

Secure Electronic Report Submission

Marsibil Ingibjörg Hjaltalín

M.Sc. Thesis No. 18

2005
Computer systems engineering
Informatics and Mathematical Modelling
Technical University of Denmark

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk
IMM-THESIS 2005-18
Printed at IMM, DTU

Preface

This thesis is the final requirement for obtaining the degree Master of Science in Computer Systems Engineering. The work was carried out over a period of seven and a half months, from 16th of August to 31st of March. The work was carried out at the department of Informatics and Mathematical Modelling at the Technical University of Denmark. The project was supervised by Dr. Robin Sharp.

Acknowledgments

I would like to thank my supervisor Robin Sharp for his guidance, support and all the discussion we had during our meetings. I would like to thank Steen Pedersen and Christian Westrup Jensen for providing me with their intuitions regarding the project.

I would also like to thank Morten Johnstad-Møller and Rasmus Beck at Arcanic for giving me information about the CampusNet, and Jóhanna Guðrún Guðmundsdóttir for proofreading.

Finally I would like to thank my husband and my daughter for their support and encouragement.

Lyngby, March 31st 2005

Marsibil Ingibjörg Hjaltalín

Abstract

This master's thesis describes the process of creating a prototype of a Report and Exam Submission System, where students can submit documents to teachers in a secure manner.

Students use digital signatures to sign report- and exam documents. The system allows multiple signatures, many students can sign the same document. The document can be encrypted if the content should be kept confidential. Teachers and external examiners can view documents and manage access the documents. They can also decrypt the documents if they have been encrypted.

The security aspects of such a system, like digital signatures, encryption and decryption were examined and are explained in the thesis. When creating a prototype the following software engineering parts are essential: analysis, design, implementation and testing. The analysis was done by identifying the requirements and then writing the use cases. The design part consists of describing the system architecture and designing the database. The prototype was thereafter implemented by using the analysis and the design descriptions.

Contents

- 1 Introduction** **3**

- 2 Security Analysis** **5**
 - 2.1 Security Problems 6
 - 2.2 Security of Paper Documents 6
 - 2.3 Security of Electronic Submission 7
 - 2.3.1 Secure exams 7
 - 2.4 Comparison 8
 - 2.5 Advantages of a System 8
 - 2.6 Summary 9

- 3 Applied Cryptography** **11**
 - 3.1 Encryption 11
 - 3.1.1 Secret Key Encryption 11
 - 3.1.2 Public Key Encryption 12
 - 3.2 Digital Signatures 12
 - 3.2.1 The Protocol 13
 - 3.2.2 Multiple Signature 14
 - 3.3 Certificates 15
 - 3.4 Summary 15

- 4 Requirement Analysis** **17**
 - 4.1 Requirements 17
 - 4.2 The Operations of the System 18
 - 4.3 Information from other Systems 20
 - 4.4 The Actors and their Operations 21
 - 4.5 Use Cases 22

| | | |
|--------|---|----|
| 4.5.1 | Log on | 22 |
| 4.5.2 | Log out | 23 |
| 4.5.3 | Document notification | 23 |
| 4.5.4 | List and select a course | 23 |
| 4.5.5 | List and select a signed file | 24 |
| 4.5.6 | Sign a file | 24 |
| 4.5.7 | Encrypt a signed file | 26 |
| 4.5.8 | Confirmation | 26 |
| 4.5.9 | View the details of a signed file | 27 |
| 4.5.10 | Delete a signed file | 27 |
| 4.5.11 | Submit a file with parallel multiple signatures | 27 |
| 4.5.12 | List accessible files in an e-box | 28 |
| 4.5.13 | View the details of a signed file | 28 |
| 4.5.14 | Insert a folder | 29 |
| 4.5.15 | Edit a folder | 29 |
| 4.5.16 | Move files between folders | 29 |
| 4.5.17 | View access to folders | 29 |
| 4.5.18 | Grant access to a folder | 30 |
| 4.5.19 | Revoke access to a folder | 30 |
| 4.5.20 | View my access to folders | 30 |
| 4.5.21 | Create a comment on submission | 30 |
| 4.5.22 | Delete a comment on submission | 31 |
| 4.5.23 | View a signed file | 31 |
| 4.5.24 | Decrypt a signed file | 31 |
| 4.5.25 | Download to local drive | 31 |
| 4.5.26 | Archive a signed file | 32 |
| 4.5.27 | Search for a file | 32 |
| 4.5.28 | Search for a text in the file database | 32 |
| 4.5.29 | View the course's assignments | 33 |
| 4.5.30 | Insert an assignment | 33 |
| 4.5.31 | Edit an assignment | 34 |
| 4.5.32 | Delete an assignment | 34 |
| 4.5.33 | View the transaction log | 35 |
| 4.6 | The Transaction Log | 35 |
| 4.6.1 | The Attributes | 36 |
| 4.7 | Summary | 37 |

| | | |
|----------|---|-----------|
| 5 | System Design | 39 |
| 5.1 | System Architecture | 39 |
| 5.1.1 | The Client | 39 |
| 5.1.2 | The Application Server | 40 |
| 5.1.3 | The Database Server | 40 |
| 5.2 | Development and Runtime Environment | 40 |
| 5.2.1 | Client's Requirements | 40 |
| 5.2.2 | Programming Language | 40 |
| 5.2.3 | Development Environment | 41 |
| 5.2.4 | Runtime Environment | 41 |
| 5.3 | User Interface | 41 |
| 6 | Database Design | 43 |
| 6.1 | The ER Model | 43 |
| 6.1.1 | The Entities | 44 |
| 6.1.2 | The Relations | 45 |
| 6.1.3 | The Attributes | 45 |
| 6.2 | The Relational Model | 46 |
| 6.3 | Summary | 52 |
| 7 | Security | 55 |
| 7.1 | Security Concepts | 55 |
| 7.2 | Attacks | 56 |
| 7.2.1 | Web System Vulnerabilities | 56 |
| 7.2.2 | Other Vulnerabilities | 58 |
| 7.2.3 | Methods of Defense | 58 |
| 7.3 | Summary | 60 |
| 8 | Implementation | 61 |
| 8.1 | The Client | 61 |
| 8.1.1 | The Applet | 61 |
| 8.1.2 | The Web Interface | 63 |
| 8.2 | The Application Server | 63 |
| 8.3 | The Database | 65 |
| 8.3.1 | Archiving documents | 66 |
| 8.4 | The signing process | 66 |
| 8.5 | Summary | 66 |

| | | |
|----------|--|-----------|
| 9 | Future work | 69 |
| A | The user interface | 71 |
| A.1 | The interface for the student | 71 |
| A.1.1 | The submission list | 72 |
| A.1.2 | View the submission details | 73 |
| A.1.3 | Forward to another student | 74 |
| A.1.4 | Submit to the administrators of the course | 75 |
| A.2 | The interface for the teacher | 76 |
| A.2.1 | The e-box | 76 |
| A.2.2 | View the submission details | 77 |
| A.2.3 | Manage folders | 78 |
| A.2.4 | Move files | 78 |
| A.2.5 | Access of others | 79 |
| A.2.6 | The access of the user | 79 |
| A.2.7 | Assignments of the course | 80 |
| A.3 | Common interface | 81 |
| A.3.1 | The settings of the user | 81 |
| B | Source code | 83 |
| B.1 | Client | 83 |
| B.2 | Server | 83 |
| B.3 | Database | 83 |
| C | Testing | 85 |
| C.1 | User Tests | 85 |
| C.2 | Structural Tests | 86 |

List of Figures

| | | |
|------|---|----|
| 3.1 | Sending a message | 13 |
| 3.2 | Receiving a message | 14 |
| 5.1 | A flow graph for the User Interface | 42 |
| 6.1 | The ER diagram | 53 |
| 8.1 | A simple JSP page with database connection | 64 |
| A.1 | The main page for the student | 72 |
| A.2 | The details of the submission | 73 |
| A.3 | The signing process: Enter the submission details | 74 |
| A.4 | The signing process: Signing the document | 75 |
| A.5 | The signing process: Confirming the submission | 75 |
| A.6 | The signing process: Enter the submission details | 76 |
| A.7 | The main page for the teacher and external examiner | 76 |
| A.8 | The details of the submission | 77 |
| A.9 | Edit or Delete a folder | 78 |
| A.10 | Move files between folders | 78 |
| A.11 | Access of others to the users e-box | 79 |
| A.12 | The access of the user | 79 |
| A.13 | View assignment settings | 80 |
| A.14 | Edit assignment settings | 81 |
| A.15 | The user's settings | 81 |

List of Tables

| | | |
|------|--|----|
| 2.1 | The security problems when delivering | 6 |
| 2.2 | Comparison of security | 8 |
| 4.1 | Attributes when signing a file | 18 |
| 4.2 | Attributes of the folder | 19 |
| 4.3 | Attributes of the allowed access | 19 |
| 4.4 | Attributes of the comment | 19 |
| 4.5 | Attributes of the assignment settings | 20 |
| 4.6 | Attributes of the user | 20 |
| 4.7 | Attributes of the course | 21 |
| 4.8 | Attributes describing the use cases | 22 |
| 4.9 | The attributes inserted for each transaction | 35 |
| 4.10 | The attributes logged when a file is signed | 35 |
| 4.11 | The attributes logged when a receiver information is changed | 36 |
| 6.1 | The relationship between entities | 45 |
| C.1 | Functional test for the student | 86 |
| C.2 | Functional test for the teacher | 87 |
| C.3 | Structural test for the student | 88 |
| C.4 | Structural test for the teacher | 89 |

Chapter 1

Introduction

Students at DTU and other Danish universities do not have the option of delivering exam and report documents electronically and securely at this time. The only way students can send documents to teachers is through email. Documents that are submitted without a signature do not fulfill the legal requirements of report- and exam documents. There are some existing products, for example from Adobe, that allow users to sign documents but it is expensive for the universities to buy rights to such systems for all students. It would also not be a good idea to mix the report and exam documents with other documents in the mailbox of the teacher. The solution to the problem is to create a separate system that deals with the submissions.

A digital signature on an electronic document is equivalent to a handwritten signature on a paper document. The signature is a proof for the authorship. It must be unforgeable and firmly attached to the message that has been signed. It should not be possible to alter the signature after it has been transmitted, or reuse the signature as a signature for another message.

The aim of this master's project is to create a prototype of a system that enables students to make electronic submissions of reports and exam files. The files are signed electronically by a student or students before it is submitted and thereafter delivered to the administrators of the course in a secure manner.

The report is organized in the following way:

- Chapter 2 contains the security analysis for the Report and exam submission system. It describes the security problems in the process itself and then compares the security when documents are delivered on paper and when they are delivered electronically.
- Chapter 3 discusses cryptography. It covers both symmetric- and asymmetric encryption. It discusses the digital signatures protocol, and how signatures are generated and verified. Digital certificates are also described.
- Chapter 4 analyzes the requirements for the prototype. The requirements and the actors are identified and described. The use cases are written based on the requirements that have been identified.

- Chapter 5 describes the design of the prototype. The architecture used to design the prototype is described along with the development and runtime environment that was used. The design of the user interface is demonstrated in a diagram.
- Chapter 6 aims at designing the database. An ER Model is created and then transformed into a Relational Model. The Relational Model can be used by a Database Management System to create the database.
- Chapter 7 introduces some security concepts and describes attacks on computer systems. It focuses on attacks on Web systems and the countermeasures that can be taken to prevent or deter attacks.
- Chapter 8 describes how each layer of the prototype was implemented. The flow of the signing process is explained.
- Chapter 9 discusses the status of the project and the future work that needs to be done to transform the prototype into a real system.

Chapter 2

Security Analysis

There are many security problems that arise when a report or an exam document is delivered on paper. The risk of some of these problems would be diminished if the files were submitted electronically. This chapter discusses the security problems concerning submission of reports and exam documents.

The first section introduces the security problems with the delivery. The second section discusses the problems with paper delivery. The third section discusses the problems with electronic submission. The fourth section compares the security of returning documents on paper to an electronic submission, and the fifth section discusses the advantages of a secure electronic submission system. The final section summarizes the chapter.

The following persons are a part of the delivery scenario, in the examples in this section. Alice is the student that is delivering a report or an exam document. Bob is the teacher that is supposed to get the document. Carol is another student or an outside person that wants to provide unauthorized help to Alice. Mallory is a person with bad intentions and no authorized access to the document, that wants to tamper with the document.

2.1 Security Problems

The following security problems are detected in the delivery scenario:

1. Mallory gets the document. He could read, modify or delete it.
2. Carol writes the document for Alice.
3. Carol helps Alice on an exam.
4. Alice falsely maintains her document was removed.
5. The document is lost by Alice or Bob.
6. Alice returns an old document from Carol.
7. Alice views the exam text before the exam.
8. Mallory blocks access to documents.

Table 2.1: The security problems when delivering

2.2 Security of Paper Documents

This section discusses each of the security problems in Table 2.1 when documents are delivered on paper.

1. *The document could get into the hands of Mallory.* Documents are sometimes delivered into unprotected locations or mailboxes with large gaps, from which they could easily get into Mallory's hands.
2. *Carol wrote the document for Alice.* Carol can easily write the document and then Alice signs it, it does not matter if it is delivered on paper or submitted electronically. When Alice signs the document she is maintaining that the document is her own work. If it is later discovered that it is the work of someone else Alice is in serious trouble.
3. *Alice gets a help from Carol on an exam.* There are many ways for Alice to cheat, regardless if the exam is carried out electronically or on paper. Carol could impersonate Alice which is rather hard because personal certificates are viewed when exams take place. Alice could also get the solutions from Carol in various ways if she is in the same building or room.
4. *Alice falsely maintains her document was removed and gets an extended deadline.* It is possible that this could have happened, so Bob has to take her word for it and accept the late delivery.
5. *The document is lost by Bob or Alice.* Alice could easily put the document into the incorrect mailbox and Bob can easily misplace a document.
6. *Alice delivers a report Carol wrote in a previous semester.* The fraud might be detected if the same teacher reads both reports.
7. *Alice is able to view the exam text before the exam and solve it beforehand.* Bob places the exam text in an unsecure place where Alice is able to view it.
8. *Mallory blocks access to documents.* Mallory can block access to documents in several ways, i.e. by destroying the mailbox documents are delivered in, but it is more likely that he would try to block access to an electronic system because that is more challenging.

2.3 Security of Electronic Submission

This section discusses each of the security problems in Table 2.1 when files are submitted electronically.

1. *The file could get into the hands of Mallory.* It is hard for Mallory to get the files from a secure system. He could get access if he could log in as Alice or Bob.
2. *Carol wrote the file for Alice.* Carol can easily write the file and then Alice signs it, it doesn't matter if it is delivered on paper or submitted electronically.
3. *Alice gets a help from Carol on an exam.* There are many ways for Alice to cheat as before. There is an additional risk when the exam is electronic. Carol could solve the exam from another location and submit it in Alice's name. This situation has to be prevented.
4. *Alice falsely maintains her file was removed and gets an extended deadline.* If the system has transaction logs then it is hard for Alice to maintain the lie. The system should return a message to Alice if the file was not delivered to Bob.
5. *The file is lost by Bob or Alice.* The file can be traced in the system if Alice submitted it incorrectly and Bob cannot lose the file if it is kept in a secure database.
6. *Alice delivers a report Carol wrote in a previous semester.* Bob could compare a suspicious text to other reports in the database.
7. *Alice is able to view the exam text before the exam and solve it beforehand.* The exam text only exists electronically and therefore it is hard for Alice to get to it.
8. *Mallory blocks access to documents.* Mallory can be a hacker that wants to block access to the system with attacks on the server or the application itself. It is more likely that a hacker tries to hack a system than it is for someone to physically block access to documents, but if the system is secure it should be harder to block the access.

2.3.1 Secure exams

When electronic exams are carried out in class rooms, some security precaution are necessary to limit the electronic communication in the room and to make sure that that the student does not get an outside help to solve the exam.

One way to limit the electronic communication in the exam room is if the student is not allowed to bring his own computer, but uses a computer located in the exam room which is set up by an employee of the university. Then all options for students to communicate among themselves could be made inactive. This might be an unrealistic condition and it might be better to limit wireless communication in the room.

Students can use Bluetooth and infrared ports to establish wireless communication. Those technologies can only be used if the parties that are communicating within a small area. The students in a classroom could use the technology to communicate and also parties that stand right outside

the classroom walls. A room can be made secure from outside radio signals by using a wallpaper or paint on each wall, ceiling, floor and window. But those countermeasures do not prevent students from communicating within the room. The university could invest in a technology that detects wireless communication like Bluetooth within the room. Pocket size products to detect wireless communication can be bought at a reasonable price on the Internet. Students are often quick to adapt to new technologies and the university has to keep up with the latest technologies to prevent students from using the latest communication products without being detected.

The network traffic in the exam room should be limited so students cannot use the network to communicate with each other or someone on the outside. Students could for example be able to submit their exam document to a secure network, but not to receive files. The network administrator can limit the traffic on the network.

2.4 Comparison

The security problems that could arise are shown in the table below. The security of a paper delivery is compared to the security of an electronic submission. The table shows how easy or hard it is to break the security. It also shows if the security is improved by submitting the files electronically. The system handling the electronic submission is assumed to be secure and fulfill all the requirements in section 4.1.

| Security problem | On paper | Electronic | Improved |
|---|-------------|-------------|----------|
| Mallory gets the document | Medium | <i>Hard</i> | Yes |
| Carol writes the document for Alice | <i>Easy</i> | <i>Easy</i> | |
| Carol helps Alice on an exam | Medium | Medium | |
| Alice maintains her document was removed | <i>Easy</i> | <i>Hard</i> | Yes |
| The document is lost by Alice or Bob | Medium | <i>Hard</i> | Yes |
| Alice returns an old report from Carol | <i>Easy</i> | Medium | Yes |
| Alice views the exam text before the exam | <i>Hard</i> | <i>Hard</i> | |
| Mallory blocks access to documents | Medium | Medium | |

Table 2.2: Comparison of security

2.5 Advantages of a System

There are many advantages of having a electronic submission system for reports and exam files.

- Paperwork will be reduced by having the files on electronic form.
- Mobility of users would be increased. Student and teachers don't have to go to a certain location to hand in or get the files, this could be done from home or any other location. This could make it easier for student to take courses in other universities.

- It is easier to find old files when they are kept in a database than if they are stored in some archive of documents.
- The security is improved when the submission is electronic, see Table 2.2.

2.6 Summary

This chapter has covered the security problems involved in the process of submitting document. The chapter establishes that the advantages of having a Report and exam submission system are great and it is well worth the effort to implement such a system.

Chapter 3

Applied Cryptography

This chapter will discuss how cryptography can be used within the context of creating a secure system to handle submission of documents. One or more students should be able to sign a document, and the teachers needs to verify the signatures of the documents they have received. If the content of the document is confidential then it must be possible to encrypt the document.

The first section discusses encryption, both with secret key and public key. The second section covers the digital signatures protocol, and how signatures are generated and verified. Digital certificates are described in the third section and the final section summarizes the chapter.

3.1 Encryption

Encryption is the process of encoding a message so that the meaning is no longer clear. Decryption is transforming the encrypted message back to the original form. There are two forms of encryption, symmetric and asymmetric encryption. Those encryptions are in other words called secret key encryption and public key encryption.

3.1.1 Secret Key Encryption

Secret key encryption or Symmetric encryption, uses the same secret key for encryption and decryption. The key is shared by both the encryption and decryption party and should be kept secret.

The disadvantages of the Secret key encryption is that each pair of users need a separate key. A system with n users needs $n * (n - 1)/2$ keys. If the system has many users, then it is difficult to determine and distribute these keys. The users cannot be expected to memorize so many keys, so it is difficult to maintain security for the keys that have been distributed.

The most common symmetric encryption standards are DES and AES. DES uses a 56-bit key and has been controversial since it was introduced in 1976. The short key length of DES could make it easy to crack, even if DES has not yet been found insecure. An improvement of the DES has been introduced, the triple DES. It doubles the key length.

The AES was adopted in 2001. The AES uses key of size 128, 192 and 256 bits. The algorithm is unclassified, publicly disclosed and royalty-free. At present, AES seems to be a significant improvement over DES.

3.1.2 Public Key Encryption

Public key encryption or Asymmetric encryption was proposed in 1976 by Diffie and Hellman. Each subject has two keys, one that should be kept private and another that is public and should be made public to those that the subject wishes to share information with. The private key can for example be stored on an encrypted portion of a hard drive, on a Smart card, or on a secure part of a network.

One of the keys is used to decrypt and the other to encrypt. It depends on the nature of the communication which key is used to decrypt and encrypt. If the purpose is to send encrypted message from A to B then A encrypts the plaintext P with B 's public key. The message can only be decrypted with the private key of B , and B is the only one that should have it.

$$P = D(k_{PRIV}, E(k_{PUB}, P))$$

If the purpose of the encryption is to confirm origin of the message then A encrypts the plaintext P with his private key. The recipient decrypts the message using the public key of A and confirms that the message indeed came from A .

$$P = D(k_{PUB}, E(k_{PRIV}, P))$$

The Rivest-Shamir-Adelman (RSA) cryptosystem is a public key system. The algorithm was introduced in 1978 and is based on the underlying hard problem of factoring large numbers.

Public key encryption can take 10,000 times as long as Secret key encryption. Those two types of encryptions are therefore often combined to capitalize the best features of each.

3.2 Digital Signatures

Handwritten signatures have been used for centuries as a proof for authorship. A digital signature is used to sign an electronic document. The digital signature should meet the following properties:

- It must be unforgeable. If person P signs the message M then it is not possible for anyone else to produce the pair $[M, S(P, M)]$.
- It must be authentic. If a person R receives the pair $[M, S(P, M)]$ from P , then R can check that the signature is really from P . The signature is firmly attached to M .
- It is not alterable. M cannot be changed by anyone after it has been transmitted.

- It is not reusable. The signature is a part of the file and cannot be sent again to R , without being detected.

Digital signature schemes can be described by two general classes. They are digital signature scheme with appendix, which requires the original message M as an input to be able to verify the signature, and digital signature scheme with message recovery. The latter one does not require the M as an input to verify the signature, M is recovered from the signature itself. The scheme with the appendix is more common. This section will hereafter only cover digital signature scheme with appendix.

Digital signatures rely on public key cryptography (PKC). The following three algorithms are standard for digital signature generation and verification according to Nist[Ni00]:

- Digital Signature Algorithm (DSA)
- The RSA (Rivest-Shamir-Adleman)
- The elliptic curve digital signature algorithm (ECDSA)

3.2.1 The Protocol

If Alice wants to send a signed message to Bob, she creates a message digest by using a hash function on the message. If any part of the message is changed then the hash function returns a different digest. The hash function is designed in a way that makes it not feasible to try to find a message that returns the same digest as the original message. Alice encrypts the message digest with her private key, the encrypted message digest is the digital signature for the message. Alice sends the message and the digital signature to Bob.

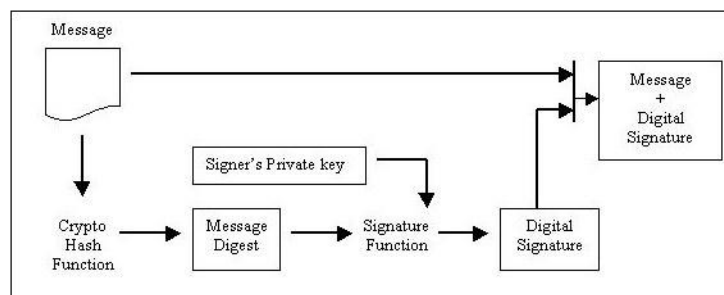


Figure 3.1: Sending a message

Bob receives the message and the digital signature from Alice. Bob decrypts the digital signature using Alice's public key, which returns the message digest. He then hashes the message with the same hash function Alice used and compares the result with the message digest he got from decrypting the digital signature. If those two digests are equal then Bob has verified that the message was signed by Alice. If they are not equal the message was either altered after it was signed or it was not signed by Alice.

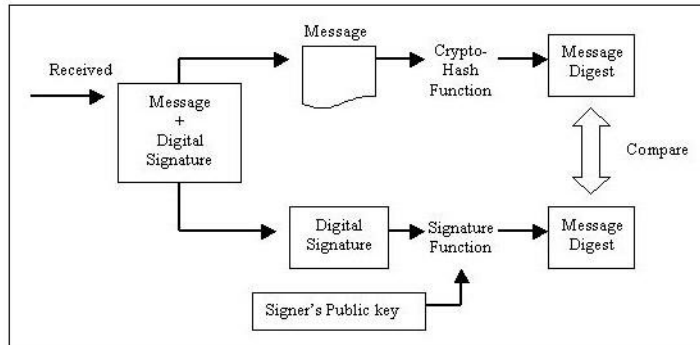


Figure 3.2: Receiving a message

3.2.2 Multiple Signature

It is often necessary to have signatures from more than one party on a document. According to the purposes and operations of multiple signatures, these fundamental cases are identified [LeHu00]:

- Sequential Multiple Signature. A signature is needed from two or more parties. The first signer signs the document, the second signer signs the document and the signature of the first signer. This goes on until all signatures are appended.
 - Independent Sequential Multiple Signature. The sequence of signing is not important. The parties are not validating each others signatures, i.e. the signer only signs the content of the message.
 - Dependent Sequential Multiple Signature. The sequence of signing is important. The last signer is not only signing the content of the message but also the signatures of the previous signers.
- Parallel Multiple Signature. A document must be signed concurrently by a number of parties. Each signer is signing the content of the form, but not the signatures of the other parties.
- Anonymous Signature. A document is signed but the identity of the signers is kept secret.

Schneier [Sc96] introduces a general scheme to implement multiple signatures on message M . The scheme uses one way hash functions. The scheme is as follows:

- Alice signs the hash of the M
- Alice sends her signature and the message M to Bob

- Bob signs the hash of M
- Bob sends the message M , his signature and Alice's signature to Carol
- Carol can verify both Alice's signature and Bob's signature

3.3 Certificates

A digital certificate is an electronic way of establishing the credentials of a subject. The certificate is signed with the digital signature of some trusted party. The certificate contains the following information:

- The certified information, i.e. the name, a serial number and the expiration dates.
- A copy of the certificate owner's public key. The public key is used for encrypting and decrypting messages, and verifying digital signatures.
- The digital signature of the trusted party so the recipient can verify that the certificate is real. The trusted party can be a common respected individual in a large company or a Certificate Authority (CA) like VeriSign or TDC in Denmark.

The most common certificate standard on the Web is X.509. Certificates are useful if two parties want to communicate electronically in a secure manner. They can trust each other even if they have never met, if they trust the party that signs the certificate.

3.4 Summary

This chapter has discussed some fundamental concepts of cryptography like encryption, decryption, digital signatures and certificates. The understanding of those concepts is important to be able to create a secure system that handles submission of documents.

The system will use AES encryption to encrypt documents. The AES encryption was chosen because it has longer keys than DES, and seems to be more secure.

The system implemented supports OCES certificates. The OCES certificates use the X.509 certificate standard, which is the most common on the Web. The algorithm used for the digital signature is RSA. SHA1 is the cryptographic hash function that will be used.

Chapter 4

Requirement Analysis

This chapter contains the requirement analysis. First the general requirements of the system are presented. Then the operations of the system are listed along with their attributes. Some of the data used by the system is read from other systems within the institution, the third section shows these attributes. The fourth section identifies the actors of the system and the operations they have access to. The use cases of the system are described in detail in section five and the sixth section describes how the transactions are logged in the system. The final section summarizes the chapter.

4.1 Requirements

The system should fulfill the following requirements:

- A user signs in by using a digital certificate and a password.
- A file can be signed by one or many students and submitted to the administrators and the external examiners of the course. It is thereafter available in their e-boxes.
- The student can submit a file in courses he is registered as a student.
- Files are kept in a secure database and only authorized persons have access to them.
- Submission can be limited with certain locations and time, when an exam is carried out.
 - Exams can only take place in special rooms where outgoing traffic is limited. Users should only be able to upload the exam file to the system, but not to send emails or upload files in any other way. This is a technical problem, which is outside the scope of the system. This is discussed in more details in section 2.3.1.
- All updates in the system are logged in a transaction log. The time stamp, the user making the transaction and the location is kept along with the relevant attributes. The log cannot be deleted unless X years have passed.
- The user gets a receipt when a file has been submitted.
- The user gets an error message if the file was not submitted successfully.

- Only files with the correct postfix can be submitted. The responsibility of delivering a readable file is with the user. The system asks the user to confirm before the file is submitted to diminish the risk that the user sends the wrong file.
- A text search can be carried out to compare the file with other files in the database, to detect fraud.
- It is possible to encrypt a file, to protect confidential information. Only the recipient should be able to decrypt the file.

4.2 The Operations of the System

The system should have the following operations to fulfill the requirements in section 4.1:

- Log on using a digital certificate and a password.
- Log off
- Sign a file and submit it to the administrators and external examiners of the course or forward it to a participant in the course. The attributes needed when a file is being submitted or forwarded are:

| | |
|---------------|--|
| File name | The name and location of the file |
| Assignment id | The assignment is selected from registered assignments within the course |
| Document name | The title of the report or exam file |
| Location | The location where the signature takes place |
| Recipients | This can be all the registered receivers for an assignment in a course or one of the other students. |
| Encryption | Should the file be encrypted? |
| File | The file that should be submitted |
| Certificate | The certificate used to sign the file |

Table 4.1: Attributes when signing a file

- Encrypt a file. The file is encrypted so only the registered recipients can view the file.
- The student confirms that he wishes to sign a file before it is signed and submitted or forwarded.
- Delete a signed file. A student can delete a signed file sent from another student if he does not wish to sign it.
- List accessible files in an e-box. The recipient views files in a folder structure he can create himself, originally all files are in the Inbox. He can also search for a file or select all files submitted for a certain course or assignment. If a file is submitted before or after it should be submitted, then the assignment will be distinguished by a red font.

| | |
|--------|---------------------------------|
| Name | The name of the folder. |
| Parent | The parent folder. |
| Owner | The user that owns this folder. |

Table 4.2: Attributes of the folder

- Manage folders in the e-box. The recipient can create a folder, edit the name of the folder and delete a folder in the e-box. A new folder is created with the attributes in table 4.2.
- Move files from one folder to another in the e-box.
- Grant or revoke access to another user, to one folder in the e-box. The following attributes are needed when access is granted:

| | |
|----------|--|
| Password | The password that the user and the another user agree on. |
| Active | Is the access active or not. When access is revoked then this becomes false. |
| Folder | A link to the folder the another user should have access to. |
| User | The user that should have access. |

Table 4.3: Attributes of the allowed access

- The recipient of a file can create or delete a comment on the submission. The following attributes are needed when a comment is created.

| | |
|------------|---------------------------------------|
| Text | The text of the comment. |
| Date | The date the comment was created. |
| User | The user that created the comment. |
| Submission | The submission the comment refers to. |

Table 4.4: Attributes of the comment

- Ask for a notification by email or SMS when a new submission is in the e-box or in the submission list of the student.
- View a file by clicking on a link in the e-box. If the file is encrypted then the user gets the option to decrypt it.
- Decrypt a file that was encrypted by the sender. The recipient inputs a key to decrypt the file.
- Archive received files. When files are archived they are no longer available from the Inbox of the user but can be viewed by selecting the archive folder. A search can be done within the archive.
- Search for a text in the file database to detect if the solution is a copy of another solution.
- View the course's assignments. The information about the assignments of the course can be viewed and edited.

- Insert, Edit or Delete an assignment. An assignment can be deleted if a submission has not yet been submitted for the assignment. A new assignment is created with the following attributes:

| | |
|------------------|---|
| Type | The assignment type, i.e. Assignment 1 or Final report. |
| Date from | The date an assignment can be delivered from. |
| Date to | The date an assignment can be delivered to. |
| Location id | The location if the file should only be submitted from a certain room or building during an exam. The location is selected out of list of location groups, each location group has one or more IP addresses. |
| To all or group | Choose if the student can forward files to all participants of the course or just within a group he is registered in. |
| Allow encryption | Should users be able to encrypt files for this assignment? Default value is no. |
| Recipients | Which of the administrators and external examiners of the course should receive files for the assignment. For each recipient it can be registered how many of the files he should get, i.e. 1 of 3 starting with the 2. file delivered. The default value is all the files. |

Table 4.5: Attributes of the assignment settings

- View the transaction log. All transactions in the system are logged and can be viewed by authorized users. The attributes of the transaction log are different according to what is being logged. See details in section 4.6.

4.3 Information from other Systems

The Report and exam submission system is thought of as an addition to the existing information systems at an educational institution. Some of the information used by the system are read from other systems. The following data is not registered within the system but read from other systems like the CampusNet.

- Information about users. The following attributes are read:

| | |
|------------|-------------------------------|
| Id | The user id |
| First name | The first name of the user |
| Surname | The surname of the user |
| Email | The email address of the user |
| PID | Id of the digital signature |

Table 4.6: Attributes of the user

- Information about courses. The following attributes are read:

| | |
|--------------------|--|
| Id | The id of the course of a semester |
| Course name | The name of the course |
| Course number | The course number |
| Version | The year and name of the semester |
| Access | The users that are registered users for the course |
| Administrators | The users that are registered administrators for the course |
| External examiners | The users that are registered as external examiners for the course |

Table 4.7: Attributes of the course

4.4 The Actors and their Operations

The actors in the system are the general user, the course administrator, the external examiner and the system administrator.

All the actors can:

- Log in.
- Log out.
- Select a course from courses the user has access to.
- Ask for a notification when there is a new submission in the e-box or in the submission list.

The general user is a student submitting a report or an exam file. The user can:

- Receive a signed file from another user.
- Send a signed file to the administrators and the external examiners of a course or to another student, which is a participant in the course or a member of the same group in the course.
- View the file before it is signed and sent.
- Encrypt a file after it has been signed.
- Delete a file, not yet submitted.
- View his submissions, who has access to it and who has signed it.

The course administrator can be the teacher or another administrator of the course, he can:

- Change the settings of an assignment.

The external examiner views report and exam documents along with the administrator of the course, they both can:

- Receive a signed file.
- Manage folders in the e-box.
- Move files between folders.
- Grant or revoke access to another user, to one folder in the e-box.
- Create comments on submissions.
- Archive files.
- View a signed file.
- Decrypt a file.
- Make a text search in the file database to detect fraud.

The system administrator is a person that has access to all the data of his subordinates, he can:

- View and search in the transaction log for all transactions.

4.5 Use Cases

The use cases are described in detail in this section. The use cases are described by the following attributes:

| | |
|----------------|--|
| Name | The name of the use case |
| Actors | The actors that have access to the use case |
| Purpose | The purpose of the use case |
| Preconditions | The conditions that have to be fulfilled before the user gains access to the use case |
| Description | A detailed description of the use case |
| Postconditions | The conditions that are fulfilled after the use case has been executed |
| Variations | A description of instances when a use case is not executed in the most common way, i.e. when error or warning messages are shown instead of finishing the execution. |

Table 4.8: Attributes describing the use cases

4.5.1 Log on

Name: Log on.
 Actors: All actors.
 Purpose: Logging on to the system.
 Preconditions: The user information is registered in the system.
 Description: This function comes from the Campusnet.

4.5.2 Log out

Name: Log out.
Actors: All actors.
Purpose: Logging out of the system.
Preconditions: The actor had logged on.
Description: This function comes from the Campusnet.

4.5.3 Document notification

Name: Document notification.
Actors: All actors.
Purpose: Ask for a notification when a new submission is in the e-box or in the submission list of the student.
Preconditions: The actor is logged on.
Description: The actor selects if he wants to get notification with email and/or SMS. The email and SMS is retrieved from the Campusnet. He can Submit or Cancel.

4.5.4 List and select a course

Name: Select a course.
Actors: The general user.
Purpose: Select a course, to be able to sign and submit files and view submission receipts.
Preconditions: The actor is a participant in the course.
Description: The number and name of the actor's active courses is listed. The actor has the option Submit and Forward for each course (see 4.5.6). The Submit option allows the actor to sign a file and send it to the course's administrators. The Forward option allows the actor to sign a file and submit it to another user, for further signing.

4.5.5 List and select a signed file

| | |
|----------------|--|
| Name: | Select a signed file. |
| Actors: | The general user. |
| Purpose: | Select a file that has been signed by other users, to be able to sign and submit it. |
| Preconditions: | The file has been signed by another user and forwarded to the actor. |
| Description: | The number and name of the course, the type and name of the assignment, document name and signature information of files submitted to the actor are listed. The actor has the option Submit and Forward for each file in the list (see 4.5.6). The Submit option allows the actor to sign the file and send it to the course's administrators. The Forward option allows the actor to sign the file and submit it to another user, for further signing. If the multiple signatures are parallel then the actor only has the option Sign, which allows the actor to sign the document (see 4.5.6). If the actor initiated the submission he also has the option Submit, which submits the document to the course's administrators (see 4.5.11). |

4.5.6 Sign a file

| | |
|----------------|---|
| Name: | Sign a file and submit it. |
| Actors: | The general user. |
| Purpose: | Sign a file in a non-refutable manner and submit it. |
| Preconditions: | The actor has a registered digital signature and the course has been selected. |
| Description: | The multiple signatures are assumed to be sequential. One student initiates the signing process, and then the file is forwarded to zero or more students. Each student can only forward the file to one other student. The student that signs the documents last, submits the file to the administrators of the course. Another option is for the multiple signatures to be parallel. One student initiates the signing process and he is in charge of the submission, he selects all the students that should sign the document. The students that receive the document sign it, when some or all students have signed the document, the student that initiated the submission can submit the document. Some of the use cases are altered slightly if the signatures should be parallel, the alterations are mentioned in the use cases themselves, see section 4.5.5, 4.5.6 and 4.5.11. |

The signing consists of the following items:

- The name and number of the selected course is shown.
- The actor selects a file from a drive, which he wants to sign. The assignment type is selected from registered assignments within the course and the assignment title is registered. If the file has been signed by another user then the name of the file and the type and name of the assignment is shown and cannot be changed. The name, date and location information of the signatures are shown. If the file has been signed before, then the actor has the option to download the file.
- The location where the signature takes place can be registered.
- If the actor wants to forward the file to another user for further signing then the actor can choose which user gets the file. He can choose from all the participants of the course or the members of his group, that is up to the administrator of the course. If the number of participants is more than 25 then a search can be done to find the correct recipient. If the multiple signatures should be parallel then the actor can select more than one recipient.
- The actor has the option of encrypting the file if it contains confidential information. The file is encrypted with the public key of the recipient, see section 4.5.7. This option can be removed by the course's administrators if none of the submitted files should contain confidential information.
- The actor can Submit or Cancel.

Postconditions: If the signed file was forwarded then it is available in the submission list of the recipient. If the file was submitted, it is available in the e-box of the course administrators.

- Variation A: No assignments have been registered for the course. The system returns to the previous page and shows an error message.
- Variation B: No recipients can be found. The system returns to the previous page and shows an error message.
- Variation C: The file has an incorrect postfix. A message to select a new file is shown.
- Variation D: No file has been selected. A message to select a file is shown.
- Variation E: The file has been changed since it was last signed. An error message is shown.
- Variation F: A file has already been signed and submitted for the assignment by the actor. A warning message is shown and the actor can choose to Continue or cancel.
- Variation G: More than one recipient has been chosen, show error message. This does not apply if the multiple signatures are parallel.

4.5.7 Encrypt a signed file

- Name: Encrypt a signed file.
- Actors: The general user.
- Purpose: The actor can encrypt a file after it has been signed.
- Preconditions: The file has been signed.
- Description: The file is encrypted with the public key of the recipient. If there are more than one recipients then the file is encrypted for each of them.

4.5.8 Confirmation

- Name: Confirmation.
- Actors: The general user.
- Purpose: The actor can confirm or cancel the signing process.
- Preconditions: The actor has requested to sign a file.
- Description: The name and number of the course, filename, the type and name of the assignment, location and recipients of the file are shown. If other users have signed the file, their signatures are shown. The actor is asked if he has viewed and understood the content of the file he is about to sign. By signing he is stating that the file is his own work or the work of his group. He agrees to submit it to the recipients. The actor can Confirm or Cancel. The transaction is logged if the actor chooses to Confirm.

4.5.9 View the details of a signed file

| | |
|----------------|--|
| Name: | View the details of a signed file. |
| Actors: | The general user. |
| Purpose: | The actor can view the details of a signed file. |
| Preconditions: | The file has been sent to the actor or signed by the actor. |
| Description: | The actor can view the details of a submission, he sees the course information, assignment information, submission title and file type. He can download the document (see section 4.5.23 or decrypt it, if it has been encrypted (see section 4.5.24). The actor sees who has signed the document, who has the document but has not signed it and the administrators it has been submitted to. |

4.5.10 Delete a signed file

| | |
|----------------|--|
| Name: | Delete a signed file. |
| Actors: | The general user. |
| Purpose: | The actor can delete a signed file sent from another user. |
| Preconditions: | The file has been sent to the actor. |
| Description: | The actor can select one or more signed files and delete them if he doesn't wish to sign and submit them. The transaction is logged. |

4.5.11 Submit a file with parallel multiple signatures

| | |
|-----------------|---|
| Name: | Submit a multiple signed file. |
| Actors: | The general user. |
| Purpose: | The actor can submit a file that has been signed by himself and perhaps other students. |
| Preconditions: | The user has forwarded the file to one or more students. The multiple signatures scheme is parallel. |
| Description: | The actor submits a file that he had forwarded to one or more students, and signed by himself and zero or more other students. The transaction is logged. |
| Postconditions: | The signed file is available in the e-box of the course administrators. |

4.5.12 List accessible files in an e-box

| | |
|----------------|---|
| Name: | List received files. |
| Actors: | The course administrator and the external examiner. |
| Purpose: | List files the actor should have access to. |
| Preconditions: | The actor is logged on. |
| Description: | The actor can view the list of files in a folder structure he himself has created. He can also search for a file (see 4.5.27) or select all files submitted in a certain course or assignment. If a file is submitted before or after it should be submitted, then the assignment will be distinguished by a red font. The following information is shown for each document. The name and number of the course, the type and name of the assignment and signature information. The signature information contains name, date and location. The file can be opened by clicking on the assignment title (see 4.5.23). The actor can move, archive or download one or more files (see sections 4.5.16, 4.5.26 and 4.5.25). Each file could have the option Grade, that allows the teacher to give the students, submitting the file, a grade. The grading system is not a part of this system. |
| Variation A: | If the actor has been registered as an administrator after the file was encrypted, then he does not have the option to decrypt it and an error message is shown, including the names of the recipients that can decrypt the file. |

4.5.13 View the details of a signed file

| | |
|----------------|--|
| Name: | View the details of a signed file. |
| Actors: | The course administrator and the external examiner. |
| Purpose: | The actor can view the details of a signed file. |
| Preconditions: | The file has been submitted to the actor. |
| Description: | The actor can view the details of a submission, he sees the course information, assignment information, submission title and file type. He can download the document (see section 4.5.23) or decrypt it if it has been encrypted (see section 4.5.24). The actor sees who has signed the document, to whom it has been submitted and the comments that are attached to the submission. He can create or delete comments, (see sections 4.5.21 and 4.5.22). |

4.5.14 Insert a folder

Name: Insert a folder.
Actors: The course administrator and the external examiner.
Purpose: Insert a folder to make the e-box more organized.
Preconditions: The actor is logged on.
Description: The name of the folder is entered and the parent folder is selected. The actor can Insert or Cancel.

4.5.15 Edit a folder

Name: Edit a folders.
Actors: The course administrator and the external examiner.
Purpose: Rename or delete a folder.
Preconditions: The actor is logged on.
Description: The folder that should be edited is selected. The new name can be entered and then the actor can Edit, or he can Delete. He can also choose to Cancel.

4.5.16 Move files between folders

Name: Move files between folders.
Actors: The course administrator and the external examiner.
Purpose: Move files to make the e-box more organized.
Preconditions: Files have been selected.
Description: The destination folder is selected. Then the actor can Move or Cancel.

4.5.17 View access to folders

Name: View access to folder.
Actors: The course administrator and the external examiner.
Purpose: View the users that have access to the e-box.
Preconditions: The actor is logged on.
Description: Information about the access of other users is shown. The user name and number, the folder name he has access to and status of the access. The actor has the option to grant and revoke access. The actor can select one or more active access records and press Revoke (see 4.5.19). Then the actor can Revoke or Cancel. He can also grant access to someone that has been revoked or grant access to a new user (see 4.5.18).

4.5.18 Grant access to a folder

Name: Grant access to folder.
Actors: The course administrator and the external examiner.
Purpose: Allow another user to view files in a folder.
Preconditions: The actor is logged on.
Description: The number of the user is entered or he has been selected, and the folder that the user should have access to is selected. Then the actor can Submit or Cancel.
Postconditions: Another user has access to a folder of the actor.

4.5.19 Revoke access to a folder

Name: Revoke access to folder.
Actors: The course administrator and the external examiner.
Purpose: Don't allow another user to view files in a folder anymore.
Preconditions: The other user has been selected.
Description: The actor is asked if he wants to revoke access for the other user. He can Submit or Cancel. If the actor submits then the other user no longer has access to the folders of the actor.

4.5.20 View my access to folders

Name: View my access to folders.
Actors: The course administrator and the external examiner.
Purpose: View the folders the actor has access to.
Preconditions: The actor is logged on.
Description: The actor can view the folders he has access to and view the files in the folder he has access to.

4.5.21 Create a comment on submission

Name: Create a comment.
Actors: The course administrator and the external examiner.
Purpose: Comment on a submission.
Preconditions: The submission has been selected.
Description: A comment can be created on the selected submission. A text is entered and the actor can Insert or Cancel.

4.5.22 Delete a comment on submission

Name: Delete a comment.
Actors: The course administrator and the external examiner.
Purpose: Remove a comment from submission.
Preconditions: The comment has been registered.
Description: Information about the comment is shown. The actor can Delete or Cancel.

4.5.23 View a signed file

Name: View a signed file.
Actors: The course administrator and the external examiner.
Purpose: The administrator can view a file.
Preconditions: The actor has access to the file and a file has been selected.
Description: The file is opened in the corresponding editor. If the file has been encrypted then there is an option to decrypt the file when it is opened.

4.5.24 Decrypt a signed file

Name: Decrypt a signed file.
Actors: All actors.
Purpose: Decrypt an encrypted file before it is viewed.
Preconditions: The actor has access to the file and an encrypted file has been selected.
Description: The actor decrypts the file by using his private key.

4.5.25 Download to local drive

Name: Download to local drive.
Actors: The course administrator and the external examiner.
Purpose: The actor can download a number of files to local drive.
Preconditions: The actor has access to the files.
Description: The actor can select one or more signed files and download them. The files are saved to a local drive selected by the actor. The filenames include the student number of the students that have signed the document.

4.5.26 Archive a signed file

Name: Archive a signed file.
Actors: The course administrator and the external examiner.
Purpose: The actor can archive a signed file.
Preconditions: The actor has access to the file.
Description: The actor can select one or more signed files and archive them. The files will no longer be available from the folder, only from the archive.

4.5.27 Search for a file

Name: Search for a file.
Actors: The course administrator and the external examiner.
Purpose: Search for a file in the e-box.
Preconditions: The actor has logged on.
Description: The actor can enter a search text and select to Search. The search text is compared with the user names and numbers, the assignment name and the submission name. The search result is listed.

4.5.28 Search for a text in the file database

Name: Search for a text in the file database.
Actors: The course administrator and the external examiner.
Purpose: Searching for a text to detect if the file is a copy of an old assignment.
Preconditions: The actor is logged on.
Description: The text to search for is entered. The actor can select if he wants only to search in files submitted in the course this semester or include old semesters. He can enter a date to search from. If the search has any results then the type and name of the assignment and the signatures is shown. If the actor has access to a file he can view it, if not he can see which persons have access to it.

4.5.29 View the course's assignments

| | |
|----------------|--|
| Name: | View the course's assignments. |
| Actors: | The course administrator. |
| Purpose: | View the course's assignments and have the option to change them. |
| Preconditions: | The course has been registered. |
| Description: | Information about assignments is shown: the course, the semester, the type of the assignment, data from and to, location, if the participant can only forward within a group and those that should receive submissions. The actor has the option of creating a new assignment (see 4.5.30) or edit existing ones (see 4.5.31, 4.5.32). |

4.5.30 Insert an assignment

| | |
|----------------|---|
| Name: | Insert an assignment. |
| Actors: | The course administrator. |
| Purpose: | Insert an assignment for a course. |
| Preconditions: | The course has been registered. |
| Description: | Information about the course and semester is shown. The actor can create a new assignment and register the assignment settings attributes: the type of the assignment, date from and to, location, if the participant can only forward within a group and those that should receive submissions. Those that should receive files for the assignment are selected from the list of administrators and external examiners for the course. Users can receive only a part of the files for a course. This is done by dividing the students in a course into subgroups and then let a user be a receiver for that group. The actor can Insert or Cancel. The transaction is logged when the assignment is created. |
| Variation A: | Invalid date entered into a date field. Show an error message. |
| Variation B: | The date of the date field is in the past. Show an error message. |

4.5.31 Edit an assignment

| | |
|----------------|---|
| Name: | Edit an assignment. |
| Actors: | The course administrator. |
| Purpose: | Edit an assignment for a course. |
| Preconditions: | An assignment has been registered. |
| Description: | Information about the course and the semester is shown. The actor can edit the assignment settings: the type of the assignment, data from and to, location, if the participant can only forward within a group and those that should receive submissions. Those that should receive files for the assignment are selected or unselected from the list of administrators and external examiners for the course. The actor can Edit or Cancel. The transaction is logged. |
| Variation A: | A file has been submitted for the assignment, show a warning. The actor can Continue or Cancel. |
| Variation B: | The recipients are changed after a encrypted file has been submitted for the assignment, then a warning is showed. The actor can choose to Continue or Cancel. |
| Variation C: | Not a valid date entered into a date field. Show an error message. |
| Variation D: | The date of the date field is in the past. Show an error message. |

4.5.32 Delete an assignment

| | |
|----------------|---|
| Name: | Delete an assignment. |
| Actors: | The course administrator. |
| Purpose: | Delete an assignment for a course. |
| Preconditions: | The assignment has been registered. |
| Description: | Information about the assignment is shown. The actor can Delete or Cancel. |
| Variation A: | Submissions have been submitted for the assignment. The assignment cannot be deleted. |

4.5.33 View the transaction log

| | |
|----------------|---|
| Name: | View the transaction log. |
| Actors: | The system administrator. |
| Purpose: | View the transaction log. |
| Preconditions: | None. |
| Description: | The actor can search in the log by all fields in the log. The log shows the course, the semester, the type and name of the assignment, the user making the transaction, the transaction date, the location of the transaction, the transaction type, the attributes that were changed and the new values. |

4.6 The Transaction Log

Updates in the system are logged so it can be checked who created and edited information in the system. For each transaction that is logged these attributes are inserted:

| | |
|------------------|--|
| Transaction date | The date of the transaction. |
| User id | The user that is logged on and is making the transaction. |
| Location | The IP address of the computer used to make the transaction. |
| Key | The id of the record that was inserted or edited during the transaction. |
| Transaction type | The type of the transaction |

Table 4.9: The attributes inserted for each transaction

The following transactions in the system are logged:

- Confirm to sign and submit a file. The basic attributes in Table 4.9 are registered along with the following attributes:

| | |
|--------------------|--|
| Transaction type | $\text{type} \in \{\text{Submit}, \text{Forward}\}$ |
| Assignment id | The id of the assignment the file is delivered in. From the id all the information about the assignment and the course can be retrieved. |
| Name | The title of the report or the exam file. |
| Signature date | The date when the file was signed. |
| Signature location | Location where the file was signed. |
| Signature | The name and other information about the signer. |
| Recipient | The user information of the recipients of the file. |
| Encrypted | Was the file encrypted? |

Table 4.10: The attributes logged when a file is signed

- Delete a signed file. The basic attributes in Table 4.9 are registered along with the transaction type, which is Delete.
- Insert or edit an assignment. The basic attributes in Table 4.9 are registered along with all the attributes of the assignment (Table 4.5), that were changed. The transaction type is Insert assignment or Edit assignment.
- Add or remove users that should receive files for an assignment by selecting a user or users from a list of administrators and external examiners for a course. The basic attributes in Table 4.9 are registered along with the following attributes:

| | |
|------------------|---|
| Transaction type | type \in {Insert receiver, Update receiver, Delete receiver}. |
| Assignment id | The assignment the user should receive files for. |
| User | The user information of the administrator or external examiner that is updated. |

Table 4.11: The attributes logged when a receiver information is changed

4.6.1 The Attributes

The following tables shows all the attributes of the transaction log and for which transactions they are used.

| Type | user | date | loc. | key | type | user id | ass. id | encr. | name |
|-------------|------|------|------|-----|------|---------|---------|-------|------|
| Submit | X | X | X | X | X | X | X | X | X |
| Forward | X | X | X | X | X | X | X | X | X |
| Delete | X | X | X | X | X | | | | |
| Insert ass. | X | X | X | X | X | | | X | X |
| Edit ass. | X | X | X | X | X | | | X | X |
| Delete ass. | X | X | X | X | X | | | | |
| Insert rec. | X | X | X | X | X | X | | | |
| Edit rec. | X | X | X | X | X | X | | | |
| Delete rec. | X | X | X | X | X | X | | | |

| Type | signature | sign loc | from | to | loc id | all group |
|-------------|-----------|----------|------|----|--------|-----------|
| Submit | X | X | | | | |
| Forward | X | X | | | | |
| Delete | | | | | | |
| Insert ass. | | | X | X | X | X |
| Edit ass. | | | X | X | X | X |
| Delete ass. | | | | | | |
| Insert rec. | | | | | | |
| Edit rec. | | | | | | |
| Delete rec. | | | | | | |

4.7 Summary

This chapter has discussed the requirements and the use cases. Identifying the requirements and creating the use cases is an important part of creating an analysis for a system. Analysis is the first step in the software development process and is necessary if the goal is to create a software that fulfills the expectations of the users.

Chapter 5

System Design

This chapter describes the design of the prototype. The first section discusses the architecture used for the system. The second section describes the development and runtime environment of the prototype. The third section describes the design of the user interface.

5.1 System Architecture

The Report and exam submission system has some data that is convenient to store in a database, like the structure of the assignments and the submission itself. Users should be able to use the system if they have access to a Web browser. Most of the work should be done on an application server. The architecture that was chosen is the three-tier architecture.

The three-tier is a special type of server/client architecture. Each tier runs on a different platform. These are the three tiers:

- The client, which is the user interface and runs on the user's computer.
- The application server, composed of functional modules that process data and run on a server. This is often called a middle tier.
- The database server, a database management system (DBMS) that stores the data required by the middle tier.

5.1.1 The Client

The system is a web system and is accessed through a browser. When files are signed, a private key is used to generate the signature. The middle tier cannot generate the signature because then the server would need to know the private key of the user, but the private key must be kept secret by the client. The signature must therefore be generated on the client. The same applies to the decryption process, it also needs the private key of the user, and must therefore be carried out on the client. The browser can for example use Java Applets or plug-ins to run programs on the client.

An applet is a program written in the JavaTM programming language that can be included in an HTML page. When a Java technology-enabled browser is used to view a page that contains an applet, the applet's code is transferred to the user's system and executed by the browser's Java Virtual Machine (JVM).

Plug-in is a set of technologies that provide a tool for linking a desktop application to the Web. Java Plug-in technology is a part of the Java 2 Runtime Environment, Standard Edition (JRE) and links Java desktop applications to the browser. ActiveX is Microsoft's answer to the Java technology. ActiveX links Microsoft based applications to the Web. The application can be Microsoft based software like Word, Excel or a custom made application created by using Microsoft technology, for example a Visual Basic program.

The client part will be implemented as a Java applet. Java programs are platform independent, the compiled Java bytecode can run on the Java Virtual Machine of any computer. It does not matter if the platform is a Windows XP, Solaris or Linux. Therefore Java is the preferred alternative.

5.1.2 The Application Server

The server used for development and testing is an Apache Jakarta Tomcat. The Tomcat server is a Java based Web Application container that was created to run Servlets and JavaServer Pages (JSP) in Web applications.

5.1.3 The Database Server

The database used for the development and testing is MySQL. MySQL is fast, reliable, and easy to use. MySQL is a popular and free database. It has all the functionality needed for this project and was therefore the database of choice. The Campusnet uses Microsoft SQL server and the runtime database is therefore a SQL server. This is taken into consideration when developing the middle tier.

5.2 Development and Runtime Environment

5.2.1 Client's Requirements

The client has to have a browser that can run Java Applets. The user also has to have a digital certificate to be able to sign or decrypt documents.

5.2.2 Programming Language

Java is used to create the applet used to sign submitted documents and decrypt received files. JavaServer Pages (JSP) 2.0 is the language used to create dynamic HTML pages for the Web system. The JSP pages also use JavaScript for input validation and to make the HTML pages more user friendly. Cascading Style Sheets are used to manage the style of the user interface. Java is the programming language used on the application server.

5.2.3 Development Environment

The system is developed on Windows XP Professional operating system. The Web server used is Tomcat 5.0 and the database is MySQL 4.1.

5.2.4 Runtime Environment

The Runtime environment is the environment the Campusnet is running in. It runs on Microsoft Windows Server. The Web servers are IIS from Microsoft that runs ASP.NET pages and Tomcat which runs JSP pages. The database used is Microsoft SQL Server.

5.3 User Interface

The diagram in figure 5.1 shows the flow between the frames in the User Interface. The arrows from one frame to another are often both ways, the user can either select to go back or is sent back after a certain action.

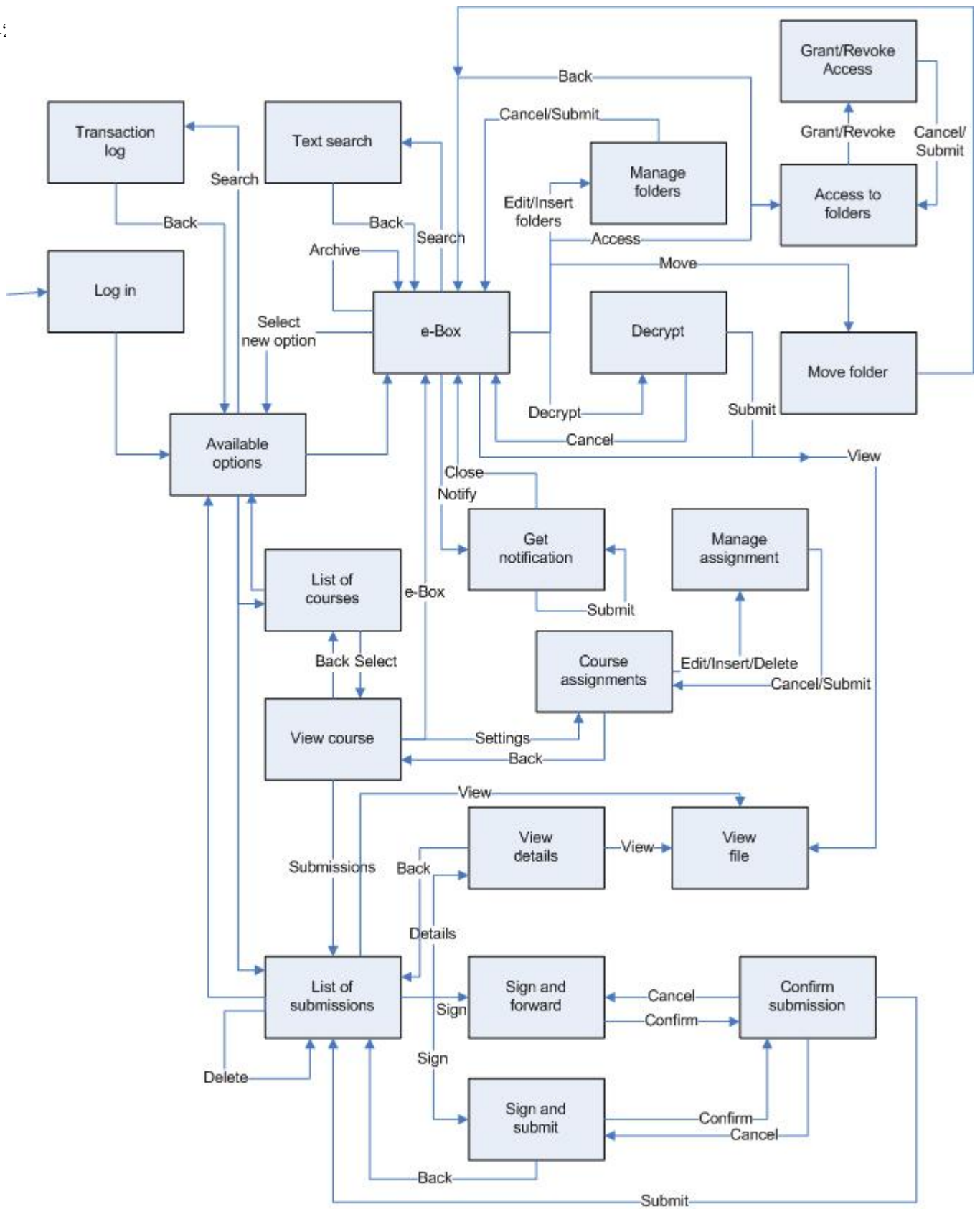


Figure 5.1: A flow graph for the User Interface

Chapter 6

Database Design

This chapter discusses how the database for the system was designed. The first level of the database design is to acquire understanding of the database needed to fulfill the requirements of the system. This is done in the first section by creating an ER model. The second section describes how the ER model is transformed into a Relational model, which is used by Database Management Systems to create the actual database. The final section summarizes the chapter.

6.1 The ER Model

The Entity-Relational (ER) model is a high-level conceptual data model. The aim of a conceptual model is to acquire a complete understanding of the database structure, semantics, relationships and constraints of the system to be built. The basic concepts of an ER model are entities, attributes and relations. An entity represents an object, the attribute is a value that describes the entity and the relation is an association between two or more entities. The ER model will then be mapped to a Relational model, the Relational model is used to implement the database.

The ER diagram is shown in a figure 6.1, at the end of the chapter. The entities, attributes and relations or the diagram will be discusses in the following subsections.

6.1.1 The Entities

The following entities were found in the electronic report and exam submission problem:

- **User** is a person using the system.
- **Course** is a course taught on a semester.
- **Assignment** is an assignment to be delivered in a certain course on a certain semester.
- **Location** is a physical location where an exam can be carried out in. Location can for example be a building, a room or many rooms.
- **Address** represents a computer.
- **Submission** is the submission of a report or an exam file.
- **Submission notification** can be sent by email or SMS to a user that receives a new submission.
- **Folder** can be created in the e-box to organize files.
- **Transaction log** keeps the transaction log for the system.
- **Transaction type** is the types of transactions that are logged in the system.

6.1.2 The Relations

The relationship between the entities is given in the table below.

| No. | Entity 1 | Relation | Entity 2 | Type |
|-----|-----------------|-----------------------------|------------------|------|
| 1. | User | registered | Course | N:M |
| 2. | User | gets | Submission | N:M |
| 3. | User | signs | Submission | N:M |
| 4. | User | gets an encrypted | Submission | N:M |
| 5. | Element | has | Assignment | 1:N |
| 6. | Assignment | gets | Submission | 1:N |
| 7. | User | should get submissions for | Assignment | N:M |
| 8. | Assignment | is carried out at | Location | 1:1 |
| 9. | Location | has | Address | N:M |
| 10. | User | views | Submission | N:M |
| 11. | User | makes | Transaction log | 1:N |
| 12. | User | comments on | Submission | N:M |
| 13. | User | gets submissions, logged in | Transaction log | 1:N |
| 14. | Assignment | gets submission, logged in | Transaction log | 1:N |
| 15. | Transaction log | has | Transaction type | 1:1 |
| 16. | Element | originates from | Element | 1:1 |
| 17. | User | has access to | Folder | N:M |
| 18. | User | owns | Folder | 1:N |
| 19. | Folder | has a parent | Folder | 1:1 |
| 20. | User | creates | Submission | 1:N |
| 21. | User | asks for | Notification | 1:N |

Table 6.1: The relationship between entities

6.1.3 The Attributes

The entities have the following attributes:

User: The name, id, email and SMS of the user and his public key, which is needed when a file is encrypted and sent to him.

Element (Course, group, ...): The name and number of the element and the semester name.

Assignment: The time frame when a submission can occur and the name of the assignment.

Submission: The name of the report or exam file, the file itself, the file type and a true/false variables which tell if the file was encrypted and if it has been submitted yet.

Submission notification: Boolean variables that state if a user wants to get notifications on SMS or email, when a new submission is submitted to him.

Folder: The name of the folder.

Location: The name of the location. Location can for example be a building, a room or many rooms.

Address: The name and IP address of a computer.

Transaction log: Contains all the attributes needed to log each transaction in the system.

Transaction type: The name of the transaction.

Some of the relations have attributes on them. Here are the relations which have attributes:

A user **registered** in an element has the role as an attribute.

A user **gets an encrypted** submission has the encrypted file, the file type and the encrypted key as attributes.

A user **signs** a submission has the attributes date and location of the signature, the signature itself and the public key used to create the signature. The public key is used for verification in case of multiple signatures.

A user **gets** a submission has the attributes archived, which tells if the user has archived the file.

A user **views** a submission has the attribute date.

A user **comments on** a submission has the attributes date and the comment text.

A user **has access to** a folder has the attributes password and status of the access.

6.2 The Relational Model

The Relational model is a data model and is commonly used by Database Management Systems (DBMS). Data is stored in two-dimensional tables called relations. The table row is called a tuple and the column headers of the tables are called attributes. Every attribute value in a tuple has to be atomic, following the first normal form assumption.

The database is normalized with 3NF, which means that it fulfills the 1NF and the 2NF. To fulfill each normal form the following properties have to be met:

- 1NF: Eliminate duplicative columns from the same table. Create separate tables for each group of related data and identify each row with a unique column (the primary key).
- 2NF: There is no subsets of data that applies to multiple rows of a table. The data subsets should be placed in separate rows and the relationship between these new tables and their predecessors is through the use of foreign keys.
- 3NF: Columns that are mutually independent and fully dependent upon the primary key.

The ER model for the system is mapped into a Relational model by creating a relation for each entity and N:M relationship. One or more of the attributes are selected as a primary key. When a N:M relationship creates a relation, then foreign keys are added to both participating relations and the primary key is the combination of both of the foreign keys. Sometimes one or more of the attributes is also needed to form the primary key. When the relations have been created then the attributes of the entities and relationships are added to the relations. In the case of a 1:N relationship, then a foreign key is added to the N-side of the relation, to the other relation. Relationship attributes are added to the N-side relation.

The relations in the model represents a table in a database that has a name and some attributes. One or more of the attributes are the primary key and zero or more attributes are foreign keys. The relations will be described in the tables below. The primary key is displayed as bold and a foreign key is displayed in italic.

User

| | | |
|----------------|--------|-----------------------------------|
| user id | number | The unique user id, a primary key |
| first name | char | The user's first name |
| surname | char | The surname of the user |
| PID | char | The id of the user's certificate |
| email | char | The email of the user |
| SMS | char | The SMS number of the user |

Registered

| | | |
|-------------------|--------|--|
| id | number | The primary key |
| <i>user id</i> | number | The user that is registered |
| <i>element id</i> | number | The course a user is registered in on a certain semester |
| role | number | The role of the user, student, teacher or examiner |

Element (course, group)

| | | |
|-------------------|--------|---|
| element id | number | The primary key of the course on a certain semester |
| name | char | The name of the element |
| name2 | char | The name of the element in a second language |
| number | char | The unique number used to identify the element |
| semester name | char | The name of the semester |

Assignment

| | | |
|----------------------|---------|--|
| assignment id | number | The primary key |
| <i>element id</i> | number | The course that has the assignment |
| name | char | The name of the assignment, i.e. Assignment 1 or Final report |
| time from | date | The date/time when an assignment can be submitted from |
| time to | date | The date/time when an assignment can be submitted to |
| location id | number | The id of the location where the exam takes place |
| all group | boolean | Can the assignment be forwarded to all students in the course or just the ones in the same group within the course |

Location

| | | |
|-----------|--------|---------------------------------|
| id | number | The primary key of the location |
| name | char | The name of the location |

Address

| | | |
|-----------|--------|------------------------------------|
| id | number | The primary key of the address |
| ip | char | The IP address of the computer |
| name | char | The name of the address (computer) |

Location addresses

| | | |
|--------------------|--------|------------------------|
| id | number | The primary key |
| <i>location id</i> | number | The id of the location |
| <i>address id</i> | number | The id of the address |

Submission

| | | |
|----------------------|---------|--|
| submission id | number | The primary key |
| <i>assignment id</i> | number | The assignment the file is being submitted in |
| name | char | The name of the submission, the title of the assignment given by the student |
| file | blob | The file submitted |
| encrypted | boolean | Is the file encrypted |
| submitted | boolean | Has the file been submitted to the administrators |
| file type | char | The file type |
| <i>created by</i> | number | The user that created the submission |

Should get submission

| | | |
|-------------------|--------|--|
| id | number | The primary key |
| <i>user id</i> | number | The user that should get the submission |
| <i>element id</i> | number | A foreign key to the element that has the submission |

Submitted to

| | | |
|----------------------|---------|---|
| id | number | The primary key |
| <i>user id</i> | number | The user that receives a file for an assignment |
| <i>submission id</i> | number | A foreign key to the submission of the file |
| <i>folder id</i> | number | A foreign key to the folder, where the file is kept |
| archived | boolean | Has this user archived the file |

Gets encrypted

| | | |
|----------------------|--------|--|
| id | number | The primary key |
| <i>user id</i> | number | The user that receives an encrypted file for an assignment |
| <i>submission id</i> | number | A foreign key to the submission of the file |
| file | blob | The encrypted file |
| enc key | blob | The encrypted symmetric key used to decrypt the file |
| file type | char | The file type |

Signs

| | | |
|----------------------|--------|---|
| id | number | The primary key of the singature |
| <i>user id</i> | number | The user that signs a file for an assignment |
| <i>submission id</i> | number | A foreign key to the submission of the file |
| date | date | The date when the file was signed |
| location | char | The place where the file was signed |
| signature | blob | The information about the signature used to sign the document |
| public key | blob | The public key that should be used to verify the signature |

Views submission

| | | |
|----------------------|--------|--|
| id | number | The primary key |
| <i>user id</i> | number | The user that views a file for an assignment |
| <i>submission id</i> | number | A foreign key to the submission of the file |
| date | date | The date when the file was viewed |

Comments on submission

| | | |
|----------------------|--------|---|
| id | number | The primary key of the comment |
| <i>user id</i> | number | The user that comments on a submission |
| <i>submission id</i> | number | A foreign key to the submission of the file |
| comment | blob | The comment text for the submission |
| date | date | The date when the comment was created |

Submission notification

| | | |
|----------------|---------|---|
| id | number | The primary key of the notification |
| <i>user id</i> | number | The user that should get the notification |
| get SMS | boolean | Should an SMS be sent when a new submission arrives |
| get email | boolean | Should an email be sent when a new submission arrives |

Folder

| | | |
|------------------|--------|------------------------------------|
| id | number | The primary key of the folder |
| <i>user id</i> | number | The user that owns the folder |
| <i>parent id</i> | number | A foreign key to the parent folder |
| name | char | The name of the folder |

User has access to folder

| | | |
|------------------|--------|---|
| id | number | The primary key |
| <i>user id</i> | number | The user that has access to the folder |
| <i>folder id</i> | number | A foreign key to the folder |
| password | passw | The password the users agree on |
| status | bit | The status of the access, 0=revoked 1=granted |

Transaction log

| | | |
|----------------------|---------|--|
| trans id | number | The primary key of the transaction |
| <i>user id trans</i> | number | A foreign key to the user making the transaction |
| trans date | date | The date of the transaction |
| location | char | The IP address where the transaction was made |
| key | number | The key of the row that was changed |
| <i>type</i> | number | The transaction type, i.e. Submit, Forward, Edit Assignment |
| <i>user id</i> | number | The user id of the user that gets a submission, or a user id of the user that should receive submissions for assignments |
| assignment id | number | The assignment the transaction applies to |
| encrypted | boolean | Is the file encrypted, or should it be allowed to be encrypted |
| name | char | Name of the submission or assignment |
| signature | blob | The information about the signature used to sign the document |
| sign location | char | The place where the file was signed |
| time from | date | The date/time when an assignment can be submitted from |
| time to | date | The date/time when an assignment can be submitted to |
| location id | number | The id of the location where the exam takes place |
| all group | boolean | Can the assignment be forwarded to all students in the course or just the ones in the same group within the course |

Transaction type

| | | |
|-----------|--------|-----------------------------|
| id | number | The primary key of the type |
| name | char | The name of the transaction |

6.3 Summary

The database was designed by first creating an ER model and then by transforming that model into a Relational model. The relational model can easily be transformed into SQL commands. The SQL commands are then executed on the Database Server and the Database is ready to use. The SQL commands used to create the database can be found in Appendix B.3.

6.

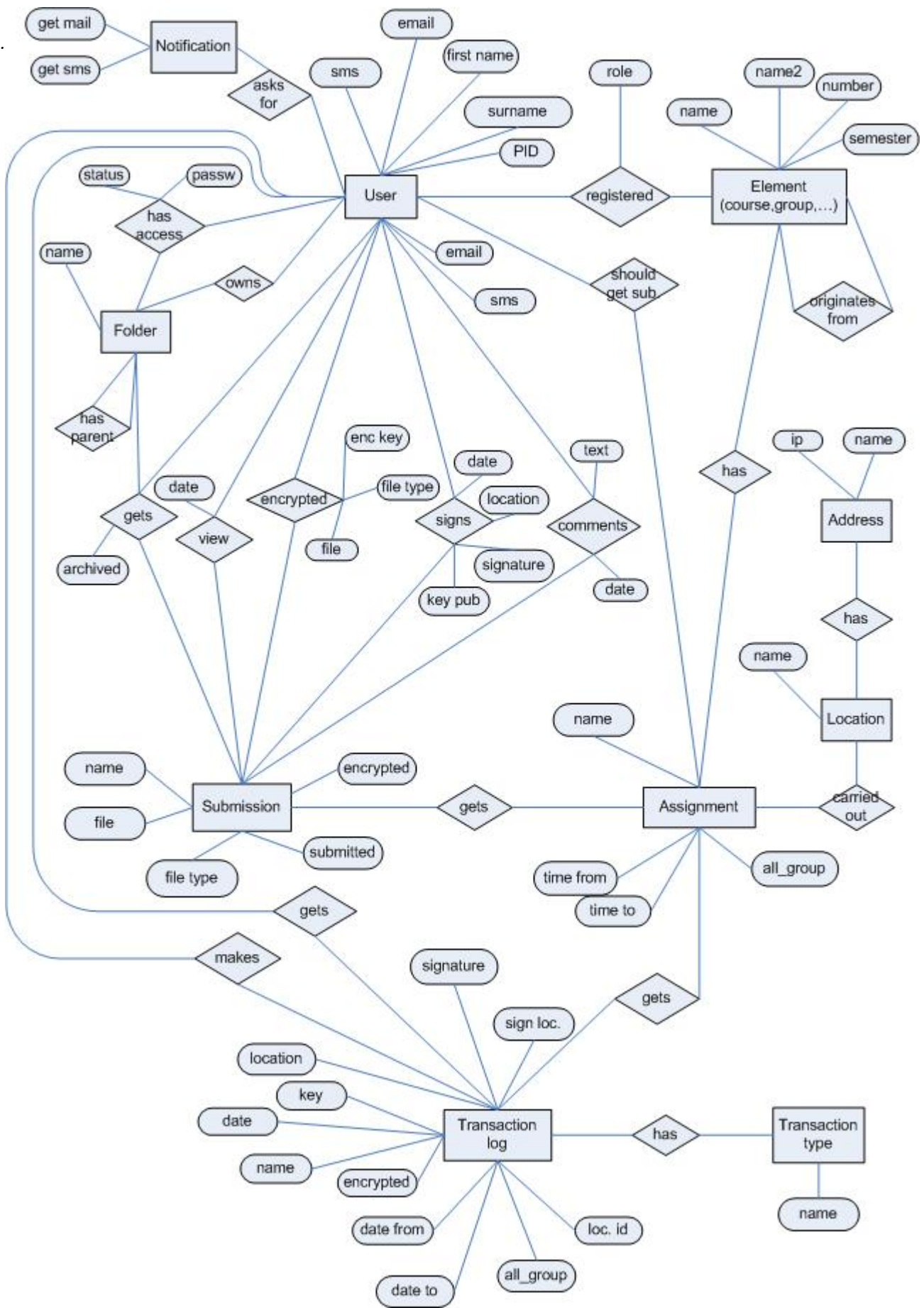


Figure 6.1: The ER diagram

Chapter 7

Security

This chapter will discuss how to make a Web based system, like the Report and exam submission system, secure. The first section introduces some security concepts. The second section describes attacks that could take place. It focuses on attacks on Web systems and the countermeasures that can be taken. The last section summarizes the chapter.

7.1 Security Concepts

Security is an important part of every computing system. Without security malicious parties could for example gain access to unauthorized actions or data, steal the computer hardware running the software, or delete the software itself. A basic security is important so that the software or hardware is not stolen, destroyed or modified by a malicious or an accidental action, even if the information kept in the system is not confidential.

In this section, the focus is on security in secure systems. A secure system is a system that protects an asset from unauthorized actions. An asset is a resource of value such as the data in a database or on the file system, or a system resource.

A secure system has three important aspects; Confidentiality, integrity and availability.

- Confidentiality means that only authorized parties can get access to the asset. This aspect is often referred to as privacy or secrecy.
- Integrity ensures that assets can only be modified by authorized parties and authorized actions. Modification includes writing, creating, editing and deleting.
- Availability implies that the system is available to authorized parties when it is needed.

The problem in building a secure system is to find the right balance between these three aspects. A system could preserve the confidentiality of an asset so well that it is not even available to authorized parties. This system is not secure because it does not ensure availability. A secure system has to ensure all three aspects to be considered secure.

7.2 Attacks

A person that exploits a vulnerability of a system perpetrates an attack on the system. An attack can also be launched by another system by sending messages that make the system unavailable. This is called Denial of service attack.

A vulnerability is a weakness in the system allowing unauthorized action. For instance, a system might be vulnerable to unauthorized user authentication because it does not check if the user enters an SQL command as the password and therefore gains access to the system. The user might enter `' or 1=1` as a password and gain unauthorized access.

A threat to a computing system is a set of circumstances that could allow unauthorized action. Threats can be initiated by persons, computers or the nature. Persons can cause errors, the hardware can malfunction and a flood or an earthquake can bring a system down. The threats can be divided into these four classes:

- **Interception:** An unauthorized party gains access to an asset. For example wiretapping to obtain data in a network and bypassing the user authentication by inserting SQL commands as passwords.
- **Interruption:** An asset becomes lost, unusable or unavailable. Examples of this type are destruction of a hardware, malfunction of an operating system, or erasure of a program or a data file.
- **Modification:** An unauthorized party gains access to the system and tampers with the assets. He could for example alter values in the database.
- **Fabrication:** An unauthorized party can create a fabrication of objects in a computer system. For example add record to a database or insert a transaction to a network communication.

7.2.1 Web System Vulnerabilities

Web attacks are attacks that use the HTTP/HTTPS protocol. Attackers exploit vulnerabilities in the Web server software or in the Web application itself.

All Web server software has hidden vulnerabilities. When vulnerabilities are found in Web servers, a patch is usually released that corrects the error. After a vulnerability has been discovered and made public, malicious parties can exploit the vulnerability in servers that have not yet been patched.

Attacks on the Web application exploit vulnerabilities in the code of the application. The vulnerability may be errors in HTML forms, scripts, SQL sentences, etc.

There are limited number of types of web attacks. Classification can help software developers to understand attacks and build more secure systems. The vulnerabilities of a Web applications can be classified into the following classes [AlPe03]:

- Code injection. The user injects a user-chosen code into a page. This vulnerability is caused by none or poor input validation. Code can be injected in the following way:
 - Script injection. This attack inserts new, malicious script that can execute on the victims browser, the victim thinks the script comes from the server he was communicating with. The attacker could get information like credit card number from the victim. [Mi00].
 - SQL injection. The attacker changes or alters a SQL command. He might get unauthorized access to data or execute system level commands on the host. He could for example drop tables and create new tables. [An02,Ce02].
 - XPath injection. XPath addresses parts of an XML document. An attacker can modify search strings and access unauthorized data in XML document [Kl04].
- Canonicalization. This occurs when security decisions are made based on a name, i.e. filename or folder name. The attacker can change URL input parameters, cookies, or HTTP request headers and gain access to commands and data he should not have access to.
- HTML manipulation allows a malicious user to modify data, that the user was not intended to have access to. The data is then sent from the browser to the application.
- Buffer overflows happens when a program writes past the bounds of the buffer. This can cause a number of different behaviors. The program could do something unexpected or fail completely.
- Misconfiguration happens when the platform and the Web server are not correctly configured. Attackers can exploit vulnerabilities that arise because of this.

User authentication

Authentication attacks in a web system aim at bypassing identification controls. This can be done by using a number of different techniques, some of them have been mentioned earlier. For example wiretapping can be used to intercept plaintext passwords, and SQL injection can be used instead of name or password to gain unauthorized access. Sessions are commonly used in Web systems and there are number of attacks that aim at exploiting sessions to get access. Here are some authentications attacks that can be made against Web systems:

- Session hijacking entails connecting to a Web site and accessing someone else's session state. This attack is usually perpetrated by session ID guessing or by stealing session ID cookies. It is not feasible for attackers to try to guess session IDs when large random numbers are used as a session ID. Session ID cookies can be stolen by number of ways, for example with man in the middle attack, cross-site scripting and network monitoring software to capture the cookie. If a session cookie has been stolen it is not possible to detect if it is an attacker or an authorized user that is using the cookie. The attacker can spoof the session of the user and gain access to the application with the same privileges as the user. The session state can contain confidential information like a credit card number and therefore it is important that unauthorized parties cannot gain access to it.

- Session replay occurs when a session ID of a user is intercepted by an attacker. The attacker then uses the session ID to bypass user authentication.
- Session fixation involves that an attacker fixes the session ID before the user logs into the target server [Ko02].
- Identity spoofing is when an attacker uses stolen credentials or false IP addresses to gain access to a system.
- Brute force attacks rely on computational power to crack passwords.
- Gaining the credentials of a user by monitoring the user as he types in the name and password. The attacker can thereafter use the name and password to gain access to the system.

7.2.2 Other Vulnerabilities

The vulnerabilities discussed in the former section cover data and software vulnerabilities concerning the Web application. There can also be vulnerabilities in the hardware, the application is running on, and the software supporting the application, for example in the software of the web server or the operation system.

Hardware is composed of physical objects and is therefore vulnerable to physical access, where devices are added, changed or removed. Water, fire and other physical objects can damage the hardware.

7.2.3 Methods of Defense

There are numerous ways for an attacker to attack a system. Fortunately the system developer and the system administrator also have many countermeasures or controls [Mi04]. Some controls can prevent an attack or deter it by making the attack harder for the attacker. If an attack has been successfully carried out then controls to detect the attack and recover from its effect are useful.

Controls for the system developer

The input from the user should never be trusted. All user input should be validated to prevent script, SQL, XPath injection and buffer overflow.

- HTML Encode and URL Encode functions should be used to encode any output that includes user input. This converts executable script into harmless HTML.
- Use parameterized stored procedures when accessing the database. Then the statements are not treated as executable statements. An alternative is to use SQL parameters when building SQL commands.
- Constrain input by validating it for type, length, format and range to prevent buffer overflow.

Brute force and wiretapping are examples of attacks made to try to get information about user's password. The following controls aim at preventing the success of such attacks:

- Passwords that are transmitted over a network should not be plaintext, use HTTPS or encrypted communication channel like SSL.
- Enforce strong passwords. Passwords should not be regular words, they should contain a mixture of upper and lower case, numeric, and special characters. The minimum password length should not be less than 8 characters.
- Passwords have limited lifetime, the user has to change it within a maximum time.
- Store the password history, a number of previous passwords must not be reused when a password is changed.
- Store passwords as one way hashes with added salt.
- Close the user account after a set number of login attempts.

Cookies are often used to manage the session of the user in a Web system. The following controls aim at preventing session replay and session hijacking:

- Use SSL whenever an authentication cookie is transmitted.
- Use cookie timeout so the user has to authenticate himself within a certain time.

The following controls are used to keep data confidential. Even if the attacker retrieves the data he is not able to understand its content.

- Use the HTTPS protocol so that malicious users cannot make use of confidential data that is intercepted with wiretapping or network monitoring tools.
- Store confidential and sensitive data in files and databases by using standard encryption.

Finally, here is a number of various useful controls:

- Use accounts with low privileges to connect the application to the database. If a malicious party should gain unauthorized access he can only cause limited harm.
- Avoid input file names if possible to avoid canonicalization. Use absolute file paths that cannot be changed by the user.
- Create a functionality so that user credentials are not put in the browser cache. This prevents credential theft.
- Perform role check before allowing access to actions that require certain user privileges.

Control for the system administrator

This section has discussed the controls the system developer has. But there are also many controls that the system and network administrators have, to make a system secure. The most important control for the hardware is physical controls, for example locks on the door and guard limiting access to the hardware.

The administrators have a number of software controls to ensure the applications run successfully. Here is a number of important controls:

- Use an Intrusion detection systems to detect and prevent Denial of service attacks.
- Use an Anti virus system to scan for viruses.
- Use firewalls and block all unnecessary ports at the firewall.
- Apply the latest server packs and patches.
- Configure routers to restrict their responses to footprinting requests. This prevents the attacker from getting too much information about the server.
- Prevent footprinting by disabling unused protocols and unnecessary ports, if the operating system hosts network software.
- Weak, default configuration settings should be hardened.

7.3 Summary

This chapter has discussed the concept of security in computer systems. It focused on attacks on Web systems and countermeasures that can be taken. People using the countermeasures have to be aware of the need for security. The application is not secure if the people using it do not follow the security policy and use the controls available. There is no use in having a lock on the door if nobody uses it. It is also useless to have strong passwords if the users generally write it down and put it under the keyboard. Security can be no stronger than its weakest link.

The prototype was implemented with focus on security and the following countermeasures were taken to create a secure system. All user input was encoded and validated. Before showing some data to the user, a check was done to see if he had access to the data. Otherwise the user might change some parameters and get access to data he should not have access to. Queries to the database were executed using SQL parameters. The connection to the database was through an account with low privileges and the initial database accounts were secured by assigning a password to them. The real system will be set up on a HTTPS server.

Chapter 8

Implementation

This chapter describes how the prototype for the Report and exam submission system was implemented. The most important aspects of the classes created are described. A reference is made to important packages that were used by the classes.

The first section describes how the Java applet that runs on the client side, and the Web user interface were implemented, an example of a simple JSP page is demonstrated. The second section describes the classes that run on the application server, those classes are often used by the JSP page. The third section discusses how the database was set up. The fourth section describes the signing process and the final section summarizes the chapter.

8.1 The Client

The system is a Web system which is written in JSP and Java. It has a Java Applet to sign and decrypt documents. The applet runs on the client and passes information to the server. The Web Interface is created by using JSP pages and Java servlets.

8.1.1 The Applet

The applet extends an applet created by the OpenOCES project [OpOc]. The OpenOCES applet allows a user to enter a text that should be signed, select a certificate file and enter the password for the certificate.

The applet that was created enables users to locate a file that should be signed, instead of just signing text like is done in the OpenOCES applet. The applet also shows information about the submission that is about to be signed. The same applet is used, with minor alterations, to decrypt the document. The applet looks almost the same. Some labels are changed and it has a Decrypt button instead of a Sign button.

Below is a description of the classes that are used to implement the applet. The source code is on a disk that was handed in with the report, see details in appendix B. The path to the files is: Client/org/assignmentdocs/client.

The class AbstractApplet

This class is a copy of the class org.openoces.opensign.client.applet.AbstractApplet, created by the OpenOCES. This class is extended by the applet used to sign and decrypt documents.

The class AesEncrypter

The class is located in the package org.assignmentdocs.common and is used by both the Applet and the classes running on the server side. The AesEncrypter uses an instance of the class javax.crypto.Cipher for the algorithm AES. The Cipher is initiated with a SecretKey. AesEncrypter has methods to encrypt and decrypt streams of data.

The class CryptoTools

Contains methods to sign and verify documents, along with some help functions. The algorithm used to sign and verify is SHA1withRSA, which is the algorithm used by the OCES certificates provided by TDC in Denmark. At this stage only PKCS12 certificates can be used to sign documents. Support for Windows Certificates is allowed for and can be added later.

The class DecryptedXML

Has a method to decrypt a file, which is called when the user presses the Decrypt button on the applet. Retrieves the encrypted data, the encrypted symmetric key and the signatures from a XML file. The XML file has to be downloaded by the user beforehand. The symmetric key is decrypted with the private key of the user. The AesEncrypter is initialized with the decrypted key and the file is decrypted by using the AesEncrypter. Then the signatures attached to the file are verified. The decrypted file is saved on the local drive and the user is made aware of the path to the decrypted file. The class uses classes from the package org.w3c.dom to read the xml file.

The class FileLog

A simple multithreaded safe file logger. Used for debugging purpose.

The class MultipartServletPost

Creates the multipart/form-data request sent to the server. The request is created with parameters read from the applet when the Sign button has been pressed. The classes URLConnection and HttpURLConnection, from the package java.net are used to create a connection from the applet to a servlet on the server.

The class SignDecrypt

The applet used to sign or decrypt documents. Extends the AbstractApplet class and adds some buttons, labels and text fields.

The class SignListener

This class implements the ActionListener for the buttons in the SignDecrypt applet, i.e. the action

that is taken when each button in the applet is pressed.

8.1.2 The Web Interface

The Web Interface is created by using JSP pages, Java classes, Cascading Style Sheets and JavaScript.

The basic JSP page is organized by first importing the Java packages that should be used on the page. A check is done to see if the user is logged on and the header of the page is printed out. If the page connects to a database, then a connection to the database is made and a query is executed. The result of the query is printed out along with the HTML and JavaScript code, to create the user interface. Finally the last part of the page is written.

Some parts of the JSP pages are common with many pages, like checking if the user is logged on, printing the header, encoding the request parameter and getting the connection to the database. Those functions are implemented using Java classes and used by the JSP pages. The Java classes run on the server and are discussed further in section 8.2.

The following is an example of a simple JSP page that connects to the database:

8.2 The Application Server

This section describes the Java classes that run on the Application Server and contain methods that can be used by the JSP pages that create the user interface. The source code is on a disk that was handed in with the report, see details in appendix B. The path to the files is: `Server/org/assignmentdocs/server`.

The class `AesEncrypter`

The class is located in the package `org.assignmentdocs.common` and is used by both the Applet and the classes running on the server side. The `AesEncrypter` uses an instance of the class `javax.crypto.Cipher` for the algorithm AES. The Cipher is initiated with a `SecretKey`. `AesEncrypter` has methods to encrypt and decrypt streams of data.

The class `ConnectionAssignmentDocs`

This class has a function that returns an instance of the connection to the database. The class uses a MySQL driver to connect to the database. The class for the `DriverManager` is `org.gjt.mm.mysql.Driver`. The Jar file containing this class can be downloaded. If the database is changed in any way, then this and only this class needs to be altered.

The class `EboxMenu`

Contains methods that manage the folder structure created for the e-box of the user. The folder structure is used to create the tree structure in the menu, and also when folders are edited or when files are moved between folders.

```

<%@ page import="java.sql.*,org.assignmentdocs.server.*" %>
<%
    // Check if the user is logged on
    Init.loggedOn(session,request);

    // Print out the header of the page, common with all pages in the system
    Init.renderTop(session,out);

    // Get the encoded parameters, if the parameter is a text then it is decoded again.
    String paramNumeric = URLEncoder.encode(request.getParameter("Id"));
    String paramText = URLEncoder.enDeCode(request.getParameter("Text"));

    // Get a connection to the database
    Connection con = ConnectionAssignmentDocs.getConnection();

    // Execute a database query
    String sql = "select first_name from user where id = ?";
    PreparedStatement pstmt = con.prepareStatement(sql);
    pstmt.setString(1,paramNumeric);
    ResultSet rc = pstmt.executeQuery();

    // Print out the result
    if ( rc.next() ) { %>
        <p><%= rc.getString(1) %></p>
    }

    // Print out the final part of the page, common with all pages in the system
    Init.renderBottom(out);
%>

```

Figure 8.1: A simple JSP page with database connection

The class EncryptedFileXml

Creates an XML file that can be downloaded when the file has been encrypted. If the file has been encrypted then it is not enough to download just the encrypted file, if the file is to be decrypted. The encrypted symmetric key used to encrypt the file should also be attached along with the signatures of the file. Only the recipient of the file should be able to decrypt the encrypted key and use it to decrypt the file. When the file has been decrypted then the signatures can be verified. The decryption and the verification is done in the Java Applet on the client side.

The class Execute

The class has methods to update, insert and delete records from tables. Classes located in the package `java.sql` are used to execute queries in the database. SQL parameters are used when building SQL commands. This ensures that malicious code entered by the user cannot cause harm in the database.

The class FileLog

A simple multithreaded safe file logger. Used for debugging purpose.

The class Init

The class contains various methods that are in common with many JSP pages. It has a method that checks if the user is logged on, and a method that prints out the common header and footer of the user interface. If the user interface should be changed then it is sufficient to change those methods and the style sheet for the system.

The class SignServlet

The client side Java Applets connects to this Servlet with a multipart request. The class uses `com.oreilly.servlet.MultipartRequest`. The Jar file containing the class can be downloaded. The `MultipartRequest` class is used to get the parameters from the multipart request along with the file that should be uploaded.

A few checks are done in the Servlet before the submission is created or edited. If the submission already exists and the file is not encrypted then the existing signatures are verified to ensure that the file has not been changed and all the signatures sign the same document. If the file should be encrypted then a check is done to make sure that the recipient has a registered public key.

The class Submission

The class has methods that create the submission according to the parameters that are passed from the signing Applet. If the submission does not exist then it is created. Then the new signature is attached to the document and the recipients are registered.

If the document should be encrypted then it is encrypted in this class, one encryption is done for each recipient. It is important to encrypt the document at the server side because then the file can be scanned for viruses before it is encrypted. The file is encrypted using AES encryption. A symmetric AES key of size 128 bits is created and the `AesEncrypter` is initialized with the key. Then the file is encrypted using the `AesEncrypter`. The symmetric key is encrypted with the public key of the recipient. Both the encrypted file and the encrypted key are stored and can be retrieved by the recipient when he wishes to decrypt the file.

The class URLEncoder

Contains functions to encode and decode text entered by the user. A malicious code entered by the user is turned into harmless code by using the encoding.

8.3 The Database

The database was created by using SQL commands derived from the Relational Model created during the database design explained in chapter 6. The database used in the development is a MySQL server and the SQL sentences are to some extent depended on that. Some modifications have to be made if the database is changed, e.g. into Microsoft SQL or Oracle. The modification would be done on the SQL sentences and on the `ConnectionAssignmentDocs` class that runs on the server.

8.3.1 Archiving documents

A problem with a database system that keeps information that cannot be deleted is that the database gets bigger and bigger every day. The data that is no longer used, but should still be kept should be archived. The data has to be available when it is needed. One way to solve this problem is to create a procedure that selects unused data out of the tables and writes it to files. If the data should be made available then it is restored from the files into the database. The uploaded files could be written to disk, but the submission record would still be in the database. The file takes up the most of the memory in the database.

8.4 The signing process

This section will discuss the flow in the signing process. The multiple signatures are sequential. One student initiates the signing process, and then the file is forwarded to zero or more students. Each student can only forward the file to one other student. The student that signs the documents last, submits the file to the administrators of the course.

The multiple signatures could also be parallel. One student initiates the signing process and he is in charge of the submission, he selects all the students that should sign the document. The students that receive the document sign it. When some or all students have signed the document, the student that initiated the submission can submit the document. The parallel signatures were not implemented in the prototype at this stage.

The constructor in class `Submission` manages the submissions. When the first student signs the submission then a new submission is created and thereafter the first signature is attached. If the submission has been signed before then the next signature is attached, but the submission itself is left unchanged. Only one version of the file is stored in the database, except if the file should be encrypted. Then an encrypted version for each recipient has to be kept. If the file is being forwarded then the user that should sign the file next, gets a link to the file. If the file is being submitted then all the administrators that should get files for the assignment, get a link to the file. A link from a user to a file is created by inserting a record into the table `user_gets_submission`, which connects the user to the submission.

When a user wants to view or manage his submissions then a query is made to the database to find all the submissions the user has access to. The user has access to the submissions he has a link to through the table `user_gets_submission`.

8.5 Summary

The user interface of the prototype that was implemented is described in appendix A. The user interface uses style sheets which makes it easy to change the look of the interface. The style sheet controls for example the colors, font sizes and table layouts of the interface.

The prototype was tested in a systematic way by the author and the test descriptions can be found in appendix C. The idea was to set the prototype up on a server and allow some users to test it. One problem followed the other in the process of getting the Tomcat to run correctly, and the server could not be set up in time so users could test it. The prototype was therefore not tested by potential users, but it was demonstrated to several people including a number of teachers at DTU.

The source code for the prototype is on a CD-ROM that was handed in with the report, see details in appendix B.

Chapter 9

Future work

I was interested in doing a thesis that involved computer security and Robin Sharp came up with the idea of creating a system that would allow students to submit exam and report documents electronically in a secure manner. The goal of the project was to create a prototype of such a system. The size of the masters project did not allow for the whole system to be implemented. The following improvements are important before the system can be used as a real system.

- Test the system for various operating systems, certificate files and browsers. The prototype works for Windows XP, PKCS12 certificates and IE 6.0 browser.
- Allow the use of Windows Certificates, the prototype only supports PKCS12 Certificates.
- Check the revocation list to see if the certificate has been revoked.
- Add multi language support. The multi language support has been implemented, but not all text strings in the system have been replaced with a resource variable.
- Test that the system can handle many connections to the database at a time.
- Create the archiving process, that archives old files that are not used anymore.
- Scan files for viruses, before they are uploaded from the user.
- The focus of this project has been on report submission. The system can be used for exam purpose if the location, where the exam is carried out, is attached to the assignment. This has been discussed but not yet implemented.
- Implement the transaction log. Find some convenient tools to browse and get information from the log files.
- Implement some ideas that came up at the last minute and could not be implemented at this stage. These ideas are described in the use cases, but not yet implemented.
 - Multiple signatures should be parallel, one student is the master student and forwards the document to one or more students. The master student then submits the document when he thinks it is appropriate. The document could also be submitted automatically when all students have signed it.

- The user of the e-box can select one or more files and download them to the local drive.

If the prototype is extended with the work mentioned above then it would become a secure and efficient Report and exam submission system, that could be used by universities and other educational institutions. The system could also be used to handle other kinds of document submissions.

Appendix A

The user interface

This chapter describes the user interface of the prototype that was implemented. Pictures of the frames for each interface are shown, and a reference is made to the use cases that was implemented in the particular interface. The path to the JSP page used to create the interface is written. The source code is on a disk that was handed in with the report, see details in appendix B.

The first section describes the interface for the student, the second section describes the interface for the course's administrators and the third section describes interfaces that are in common with all users of the system.

A.1 The interface for the student

This section describes the interface for the student using the system.

A.1.1 The submission list

When a student logs in he sees the list of the documents he has submitted, and a list of the submissions that have been signed by other students and forwarded to him. He has the option to Submit and Forward new documents. The student can delete submissions that have been forwarded to him if he does not wish to sign them. This implements the use case in sections 4.5.5 and 4.5.10. The path to the source code is: Webapp/assignmentdocs/submission/submissions.jsp.

The screenshot shows the 'Report submission' page for a student at Danmarks Tekniske Universitet. The page includes a header with the university name and user information (Thomas Lund). Below the header, there is a section for reporting a submission, with a dropdown menu showing '02220 Concurrent Systems (v03)' and buttons for 'Submit to teacher(s)' and 'Forward to student'. A paragraph explains that reports and exam documents can be submitted to teachers or administrators, or forwarded to a fellow student. Below this, there is a section for 'Signed documents from other students' with a 'Delete' button and a 'My settings' link. A table lists signed documents with columns for Course/Group, Assignment, Submission, Signed by, Details, Forward, and Submit. The table contains one entry: '02224 Real-Time Systems' for 'Assignment 1' with submission 'A1', signed by 'Mansbil Hiakelin' on '11.03.2005'. At the bottom, there is a section for 'My submissions' with a table listing the student's own submissions. The table has columns for Course/Group, Assignment, Submission, Signed, and Details. It contains one entry: '02220 Concurrent Systems' for 'Report 1a' with submission 'R1a', signed on '11.03.2005'.

| Course/Group | Assignment | Submission | Signed by | Details | Forward | Submit |
|--|--------------|------------|-----------------------------|-------------------------|-------------------------|------------------------|
| <input type="checkbox"/> 02224 Real-Time Systems | Assignment 1 | A1 | Mansbil Hiakelin 11.03.2005 | Details | Forward | Submit |

| Course/Group | Assignment | Submission | Signed | Details |
|--------------------------|------------|------------|------------|-------------------------|
| 02220 Concurrent Systems | Report 1a | R1a | 11.03.2005 | Details |

Figure A.1: The main page for the student

A.1.2 View the submission details

The student can view the details of a submission that he has submitted or has been submitted to him. This implements the use case in section 4.5.9. The path to the source code is: Webapp/assignmentdocs/submission/details.jsp.

The screenshot displays the 'Submission details' page. At the top, it says 'DANMARKS TEKNISKE UNIVERSITET'. Below that, there are links for 'Help | Log out' and the user's name 'Marsibil Hjaltalin'. The main content is divided into sections: 'Submission information', 'Signed by', 'Document forwarded to, but not yet signed by', and 'Document submitted to'. The 'Signed by' section contains a table with columns for ID, Name, Date, and Location.

| Submission information | | | |
|--|-------------------------------|------------------|----------|
| Course | 02224 Real-Time Systems | | |
| Assignment | Assignment 1 12.12.2004 09:00 | | |
| Document | A1 Download | | |
| FileType | pdf | | |
| Signed by | | | |
| | Name | Date | Location |
| e031643 | Marsibil Hjaltalin | 11.03.2005 10:37 | Lyngby |
| s3 | Thomas Lund | 11.03.2005 10:41 | Gledeexe |
| Document forwarded to, but not yet signed by | | | |
| | | | |
| Document submitted to | | | |
| | v1 Anker Enkalund | | |

[Back to submissions](#)

Figure A.2: The details of the submission

A.1.3 Forward to another student

When a student forwards a document to another student he begins with entering the submission details, the second step consists of selecting the file that should be signed and the certificate used to sign the document. Finally the student confirms the submission. This implements the use case in section 4.5.6. The path to the source code is: `Webapp/assignmentdocs/submission/submission_submit.jsp`.

DANMARKS TEKNISKE UNIVERSITET

Report submission Help | Log out
Marsibil Hjaltalin

Forward a document to a fellow student for further signing

Enter submission details -> Sign -> Confirm

| Submission | |
|-----------------|---|
| Course | 02224 Real-Time Systems |
| Assignment | Assignment 1 |
| Assignment name | AI |
| Location | Lyngby |
| Encrypt | <input checked="" type="radio"/> No <input type="radio"/> Yes |

| Submit document to | |
|--------------------|--|
| Recipient | <input type="checkbox"/> s2 Per Pseudonym |
| | <input checked="" type="checkbox"/> s3 Thomas Lund |
| | <input type="checkbox"/> s1 Waldemar Nielsen |

Figure A.3: The signing process: Enter the submission details

DANMARKS TEKNISKE UNIVERSITET

Report submission Help | Log out
Marsibil Hjaltalin

Enter submission details -> **Sign** -> Confirm

Course: 02224 Real-Time Systems
Assignment: Assignment 1
Assignment title: A1
Location: Lyngby
Submit document to: s3 - Thomas Lund

File to sign:
Certificate:
Password:

Figure A.4: The signing process: Signing the document

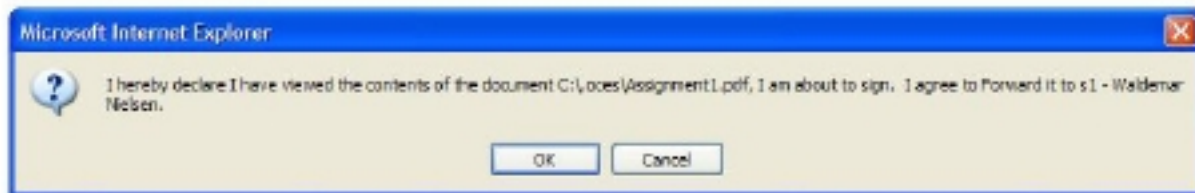


Figure A.5: The signing process: Confirming the submission

A.1.4 Submit to the administrators of the course

When a student submits a document to the administrators of the course he begins with entering the submission details, the second step consists of selecting the file that should be signed and the certificate used to sign the document. Finally the student confirms the submission. This implements the use case in section 4.5.6. The path to the source code is: `Webapp/assignmentdocs/submission/submission_submit.jsp`.

The figure belows shows a submission that has been forwarded from another student.

The student then signs the documents and confirms the submission, see figures A.4 and A.5.

DANMARKS TEKNISKE UNIVERSITET

Report submission Help | Log out
Thomas Lund

Submit a document

Enter submission details -> Download the file -> Sign -> Confirm

Submission

Course: 02224 Real-Time Systems
 Assignment: Assignment 1
 Assignment name: A1
 Location: Gedssø
 Encrypt: No
 FileType: pdf

Download document

Download: The file is in database, you should download it and then sign it in the next step. If the file is changed, then the submission will not be successful. [Download](#)

Figure A.6: The signing process: Enter the submission details

A.2 The interface for the teacher

This section describes the interface for the teachers and the external examiners.

A.2.1 The e-box

When a user log is he sees his e-box and the operations he has access to. This implements the use case in section 4.5.12. The path to the source code is: Webapp/assignmentdocs/ebox/ebox.jsp.

DANMARKS TEKNISKE UNIVERSITET

E-box Help | Log out
Anker Enkelund

My settings
 My access
 Access of others
 Assignment settings

New Folder Edit folders

- Inbox
- Robin's folder
- Testing
- Archive

Files received from students in:

Inbox All files

Search for text:

in Inbox or All files

Inbox

| <input type="checkbox"/> | Course | Assignment | Signed by | Functions |
|--------------------------|--------|--------------|--|-----------|
| <input type="checkbox"/> | 02220 | Report Ia | Thomas Lund 11.03.2005 | More |
| <input type="checkbox"/> | 02224 | Assignment 1 | Margibil Hiakalin 11.03.2005 Thomas Lund 11.03.2005 | More |

Figure A.7: The main page for the teacher and external examiner

A.2.2 View the submission details

The user can view the details of the submission that has been submitted to him. This implements the use case in section 4.5.13. The path to the source code is: Webapp/assignmentdocs/ebox/details.jsp.

The screenshot displays the 'Submission details' page within an 'E-box' interface. At the top, the header includes 'DANMARKS TEKNISKE UNIVERSITET' and navigation links for 'Help | Log out' and the user name 'Anker Enkelund'. The main content is organized into several sections:

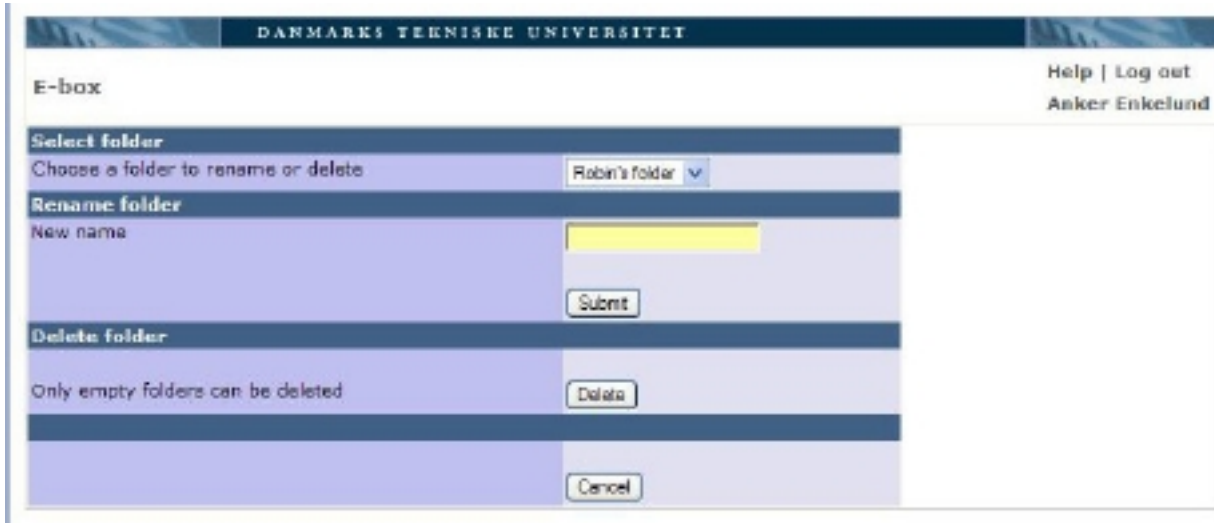
- Submission information:** A table listing course details: Course (02220 Concurrent Systems), Assignment (Report 1a 01.12.2004-04.12.2004), Document (R1a [Download](#)), and FileType (pdf).
- Signed by:** A table with columns for Name, Date, and Location. It shows a signature from 's3 Thomas Lund' dated '11.03.2005' from 'Herlev'.
- Comments:** A table with columns for Comment, Date, Commented by, and Delete. A 'Create comment' link is visible at the bottom right of this section.
- Document submitted to:** A list showing two versions: 'v2 Robin Sharp' and 'v1 Anker Enkelund'.

A 'Back to E-box' link is located at the bottom right of the page.

Figure A.8: The details of the submission

A.2.3 Manage folders

A user can insert, edit and delete folders in his folder structure. This implements the use cases in sections 4.5.14 and 4.5.15. The path to the source code is: `Webapp/assignmentdocs/ebox/folder_edit.jsp` and `Webapp/assignmentdocs/ebox/folder_insert.jsp`.



The screenshot shows a web interface for managing folders. At the top, there is a header with 'DANMARKS TEKNISKE UNIVERSITET' and 'Anker Enkelund'. Below the header, there is a section titled 'E-box' with 'Help | Log out' and 'Anker Enkelund' links. The main content area is divided into three sections: 'Select folder' with a dropdown menu showing 'Robin's folder', 'Rename folder' with a text input field and a 'Submit' button, and 'Delete folder' with a 'Delete' button and a 'Cancel' button at the bottom.

Figure A.9: Edit or Delete a folder

A.2.4 Move files

The user can move files from one folder to another. This implements the use case in section 4.5.16. The path to the source code is: `Webapp/assignmentdocs/ebox/move.jsp`.



The screenshot shows a web interface for moving files. At the top, there is a header with 'DANMARKS TEKNISKE UNIVERSITET' and 'Anker Enkelund'. Below the header, there is a section titled 'E-box' with 'Help | Log out' and 'Anker Enkelund' links. The main content area has a section titled 'Move selected files' with a dropdown menu showing 'Robin's folder' and 'Move' and 'Cancel' buttons.

Figure A.10: Move files between folders

A.2.5 Access of others

The user can see the users that he has granted access to the folders in his e-box, he can grant new access and revoke access that has been granted. This implements the use case in section 4.5.17. The path to the source code is: Webapp/assignmentdocs/ebox/access.jsp.



Figure A.11: Access of others to the users e-box

A.2.6 The access of the user

The user can view the folders he has access to in the e-boxes of other users. This implements the use case in section 4.5.20. The path to the source code is: Webapp/assignmentdocs/ebox/my_access.jsp.



Figure A.12: The access of the user

A.2.7 Assignments of the course

The user can view and manage the assignment settings for the courses he administrates. This implements the use case in section 4.5.29. The path to the source code is: Webapp/assignmentdocs/ebox/assignments.jsp.

The screenshot displays the 'View assignment settings' page. At the top, the university name 'DANMARKS TEKNISKE UNIVERSITET' is visible. The page is titled 'E-box' and includes a user profile 'Anker Enkelund' with 'Help' and 'Log out' links. A dropdown menu shows the selected course '02220 Concurrent Systems (v03)' and a 'View settings' button. The 'Information' section provides details about the course: Course (02220 Concurrent Systems), Semester (v03), Super course/group, and Sub courses/groups. The 'Assignments' section is a table with the following data:

| Name | From | To | Submitted to | Edit |
|-----------|------------------|------------------|-------------------------------------|------|
| Report 1a | 01.12.2004 09:00 | 04.12.2004 06:14 | v1 Anker Enkelund v2 Robin Sharp | Edit |
| R2 | | | v2 Robin Sharp | Edit |

At the bottom right of the table, there is a link for 'New assignment'. At the bottom center of the page, there is a link for 'Back to E-box'.

Figure A.13: View assignment settings

The screenshot shows the 'Edit assignment settings' page. At the top, there is a header for 'DANMARKS TEKNISKE UNIVERSITET' and a user profile for 'Anker Enkelund' with links for 'Help' and 'Log out'. The main content area is titled 'Assignment settings' and includes the following fields:

- Course/group:** 02220 Concurrent Systems
- Assignment name:** A yellow input field.
- Time limits for assignment (not required):**
 - Date from:** dd.mm.yyyy and time (hh:mi) [input]
 - Date to:** dd.mm.yyyy and time (hh:mi) [input]
- Submit to:**
 - v1 Anker Enkelund
 - v2 Robin Sharp

At the bottom of the form are 'Submit' and 'Cancel' buttons.

Figure A.14: Edit assignment settings

A.3 Common interface

This section describes the interface all actors of the system have.

A.3.1 The settings of the user

The user can view and change his settings. This implements the use case in section 4.5.3. The path to the source code is: `Webapp/assignmentdocs/ebox/settings.jsp` and `Webapp/assignmentdocs/submission/settings.jsp`.

The screenshot shows the 'The user's settings' page. At the top, there is a header for 'DANMARKS TEKNISKE UNIVERSITET' and a user profile for 'Anker Enkelund' with links for 'Help' and 'Log out'. The main content area is titled 'E-box settings' and includes the following fields:

- Get notification:**
 - Get email:** No Yes. Send to address: anker@test.com
 - Get SMS:** No. No SMS number registered

At the bottom of the form are 'Submit' and 'Cancel' buttons.

Figure A.15: The user's settings

Appendix B

Source code

The source code can be found on a CD-ROM that was handed in with the report.

B.1 Client

The Java files that implemented the applet which signs and decrypts documents can be found at paths:

Client/org/assignmentdocs/client

Client/org/assignmentdocs/client/resources

Client/org/assignmentdocs/common

B.2 Server

The Java files that were implemented and run on the server can be found at path: Server/org/assignmentdocs/server

The JSP files that create the user interface can be found at path: Webapp/assignmentdocs/

- The JSP files for the student's user interface can be found at path: Webapp/assignmentdocs/submission
- The JSP files for the teacher's user interface can be found at path: Webapp/assignmentdocs/ebox
- The stylesheet for the Web interface can be found at path: Webapp/assignmentdocs/css
- The JavaScript file for the system can be found at path: Webapp/assignmentdocs/script

B.3 Database

The path to the SQL file needed to create the database is: DB/Database.sql

Appendix C

Testing

The system was tested with a structural test and a functional test. The structural test is also called a White Box test. The test cases are discovered by going through the code to find each case that could arise. The functional tests were designed for each actor, one for the student and one for the teacher. The functional test tested all the buttons and links in the system.

C.1 User Tests

This section contains a table with the functional tests for each actor. Each person that tested the system filled out a scheme to indicate if they found any errors and if it was easy or hard for them to find out how the functionality was.

| Test description | Tested successfully | | Difficulty of testing | | Comments |
|--|---------------------|----|-----------------------|------|----------|
| | Yes | No | Easy | Hard | |
| Log on | | | | | |
| Send an assignment document to the teacher | | | | | |
| Send an encrypted assignment document to the teacher | | | | | |
| Let other students sign the assignment document | | | | | |
| Submit a file sent from another student | | | | | |
| Send a file to another student, that has been signed by other students | | | | | |
| Ask to receive email or SMS when a new file arrives | | | | | |
| Delete a file, you don't want to sign | | | | | |
| View a file in the list | | | | | |
| View an encrypted file in the list | | | | | |
| Decrypt an encrypted file | | | | | |
| View the details of the submission | | | | | |
| In each frame, go back | | | | | |
| Log out | | | | | |

Table C.1: Functional test for the student

C.2 Structural Tests

Test cases were identified by going through each program and finding test cases to test all branches of the program. The test cases already stated in the functional test part are not repeated. The structural test was carried out by the author.

| Test description | Tested successfully | | Difficulty of testing | | Comments |
|--|---------------------|----|-----------------------|------|----------|
| | Yes | No | Easy | Hard | |
| Log on | | | | | |
| View received submissions | | | | | |
| View a file in the list | | | | | |
| View an encrypted file in the list | | | | | |
| Find a submission in the e-box | | | | | |
| View the details of a submission | | | | | |
| Create a new folder | | | | | |
| Edit a folder name | | | | | |
| Move files to a new folder | | | | | |
| Ask to receive email or SMS when a new file arrives | | | | | |
| View the access that others have to you e-box | | | | | |
| Revoke permission to access your folder | | | | | |
| Allow some user to access one of your folders | | | | | |
| Create a comment on a submission | | | | | |
| Remove a comment you created | | | | | |
| View a file in the e-box | | | | | |
| Decrypt an encrypted file | | | | | |
| Archive files you don't want to have in your folders | | | | | |
| Restore archived files | | | | | |
| View the assignment of the course | | | | | |
| Create a new assignment | | | | | |
| Edit information about assignment | | | | | |
| Delete an assignment | | | | | |
| View files that other people have given you access to | | | | | |
| Search in files that other people have given you access to | | | | | |
| In each frame, go back | | | | | |
| Log out | | | | | |

Table C.2: Functional test for the teacher

| Test description | Tested successfully | | Comments |
|---|---------------------|----|----------|
| | Yes | No | |
| Cancel instead of signing a submission | | | |
| Alter the file that was forwarded to you, before you sign it | | | |
| Encrypt a file for a recipient that does not have a registered certificate | | | |
| Sign files for elements or submissions you don't have access to | | | |
| Select only one recipient when forwarding | | | |
| Select more than one recipient when forwarding | | | |
| Download document before signing | | | |
| Submit another document for the same assignment | | | |
| Enter non existing filenames when signing | | | |
| Enter wrong password when signing | | | |
| Enter wrong password when viewing the certificate before signing or decrypting | | | |
| View the certificate used before signing or decrypting a file | | | |
| Delete a file that you created, and was deleted by the recipient | | | |
| Download a PDF, DOC and TXT file | | | |
| Download a file, that you don't have access to | | | |
| Download an encrypted document | | | |
| Decrypt a file, that you don't have access to | | | |
| Enter non existing filenames when decrypting | | | |
| Enter wrong password when decrypting | | | |
| View details of a submission you don't have access to | | | |
| Ask for a notification for another user | | | |
| Ask for a notification on email or SMS, when it is not registered in the database | | | |

Table C.3: Structural test for the student

| Test description | Tested successfully | | Comments |
|--|---------------------|----|----------|
| | Yes | No | |
| Enter a non existing student number when granting access to a user | | | |
| Grant access to folder that you don't own | | | |
| Edit an access that has been registered, but is not to your folders | | | |
| Archive documents that are not yours | | | |
| Edit assignments you don't have access to | | | |
| Insert assignments for an element you don't have access to | | | |
| Change the recipients of the assignment | | | |
| Let the date from be after date to, when inserting a new assignment | | | |
| Enter illegal dates in the date fields | | | |
| Download an encrypted document | | | |
| Decrypt a file, that you don't have access to | | | |
| Enter non existing filenames when decrypting | | | |
| Enter wrong password when decrypting | | | |
| View details of a submission you don't have access to | | | |
| Download a PDF, DOC and TXT file | | | |
| Download a file, that you don't have access to | | | |
| Select the folder, when selecting files in the e-box that are from a certain course | | | |
| Select all folders, when selecting files in the e-box that are from a certain course | | | |
| Select the folder, when searching for a file in the e-box | | | |
| Select all folders, when searching for a file in the e-box | | | |
| View files in a certain folder | | | |
| View archived files | | | |
| Restore archived files | | | |
| Ask for a notification for another user | | | |
| Ask for a notification on email or SMS, when it is not registered in the database | | | |

Table C.4: Structural test for the teacher

Bibliography

- [AlPe03] Alvarez, G., Petrovic, S., *A new taxonomy of Web attacks suitable for efficient encoding*, Technical report, Computers and Security, Vol.22 Issue.5, 2003, p 435-449.
- [An02] Anley, C., *Advanced sql injection in sql server*, Next Generation Security Software, Technical report, January 2002.
- [Ce02] Cerrudo, C., *Manipulating Microsoft sql server using sql injection*, Technical report, Application Security, Inc., 2002.
- [Ki04] Kilemark, F., *Secure Working from Home in an Industrial Context*, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, 2004
- [Kl04] Klein, A., *Blind XPath injection*, Technical report, Sanctum Inc., 2004.
- [Ko02] Kolsek, M., *Session fixation vulnerability in webbased applications*, Technical report, Acros, 2002.
- [LeHu00] Leung, K.R.P.H., Hui, L.C.K., *Multiple Signature Handling in Workflow Systems*, Technical report, Proceedings of the 33rd Hawaii International Conference on System Sciences - 2000, 2000.
- [Mi00] Microsoft Corporation, *Cross-site scripting security exposure executive summary*, Technical report, 2000.
- [Mi04] Microsoft Corporation, *Web Security Threats and Countermeasures*, MSDN Magazine, January 2004.
- [MiMi01] Mitomi, S., Miyaji, A., *A general model of multisignature schemes with message flexibility, order flexibility, and order verifiability*, Technical report, SCIS 2000, 2001.
- [Mo03] Moussa, C.M.R., *Digital Signatures and Multiple Signature: Different cases for Different Purpose*, Technical report, SANS Institute, 2003

[Ni00] (National Institute of Standards and Technology). *Digital Signature Standard*, FIPS PUB 186-2, 2000.

[OpOc] OpenOCES project, <http://www.openoces.dk>.

[PfPf03] Pfleeger, C.P. and Pfleeger, S.L., *Security in Computing, Third Edition*, Prentice Hall, 2003.

[Pr04] Proise, J., *Wicked code, Foiling Session Hijacking Attempts*, Technical report, MSDN Magazine, August 2004.

[RaGe03] Ramakrishnan, R. and Gehrke, J., *Database Management Systems, Third edition*, McGraw-Hill, 2003.

[Sc96] Schneier, B., *Applied Cryptography*, John Wiley & Sons, Inc., 1996

[ShLiYaSu03] Shieh, S., Lin, C., Yang, W., and Sun, H., *Digital Multisignature Schemes for Authentication Delegates in Mobile Code System*, IEEE transactions on vehicular technology, Vol. 49, No. 4, July 2000

[Øs04] Østerby, T., *Chapter 4: Information Modeling*, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, 2004.