The Cloud, a security risk?

A study on cloud computing and efficient encryption.

Digital Forensics 15 credits

Halmstad 2018-08-02
Josef Al-khayatt, Robin Nguyen
The Cloud, a security risk?  
A study on cloud computing and efficient encryption.

Bachelors thesis  
2018

Authors:  
Josef Al-khayatt  
Robin Nguyen

Mentor: Malin Bornhager  
Examiner: Urban Bilstrup

IT Forensics and Information Security  
School of Information Technology  
Halmstad University
Acknowledgments

This project is a bachelor's thesis for the program of IT Forensics and Information Security at Halmstad University (Bachelor of Science). We would like to thank for the service and help provided by the University of Halmstad.

A special thanks to Malin Bornhager our supervisor at ITE for the guidance, materials, insight and communication throughout the process and to Eric Järpe at ITE for the guidance regarding the mathematical calculations, experiment and always answering our questions.

Last but not least, a very special thank you to our families and loved ones.
Abstract

After many incidents of cloud services being attacked and personal user data leaking, a secure way of storing data on the cloud would be to encrypt. Even if an attacker would have access to the files he/she would not be able to decrypt the files without the secret key. In this thesis a software will be developed and works by securing user data by encrypting, prior uploading to the cloud. This software is a potential solution to the existing threat.

The thesis presents many different methods of encryption but narrows it down to two, AES and XOR, which will be implemented into the software. This is done by statistically comparing the speed of each system. As well as comparing the expected time by using a student’s t-test calculating what degree of certainty one will be faster than the other. The thesis discusses and highlights level of security regarding cloud security. Comparisons of the algorithms are made to determine which method to be used under what circumstances.
# Table of content:

<table>
<thead>
<tr>
<th>Chapter 1 Introduction</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 Background</td>
<td>2</td>
</tr>
<tr>
<td>1.2 Purpose</td>
<td>3</td>
</tr>
<tr>
<td>1.3 Problem definition</td>
<td>3</td>
</tr>
<tr>
<td>1.3.1 Delimitations</td>
<td>4</td>
</tr>
<tr>
<td>1.3.2 Problematization</td>
<td>4</td>
</tr>
<tr>
<td>Chapter 2 Method</td>
<td>6</td>
</tr>
<tr>
<td>2.1 Literature</td>
<td>6</td>
</tr>
<tr>
<td>2.2 Experiment</td>
<td>7</td>
</tr>
<tr>
<td>2.3 Problematization</td>
<td>9</td>
</tr>
<tr>
<td>2.4 Ethical standpoint</td>
<td>9</td>
</tr>
<tr>
<td>Chapter 3 Previous works</td>
<td>10</td>
</tr>
<tr>
<td>3.1 Identity-Based Encryption with Outsourced Revocation in Cloud Computing</td>
<td>10</td>
</tr>
<tr>
<td>3.2 Key-Aggregate Searchable Encryption (KASE) for Group Data Sharing via Cloud Storage</td>
<td>11</td>
</tr>
<tr>
<td>3.3 SecSVA: Secure Storage, Verification, and Auditing of Big Data in the Cloud Environment</td>
<td>12</td>
</tr>
<tr>
<td>3.4 An Efficient File Hierarchy Attribute-Based Encryption Scheme in Cloud Computing</td>
<td>13</td>
</tr>
<tr>
<td>3.5 Hybrid Attribute- and Re-Encryption-Based Key Management for Secure and Scalable Mobile Applications in Clouds</td>
<td>14</td>
</tr>
<tr>
<td>3.6 Virtualization of the Encryption Card for Trust Access in Cloud Computing</td>
<td>14</td>
</tr>
<tr>
<td>3.7 Encryption based solution for data sovereignty in federated clouds.</td>
<td>15</td>
</tr>
<tr>
<td>3.8 Evaluation of Four Encryption Algorithms for Viability, Reliability and Performance Estimation</td>
<td>15</td>
</tr>
<tr>
<td>3.9 Comparison of Encryption Algorithms for Multimedia</td>
<td>16</td>
</tr>
<tr>
<td>Chapter 4 Theory</td>
<td>17</td>
</tr>
<tr>
<td>4.1 Cloud Computing</td>
<td>17</td>
</tr>
<tr>
<td>4.2 Cloud Models</td>
<td>17</td>
</tr>
<tr>
<td>4.3 Cryptology</td>
<td>18</td>
</tr>
<tr>
<td>4.3.1 Cryptography</td>
<td>18</td>
</tr>
<tr>
<td>4.3.2 Cryptanalysis</td>
<td>18</td>
</tr>
<tr>
<td>4.3.3 Symmetric &amp; Asymmetric Encryption Algorithms</td>
<td>19</td>
</tr>
<tr>
<td>4.3.4 Advanced Encryption Standard</td>
<td>20</td>
</tr>
<tr>
<td>4.3.5 Exclusive OR</td>
<td>21</td>
</tr>
<tr>
<td>Chapter 5 Empirics</td>
<td>22</td>
</tr>
</tbody>
</table>
5.1 Technical setup 22
5.2 Experiment setup 22
  5.2.1 Software 23
  5.2.2 Calculations 26
Chapter 6 Results 29
  6.1 Software 29
  6.2 Result from the test XOR vs AES. 33
    6.2.1 Test data with initializing value 33
    6.2.2 Test data with without initializing value 34
  6.3 Calculations for statistical evidence 36
  6.4 Calculations for Student’s t-test 38
    Student’s t-test for class 1: 38
    Student’s t-test for class 2: 39
    Student’s t-test for class 3: 40
  6.5 Statistical evidence without initializing value 42
  6.6 Statistical evidence with initializing value 43
Chapter 7 Discussion 45
  7.1 Method Discussion 45
  7.2 Result Discussion 45
  7.3 Future works 50
Chapter 8 Conclusion 51
  8.1 Conclusion 51
Appendix 52
  AES Encoder class code used in AES vs XOR Experiment 52
  XOR Encoder class code used in AES vs XOR experiment 53
  Software flowchart 56
    AES Class 56
    XOR Class 58
    FTP Upload Class 61
    AES vs XOR Class 62
References 63
Chapter 1 Introduction

Today, almost everyone uses cloud storage one way or another. But how sure is it that the information will never be seen or used by anyone else? Or that the files chosen to be deleted, actually get removed completely?

In today’s modern society, cloud technologies are a commonplace. Huge numbers of users use various cloud-based services from many providers such as Google, Apple, Microsoft and IBM. These are just a few examples of everyday cloud service providers. You upload images and documents on exemplary Google Drive or iCloud in hope that your files will be secured on the cloud. This is not always the case. Back in 2014 lots of celebrities iCloud accounts were breached and nude photos were posted all over the internet. This was due to a combination of weak passwords, easy-to-guess security questions, a bug in Apple's photo backup service, and the unlimited ability to try passwords without getting locked out.[1]

Cloud service providers struggle to manage the flood of data, and to deal with these issues your data may end up in other companies servers such as Cisco, IBM, Verizon, etc.. Even if you delete your files from your phone and from your cloud service, there still might be a chance that your files have been copied to another server which you can’t access.[2]

So, what is cloud computing? A small definition is made here and will be deeper explained in chapter 4.

According to National Institute of Standards and Technology (NIST), cloud computing is defined as following: “Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models.”[3]

And according to International Organization for Standardization (ISO), cloud computing is defined as following: “Cloud computing is a paradigm for enabling network access to a scalable and elastic pool of shareable physical or virtual resources with self-service provisioning and administration on-demand. The cloud computing paradigm is composed of key characteristics, cloud computing roles and activities, cloud capabilities types and cloud service categories, cloud deployment models and cloud computing cross cutting aspects that are briefly described.”.[4]

Both ISO and NIST have similar definitions of cloud computing and in this thesis NIST’s definition will be used to explain cloud computing as a personal choice, along with focus on available and newly developed encryption algorithms. The work will include a software built with current encryption methods combating cloud security and integrity threats as well as an
experiment where the goal is to compare encryption processing time between two algorithms, AES and XOR.

1.1 Background

The need to securely transport information has always been important throughout human history. Through the ages the techniques vary but the basic principle is the same. To protect information from being read by nonauthorized individuals. One of the earliest known cryptography texts are believed to have been created nearly 4,000 years ago. The scribe of nobleman Khnumhotep II drew his master’s life on his tomb where he used a huge number of unusual symbols to obscure the sense of inscriptions. This method is an example of a substitution cipher, which substitutes one or more character/symbol for another.[5]

In Greece around 500 B.C the Spartans developed a device that would be used throughout Greece, to send and receive secret messages named Scytale. The device was a cylinder where a message was written lengthwise on paper or cloth. Once unwound, the message on the paper or cloth would be unreadable. To receive the correct message an identical cylinder was needed, it was only then the letters would line up correctly to become the original message. [5,6]

Another famous cipher was used by the romans around 2000 years ago. Named after the emperor Julius Caesar, who developed a substitution cipher for usage in the military field when transporting secret messages. [5]

Another famous user of substitution cipher was Queen Mary of Scots, who used an outdated method that codebreakers already knew how to decrypt using frequency analysis. Mary wanted to assassinate her cousin Queen Elizabeth I. She used enciphered messages to communicate with her co-conspirators. She was eventually captured and Elizabeth’s chief codebreaker cracked the cipher. She was arrested, put on trial and executed for treachery. All because her cipher was cracked. You could make the argument that her life depended on the strength of her cipher, since the deciphered messages was used as evidence against her.[7,8]

The previous were only some examples of ciphers that were all cracked throughout history and exemplifies the importance of strong ciphers and protecting information. The role that modern ciphers and encryption methods have in today’s society is still the same as their predecessors and it is to protect information. The difference being the sheer amount of information that is exchanged with the use of the internet. In today’s society on a daily basis you could say that the amount of information is almost endless. With valuable information comes strong ciphers and encryption methods, but also threats of individuals cracking said ciphers. The main difference between past and present is that the traditional pen and paper way of cracking ciphers has been replaced by computers. With the help of computers processing power, it is now way easier to find flaws and weaknesses in ciphers. [5,8]
An example of a cracked cipher is Data Encryption Standard (DES), which is a federal encryption standard that uses a symmetric key and was developed during the 1970’s to protect unclassified computer data and communications. DES was cracked back in January 1998 by Electronic Frontier Foundation (EFF). The cracking method used, was a custom-built machines that could read information encrypted with DES by finding the key. The EFF DES Cracker is an ordinary computer with approximately 1500 chips. The software in the computer searches the custom chips for interesting keys periodically while the hardware eliminates most incorrect answers. This means that the software quickly can search through the remaining keys using thousands of chips at a time since the software only has to search a tiny fraction of the key space to find the answer.[9]

A commonly used algorithm that can be attacked is the Diffie-Hellman (DH) algorithm. It was invented back in 1976 by Whitfield Diffie and Martin Hellman and is the basis of most modern automatic key exchange methods and also one of the most commonly used protocol in networking today. It is commonly used in data exchange such as IPsec, VPN, SSL, TLS and SSH. DH is a method that securely exchanges keys used to encrypt data with. It works by generating two identical shared secret on both systems using a mathematical algorithm. Since asymmetric keys system are extremely slow for encrypting bulk of traffic it is more common to use symmetric algorithms such as DES, 3DES and AES. [10]

With the introduction of clouds and cloud services this sparked the interest when it comes to cloud security, especially when it comes to cloud encryption. As proven throughout history the need to protect data is a vital part of the modern societies infrastructure. As the the use of cloud services are more and more common, the need to investigate the encryption methods in use, as well as their efficiency in matter of encryption time.

1.2 Purpose

The contribution with this thesis is to raise awareness when it comes to storing data on the cloud and the risks of not securing the data, as well present a comparison in speed between two different encryption algorithms, to determine time and efficiency.

1.3 Problem definition

The goal with this bachelor thesis is to investigate different types of encryption algorithms commonly used in today’s modern world to secure data on the cloud.

An analyze of different encryption algorithms will be made to determine which algorithm is the fastest while still being robust against attacks.

Our questions at issue will be as following:

- How may one secure one’s data on the cloud?
- How do the two algorithms compare to each other in matter of encryption time?
1.3.1 Delimitations

Previous works within the field have discussed the question of cloud encryption and presented many models for making users data more secure, and some methods are being used by some cloud services today. Previous works have done mathematical comparisons comparing two or more datasets to each other. This thesis will analyze existing encryption methods and systems. The security threat of analyzing a system that is in use by a cloud service, as cloud providers (for ethical reasons) won’t publish what encryption method is in use. Therefore, the choice to use existing encryption methods have been made. It is also unreasonable that a new encryption method, system or algorithm will be developed in this project, since the mathematical prowess is not great enough to create a new encryption method. The project will limit itself to analyzing and comparing two different existing encryption methods as well as enabling the user to secure their data before it is sent to the cloud. The software is limited to one programming language of choice.

It will be developed and tested with the equipment and devices at hand. Therefore, assumptions cannot be done that the software will perform equally on all devices and systems. This means, when it comes to testing encryption speed, the results will vary depending on the equipment. As far as implementing both encryption methods and being able to upload, it will not cover against all attacks, as there is no guarantee that the communication will be completely secure on the Internet. The intention of this implementation is to raise the security level for the common user and companies as some services do not provide encryption on the client side, as well as no encryption method on site where the data is stored, making the data vulnerable to other kinds of attacks. Even though the software might not be the most optimal, it might raise the security level and protect the user’s data against common threats.

1.3.2 Problematization

The possibility of security flaws might emerge since the encryption systems used are existing ones. Although the selected systems have been proven by the test of time modifying it to the needs of the experiment may expose a weakness or invite the possibility of another kind of attack. Previous work throughout history has mathematically compared data as well as calculated the expected time through many different calculations. A concern is that the choice of amount of test data pool might not be large enough for a statistical comparison. This might affect the calculations and ultimately affect end results. The processing time when encrypting the files used in the experiment will also be affected subjectively to the equipment and the processing power of the CPU on the computer.

Many have attacked the problem of how to secure one’s data on the cloud? There are many possible solutions to this problem some may even be recommended by standards or used by companies today. This thesis must use a pool of choice from these
different methods for example cloud encryption. It is not a possibility to explore all the existing encryption systems, so a choice must be made which one to use in the thesis. So, compared to previous research this thesis focuses on comparing the two different methods of XOR and AES with these tests and calculations based of a self-developed software.
Chapter 2 Method

The methods chosen to find an answer to the questions at issue in this study is a combination between a qualitative literature study and a scientific experiment comparing the encryption speed of two different chosen algorithms.

2.1 Literature

Relevant and important information regarding the background, theory, experiment and analysis of the results will be attained using a qualitative literature study. This chosen method is a stepping stone enabling the execution of the experiment which is a combination of different scientific areas. The experiment involves cryptology, cloud security and cloud computing. These scientific subjects are used as parameters when searching for information in libraries and scientific databases.

Relevant sources and references found in articles and books will be researched to find more literature. The information from the sources will be analyzed and compared to sift through all information to attain knowledge. This will help to explain models for experiment setups, technical aspects, current standards within cloud encryption and previous research within cryptography regarding the methods that are relevant. The qualitative literature study basis is to find examined books or articles that refer to other articles or books that has been examined and publicized regarding relevant areas.

This method where the content is examined after signs of stipulative (where many different experiments reach the same conclusion) or conclusive (statistically proven) knowledge which is then compared with other sources to find some kind of unified common ground. This is highlighted by Haraldson in [11].

Literature that has been sorted. To then be read to gain the right knowledge, which is then contextualized and applied in the experiment. The empirical data that the experiment will provide is then analyzed and compared with the knowledge that has been attained to try to find patterns or signs of deviation from already established standards. If it is possible then present theories of potentially predict the results of the experiment. Theories will be tested and evaluated to research if they are realistic or not. Methods to attain knowledge within a subject or area is often linked to Bloom's taxonomy or “knowledge pyramid”[12].
2.2 Experiment

The second method in this study is an experiment which involves the development of a software that can secure the users data by encrypting files with one of the two systems tested during this experiment. The systems are AES and XOR.

A statistical comparison is needed to determine which system out of the two is the optimal one for personal use to encrypt files before they are uploaded to the cloud. To determine the faster or more efficient one out of the two the software will be developed with a function, which is able to read files in a directory and encrypt the files using both AES and XOR. While simultaneously recording and present the time it took in nanoseconds.

The files tested are listed into three classes in the chart as seen in Table 2.2.1.

Table 2.2.1: Chart listing the number of files that will be used with each class and how long the text will be in pages.

<table>
<thead>
<tr>
<th>Type of File</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-1 A4 document pages with text</td>
<td>10</td>
<td>2-10 A4 document pages with text</td>
<td>11-30 A4 document pages with text</td>
</tr>
<tr>
<td>No. Files</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

The reason for the specific classification of the files were deemed to be the most common sizes and most used when it comes to files stored on the cloud. The results of the encryption tests will be the time it took to encrypt each file in nanoseconds, which will then be used in further tests and calculations determining the average time and the expected time using a student’s t-test.

The formula for calculating this is as followed:

$$
\begin{align*}
H_0: \mu_1 &= \mu_2 \\
H_1: \mu_1 &< \mu_2 \\
\overline{x}_1 - \overline{x}_2 \\
\sqrt{\frac{(n_1 - 1)s_1^2 + (n - 1)s_2^2}{n_1 + n_2 - 2}} \left( \frac{1}{n_1} + \frac{1}{n_2} \right)
\end{align*}
$$

(2.1)
The expected results from this should be $u < -t_{\alpha} - (n_1 + n_2 - 2)$ and the $p-value = < -t_{\alpha}(n_1 + n_2 - 2)$.

Where $\mu_1$ is the encryption speed of XOR and $\mu_2$ is the speed of AES.

And if the expected time of AES is larger than the expected time of XOR the $H_0: \mu_1 = \mu_2$ will be discarded resulting in $H_1: \mu_1 < \mu_2$ as the truth with a certain significance level.

The results will be presented in the form of the statistical comparison between XOR and AES as well as the comparison of the expected values using the student's t-test. The system of choice will be implemented into a fully functional software that is able to encrypt a file using one of these systems as a built-in function as well as it being able to upload a file the cloud.

As a parameter the choice of encryption methods to analyze are: XOR, AES.

This test will ultimately determine which one of the two chosen and implemented systems is the more efficient one regarding to speed. This will be presented in the form of calculations showing mathematical results, and diagrams to show the statistical evidence.

To combat the second issue regarding cloud encryption is to combat the question of security. To solve this issue since the software is supposed to be for personal use only, the decision was made to generate and store the decryption keys locally. This way it is possible to circumvent and avoid external attacks regarding encryption cracking while on the cloud or keys in transit, since the keys are packaged in the encrypted file. The important thing to show in the fully developed software is what kind of Random Number Generator is used, since it directly affects the keys strength and the overall security of the system.

Previous works have presented encryption systems, cloud encryption and mathematically compared statistics. But what separates this method from previous is the fact that it combines cloud encryption with testing and comparing two selected encryption systems through statistical tests and calculations regarding the efficiency of each one. The results of these tests will then be one of the determining factors for which one of the two systems will be used as the encryption system to encrypt the files before it being sent to the cloud.

The idea is that these two methods will enable the creation of a unique software which will be the final product and the result of this study. The final product will serve to answer the questions at issue with the self-made software which encrypts/decrypts data from the client side to secure the data before it is uploaded to the cloud by the user.
2.3 Problematization

There have been many different methods and implementations of encryption software in previous published research. Since the encryption system used in the software is based upon existing systems it is important to identify which ones are applicable in this specific scenario, meaning the “faster” encryption systems. Therefore, it is important to early on through literature gain a thorough overview of existing systems to be able to identify these and implement them into the software. As well as proficiency within a programming language to be able to implement the theories into the software. Encryption systems that will be researched and discussed can be discarded in case they turn out to be irrelevant to the experiments end result. To combat this a wide variety of current standards has to be chosen and with expert help will be rendered down to just a small amount that will be implemented and tested. The sample size used in the tests may prove to be to small/large. This can produce weaker or misleading results as maximum points may occur that directly affects the experiment and the calculations to follow. This can also be combated by having sufficient knowledge and expert help on how to perform statistical test (Eric Järpe, Ph.D. in Statistics). The results may prove to contradict the presented and underlying research done. This can lead to false conclusion regarding the literature used and totally question the method of choice. With that in mind it is still important to do the experiment to fulfill the purpose of this study.

The difference between this thesis and other thesis is the developed software which focuses on that the user self is responsible for the key management which leaves the security in the user’s hand. None of the thesis used for research combines the method of a self-developed software with comparisons of encryption time of common encryption systems with their own system at hand.

2.4 Ethical standpoint

The results from the experiment should be possible to perform from the given conditions. A successful experiment with a complete experiment setup will make it easy for an individual to recreate the work done and is vital for presenting scientific evidence and results. There is an ethical dilemma when it comes to security regarding cloud services. Why this is because of the jurisdiction regarding the cloud. Different geographical areas have different laws regarding the cloud. An example is there is no need for the provider nor the government to inform the user when their data is moved, read, or copied. Wherever in the EU users needs to be informed. This all seem clear cut, but the cloud is in a sort of a grey zone. That meaning the different services source and trade spaces very often meaning that the location of the service might be in the EU but the file is actually stored in the US.[19] By encrypting the files beforehand a user can secure their own data and add another layer of security. This does not affect any jurisdiction as a user may upload any file they wish, as long as they do not cause harm to the cloud systems (ethically speaking).
In some cases, it might be hard to precisely recreate the software developed, for the choice to not present the complete source code has been made because of the time and effort put in to develop the software. The thesis is made with an intent to research and highlight different encryption algorithms efficiency and processing power.

Chapter 3 Previous works

There are many different encryption systems, schemes, models, etc. proposed and or used. The matter remains that all of them fulfill different types of challenges in different infrastructures such as, how to reduce computational costs for key management.

What separates previous works from this thesis is that within the software a comparison is made between AES and XOR, and that has not been done before. The self-made software is also able to encrypt and compare number of files, by comparing the factors of speed and security. Recommendations are then made to encrypt files using either AES or XOR. Storing the keys locally eliminates many different security threats and is a unique method of handling key security regarding encryption systems. As our programming knowledge isn’t high enough to create a server software that handles keys, storing the keys locally was made. The option is then made to upload the file to the cloud, but by encrypting the file beforehand it adds a higher level of personal security regarding the user’s files. Previous works has done one or the other of these methods but this one is unique in the sense of combining many different approaches to reach the end goal of cloud security.

The different types of encryption systems that have come along our essay, work as following:

3.1 Identity-Based Encryption with Outsourced Revocation in Cloud Computing

Identity-Based Encryption (IBE) is an alternative to public key encryption since it uses human-intelligible identities such as unique name, email address, IP address, etc instead, as the public key. This is proposed to simplify key management in a certificate-based Public Key Infrastructure (PKI) by encrypting the data with the receiver’s identity without having to look up public key and certificate. The receiver which obtains the private key with the matching identity is able to decrypt the ciphertext from the Private Key Generator (PKG). An issue with using PKG with IBE is the more the number of users increase, the more bottlenecked the system will become as its online and has to maintain all transactions in a secure manner.

A typical IBE scheme consists of four different algorithms and they are Setup, KeyGen, Encrypt and Decrypt. Setup algorithm takes security parameter as input and outputs a public key & master key (the master key is kept secret). KeyGen algorithm also known as private key generation algorithm which takes the master key and the user’s identity as input to return a private key that corresponds to the identity. Encrypt algorithm uses the receiver’s identity and message to encrypt and outputs the ciphertext (run by the sender). Decrypt algorithm
uses the ciphertext and the private key to return a message or an error (run by the receiver). Key-Update Cloud Service Provider (KU-CSP) can be seen as a public cloud by a third party that helps the PKG deliver basic computing by providing a temporary extension to an infrastructure. KU-CSP will then be used by the unrevoked users for updating a component in their private keys. Combining the KU-CSP with IBE changes the scheme of three out of four algorithms and adds another two. It changes KeyGen by adding revocation list and time list as input, and adds time period to Encrypt/Decrypt. Revoke algorithm is run by the PKG and basically adds a time period to users until they are revoked. KeyUpdate algorithm is run by KU-CSP and basically updates users time period to avoid revocation and computing cost from the PKG. The proposed method is to use a hybrid private key for each user that connects two subcomponents, identity component and time component to achieve efficient revocation. This is done by outsourcing the revocation scheme to KU-CSP which the PKG sends. This makes the scheme semantically secure against adaptive chosen-ciphertext attack. The only way a user can decrypt a ciphertext is if the time period and identity in the private key matches the associated ciphertext. The KU-CSP role with PKG is to update the keys if they aren’t on the revocation list and the PKG updates the revocation list on the KU-CSP.[13]

3.2 Key-Aggregate Searchable Encryption (KASE) for Group Data Sharing via Cloud Storage

Even if cloud storage is viewed as a promising solution for on-demand access, data leaks is also a big concern either by malicious adversary or a misbehaving cloud operator. This may cause serious damage either to individuals or businesses. The most common way to avoid potential data leaks is to encrypt the data before uploading it to the cloud so that only those with the decryption keys may retrieve and decrypt the data. This method is called cryptographic cloud storage, and one of its issues is for users who wish to search for data using keywords. A solution for this is usually Searchable Encryption (SE) scheme where the data owner has to encrypt keywords and upload them with the data to the cloud. Even if combining these two methods, the issue remains that larger scale applications will have issues efficiently managing encryption keys, as each key will have different permissions. The proposed scheme is Key-Aggregate Searchable Encryption (KASE) which addresses these challenges. The scheme may be applied to any cloud storage that supports searchable group data sharing which means that a group of files may be used by a group of people. The KASE scheme is the first known scheme that can satisfy both requirements.

The KASE framework consists of seven algorithms and it works as following, starting with Setup algorithm which is run by the cloud service provider, it creates parameters of which files belongs to the data owner. Keygen algorithm is run by the data owner to produce a public/master-secret key pair. Encryption algorithm is run by the data owner to encrypt files and generate its keywords ciphertext with a unique searchable encryption key. Extract algorithm is run by the data owner to generate an aggregate searchable encryption key for a group of selected files using the master-secret key. Trapdoor algorithm is run by the user to perform a search using his/her aggregate key. Adjust algorithm is run by the cloud server to
update trapdoors for each file. Lastly the Test algorithm which is run by the cloud server is used to check if a file contains the keyword that was searched for.[14]

3.3 SecSVA: Secure Storage, Verification, and Auditing of Big Data in the Cloud Environment

Since Internet-enabled devices has shown an increase in large numbers, this means that the load on network infrastructure increases too, and to effectively manage big data storage, data has to be stored on a closer location to the individuals whom use it, making it accessible at any time, and replicated across other data centers for backup/availability. The most challenges of big data stored on the cloud are data security, data authentication, data integrity, data confidentiality, data availability and data deduplication. Data security ensures that the stored data is secure, data authentication ensures that the user who access the cloud storage is genuine (who they say they are), data integrity ensures that the data is the same from the moment it leaves the user uploading it until it reaches the cloud, data confidentiality ensures that the data won’t be accessed unauthorised by encrypting the data, data availability ensures that the data can be accessed at anytime from anywhere, data deduplication ensures that data doesn’t get duplicated. The contributions the author made are, “a secure storage, verification, and auditing architecture (SecSVA), an attribute-based security framework with secure deduplication, a Kerberos-based identity verification and authentication scheme, and a data integrity scheme”.

SecSVA consists of multiple individuals such as, a client which is a user who wishes to access the data, a data service provider (DSP which is the owner of the data, a cloud storage manager (CSP) that controls a public & private cloud, a trusted party auditor (TPA) that verifies data integrity and a Keberos server which handles authentication and verification of a user. These are the steps in SevSVA, Secure Storage is a scheme that contains steps to secure data on the cloud. Secure Verification contains steps to provide access and verify the user, and Secure Auditing which provides steps that provides verification of data integrity.

Moving on the Attribute-Based Security Framework which is a public-key-cryptographic-based scheme that means only a user who is verified and authorized can access the files he/she are permitted too, either by Key-Policy ABE (KPABE) or Ciphertext-Policy ABE (CP-ABE). CP-ABE doesn’t support deduplication of encrypted data and deduplication helps to save storage space and bandwidth.

Lastly Kerberos-Based Verification and Authentication scheme means using a third party authenticator that handles service request made by the authorized user.[15]
3.4 An Efficient File Hierarchy Attribute-Based Encryption Scheme in Cloud Computing

Since the increase of data sharing, cloud computing has been a promising application platform to solve these issues. A challenge still remains, to protect data from leaking and being accessed by unauthorized users. And to solve these issues, encrypting data before sharing it has been the most common way. A preferred encryption technology for secure data sharing on the cloud is Ciphertext-policy Attribute-based Encryption (CP-ABE) as it's more suitable for general applications because of its flexibility. CP-ABE works as following, for authorized users to access encrypted data on the cloud they must possess certain attributes that are determined by the data owner. Their contribution is mostly to be used within healthcare and military where access levels of data are needed to keep sensitive data from being accessed by unauthorized users.

The authors propose an attribute-based encryption scheme that supports file hierarchy efficiently as it hasn’t been explored in CP-ABE.

The authors contribution is based on CP-ABE with a layered model of the access structure called file hierarchy CP-ABE scheme (FH-CP-ABE). It extends CP-ABE with a hierarchical structure of access policy to simplify access control. The system model for their scheme consists of four entities, Authority which is a trusted entity that adds users to the system and executes the Setup and KeyGen operations, Cloud Service Provider (CSP) which is the host of the service that performs the tasks given to it and returns correct results, Data Owner which is the user who uploads the data to the cloud, controls permissions for the data and executes the Encrypt operation, and lastly User which is the entity whom wishes to access the data and executes the Decrypt operation.

The scheme works as following:
1) Setup: Run by Authority to create a public and master key
2) KeyGen: Run by Authority using the public & master key to generate a secret key.
3) Encrypt: Run by Data Owner using the public key and hierarchical access tree to generate ciphertext.
4) Decrypt: Run by User using public key and ciphertext which includes an integrated access structure. If some keys match the integrated access structure, then some of the data content will be decrypted. If the keys match completely, all of the content will be decrypted.

The authors also improve their scheme to reduce computational costs by removing some transport nodes (if the nodes doesn’t carry any level nodes). They call it FH-CP-ABE with improved Encryption.[16]
3.5 Hybrid Attribute- and Re-Encryption-Based Key Management for Secure and Scalable Mobile Applications in Clouds

The trend in cloud computing applications is primarily for data stored on the cloud to be accessible by mobile devices (tablets and smartphones) using protocols providing security. Access to cloud-based data is increasing by resource-constrained mobile devices (devices with limited processing and storage capabilities), henceforth processing and communication cost must be minimized to preserve battery life. In this academic journal they propose modifications to attribute-based encryption to allow authorized users to access the cloud data much faster by assigning the cryptographic computational load to the cloud provider so that the communication cost is lowered for the mobile user. They also mention that the data re-encryption may be optionally implemented by the cloud provider to reduce users being revoked from the cloud service while still preserving users data privacy stored on the cloud. There are numerous solutions cloud services provide to exchange encrypted data in a secure manner without the provider being entirely trusted with key material. For example, the drawback of RSA is that it requires the owner of the data to provide an encrypted version of data for each user who wishes to access it. If the data is encrypted with a single key, the key has to be shared with all users who has the authorization to that data which in turn increases network traffic, especially if the user uses his/her cellular data.

The proposed algorithm to use to reduce these costs is by improving ciphertext-policy attribute-based encryption (CP-ABE) as it offers numerous of advantages. CP-ABE works as following, for authorized users to access encrypted data on the cloud they have to possess certain attributes that are determined by the data owner. The data owner is responsible for who may access the data stored on the cloud by granting access permission through an access tree. This requires the constant availability by the data owner in order for authorized users to gain permission to the data. This becomes a problem when the data owner is on a mobile device, that is why the authors have come up with a solution for key generation, distribution and usage.[17]

3.6 Virtualization of the Encryption Card for Trust Access in Cloud Computing

Since virtualization is used a lot in cloud computing it is hard to use the encryption card directly in the user domain because the mechanism in virtualization is complicated. Not to mention the security problems in the user key and the user private data flow. An encryption card is a hardware device which encrypts and decrypts information and provides higher-level security with higher efficiency compared to encryption softwares. The communication virtual machines use with encryption cards is using a split device-driver model, which is a frontend driver within the virtual machine that accesses a virtual encryption card instance.

In this academic journal they address these issues by implementing a new virtualization architecture to ensure the trustworthiness of encryption cards. They designed a virtual
encryption card system which allows functionality in virtual machines which they called vEC Privacy Preserving Model (vEC-PPM). The model manages the encryption resource schedule. This encryption card is based on the BLP model which is a popular state machine model to enforce access control of applications.[18]

3.7 Encryption based solution for data sovereignty in federated clouds.

In this journal the authors present the problem of the legal issues when dealing with cloud services. How different parts of the world handle and classify the security of the stored information differently. It is about data being stored in multiple data centers located in different countries, exacerbated in a federated cloud environment. To maximize resource utilization software within the federation middleware can replicate and/or move data between different cloud services that possibly are located in different countries without informing or having the consent of the owner. This makes the cloud providers unable to pinpoint the exact location of said data which makes the system inconsistent when it comes to legal issues, enabling the providers to keep moving data without consent.

To combat this problem the authors presented a model that required geological location when generating encryption keys which is the key to the algorithm used to encrypt the data. The data is then sent to the cloud and only users with the right geolocation would have the right decryption key to decrypt the data. This did not solve the problem of providers moving and/or replicating data without users consent not protecting the data but the solution instead is to guarantee the safety of said data as well as no unauthorized person would be able to intercept or read said data without the proper authentication as well as geolocation.[19]

And here to follow, a similar experiment to test efficiency.

3.8 Evaluation of Four Encryption Algorithms for Viability, Reliability and Performance Estimation

In today’s modern society everything is slowly being digitized, henceforward this makes security the most important aspect. To ensure that data or information is secured, the most common approach is to encrypt data (to make readable data unreadable for the unauthorized). The need for protecting files became more evident for systems that are accessed over the internet. Since the amount of network related attacks increases so does the unauthorized access by theft because of stored unsecured data.

In this academic journal, the authors decide to conduct an experiment similar to ours. The authors implement the following algorithms into Java programming language: DES, 3DES, AES and BLOWFISH. Their experiment is conducted on a computer using CORE i5 64-bit processor with 4GB of RAM. The experiment is conducted a couple of times to reassure that the results are consistent throughout the experiment. They decide to find out how the four
different algorithms perform on the same file by looking at the memory consumption in megabyte for each algorithm, CPU utilization time period in %, encryption speed in milliseconds, and different key size in bits.[20]

3.9 Comparison of Encryption Algorithms for Multimedia

To protect information from unauthorized users, is usually done by encrypting it with one or another encryption technique. Encryption techniques have different mathematical algorithms providing strength on different levels to withstand attacks. There are two different encryption algorithms, Symmetric- and Asymmetric-Key algorithms. Symmetric-Key algorithms uses a single-key to encrypt and decrypt the data. Whereas Asymmetric-Key algorithms uses two different keys, Private Key which is for the intended users to use for decrypting the data and the Public key which is used for encrypting data. A secure algorithm is an algorithm which is hard to guess, exemplary resistance against brute-force attacks.

In this academic journal the authors develop a simulator using Java programming language where Blowfish, AES, XOR and RSA algorithms are implemented to encrypt text, images, audio and video files. A similar experiment to this bachelor thesis is made where encryption and decryption time are calculated on the amount of data size.[21]
Chapter 4 Theory

4.1 Cloud Computing

Cloud computing is all about enabling access to data anywhere at any time, which involves large numbers of computers connected through a network. It goes by a “pay-as-you-go” model which allows organizations to flexibly use computing & storage as a utility in infrastructure since it can be used to reduce operational costs. It also supports a variety of data management issues such as access to data anywhere at any given time, subscriptions to only needed services, reduce cost for onsite IT equipment, maintenance, management, equipment, physical plant requirements, personnel and the ease to swiftly increase data volume.

Cloud computing services come in variety of options to meet the customer's need, however there are three main services which are defined by the National Institute of Standards and Technology (NIST), and they are as following:

- **Software as a service (SaaS)**: Provides access to email, communication and Office 365, which are delivered over the Internet. Users only need to provide the data.
- **Platform as a service (PaaS)**: Provides access to development tools and services to deliver applications.
- **Infrastructure as a service (IaaS)**: Provides to network equipment, virtualized network services, and supporting network infrastructures.

IT as a service (ITaaS) is an extended model of cloud services which providers have come up with to provide IT support for each cloud computing service. For businesses this means extended support without the costs for software licenses, training personnel or investing in new infrastructure as the service can be delivered on demand to any device in the world without compromising security or function of the service.[22]

4.2 Cloud Models

In addition to the cloud services there are primarily four cloud models which providers may offer, they are as following:

- **Public cloud**: Cloud-based applications and services which is available to the general population, either for free or pay-per-use and uses the Internet to provide these services.
- **Private cloud**: Cloud-based applications and services which is available to a specific organization or entity, such as the government. This type of cloud model can be set up to use an organization’s private network. This will require additional cost to build and maintain. An outside organization can manage this type of cloud with strict access security.
- **Hybrid cloud**: A model which is made up of two or more clouds, such as private, community or public, where each model is separated, yet still connected using a single architecture. Users on this type of cloud have different types of access to various services depending on their user access right.

- **Community cloud**: A model which is created for exclusive use by a specific community. The difference between this model and the public model is the functionality needed that have been customized for the community, for instance certain laws and/or policies.

Cloud computing is only possible because of data centers and the two terms are often used incorrect. A data center is typically a facility that processes and stores data. It’s managed by either the organization or leased offsite. Cloud computing is a service that offers access to a shared pool of configurable computing resources at any time from an offsite location. The only reason cloud computing is possible is because of data centers as organizations use their data centers to host cloud services and cloud-based resources. Providers often have several remote data centers to ensure the availability of their services. Another foundation to cloud computing is through virtualization, as cloud computing separates applications from hardware and virtualization separates the operating system from the hardware. This enables various providers to offer virtual cloud services which is a simple way for customers to decide the computing resources they need. [22]

### 4.3 Cryptology

As implied in the book [10] “Cryptography is the science of making and breaking secret code”, cryptology is the combination of cryptography which is the development and use of code, and cryptanalysis is to break the code.

#### 4.3.1 Cryptography

Cryptographic services are used to ensure the protection of data from being exposed to unauthorized individuals by using different policies and encryption methods. The main components of cryptography are authentication, integrity and confidentiality. The key to cryptography and a successful security policy, is by understanding the basic functions of cryptography and how encryption provides confidentiality and integrity. [10]

#### 4.3.2 Cryptanalysis

Cryptanalysis is the practice and study to cracking the code without knowing the shared secret key. There are different methods used in cryptanalysis, and they are as following:

- **Brute-Force Attack**: An attack which all encryption algorithms are vulnerable to. It works by trying every possible key with a decryption...
algorithm to eventually gain access. 50 percent of the way, the brute force attack succeeds by trying the set of all possible keys.

- **Ciphertext-Only Attack:** An attack where the attacker gathers several messages, where all of the messages use the same encryption algorithm, without knowing the plaintext. If the attacker can guess and find the key/keys used to encrypt the messages, they may be used to decrypt the messages. An alternative method to cracking the messages is by using statistical analysis to guess the keys used to encrypt them. This method no longer works on modern algorithms as they are resistant.

- **Known-Plaintext Attack:** An attack where the attacker knows something from the ciphertext about the underlying plaintext. Brute-force attacks are usually used to try the keys until a correct key produces a meaningful result.

- **Chosen-Plaintext Attack:** An attack where the attacker chooses which type of encryption was used to encrypt the data and then observes the ciphertext output.

- **Chosen-Ciphertext Attack:** An attack where the attacker may choose which ciphertexts are to be decrypted. The attacker also has access to the decrypted plaintext. With both the plaintext and ciphertext the attacker can search through sets of possible keys to determine which key decrypts the chosen ciphertext in the captured plaintext.

- **Meet-in-the-Middle:** An attack which is also known as *Known-Plaintext Attack*. This is where the attacker knows some parts of the ciphertext and the plaintext. the plaintext is encrypted with every possible key, and the results are stored. Every key is used to try decrypt the ciphertext until one of keys matches the stored value.[10]

### 4.3.3 Symmetric & Asymmetric Encryption Algorithms

There are two basic classes of encryption algorithms, *Symmetric* and *Asymmetric*. Symmetric encryption algorithms uses a single key to encrypt and decrypt data, and the key is commonly called the secret key. The key used must be pre-shared between both parties to commence an encrypted communication.

Asymmetric encryption algorithms uses different keys to encrypt and decrypt data, and the keys are commonly called Private key and Public key. This means that it does not need to have a pre-shared key.

It takes longer to generate asymmetric keys as both parties do not have pre-shared keys, meaning that the key length must be longer to withstand attacks. While symmetric keys can be shorter as only the authorized parties know the secret key to decrypt the data. So asymmetric encryption algorithms are more resource demanding and takes longer to generate keys than symmetric encryption algorithms.[10]
4.3.4 Advanced Encryption Standard

Also known as AES is an encryption scheme that was announced in 1997 to replace DES, as DES was recognized to eventually reach the end of its usefulness. The reason AES was invented was that there multiple reasons to replace DES, firstly because AES key length was much stronger, secondly because AES runs faster and is more efficient than DES/3DES on comparable hardware, thirdly AES is more suitable for high throughput, low latency environment.[10]

AES works as following: AES allows different block sizes starting with 128 and ending in 256, meaning that in between 168, 192, 224 also are accepted blocks. AES allows only 128, 192 and 256 bits of size. If we call the block size \( N_b \) and the key size \( N_k \) where \( N_b \) refers to the number of columns in the block. For example, if we use AES-128, it means that each block will consist of 128 bits. \( N_b \) can be calculated by dividing 128 by 32. This means that each column will be 32 bytes and \( N_b = 4 \). So, for the exemplary text “This is a test”, will be stored in blocks shown in Figure 4.3.4.1.

![Block example for AES 128-bit](image)

As we can see above, each character has its own cell inside the block, even the blank cells which are spaces in the text. The characters in the cells may be stored as integer values, hexadecimal values or binary strings depending on how the algorithm is implemented. All blocks must be filled because of the Rijndael algorithm. In order to fill all the blocks padding is used and padding means that extra bits are added to the original data until the desired size is complete. The key or cipher key is stored as blocks, the same way as above and has to be the same size. This means that as long as the key has the correct length, it may have any values without any restrictions. For ease of understanding of how it works Rijndael Rounds will be explained as following: “At a basic level the Rijndael algorithm uses a number of rounds to transform the data for each block. The number of rounds used is 6 + the maximum of \( N_b \) and \( N_k \). Following from the previous example of AES-128, the number of rounds is 10. This is calculated from 6 plus the maximum of (4,4). Since \( N_k \) and \( N_b \) are both 4, the number of rounds is 6 + 4 = 10 [2]. The initial block (also known as a state) is added to an expanded key derived from the initial cipher key. Then the round
processing occurs consisting of operations of the S-box, shifts, and a MixColumn. The result state is then added to the next expanded key. This is done for all ten rounds, with the exception of the MixColumn operation of the final round. The final result is the encrypted cipher block.”[23]

4.3.5 Exclusive OR

XOR is a logical function that can be applied to binary bits and is a cipher that is well known for its simplicity. Just like AES, it’s a symmetric encryption algorithm. XOR works by two arguments having two different values returning true, this means that it’s cipher is derived from Boolean algebra XOR function. The longer a random key is the better security performance as well as counter brute-force attacks.[24]

An explanation on how XOR works: "For the XOR encryption part, the letter in a plaintext is XOR bitwise 1 by 1 with the 8 bit secret key to form first encrypted text. Then each letter in the first encrypted text is shifted to a fixed position separated by a numerical value. Assume that:
Plaintext, Secret key, Numerical value, First encrypted text, Final Ciphertext
C1 = M \oplus K
C2 = C1+N

Therefore, the overall equation is,
C2 = (M \oplus K+N)

For example, let's say plaintext M has an “A” letter which is expressed in binary “01000001” and a binary secret key K “01111000”, the first encrypted text is C1 = 01000001 \oplus 01111000 = 0011100 (in hex 39H)

The C 1 then is shifted to right by the adding a numerical value, N which is 5 (in binary is “00000101”) into C1.
C2 = C1 + 00000101 = 00111110 (in hex 3EH)

The above example is the basic idea of the combined encryption technique.”[25]
Chapter 5 Empirics

Presented in this chapter will be the encryption methods used in the experiment, how the experiment will be conducted and how the software will be developed.

5.1 Technical setup

The technical equipment and software used in the experiment is listed here. The programming language of choice is Java since it is a cross platform language and it’s the programming language which we have knowledge in.[26] The software of choice is Eclipse IDE for Java developers[27] and operating system Microsoft Windows 10. The following packages were used to develop the software Java SE Development Kit (JDK) 8, Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files 8, Apache Commons IO 2.6 and Apache Commons Net 3.6. The software is tested on two different computers with two different specifications to validate the results. Relevant specifications are listed below:

Computer 1: Lenovo Y50-70 (Laptop)
OS: Windows 10 Home 64-bit (10.0, Build 16299)
Processor: Intel(R) Core(TM) i7-4720HQ CPU @ 2.60GHz (8 CPUs), ~2.6GHz
Memory: 16GB RAM
GPU: NVIDIA GeForce GTX 960M

Computer 2: Custom build
OS: Windows 10 Home 64-bit (10.0, Build 16299)
Processor: Intel(R) Core(TM) i7-8700K CPU @ 3.70GHz (12 CPUs), ~3.7GHz
Memory: 32GB RAM
GPU: NVIDIA GeForce GTX 980

The reasoning for this set up is that processing power is a factor when it comes to encryption/decryption using different methods.

5.2 Experiment setup

How the experiment was done is presented here to be able to explain how results and findings are retrieved to enable the possibility of recreation of the experiment.
5.2.1 Software

The software begins by presenting a main menu where you have five options, exit, encrypt a file, decrypt a file and upload a file to the cloud through FTP. If a user types in anything else than the options presented it prompts the user to re-type an option.

- **Exit**: Terminates the software’s session.
- **Encrypt/Decrypt using AES**: The software will go in to another menu where the user may choose three options (Encrypt, Decrypt, Exit). If the user chooses Encrypt, the software will ask the user which file to encrypt by typing in the full pathname to the file, which password to encrypt the file with, which directory to save the key and encoding file and if the user wishes to upload their encrypted file to the cloud.

Based on the user’s choice of uploading or not, the software takes different directions. If the user wishes to upload it to the cloud, he/she has to type in the *IP-address* or *Hostname* to the server along with *Username*, *Password* and which *Port* to connect on, before creating an encrypted file. If the user wishes to save the encrypted file...
locally he/she chooses the directory. The software will create a duplicate file of the original file with an extension of “aes.enc” along with an encoding file (Salt) and the IV file which adds randomness to the file making it unreadable unless its decrypted with the software. If the user chooses to upload it to the cloud, the software will save the file temporary to the same directory of the original file and then delete it after uploading it.

If the user chooses Decrypt, the software will ask the user which file to decrypt by typing in the full pathname to the file, password to the file, and the full pathname to both Salt and IV files for AES. The software then decrypts the file using the three files provided to it. When the decryption is done it asks the user if it would like to delete the encrypted file along with the IV and Salt files. Depending on the user’s choice it either deletes the files or keeps them.

To return to main menu, the user simply types in the number “0” and hits enter.

- **Encrypt/Decrypt using XOR:** The software will go in to another menu where the user may choose four options (Encrypt, Decrypt, Generate Key, Exit). If the user chooses Encrypt, the software will ask the user which file to encrypt by typing in the full pathname to the file, what key to encrypt the file with (if the user hasn’t generated a key), and if the user wishes to upload their encrypted file to the cloud. Based on the user’s choice of uploading or not, the software takes different directions. If the user wishes to upload it to the cloud, he/she has to type in the IP-address or Hostname to the server along with Username, Password and which Port to connect on, before creating an encrypted file. If the user wishes to save the encrypted file locally he/she chooses the directory. The software will create a duplicate file of the original file with an extension of “xor.enc”. If the user chooses to upload it to the cloud, the software will save the file temporary to the same directory of the original file and then delete it after uploading it.

If the user chooses Decrypt, the software will ask the user which file to decrypt by typing in the full pathname to the file and which key was used to encrypt the file. The software then decrypts the file using the key provided to it. When the decryption is done it asks the user if it would like to delete the encrypted file. Depending on the user’s choice it either deletes the file or keeps it.

If the user chooses Generate Key feature, the software will generate a key and asks if the user wishes to save the key by typing in the full path including the wished filename. As long as the menu isn’t closed the generated key will be stored when the user decides to encrypt a file. If a user encrypts a file with the generated key, the variable will reset until a new key is generated.

To return to main menu, the user simply types in the number “0” and hits enter.
- **AES vs XOR**: This option does an encryption time test between AES and XOR. It prompts the user to type in the path to the directory containing files to be test encrypted, and how many times the user wishes to encrypt each file. It starts off with creating a new temporary directory called “encrypted files”. Then it encrypts each file in the directory with XOR encryption and displaying the time in nanoseconds. When all the files have been encrypted with XOR, the software will encrypt with AES and display the time it took to encrypt in nanoseconds. Lastly it deletes all the encrypted files along with the temporary created directory.

- **FTP Upload**: Asks the user which file to upload by typing in the full pathname, IP-address or Hostname to the server along with Username, Password and which Port to connect on. The software will then try connecting to the FTP server and upload the file if the connection is accepted, if not, it will return an error message to the user explaining what went wrong in the process.
5.2.2 Calculations

The files tested are listed into three classes in the chart as seen in Table 5.2.2.1.

Table 5.2.2.1: Chart listing the files of each class that will be used in the experiment.

<table>
<thead>
<tr>
<th></th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type of Content</td>
<td>0-1 A4 document pages with text</td>
<td>2-10 A4 document pages with text</td>
<td>11-30 A4 document pages with text</td>
</tr>
<tr>
<td>Text Format</td>
<td>Standard formatting</td>
<td>Standard formatting</td>
<td>Standard formatting</td>
</tr>
<tr>
<td>File Size</td>
<td>1-4 kilobyte</td>
<td>6-36 kilobyte</td>
<td>48-179 kilobyte</td>
</tr>
<tr>
<td>Type</td>
<td>.txt</td>
<td>.txt</td>
<td>.txt</td>
</tr>
<tr>
<td>No. Files</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

The Average time of encrypting files will be calculated using the formula:

\[
\frac{1}{n} \sum_{i=1}^{n} a_i = \frac{1}{n}(a_1 + a_2 + \ldots + a_n)
\]

(5.1)

The expected time for each class of files will be calculated using the student's t-test:

\[
\begin{align*}
\{H_0: \mu_1 &= \mu_2 \\
\{H_1: \mu_1 &< \mu_2 \\
\end{align*}
\]

\[
\sqrt{\frac{(n_1 - 1)s_1^2 + (n - 1)s_2^2}{n_1 + n_2 - 2} \left( \frac{1}{n_1} + \frac{1}{n_2} \right)}
\]

(2.1)

Results from this is should be \( u < -t_\alpha - (n_1 + n_2 - 2) \) and the \( p - value = < -t_\alpha(n_1 + n_2 - 2) \).

Where the \( \bar{x}_n \) is the average for each encryption type of each file example:

\( \bar{x}_1 \) is equal to the average time for the 10 files to be encrypted using XOR.

\( \bar{x}_2 \) would be the same but the time for the software to encrypt 10 files using AES.
\( n_1 \) and \( n_2 \) in this case would be the same since the sample size is the same for each encryption type used which is 10 files.

\( t_\alpha \) is the percentile of the t-distribution.

The sample variance \( s^2 \) will be calculated using the formula:

\[
s^2 = \frac{1}{n-l} \left( \sum_{i=l}^{n} x_i^2 - n \bar{x}^2 \right)
\]

(5.2)

This will be done for each encryption method for each file class as seen in Table 5.2.2.2.

Table 5.2.2.2: Chart listing each class and the numbered results.

<table>
<thead>
<tr>
<th>Name</th>
<th>XOR</th>
<th>AES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class 1 (10 files of 0-1 pages of text)</td>
<td>Result 1</td>
<td>Result 4</td>
</tr>
<tr>
<td>Class 2 (10 files of 2-10 pages of text)</td>
<td>Result 2</td>
<td>Result 5</td>
</tr>
<tr>
<td>Class 3 (10 files of 11-30 pages of text)</td>
<td>Result 3</td>
<td>Result 6</td>
</tr>
</tbody>
</table>

The student’s t-test will be performed a total of three times one for each class type comparing each type of encryption system.

The statistical evidence will be presented in the form of a bar diagram and a box diagram.

The bar diagram will serve as a normal way of showing the statistical evidence. In this case is the average time for each encryption system to encrypt each file class. By using the formula for calculating the average time.

Two box charts will be presented one for each encryption system.

To plot these charts the median and the first and third quartile needs to be determined by using the following formulas:

Median: \( md (X_1, \ldots, X_n) = \)

\( \{X_{\frac{n+i}{2}}\} \) if \( n \) is odd.

\( \{\frac{1}{2}X_{\frac{n}{2}} + X_{\frac{n}{2}+2}\} \) if \( n \) is even.

First quartile: \( Q_1 = \)
\{md(X_{(1)}, \ldots, X_{\frac{n-1}{2}})\} if n is odd.
\{md(X_{(1)}, \ldots, X_{\frac{n}{2}})\} if n is even.

Third quartile: \(Q_3 =
\{md(X_{(\frac{n+3}{2})}, \ldots, X_{(n)})\} \text{ if } n \text{ is odd.}
\{md(X_{(\frac{n+1}{2}+1)}, \ldots, X_{(n)})\} \text{ if } n \text{ is even.}

(5.4) (5.5)
Chapter 6 Results

The results from the experiment are presented here with figures and charts to support findings and observations.

6.1 Software

The software has successfully been developed. The exit option in the menu exists which can be seen in Figure 6.1.1. In Figure 6.1.2 and Figure 6.1.3 the menu for AES and XOR can be seen.

![Image of the software’s main menu.](image1)

Figure 6.1.1: The software’s main menu.

![Image of the software’s AES menu.](image2)

Figure 6.1.2: The software’s AES menu.

![Image of the software’s XOR menu.](image3)

Figure 6.1.3: The software’s XOR menu.

The encryption class is successfully encrypting files (Figure 6.1.4) by creating a duplicate file with the same name as input file with an extension. AES has an extension of “aes.enc” which can be seen in Figure 6.1.5. The encryption method used is 256-bit AES as a personal choice, since it’s considered to be one of the commonly known and secure encryption algorithms used today. The second encryption method used is XOR because of its encryption speed and robustness. Figure 6.1.6 shows the software encrypting the file using XOR with an “xor.enc” extension. XOR has an option to generate a key as can be seen in 6.1.7.
Figure 6.1.4: File encrypted with 256-bit AES and saved locally.

Figure 6.1.5: Successful encryption, creating a duplicate file.

Figure 6.1.6: Successful encryption using XOR.

Figure 6.1.7: Successful key generation.

The decryption class decrypts the encrypted file given to it with the matching password/key. For AES the encoding file and key file has to match too, as can be seen in (Figure 6.1.8). As seen in Figure 6.1.9, the software successfully removes the encrypted file along with its Salt & IV file, leaving nothing but the decrypted file. For XOR same principles apply as can be seen in Figure 6.1.11.
FTP Upload class uploads the given file to the FTP server with the correct credentials and hostname/ip address along with the port number and displays error messages containing the problem. As seen in Figure 6.1.1.2 and Figure 6.1.1.3, the software encrypts and uploads the file to the “cloud” using FTP.
As seen in Figure 6.1.1.4, *AES vs XOR* successfully takes in the directory provided to it along with files in the given directory. It creates an arraylist of the filenames and creates a temporary directory which in turn encrypts the files one by one and measures the time in nanoseconds for the time taken to encrypt each file. After it encrypts the files with both encryption algorithms it deletes the temporary directory along with the encrypted files within it. An issue that came up was if the directory contained a sub directory the software couldn’t enter the subdirectory and encrypt it’s files.

![Attn! Make sure there are no folders in your test directory!](image1)

The following files will be used in this test: [1.txt, 10.txt, 2.txt, 3.txt, 4.txt, 5.txt, 6.txt, 7.txt, 8.txt, 9.txt]

<table>
<thead>
<tr>
<th>XOR Encrption Time Test</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>XOR Processing Time for file 1.txt = 122538008(ns)</td>
<td></td>
</tr>
<tr>
<td>XOR Processing Time for file 10.txt = 9039146(ns)</td>
<td></td>
</tr>
<tr>
<td>XOR Processing Time for file 2.txt = 80439782(ns)</td>
<td></td>
</tr>
<tr>
<td>XOR Processing Time for file 3.txt = 2853298(ns)</td>
<td></td>
</tr>
<tr>
<td>XOR Processing Time for file 4.txt = 2168401(ns)</td>
<td></td>
</tr>
<tr>
<td>XOR Processing Time for file 5.txt = 1933520(ns)</td>
<td></td>
</tr>
<tr>
<td>XOR Processing Time for file 6.txt = 3239388(ns)</td>
<td></td>
</tr>
<tr>
<td>XOR Processing Time for file 7.txt = 2173011(ns)</td>
<td></td>
</tr>
<tr>
<td>XOR Processing Time for file 8.txt = 3197531(ns)</td>
<td></td>
</tr>
<tr>
<td>XOR Processing Time for file 9.txt = 7358270(ns)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AES Encryption Time Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>AES Processing Time for file 1.txt = 393178738(ns)</td>
</tr>
<tr>
<td>AES Processing Time for file 10.txt = 107795238(ns)</td>
</tr>
<tr>
<td>AES Processing Time for file 2.txt = 109747966(ns)</td>
</tr>
<tr>
<td>AES Processing Time for file 3.txt = 104965888(ns)</td>
</tr>
<tr>
<td>AES Processing Time for file 4.txt = 107295684(ns)</td>
</tr>
<tr>
<td>AES Processing Time for file 5.txt = 106457166(ns)</td>
</tr>
<tr>
<td>AES Processing Time for file 6.txt = 131858790(ns)</td>
</tr>
<tr>
<td>AES Processing Time for file 7.txt = 105699385(ns)</td>
</tr>
<tr>
<td>AES Processing Time for file 8.txt = 105035323(ns)</td>
</tr>
<tr>
<td>AES Processing Time for file 9.txt = 1093136309(ns)</td>
</tr>
</tbody>
</table>

Figure 6.1.1.4: Successfully encrypting files and presenting processing time in nanoseconds.
6.2 Result from the test XOR vs AES.

The calculations will be presented here with the correct values from the tests.

6.2.1 Test data with initializing value

Table 6.2.1.1: Encryption time with initializing value XOR vs AES Class 1

<table>
<thead>
<tr>
<th>C1 0-1 Page size</th>
<th>XOR File</th>
<th>Test 1 XOR time in ns</th>
<th>AES File</th>
<th>Test 1 AES time in ns</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>302639697</td>
<td>1</td>
<td>587688255</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>296589264</td>
<td>2</td>
<td>552783093</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>282167229</td>
<td>3</td>
<td>550052167</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>281564830</td>
<td>4</td>
<td>540679068</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>276781961</td>
<td>5</td>
<td>532086798</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>276353650</td>
<td>6</td>
<td>584165053</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>274717778</td>
<td>7</td>
<td>55546515</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>281937086</td>
<td>8</td>
<td>532828150</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>295179983</td>
<td>9</td>
<td>544265037</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>288429646</td>
<td>10</td>
<td>532093508</td>
</tr>
</tbody>
</table>

Average time

<table>
<thead>
<tr>
<th>XOR File</th>
<th>Test 1 XOR time in ns</th>
<th>AES File</th>
<th>Test 1 AES time in ns</th>
</tr>
</thead>
<tbody>
<tr>
<td>285636112,4</td>
<td>551208764,4</td>
<td>8,16714×10^+17</td>
<td>3,04198×10^+18</td>
</tr>
</tbody>
</table>

X^2

Table 6.2.1.2: Encryption time with initializing value XOR vs AES Class 2

<table>
<thead>
<tr>
<th>C2 2-10 Page size</th>
<th>XOR file</th>
<th>Test 2 XOR time in ns</th>
<th>AES File</th>
<th>Test 2 AES time in ns</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>11</td>
<td>324778039</td>
<td>11</td>
<td>534253221</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>328317820</td>
<td>12</td>
<td>541725173</td>
</tr>
<tr>
<td>13</td>
<td>13</td>
<td>354267535</td>
<td>13</td>
<td>530461585</td>
</tr>
<tr>
<td>14</td>
<td>14</td>
<td>362160297</td>
<td>14</td>
<td>544290301</td>
</tr>
<tr>
<td>15</td>
<td>15</td>
<td>454879526</td>
<td>15</td>
<td>573987045</td>
</tr>
<tr>
<td>16</td>
<td>16</td>
<td>433043569</td>
<td>16</td>
<td>534225983</td>
</tr>
<tr>
<td>17</td>
<td>17</td>
<td>125890580</td>
<td>17</td>
<td>545071919</td>
</tr>
<tr>
<td>18</td>
<td>18</td>
<td>1326839062</td>
<td>18</td>
<td>545008757</td>
</tr>
<tr>
<td>19</td>
<td>19</td>
<td>540964477</td>
<td>19</td>
<td>555917065</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>958670176</td>
<td>20</td>
<td>549692544</td>
</tr>
</tbody>
</table>

Average time

<table>
<thead>
<tr>
<th>XOR File</th>
<th>Test 2 XOR time in ns</th>
<th>AES File</th>
<th>Test 2 AES time in ns</th>
</tr>
</thead>
<tbody>
<tr>
<td>634282630,7</td>
<td>545463359,3</td>
<td>5,42142×10^+18</td>
<td>2,97674×10^+18</td>
</tr>
</tbody>
</table>

X^2

Table 6.2.1.3: Encryption time with initializing value XOR vs AES Class 1
### 6.2.2 Test data without initializing value

#### Table 6.2.2.1: Encryption time without initializing value XOR vs AES Class 1

<table>
<thead>
<tr>
<th>C1 0-1 Page size</th>
<th>XOR File</th>
<th>Test 1 XOR time in ns</th>
<th>AES File</th>
<th>Test 1 AES time in ns</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4803792</td>
<td>1</td>
<td>104720617</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3650313</td>
<td>2</td>
<td>105800671</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>4910772</td>
<td>3</td>
<td>105631320</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2601051</td>
<td>4</td>
<td>105210510</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>2778297</td>
<td>5</td>
<td>105560659</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1609027</td>
<td>6</td>
<td>109092151</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>1287301</td>
<td>7</td>
<td>105614346</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>1111634</td>
<td>8</td>
<td>105581581</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>2795271</td>
<td>9</td>
<td>104655482</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>8965710</td>
<td>10</td>
<td>106890593</td>
<td></td>
</tr>
<tr>
<td>Average time</td>
<td>3451316,8</td>
<td></td>
<td>105875793</td>
<td></td>
</tr>
<tr>
<td>X^2</td>
<td>1,68680622406854</td>
<td></td>
<td>1,12112×10^+17</td>
<td></td>
</tr>
</tbody>
</table>

#### Table 6.2.2.2: Encryption time without initializing value XOR vs AES Class 2
<table>
<thead>
<tr>
<th>C2 2-10 Page size</th>
<th>XOR file</th>
<th>Test 2 XOR time in ns</th>
<th>AES File</th>
<th>Test 2 AES time in ns</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>11774008</td>
<td>11</td>
<td>104664561</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>13358166</td>
<td>12</td>
<td>109010831</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>28324175</td>
<td>13</td>
<td>107286140</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>33469038</td>
<td>14</td>
<td>114226749</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>45243835</td>
<td>15</td>
<td>109538225</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>62641544</td>
<td>16</td>
<td>106301222</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>382530129</td>
<td>17</td>
<td>108440803</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>438314753</td>
<td>18</td>
<td>108176316</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>91484036</td>
<td>19</td>
<td>131457082</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>306175137</td>
<td>20</td>
<td>115392465</td>
<td></td>
</tr>
</tbody>
</table>

Average time 114313482,1 111449439,4

X^2 4,48772×10^+17 1,24753×10^+17

All these values are used in the upcoming calculations.

<table>
<thead>
<tr>
<th>C3 11-30 Page size</th>
<th>XOR file</th>
<th>Test 3 XOR time in ns</th>
<th>AES file</th>
<th>Test 3 AES time in ns</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>12810965705</td>
<td>21</td>
<td>150034719</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>1331581272</td>
<td>22</td>
<td>141305073</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>1186811816</td>
<td>23</td>
<td>141197700</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>8228043740</td>
<td>24</td>
<td>147053123</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>3363272326</td>
<td>25</td>
<td>143792434</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>3426265994</td>
<td>26</td>
<td>145852668</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>5928810002</td>
<td>27</td>
<td>161682010</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>4290389218</td>
<td>28</td>
<td>145784376</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>12257080057</td>
<td>29</td>
<td>151272675</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>763611501</td>
<td>30</td>
<td>152686299</td>
<td></td>
</tr>
</tbody>
</table>

Average time 5358683163 148066107,7

X^2 4,62431×10^+20 2,19579×10^+17
6.3 Calculations for statistical evidence

Start of by calculating the median using the equation (5.3). The second formula will be used since \( n = 10 \) which is an even number. Using the formula to calculate the median for each test resulting in:

Class 1 median for XOR:
\[
\left\{ \frac{1}{2} X_{\left(\frac{10}{2}\right)} + X_{\left(\frac{10}{2}+1\right)} \right\} = \frac{2778297 + 2795271}{2} = 2786784
\]

Class 1 median for AES:
\[
\left\{ \frac{1}{2} X_{\left(\frac{10}{2}\right)} + X_{\left(\frac{10}{2}+1\right)} \right\} = \frac{105581581 + 105614346}{2} = 105597963,5
\]

As shown above we calculated the median using the formula (5.3) where the 5th and 6th test values from each sorted list is used as the X within the equation. The same calculations using the formula (5.3) for Class 2 and Class 3. The results are shown in Table 6.3.1.

Table 6.3.1: Encryption time XOR vs AES average time.

<table>
<thead>
<tr>
<th>Median</th>
<th>XOR</th>
<th>AES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class 1</td>
<td>2786784 ns</td>
<td>105597963,5ns</td>
</tr>
<tr>
<td>Class 2</td>
<td>53942689,5ns</td>
<td>108725817ns</td>
</tr>
<tr>
<td>Class 3</td>
<td>3858327606ns</td>
<td>146452895,5ns</td>
</tr>
</tbody>
</table>

Calculating the quartiles for the box charts.

Same with this case the second formula will be used since \( n = 10 \)
First quartile is calculated using (5.4) where the X is the indicator of which value from the test list to be extracted as the result from the formula in this case \( X_{(3)} \).

Class 1 first quartile for XOR using (5.4):
\[
md(X_{(1)}, \ldots, X_{(5)}) = md(X_{(3)}) = Q_1
\]

The same steps will be taken for all the classes as shown in Table 6.3.2.

Table 6.3.2: Encryption time XOR vs AES first quartile.

<table>
<thead>
<tr>
<th>First quartile</th>
<th>XOR</th>
<th>AES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class 1</td>
<td>1609027 ns</td>
<td>105210510 ns</td>
</tr>
<tr>
<td>Class 2</td>
<td>28324175 ns</td>
<td>107286140 ns</td>
</tr>
<tr>
<td>Class 3</td>
<td>1331581272 ns</td>
<td>143792434 ns</td>
</tr>
</tbody>
</table>
The third quartile is calculated using (5.5) where the X is the indicator of which value from the test list to be extracted as the result from the formula in this case $X_{(8)}$.

Class 1 third quartile for XOR using (5.5):

$$md(X_{(6)}, ..., X_{(10)}) = md(X_{(8)}) = Q_3$$

The same steps will be taken for all the classes as shown in Table 6.3.3.

Table 6.3.3: Encryption time XOR vs AES third quartile.

<table>
<thead>
<tr>
<th>Third quartile</th>
<th>XOR</th>
<th>AES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class 1</td>
<td>4803792 ns</td>
<td>105800671 ns</td>
</tr>
<tr>
<td>Class 2</td>
<td>306175137 ns</td>
<td>114226749 ns</td>
</tr>
<tr>
<td>Class 3</td>
<td>8228043740 ns</td>
<td>151272675 ns</td>
</tr>
</tbody>
</table>

Table 6.3.4: Encryption time XOR vs AES first smallest observed value.

<table>
<thead>
<tr>
<th>Smallest observed value</th>
<th>XOR</th>
<th>AES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class 1</td>
<td>1111634 ns</td>
<td>104655482 ns</td>
</tr>
<tr>
<td>Class 2</td>
<td>11774008 ns</td>
<td>104664561 ns</td>
</tr>
<tr>
<td>Class 3</td>
<td>763611501 ns</td>
<td>141197700 ns</td>
</tr>
</tbody>
</table>

Table 6.3.5: Encryption time XOR vs AES first largest observed value.

<table>
<thead>
<tr>
<th>Largest observed value</th>
<th>XOR</th>
<th>AES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class 1</td>
<td>8965710 ns</td>
<td>109092151 ns</td>
</tr>
<tr>
<td>Class 2</td>
<td>438314753 ns</td>
<td>131457082 ns</td>
</tr>
<tr>
<td>Class 3</td>
<td>12810965705 ns</td>
<td>161682010 ns</td>
</tr>
</tbody>
</table>
6.4 Calculations for Student’s t-test

\[ \begin{align*}
{H_0: \mu_1 &= \mu_2} \\
{H_1: \mu_1 &< \mu_2}
\end{align*} \]

\[ x_1 - x_2 \]
\[ \sqrt{\frac{(n_1 - 1)s_1^2 + (n - 1)s_2^2}{n_1 + n_2 - 2 \left( \frac{1}{n_1} + \frac{1}{n_2} \right)}} \]

\[ \{H_0: \mu_1 = \mu_2 \text{ where } \mu_1 \text{ is the expected time for XOR to encrypt a file and } \mu_2 \text{ is the expected time for AES to encrypt a file.} \]

\[ \{H_1: \mu_1 < \mu_2 \text{ states that the expected time for AES to encrypt a file should be greater than the expected time for XOR to encrypt a file.} \]

\[ x_1 - x_2 \text{ Average time for } \text{XOR} - \text{AES} \]

\[ n_1 = n_2 \text{ Since the sample size is the same in this case 10 files.} \]

**Student’s t-test for class 1:**

The first step is to calculate the sample variance \( s_n^2 \) will be calculated using the formula (2.1)

\[ s^2 = \hat{\sigma}^2 = \frac{1}{n - l} \left( \sum_{i=l}^{n} x_i^2 - n\bar{x}^2 \right) \]

\[ (5.2) \]

\[ \frac{1}{9} \left( \sum_{i=1}^{10} 168680622406854 - 10 \times 3451316,8^2 \right) \]

\[ s_1^2 = \frac{1}{9} \left( 168680622406854 - 1.191158765396224 \times 10^6 \right) = \]
\[ s_1^2 = 1.87422912461883594004195 \times 10^{13} \]
The same formula (5.2) is used to calculate $s_2^2$ as following.

$$s_2^2 = \frac{1}{9} \left( \sum_{i=1}^{10} 1.12112 \times 10^{17} - 1.1209683543378849 \times 10^{17} \right) = 1.684951801278 \times 10^{12}$$

With the $s^2$ values calculated the student’s t-test is ready to be calculated using (2.1).

$$\frac{\bar{x}_1 - \bar{x}_2}{\sqrt{(n_1 - 1)s_1^2 + (n - 1)s_2^2 \left( \frac{1}{n_1} + \frac{1}{n_2} \right)}}$$

$$\sqrt{\frac{(9)1.87422912461883594004195 \times 10^{13} + (9)1.684951801278 \times 10^{12}}{18} \left( \frac{1}{10} + \frac{1}{10} \right)} - 102424476.2$$

$$= -71.66364166$$

$$u < -t_\alpha (n_1 + n_2 - 2) = -71.66364166 < -t_\alpha (18) = -71.66364166 < -t_\alpha (18) = -71.66364166 < -t_\alpha (18)$$

and the $p-value = < 0.00001 \approx 0$.

Which makes the statement true on every level so $\{ H : \mu_1 = \mu_2 \}$ is then discarded resulting in the $\{ H_1 : \mu_1 < \mu_2 \}$ statement as true.

**Student’s t-test for class 2:**

The same formula (2.1) is used for both the $s^2$ values in class 2

$$s_2^2 = \frac{1}{9} \left( \sum_{i=1}^{10} 4.48772 \times 10^{17} - 10 \times 141331482.1^2 \right)$$
\[ \frac{1}{9} \left( \sum_{i=1}^{10} 1.24753 \times 10^{17} - 10 \times 111449439.4^2 \right) \]

\[ s_1^2 = \frac{1}{9} (4.48772 \times 10^{17} - 1.997458783258262041 \times 10^{17}) = 2.766956907490819954 \times 10^{16} \]

\[ s_2^2 = \frac{1}{9} (1.24753 \times 10^{17} - 1.242097754257427236 \times 10^{17}) = 6.035828602858626 \times 10^{13} \]

Calculating student’s t-test for class 2 using the formula (2.1) with the values as shown below.

\[
\frac{141331482.1 - 111449439.4}{\sqrt{(10 - 1)2.766956907490819954 \times 10^{16} + (10 - 1)6.035828602858626 \times 10^{13} \times \left( \frac{1}{10} + \frac{1}{10} \right)}}
\]

\[ = 0.567460865 \]

\[ u < -t_\alpha (n_1 + n_2 - 2) = 0.567460865 < -t_\alpha (18) = 0.567460865 < -t_\alpha (18) \]

Which makes the statement not true on every level so \( H_0: \mu_1 = \mu_2 \) is then not discarded resulting in the \( H_1: \mu_1 < \mu_2 \) statement as false.

**Student’s t-test for class 3:**

The same formula (2.1) is used for both the \( s^2 \) values in class 3

\[ \frac{1}{9} \left( \sum_{i=1}^{10} 4.62431 \times 10^{20} - 10 \times 5358683163^2 \right) \]

\[ s_1^2 = \frac{1}{9} (4.62431 \times 10^{20} - 2.871548524356315783424 \times 10^{20}) \]

\[ s_2^2 = 1.947512750715204685084 \times 10^{19} \]
\[
\frac{1}{9} \left( \sum_{i=1}^{10} 2.19579 \times 10^{17} - 10 \times 148066107.7^2 \right)
\]

\[s_2^2 = \frac{1}{9} \left( 2.19579 \times 10^{17} - 2.192357224942799929 \times 10^{17} \right) =
\]

\[s_2^2 = 3.814194508000078 \times 10^{13}
\]

Calculating student’s t-test for class 3 using the formula (2.1) with the values as shown below.

\[
\frac{5358683163 - 148066107.7}{\sqrt{(10 - 1)1.947512750715204685084 \times 10^{19} + (10 - 1)3.814194508000078 \times 10^{13} \cdot \frac{1}{10 + 10 - 2}}}
\]

\[
\frac{5210617055.3}{\sqrt{(9)1.947512750715204685084 \times 10^{19} + (9)3.814194508000078 \times 10^{13} \cdot \frac{1}{18}}}
\]

\[u > t_\alpha(n_1 + n_2 - 2) = 3.7337787227 > t_\alpha(18) = 3.7337787227 > t_\alpha(18)
\]

\[3.7337787227 > t_\alpha(18)\text{ and the } p\text{-value} = < 0.00001 \approx 0
\]

Which makes the statement true every level \(\{H_0: \mu_1 = \mu_2\}\) is then discarded resulting in the \(\{H_1: \mu_1 > \mu_2\}\) statement as true.
6.5 Statistical evidence without initializing value

Figure 6.5.1 Bar plot of average time XOR vs AES.

Here it is displayed in a form of direct comparison between XOR and AES how much faster XOR is compared to AES. The values of class 1 if converted from nanoseconds would be: 0.00345132 seconds for XOR versus the 0.105876 Seconds of AES.

Figure 6.5.2 Box plot of average time XOR for class 1-3.
Figure 6.5.3 Box plot of average time AES class 1-3.

The boxplots represent the time and spread for each encryption type and the spread of the test values. With the “boxes” representing majority of the values. The “whiskers” above and below represent the largest and smallest observed value when encrypting. This also show that both XOR and AES are pretty stable when it comes to encryption.

6.6 Statistical evidence with initializing value

Figure 6.6.1 Bar plot of average time XOR vs AES with the initializing value.
Figure 6.6.2 Box plot of average time XOR for class 1-3 with the initializing value.

Figure 6.6.3 Box plot of average time AES class 1-3 with the initializing value.
Chapter 7 Discussion

7.1 Method Discussion
The literature was focused on cloud computing, cloud encryption and cryptography. Mostly newly developed solutions to cloud security to gain a deeper understanding of the field as well as projects which had similar ideas. It was a good setup for the thesis. The literature provided a knowledge foundation to get started, for security reasons all the methods were only presented in theory. When it came to the encryption systems themselves they were too hard to comprehend mathematically so creating a new encryption method was out of the question.

The end goal was to create a software that could encrypt a file before it was to be uploaded to the cloud. As time was a factor the choice was made to develop the project in Java as experience in it had been gained prior and a basic level had already been established. Another programing language might have been better to use since it felt like Java is not the most optimal language when it comes to cryptographic programming, which are always relevant when it comes to encryption systems.

To secure a user’s file on the cloud, an encryption software was developed. Which is where the direction of the project had to be altered since the question of what system is to be implemented presented itself. After research with more literature and with expert help the two methods of XOR and AES was selected. The reasoning for this was that XOR was believed to be a really fast method of encryption and AES also one of the faster ones but also one of the most secure. For this reason, a method of comparing them had to be developed.

The solution was a part of the software were to encrypt the same files and record the time in nanoseconds presenting the time of encryption for both methods. The comparison was to be made without the software but by hand with the help of expert help the solution was to be a student’s t-test so not only compare the statistical values from the software but also the expected time by performing this mathematical test.
With no prior knowledge to how fast these methods were no hypothesis of results were developed but the results themselves were polarizing.

As events in modern times has proved that cloud security is a real problem as well as the literature study shows the development of new cloud security solutions. This software was developed to combat an existing threat to private data stored on clouds.

7.2 Result Discussion
Based on the results from the different tests it proved to be true that the implemented version of XOR is faster than the implemented version of AES when it came to the first class. This is proved with in the results with $p-value = < 0.00001 \approx 0$
which means that there is almost a 100% certainty that the results will be a value of -71 or smaller on the first test. That meaning that there is an almost 100% chance that the implemented version of XOR encryption will be faster than the AES for txt files between 0-1 pages.

A different result is shown in the second student t-test. Here the results show that the statement of XOR is faster than XOR as false. Which is shown in the calculations but also can be confirmed in the statistical evidence since XOR and AES are very close in terms of encryption speed for the second-class txt files with 2-10 pages. The difference in the average speeds of these two systems are very small it is around 0.03 seconds.

The third test showed the opposite results compared to the first one. Instead of showing that XOR is faster, the test proved that AES was faster. This was confirmed in the results where the \( p-value = < 0.00001 \approx 0 \) which means that there is almost a 100% certainty that the results will be a value of 3.7337787227 or large. Meaning that the AES encryption will be faster in terms of encryption speed when it comes to txt files of class 3, 11-30 pages. This can also be confirmed by looking at the average encryption times of class 3 for AES and XOR. The difference is almost 4.5 seconds.

This means in no way that there are no faster versions of either XOR nor AES. The choice was made to use existing encryption systems and implement them into a self-developed software. That can mean that the implemented versions might not be the fastest versions of each system. But they are still a version of the two existing systems compared to each other which why the experiment tests and results are relevant and worth observing. The choice was made to do the calculations on the test values without the initializing value. But the results from the performance tests are also presented in the result. It is included for the reason that the first file that is encrypted the encryption time is inflated naturally by the code as the program is trying to find the directory the file is in. As shown in the results this affects the encryption time a lot, as shown in the figure below.
The observation was made that there was a lack of linear progression in terms of encryption time. As the encryption time for AES class 1 was larger than the one of class 2.

Therefore, another test was run to confirm the linear progression of AES and XOR. The test was done in the same way as previous test but with a 1MB large file of text. As the theory is that encryption speed is time vs amount of characters.

As the results shows we were able to confirm that there was indeed a direct correlation between time vs amount of characters when it came to AES.
When it comes to speed the results showed that in all cases of the test with the three different classes the implemented version of XOR was more efficient when encrypting class 1. AES and XOR are tied when it comes to class 2. Lastly AES is faster when it comes to the larger files of class 3. This was also confirmed with the confirmation test with the 1MB file. The reason why AES performs better when it comes to larger files is that it encrypts in terms of blocks which is faster compared to XOR which encrypts character by character. The results differ a bit from a similar test that was done in [21]. Where the authors used different test data but the same trend was observed that AES was very stable in terms of time when it comes to handling smaller and larger files. XOR is fast when handling smaller but takes longer when handling larger files. Which was also the case in this experiment. The results differ from this thesis there are many factors why that is, but the same trends could be observed.

But there was also a security question when it comes to encrypting files. The whole purpose of the software is to encrypt files before they are uploaded to the cloud. Since different cloud services has varying security levels and different geographical areas has different jurisdiction when it comes to handling data on the cloud. This was a way for a user to secure their own data so that even if a cloud service is breached through some kind of an attack. The users file would still be safe and it would be really hard to crack the encrypted files.

The question when it comes to cloud encryption is security versus efficiency. Which is why the option to choose which encryption method to use was implemented. The threat of keys being sent when dealing with sending encrypted files was eliminated by generating and storing the keys locally. It gives the user the responsibility and security of the keys. And user themselves are potential security threats. Based on the tests and the understanding gained from literature regarding the encryption systems, the recommendation would be to use XOR.
encryption when uploading files of less importance as well as mass encrypting a large number of files. AES should be used when encrypting files of higher importance as it is deemed to be more secure since it is more commonly used today.

That does not mean that XOR is a weak encryption system but as presented in the results they operate differently when it comes to encryption, which is why AES is thought to be the slower one out of the two. AES is older, but time has proven it is still very strong and that is why it is still in use today. There are of course modifications to that can optimize the speed and security of every system. But in the end a cipher's strength is how strong the key is. The two methods used is inherently strong on their own but for a higher security level all that is needed is to enlarge the key length, exemplary from 256 bit to 512 bit.

There could have been interference when it comes to the processing power of the computer's CPU which might alter the values, but even this is accounted for and deemed relevant since that is a realistic event that might occur when any program is executed. There was also the case that because of how the program was designed, the first file that is to be encrypted takes longer time since the program needs to locate the directory of said file.

This first file was also used as the test result since the goal was to also emulate a realistic scenario, since realistically an user would only run the program once to encrypt a selected file. An example of this this scenario occurring is when the choice is made by the user to encrypt 1 or more files the first one will always be slower. After observing this phenomenon, the decision was made to use the first file value in the tests to represent a more realistic picture. The fault is in the program itself, but it was still the best way to perform the tests.

The program also has a flaw of when the encrypted file is to be sent to the cloud through ftp and fails to upload, the software will delete the encrypted file stored temporary but keep the encoding & randomness file meaning that the user will have to re-encrypt the file. An issue that came up with AES vs XOR was if the directory contained a sub directory the software couldn’t enter the subdirectory and encrypt its files. The main functions work so it is more a question of optimizing the software and proofing it against every imaginable event that might crash the software, that is something for the future.
7.3 Future works

For future projects and further development, it is recommended to either develop a new encryption method and comparing that to existing ones. Another option is to try to find the optimal version of each encryption method and comparing them. There is also an interest to try to implement and compare more than two algorithms, it would give a wider perspective of the efficiency of existing encryption methods. An interest to do would be to implement existing encryption algorithms into C, Python and Java, to compare which of the programming languages is most optimal to use for cryptographic tasks.

More optimal testing equipment should be used, exemplary test it on existing market computers that are being used today and comparing those results for a more realistic expected result.
Chapter 8 Conclusion

8.1 Conclusion
A software was developed to protect a user’s data by encrypting it and storing the keys locally before uploading it to the cloud. This is our way of combating the security breaches that has happened these last years and secure data on the cloud against potential new threats.

Since different cloud services offer varying level of security as well as different jurisdiction depending on the geographic area when it comes to handling data on the cloud. By using this software even if a cloud service were to be breached by an attacker and they would have access to a user’s files. This way the files would still be securely encrypted and very hard to decrypt. This leaves the responsibility and security of the files in the hands of the user themselves. To determine the optimal encryption system to use two were selected AES and XOR. These were compared in terms of time efficiency and the results showed that XOR was the more efficient one out of the two. As the efficiency varies a lot the result is that the two methods should be used for different purposes. As the society moves more towards digitalization more research, methods and discussions should be had to secure the future of the cloud.
Appendix

AES Encoder class code used in AES vs XOR Experiment

```java
import java.io.File;
import java.io.FileInputStream;
import java.security.AlgorithmParameters;
import java.security.SecureRandom;
import java.security.spec.KeySpec;
import javax.crypto.Cipher;
import javax.crypto.SecretKey;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.PBEKeySpec;
import javax.crypto.spec.SecretKeySpec;
public class aesencoder
{
    static void encode(String inputFile, String outputFile, String names) throws Exception
    {
        long startTime = System.nanoTime();

        FileInputStream inFile = new FileInputStream(inputFile);
        FileOutputStream outFile = new FileOutputStream(outputFile + "aes.enc");

        String password = "pa55w0rd";
        File kname = new File(inputFile);
        String key = kname.getName();
        byte[] salt = new byte[8];
        SecureRandom secureRandom = new SecureRandom();
        secureRandom.nextBytes(salt);
        FileOutputStream saltOutFile = new FileOutputStream(outputFile + key + "salt.enc");
        saltOutFile.write(salt);
        saltOutFile.close();

        SecretKeyFactory factory = SecretKeyFactory.getInstance("PBKDF2WithHmacSHA256");
       KeySpec keySpec = new PBEKeySpec(password.toCharArray(), salt, 65536, 128);
        SecretKey secretKey = factory.generateSecret(keySpec);
        SecretKey secret = new SecretKeySpec(secretKey.getEncoded(), "AES");
    }
}
```
Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
cipher.init(Cipher.ENCRYPT_MODE, secret);
AlgorithmParameters params = cipher.getParameters();

FileOutputStream ivOutFile = new FileOutputStream(outputFile+key+"iv.enc");
byte[] iv = params.getParameterSpec(IvParameterSpec.class).getIV();
ivOutFile.write(iv);
ivOutFile.close();

byte[] input = new byte[64];
int bytesRead;

while ((bytesRead = inFile.read(input)) != -1)
{
    byte[] output = cipher.update(input, 0, bytesRead);
    if (output != null)
        outFile.write(output);
}

byte[] output = cipher.doFinal();
if (output != null)
    outFile.write(output);

inFile.close();
outFile.flush();
outFile.close();

long stopTime = System.nanoTime();
long elapsedTime = stopTime - startTime;
System.out.println("AES Processing Time for file "+names+" = "+elapsedTime+"(ns")");
return;
}
}

**XOR Encoder class code used in AES vs XOR experiment**

```java
import java.io.FileInputStream;
import java.io.BufferedInputStream;
import java.io.BufferedReader;
```

```java
Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
cipher.init(Cipher.ENCRYPT_MODE, secret);
AlgorithmParameters params = cipher.getParameters();

FileOutputStream ivOutFile = new FileOutputStream(outputFile+key+"iv.enc");
byte[] iv = params.getParameterSpec(IvParameterSpec.class).getIV();
ivOutFile.write(iv);
ivOutFile.close();

byte[] input = new byte[64];
int bytesRead;

while ((bytesRead = inFile.read(input)) != -1)
{
    byte[] output = cipher.update(input, 0, bytesRead);
    if (output != null)
        outFile.write(output);
}

byte[] output = cipher.doFinal();
if (output != null)
    outFile.write(output);

inFile.close();
outFile.flush();
outFile.close();

long stopTime = System.nanoTime();
long elapsedTime = stopTime - startTime;
System.out.println("AES Processing Time for file "+names+" = "+elapsedTime+"(ns")");
return;
}
```
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.FileWriter;
import java.security.SecureRandom;

import org.apache.commons.io.FileUtils;
public class xorencoder
{
    static void encode(File inputFile, String outputFile, String names) throws Exception{
        long startTime = System.nanoTime();
        SecureRandom randomGen = new SecureRandom();
        SecureRandom random = SecureRandom.getInstance("SHA1PRNG");
        int seed = randomGen.nextInt();
        random.setSeed(seed);

        byte[] keyStream = new byte[8];
        random.nextBytes(keyStream);

        FileOutputStream outkey = new FileOutputStream(outputFile + "xor.key");
        outkey.write(keyStream);
        outkey.close();

        byte[] tempbyte = new byte[2];
        byte[] tempkey = new byte[2];

        String outfile = "";
        String outstr = "";
        String txt = "";
        String key;
        File file = inputFile;
        FileReader fr = new FileReader(file);
        BufferedReader br = new BufferedReader(fr);

        key = Integer.toString(seed);
        key = key.substring(0, 2);
        tempkey = key.getBytes("utf-8");
        int temp0 = br.read();

        while(temp0 != -1)
        {
            txt += (char)temp0;
            temp0 = br.read();
        }

        if(txt.length() % 2 != 0)
        txt += " ";
for(int i = 0; i < txt.length()/2; i++)
{
    tempbyte[0] = (byte)txt.charAt(i*2);
    tempbyte[1] = (byte)txt.charAt(i*2+1);
    tempbyte[0] = (byte)(tempbyte[0] ^ tempkey[0]);
    outstr += (char)tempbyte[0] + " " + (char)tempbyte[1];
}
outfile += file;

FileWriter fw = new FileWriter(new File(outfile));
BufferedWriter bw = new BufferedWriter(fw);

bw.write(outstr);
bw.close();
fw.close();
fr.close();
br.close();

long stopTime = System.nanoTime();
long elapsedTime = stopTime - startTime;
System.out.println("XOR Processing Time for file "+names+" = " +elapsedTime+"(ns)");
Software flowchart

AES Class

Figure 9.1: Flowchart of AES Encryption class code.
Figure 9.2: Flowchart of AES Decryption class code.
XOR Class

Figure 9.3: Flowchart of XOR Encryption class code.
Figure 9.4: Flowchart of XOR Decryption class code.
Figure 9.5: Flowchart of XOR key generator class code.
FTP Upload Class

Figure 9.6: Flowchart of FTP upload class code.
AES vs XOR Class

Figure 9.7: Flowchart of AES vs XOR class code.
References

(Retrieved 2018-03-24)
(Retrieved 2018-03-24)


[26] Java API. URL: https://docs.oracle.com/javase/7/docs/api/ (Retrieved 2018-03-25)

Josef Al-khayatt

Robin Nguyen