File Submission	No	No	No	No	Yes	No	Yes
MU Search	No	No	Yes	No	Yes	No	Yes
MU Key Optimal	No	No	No	No	Yes, use u public keys	No	Yes
Revocation	-	-	Yes, decrypt not mentioned	-	No	-	Partial***
Symmetric / asymmetric	Symmetric	Symmetric	Symmetric	Asymmetric	Asymmetric	Symmetric	Symmetric

Table 1: Comparison of searchable encryption schemes

* Based on their second scheme, their first scheme uses a Shamir Secret Sharing mechanism.

** Adversary not allowed to search, incomplete security definition

*** Ability to revoke search rights, but not to revoke decryption rights.

In table 1 the letters n , w and u are used to denote the number of documents, the number of keywords and the number of users respectively. The abbreviation pr is used to denote the computationally expensive pairing operation. In the left column MU is used several times to abbreviate Multi User.

The solution for Curtmola et al [11] seems most efficient since the index and search complexity is only dependent on the number of keywords. In order to guarantee security dummy entries are added to the indexes which cause the index size and search complexity to grow to almost $O(w \cdot n)$.

2.1.2. Range Queries and Other Secure Search Techniques

In contrast to the equality queries in keyword searchable encryption schemes, range queries on numeric data mostly reveal the content upon a match. There is, however, a distinction in the amount of information revealed upon a match. Two different security models have been defined [29]; Match Concealing (MC) and Match Revealing (MR). Upon successful decryption the attribute values remain secret in MC secure schemes while those are revealed in MR secure schemes.

Range queries can be particularly useful in selectively releasing parts of a network audit log file. When some network problems occur, a virus outbreak for example, only the relevant parts of the network audit log can be released by only releasing the key(s) or trapdoor(s) for that certain timeframe. Another application is the ability to decrypt a message or transaction when it is classified as urgent or when the transaction value is above a certain amount.

Boneh and Waters [5] are the first to describe a scheme which besides equality queries also supports range and subset queries on encrypted data. The underlying technique is a generalization of Anonymous Identity Based Encryption (AIBE). The scheme uses a predefined vector of search terms which can be used for single value or conjunction equality, comparison or subset queries. Drawback is the efficiency of the scheme. The trapdoor size is linear to the amount of conjunctive searched keywords and their ciphertexts are larger as well. Another point is that their scheme uses a setting in which the owner of the secret key gives rights for certain index terms to specific users which can conditionally decrypt. For this reason it is not really useful in a multi-user file storage system.

Shi et al [29] described a system that supports multidimensional range queries which is illustrated with a network audit log example. Source and destination IP address, port number and time are, among others, different dimensions which all have their own specific range. In case of a problem a special range and domain dependent key can be released to an external party to investigate the problem. The system is thus able to selectively reveal data that matches the ranges where the token key is defined for. In contrast with the scheme by Boneh and Waters [5] the attributes are revealed upon a match, match revealing. The construction is based on binary search trees of height L and range T . Every dimension has its own tree and every node has its own unique ID. Plaintexts are encrypted under all IDs from the leaf to the root.

Although the above described techniques can be used for keyword searchable encryption it is not recommended since the previously described keyword search schemes are more applicable for our scenario. The just described techniques are more applicable in the case where a data owner needs to offer selective decryption rights. In multi user data storage application there are generally multiple data owners. It might be worthwhile to use both techniques in conjunction for example when besides keywords, date is important as well.

2.2. Proxy Re-Encryption

Besides the capability to search for all authorised users they should also be able to decrypt the information. In the introduction it was mentioned that a multi user secure file storage system exist without keyword search capabilities. This system makes use of a technique called proxy re-encryption. The concept of proxy re-cryptography has been introduced by Blaze, Bleumer and Straus [4]. Proxy re-encryption makes it possible to re-encrypt (transform) a message that was encrypted under the public key of Alice so that it can be decrypted by Bob. This is done without the need for Alice to release her secret key. Re-encryption is done with a special re-encryption key $rk_{A \rightarrow B}$ which can be computed by Alice. To compute the re-encryption key Alice uses her secret key and public key information of Bob. The message does not become available in plaintext during re-encryption. In this way neither the server nor any other person performing the re-encryption is able learn anything about the plaintext. Of course, the re-encryption key $rk_{A \rightarrow B}$ should only make it possible to re-encrypt messages from Alice to Bob and not from Bob to Alice. Schemes that have this property are called unidirectional. If re-encryption from Alice to Bob and from Bob to Alice is possible, the scheme is called bidirectional.

Most proxy re-encryption schemes offer both a first and a second level encryption of the plaintext. First level encryptions can only be decrypted by the intended recipient. Second level encryptions can be decrypted by the intended recipient and can be re-encrypted such that it can be decrypted by any of its delegates.

The first semantically secure unidirectional proxy re-encryption scheme is described by Ateniese et al [1]. Their scheme is illustrated in a secure multi user file storage system. They have also managed to implement their solution and show that calculations are performed within reasonable time. Also, they have constructed a list of useful properties for proxy re-encryption. The following list shortly describes these properties:

- **Unidirectional:** Delegation from Alice to Bob does not allow delegation from Bob to Alice.
- **Non-interactive:** Re-encryption keys from Alice to Bob can be generated by Alice with Bob's public key without interaction with a trusted party or Bob.
- **Proxy invisibility:** Alice and Bob are both aware of the proxy re-encryption protocol, but do not know whether the proxy has re-encrypted the message.

- **Original access:** Messages originally encrypted for Alice can still be opened by Alice after re-encryption.
- **Key optimal:** The amount of keys Bob has to store is independent of the amount of delegations he accepts.
- **Collusion safe:** Collusion of Bob and the Proxy should not make it possible to recover the secret key of Alice.
- **Temporary:** Bob is only able to decrypt re-encrypted messages which were originally intended for Alice during a certain time period.
- **Non-transitive:** The proxy cannot on itself generate delegation rights. For example, it is not possible to compute $rk_{A \rightarrow C}$ from $rk_{A \rightarrow B}$ and $rk_{B \rightarrow C}$.
- **Non-transferable:** Colluding delegates and the proxy cannot re-delegate decryption rights.

Not all of the properties listed above are required for the construction in this work. The security of the system is not harmed when users can determine if a message is re-encrypted. In fact all users should know this since they have to encrypt all messages under the master public key since all re-encryption keys are computed from the master secret key and the user public keys. Proxy invisibility is therefore not required. Original access is required, but not in the sense as stated above. The user and data manager should always be able to access the content stored at the Data Storage Service Provider (DSSP). It does, however, not need access to the content after re-encryption since the original ciphertext is still stored unchanged at the DSSP. Temporary is also not required in this setting since this will be covered by an authorisation and authentication mechanism. Access rights are revoked by simply deleting the re-encryption key of the respective user.

Canetti and Honenberger [8] and Libert and Vergnaud [20] have both described an improvement to the scheme to make it secure against Chosen Ciphertext Attacks (CCA). This improvement is at the cost of more complicated and less efficient constructions. The scheme by Canetti and Honenberger is bidirectional and can therefore not guarantee security if the proxy collude with any user [8]. Libert and Vergnaud have solved this issue. Their system is capable of detecting alterations to the ciphertext as well.

In this work the proxy re-encryption scheme by Ateniese et al [1] is used in the algorithm description. The other schemes could be used as well, but are not chosen since it will result in longer and more complicated algorithm descriptions.

2.3. Summary

In this chapter an overview is given of solutions found in scientific literature related to searchable encryption or proxy re-encryption. Literature describes both symmetric as asymmetric solution of which most support conjunctive keyword queries. Some schemes offer a multi user extension, but only one paper describes a true multi user system. Also, a comparison is given between the different schemes to evaluate the system constructed in this work.

Most proxy re-encryption schemes are based on the work of Ateniese et al [1], but are improved to be secure against chosen ciphertext attacks at the cost of more overhead. The solution by Ateniese et al [1] is chosen for its efficiency and simplicity. Other solutions are possible as well, but will further complicate the system's description.

In the next chapter the scenario that is shortly introduced in the introduction is further elaborated upon. It identifies the different actions and services and gives an overview of the different parties involved. The responsibilities of the parties are given and finally some assumptions are posed on the different parties and the system as a whole.

3.

Setting the Scene

Before moving on to the system construction, at first the scene which is already shortly introduced in the introduction is further elaborated on in this chapter. In the first section a distinction is made between the different actions and services within the scene and a description of these actions and services is given. After summarising the different actions and services the different parties involved, the actors, are defined. This includes a description of their tasks and responsibilities. Both the actions and services and the actors with their tasks are graphically illustrated.

Like in any system that uses cryptography several assumptions are made on the different actors and the system as a whole. The assumptions are given in the second section and describe the trust model and other assumptions posed on the different actors and the system. In the final section the requirements for the final system are posed based on the research goals given in the introduction.

3.1. Scene Description

In the introduction the problem of IT service outsourcing was raised. This report considers the scenario where data storage including its management is outsourced. In order for data storage outsourcing to be successful and to meet the research goals data confidentiality and manageability must be ensured.

When data confidentiality is not ensured, data leakage incidents might occur. Data leakage is a major problem for companies since the consequences can have enormous impact. Some of the consequences are a decreased corporate brand image or reputation, lost business opportunity and financial costs [32]. Data confidentiality is important and required by the research goals, but there will always be a trade-off between security and functionality. The confidentiality trade off can not be at any price and an optimal balance is needed. A highly secure system lacking functionality is doomed to fail. Of course employees still need to access their, encrypted, information. Employees do not only need to access their information internally or only with their laptops; the data has to be available from both within as outside the company and from both laptops as handheld devices. Besides availability and accessibility some search functionality needs to be in place.

Manageability of the system is important as well and also defined in the research goals. When new users can not efficiently be enrolled and revoked, the system can not work in practical environments where flexible systems are required. In most companies new employees are starting to work while

others are leaving. Authentication and authorisation mechanisms thus need to be updated frequently. Besides user management data management needs to be supported as well, for example to remove old documents.

A system that meets the scenario description given above can be decomposed in several actions and services. The different actions and services that can be distinguished are graphically illustrated below.

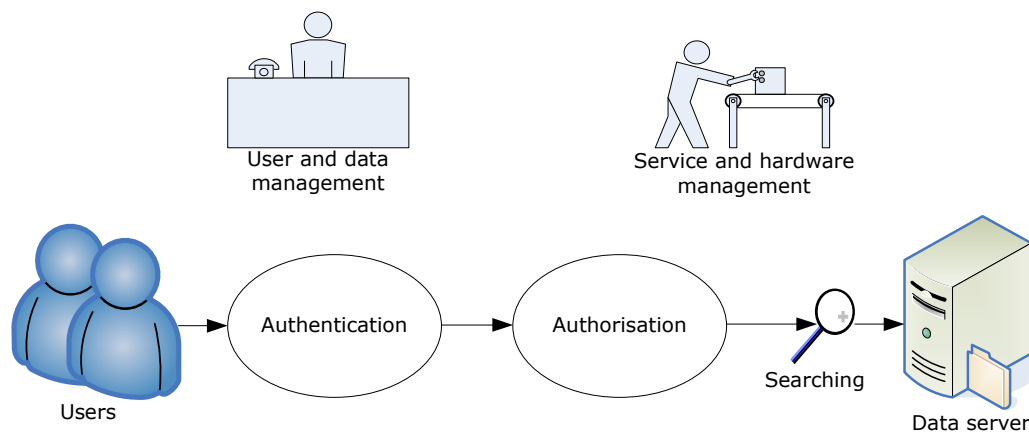


Figure 1: Services, actions and actors overview

The arrows in the picture above indicate the different stages in a user's search action. The different services and actions illustrated in the picture above are shortly described below.

- **Authentication:** Verification of the employee's identity that is connecting to the system. A simple form of authentication is checking the validity of a password supplied with the login name.
- **Authorisation:** Verification of the authenticated employee's credentials for the requested action or operation.
- **Service and hardware management:** Maintenance operations of both hardware and software to ensure an optimal server performance in encrypted data storage and search operations. Besides hardware management continuity and availability tasks such as creating backups ensuring enough additional power supplies belong to service and hardware management. Purchasing of new hardware as demands increase is included as well.
- **User and data management:** User management consists of all operations related to or involving users such as the enrolment of new users and revocation of existing users. Data management consists of all maintenance operations on the data residing on the data server such as cleaning up and removing old documents. Since security is an important aspect in this scenario, operations to ensure confidentiality of the stored data are considered data management as well.
- **Searching:** Finding the documents that contain the specified keyword(s) and returning the documents that match the criteria to the authorised employee.

The services and actions described above are divided over different actors or groups of actors. One of the system's goals is to outsource as much as possible. As a consequence as much tasks as securely and conveniently possible are delegated to the Data Storage Service Provider (DSSP). Some tasks such as user management can for security reasons not be outsourced and has to be managed by a trusted local employee. An overview of the different actors in this scenario are listed and described below.

- **Data Storage Service Provider (DSSP):** It is obvious that the DSSP will store the encrypted documents; otherwise no system has to be developed. The management and maintenance of the hardware and software is clearly a task of the DSSP as well as this is one of the main reasons to outsource data storage. Authentication and authorisation can be done at the customers company as

well, but a cost of less convenience since the local company has to be contacted. This inefficiency is especially true for mobile and other handheld devices with limited capabilities. Outsourcing authentication and authorisation delegates another task to the DSSP which is in line with the defined goals. Searching for data is a task for the data storage service provider as well, it makes no sense to search for data on another location as where it is residing.

- **User and data manager:** User management is a critical task regarding data confidentiality since it enrolls and revokes users from the system. In the enrolment process credentials are created. When the DSSP is able to perform this operation it has access to all data which poses a high security risk which is an undesirable situation. Data management operations like cleaning up can only be performed by the user and data manager since the DSSP has no access to the content. It is obvious that confidentiality related data management operations are performed by the user and data manager as well.
- **Users:** Should be able to use the complete functionality of the system according to their credentials. The system's functionality consists of document submission including encryption and specification of keywords to generate index elements, keyword searching and document decryption.

A graphical representation of the actor descriptions with their responsibilities and their place in the system given above is illustrated in the picture below.

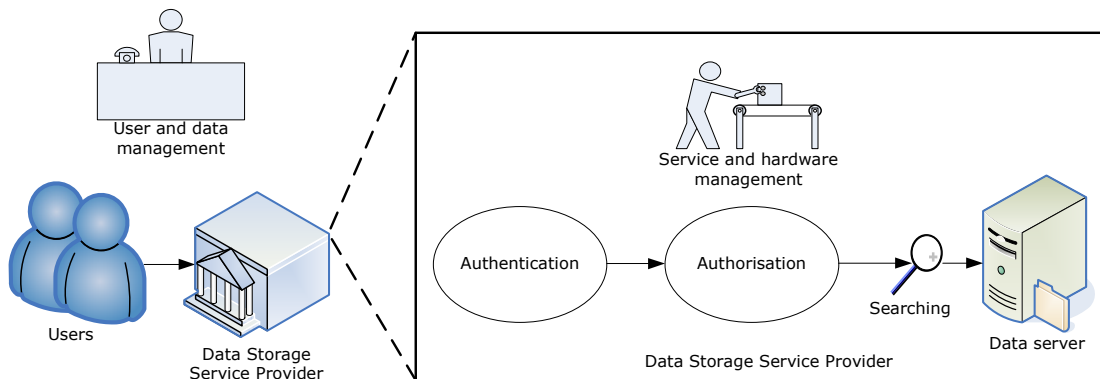


Figure 2: Actor overview in an outsourced data storage setting

In the rectangular box the responsibilities of the DSSP are illustrated. The next section describes the assumptions posed on the system as a whole and on the different actors.

3.2. Assumptions on Functionality and the Actors

This section gives a description of the assumptions posed on the different actors and the functionality of the system as a whole since choices have to be made. Assumptions related to the functionality of the system describe limitations on the capabilities or exclude some techniques from integration or implementation. Actor related assumptions describe limitations on the computational, storage or communicational complexity. Also, the honesty or intentions of the actors and their capabilities are described in this section.

One of the most common assumptions is that of a completely honest or trusted actor or third party. Trust assumptions are needed to guarantee correct functioning or security of the system. In this thesis assumptions are posed to make a solution feasible.

Assumptions on the system's functionality

Searchable encryption may sound as something that should be impossible since an encrypted message should reveal nothing about its underlying plaintext. For this reason it is not the actual encryption that is searched but a separately generated keyword index. Chapter two shows that this approach is taken in other searchable encryption schemes as well. In this work search capabilities will be limited to keyword search, including conjunctions, as well. The technique is comparable to tagging as what is often done to label blog entries.

Another technique to enhance the privacy of search operations is called Private Information Retrieval (PIR). This technique tries to hide which information is requested or retrieved and by who. PIR is a research topic on its own and is therefore not included.

Finally, this report only deals with the technical aspects of the system's core functionality. Issues such as integration with existing business applications, specifications of server hardware and software or specifications of communication channels or protocols are outside the scope of this assignment.

Assumptions on the data storage service provider

The data storage service provider is assumed to be honest but curious. This means that it will execute its tasks properly, but at the same time tries to uncover the secret information it stores. Uncovering the secret information can be done by trying to break the encryption or by trying to learn from observations. The honest but curious assumption seems realistic since correctly executing its tasks is of personal interest as well. When tasks are not correctly executed no customers will use the service.

Since correct operation is of personal interest for the DSSP the amount of computational power, the storage space and the bandwidth is assumed to be considerable. To keep being attractive as a service provider the provider is willing to invest in new and dedicated hardware.

Assumptions on the user and data manager

The user and data manager is considered to be honest. This is a reasonable assumption since this internal person is chosen by the client company itself. Correct execution and confidentiality is considered to be at his personal interest as well since it will be one of his or her evaluation criteria. If the user and data manager is corrupt, the security of the entire system is compromised. To increase the confidentiality insurance of the user and data manager a construction where every action always needs an approver could be used, but this is outside the scope of this assignment. Also, for the sake of simplicity only one user and data manager is considered.

The computational power of the user and data manager is assumed to be reasonable. Computational power of the user and data manager is not very important since its actions are not executed very often and have no influence on the system speed.

Users

Users are assumed to be honest since they are already authorised and thus there is no need to break into the system. The only way grant authorisation on the system to other persons is to reveal their personal secret information since this is considered something no user will do. Also, it is assumed that the user will not collude with the data storage service provider. A user can, however, share information with others after decryption, but no encryption mechanism can protect against this.

Computational power of users depends on their access medium which can be a computer such as a desktop or laptop or a handheld device such as a blackberry. On computers the computational power is assumed to be reasonable while the computational power on handheld devices will be limited. Document submission will be computationally more expensive than searching since query computation involves only a small set of keywords. Fortunately, most documents will be submitted from computers where they are created and handheld devices are mostly used to view documents.

3.3. Requirements

In this section the requirements to the system are formalised. The requirements are based on the research goal and its sub goals mentioned in the introduction covering the security, the manageability, the outsourcing aspects of the system and the system's efficiency. Also, some requirements concerning the system's functionality are posed.

Security

For the system's security and the data confidentiality the following requirements are posed.

- The data that is stored at the DSSP will contain sensitive information, for this reason data confidentiality must be ensured.
- The system will include keyword search functionality which means that queries are submitted to the system. These queries must have a construction in which the DSSP is unable to derive the queried keyword(s). Only the query itself and its result should be revealed.
- If the search technique uses indexes, as most techniques which are described in chapter 2 does, the confidentiality of these indexes must be ensured.
- It is desirable that the security of the final construction will be proven since the security and privacy an important aspect of the system.

Manageability

With manageability the tasks for the user and data manager are concerned. The following requirements are posed on the system's manageability.

- The user and data manager should be able to enrol new users and to revoke existing users from the system without the need to redistribute keys to existing users.
- The user and data manger should be able to access the contents stored at the DSSP to enable data management operations.

Outsourcing

This thesis deals with data storage outsourcing. Data storage outsourcing can be divided in several components as one could have seen in this chapter. The system should meet the following outsourcing requirements.

- All data storage and the search operations on the data need to be at the side of the DSSP.
- It is desirable that authentication and authorisation can be performed by the DSSP.

Efficiency and Functionality

Important for the system's success is the amount of available functionality and the system's efficiency. The following requirements are posed on the system's efficiency and functionality.

- Not only keyword search, but also conjunctive keyword search should be possible.
- All authorised users should be able to submit new documents, query the database and decrypt the matching documents.
- The system should have an at least comparable efficiency and performance (index size, data storage, query size and search complexity) as existing schemes. The efficiency and performance is evaluated based on the results given in table 1.
- The system's efficiency, search speed, should allow for practical applications. A search operation should be finished within a reasonable amount of time.
- It is desirable that a distinction can be made between users to make certain data only available for certain groups of users.
- It is desirable that the construction is implemented to verify the constructed system and to make speed measurements to test its practical performance.

3.4. Summary

In this chapter the system as described in the introduction was further elaborated on into a complete scene description. Within this scene different actions and services that should be available such as authentication and authorisation and search functionality were identified and illustrated. Hereafter, the different actors namely the DSSP, the user and data manager and the users were introduced. Also, their responsibilities were illustrated and described. In the next section various assumptions were posed on the systems functionality, such as that only keyword search is available, and on the actors. Actor assumptions are related to their intention and the amount of trust that can be placed in them. Also, assumptions are made on their computational complexity. In the final section requirements were posed on the system. The requirements are related to the research goals covering the security (confidentiality), manageability, outsourcing, efficiency and functionality aspects of the system.

In the next chapter a high level system construction and design is given that meets the scenario as it is introduced in this chapter. It makes use of the different actions and services building blocks as defined in this chapter.

4.

High Level System Construction

Now the scene is described and the assumptions on the actors are defined a high level design of the system can be constructed. This chapter describes the different possibilities came across during the system's design. Each option is evaluated and its advantages and disadvantages are discussed. Possible problems are identified and solutions proposed. Decisions for techniques to solve the problems that might arise are not yet made and will be given in the system's detailed description.

The first section describes the techniques that can be used for keyword search. Once the search technique is clear, index and query construction are described. Keyword index solutions can be constructed by using one index for the complete document collection or by using a separate keyword list (index) for every document. The section about query construction also describes the technique used to match the query with the indexes. Once the search part is clear, the possibilities to efficiently encrypt the data for all authorised users are described. Of course, only authorised users are allowed to search and thus an authentication and authorisation mechanism must be in place. The authentication and authorisation mechanism is described in section five. To get an idea of the complete system an illustration of how all parts are put together is given in the final section.

4.1. Search Technique

In this thesis security and data confidentiality is an important sub goal. Since the purpose of encrypting data is to hide all information about the underlying plaintext, searching in encrypted data may sound a bit unnatural. With a standard encryption scheme searching in ciphertexts is therefore not possible. Current research is focusing mainly on keyword search by constructing separate indexes. Approaches where individual words are encrypted are less secure than approaches where indexes are constructed.

An approach to support individual word search is to encrypt each word individually by using either block or stream ciphers. The use of block ciphers will result in a large overhead since word bit strings have to be padded to the minimum block length of the encryption algorithm. The average word length is approximately five characters [33] and takes thus 40 bits to represent in standard ASCII. In standard DES the block length is 64 bits, for an average word of length five it results in a data expansion by 60%. The now recommended AES standard has a minimum block length of 128 bits resulting in a data expansion even worse. Another concern with this approach is the vulnerability to statistical and dictionary attacks. One might say that this can be circumvented by using different initialisation

vectors, but then the search algorithm does no longer work. Different initialisation vectors cause different ciphertexts for the same word. Stream ciphers do not suffer from data expansion but have similar statistical and dictionary attack possibilities as the same input key stream is used multiple times.

Index solutions offer less flexibility, but a higher level of security. The indexes can be constructed independent of the encryption mechanism used. A standard encryption scheme is used to encrypt the data and another to construct the indexes. Every document is associated with one or more keywords in order to make keyword search possible. With index construction care has to be taken that indexes are constructed in a secure way. Although suffering less from data expansion and having less security concerns, it is at the cost of more limited search capabilities for text based documents. On the contrary this approach is independent of the content format and can thus also be used for other formats such as audio, video, compressed data, technical designs and so on.

The approach using indexes is more secure, since even if the indexes are constructed in an insecure way only a limited amount of information is revealed; it is thus more secure than the approach where every keyword is encrypted individually. An index based approach is thus chosen for security reasons.

4.2. Keyword Index Construction and Storage

The keyword searchable encryption scheme which is constructed in this thesis requires relevant keywords to be manually selected for every document. In order to circumvent ambiguity problems a predefined set of possible keywords, called the dictionary, has to be specified. Instead of manual keyword selection a software tool to automatically select relevant keywords can be used, but this does not influence operation of the developed system. When users or employees can choose any keyword they like it has the advantage that it is very flexible in the sense that a suitable keyword can always be found. This flexible solution looks very attractive at first sight, but it is not always an advantage while searching for documents. For example, some keywords have synonyms or on the contrary a keyword can have two different meanings depending on the context.

Data confidentiality is one of the system's sub goals and specified in the requirements. Therefore it is obvious that the selected keywords can not be stored in plaintext since it reveals secret information. To hide the keywords, they need to be encrypted or a one way function such as a hash function needs to be applied. Applying a one way hash function on every keyword is the most optimal choice and makes searching an easy operation. By using the same hash function on the queried keywords, searching is possible with an easy equality check. A property of hash functions is that they map any (length) input of to a fixed length output. Hash functions successfully hide the length of keywords, but due to the relatively small amount of keywords an attacker can easily compute the hash values of every keyword. Encrypting the keywords under the same symmetric key makes it possible to perform equality searches as well. Drawback of this approach is that key management can be complicated and that it is vulnerable to statistical attacks. In order to overcome the vulnerability problems with hash functions and encryptions a probabilistic approach has to be used since this makes it infeasible to compute all possibilities. Randomness can for example be introduced by blinding, a mathematical operation with a random number.

The way in which keywords are stored will not have much influence on the manageability of the system and is therefore not discussed in this section. The outsourcing aspect is not discussed since the indexes will be stored together with the data at the DSSP.

There are two different approaches to index construction. The first is to have one global index for all documents and in the second approach every document has its own index of matching keywords. The

next paragraphs give a description and an illustration of both approaches to decide which construction is most applicable in our scenario.

4.2.1. One Index for All Documents

Whether documents are stored in a database or in a repository it seems like a good idea to have only one index to search in. This means that there is one central place to search and that the keywords are not attached to the documents themselves. The construction is straightforward and as follows. Every keyword in the dictionary has a list with a number of as little as zero or as many as n id's of documents associated with the keyword. A drawback of this construction it is that possible to see how many documents are associated with a particular keyword. In the figure 3 a graphically illustration of this index construction is given.

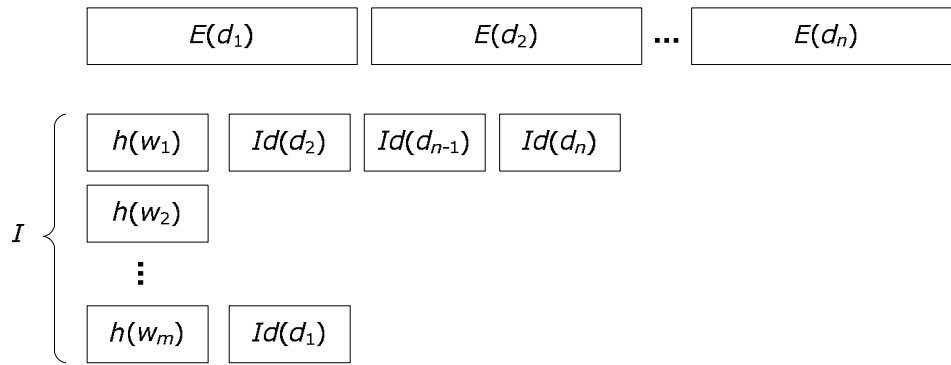


Figure 3: One index for all documents

In the top of figure 3 the collection of n encrypted documents $E(d)$ is shown. Below the index I is shown with on the left side m hashed keywords $h(w)$ and on the right side the identifiers $Id(d)$ of documents that are associated with the particular keywords. Another observation that can be made is that this solution is both computationally and space efficient since only one index has to be matched and stored instead of one for every document. This is especially true for a large document collection with a small dictionary.

The security goal states that the amount of information that can be deduced from the information stored at the DSSP and thus from the index has to be minimized. Strong points are that amount of keywords per document and the amount of documents per keyword can be hidden. On the contrary it is not possible to securely and efficiently perform conjunctive keyword searches. When the document identifiers are stored in a probabilistic manner the amount keywords per document is automatically hidden. The number of documents per keyword can easily be hidden by adding dummy items to the index, but this adds some overhead to the system. Due to the construction it is impossible to search for documents containing multiple keywords without revealing which documents only partially match. It is possible to store separate conjunctive keyword index terms, but then the index size grows exponentially. This is especially a problem with large dictionaries.

Updating the index when documents are added or removed is more complicated as well. Instead of removing or adding a new index, the global index has to be changed in a secure way. In this construction some information is revealed to the server.

4.2.2. A Separate Index for Every Document

Instead of one index for the complete document collection a separate index containing the relevant keywords can be constructed for every document. A separate index for every document is an easy approach since no measures have to be taken to securely update an existing global index. The user submitting the document is responsible for both document encryption and index construction. To automatically hide the number of keywords per document a fixed length index is used. An index with length equal to the number of keywords in the dictionary is most convenient. The index construction is straight forward; every document is appended with several relevant keyword hashes. In figure 4 a graphical illustration of this index construction is given.

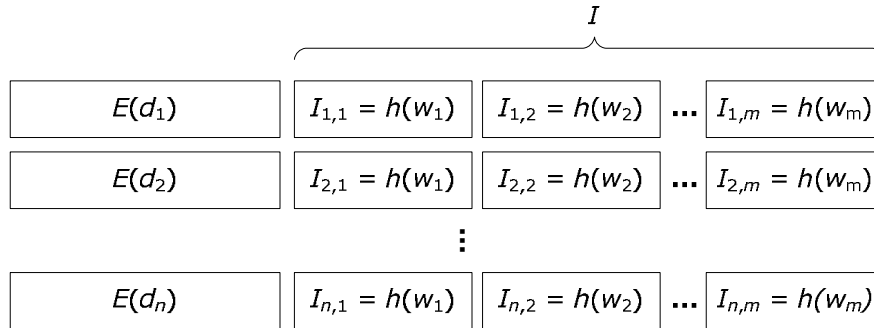


Figure 4: A separate index for every document

On the left side of the picture the complete document collection containing n encrypted documents $E(d)$ is shown. The indexes I for every document are shown on the right side and consists of m index terms $h(w)$ with keywords w .

For both security and convenience reasons fixed length indexes of length equal to the dictionary size will be used. Variable length indexes have the advantage that less storage space is needed, but on the contrary it reveals the amount of keywords that are added to the document. In order to reveal the least amount of information to the server the number of keywords per document has to be hidden. Indexes of a fixed length do not have this problem and are for this reason preferred. Fixed length indexes can be constructed in two ways. Either by choosing a reasonable size maximum number of keywords or by constructing indexes of length equal to the number of keywords in the dictionary. By using dictionary length indexes it is a good idea to always place the same keyword on the same location in the index. Keyword fields at the location of irrelevant keywords can use a random number instead of the keyword as input to the hash function. This solution has it advantages with query matching as will become clearer in the next section when query construction and matching is described. A drawback is that, depending on the matching technique, it might enable to server to learn more information. Also, more storage space is needed with dictionary length indexes. When the popular SHA-1 hash function is used, only 160 bits or 20 bytes are needed to store a hash value. Considering the size of an empty Word 2003 document, which is 19.968 bytes, the amount of space needed to store an index is reasonable.

4.3. Query Construction and Matching

When sending a query to the server at least some information is revealed, e.g. the fact that a user executes a query and its result. The challenge is to minimise the amount of information the server is able to learn while at the same time keeping the computational and storage complexity at a reasonable level.

For the construction of queries there are two possibilities either having a separate value for every keyword in the query or having one value covering all keywords in the query. Below the construction of both possibilities for query q in case where two keywords are queried is given.

- **Multiple values:** $q(w_1, w_2) = (h(w_1), h(w_2))$
- **One value:** $q(w_1, w_2) = (h(w_1) \times h(w_2))$

As one can see queries are constructed in a similar way as indexes except that no hash values are computed for irrelevant keywords. This construction makes matching with simple equality checks possible.

The second construction where one value for all queried keywords is submitted is chosen since it is both more efficient and more secure although it can decrease the search quality. It is more efficient since less information has to be transmitted to the DSSP. When separate values for every queried word are submitted it becomes similar to executing a query for every individual keyword and returning the intersection. The server is thus able to determine which documents partially match. With only one value submitted for all queried keywords the server is no longer able to deduce this information. Drawback of the second approach using only one value for all queried keywords is that multiplication of the hashed keywords can be equal to a multiplication of two different hashed keywords resulting in a false match, e.g. $h(w_1) \times h(w_2) = h(w_3) \times h(w_4)$. Despite the chance for false positives it is chosen since the impact of a few false positives is relatively low and the system's security and efficiency are considered more important.

Dependent on the index construction there are several different constructions possible to match documents. The list below gives an overview of some scenarios for successful matching including a short analysis. For the descriptions below it is assumed that every document has its own index. Some of the descriptions might be applicable in case one global index was used as well.

- **Compute or store all combinations:** In case of a query with multiple keywords all keyword combinations can be computed beforehand and stored in the index or all index combinations can be computed at query execution. Both naive solutions are not optimal, either the storage complexity or the computational complexity is very high.
- **Remember keyword locations for every file:** If there are no fixed keyword locations the keywords locations for every file have to be remembered. This makes no sense since keywords itself can be remembered instead, or a special way in which index locations can secretly be found has to be thought of.
- **Fixed keyword locations:** When all documents have an index size equal to the size of the dictionary and if every keyword appears always at the same position in the index efficient matching is possible if the keyword locations are specified to the server.

The most applicable solution is the use of fixed keyword locations. In case where a relatively small dictionary is used the computation of all combinations might be doable, but with large dictionaries and queries containing multiple keywords this becomes infeasible. At the cost of giving more information to the server (keyword locations) and thus being slightly less secure the server is able to match far more efficiently.

For searching the most important property is called *query correctness*. This property states that for every valid query submitted, the correct result should always be returned from the system. When this is not the case it makes no sense to offer search capabilities.

4.4. Multi User Data Encryption

All documents that are stored at the DSSP should be accessible to all authorised users and thus all authorised users should be able to decrypt them. For this reason a multi user data encryption mechanism is needed. Although multi user data encryption is not a standard cryptographic term, in this thesis it means a public key encryption scheme in which the data is encrypted in such a way that all authorised users can decrypt it without revealing their secret key or other secret information.

A naive approach is to store a different encryption for every user. This approach fulfils the security requirements, but is not easily manageable and suffers from a large storage overhead. Revoking users from the system is easily possible by deleting all encryptions for the respective user. Enrolling new users is a more complicated process. To make all content available for a new user all documents have to be decrypted first and then encrypted with the symmetric key of the new user. Especially in systems with large document collection, this is a very inefficient process. Since for every document a different copy is stored for every user the amount of storage overhead on the server is enormous. It is obvious that a more efficient data encryption mechanism should be found than this naive approach.

The system will be far more efficient if all data is encrypted under the same symmetric key. In this approach all users share a same common secret, the symmetric key, and is thus potentially less secure. Also, it is not possible to efficiently revoke existing users. If a user is revoked from the system the user still has its symmetric key and is thus still able to decrypt the information. In order to revoke decryption rights all data has to be decrypted, then encrypted with a new symmetric key and the new symmetric key has to be distributed to all authorised users. Revocation is a frequent process in a multi user system. Repeated key distribution is not ideal and therefore a solution with a minimum amount of key redistribution is preferred.

Broadcast encryption tries to solve the problem of efficient sharing encrypted data between a constantly changing group of users, but assumes only one data owner. Current broadcast encryption schemes are more efficient and better manageable than the naive approach. In order to make decryption possible for all authorised users some overhead in the form of an additional header is attached to the message and all authorised users need to store several keys. The problem broadcast encryptions tries to solve is, however, slightly different than the one in this thesis. In this thesis every user can be a data owner and it is not required that every user receives the same message, only that they are able to decrypt. Another difference is that not all users have to receive all messages, but only the user who requested the message. Therefore it poses no burden on the communications if every user receives a different encryption of the same message upon request.

Proxy re-encryption and proxy encryption are asymmetric encryption schemes which have a similar scenario as in this thesis. Although asymmetric encryption algorithms are not known for their efficiency, they can efficiently be applied when combined with a symmetric encryption scheme. Since proxy re-encryption is more secure than proxy encryption it is selected. The difference between proxy re-encryption and proxy encryption is that in proxy encryption the plaintext will become visible which is not the case in proxy re-encryption. With proxy re-encryption a lesser amount of trust in the proxy is required. In order to make the proxy re-encryption scheme more efficient every document is encrypted under a different symmetric key. This symmetric key is, on its turn, encrypted with the proxy re-encryption algorithm under an asymmetric key.

In a proxy re-encryption scheme decryption rights can be delegated. For example, Alice is going on a holiday and wants Bob to take over her work. To successfully take over Alice's work Bob need access to her private information. Alice does, however, not want to reveal her secret key material to Bob so she computes a special re-encryption key which enables Bob to decrypt the messages originally intended for Alice. The proxy re-encryption scheme works in the following way. The symmetric keys used for document encryption are encrypted under a master public key. From this master public key a different re-encryption key is computed for every authorised user. In this way the DSSP is able to re-

encrypt the documents such that all authorised users are able to decrypt. Revocation is easily possible by deleting a user’s re-encryption key from the data sever. This construction allows to securely outsource multi-user data encryption and is efficiently manageable, it thus satisfies the outsourcing and manageability requirements. The construction is graphically illustrated in the picture below.

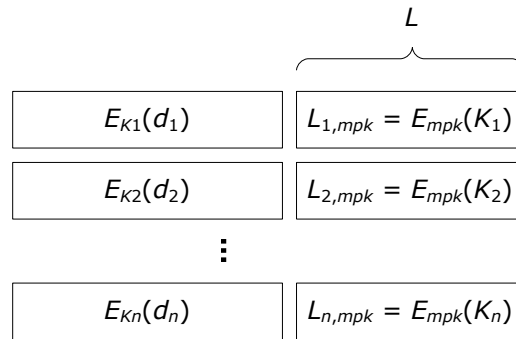


Figure 5: Documents with their lockboxes

On the left side of the picture one can see the encrypted document collection of n encrypted documents $E_K(d)$. The documents are encrypted with a symmetric encryption algorithm $E(\cdot)$ under key K . The right side of the picture shows the lockboxes L_{mpk} . Lockboxes are the symmetric keys K encrypted with the proxy re-encryption algorithm $E(\cdot)$ under the master public key mpk .

4.5. Authentication and Authorisation

Security and data confidentiality is defined as one of the research goals and therefore an access control mechanism performing authentication and authorisation should be in place such that only authorised users are able to use the system’s functionality. Traditionally access control mechanisms are placed in a trusted environment such as within the company or at a trusted third party. Control of these access control servers means access to all functionality and the ability to generate new credentials.

In this thesis authentication and authorisation is outsourced as well which is in line with the research goals and requirements. For this reason a different authentication and authorisation mechanism should be applied. This construction should no longer require a high amount of trust on the server performing authentication and authorisation. The idea to make sure that only authorised users can use the system’s functionality is that information from both the DSSP and information from the authorised user is needed in order to perform an action. In this way authentication and authorisation is one operation. If an unauthorised user requests an action while claiming to be an authorised user the action will be performed, but it will result in failure even is the submitted information is in the correct format.

The following paragraphs first explain the authentication and authorisation procedure for query and index authorisation where after the authentication and authorisation procedure for decryption is described.

4.5.1. Query and Index Authentication and Authorisation

The authentication and authorisation mechanism for queries and indexes is similar and therefore described in the same paragraph. As already mentioned actions can only be performed if both parties, namely the DSSP and the user, cooperate. To meet the system's security requirements authentication and authorisation should be possible for users without the need to reveal his secret key material or share a common secret.

Authentication and authorisation is based on two keys for every user from which one is stored at the DSSP (the complementary search key) and one at the user (the secret search key). The authentication and authorisation procedure now works in the following way. First, the user computes a query or an index as described earlier in the chapter and processes this under his secret search key. The query or index together with the user's identity is now sent to the DSSP. At the DSSP the user's identity is used to find the corresponding complementary search key. The complementary search key of the user is now used to perform the authentication and authorisation step. This step removes the user specific term from the query or index such that the submitted queries or indexes are respectively executed or stored in a format independent of the submitter. The transformation to user independent indexes and queries make it possible to search for all authorised users without having to share a common secret. This is the same approach as taken in [3]. In the figure below a graphical illustration of the query submission process is given.

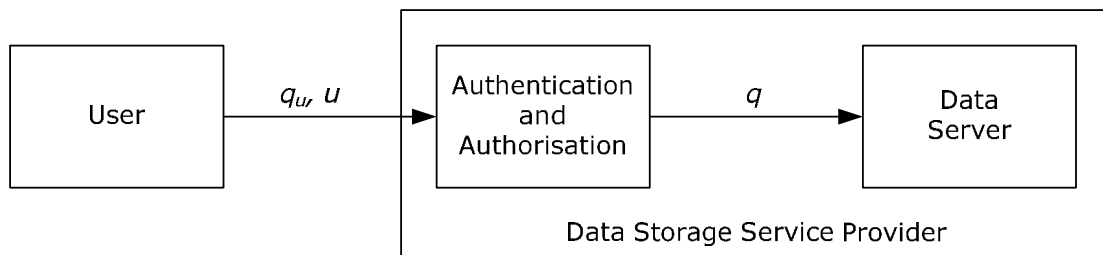


Figure 6: Query authorisation

In the figure 6 one can see the process of query authorisation. The process is initiated by the user who submits a user dependent query q_u . This user dependent query is sent to the DSSP together with the user identity u . At the DSSP the authentication and authorisation step is performed with the complementary search key for user u and the result is a user independent query q .

An unauthorised user submitting a correctly constructed query does not negatively effect the system's operation other than some computational time and a query without any results. Wrongly selected queries will fail immediately and are thus not considered as a problem. Unauthorised users submitting a document with a correctly constructed index result in a never matching document on the server. The user and data manager should be able to remove those documents.

4.5.2. Decryption Authentication and Authorisation

The authentication and authorisation mechanism for decryption is different than that of indexes and queries as described in the beginning of this section. With indexes or queries the authentication and authorisation procedure transforms the content to a user independent format. Decryption authentication and authorisation is exactly the opposite way. Here the content which is encrypted on under the master public key is transformed to as if was encrypted under the user's public key. Similar to index and query authentication and authorisation decryption is only possible if both parties, namely the DSSP and the user, cooperate.

Decryption authentication and authorisation is based on two keys for every user from which one is stored at the DSSP (the re-encryption key) and one at the user (the secret key of his or her public key pair). This is exactly the same idea as with index and query authorisation. The re-encryption key is computed from the master secret key and the user's public key. Since the user and data manger holds the master secret key it is an interesting attack target for malicious parties and therefore the master secret key should be stored offline. When a user wants to decrypt a message the authentication and authorisation procedure now works to following way. First, the user sends a decryption request for a particular document together with its user identity to the DSSP. The DSSP finds both the requested document and the re-encryption key for the respective user. Only if both parts are found the DSSP can continue and re-encrypts the lockbox of the requested document. The result is send back to the user who uses his secret key to decrypt the symmetric key in lockbox. This symmetric key is now used to decrypt the requested document. In the figure below a graphical illustration of this process is given.

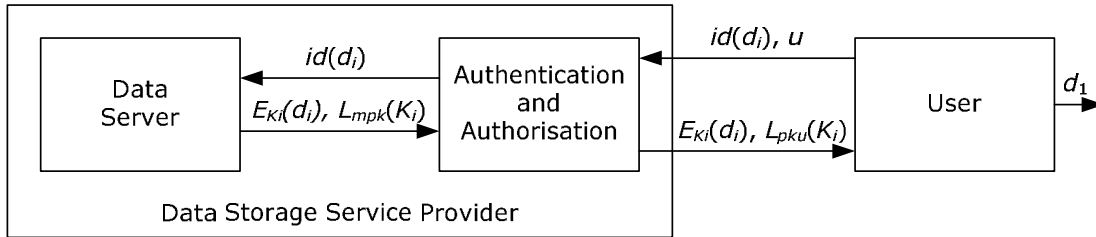


Figure 7: Decryption authorization

In figure 7 above one can see that the user initiates the document decryption process by sending an identifier of the requested document $id(d)$ and the user identity u to the DSSP. The DSSP starts verifying the user identity by finding the re-encryption key rk_u of user u . When this key is available the requested document $E_K(d)$, encrypted under symmetric key K , together with its lockbox $L_{mpk}(K)$, encrypted under the master public key mpk , is found and the lockbox is re-encrypted to as if it was encrypted under the users public key pk_u . The encrypted document together with the re-encrypted lockbox $L_{pku}(K)$ is now returned to the user who uses his secret key sku to decrypt document d .

4.6. Putting Together

In the previous sections the designs of the different parts of the system have been described as individual pieces. Most parts need input in a specific format and can work individually of other parts as long as the input is in the correct format. The input of the different parts is mostly based on information submitted by the user at the time of an action and on information stored at the DSSP.

The next section describes and gives an overview of how the data and keys are stored at the DSSP. It includes a graphical representation of the data storage. In the final section a system overview is given which is graphically illustrated as well.

4.6.1. DSSP Data and Key Storage Representation

Both the encryption mechanism and the keyword index add overhead to the document as became clear from the descriptions in previous sections. The amount of overhead added due to encryption will be reasonable. Symmetric encryption schemes do not suffer much from data expansion and a public key encryption of a symmetric key does only take a relatively small amount of space. Indexes will lead to

a larger amount of overhead. The exact amount depends on the number of distinct keywords in the dictionary. Below a graphical illustration of the data representation at the DSSP is given.

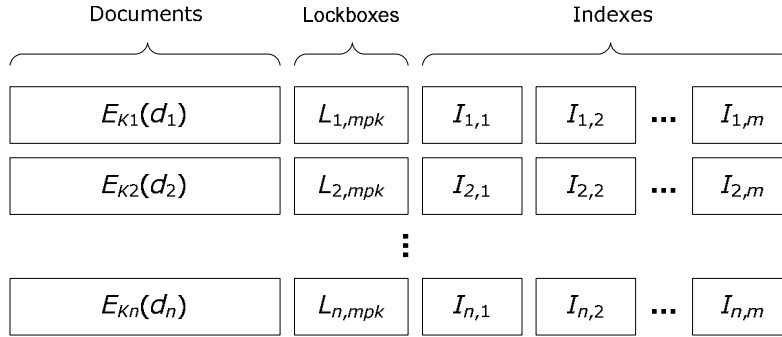


Figure 8: Data representation on the data server

In the figure 8 n denotes the number of documents stored at the DSSP and m denotes the number of keywords in the dictionary. For clarity reasons lockboxes and indexes are denoted by L and I respectively and are constructed as described in their sections.

Besides the actual content and the overhead for decryption and search functionality the DSSP stores key material for authentication and authorisation purposes. This authentication and authorisation key material can easily be stored in a database table. The first field then contains the user’s unique identity and the second column stores the user’s complementary search key needed to remove the user dependent factor from indexes and queries. In the final and third column the re-encryption key is stored which enables re-encryption of documents to a format the users can decrypt.

4.6.2. System Overview

In figure 9 one can see a complete overview of the system. The picture illustrates three of the system’s most important operations, query submission and searching followed by decryption.

In the figure the management functions such as user and data management and service management such as hardware and software management are omitted. A figure which includes their place in the system can be found in chapter 3.

A step by step overview from query generation by the user until decryption of the results is given in figure 9. The process starts with the user choosing relevant keywords, in this case ‘sales’ and ‘secret’. From these keywords a query is computed with the user’s secret search key. The resulting query is, together with the location of the keywords in the index and the user identity, send to the DSSP. At the DSSP the complementary search key for the specified user is found for query authentication and authorisation. After successful authentication and authorisation a user independent query is obtained which is, together with the keyword locations, send the data server. The data server now checks the keywords at the specified locations with the submitted keywords for every document. All matching documents are now returned for decryption authentication and authorisation (proxy re-encryption). Once the matching documents are authenticated and authorised for decryption the resulting set of documents is returned to the user. The user now uses his secret key to decrypt the lockboxes to obtain the symmetric keys which are used to decrypt the matching documents.

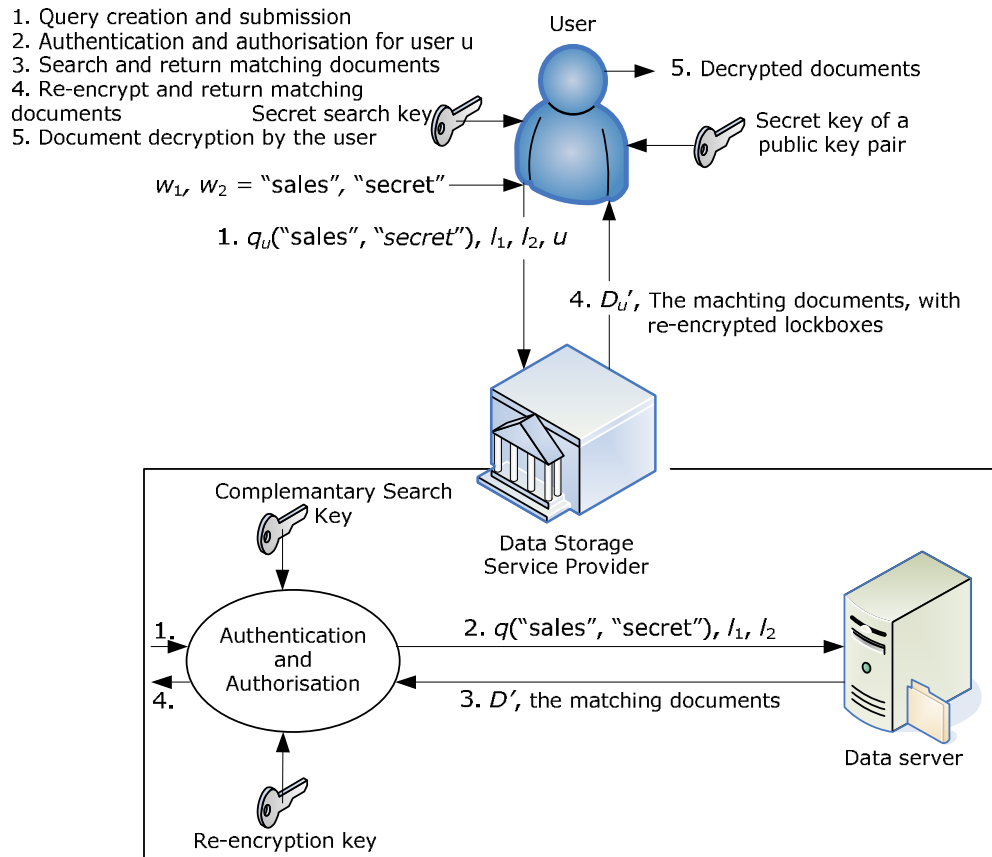


Figure 9: From query submission to decryption (schematic overview)

4.7. Summary

In this chapter a description of the system's high level design is given. During the design several different options came across and the best options are selected.

First, the search technique was described and a solution using keyword indexes is chosen. The relevant keywords are hashed and associated with the document. In order to overcome ambiguity problems a fixed set of keywords, called the dictionary, is defined. Keyword indexes can be constructed for the system as a whole or for each document individually. A separate index for every document offers more flexibility and a higher level of security compared to a global index and is for this reason chosen. Due to the relatively small amount of keywords plain hash values are insecure and for this reason some randomness has to be introduced.

Queries that are submitted to the system can have a value for every queried keyword or one value covering all queried keywords. One value covering all queried keywords is chosen since this reveals less information to the DSSP. In order to efficiently match, in the sense that less computational power is needed, the locations of the queried keywords are specified.

The most efficient and most applicable multi user data encryption scheme for this scene is proxy re-encryption. Proxy re-encryption offers delegation of decryption rights, it can enable decryption of documents to specific users, in this case all authorised users.

Authentication and authorisation is based on a combination of information from which one part is stored by the user and the other part is stored at the DSSP. Only if both parties cooperate the system's functionality can be used.

In the next chapter some mathematical concepts and preliminaries are introduced needed to understand the system's description. Also, the system is formally defined and security definitions for the system are posed.

5.

Preliminaries

Before giving a detailed description of the system, some mathematical preliminaries and a formal definition of the system are given. The system's detailed description is based on mathematics and, similar to most keyword searchable encryption schemes, frequently uses the bilinear pairing operation. A clear mathematical definition of a bilinear map including a description of the underlying mathematics illustrated with an example is given in the first section. Hereafter proxy re-encryption is formally defined.

After defining the mathematical preliminaries the system is formally defined. In the last part of this chapter the security definition of the system is given. This security definition makes it possible to prove the system's theoretical security.

5.1. Cryptographic Preliminaries

The solutions proposed in this work use an already existing proxy re-encryption scheme combined with newly developed searchable encryption techniques. The focus is thus on searchable encryption. Existing techniques are used as a basis and will further be improved. Since the proposed scheme will use bilinear maps a short definition is given to understand the proposed techniques. A definition of proxy re-encryption is given for the same reason.

5.1.1. Bilinear Map (Pairing)

Most proposed keyword searchable encryption schemes use a bilinear map (also called pairing) and the scheme developed in this work is no exception. The proxy re-encryption techniques used in this work uses bilinear maps as well. A bilinear map is defined in the following way [3].

Definition 5.1 (Bilinear Map)

Let G_1 , G_2 and G_T be multiplicative groups of prime order q . A bilinear map is a function $e : G_1 \times G_2 \rightarrow G_T$, satisfying the following properties:

1. **Bilinear:** For all $g \in G_1$, $h \in G_2$ and all $x_1, x_2 \in \mathbb{Z}_q^*$, $e(g^{x_1}, h^{x_2}) = e(g, h)^{x_1 x_2}$.

2. **Non-degenerate:** If g is a generator of G_1 and h is a generator of G_2 , then $e(g, h)$ is a generator of G_T .
3. **Computable:** $e(g, h)$ can efficiently be computed for any $g \in G_1$ and $h \in G_2$.

In the above definition of a bilinear map two different groups G_1 and G_2 are used to construct G_T , but using G_1 twice is possible as well. When the same group is used twice this is called a symmetric bilinear map.

While the properties described above can be used without understanding the mathematics of the pairing function a short description of the pairing function will be given following the descriptions given in [22]. Pairing functions operate over groups of points on elliptic curves. An elliptic curve E over the field \mathbb{F}_q has the following form: $E : Y^2 = X^2 + aX + b$. In literature there are different pairing functions described, but the two most popular pairing functions are the Weil and the Tate pairing. In this section the pairing function is illustrated with the Tate pairing. Both pairing functions use Miller's algorithm for their computations. Miller's algorithm makes use of tangents, verticals and lines between two points. The tangent, verticals and lines for a point $P(x, y)$ can be computed in the following way.

- Tangents $T_P : -(3x^2 + a)X + (2y)Y + (-2y^2 + x(3x^2 + a))$
- Verticals $V_P : X + (-x)$
- Lines $L_{Z,P} : (y_1 - y_2)X + (x_2 - x_1)Y + (x_1y_2 - x_2y_1)$ with $Z(x_1, y_1)$ and $P(x_2, y_2)$

Miller's algorithm for the Tate pairing, denoted by $f_P(Q)$, takes points P (of order r) and Q and a binary representation of the group size $r = \{r_t, \dots, r_0\}$ with $t = \lfloor \log_2 r \rfloor$ as input and is as follows.

Miller's Algorithm
<pre> f ← 1 Z ← P for i ← t - 1, ..., 0 do f ← f² · T_Z(Q) / V_{2Z}(Q) Z ← 2Z if r_i = 1 then f ← f · L_{Z,P}(Q) / V_{Z+P}(Q) Z ← Z + P end if end for </pre>

Figure 10: Miller's Algorithm

To illustrate the Tate pairing consider the elliptic curve $E : Y^2 = X^3 + X$ ($a = 1$ and $b = 0$) over \mathbb{F}_{23} which contains 23 points, all operations are thus modulo 23. Let group G_1 be a subgroup of points in \mathbb{F}_{23} of size $r = 6$ and $P = (11,10)$ be a generator of G_1 . Group G_1 then contains the points $P = (11,10)$, $2P = (13,18)$, $3P = (15,20)$, $4P = (9,18)$, $5P = (19,22)$ and $6P = (1,5)$. For super singular curves the points in G_2 can be found by using a distortion map (a projection) from the points in G_1 . The distortion map is defined as $\Phi(x, y) = (-x, iy)$ (with $i^2 = -1$) and G_2 then contains the points $Q = (-11,10i)$, $2Q = (-13,18i)$, $3Q = (-15,20i)$, $4Q = (-9,18i)$, $5Q = (-19,22i)$ and $6Q = (-1,5i)$. The embedding degree k is 2 since this is the first k that satisfies $6 \mid 23^k - 1$ and the binary representation of the group size $r = \{1,1,0\}$ with $t = \lfloor \log_2 6 \rfloor = 2$. By using Miller's algorithm the Tate pairing for points $P = (11,10)$ and $Q = (-11,10i)$ is now computed in the following way:

1. $f_1 = 1$
2. $Z = (11,10)$
3. For $i = 1$:
 Compute tangent for $Z = (11,10)$ and the vertical for $2Z = (13,18)$

$$T_P : 4X + 20Y + 9 = 14X + Y + 20$$

$$V_{2P} : X + 10$$

$$T_P(Q) = T_P(12, 10i) = 14 \cdot 12 + 10i + 20 = 4 + 10i,$$

$$V_{2P}(Q) = V_{2P}(12, 10i) = 12 + 10 = 22.$$

$$f_2 = f_1^2 \cdot T_P(Q) / V_{2P}(Q) = 1^2 \cdot (4 + 10i) / 22 = 19 + 13i.$$

$$Z = 2Z = (13, 18)$$

Since $m_1 = 1$ compute the line between Z, P and vertical for $Z + P = 3P = (15, 20)$

$$L_{2P} : 8X + 21Y + 1 = 19X + Y + 11$$

$$V_{3P} : X + 8$$

$$L_{2P}(Q) = L_{2P}(12, 10i) = 19 \cdot 12 + 10i + 11 = 9 + 10i,$$

$$V_{3P}(Q) = V_{3P}(12, 10i) = 12 + 8 = 20.$$

$$f_3 = f_2 \cdot L_{2P}(Q) / V_{3P}(Q) = (19 + 13i) \cdot (9 + 10i) / 20 = 17 + 5i.$$

$$Z = Z + P = (15, 20)$$

For $i = 0$:

Compute tangent for $Z = (15, 20)$ and the vertical for $2Z = (1, 5)$

$$T_{3P} : 14X + 17Y + 2 = 13X + Y + 15$$

$$V_{6P} : X + 22$$

$$T_{3P}(Q) = T_{3P}(12, 10i) = 13 \cdot 12 + 10i + 15 = 10 + 10i,$$

$$V_{6P}(Q) = V_{6P}(12, 10i) = 12 + 22 = 11.$$

$$f_4 = f_3^2 \cdot T_{3P}(Q) / V_{6P}(Q)$$

$$= (17 + 5i)^2 \cdot (10 + 10i) / 11$$

$$= (11 + 9i) \cdot (10 + 10i) \cdot 21$$

$$= 6 + 14i.$$

4. In the final step the output of Miller's algorithm is standardised to get an unique value:

$$f_P(Q)^{(q^k-1)/r} = (6+14i)^{(23^2-1)/6} = 11+15i.$$

As a final check the answer it is raised to the power of r which should give 1. For our example this is $(11 + 15i)^6 = 1$, which is as expected.

5.1.2. Proxy Re-encryption

The proxy re-encryption algorithm used in the proposed schemes is described in [1] and consists of six Probabilistic Polynomial Time (PPT) algorithms, it is defined as follows.

Definition 5.2 (Proxy Re-encryption) *A proxy re-encryption scheme consists of the following PPT algorithms.*

Setup and Key Generation: It takes a security parameter k as input and generates (public) system parameters and a public key pair for every user u (pk_u, sk_u).

Re-encryption Key Generation: Takes as input (sk_A, pk_B) , A 's secret key and B 's public key to generate the re-encryption key $rk_{A \rightarrow B}$ from user A to user B .

First Level Encryption: On input pk_A and message m it outputs a ciphertext $C_{A,1}$. A first level encryption can only be decrypted by the intended recipient and can not be re-encrypted.

Second Level Encryption: On input pk_A and message m it outputs a ciphertext $C_{A,2}$. A second level encryption can be decrypted by the intended recipient and can be re-encrypted so it can be decrypted by any of its delegates.

Re-encryption: On the input of re-encryption key $rk_{A \rightarrow B}$ and ciphertext $C_{A,2}$ it outputs the re-encrypted ciphertext $C_{B,1}$. It re-encrypts a second level encryption for user A into a first level encryption for user B .

Decryption: On the input of sk_A and a first or second level ciphertext $C_{A,1}$ or $C_{A,2}$ it outputs message m .

In the definition of most proxy re-encryption schemes a distinction is made between first and second level encryptions. Since in this work only the second level encryptions are used, encryption with a proxy re-encryption scheme always means a second level encryption. A detailed description of proxy re-encryption can be found in [1].

5.2. System Definition

In this section the multi user encrypted file storage system with keyword search capabilities developed in this thesis is defined. This system consists of the following PPT algorithms:

Setup(k_s, k_p): A probabilistic algorithm executed by the user and data manager to set up the system and to initialise the system parameters. Input argument k_s and k_p are the security parameters. The algorithm will output the dictionary W , the secret key for the user and data manager x and the master public key pair for the re-encryption scheme (mpk, msk).

Enroll(x, msk, u): This algorithm is executed by the user and data manager to add a new user u to the system. It outputs a secret search key and a complementary search key pair (x_u, CSK_u), a public key pair (pk_u, sk_u) and a re-encryption key rk_u .

AddDocument($x_u, w_1, \dots, w_i, l_1, \dots, l_i, d, |W|, mpk; CSK_u, u$): The first part of this algorithm is run by the user u and the second part by the DSSP to encrypt the document and to generate the index. The user generates the index using the keywords w_1, \dots, w_i and encrypts the document d under a randomly generated symmetric key K . The symmetric key K is encrypted with the proxy re-encryption scheme under the master public key mpk and the result together with the user identity is sent to the DSSP. The DSSP performs the access control computations based on the input and the CSK_u and adds the document d' to the document collection D .

SubmitQuery($x_u, w_1, \dots, w_i, l_1, \dots, l_i; CSK_u, u$): This first part of this algorithm is run by the user u and the second part by the DSSP to generate a valid query for the specified keywords w_1, \dots, w_i selected from dictionary. The query q_u including the keyword locations in the index l_1, \dots, l_i together with the user's identity u are sent the DSSP which performs the authentication and authorisation computations based on CSK_u and outputs q .

Search(q, D, l_1, \dots, l_i): This algorithm is run by the DSSP. It takes as input a query q , the document collection D and a set of keyword locations l_1, \dots, l_i and searches for the matching keywords on the specified locations. It outputs the set of matching documents D' .

Decrypt($D', rk_u, u; sk_u$): The first part of this algorithm is run by the DSSP and the second part by the user u to decrypt a (set of) document(s). It re-encrypts the lockboxes of document set D' with the re-encryption key rk_u and sends the result D_u' to user u . The user decrypts the re-encrypted lockboxes with his or her secret key sk_u to obtain the symmetric keys K which are used to decrypt the documents.

Revoke(u): This algorithm is run by the user and data manager to revoke a user from the system. It removes the complementary search key CSK_u and the re-encryption key rk_u for user u from the DSSP and thus the user is no longer able to search and decrypt documents.

Figure 11: System definition

5.3. Security Definitions

Every searchable encryption scheme will reveal at least some information to the server, but this information has to be minimised. Data, index and query confidentiality are covered in the security and confidentiality sub goal and for this reason a proof of the system's security is required. First, a distinction is made between different security and confidentiality categories. The different security aspects / categories of the system are listed below.

- **Data Confidentiality:** The DSSP or any other unauthorised party is unable to determine the content stored at the DSSP. This is a requirement for the proxy re-encryption scheme that will be used, which is independent of the index construction.
- **Index Confidentiality:** The DSSP or any other unauthorised party with access to the document collection and indexes is unable to determine the keyword(s) associated with a document.
- **Query Confidentiality:** The amount of information the DSSP learns from a search operation should be limited to the search outcome and information derived thereof, e.g. the database access pattern.
- **Query Uniqueness:** Besides that a query for a keyword w should be unique for every user, every generation of a query for keyword w by the same user should be different as well. Also, neither a user nor the DSSP should be able create a query on behalf of another user. This is a stronger notion of security than defined by F. Bao et al [3] where only different queries for different users are required.
- **Revocation:** Ability to revoke search and decryption rights for users which are no longer legitimate. Ideally, this should be done without the need to redistribute keys.

As mentioned before every searchable encryption scheme will inevitably reveal some information unless Private Information Retrieval (PIR) techniques are used. It is, however, important to minimise the amount of information leaked from a search outcome and information derived thereof (e.g. the access pattern). No information about the documents itself (data confidentiality), the keywords searched for (query confidentiality) or the keywords associated with the document (index confidentiality) should be revealed.

For data encryption a proxy re-encryption scheme is chosen and used without alterations to the algorithms. The chosen scheme should of course guarantee data security. For this reason defining or proving the security of such a scheme makes no sense and will thus not be included in this work.

Neither the server nor any other party should be able to determine the keywords in the indexes and queries and the security and privacy of both has to be proven. It has been proven that secure indexes and secure queries does not guarantee complete security if they are proven secure separately [11]. It is therefore important to incorporate the secure indexes and secure queries in one definition. The general security definition will be given in the next section.

Query uniqueness is part of the authentication and authorisation mechanism, but contributes to query privacy and the general security of the system as well. Every user that generates a query for a keyword w should have a different query in order to perform authentication and authorisation. This requirement is typical for a multi-user setting. In addition to this requirement a user generating a

query for the same keyword w twice should end up with two different queries as well. The main reason is for security so that an eavesdropper cannot perform statistical attacks.

Finally, in the last section a definition of revocability is defined. It requires that revoked users are no longer able to construct valid queries and indexes and use the system's functionality.

5.3.1. Security Definition

Different security definitions exist, some offer only non-adaptive security while others offer adaptive security as well. In non-adaptive security definitions the system only guarantees security when all queries are executed at once. A stronger notion of security is adaptive security where query information of previous rounds is taken into account while constructing new queries. This is a more realistic definition since in practice searching is often a process of fine tuning the query terms. Both adaptive and non adaptive definitions exist in two kinds; indistinguishability and semantic security definitions. An intuitive notion of an indistinguishability problem is the following; given two plaintexts and an encryption of one of them the adversary is unable to distinguish to which plaintext the encryption belongs. Semantic security definitions say that it is computationally infeasible for any Probabilistic Polynomial Time (PPT) adversary A to derive any significant information from a ciphertext. It has been proved that those definitions are equivalent for a keyword searchable encryption scheme [11].

To be able to compare the security of the system's searchable part with other schemes, general notions of security are used. The first notion of adaptive security for keyword searchable encryption was introduced by Curtmola et al [11][10]. These definitions did also form the basis of the definitions by F. Bao et al [3], which also describe a multi-user searchable encryption scheme. Their scene setting is closely related to the scene used in this report and their definitions are therefore used as a basis.

During operation there necessarily is interaction between users and the DSSP. This access pattern can be observed by an adversary or the DSSP. The access pattern consists of instances $\Omega(q, loc) = \{u_q, a_q\}$ where q is the query, loc the encrypted location information, u_q the issuer of the query and a_q the set of matching documents. A sequence of t submitted queries is denoted by $Q_t = (q_1, q_2, \dots, q_t)$ and the corresponding keyword locations are denoted by $Loc_t = (loc_1, loc_2, \dots, loc_t)$. The corresponding keywords and replies are denoted by $W_t = (w_1, w_2, \dots, w_t)$ and $A_t = (a_1, a_2, \dots, a_t)$ respectively. All information that is left behind from the access pattern is called the trace denoted by $T_t = (|D|, \Omega(q_1, loc_1), \dots, \Omega(q_t, loc_t), |U_A|)$ where $|U_A|$ denotes the number of users in the system. This is all information the adversary is allowed to know. The DSSP has access to more information, besides the trace it has access to the all information stored on its servers as well. All information the DSSP is able to learn over t queries is called the view and denoted by $V_t = (d_1 = (E_K(d_1), I_1), \dots, d_n = (E_K(d_n), I_n), CSK-list, Q_t, Loc_t, A_t)$. In order to guarantee adaptive security the simulator should be able to simulate a partial view $V_t^p = (d_1 = (E_K(d_1), I_1), \dots, d_n = (E_K(d_n), I_n), CSK-list, Q_p, Loc_p, A_p)$ given only a partial trace $T_t^p = (|D|, \Omega(q_1, loc_1), \dots, \Omega(q_p, loc_p), |U_A|)$, where $0 \leq p \leq t$. For every p between 0 and t the simulator should simulate partial view V_t^p given only the partial trace T_t^p . The following simulation based definition defines the security of a searchable encryption scheme.

Definition 5.3 (Adaptive semantic security for searchable encryption). *A multi-user encrypted database system is semantically secure if for the document collection D , for all $t \in \mathbb{N}$, for a PPT algorithm A , there exists a PPT algorithm (the simulator) S , such that for all V_t, T_t , all $0 \leq p \leq t$ for any function f :*

$$|\Pr[A(V_t^p) = f(D, W_p)] - \Pr[S(T_t^p) = f(D, W_p)]| < 1 / p(k)$$

where the probability is taken over the internal coins of A and S .

In words the definition above states the following; everything that can be computed from the partial view of the DSSP V_i^p (the contents stored at the server and the access pattern) can also be computed from observations, the partial access pattern T_i^p . The access pattern is all the adversary is allowed to know. User-server collusion is not included in this security definition. This is in line with the assumptions made earlier in this report. Also in literature there are no definitions found that includes user-server collusion.

5.3.2. Revocability

Important in a multi user system is the possibility to revoke search and decryption rights for users which are no longer authorised. Complementary search key material stored on the access control server is needed to perform the search operation. Without this material it is impossible to distinguish between matching and mismatching indexes. Search revocability is thus satisfied by deleting the user's complementary search key. For a user to decrypt a document a re-encryption key has to be available. Without this key it is impossible to decrypt a document. Decryption revocability is thus satisfied by deleting the user's re-encryption key.

A practical property of the efficiency of revocation is called *key optimal*. A system that is key optimal has efficient revocation and enrolment of users without the need of redistributing keys. The number of secret keys users have to hold is independent of the number of documents and users the system holds.

5.4. Summary

In this chapter the mathematical definitions were given of a bilinear map and proxy re-encryption. The pairing of these bilinear maps is an important operation in the system's construction. For multi user data encryption an already existing proxy re-encryption scheme is used which is defined in the beginning of this chapter as well.

Besides the mathematical preliminaries some formal definitions are given. First, the system developed in this work is formally defined. To prove the system's security a formal security definitions was given as well.

In the next chapter a detailed description of the system is given according to the formal definition given in this chapter. The preliminaries and definitions introduced in this chapter are used in the detailed descriptions.

6.

Constructing a Searchable Encryption Scheme

This chapter gives a detailed description of the constructed systems. Two attempts are made to come to the final system which satisfies the research goal and its sub goals. The first solution is very efficient, but offers only single keyword search. In the second scheme conjunctive keyword search is enabled and some other improvements have been made. Both attempts are analysed and their capabilities and limitations discussed.

Since security is defined as one of the system's goals, the final solution is proven secure according to the definition given in chapter 5.

After the detailed description an extension to the system is given to support Role Based Access Control (RBAC) where groups of users have different document access rights depending on their function or role. The final section gives a more high level security analysis and describes possible prevention measures. Finally, some additional security measures and functionality which might contribute to the system's security and functionality are described.

6.1. Attempt 1 – A Keyword Searchable Scheme

In this section a description of the first attempt made to construct a keyword searchable encryption scheme satisfying the research goal and its sub goals is given. The construction builds on the high level design as given in chapter 4.

In chapter 4 a decision was made for keyword searchable encryption using indexes. Every document has its own index with the associated keyword taken from the dictionary. In chapter 4 the problem of using a hash value of keywords taken from a dictionary was already identified. This issue can be resolved by introducing randomness to the hashed values. An easy and often used technique to gain probabilistic output is called blinding. With blinding a mathematical operation with a random number is performed. The challenge is to retrieve or discard this random number, called the blinding factor. In order to still be able to search some additional information containing the blinding factor has to be specified with the indexes and queries.

As already mentioned in chapter 4 the actual data is encrypted by using the proxy re-encryption mechanism by Ateniese et al [1]. There are no alterations made to the proxy re-encryption algorithms and are for this reason not further discussed.

Chapter 4 also discussed the authentication and authorisation step which needed the cooperation of both the user and the DSSP. This attempt uses a version based on the construction by Bao et al [3] which is as follows. The user and data manager generates a random number x which is his secret search key (the master secret search key) and chooses generator g . For every user a distinct key pair is generated. The user's secret search key x_u is determined by generating a random number and the user's complementary search key CSK_u is computed as g^{x/x_u} . Authentication and authorisation is now performed by calculating $(CSK_u)^{x_u} = (g^{x/x_u})^{x_u} = g^x$. One can see that this step removes the user dependent factor x_u . Only if a corresponding complementary search key is used, a valid value of g^x is obtained.

In this attempt the authentication and authorisation approach described above is used and a new search algorithm is developed. It is obvious that the user can not send his secret search key x_u to the DSSP in plaintext. Also the DSSP must not be able to determine the user's secret search key. Similar to the hashed keywords this can easily be resolved by blinding. The same blinding factor is used to blind the user's secret search key and the hashed keyword, but the blinding factor is different for every index or query. Indexes and queries are constructed the same way and to be able to match the blinding factor has to be removed. Removal of the blinding factor is done in the authentication and authorisation step. An index or query is constructed in the following form: $(x_u r, h(w)^{1/r})$, where r is the blinding factor. The DSSP performs a pairing operation using CSK_u to remove the blinding factor and the user's secret search key, $e(g^{x/x_u}, h(w)^{1/r})^{r x_u} = e(g^x, h(w))$. One can easily see that this results in a user independent encryption of the hashed keyword which can not be computed by either the user or the DSSP alone.

A complete description of the first attempt is given below. Both the search part and the proxy re-encryption part use bilinear maps. The bilinear maps and the corresponding pairing operations are, however, different. For this reason the subscript p is used to denote the symbols belonging to the proxy re-encryption algorithms and the subscript s is used to denote the symbols belonging to the search algorithms. An explanation of the used symbols is given in appendix A.

Function	Explanation
Setup(k_s, k_p):	<p>User and data manager:</p> <p>Given security parameter k_s choose groups $G_{s,1}$, $G_{s,2}$ and $G_{s,T}$ of prime order $q > 2^{k_s}$ with bilinear map $e_s : G_{s,1} \times G_{s,2} \rightarrow G_{s,T}$ and a generator $g_s \in_R G_{s,1}$ for the search and index algorithms.</p> <p>Given security parameter k_p choose groups $G_{p,1}$ and $G_{p,T}$ of prime order $q > 2^{k_p}$ with bilinear map $e_p : G_{p,1} \times G_{p,1} \rightarrow G_{p,T}$ and a generator $g_p \in_R G_{p,1}$ for the proxy re-encryption algorithm.</p> <p>Select the user and data manager secret search key $x \in_R \mathbb{Z}_q^*$ and store it offline.</p> <p>Generate the master public key pair in the form $mpk = (Z^{udm_1}, g_p^{udm_2})$, $msk = (udm_1, udm_2)$, where $Z = e_p(g_p, g_p)$ and with $udm_1, udm_2 \in_R \mathbb{Z}_q^*$.</p> <p>Define the dictionary W and a hash function $h : \{0,1\}^* \rightarrow G_{s,2}$.</p>
Enroll(x, msk, u):	<p>User and data manager:</p> <p>Select a user u's secret search key $x_u \in_R \mathbb{Z}_q^*$.</p> <p>Compute a user u's complementary search key $CSK_u = g_s^{x/x_u}$.</p> <p>Generate a user u's public key pair $pk_u = (Z^{u_1}, g_p^{u_2})$, $sk_u = (u_1, u_2)$ with $u_1, u_2 \in_R \mathbb{Z}_q^*$.</p> <p>Compute a user u's re-encryption key $rk_u = g_p^{udm_1 u_2}$.</p> <p>Send (u, CSK_u, rk_u) securely to the DSSP who adds it to the authorised users list U_A.</p> <p>Send $(x_u, (pk_u, sk_u), W, g_p)$ securely to user u.</p>

AddDocument $(x_u, w, d, mpk, CSK_u, u)$:	<p>User: Select blinding elements $r, v \in_R \mathbb{Z}_q^*$. Compute the user dependent index $I_u = (x_u r, h(w)^{1/r})$ Encrypt the document using a standard symmetric encryption algorithm $E_K(d)$, e.g. AES with a randomly chosen symmetric key K. Construct the lockbox by using the proxy re-encryption encryption algorithm $L_{mpk} = (g_p^v, KZ^{udm_1 v})$. Send $d_u' = (E_K(d), L_{mpk}, I_u)$ and the user identity u to the DSSP.</p> <p>DSSP: Let $I_u = (I_{u,0}, I_{u,1}) = (x_u r, h(w)^{1/r})$. The DSSP finds CSK_u for user u and returns 'invalid' if CSK_u is not available, otherwise compute $I = e_s(CSK_u, I_{u,1}(w))^{I(w)_{u,0}} = e_s(g_s^{x/x_u}, h(w)^{1/r})^{r x_u} = e_s(g_s^x, h(w))$. Add $d' = (E_K(d), L_{mpk}, I)$ to the document collection D.</p>
SubmitQuery(x_u, w, CSK_u, u):	<p>User: Select blinding element $s \in_R \mathbb{Z}_q^*$. Construct a user dependent query $q_u = (x_u s, h(w)^{1/s})$. Send q_u and the user identity u to the DSSP.</p> <p>DSSP: Let $q_u = (q_{u,0}, q_{u,1}) = (x_u s, h(w)^{1/s})$. The DSSP finds CSK_u for user u and returns 'invalid' if CSK_u is not available, otherwise compute $q = e_s(CSK_u, q_{u,1})^{q_{u,0}} = e_s(g_s^{x/x_u}, h(w)^{1/s})^{s x_u} = e_s(g_s^x, h(w))$. Send q to the data server.</p>
Search(q, D):	<p>DSSP: For every document $d_i' \in D$ test if the query matches the index $q / I_i = 1$, add matching documents d_i' to D' and return the results.</p>
Decrypt($D', rk_u, u; sk_u$):	<p>DSSP: Let $L_{mpk} = (L_{mpk,0}, L_{mpk,1}) = (g_p^v, KZ^{udm_1 v})$. For every document $d_i \in D'$ compute $e_p(L_{mpk,0}, rk_u) = e_p(g_p^v, g_p^{udm_1 u_2}) = Z^{udm_1 u_2 v}$ and re-encrypt all lockboxes $L_{i, pku} = (Z^{udm_1 u_2 v}, KZ^{udm_1 v}) = (Z^{u_2 v'}, KZ^{v'})$, with $udm_1 v = v'$, and send D_u', the documents and the re-encrypted lockboxes, to user u.</p> <p>User: Let $L_{pku} = (L_{pku,0}, L_{pku,1}) = (Z^{u_2 v'}, KZ^{v'})$. For every document $d_{u,i} \in D_u'$ the user decrypts the symmetric keys using the secret key $sk_u = (u_1, u_2)$ of his public key pair, $K = \frac{L_{pku,1}}{L_{pku,0}^{1/u_2}} = \frac{KZ^{v'}}{(Z^{u_2 v'})^{1/u_2}} = \frac{KZ^{v'}}{Z^{v'}}$ and decrypts the documents $d_i = D_K(E_K(d))$.</p>
Revoke(u):	<p>User and data manager: The user and data manager deletes the re-encryption key rk_u and the complementary search key $CSK_u(u, rk_u, CSK_u)$ from the user table stored at the DSSP.</p>

Figure 12: Attempt 1 - A keyword searchable encryption scheme

Searching will be the most frequent operation in the system. Before a search operation can be performed a valid query has to be submitted by a user. After the actual search operation the set of matching documents D_u' has to be returned to query submitter who decrypts the documents. During this action authentication and authorisation is performed as well. The complete process from the

submission of a query to the decryption of the matching information covers three algorithms from the description above: *SubmitQuery*, *Search* and *Decrypt*.

Below a picture is given to illustrate the process from query submission to the decryption of the results. In the picture the different steps, the communication and the input are shown. The first step, denoted by 1, is the construction and submission of a query by the user to the DSSP. On the left side of the user the input for this action is illustrated. The input consists of the user's secret search key x_u and the keyword w which is "sales". Query q_u and the user identity u are sent to the DSSP for the second step, authentication and authorisation. As one can see in the picture the complementary search key CSK_u is needed for this operation. The result q is sent to the data server who executes the query and sends back the results, the third step. Before the result is returned to the user, decryption authentication and authorisation is performed with the user's re-encryption key rk_u . The set of matching documents and their re-encrypted lockboxes is then returned to the user, step four. In the fifth step the user's secret key is used to decrypt the lockboxes. The keys inside the lockboxes are used to decrypt the documents such that the user ends up with the documents in plaintext.

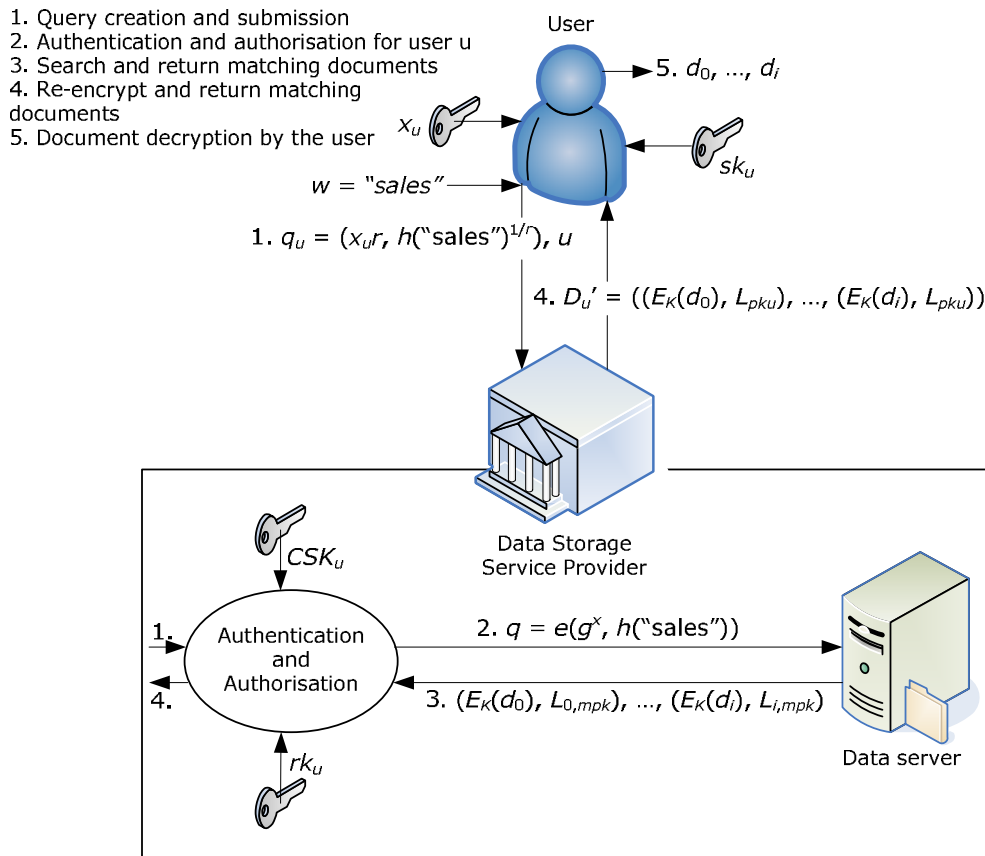


Figure 13: Attempt 1 - Query submission, searching and document decryption

6.1.1. Evaluation

The first attempt to construct a keyword searchable encryption scheme has a very efficient search operation, but is only able perform single keyword search. An evaluation of the system based on the system's goals is given below.

Although no formal security proof is given the system is considered secure in the sense that only authorised users can use the system's functionality. As described in chapter 4 the authentication and authorisation mechanism is based on cooperation of both the user and the DSSP. All information the users send to the DSSP, including the secret search key x_u and the hashed keywords, is blinded and thus multiple submissions of the same information will result in different values. For this reason no secure communication channels are needed to ensure data confidentiality. Another advantage of the blinding operation is that the DSSP is not able to determine the user's secret search key x_u and is therefore not able to perform actions on its own. The pairing operation in the authentication and authorisation step removes the blinding factor and the user's secret search key from both the additional information and the hashed keyword. This enables the system to match queries and indexes. A consequence is that the indexes, although they are stored encrypted, are no longer probabilistic and can thus be vulnerable to statistical attacks.

The first attempt is easily manageable. Every user is able to perform actions based on key information in his personal storage and key information stored at the DSSP. This key material is independent of other users, the number of users in the system or the number of documents in the system. The user and data manager is thus able to efficiently revoke users from the system and to enrol users to the system without redistributing keys to existing users.

This attempt satisfies the outsourcing sub goal since as much tasks as securely possible are outsourced to the DSSP. It is clear that storage of the data and the search operations on the indexes of this data have to be outsourced. Authentication and authorisation which is usually performed locally or at a trusted client is outsourced to the DSSP as well even though it is not considered trusted. Due to the used mechanism where the DSSP is not able to generate rights this is not a security threat. Actions which can only be performed by the users and user management, which is a trust function, are not outsourced.

The construction of attempt one is very efficient since searching takes only one equality check per document and is independent of the number of users. A bilinear pairing is an expensive operation compared to a multiplication or an exponentiation. Attempt one needs only one pairing operation for authentication and authorisation. The execution of a query needs thus only one pairing and a number of equality checks based on the number of documents. This construction is thus more efficient than the solution by Bao et al [3]. The difference is that the indexes in their solution are stored in probabilistic format. In this system the number of secret keys a user has to store is very efficient, only one key for the search part and only one key for the decryption is needed. The system is key optimal since no redistribution of keys is required when enrolling or revoking users.

6.2. Attempt 2 – A Conjunctive Keyword Searchable Scheme

In this section the second attempt to construct a conjunctive keyword searchable encryption scheme that satisfies the research goal and its sub goals is described. The construction given in this section is based on the high level design description in chapter 4 and the construction of attempt one, but has several improvements.

The two most important improvements are the ability to perform conjunctive keyword searches and probabilistic index storage. As described in chapter 4 every document has its own index of length equal to the number of keywords in the dictionary. In addition an extra term with additional information is added for authentication and authorisation purposes. Similar to attempt one the user's secret search key x_u and the index terms $h(w)$ or queried keywords are blinded. Although the key construction remains unchanged the blinding factor is not removed during the authentication and authorisation procedure. The user's secret search key is blinded in the same way as in attempt one, $x_u r$ where r is the blinding factor. Index terms or the queried keywords are blinded by an exponentiation

with $r, h(w)^r$. In order to still be able to match the blinding factor is compensated for during the search process.

Authentication and authorisation is performed by calculating $CSK_u^{I_{u,0}} = g^{rx_u/x_u} = g^{rx}$. This computation removes the user specific information, the user's secret search key, from the additional information. Queries and indexes are constructed in a similar way and matching is made possible by two bilinear pairings where on both sides of the equation the blinding factor of the index (r) and the blinding factor of the query (s) is included, $e(g^{sx}, h(w)^r) = e(g, h(w))^{rxs}$. On the other side of the equation the locations of r and s swapped. In this way the blinding factor is compensated for and matching is made possible. In case of multiple keywords the hash values are multiplied before pairing.

In comparison with attempt one only the *Setup*, *AddDocument*, *SubmitQuery* and *Search* actions have changed. For completeness reasons a complete description of the second attempt is given. The changes are printed in bold, except the authentication and authorisation formulas for document and query submission and the search formula. A more high level system definition can be found in section 5.2 and an explanation of the used symbols is given in appendix A.

Function	Algorithm
Setup(k_s, k_p):	<p>User and data manager:</p> <p>Given security parameter k_s choose groups $G_{s,1}$, $G_{s,2}$ and $G_{s,T}$ of prime order $q > 2^{k_s}$ with bilinear map $e_s : G_{s,1} \times G_{s,2} \rightarrow G_{s,T}$ and a generator $g_s \in_R G_{s,1}$ for the search and index algorithms.</p> <p>Given security parameter k_p choose groups $G_{p,1}$ and $G_{p,T}$ of prime order $q > 2^{k_p}$ with bilinear map $e_p : G_{p,1} \times G_{p,1} \rightarrow G_{p,T}$ and a generator $g_p \in_R G_{p,1}$ for the proxy re-encryption algorithm.</p> <p>Select the user and data manager secret search key $x \in_R \mathbb{Z}_q^*$ and store it offline.</p> <p>Generate the master public key pair in the form $mpk = (Z^{udm_1}, g_p^{udm_2})$, $msk = (udm_1, udm_2)$, where $Z = e_p(g_p, g_p)$ and with $udm_1, udm_2 \in_R \mathbb{Z}_q^*$.</p> <p>Generate a public key pair for the DSSP from a standard public key algorithm e.g. ElGamal, (pk_{DSSP}, sk_{DSSP}).</p> <p>Define the dictionary W and a hash function $h : \{0,1\}^* \rightarrow G_{s,2}$.</p>
Enroll(x, msk, u):	<p>User and data manager:</p> <p>Select a user u's secret search key $x_u \in_R \mathbb{Z}_q^*$.</p> <p>Compute a user u's complementary search key $CSK_u = g_s^{x/x_u}$.</p> <p>Generate a user u's public key pair $pk_u = (Z^{u_1}, g_p^{u_2})$, $sk_u = (u_1, u_2)$ with $u_1, u_2 \in_R \mathbb{Z}_q^*$.</p> <p>Compute a user u's re-encryption key $rk_u = g_p^{udm_1 u_2}$.</p> <p>Send (u, CSK_u, rk_u) securely to the DSSP who adds it to the user list U_A.</p> <p>Send $(x_u, (pk_u, sk_u), W, g_p)$ securely to user u.</p>

<p>AddDocument $(x_u, w_1, \dots, w_i,$ $l_1, \dots, l_i, d,$ $W , mpk;$ $CSK_u, u):$</p>	<p>User: Select blinding elements $r, v \in_R \mathbb{Z}_q^*$. For all specified locations l_1, \dots, l_i hash the keyword specified at the same location in w_1, \dots, w_i. For all other locations $1, \dots, W$ hash a random number. Compute the user dependent index $I_u = (rx_u, h(w_1)^r, \dots, h(w_{ W })^r)$, where w can also be a random number as described above. Encrypt the document using a standard symmetric encryption algorithm $E_K(d)$, e.g. AES with a randomly chosen symmetric key K. Construct the lockbox by using the proxy re-encryption encryption algorithm $L_{mpk} = (g_p^v, KZ^{udm_1v})$. Send $d_u' = (E_K(d), L_{mpk}, I_u)$ and the user identity u to the DSSP. DSSP: Let $I_u = (I_{u,0}, I_{u,1}, \dots, I_{u, W }) = (rx_u, h(w_1)^r, \dots, h(w_{ W })^r)$ The DSSP finds CSK_u for user u and returns 'invalid' if CSK_u is not available, otherwise compute $I = (CSK_u^{I_{u,0}}, I_{u,1}, \dots, I_{u, W }) = (g_s^{rx_u/x_u}, h(w_1)^r, \dots, h(w_{ W })^r) = (g_s^{rx}, h(w_1)^r, \dots, h(w_{ W })^r)$. Add $d' = (E_K(d), L_{mpk}, I)$ to the document collection D.</p>
<p>SubmitQuery($x_u, w_1, \dots, w_i,$ $l_1, \dots, l_i,$ $pk_{DSSP}; CSK_u,$ $u):$</p>	<p>User: Select blinding element $s \in_R \mathbb{Z}_q^*$. Construct a user dependent query $q_u = (sx_u, h(w_1)^s \times \dots \times h(w_i)^s)$. Encrypt the corresponding keyword locations $loc = E_{pk_{DSSP}}(l_1, \dots, l_i)$. Send q_u, the encrypted keyword locations loc and the user identity u to the DSSP. DSSP: Let $q_u = (q_{u,0}, q_{u,1}) = (sx_u, h(w_1)^s \times \dots \times h(w_i)^s)$. The DSSP finds CSK_u for user u and returns 'invalid' if CSK_u is not available, otherwise decrypt the encrypted keyword locations $D_{sk_{DSSP}}(loc)$ and compute $q = (CSK_u^{q_{u,0}}, q_{u,1}) = (g_s^{sx_u/x_u}, h(w_1)^s \times \dots \times h(w_i)^s) = (g_s^{sx}, h(w_1)^s \times \dots \times h(w_i)^s)$. Send q and the keyword locations l_1, \dots, l_i to the data server.</p>
<p>Search($q, D,$ $l_1, \dots, l_i):$</p>	<p>DSSP: For every document $d_i' \in D$ test if the index matches $\frac{e_s(q_0, \prod_{n=1}^{n=i} I_n)}{e_s(I_0, q_1)} = \frac{e_s(g^{sx}, h(w)_i^r \times \dots \times h(w)_i^r)}{e_s(g^{rx}, h(w)_1^s \times \dots \times h(w)_i^s)} = 1$, add matching documents d_i' to D' and return the results.</p>
<p>Decrypt($D',$ $rk_u, u; sk_u):$</p>	<p>DSSP: Let $L_{mpk} = (L_{mpk,0}, L_{mpk,1}) = (g_p^v, KZ^{udm_1v})$. For every document $d_i \in D'$ compute $e_p(L_{mpk,0}, rk_u) = e_p(g_p^v, g_p^{udm_1u_2}) = Z^{udm_1u_2v}$ and re-encrypt all lockboxes $L_{i,pku} = (Z^{udm_1u_2v}, KZ^{udm_1v}) = (Z^{u_2v'}, KZ^{v'})$, with $udm_1v = v'$, and send D_u', the documents and the re-encrypted lockboxes, to user u. User: Let $L_{pk_u} = (L_{pk_u,0}, L_{pk_u,1}) = (Z^{u_2v'}, KZ^{v'})$. For every document $d_{u,i} \in D_u'$ the user decrypts the symmetric keys using the secret key $sk_u = (u_1, u_2)$ of his public key pair, $K = \frac{L_{pk_u,1}}{L_{pk_u,0}^{1/u_2}} = \frac{KZ^v}{(Z^{u_2v'})^{1/u_2}} = \frac{KZ^v}{Z^v}$ and decrypts the documents $d_i = D_K(E_K(d))$.</p>

Revoke(u):	User and data manager: The user and data manager deletes the re-encryption key rk_u and the complementary search key $CSK_u(u, rk_u, CSK_u)$ from the user table stored at the DSSP.
----------------	--

Figure 14: Attempt 2 – A conjunctive keyword searchable encryption scheme

In the previous section where attempt one was described a picture was given of the process from query construction to decryption. Below an illustration is given of the same process in attempt two. In comparison to attempt one, only the first two steps have changed. See below for a graphical representation from query construction to decryption in attempt two.

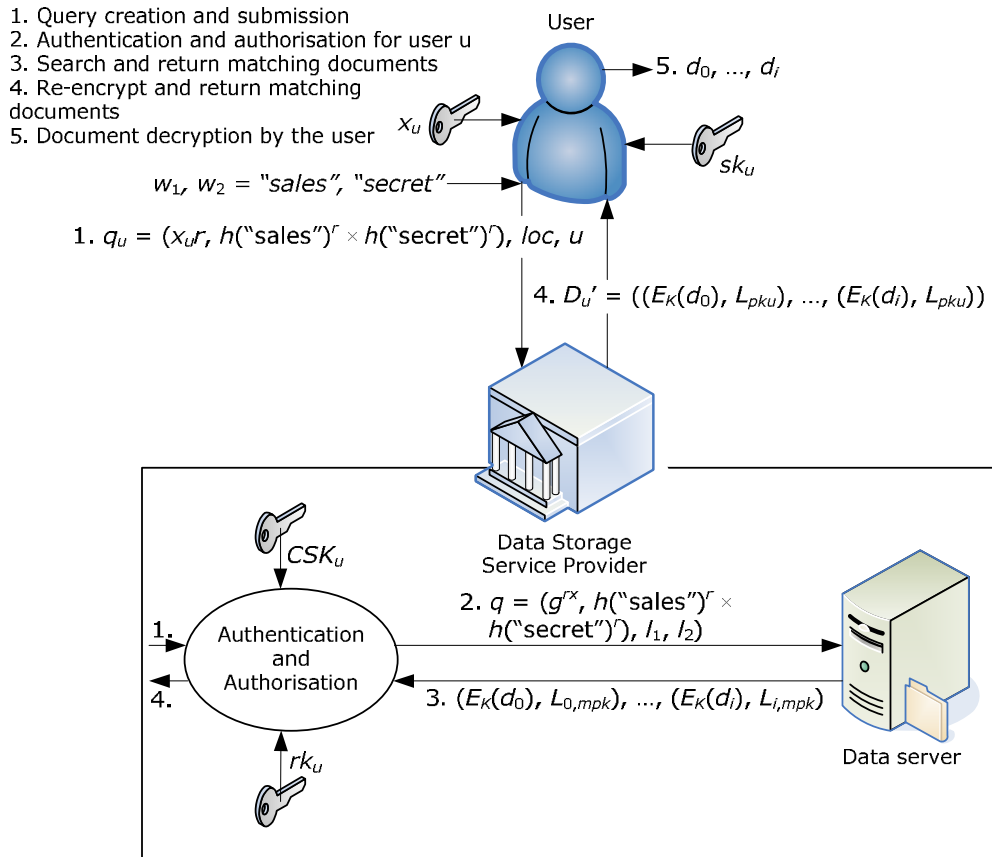


Figure 15: Attempt 2 - Query submission, searching and decryption

The first difference one might notice from the picture above when comparing with figure 12 is that in the figure above two keywords are queried instead of one. Another difference is that more information, the locations of the queried keywords, is submitted to the DSSP. In order to hide the keyword locations for eavesdroppers the location information is send encrypted. Blinding of the user’s secret search key x_u has not changed. The queried keywords, in this example “sales” and “secret”, are hashed, multiplied and blinded by an exponentiation with r instead of $1/r$ as in attempt one. The most important change regarding the system’s security is that all index terms are stored in probabilistic format and are thus no longer vulnerable to statistical attacks. During the query authentication and authorisation procedure only the user’s secret search key is removed. Step three and four, which involves decryption authorisation and decryption, have not changed in attempt two and are thus not described again. A description of these steps can be found in section 6.1.

6.2.1. Security Proof

In this section a formal security proof of the system constructed in attempt two is given. The system is proven secure under definition 5.3 given in chapter 5. The proofs given by Curtomla et al [11][10] and Bao et al [3] form the basis and are altered where necessary to match the system constructed in this thesis.

Proof. The idea of the proof is to describe a simulator \mathcal{S} that given the partial trace T_t^p can simulate the partial view of the adversary $\mathcal{A}(V_t^p)$. For all $t \in \mathbb{N}$, all PPT adversaries \mathcal{A} , all functions f and all $0 \leq p \leq t$, PPT simulator \mathcal{S} should be able to simulate $\mathcal{A}(V_t^p)$ given T_t^p with probability negligible close to one. It is shown that for all $0 \leq p \leq t$, $\mathcal{S}(T_t^p)$ can generate a view V_t^{p*} which is computationally indistinguishable from V_t^p the actual view of the adversary \mathcal{A} .

In the construction of the second attempt every document has its own index consisting of one field additional information and a number of keyword fields with hash values equal to the dictionary size. For sake of simplicity the encryption of a document and its lockbox is denoted by E_K . Recall the following notations given in section 5.3:

- $V_t = (d_1 = (E_K(d_1), I_1), \dots, d_n = (E_K(d_n), I_n), 1, \dots, n, CSK\text{-list}, Q_t, Loc_t, A_t)$
- $V_t^p = (d_1 = (E_K(d_1), I_1), \dots, d_n = (E_K(d_n), I_n), 1, \dots, n, CSK\text{-list}, Q_p, Loc_p, A_p)$, where $0 \leq p \leq t$
- $T_t = (|D| = n, 1, \dots, n, \Omega(q_1), \dots, \Omega(q_b, loc_i), |U_A|)$
- $T_t^p = (|D| = n, 1, \dots, n, \Omega(q_1), \dots, \Omega(q_p, loc_p), |U_A|)$, where $0 \leq p \leq t$

At time $p = 0$ before any queries are submitted ($Q_p = \emptyset, Loc_p = \emptyset, A_p = \emptyset$) the simulator \mathcal{S} must simulate the partial view V_t^{0*} from the partial trace T_t^{0*} . Simulator \mathcal{S} builds the view V_t^{0*} as follows. For $1 \leq i \leq n$, it selects $E_K(d_i) \in_R \{0,1\}^{E_K(d_i)}$ and for $I_{i,0}^* = g_s^{rx^*} = g_s^{z_i}$, where $z_i \in_R \mathbb{Z}_q^*$. The remaining index terms $I_{i,1}, \dots, I_{i,m}$, where m denotes the number of keywords in the dictionary are constructed as follows. For $1 \leq i \leq n$ and $1 \leq j \leq m$ compute $I_{i,j}^* = h(p_{i,j})^{z_i}$, where $p \in_R \mathbb{Z}_q^*$ is random number, random keywords could be used as well. The $CSK\text{-list}^*$ is constructed by selecting random elements from $G_{s,1}$ for every authorised user u enrolled to the system. One can see that the indistinguishability of the simulated view V_t^{0*} is guaranteed if the encryption function E is pseudorandom. It is easy to see that also the index terms and the $CSK\text{-list}^*$ are indistinguishable since they are blinded with a random z_i or randomly selected.

For $1 \leq p \leq t$, the simulator \mathcal{S} includes the partial view V_t^{0*} and builds $V_t^{p*} = (d_1^* = (E_K(d_1)^*, I_1^*), \dots, d_n^* = (E_K(d_n)^*, I_n^*), CSK\text{-list}^*, 1, \dots, n, Q_t^*, Loc_t^*, A_t^*)$. Recall that the trace T_t^p includes the search pattern over p queries. It remains to show the construction of queries (q_1^*, \dots, q_p^*) , their corresponding location information $(loc_1^*, \dots, loc_p^*)$ and their replies (a_1^*, \dots, a_p^*) included in $(V_t^p)^*$. For every authorised user $1 \leq j \leq |U_A|$ \mathcal{S} selects $x_j^* \in_R \mathbb{Z}_q^*$, the secret search keys. The simulator \mathcal{S} reuses the queries $(q_1^*, \dots, q_{p-1}^*)$ and location information $(loc_1^*, \dots, loc_{p-1}^*)$ included in $(V_t^{p-1})^*$, where it is assumed that \mathcal{S} remembers the queries, locations and their replies. Simulator \mathcal{S} starts checking the trace $T_t^{p-1}(\Omega(q_1, loc_1), \dots, \Omega(q_{p-1}, loc_{p-1}))$ if it contains the queried value w_t . If there does not exist a $\Omega(q_j, loc_j) = \Omega(q_p, loc_p)$ (query for the same keyword value) select a element from $x_1, \dots, x_{|U_A|}$ and compute $q_t = (rx_i, h(w_t)^r)$ and encrypt random keyword locations $loc_t = E_{pk_{DSSP}}(l)$. Otherwise it reuses the q_j and loc_t associated to w_t and assigns it to q_p . The simulated queries q_1^*, \dots, q_p^* and locations loc_1^*, \dots, loc_p^* in $(V_t^p)^*$ are indistinguishable from the queries q_1, \dots, q_p and locations loc_1, \dots, loc_p , otherwise one could distinguish between the blinded values and random values of the same size. The partial simulated partial view $(V_t^p)^*$ ($\Pr[\mathcal{S}(T_t^p) = f(D, W_p)]$) is thus indistinguishable from the actual view V_t^p ($\Pr[\mathcal{A}(V_t^p) = f(D, W_p)]$), for all $0 \leq p \leq t$ and thus satisfies the security definition $|\Pr[\mathcal{A}(V_t^p) = f(D, W_p)] - \Pr[\mathcal{S}(T_t^p) = f(D, W_p)]| < 1/p(k)$.

6.2.2. Evaluation

In this section an evaluation is given of the second attempt based on the research goals and the scheme constructed during the first attempt. The security and functionality of the scheme has improved in several areas at the cost of a computationally more expensive search operation. In the last part of this section the scheme is compared with existing schemes based on the overview in table one.

From the security proof in the previous section one could already see that the system is secure. In comparison with the first scheme the indexes are stored in probabilistic format, it becomes infeasible to deduce any information from the indexes. Due to the random blinding factor the indexes are no longer vulnerable to statistical attacks. This construction thus satisfies the security goal.

Enrolment of new users and revocation of existing users has not changed in the second construction. The user management of the system remains thus efficiently manageable without the need for redistribution of keys and the manageability goal is thus satisfied.

The first attempt already satisfies the outsourcing aspects of the system and the second construction does as well. All tasks except those that can only be done by the users (e.g. query construction) and user and data management are outsourced to the DSSP.

The most notable functional improvement is that the second construction supports conjunctive keyword search. In order to support efficient conjunctive keyword matching the locations of the keywords in the index is specified. The conjunctive keyword search functionality is at the cost of a less efficient search operation. Authentication and authorisation is more efficient since only one exponentiation is performed instead of a pairing operation. In the first attempt only one equality check was needed to test a document for the queried keyword. The second attempt needs two pairing operations per document.

When comparing the complexity of the second attempt with the most efficient existing conjunctive keyword searchable scheme one can see that this construction is more efficient. In table 1, one can see that the solution by Hwang et al [18] is most efficient multi user conjunctive searchable encryption scheme. Their index size is large compared to other solutions since it stores an additional factor for every user. The index and query size in this construction is comparable with single user conjunctive searchable encryption schemes. Searching takes three pairing operations in the solution by Hwang et al [18] while the second attempt needs only two pairing operations per document. The second construction has thus more functionality than all other evaluated solutions while the system is more efficient than the scheme which has most similar functionality.

6.3. Extensions

The system as described in the previous section will work fine, but additional functionality will further enhance the system. In practical situations there can often be distinguished between different groups of users based on their function (role) or department. The first part of this section describes a construction to support different data access rights for groups of users.

Another feature that will further enhance the system's functionality is a dynamic dictionary. Over time the relevance of keywords in the dictionary may change. It is then useful if obsolete keywords can be removed and new keywords be added.

6.3.1. Role Based Access to Subgroups of Documents

Most business or even business units have groups of users with different roles and thus different rights. Support employees do for example not have as many access rights as the senior management does. Creating different databases which are only accessible for certain groups of users is very inefficient since a lot of documents need to be stored multiple times. To tackle this problem efficient constructions have to be found for both index construction and data encryption.

In the second attempt one could see that additional information containing the blinding factor is stored for every document. This additional information is stored in an encrypted format with the system's secret key. To create subgroups of documents a new key have to be created for every subgroup of documents. Matching is then made possible by computing and storing additional information with the group key for every group the document belongs to. The system's secret key and the group keys are stored at and managed by the user and data manager. Users store an additional key for every group they are member of. The DSSP stores the counterparts of these keys since the same authentication and authorisation mechanism is used as for indexes accessible for all users.

Encryption rights to subgroups of documents can efficiently be solved as well. Instead of creating a lockbox under the system master public key, lockboxes can be constructed under the master public key(s) of the group(s) the document belongs to. Of course, corresponding re-encryption keys for the authorised users have to be computed and stored at the DSSP as well. See the figure for a graphical overview of the data storage.

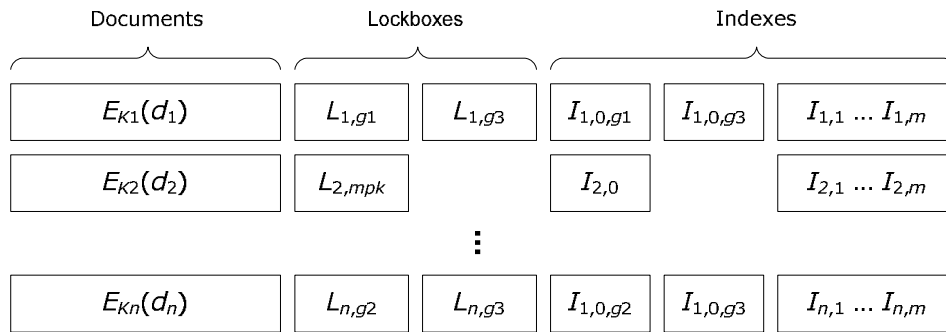


Figure 16: Document group storage

In the picture one can see that there are three different subgroups defined g_1 , g_2 and g_3 . Note that g_1 , g_2 and g_3 does not mean that the same key is used for indexes as for the lockboxes. This notation is chosen to clarify the picture. Not all documents need to belong to a subgroup. Documents not belonging to a subgroup, such as the second document in the picture, are stored unchanged. The first and last document in the picture both belong to two subgroups. They both contain the lockboxes and the additional information of the groups they belong to instead of the general lockbox and additional information.

6.3.2. Dynamic Dictionary

As stated in the introduction of this section over time some keywords may become obsolete and new keywords become relevant and have to be added. Removing old keywords or adding new keywords is a task of the user and data manager.

Removal of an existing keyword is simple since from every index only the keyword value at the location of the given keyword has to be removed. Of course, the keyword has to be removed from the dictionaries as well.

Adding a new keyword is slightly more complicated, but possible as well. First, the user and data manager has to find the blinding factor. Only the user and data manager should be able to find the blinding factor. When the blinding factor is recovered the new keyword can be hashed, blinded and added to the index. The user and data manager needs to perform this operation for every document. Similar as with keyword removal the dictionary has to be updated as well.

6.4. Threat Analysis and Measures

The system as described in this chapter will work well, but since security is an important requirement a threat analysis has to be performed. During the system's design several security measures to exclude different attack possibilities have already been taken. In the next section an overview of some threats to the system and pointers to possible solutions are given. Hereafter, in the next section some additional functionality is described which might contribute to the system's security on the long term.

6.4.1. Threat Analysis

In this section an overview is given of different attacks that could be targeted at the system. During the design, some measures have already been taken to prevent against certain attacks. The table below gives an overview of different attack possibilities and security measures that that could be taken to prevent from that particular attack.

Attack	Explanation and resolution
Statistical or dictionary attack on keywords	These attacks can be performed on both queries and indexes. By blinding the keywords in both queries and indexes this is resolved. This technique is already included in the system's design.
Dictionary including keyword locations becomes public	Since keyword locations are specified in the query the server is able to determine which keywords are queried. This is thus a serious leakage of information. To resolve this issue the user and data manger can reposition the keywords in both the dictionary as in the indexes. The user and data manager needs to re-blind the index terms so that the server cannot recognise the keywords as they were stored before.
User server collusion	This is a very serious security issue, the system is confiscated. No searchable encryption scheme is known that can prevent against this attack. The DSSP will be a reputable company that do not want to collude with users since they fear bad publicity which leads to leaving customers. The only solution is to re-encrypt all documents and reconstruct all indexes which can be done by the user and data manager and revoke the user from the system.

Attack	Explanation and resolution
Denial of service	Denial of service is trying to use a service which you are not authorised for and thereby harming the system. Due to the systems authentication and authorisation mechanism it is not capable of stopping all invalid requests. If the request has an invalid format the system will halt, but when it has the correct format it will continue its computations. Query submission by an unauthorised user will only lead to additional computation, the query will not match. An encrypted document with its indexes will be stored, but will never match due to the use of invalid keys. To prevent this, a traditional access control mechanism could be added.
Replay attack	With replay attacks previously eavesdropped information is submitted again. It can thus have the same consequences as a denial of service attack except that it tries to deduce information from the system instead of harming the systems operation. These kinds of attacks can be prevented by using secure communication channels.
Communication modification	Changed communications can have the same results as a denial of service attack except that this is usually executed to deduce information from the system instead of harming the system's operation. This attack can be prevented by using secure communication channels or by using a standard signing mechanism.

Table 2: Threat analysis

In table 2 one can see that for most threats an applicable measure can be found at the cost of additional overhead to the system. This overhead can be in terms of additional hardware, additional storage, additional computation or additional data management actions. In this thesis one of the goals is to outsource as much as securely possible and to use a cryptographic authentication and authorisation mechanism which can be used in an environment not fully trusted and not to use standard access control mechanisms. The use of secure communication channels is outside the scope of this assignment as well, but various alternatives are available e.g. SSL and TLS.

6.4.2. Additional Security Measures

In the previous section some of the additional measures to enhance or to contribute to the system's functionality were already introduced. Most of these measures are outside the scope of this assignment. Relocation of keywords, however, is one of the measures mentioned in the previous section and should be possible within the system.

To reduce the likelihood that the DSSP can deduce information from the system it is a good practice that the user and data manager shuffles the keyword locations once in a while. An approach is to send all users an updated dictionary with the new keyword locations and to make the same update on the server since they have the same order. Just changing keyword locations in the indexes will not make sense since all index terms will look exactly similar which enables the DSSP to obtain the old keyword locations. To make the index terms no longer recognisable they have to be re-randomised. Due to the blinding system used an exponentiation of all index terms with a new random number including the additional information will succeed.

Another approach is the user of a permutation function based on a secret key. A permutation function maps a keyword location to another location based on the key. By sending the authorised user a new

key for the permutation function the keyword locations are shuffled. Of course the keyword locations in the indexes have to be changed as well. This can be done by the same approach as described above.

6.5. Summary

In this chapter two different attempts to construct a conjunctive keyword searchable encryption scheme satisfying the research goals are described. The first attempt describes a scheme in which only one keyword can be attached to every document, but which has a very efficient search operation. Searching requires only one pairing operation to remove the user dependent factor and the blinding factor. Individual documents are then matched by a simple equality check. Although the keywords are encrypted they are not stored in probabilistic format.

The second construction improves the security and functionality of the first scheme at the cost of a computationally more expensive search operation. Conjunctive keywords search is now possible and every document can be associated with many keywords. The keywords in the indexes are now blinded and thus no longer vulnerable to statistical attacks. To prove the system's security a proof is given.

To further improve the system some additional functionality is described. The first additional feature makes it possible to define groups of users with certain access rights. A set of documents can only be made accessible for a certain group of employees, e.g. based on function profile. A second additional feature makes it possible to remove obsolete keywords from the dictionary and to add new ones.

The final section describes a more high level threat analysis. All threats can be resolved at the cost of additional overhead, most of which are outside the scope of this assignment. A feature that is within the scope is the ability to reshuffle keyword locations in order to minimise the amount of information the DSSP is able to learn.

Security and correctness is important, but practical performance is at least as important. For this reason the search construction of the second scheme is implemented and some speed measurements are performed to test its practical performance. The implementation and its results are described in the next chapter.

7.

Implementation

The previous chapter gave a detailed description of the system's construction and proved its theoretical security. This chapter evaluates the practical performance of the keyword search mechanism. First, some implementation details are given such as the programming language and the used cryptographic libraries. Hereafter, mathematical details about the curve parameters are given and their advantages and disadvantages evaluated.

When the details are clear a high level overview of the code to implement the keyword search part of the system is given. It shows the structure of the program and gives code snippets of both the most important function calls and selected parts of the functions itself.

Once all details are clear tests are performed regarding its practical performance in operations such as query construction, index generation and searching. These tests are performed to see if the system can be applied in practical applications. Also, it shows which parts are computational most expensive and where additional improvements are necessary. Finally, in the next section some pointers are given to further improve the system's speed and performance.

7.1. Programming Details and Cryptographic Libraries

In this section details about the code development such as programming language and used (cryptographic) libraries are discussed.

The code is written in C++ since it is known for its high performance which is important for the time measurements and its support for object oriented programming. To write code the free Integrated Development Environment (IDE) Netbeans [27] is used. The GNU GCC compiler [14] is used for compilation. Since the code is developed under Windows and the GNU GCC compiler requires a UNIX environment, MinGW with the command line interface MSYS [25] is installed.

The pairing operation is very complicated and computationally expensive and therefore the Pairing Based Crypto (PBC) library [23] is used. This library is programmed in C and needs the GNU Multi Precision (GMP) library [16] which is installed as well. For the hash algorithm the Crypto++ library [10], which contains various cryptographic algorithms, is used.

7.2. Pairing Parameters

The PBC library supports different types of pairings. Each type of pairings operates over a different elliptic curve. The different pairing types are labelled by alphabetic characters in order of discovery. In the library manual [24] one can see that type A pairings are the fastest and that type D pairings have the shortest element size. Both pairing types are suitable for cryptographic usage [24]. In this data storage outsourcing scene efficiency and thus search speed is an important aspect. Efficient data storage is important as well, but speed is considered more important and therefore type A pairings are chosen.

The PBC library includes pairing parameter files for every supported pairing type which are suitable for cryptographic use. Type A pairings operate over a super singular curve $y^2 = x^3 + x$. The parameters for type A pairings, which are used in this implementation, have the following properties.

- **Bit group order:** 160 bits. The group order r is a Solinas prime + 1. Solinas primes are chosen for their efficiency, they make the pairing computations faster.
- **Base field size:** 512 bits.
- **Embedding degree:** 2.
- **Dlog security:** 1024 bits.

Besides the above mentioned properties type A pairings have another unique property. With most pairing types one cannot mix elements from groups G_1 and G_2 , but this is possible for type A pairings since these groups are the same for type A pairings, type A pairings are symmetric.

Pairings are initialised by first opening a parameter file and then initialising the pairing with the initialised file pointer. Since the PBC library is written in C both the file opening and the pairing initialisation have a C syntax instead of C++ syntax. The above mentioned initialisation is illustrated in the figure below. Note that checks are omitted in the code snippet.

```
// open pairing parameters file
FILE *fp;
fp = fopen("a.param", "r");

// declare and initialise pairing
pairing_t pairing;
pairing_init_inp_str(pairing, fp);
```

Figure 17: Loading a parameter file and initialising a pairing

7.3. Implementation Overview

In this section an overview is given of the code design. The most important parts of the implementation are given and discussed.

The system deals with different actors who all have their own tasks. In chapter 3 the different actions and services have already been defined. These actors and the actions and services have been taken as a basis for the code development. There are separate classes defined for the user and data manager and for the users. The DSSP has been split in two classes, one for authentication and authorisation purposes and one for the data server which stores the documents and performs the search operations. As already mentioned only the search part of the algorithm is implemented and for this reason only the indexes without the documents itself are stored on the data server.

Implementation

When the program is started the different actors are initialised at first. The user and data manager takes the pairing parameters and the generator g as input and sets up the system. Hereafter the access control server at the DSSP performing authentication and authorisation is initialised. Then the data server is initialised and the users are created. New users are created by the user and data manager by passing the user id as a string and a reference of the access control server to store the complementary search key. Every user is added to the user collection. A user is retrieved by a request with the user identity (user name) to the user collection. Individual users are needed to perform the user actions such as index and query generation since they store the user specific secret search key. The discussed system initialisation procedure with the creation of two users is shown below in figure 17.

```
// initializing the user and data manager
userManager udm(pairing, g);

// initializing access control server
AccessControlServer acs(pairing);

// initializing data server
DataServer ds;

// create users
Users usrs(pairing);
udm.AddUser(pairing, string("user1"), usrs, acs);
udm.AddUser(pairing, string("user2"), usrs, acs);

User u1 = usrs.GetUser(string("user1"));
User u2 = usrs.GetUser(string("user2"));
```

Figure 18: System initialisation

After retrieving the individual users, the indexes or queries can be constructed. To create an index the matching keywords have to be specified. Relevant keywords are specified by their location in the dictionary and passed as an integer array. The number of keywords is specified as well. To compute a query the function `ComputeQuery` is called instead of `ComputeIndex`. The query computation function takes the same input arguments. In figure 18 the function call to compute an index is shown.

```
// specify the keywords in the dictionary
int kws[3] = {0, 1, 2};

// user1 computes an index
vector<basic_string<unsigned char> > index =u1.ComputeIndex(pairing,kws,3);
```

Figure 19: Index computation function call

The actual computations to construct an index are shown in figure 19. At first the required elements are initialised. There are four different element groups defined: the groups of elements which will be paired G_1 and G_2 , the group of paired elements G_T and the group of random numbers of order r Z_r . Initialisation only sets the structure for the elements and does not assign a value to it.

Computation of the additional information is simply done by generating a random number s (the blinding factor) and by multiplying this with the user's secret search key. The remaining index terms are generated by first checking if the keyword at the i th position is specified as a relevant keyword. If this is the case a hash value is calculated from the keyword, otherwise a random number is generated and hashed. To calculate the hash values the SHA1 algorithm, which is implemented in the Crypto++ library, is chosen. The resulting hash values are first mapped to an element of G_2 , in a deterministic manner, and than blinded by an exponentiation with s . Finally, the elements are converted to bytes and added to the index. The conversions from and to bytes are omitted in the code snippet in figure 19.

Implementation

```
// initialise elements
element_init_Zr(s, pairing); // the blinding factor
element_init_Zr(usk, pairing); // x_u
element_init_Zr(index_0, pairing); // the additional information
element_init_G2(index_n, pairing); // the index terms

// generate a random blinding factor
element_random(s);

// blind the user's key by multiplying it with the blinding factor
element_mul(index_0, s, usk);

// construct the remaining index terms
for(int i = 0; i < dictionary.size(); i++) {
    if (Search(keywords, size, i)) { // matching keyword
        output = new unsigned char[CRYPTOPP::SHA1::DIGESTSIZE];
        ComputeHash(output, dictionary.at(i));
    } else { // irrelevant keyword, hash a random nr
        char num[33];
        itoa(rand(), num, 10);
        output = new unsigned char[CRYPTOPP::SHA1::DIGESTSIZE];
        ComputeHash(output, num);
    }

    // map hash value into an element of G2
    element_from_hash(index_n, output, CRYPTOPP::SHA1::DIGESTSIZE);
    element_pow_zn(index_n, index_n, s);

    // after conversion to bytes (omitted) add the value to the index
    index.push_back(byte_element);
}
```

Figure 20: Index generation

Computations of queries are very similar; the additional information is calculated in the exact same way. The second query term is computed by first calculating the hash values of the specified keywords. These hash values are then multiplied and blinded by an exponentiation with the blinding factor.

The hashes produced with SHA1 have a length of 160 bits or 20 bytes. A representation of the hash value in G_2 converted to bytes has a length of 128 bytes. In chapter 4 it was mentioned that an empty word 2003 document is 19.968 bytes large which means that 155 keywords can be stored in the space required for an empty word 2003 document. The PBC library provides compression functions to reduce the required storage space. Before the data is actually compressed the length of the compressed representation can be requested. The size of a compressed element is 65 bytes, which is almost half its size. Unfortunately, the compression function has not yet been implemented for type A pairings.

Once the queries and indexes are computed they can be submitted to the DSSP who first performs the authentication and authorisation computations. After authentication and authorisation the document or in this case only the index can be stored on the data server. The authentication and authorisation mechanism for indexes and queries is exactly the same. The function calls are given in figure 20.

```
// submit the document to the DSSP for authentication and authorisation
acs.Transform(index, "user1", pairing);

// store the document (index) on the data server
ds.StoreDocument(index);
```

Figure 21: Authentication and authorisation and document storage

Implementation

The authentication and authorisation computations are simple. First, the required elements are initialised where after the complementary search key belonging to the supplied user identity is found. The actual authentication and authorisation computation is than only involved with one exponentiation. Finally, the new additional information is converted back to bytes. A code snippet of the authentication and authorisation procedure is given in figure 21.

```
// initialise the required elements
element_init_G1(comp_sk, pairing); // the complementary search key
element_init_G1(t_input_0, pairing); // new additional information
element_init_Zr(input_0, pairing); // reconstructed additional information

// find and reconstruct the user's complementary search key
element_from_bytes(comp_sk, (unsigned char*)Credentials.find(user_id)-
>second.data());

// reconstruct the additional information
element_from_bytes(input_0, (unsigned char*)input.at(0).data());

// perform the authentication and authorisation computation
element_pow_zn(t_input_0, comp_sk, input_0);
```

Figure 22: Authentication and authorisation

Once a database of documents (in this case only indexes) is build up one can submit queries to search for documents. The query execution function call takes a query after authentication and authorisation, the pairing parameters, an integer array with the keyword positions in the index and the number of keywords in the query. The function call for query execution is given in figure 22.

```
// execute an authenticated and authorised query on the data server
vector<int> result = ds.ExecuteQuery(query, pairing, mkws, size);
```

Figure 23: Query execution function call

When a query is received by the data server the query elements are reconstructed at first. With the PBC library it is possible to pre-process elements from G_1 since a lot of computations are repeated if the same element is used multiple times. The additional information in the query is an element from G_1 which is used for every query. It makes thus sense to pre-process this information. The keyword(s) value is an element from G_2 , but since type A pairings are used in the test program this can also be treated as an element from G_1 and thus be pre-processed as well to gain even faster query matching. After pre-processing all documents (indexes) are iterated over. The additional information stored with the indexes is reconstructed and the specified locations in the indexes are reconstructed and multiplied. When all elements are in the correct format the pairing is applied. After the two pairing operations it is checked if both results are equal to determine if a matching document is found. A code snippet of the search operation, where only one element is pre-processed is given below in figure 23.

```
// reconstruct query elements
element_from_bytes(query0, (unsigned char*)query.at(0).data());
element_from_bytes(query1, (unsigned char*)query.at(1).data());

// pre-process the query0 element (additional information)
pairing_pp_init(pp_query0, query0, p);

// check all documents if it satisfies the query
vector<vector<basic_string<unsigned char> > >::iterator it;
int doc_nr(0);
for (it = Documents.begin(); it < Documents.end(); it++) {

    // reconstruct additional information
    element_from_bytes(index0, (unsigned char*)it->at(0).data());
```

```
// retrieve and multiply the specified keyword locations
for(int i = 0; i < kwsz; i++) {
    element_from_bytes(temp_i, (unsigned char*)it-
>at(keywords[i]+1).data());
    if (i == 0) element_set(index1, temp_i);
    else element_mul(index1, index1, temp_i);
}

// apply the pairings
pairing_apply(temp1, index0, query1, p);
pairing_pp_apply(temp2, index1, pp_query0);

// check for a matches and add matching documents to the results
if(element_cmp(temp1, temp2) == 0) result.push_back(doc_nr);

doc_nr++;
}
```

Figure 24: The search process

In the code snippet given in figure 23 the numbers of the documents (id's) are added to the results instead of the actual documents. This is done since only the search part of the system is implemented and there are only indexes stored at the data server. For the same reason the results are immediately returned to the user and no decryption authentication and authorisation (proxy re-encryption) is performed.

Since the library was written in a UNIX environment it requires some system components which are not available in MinGW. That random numbers that are generated, used for the keys and the blinding factors, use the `/dev/urandom` system component. Because this component is not available in MinGW a deterministic number generator is used instead. Since the implementation is only to test the practical performance this is not considered a problem.

7.4. Experimental Results

In this section the test results of the system's practical performance are described. The system has been decomposed in different actions whose performance is tested individually. At first client side computations are considered. Index generation is discussed for several different index lengths and with different numbers of specified keywords. Query generation performance is tested for different amounts of specified keywords as well. Hereafter, the server side computations are considered. At first the authentication and authorisation computations are timed. The search operation is tested for different database sizes (number of stored documents) and with different queries. For the search tests the timings are given with and without pre-processing of the query elements. Finally, the correctness of the search algorithm is tested.

As already mentioned in the beginning of this chapter the program is developed under Windows and for this reason test will be performed under Windows as well. The used laptop has an Intel® Core™ 2 T5600 CPU which runs at 1.83 GHz and has 2 GB RAM.

The PBC comes with benchmark programs to test its performance for the different pairing parameters on the user's machine. After running several tests the average pairing time is about 9 ms without pre-processing and about 4 ms after pre-processing. The used timing function is not very accurate and results vary a lot. For the tests in this section the `QueryPerformanceCounter`, which has a much higher timing resolution, is used. The actual resolution depends on the system's hardware. In the code snippet in figure 24 one can see how the query performance counter is used.

Implementation

```
// initialise timing variables
LARGE_INTEGER lFreq;
LARGE_INTEGER lStart, lEnd;
double duration;

// get the high resolution counter's accuracy
QueryPerformanceFrequency(&lFreq);

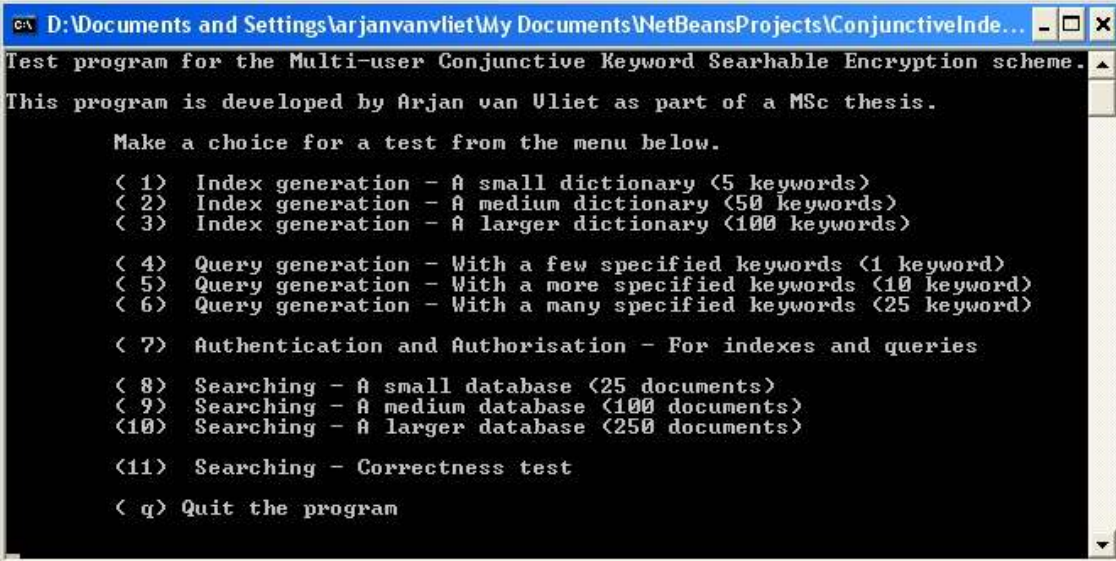
// start and stop the counter before and after code execution
QueryPerformanceCounter(&lStart);
/*
 * the code which has to be timed
 */
QueryPerformanceCounter(&lEnd);

// calculate the duration in ms
duration = (double)(lEnd.QuadPart - lStart.QuadPart) / lFreq.QuadPart * 1000;
```

Figure 25: Query performance counter

Since the resolution of the query performance counter is hardware depend the resolution on the current system is asked first. Just before the code of interest is executed the counter reads the current value of the system's counter. Right after execution of the interesting code the value of the system's counter is read again. The duration is now determined by calculating the difference of the counter values which is divided by the system's resolution. Since the result is in microseconds it is multiplied by 1000 to get the duration in milliseconds.

When the program is started one can choose one of the several tests from the menu. There are test specified to test the duration of index generation, query generation, authentication and authorisation and searching. Besides the duration tests there also is a test defined the test the correctness of search algorithm. In figure 25 one can see the test program and the test cases.



```
Test program for the Multi-user Conjunctive Keyword Searchable Encryption scheme.
This program is developed by Arjan van Uliet as part of a MSc thesis.

Make a choice for a test from the menu below.

< 1> Index generation - A small dictionary (5 keywords)
< 2> Index generation - A medium dictionary (50 keywords)
< 3> Index generation - A larger dictionary (100 keywords)

< 4> Query generation - With a few specified keywords (1 keyword)
< 5> Query generation - With a more specified keywords (10 keyword)
< 6> Query generation - With a many specified keywords (25 keyword)

< 7> Authentication and Authorisation - For indexes and queries

< 8> Searching - A small database (25 documents)
< 9> Searching - A medium database (100 documents)
<10> Searching - A larger database (250 documents)

<11> Searching - Correctness test

< q> Quit the program
```

Figure 26: Test program

Every test listed in the menu one can see in figure 25 starts with a short explanation about the tests that will be performed and start after a key stroke. The first tests measure the index generation duration for different dictionary sizes. In the first test the dictionary contains only five keywords, where the second and the third test have fifty and hundred keyword dictionaries respectively. All three

Implementation

cases measure the index generation duration with different amounts of specified keywords to see if one can see a time difference in hashing a keyword or generating and hashing a random number.

Dictionary size	Keywords specified	Duration	Duration per keyword
5 keywords	1 keyword	129.8 ms	26.0 ms
	4 keywords	129.8 ms	26.0 ms
50 keywords	1 keyword	1301.2 ms	26.0 ms
	10 keywords	1300.2 ms	26.0 ms
	25 keywords	1300.8 ms	26.0 ms
100 keywords	1 keyword	2600.7 ms	26.0 ms
	10 keywords	2602.2 ms	26.0 ms
	25 keywords	2600.1 ms	26.0 ms

Table 3: Index generation duration

For all eight test cases shown in table 3, there are five distinct users each generating two different indexes. Also, every test has been performed ten times which means that the test results in the table are the average values over 100 executions. From the results one can see that the amount of specified keywords does not influence index generation duration. Every index starts with additional information, the user's secret search key multiplied with a random number, followed by the blinded hash values. Since the average time per keyword is 26 ms in all cases the duration to calculate the additional information is considered negligible. When comparing the index generation duration for different amounts of specified keywords one can see that this does not influence the duration.

Besides indexes users generate queries as well. For this test a dictionary with fifty keywords is used of which one, ten or twenty-five keywords are specified for query generation. Similar as with the index generation tests there are five different users each generating two queries for all three test cases. The results shown in table 4 are the averages of 100 executions, ten tests where ten queries are generated.

Number of queried keywords	Duration	Duration per keyword
1 keyword	26.0 ms	26.0 ms
10 keywords	189.5 ms	19.0 ms
25 keywords	462.1 ms	18.5 ms

Table 4: Query generation duration

Since queries have the same structure as indexes with the difference that only hash values are calculated from the keywords that are of interest. For this reason and that hashing random numbers does not take more time similar results are expected as for index generation. From table 4 one can see that query generation with one specified keyword has the duration as expected. When more keywords are specified, query computation becomes more efficient as the duration per keyword decreases. This is because the most expensive operation, blinding (an exponentiation), is performed only once. The hashed keywords are first multiplied before the resulting value is blinded.

Once an index or query is generated it is submitted to the DSSP for authentication and authorisation. The procedure for indexes and queries is exactly the same. In both cases the only the first element, the additional information, is taken and thus authentication and authorisation is independent on the dictionary size or the amount of specified keywords. Similarly as in previous tests, there are five different users each generating two indexes and two queries. Also, the tests are performed ten times which means that both hundred indexes and queries are submitted to the DSSP. The results show that the duration for index and query authentication and authorisation does not differ, which is as expected. Authentication and authorisation consists of one exponentiation and takes only 7.9 ms.

The most interesting measurement is probably the system's search speed. Before the tests can be performed a database has to be set up. In this test a dictionary which contains 50 keywords is used. Constructing a database which contains 250 documents (indexes) takes more than five minutes. For

this reason the databases are constructed only once and stored in a file. In this way the database of interest can be loaded from a file at the beginning of a test. The queries are generated again for every test. Similar as in previous tests there are five users each generating two queries for every database size test and for every amount of specified keywords. To test the search speed three different databases have been defined containing twenty-five, 100 and 250 documents. Also, measurements are taken so see if the amount of specified keywords influences the search duration. To show the gains of pre-processing, all queries are executed without pre-processing, with one element pre-processed and with two elements pre-processed. Figure 26 shows the results of different search duration measurements.

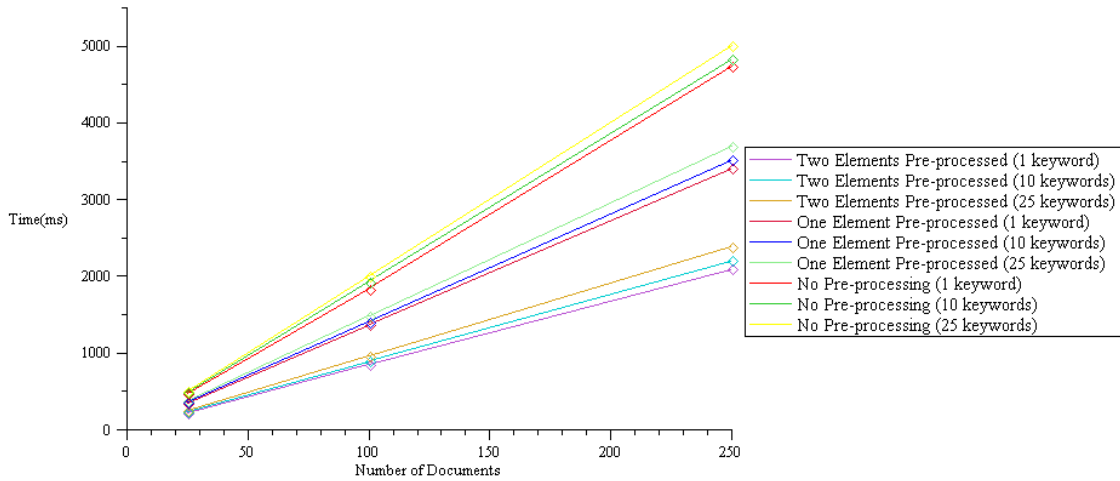


Figure 27: Search speed

The first things one might see when observing figure 26 is that the search duration linearly depends on the number of documents and that the duration of the search operation can roughly be halved when both query elements are pre-processed. Queries that contain multiple keywords are only slightly slower than queries with only one keyword; this is due to multiplication of the index terms. To get an even better impression of the system’s performance the duration per document and the number of documents that can be matched per second for different database sizes and queries with two pre-processed elements are given in table 5.

Database size	Queried Keywords	Duration per document	Documents per second
25 documents	1 keyword	9.0 ms	111.6
	10 keywords	9.4 ms	106.3
	25 keywords	10.1 ms	98.8
100 documents	1 keyword	8.5 ms	117.6
	10 keywords	8.9 ms	111.9
	25 keywords	9.7 ms	103.3
250 documents	1 keyword	8.4 ms	119.2
	10 keywords	8.8 ms	113.2
	25 keywords	9.6 ms	104.6

Table 5: Search speed with two pre-processed elements

From table 5 one can see that more than hundred documents can be matched per second and that the search speed slightly increases with larger database sizes. This slight performance increase is because the time needed to pre-process the two elements can be divided over more documents. Since the search speed is (almost) linear to the number of documents it is possible to determine the amount of time needed for a huge database which contains 10,000 documents. Let’s use the time per document in the case where ten keywords are queried. The duration to query a large database containing 10,000

documents takes than about 88 seconds. In practical applications it is expected that the amount average amount of keywords will be less then ten such that the search duration will be slightly shorter.

In figure 26 and table 5 the actual search time is considered without query generation and authentication and authorisation. To get the time from specifying the keywords till receiving the results in an actual implementation, query generation, authentication and authorisation and network trip time has to be included as well. Marshalling and un-marshalling (converting to bytes and recovering) time do not have to be added since the elements are already stored in and reconstructed from byte format.

In the paper by Ateniese et al [1] one can see that proxy re-encryption takes 22.0 ms on a machine with an AMD Athlon 2100+ 1.8 GHz processor. They have used a different cryptographic library to implement their pairing operation which means that the results might differ from a version implemented with the PBC library. The proxy re-encryption time is not considered a big problem since normally only a small subset of the documents stored at the DSSP will be returned and each document can be returned when finished re-encryption. A user will need much more time to evaluate a document than time is needed to re-encrypt another document.

An efficient and fast search algorithm is important, but correct results are even more important and for this reason a correctness test is performed. In the correctness test a database of 100 documents (indexes) is used. There are twenty distinct indexes specified and all five users generate and submit an index from these index specifications to come to a total of 100 documents. Index specifications defined the matching keywords from the dictionary. Every index has ten specified keywords. To test the correctness a total of twenty query specifications have been defined, containing one, two, three, four or five distinct keywords. Three different users generate queries from every query specification. Every query is tested without pre-processing and with one and with two pre-processed elements. This means that every query is executed three times for all three users. In figure 27 one can see a part of the output of the correctness test.

```

D:\Documents and Settings\arjanvanvliet\My Documents\NetBeansProjects\ConjunctiveInde...
User 3 1e Pre: 2 7 22 27 42 47 62 67 82 87
User 3 2e Pre: 2 7 22 27 42 47 62 67 82 87

Qry16 exp rslt: 4 10 11 24 30 31 44 50 51 64 70 71 84 90 91
User 1 No Pre: 4 10 11 24 30 31 44 50 51 64 70 71 84 90 91
User 1 1e Pre: 4 10 11 24 30 31 44 50 51 64 70 71 84 90 91
User 1 2e Pre: 4 10 11 24 30 31 44 50 51 64 70 71 84 90 91
User 2 No Pre: 4 10 11 24 30 31 44 50 51 64 70 71 84 90 91
User 2 1e Pre: 4 10 11 24 30 31 44 50 51 64 70 71 84 90 91
User 2 2e Pre: 4 10 11 24 30 31 44 50 51 64 70 71 84 90 91
User 3 No Pre: 4 10 11 24 30 31 44 50 51 64 70 71 84 90 91
User 3 1e Pre: 4 10 11 24 30 31 44 50 51 64 70 71 84 90 91
User 3 2e Pre: 4 10 11 24 30 31 44 50 51 64 70 71 84 90 91

Qry17 exp rslt: 0 14 20 34 40 54 60 74 80 94
User 1 No Pre: 0 14 20 34 40 54 60 74 80 94
User 1 1e Pre: 0 14 20 34 40 54 60 74 80 94
User 1 2e Pre: 0 14 20 34 40 54 60 74 80 94
User 2 No Pre: 0 14 20 34 40 54 60 74 80 94
User 2 1e Pre: 0 14 20 34 40 54 60 74 80 94
User 2 2e Pre: 0 14 20 34 40 54 60 74 80 94
User 3 No Pre: 0 14 20 34 40 54 60 74 80 94
User 3 1e Pre: 0 14 20 34 40 54 60 74 80 94
User 3 2e Pre: 0 14 20 34 40 54 60 74 80 94
    
```

Figure 28: Output correctness test

In figure 27 one can see that the expected result for every query is printed first. Hereafter the results of the nine query executions are printed. From the twenty distinct queries that are tested all results are as expected. Due multiplication one might come to a term which is already available in the index, but this has not happened. The correctness test is thus successfully passed.

7.5. Pointers to Further Improvements and Optimisation

In the previous section when the system's practical performance was discussed one could see that querying a very large database containing 10,000 documents takes about 88 seconds. This duration is at least very inconvenient for a user. This section discusses several alternatives to increase the system's performance.

Searching is an operation which can easily be executed parallel since every document is matched individually. By parallelising the search operation huge improvements on the search time can be achieved. Taking advantage of parallelising can be achieved in several ways. The most intuitive way is to divide the database and thus the search operation over several servers. Advantages in new hardware such as multi-core processors which allow for parallel execution can also greatly improve the search speed. The latest Intel CPUs have six cores, but more cores can be expected since modern GPUs (Graphical Processing Unit) by NVIDIA and Radeon already have ten cores [26]. With a combination of the above two techniques by dividing over different servers and by using new (faster) hardware it should be possible to increase the search speed by a factor ten.

Besides parallelising and using newer hardware one could also design or buy dedicated hardware. The pairing operations make use of elliptic curves for which dedicated hardware is available. Shi et al stated in their paper [29] that the Elliptic Semiconductor CLP-17 could reduce the exponentiation time from 6.4 ms (a pairing without pre-processing takes 5.5 ms on their machine) to 30 μ s.

Research on elliptic curves and optimising pairing operations have already decreased the pairing time from several minutes to several milliseconds [22]. Future research might be able to further reduce the pairing time. Research in hardware acceleration of the pairing operation, which is subject to academic research as well, will also contribute to a further speed improvement.

Besides performance increase in the search operation it is possible to increase the index generation performance as well. The index consists of blinded hash values of the specified keywords and blinded hash values of a random number when the keyword at the position does not match. When hashing a random number which is deterministically mapped in to an element of G_2 a new random number is obtained, this random element in G_2 is than randomised again by blinding. The result of this, timely, sequence of operations only leads to a random element in G_2 . By immediately drawing a random element from G_2 , which should be indistinguishable from elements obtained by the current approach, the index generation performance can be greatly increased.

7.6. Summary

In this chapter the implementation of the search algorithm discussed in the previous chapter is described. At first details about the Windows programming environment, the GCC compiler, MinGW and the used libraries are discussed. For the pairing operations the Pairing Based Crypto (PBC) library is used and the Crypto++ library is used for the hash functionality.

After discussing the programming environment and the cryptographic libraries the pairing parameters are discussed. The PBC library has some predefined pairing parameters of which the type A pairing parameters are used. Type A pairings have the fastest pairing operation and have the advantage that elements from G_1 and G_2 are the same group which means that all elements can be pre-processed to increase the search speed.

In the implementation overview code snippets of the most important operations are given and discussed. Also, the Query Performance Counter function used to measure the duration of the

different operations is discussed. Since the indexes add overhead the amount of overhead is measured as well. Every index term takes 128 bytes which means that approximately 150 keywords can be stored to equal the size of an empty word 2003 document.

The computation of indexes is dependent on the size of the dictionary, but independent on the amount of specified keywords, it takes about 26 ms per keyword. Query generation is dependent on the amount of specified keywords, but faster than 26 ms per keywords since blinding is done at the end instead of for every document as in index generation. Authentication and authorisation does not take much time and takes about 7.9 ms.

The speed of the search operation is the most interesting measurement. Search time is measured for different database sizes, with different amounts of specified keywords and without and with pre-processing. When both query elements are pre-processed the search duration is roughly halved and about 9 ms per document. The number of queried keywords leads to a very little time overhead. Querying twenty-five keywords instead of one takes only 1 ms additional time per document.

In a final implementation test the correctness of the search algorithm is tested for different amounts of specified keywords and with and without pre-processing. The results were always as expected which meant the system functions correct.

In the final section some pointers are given to further improve the system's speed. Parallel execution such as dividing the database over servers can greatly increase the search time. Recent developments on the CPU market show that processors contain more cores and are thus able to execute operations parallel as well. Also, specified hardware can be used to increase the search speed.

The next chapter gives the conclusion of this thesis. It mentions the contributions to literature and evaluates the results of this thesis and gives some directions to future research.

8.

Conclusion and Future Work

In this chapter the results of this thesis are evaluated and discussed. The first section describes the scientific contribution. Hereafter, in the second section, examples of other applications for searchable encryption are shortly mentioned such as encrypted e-mail forwarding. Other applications are given to get an idea of the relevance and applicability of searchable encryption.

In the conclusion section the developed system is evaluated based on the research goals as described in section 1.2 and the requirements defined in section 3.3. It summarises both the achieved theoretical and practical results and describes that there are two new schemes are developed of which the second satisfies all requirements. The implementation of the second scheme shows that it is applicable in small scale databases or where high security requirements are needed.

The final section gives some directions for future research to further improve the system as it was developed in this work.

8.1. Scientific Contribution

Several attempts to construct a searchable encryption scheme have already been made. However, none of these did completely satisfy the research goals and requirement as posed for this thesis. In this thesis several contributions have been made in order to satisfy the research goals and requirements posed for this thesis.

Before developing the actual algorithms, first a high level scene overview (general framework) was constructed to identify important actions and services needed for a successful data storage outsourcing scene. Literature has described keyword searchable encryption solutions, but has failed to describe a general framework for data storage outsourcing including data and service management operations and multi-user data encryption mechanisms. In this thesis a clear overview of the different actions and services building blocks is given. Each of the building blocks is described and has its different alternatives evaluated and discussed.

In this thesis two different searchable encryption schemes have been developed. The first is a very efficient single keyword searchable encryption scheme which is more efficient than comparable schemes. Searching is only based on equality checks. Drawback of this scheme is that the DSSP can observe which documents have the same keyword since these are stored in a probabilistic format.

The second scheme is the first true multi user conjunctive searchable encryption scheme described in literature. Other multi user conjunctive searchable encryption schemes are not truly multi-user since they face limitations such as that only one user is allowed to add documents and that users can not efficiently be revoked without key redistribution. The only other scheme that allows for efficient revocation supports only single keyword search.

Also, there are no existing multi-user searchable encryption schemes known that support multi-user data encryption. In this thesis this is achieved by combining the developed conjunctive searchable encryption scheme with an already existing proxy re-encryption scheme.

In practical applications it is often required that groups of documents can only be accessed by a limited group of users. There is no known multi-user conjunctive searchable encryption scheme that supports this feature. In this thesis an extension is described that supports Role Based Access Control (RBAC) to subgroups of documents with a very limited amount of overhead. This RBAC is available for both the search part and the decryption part of the system.

In literature there are implementations described of range queries search techniques, but the papers about keyword searchable encryption schemes read for this thesis did not describe an implementation. This work is thus one of the first that describe an implementation of a multi user conjunctive searchable encryption scheme.

8.2. Other Applications of Searchable Encryption

In this thesis a solution was developed for a secure data outsourcing application. For this reason it is not surprisingly that the developed solution fits best in a secure database or a secure data outsourcing setting. There are, however, other applications of searchable encryption that can be thought of. Since searchable encryption is enabled by adding indexes to the data or by encrypting the data in a special format, it can only work if all involved parties agree on the structure and protocol. First, some applications of keywords searchable encryption are discussed; where after applications of range query techniques are discussed. Most of the applications given in this section can also be found in the literature read for this thesis.

Keyword searchable encryption can be applied in an e-mail forwarding setting. Based on, for example, the subject of an e-mail it can be forwarded to a certain person. This can be used in combination with proxy re-encryption. All e-mails should be sent to a company in encrypted format under the company's master public key. When the e-mail server receives the message it first checks to whom the e-mail should be forwarded, based on keyword searching, and then re-encrypts the message such that only the intended recipient can decrypt the message. With this application a company can ensure that message is delivered to the right person in a secure way without having to publish his or e-mail addresses of individual employees. Encrypted e-mail forwarding can also be used on a temporary basis when an employee is on holiday and another employee should only be able to read selected messages for example those that are marked with high priority.

Range queries can be applied to filter financial transactions or to selectively reveal parts of network or financial audit logs. In contrast to keyword searchable encryption schemes range queries (or trapdoors) are constructed to reveal the data when the query matches. With this construction one can reveal a specified amount of information. When applied to filter a stream of financial transactions it can be enabled to decrypt all transactions with a value of \$1000 or more for further inspection. Another application is to selectively reveal parts of a network or financial audit log. For example, in case of a virus outbreak one can create a trapdoor that gives access to only the relevant parts of the network audit log.

8.3. Conclusion

In the introduction of this thesis, in section 1.2, the research goal to develop a secure and efficient data storage system was introduced. To support this goal a set of sub goals covering the system's security and confidentiality, the system's manageability and the outsourcing aspect of the system have been defined in the same section. In section 3.3 these research goals were formalised and a set of requirements including functionality and efficiency requirements was listed. This section gives a final conclusion and evaluates the developed system based on the requirements.

In this thesis report one could have seen that it is possible to successfully and securely outsource data storage with conjunctive keyword search capabilities. The system supports multiple users and has efficient user management. Also, it is possible to efficiently extend the system to support practical requirements such as RBAC. Although the search time can be the bottleneck on large systems it is still applicable for relatively small databases or for databases where security is more important than search time. Also, there are several possibilities to optimise the search time to increase its practical application.

The security and data confidentiality related requirements require that data confidentiality must be ensured. To ensure this the data is stored in encrypted format under the master public key owned by the user and data manager. The encrypted data is only available to authorised users and only after re-encryption by the DSSP. Since the plaintext does not become visible during re-encryption, data confidentiality is still ensured. Submitting queries inevitably reveals some information to the server, but this information has to be minimised to the query itself and its outcome. The DSSP should not be able to determine which keywords are queried. In the system in this thesis all index and query terms are randomised and therefore the DSSP is unable to determine the keywords. By performing some calculations it is only able to determine which documents (indexes) satisfy the query terms. In chapter 6 the system is proven theoretically secure as well. The developed system thus satisfies the security and confidentiality requirements.

Since the data storage outsourcing system deals with a constantly changing set of users and data it must be able to efficiently manage both users and the stored data. The developed system allows for efficient user management due to the construction that it needs input from both the user and the DSSP for an action without having the users share common secret keys. A new user can be enrolled by generating new key material for both the user and the DSSP. Revoking users is achieved by simply deleting the user's key material from the DSSP. The user and data manager is able to perform data management operations as well since it holds the master secret key under which all documents are encrypted. It is also able to add new keywords and to remove obsolete keywords from the dictionary. The manageability requirements are thus met by the developed system.

Since this system deals with data storage outsourcing it is clear that it requires that the actual data storage is outsourced. This requirement is of course satisfied. The other outsourcing requirement tells that as much tasks as possible needs to be outsourced as long as this does not harm the system's security. In traditional authentication and authorisation mechanisms control of the server means access the system's complete functionality. Since the DSSP should not be able to use the system's functionality and gain access to the information it stores traditional authentication and authorisation can not be outsourced. In this system a different authentication and authorisation mechanism has been developed in which full control of the server concerned with this operation does not mean that it can use all of the system's functionality or access to the stored data. This construction can thus successfully be outsourced without harming the system's security. User specific operations such as index and query generation and encryption are not outsourced. Also, user and data management is not outsourced since this would mean that the server can use the system's complete functionality. The defined outsourcing requirements are thus successfully met.

One of the most important functional requirements is search functionality. Besides single keyword search, which is already available in multi user searchable encryption schemes, conjunctive (multiple) keyword search should be available as well. In the first scheme only single keyword search is available, but it is more efficient than the currently available schemes. Although its efficiency it might be vulnerable to statistical attacks since the indexes are not probabilistic. In the second scheme both deficiencies are addressed to come to a system which has both conjunctive keyword search support and probabilistic index storage. The system with most comparable functionality by Hwang et al [18] needs three pairing operations per document. In the second scheme developed in this thesis only two pairing operations per document are required. When comparing the index storage complexity one can see that most solutions need an amount of space equal to the amount of keywords and one or two terms of additional information for every document. This is similar to the amount of space requirement for the second construction where the amount of space per document is the number of keywords and one term of additional information. The search functionality as well as the decryption functionality is available for all users enrolled to the system. In the additional functionality an extension is described to support efficient Role Based Access Control (RBAC) to subgroups of documents. It is possible to define groups of documents which can only be searched and decrypted by users with a certain function or belonging to a specific group with very little overhead. Finally, the second construction of the system is implemented to verify the correctness of the system and to test its practical performance. All functionality and efficiency requirements are thus met.

The implementation results show the duration of different actions; index generation, query generation, authentication and authorisation and searching. One could see that index generation can consume quite some time. An index with 100 keywords takes about 2.6 s to be computed. In the pointers to further improvement one could see that it is possible to decrease this time such that it depends mainly on the number of specified keywords. Since index generation is something that is done only once per document, this is not considered to be a serious issue. Query generation on the other hand is much quicker. It depends on the number of specified keywords and takes less than half a second with twenty five specified keywords. Authentication and authorisation is the quickest action and takes only 7.9 ms and is independent of whether indexes or queries are considered. Querying the database is most important to be efficient and fast. Without pre-processing it takes almost five seconds to query a database with 250 documents. Fortunately, pre-processing is able to more than halve the search time to a little over two seconds. With a realistic amount of five specified keywords approximately 115 documents can be searched per second. While this might fulfil the requirements for small databases it is not fast enough for very large databases although further improvements and optimisations might make this possible as well. By serialising over multiple servers and by using modern multi core processors a huge performance gain could be achieved.

8.4. Future work

The system developed in this work is fully functional and can be applied in practical situations. This does, however, not mean that there are no interesting subjects for future research. In this section some directions to interesting future work or research are given.

An intuitive and straight forward direction to future research is further optimisation of the current scheme. Optimisation can be achieved by diving deep into mathematics to find additional optimisations or new and faster curve types for the pairing operation. Developing dedicated pairing hardware will decrease the required computational time as well. Also, one can try to find a different index construction which enables more efficient matching. Another improvement would be to find an efficient matching technique that does not require specification of index locations.

When searching for a document one is often only interested in the latest submitted documents. Also, it might be possible that multiple versions of the same or more or less the same document are stored at

Conclusion and Future Work

the file repository or in a database. In such cases it will be beneficial if the submission date can be included in the search terms. For some applications it might be possible to take the date the document is submitted to the DSSP while for others custom dates might be required. An extension which adds the possibility to search on dates is thus a valuable extension.

The way in which the system is constructed in this thesis the full documents always have to be decrypted to see the actual contents. It might save time when an overview of short abstracts is returned first. From this overview the user can request only the relevant documents. An approach that can be taken is to include a short abstract of every document encrypted with a homomorphic encryption scheme. The abstracts can then be added to each other such that it looks as one piece. In the ideal case the homomorphic encryption scheme should also support proxy re-encryption.

A more advanced direction for future research is to support semantic searching. It might be possible that the keywords in the dictionary can be divided in groups. For example, the general keyword 'finance' might cover the more specific keywords 'annual report' and 'revenue'. It will be an interesting future research subject to see if hierarchical solutions can support for faster course grained and slower fine grained search. Also it might be interesting to see whether this can contribute to more efficient index storage.

Bibliography

- [1] G. Ateniese, K. Fu, M. Green, S. Honenberger. Improved Proxy Re-encryption Schemes with Applications to Secure Distributed Storage. *ACM Transactions on Information and System Security*, Vol. 9, No. 1, Pages 1-30. February 2006.
- [2] L. Ballard, S. Kamara, F. Monrose. Achieving Efficient Conjunctive Keyword Searches over Encrypted Data. *ICICS, LNCS 3783*, pp 414-426, Springer-Verlag. 2005
- [3] F. Bao, R. Deng, X. Ding, Y. Yang. Private Query on Encrypted Data in Multi-user Setting. *ISPEC 2008, LNCS Vol. 4991*, p. 71-85, Springer-Verlag. 2008.
- [4] M. Blaze, G. Bleumer, M. Strauss. Divertible Protocols and Atomic Proxy Cryptography. *Advances in Cryptology - Eurocrypt '98 LNCS*. 1998.
- [5] D. Boneh, Waters, B. Conjunctive, Subset, and Range Queries on Encrypted Data. *Theory of Cryptography Conference (TCC) LNCS 4392 Springer Verlag*. 2007
- [6] R. Brinkman. Searching in Encrypted Data. *CTIT Ph.D. Thesis Series No. 07-98*. 2007.
- [7] J. Byun, D. Lee, J. Lim. Efficient Conjunctive Keyword Search on Encrypted Data Storage System. *EuroPKI 2006, LNCS Vol. 4043*, pp. 184-1965, Springer-Verlag. 2006.
- [8] R. Canetti, S. Honenberger. Chosen-Ciphertext Secure Proxy Re-Encryption. *Proceedings of the 14th ACM Conference on Computer and Communications Security*. 2007.
- [9] Y. Chang, M. Mitzenmacher. Privacy Preserving Keyword Searches on Remote Encrypted Data. *In Proceedings of ACNS '05 LNCS Vol. 3531*, pp442-445, Springer-Verlag. 2005.
- [10] Crypto ++ Library 5.5.2. www.cryptopp.com. 2008
- [11] R. Curtmola, J. Garay, S. Kamara, R. Ostrovsky. Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions. 2006.
- [12] G. Danezis, C. Diaz, S. Faust, E. Käspar, C. Troncoso, B. Preneel. Efficient Negative Databases from Cryptographic Hash Functions. *ISC 2007, LNCS 4779*, pp. 423-436, Springer-Verlag. 2007
- [13] R. Dutta, R. Barua, P. Sarkar. Pairing-Based Cryptographic Protocols: A Survey. <http://eprint.iacr.org/2004/064.pdf>. 2004.
- [14] Free Software Foundation (FSF). GCC, The GNU compiler collection. gcc.gnu.org. 2008
- [15] Google. Google Scholar. <http://scholar.google.nl>. 2008, 2009.
- [16] GMP. The GNU Multi Precision (GMP) Bignum Library. <http://gmplib.org/>. 2008
- [17] J. Horwitz. A Survey of Broadcast Encryption. <http://math.scu.edu/~jhorwitz/pubs/broadcast.pdf>. 2003.
- [18] Y. H. Hwang, P. J. Lee. Public Key Encryption with Conjunctive Keyword Search and Its Extension to a Multi-user System. *Pairing 2007, LNCS 4575*, pp 2-22, Springer Verlag. 2007.
- [19] D.H. Lee, Y. J. Song, S. M. Lee, T. Y. Nam, J. S. Jang. How to Construct a New Encryption Scheme Supporting Range Queries on Encrypted Database. *2007 International Conference on Convergence Information Technology, IEEE Computer Society*. 2007.
- [20] B. Libert, D. Vergnaud. Unidirectional Chosen-Ciphertext Secure Proxy Re-encryption. *PCKS 2008, LNCS 4939*, pp. 360-379. 2008.
- [21] J. C. A. van der Lubbe. Basismethoden cryptografie. *Delftse Universitaire Pers*. 1997.
- [22] B. Lynn. On the Implementation of Pairing-Based Cryptosystems. *Phd dissertation, Stanford University*. 2008.
- [23] B. Lynn. Pairing Based Crypto (PBC) Library. crypto.stanford.edu/pbc. 2008
- [24] B. Lynn. PBC Library Manual 0.4.18. crypto.stanford.edu/pbc/manual.pdf. 2008
- [25] MinGW. Minimalist GNU for Windows. www.mingw.org. 2008
- [26] Multi-core. <http://en.wikipedia.org/wiki/Multi-core>. 2008
- [27] Netbeans. Netbeans 6.1 IDE. www.netbeans.org. 2008
- [28] EK. Ryu, T. Takagi. Efficient Conjunctive Keyword-Searchable Encryption. *21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07)*. 2007.

Bibliography

- [29] E. Shi, J. Bethencourt, T-H. H. Chan, D. Song, A. Perrig. Multi-Dimensional Range Query over Encrypted Data (full). <http://www.ece.cmu.edu/~dawnsong/papers/rangequery-full.pdf>. 2007.
- [30] D. Song, D. Wagner, A. Perrig. Practical Techniques for Searches on Encrypted Data. *IEEE*. 2000.
- [31] S. D. C. di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, P. Samarati. A Data Outsourcing Architecture Combining Cryptography and Access Control. *Proceedings of the 2007 ACM workshop on Computer security architecture (CSAW'07)*. 2007.
- [32] A. van Vliet. Data Leakage and its Prevention. Literature report, *TU Delft*. 2008.
- [33] Wikipedia: Size Comparisons, http://en.wikipedia.org/wiki/Wikipedia:Size_comparisons. 2008

Appendix A – Terms, Symbols and Notations

In the table below the terms, symbols and notations used in this report are summarised.

Notation	Explanation
$ · $	Length or size of group $·$, usually number of items in the group or list
$\Omega(q, loc) = \{u, id(a_q)\}$	Query and keyword locations with its submitter and search outcome
$A_t = (a_1, a_2, \dots, a_t)$	Sequence of t query replies
CSK	Complementary search key
CSK_u	Complementary search key for user u
$CSK-list$	List of CSK 's stored at the DSSP
D	Complete document collection
D'	Part of the document collection, e.g. query result
D'_u	Part of the document collection with re-encrypted lockboxes for user u
d	Plaintext document
d'	Encrypted document
DSSP	Data storage service provider
$h(\cdot)$	Hash function
I_i	The index of i 'th document
$id(d')$	Document identifier
K	Symmetric encryption key
k	Security parameter
L_{mpk}	Lockbox, symmetric key K encrypted with mpk
$Loc_t = (loc_1, loc_2, \dots, loc_t)$	Sequence of t keyword(s) locations in the index specification
mpk, msk	Master public key pair, master public key and master secret key
pk_u, sk_u	User public key pair, public key and secret key for user u
PPT	Probabilistic Polynomial Time
q	Query after authentication and authorisation
$Q_t = (q_1, q_2, \dots, q_t)$	Sequence of t queries
q_u	Query before authentication and authorisation issued by user u
RBAC	Role Based Access Control
rk_u	Re-encryption key for user u
T_t	Trace, all information that can be obtained from observing the access pattern
u	User identifier
U_A	List authorised of users
UDM	User and data manager
u_q	Query issuer
V_t	View of the DSSP, all information the DSSP has access to, the contents it stores and the access pattern
W	Dictionary
$W_t = (w_1, w_2, \dots, w_t)$	Sequence of t query words
x	Secret key of the user and data manager
x_u	Secret search key for user u