WEST UNIVERSITY OF TIMIŞOARA
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
BACHELOR STUDY PROGRAM: Computer Science in English

BACHELOR THESIS

SUPERVISOR:
Lect. Dr. Adrian Spătaru

GRADUATE:
Sergiu George Mureșan

TIMIŞOARA
2024

WEST UNIVERSITY OF TIMIŞOARA
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
BACHELOR STUDY PROGRAM: Computer Science in English

Computational Frameworks for a Decentralized Cloud

SUPERVISOR:
Lect. Dr. Adrian Spătaru

GRADUATE:
Mureșan Sergiu George

TIMIŞOARA
2024

Abstract

This scientific paper about components of decentralized cloud aim is to show that the security pays one if not the most important roles in the good working of a cloud platform. The paper aim is to come with a methodology for testing the performance of security algorithms that are used in a cloud platform. In order to do this different homomorphic encryption algorithms that are part of LightPHE library will be tested and analyzed. When using the internet people should be aware about their personal information that should not be exposed anywhere. For this we use encryption algorithms. Even though this algorithms provide the security layer of a decentralized cloud platform or a cloud platform in general not many people are concerned about their performance and how this can be tested. The performance of an encryption algorithm pays an important role because this is what makes them usable in a cloud platform. Because performance pays such an important role a methodology to test this performance will increase the way in which different algorithms are chosen by the platform.
Basically this scientific paper will provide a deep analysis of the most well known homomorphic algorithms from LightPHE library and will try to see similar particularities about each of them such that testing their performance can be derived from the information provided in the analysis of the algorithms that are nowadays used.

Contents

Chapter 1

Introduction

In the very critical chapter, the weight of security analysis in cloud computing is elaborated on. Most organizations increasingly turn to the cloud to keep their data and applications, making understanding its security implications critical. It first underlines the need for this analysis, driven by the motivation to create a safe place where sensitive information may be stored and compliance with regulatory standards ensured.

Besides, the introduction outlines the key objectives of the study, which are identifying and analyzing various security threats facing cloud environments, evaluating the efficiency of the current measures, and recommending improvements for the same to boost cloud security. In this way, a comprehensive framework is provided within the paper, which an organization can use to adopt security procedures for its cloud-based assets.

The chapter also crystallized the scope of the study apart from the objectives. It outlined a wide array of dimensions concerning cloud security, including data protection, identity management, access control, and, finally, threat detection. The study considers various typologies of cloud models, whether public, private, or hybrid, to have a more extensive perspective on security.

In any case, the chapter also highlights the constraints met in the study. The significant challenges include rapidly changing cloud technology, which outruns the development of security solutions, and variation in security practices across different cloud service providers. These are two areas in which time research and adaptation are to be ensured with these cloud security strategies.

Finally, it places the study within the wider context of scientific literature and points out its contribution to the existing knowledge. The study contributes to cloud security by reducing the gap that has been identified in previous studies and therefore is aimed at adding value in terms of action for practitioners and academia. The chapter adequately sets the scene for a detailed inquiry into effective security practices in the following sections by carefully examining security threats and defenses in the cloud.

1.1 Motivation

Cloud data that is in use must be protected to ensure privacy and security for the users. Traditional encryption methods are not suitable for computations on encrypted data because, before executing a computation over encrypted data, it needs to be decrypted and afterward re-encrypted. Due to this property, at this stage, the data becomes vulnerable to numerous security risks. Homomorphic encryption, which lets computations be performed directly on encrypted data without the data ever being decrypted, thus protects privacy all through the computational process and will offer a very great solution shortly. This capacity becomes decisive in cloud computing, where data is often centralized and therefore hands over a third-party service provider's duty, which demands the highest degree of trust and security assurance.

This paper is practical in nature, whereby it uses homomorphic encryption; to be precise, the library LIGHTPHE, which is the implementation of the Paillier and Damgård-Jurik algorithms. The evaluation of these homomorphic encryption algorithms will be demonstrated by comparing the time needed to add a set of random numbers. Moreover, the performance of an algorithm, such as the Advanced Encryption Standard (AES), can be tested and compared with a homomorphic encryption algorithm in the same LIGHTPHE library.Considering this particular fact, the performance of homomorphic encryption against the performance of traditional encryption, such as AES, is significant for its adoption in applications related to privacy.

This paper, therefore, seeks explicitly to analyze and compare the effectiveness of the homomorphic encryption algorithms with the AES algorithm, hence providing insights on which encryption methods are most appropriate for use in building secure cloud platforms.

This evaluation would be necessary to developers and organizations that consider implementing stringent security mechanisms in cloud-based environments.

1.2 Objectives

The cloud is more and more used nowadays that is way the objective of this study is to let others know and understand how the security is done. By creating an application which is interactive with the user and on which the user can try the algorithms implemented for different sets of numbers he will better understand how each algorithm performs. This study compares the performance, when talking about time, between Paillier algorithm, Damgård-Jurik algorithm and Advanced Encryption Standard algorithm on a data set which can be generated by the user and a data set that mimics bank transactions. By building the data set as a deposit bank account it is more easy for the user to understand what happens with the dataset, because the user can imagine that those deposit transactions where done over an year and we are summing them up in order to get the total amount that has been deposited. So by building a platform on which the user can test encryption algorithms the study objective is to make the introduction into the world of cloud easier for everyone and to make people that have no technical knowledge better understand why data has to be encrypted and why it is better to work with encrypted data even though it is not always faster. Another objective of this study is to make an analyze the performance of the algorithms implemented in LIGHTPHE library such that if a technical guy wants to use the library he knows better how it will perform.

1.3 Scope and Limitations

This research aims to demonstrate the significance of homomorphic encryption in cloud computing. Data encrypted on the cloud may be computed without decryption; hence, homomorphic encryption is essential. This paper compares the time taken by Paillier and Damgård-Jurik algorithms implemented in the LightPHE library and by the Advanced Encryption Standard (AES) algorithm. The comparison will be done over a CSV file mimicking a bank deposit with as many transactions as the user desires. The randomly generated data will be computed using the Paillier, Damgård-Jurik, and AES algorithms. The times taken for computation will be displayed on a specially created GUI, and if the user wishes for an improved view of the algorithm times, then a diagram could be shown.

The limitation here is that of the homomorphic encryption algorithms. For instance, the Paillier algorithm can perform addition operations; the Damgård-Jurik algorithm can perform addition but only with a constant and is optimized to run better with complex types of numbers rather than with standard numerical values. These inherent limitations confine the range in which computations may take place using these algorithms.

Another limitation is based on the personal computer's performance tested in this experiment. The results will be vastly different when conducted on a supercomputer or another more potent machine simply because the servers hosting a cloud are significantly more powerful machines than personal computers. This gap needs to be considered, as usually a cloud environment would run on a more vital type of hardware.

Finally, even the library itself imposes certain restrictions. The speed of LightPHE, implemented in pure Python, is bounded by several factors. Although Python is not the quickest programming environment, it is friendly to users, and one can more easily understand and implement the algorithms. For applications that require higher speed, it is better to use libraries such as Microsoft SEAL or HElib, which are developed in C++. However, for new users looking forward to cloud computing and developing their platforms, the LightPHE library will be beneficial because it is easily installed and comprehensible.

1.4 The structure of the paper

The aim of the paper and presenting why security pays an important role in the cloud platforms is presented in the State of the art chapter. This chapter also presents the approach of this experiment to the problem of security.

The chapter that presents theoretical concepts that have been used in order to run the experiment on homomorphic algorithms is Methodology chapter. This chapter also provides information on the data set and about the experiment design.

The chapter that presents the technical details about the experiment, what platform was used to run it and more technical details is the Experiment Setup . This chapter also includes the results of the experiment.

The chapter that presents details about the future work that can be done in order to improve the experiment is Conclusion and Future Work. In this chapters conclusions after the experiment are also included.

Chapter 2

State of the art

This chapter includes an introduction into the problem of the security when using cloud components. In this chapter information about the problem of the security when talking about a cloud platform is included and also how others managed to solved it and what this paper brings new to other solutions from past.

2.1 Security in the cloud

Back in 2000s the early days of the cloud computing started, these days were characterized by the emergence of innovative technologies and services, which laid the groundwork for the modern infrastructure.
Back in those days the authentication was weak because initial cloud services often relied on basic username and password in order to offer security but this was vulnerable to attacks like phising and brute force.

2.1.1 Emergence of Cloud Services

The notion of cloud computing started emerging in the early 2000s and was, in reality, an evolution of other earlier concepts such as utility and network-based computing as mentioned in "Cloud Computing: History and Overview"[SR19]. It envisioned making scalable and on-demand access over the internet to computing resources available to businesses for efficiently managing and deploying IT resources with a minimum upfront investment as mentioned in "A Review on Amazon Web Service (AWS), Microsoft Azure and Google Cloud Platform (GCP) Services" [GMM21]. In 2006, Amazon made its pioneering entrance into commercial cloud services with Amazon Web Services. AWS pioneered Simple Storage Service (S3) and Elastic Compute Cloud (EC2), which created scalable storage and compute resources. It enabled the storage of data and running applications on Amazon's infrastructure, paying only for those resources that were utilized. This was a massive shift in the delivery and consumption of IT services, as businesses were no longer bound to their physical servers. With Amazon in the lead, others followed with cloud services. Google launched its cloud platform in 2008 with a view to infrastructure, data analytics, and machine learning services. Google Cloud Platform, in this way, presented a credible array of tools for developers and businesses by using the vast infrastructure and cutting-edge data management technologies that underlie Google. In 2010, Microsoft launched its cloud service called Microsoft

Azure; it was bundled with a portfolio of services like virtual machines, databases, developer tools, and various enterprise software products from Microsoft. Microsoft Azure then became one of the biggest players on the market due to the fact that it provides many services that can be used by small business too. People who entered early in the world of cloud computing put the bases of the massive growth and made it part of most of the business nowadays.

2.1.2 Security concerns

When considering the first implementation of the cloud platforms, security is among the key aspects that service providers have factored in. User data must be protected from various malicious attacks, varying from unauthorized access to breaching data to other highly technological cyber threats. From this fact, areas where cloud security pertains include:

Several important measures in ensuring **data security** in the cloud are to be followed in maintaining confidentiality, integrity, and availability of data. This includes handling encryption at rest and in transit with advanced cryptographic methods, ensuring data cannot be accessed without proper authorization.

The sensitive information will be protected by ensuring that only the access of authorized personnel is granted through **robust authentication and authorization controls**, including multi-factor authentication (MFA) and role-based access control (RBAC).

Network security measures include firewalls, intrusion detection and prevention systems (IDPSs), in addition to secure network configurations, to protect cloud resources from outside attacks. Data in motion security is the protective control of information with virtual private networks and secure communication protocols like Transport Layer Security and Secure Sockets Layer.

Identity and Access Management(IAM) is required to place and control access by the user to cloud resources. IAM solutions assist in creating, managing, and monitoring user accounts and their permissions in a well-organized and controlled manner. IAM, per regulatory and compliance frameworks like GDPR, HIPAA, and PCI-DSS. According to these criteria, the data will be protected under statutory and regulatory requirements by both cloud providers and users. They provide a mechanism for real-time surveillance with the best possible detection of threats and responding capability for discovering and reacting to a threat.

Finally, **physical security** of data centers, where cloud servers are located, is ensured using methods like biometric access controls, surveillance, and secure hardware disposal. Understanding the Shared Responsibility Model is also crucial, as it delineates the security responsibilities between the cloud provider and the user, ensuring a clear understanding of who is responsible for securing different aspects of the cloud environment.

2.1.3 A solution to the problem: encryption

Encryption is one of the most essential tools for information security. It is merely the conversion of readable data, often called plaintext, to an encoded format called ciphertext that uniquely can be decrypted by a person possessing a decryption key of the correct type. This process ensures that information is safe, with particular people having access to it. This process ensures that information is safe, with particular people having access to it.

The main point of encryption is the privacy of data. It ensures that the information is safe when at rest on a device or in transit through a network; even when some unauthorized person gains access to it, they will not understand what the information is all about This process is very important in online banking, transmission of personal data over the Internet and emailing.

Encryption also ensures that data is integral and authentic. The cryptographic algorithms have the means to check the data for tampering and ensure it is not affected en route and has the correct origins. Indeed, this feature of encryption is essential because, through it, attackers are denied a chance of compromising data integrity or impersonating somebody.

In the past, encryption was used in military and diplomatic circles; the first to appear were simple substitutions. Nowadays, modern ways of encryption are way much more complex and invulnerable; they are based on supreme math algorithms that will ensure cryptographic solid protection. Such algorithms are designed to resist complex attacks and ensure that hardly decipherable codes can't be broken without the correct key.

In simple words, encryption is among the leading solutions about data security. It provides for confidentiality, integrity, and authentication; hence, current information security initiatives cannot do without it. The discussions of the encryption methods and how to apply them are extensively covered in the following sections.

2.2 Related Work

In this chapter some related work regarding the topic of cloud components can be found. The chapter presents Cloud components in general, decentralized cloud components and also the security of cloud where the main topic is homomorphic encryption.

2.2.1 Cloud Components

The presentation done by Shaik Mahaboob Basha in [MBRNKK23] about cloud components teach us how cloud works and also the biggest companies that moved into the cloud computing. The paper presents to us the differences between the cloud computing and grid computing by comparing their facilities and performances. It is also presented the technical aspects about cloud computing such as loose coupling which is the fundamental of virtualization in cloud computing. A business model can be taken from the paper, the authors explain this by comparing grid computing with cloud computing.
Shifting the focus to the foundational concepts of cloud computing, Jadeja and Modi, in [JM12], provide a comprehensive overview of cloud computing evolution.

While not explicitly decentralized, the paper lays the groundwork by discussing the fundamental concepts and architecture of cloud computing. The goal starting the cloud computing is to make better use of distributed resources, to combine them in order to achieve higher throughput and to be able to solve large scale computation problems, Cloud computing is meant to deal with virtualization, scalability, interoperability, quality of service and the delivery of models of the cloud, namely private, public and hybrid. This article also presents how all started, basically the history of cloud computing which all started with John McCarth in 1960 which had the way of thinking that in one day all the computation will be organized as public utility. Back in the 1960s the characteristics of the cloud computing were also explored by Douglas Parkhill in his book "Challenge of the Computer Utility". The article also presents the characteristics of cloud computing, it is well known that users access data applications or other services using the browser without depending on the device user is using or the location of the user. Another characteristic is that IT skills are not necessary in order to be able to implement the cloud. Different websites are there and can provide reliable services to the user and they are even suitable for business community.

Wang et al.'s contribution [WWR+12] addresses security risks in cloud storage through a distributed storage integrity auditing mechanism. The proposed design utilizes homomorphic tokens and distributed erasure-coded data to ensure both strong correctness guarantees and fast error localization.

Privacy concerns in centralized storage services are addressed by Hoang et al. in [HLGD20]. The proposed platform, built upon the InterPlanetary File System (IPFS), focuses on user anonymity, data confidentiality, and high data availability in decentralized storage systems.

Huang et al.'s work [HWY+18] proposes an efficient attribute-based authentication scheme for cloud computing. The scheme aims to achieve fine-grained access control and privacy preservation, addressing challenges related to user privacy during authentication.

Weinman's article [Wei11] provides an overarching view of cloud computing and its potential benefits. While not specifically focused on decentralization, the paper contributes to the foundational understanding of cloud computing. The book highlights cloud computing as a significant area of interest among IT and network technologists, academics, and the business community due to its demonstrated ability to reduce costs, minimize risks, increase revenue, and enhance the overall customer experience.The author defines cloud computing as "a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources." These resources include networks, servers, storage, applications, and services that can be quickly provisioned and released with minimal management effort or service provider interaction.Interactive workloads are also considered in this book concerning how they can be made more responsive. This is effected by two main ways once application and data transport protocols have been optimized: dispersion and parallelization. Dispersion of processing nodes has the effect of minimizing network propagation delays. Parallelism of suitable workloads can speed up processing, as is the case with online search applications.

2.2.2 Descentralized Cloud Components

There exists now a landscape in decentralized cloud computing influenced by a variety of research and analyses, each one bringing to light different facets of this evolving field. Among other researchers, Sharma et al. argue in [SSC22] for leaving centralized cloud storage architectures to adopt decentralized ones that can counter malicious attacks, integrity, accountability, and privacy. Based on blockchain, the architecture coupled with a quantum-safe encryption algorithm aims to give cloud platforms maximum security.

In contrast, Doan and colleagues [DPOB22] take a slightly different approach to the problem of decentralized cloud storage and use the InterPlanetary File System (IPFS). The article by the authors depicts some of the opportunities and a few challenges while outlining future directions and truly bringing out the element of decentralization that IPFS uses..

The integration of blockchain technology with cloud computing is explored by Uthayashangar et al. in [UDDG21]. This work proposes a secure file storage system where a single file is split and stored in blocks using advanced encryption algorithms. Smart contracts, implemented with Solidity, contribute to the overall security of the system.

In [SSMT20], Shah et al. introduce a decentralized cloud storage system leveraging blockchain and the InterPlanetary File System (IPFS). The authors emphasize decentralized secure data storage, high availability, and efficient utilization of storage resources.

In [BDCdVF$^+$20], Bacis et al. introduce a method that allows resource owners to protect and securely delete their resources in decentralized cloud storage. This method uses the All-Or-Nothing Transform for robust resource protection and incorporates strategies for slicing and decentralized allocation, ensuring both availability and security.

The application of blockchain for the security of decentralized cloud computing is explored by Nahar et al. in [NHT21]. The paper reviews the benefits and performance of cryptographic techniques in securing decentralized cloud storage, emphasizing the role of blockchain in providing security and credibility.

In [LSH20], Liu et al. propose a decentralized services computing paradigm built upon blockchain-based data governance. The paper discusses programmability, interoperability, and intelligence as key aspects of this paradigm.

Ramachandran et al. explore the creation of a decentralized data marketplace for smart cities in [RRK18]. The authors consider the benefits of a decentralized architecture, identify essential elements for a decentralized marketplace, and present a smart contract implementation for data product registration.

Finally, [SPA$^+$22] by Sood et al. introduces a decentralized and collaborative document editing application using blockchain. The proposed application integrates with block stacks and Gaia for decentralized storage, addressing issues related to security, privacy, and scalability inherent in centralized systems.

2.2.3 Security on the cloud

In the "Comprehensive Performance Analysis of Homomorphic Cryptosystems for Practical Data Processing" [SWN22] a detailed explanation on how homomorphic encryption works is done. The paper also explains why homomorphic encryption is important when using the cloud. It can be taken from this paper that data which is encrypted on the cloud can be computed without the need of decryption which saves time, space and money when working with the cloud. From the paper a comparison on the most well-known libraries used for this type of encryption can be analyzed. The libraries that are compared are Microsoft Seal, HElib, PYAno and also some algorithms implemented by the writers of the paper. From this we can see the performance of each library when dealing with homomorphic encryption. It can be also taken out from the paper the fact that homomorphic encryption became a nowdays standard because all the data on the cloud has to be secured in order to make users of the cloud sure that their personal data will not be stole by some other users.

Another paper about homomorphic encryption is "Analysis and Comparison of Various Fully Homomorphic Encryption Techniques" [CGSM19] in which the authors compare the techniques used in order to be able to provide those algorithms. This paper focuses more on the mathematical part of the homomorphic encryption process and provides an analysis on how algorithms that are able to do computation on encrypted data were built.

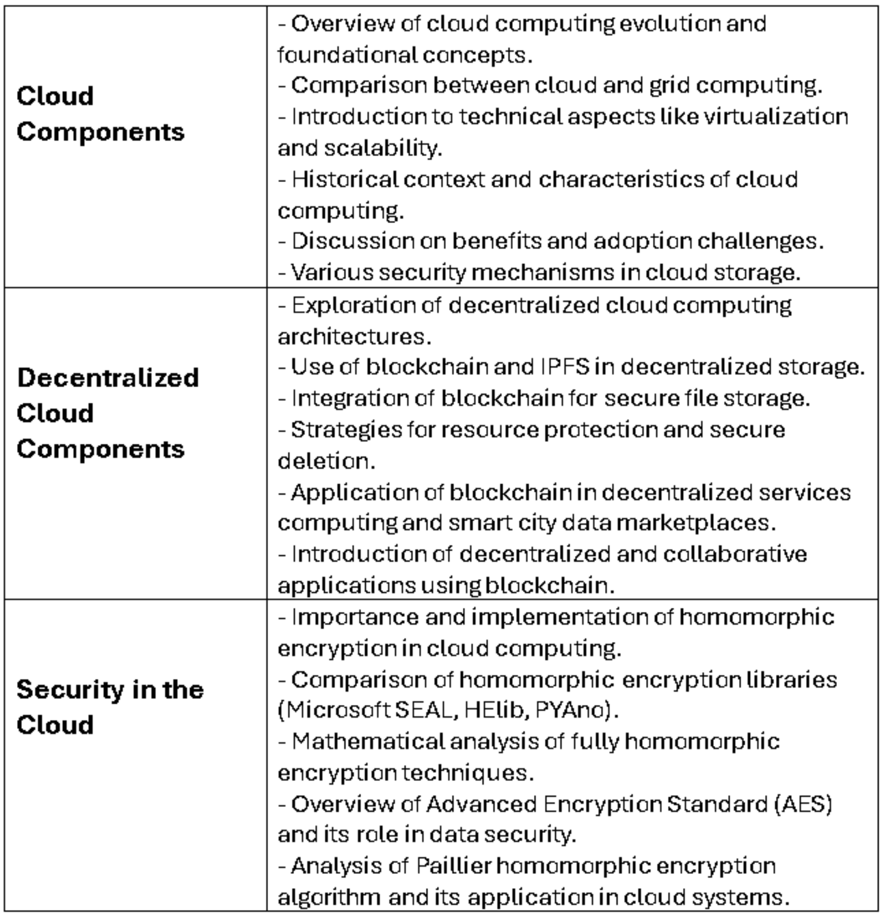
From the paper "Data Security in Cloud using Advanced Encryption Standard" [MBRNKK23] we get an analysis about the Advance Encryption Standard algorithm. Even though it is not an homomorphic one and the data that is encrypted has to be decrypted before the computations can be done, Advance Encryption Standard algorithm which is analyzed in this paper used to be a standard on encryption. It can be taken out from the paper that the Advance Encryption Standard is a symmetric algorithm used for encryption and that is still widely used across the globe to secure data. What makes this algorithm to be symmetric is the fact that the same key that is used for encryption is used for decryption also. Another aspects that can be driven out from the paper are the fact that the key size can be of 128, 192, 256 bits and the fact that the algorithm uses a series of linked operations including substitution and permutation in order to make the encryption.

From the conference article written by Nelly Fazio [FGJS17] we get some information about Paillier homomorphic encryption algorithm and about how it is used in the cloud computing systems. It can be taken out from the paper that Paillier homomorphic encryption algorithm is a public-key cryptosystem that supports only the addition operations on encrypted data. It was invented in 1999 by Pascal Paillier and is design to work with to large prime numbers as key parameters. From the paper it can be also driven out the fact that the security is based on the hardness of the composite residuosity problem which is computationally very difficult to solve. And also the use of random values in encryption which make sure that the same plaintext encrypts to different ciphertexts each time.

These information provided from the articles cited before gives a clear description on how cloud operates and is created even if it is decentralized or not and also how it ensures safety for the users of it.

2.2.4 Summary of Related Work

The following table presents of summary of the related work presented at related work. This chapter is created in order to be able to better view the work that has been already done.



Cloud Components	- Overview of cloud computing evolution and foundational concepts. - Comparison between cloud and grid computing. - Introduction to technical aspects like virtualization and scalability. - Historical context and characteristics of cloud computing. - Discussion on benefits and adoption challenges. - Various security mechanisms in cloud storage.
Decentralized Cloud Components	- Exploration of decentralized cloud computing architectures. - Use of blockchain and IPFS in decentralized storage. - Integration of blockchain for secure file storage. - Strategies for resource protection and secure deletion. - Application of blockchain in decentralized services computing and smart city data marketplaces. - Introduction of decentralized and collaborative applications using blockchain.
Security in the Cloud	- Importance and implementation of homomorphic encryption in cloud computing. - Comparison of homomorphic encryption libraries (Microsoft SEAL, HElib, PYAno). - Mathematical analysis of fully homomorphic encryption techniques. - Overview of Advanced Encryption Standard (AES) and its role in data security. - Analysis of Paillier homomorphic encryption algorithm and its application in cloud systems.

Figure 2.1: Picture of a table in order to summarise related work

2.3 Current implementations

As mention in the previously one of the solutions to the security problem is encryption. The data that is added onto the cloud has to be encrypted and for this there are a lot of methods and solutions that were provided during the years.

2.3.1 Advanced Encryption Standard

AES is a symmetric encryption algorithm designed by the U.S. National Institute of Standards and Technology (NIST) to replace the aging Data Encryption Standard (DES) in 2001. It was chosen after a competitive process.

Main Features of Adavanced Encryption Standard
Algorithmic Structure AES was designed by cryptographers Joan Daemen and Vincent Rijmen. A block cipher technique was used to encrypt data in fixed-size blocks of 128 bits, the mean data. Key Sizes: It is designed to work with three different key sizes, also considered the most popular. They are 128-bit, 192-bit, and 256-bit. Key size affects encryption strength; bigger sizes are more secure cryptographically but take more computation.

Number of Encryption Rounds: The number of encryption rounds is based on key size:

10-rounds on a 128-bits 12 rounds for 192-bit keys 14 rounds for 256-bit keys Each round consists of several operations, such as substitution, permutation, mixing, and key addition.

Encryption Process: The process of encryption consists of: SubBytes: The S-box is a substitution operation of non-linear space mapping, where each byte is replaced by another according to a look-up table. In this step, Transpose a shift, the rows of the state of the cyclic function.

MixColumns: An operation that mixes, applied per column within the state matrix. AddRoundKey: In this operation, each state byte gets combined with a subkey derived from the original encryption key. This is reversed on the decryption process.

Uses of Advanced Encryption Standard
Advanced Encryption Standard is applied in so many ways due to its robustness and efficiency. From those many places were AES is applied we can mention the following:

SSL and TLS protocols which help in securing communication, also maintaining internet security are put in functionality with the use of AES.

AES is also used in File Encryption, software like BitLocker and 7-Zip are implemented using AES in order to ensure security for the users.

Wireless Security: One of the great strengths of AES is in protecting the security of wireless networks under the WPA2 and WPA3 protocols. Government and Military: AES is accepted for protecting classified information at the top-secret level within the United States. Benefits of AES Security: The large key size and complex encryption process in AES ensure safety from brute-force attacks. Efficiency: AES is efficient in both software and hardware implementations, making it applicable to a wide range of devices—from low-power microcontrollers to high-performance servers. Standardization: AES, therefore, became an internationally recognized standard and is in use and supported on a wide range of platforms and different industries. Conclusion AES became a milestone in modern encryption because it combines security with efficiency and flexibility. Its practical applications in many areas make the AES algorithm a surety for protecting data from leakage into unauthorized hands and cybercrime. Moreover, the constant public scrutiny and stringent probing of the AES keep it the most reliable standard in encryption.

2.3.2 Microsoft Seal

Microsoft SEAL is an open source homomorphic encryption library developed by Microsoft Research. The library expands to Simple Encrypted Arithmetic Library.

By using this library to implement homomorphic encryption the computation over ciphertext without requiring decryption at any point is possible, this is a massive important thing in ensuring data privacy in cloud computing.

Now some of the features of the Microsoft SEAL library are presented such that it is easier to understand how it works.

Key Features that the library offers:

The Microsoft SEAL library supports both symmetric and asymmetric encryption schemes to enable various types of security levels according to the use case at hand, and this while maximizing versatility and functionality.

The library aims at practical performance, which is optimized for both speed and memory usage. Making it a powerful tool if a user wants to use it in order to build a cloud platform

Microsoft SEAL library is flexible and can easily be put into place with the cloud environment. Therefore, it supports different applications and platforms.

When talking about security the library uses the latest, most innovative cryptographic techniques.

The library is an open-source platform, meaning community contributions and peer reviews. It is, hence, continuously advanced and very reliable. This is also a powerful feature of this library because a new user can ask the community in order to get extra information.

Microsoft SEAL can be used for secure cloud computing, given the capacity to perform secure computations on data stored in cloud centers. Sensitive information needs to be fully protected for meaningful analysis. Hence, it enables privacy-preserving machine learning; computation over encrypted data allows training ML models with sensitive data without exposure. Financial services may enable computations with privacy preservation in encrypted data while preserving proper accuracies.

Now some examples in which the Microsoft SEAL library is frequently used such that it can be better understand where this library has an utility in real world applications.

In healthcare, for example, a use case for Microsoft SEAL could be where medical data needs to be treated confidentially in cloud settings. On the other hand, such a technology can be used in the encryption and processing of calculations on medical records without the disclosure of information that may jeopardize patients' privacy.

Another use case for Microsoft SEAL can be in fraud detection which is a must for the financial services industry, even if it often means institutions have to share sensitive data with each other. This traditionally created extensive security vulnerabilities, but with Microsoft SEAL, institutions can now collaborate without leaking sensitive information.

To sum up, Microsoft SEAL is an influential tool for implementing homomorphic encryption in cloud computing and thus presents a solid solution to ensure data security and processing in a privacy-preserving manner.

2.3.3 HElib

HElib is a free software homomorphic encryption open-source software library. Homomorphic encryption is a form of encryption that allows computations to be performed on ciphertexts while generating a ciphertext result and, upon decryption, matches the result of plaintext operation over the corresponding plaintexts. It was developed at IBM and is widely regarded as one of the most efficient and practical implementations of the Brakerski-Gentry-Vaikuntanathan (BGV) scheme.

Now some of the key features of the library will be presented such that it can be shown how strong this library is. **Key Features that the library offers:**

HElib revolves around the BGV scheme, which can perform operations like addition and multiplication on encrypted data. It lets the users of the library efficiently perform homomorphic evaluation of polynomial functions on encrypted data.

Arithmetic operations on encrypted data in HElib are done by representing the operations as polynomials. The current approach uses the Number Theoretic Transform for efficient polynomial multiplication.

HElib library does support Single Instruction, Multiple Data operations by allowing many data values to be encrypted into a single ciphertext. This dramatically boosts the throughput of homomorphic computations.

HElib is a library that offers an easy selection of parameters that result in flexible polynomial selection and the size of the ciphertext modulus and the plaintext space so that a user can obtain an appropriate balance between security and performance.

The bootstrapping process changes the fresh ciphertext for a reduced noise ciphertext, making it possible to perform an unlimited number of homomorphic operations.

HElib is also used in real world applications. The usage in real world applications is also presented such that it can be seen how the library pays a role in the industry of cloud computing.

HElib is mainly a research and development tool by which capabilities and performance in homomorphic encryption must be discovered. It finds applications in secure data analysis, privacy-preserving machine learning, and secure voting systems. There is a way, therefore, for the desired computations to be carried out while the data remains encrypted, supporting applications to use outsourced computation over possibly sensitive data.

Thus, HElib is a very strong implementation tool for the homomorphic encryption method under the BGV scheme. It is endowed with essential features that make actual applications involving encrypted computation accurate and has real-world applications in so many areas demanding secure processing of data. But these requirements and complexities in its implementation involve responsibility on the part of the users.

2.4 Proposed Approach

As mentioned before there are a lot of ways in which encryption can be done and libraries such as HElib or Microsoft Seal are very well documented and there are also a lot of studies about them.
Even though these two libraries are a standard in the industry due to the fact that they are used but most cloud applications that use homomorphic encryption in order to be able to compute encrypted data they are pretty hard to implement. For a new person into the world of cloud even to set up these two libraries can be a little bit complicated, and the usage of them is also quite hard.

Due to this aspects, this study presents a easier to use library which is LightPHE, a library that implements homomorphic encryption in Python fully and it can be easily used by just instaling the packages using the pip command in the command prompt.

This is the biggest advantage of this library, the fact that is easy to set up and then it can be used directly by calling the algorithms implemented in it. Even though it sounds like a super library to use it has some drawbacks too, because due to the fact that is built fully in python the computation times for homomorphic algorithms is bigger.

To conclude, what I bring new is a study made on a library that even though it is not used in the industry as much as the others are it has the potential to become very useful for people who want to build a cloud platform from zero but they are beginners into the technical part of things. LightPHE library is a great introduction into the world of homomorphic encryption and the comparison with the Advanced Encryption Standard algorithm which is done in this study is just to get the reader an ideea about the limitations of the LightPHE library when talking about the speed in which the operations are done.

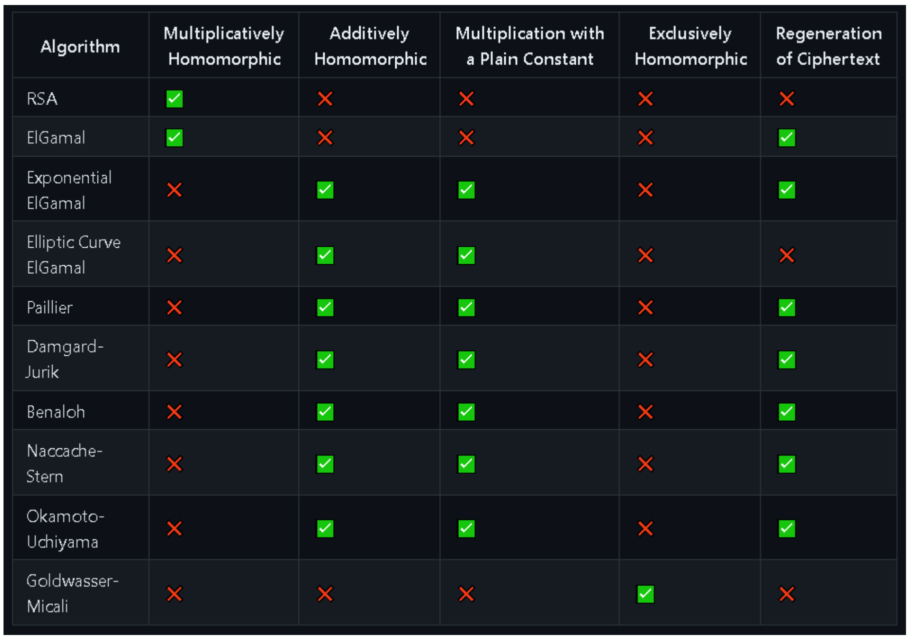
Chapter 3

Methodology

This chapter presents details about how the experiment was done. It includes details about some theoretical concepts regarding the LightPHE library and about the algorithms used from it. This part also includes information about the data set and how data that is tested is generated and also how the experiment is designed.

3.1 Theoretical Concepts

The experiment consist of implementing the LightPHE library and then using algorithms from the library. Here details about the library and also about each algorithm implemented from the library will be discussed. LightPHE is a lightweight partially homomorphic encryption library for Python programming language. The library was implemented by Sefik Ilkin Serengil and contains a lot of homomorphic schemes. Some of the most important schemes included in the library can be seen in this picture:



Algorithm	Multiplicatively Homomorphic	Additively Homomorphic	Multiplication with a Plain Constant	Exclusively Homomorphic	Regeneration of Ciphertext
RSA	✓	✗	✗	✗	✗
ElGamal	✓	✗	✗	✗	✓
Exponential ElGamal	✗	✓	✓	✗	✓
Elliptic Curve ElGamal	✗	✓	✓	✗	✗
Paillier	✗	✓	✓	✗	✓
Damgard-Jurik	✗	✓	✓	✗	✓
Benaloh	✗	✓	✓	✗	✓
Naccache-Stern	✗	✓	✓	✗	✓
Okamoto-Uchiyama	✗	✓	✓	✗	✓
Goldwasser-Micali	✗	✗	✗	✓	✗

Figure 3.1: Picture taken from GitHub where the library is documented. Link: `https://github.com/serengil/LightPHE?tab=readme-ov-file`

In the picture we can see that there are a lot of algorithms implemented in the library. To call each of them we need to first initialize the cryptosystem as it is called. This is done by using the LighPHE function, for example the cryptosytem is initialized with Paillier algorithm. "cs" will be the variable in python that will represent this system, in order to initilize it we use the following syntax:

$$cs = LightPHE(\text{algorithm\_name} = \text{"Paillier"})$$

So, the initialization of the algorithms is pretty straight forward and easy to understand. For the experiment two algorithms from this library were used: Paillier and Damgård-Jurik algorithms. In the following they will be presented as stand alone homomorphic algorithms.

The Paillier cryptosystem is an algorithm developed by Pascal Paillier in 1999 and is a basic algorithm in present-day cryptography. It shows some distinct properties of probabilistic and asymmetric in generating public keys. Unlike many cryptographic systems, Paillier does not rely on the difficulty of factorization for its security but rather on the computational difficulty of the Decisional Composite Residuosity (DCR) assumption.

Key Features of Paillier Cryptosystem:

Probabilistic Encryption: Encryption will contain a random component and, therefore, be nondeterministic—even with multiple encryptions of the same message.

Asymmetric Encryption: Paillier employs asymmetric key pairs, public and private keys, like other popular cryptographic systems such as RSA.

Homomorphic properties: The most important feature is the additive homomorphism of the scheme. That means the operations performed on plaintext will have a corresponding operation on their ciphertexts, namely addition. In other words, computations can be run on the data while it is still in an encrypted form, not precisely decrypted, and the privacy of this data is still preserved.

Paillier cryptosystem security is derived from the hardness of the decisional composite residuosity (DCR) assumption, stating that it is computationally infeasible to distinguish between quadratic residues and non-residues modulo n^2 for random cipher-texts under the condition of public key critical data, which has a public n.

Some applications where the Paillier encryption scheme is put into use include cryptographic protocols for secure multi-party computation, data mining while preserving privacy, and electronic voting and secure auctions. Named after Pascal Paillier, the Paillier cryptosystem proves to be one of the most essential modern cryptographies in terms of properties such as probabilistic encryption, asymmetric key generation, and additive homomorphism.

The Damgård-Jurik algorithm is a cryptographic algorithm used mainly for computing discrete logarithms in finite fields. The highlights for this article are as follows:

Background: Named after its inventors, Ivan Damgård and Marek Jurik. It has derived the solution for computationally discrete logarithms regarding a critical issue in various cryptographic protocols and algorithms.

Finite Fields: It is done in finite fields, which are mathematical structures where arithmetic operations are essentially carried out modulo a prime number, p, or power of a prime, p^k.

Purpose: In simpler terms, the Damgård-Jurik algorithm is aimed at giving an efficient computation for discrete logarithms—efficiency in such a way as to take special care in cases where this is inefficient through classical algorithms like the Baby-step Giant-step method or Pollard's rho algorithm due to issues of memory or time complexity.

Efficiency: It achieves efficiency through ideas that are a blend of the methods from index calculus and techniques for reducing the size of the problem space, which make it feasible in cases where other algorithms for computing discrete logarithms would be practically infeasible.

Application: Discrete logarithms-based cryptographic protocols and algorithms, in addition to some versions of Diffie-Hellman key exchange and elliptic curve cryptography, apply the Damgård-Jurik algorithm. Security: The algorithm's security is based on the difficulty of computing discrete logarithms in finite fields, which is generally believed to be a complex problem when the size of the field and other parameters are chosen appropriately.

Realization of the Damgård-Jurik algorithm provides fair detail about the finite-field arithmetic operations and setting up the parameters for a particular problem instance, which includes the modulus, p or p^k, and the generator of the multiplicative group. In short, the Damgård-Jurik algorithm is a considerable working tool of modern cryptography that enables practical computation of discrete logarithms in finite fields to maintain the security and functionality of various cryptographic protocols and systems.

Another library that is used in order for the experiment to be able to implement Advanced Encryption Standard is cryptography library from python.

The cryptography library in Python is a package that provides cryptographic tools and algorithms throughout. The package presents high-level recipes and low-level interfaces for cryptographic primitives, such as encryption and decryption schemes, digital signature generation and verification, and hashing. It also supports cryptographic key management.

Some essential features of the cryptography package include the support for symmetric key encryption (AES) and asymmetric key encryption (RSA). This supports secure data handling by the use of robust cryptographic algorithms. Digital Signatures It shall enable provisions for making and verifying digital signatures based on such algorithms as RSA and DSA to keep the data integrity and to ensure it.

Key Management: It supports securely generating, storing, and managing cryptographic keys. Essential Derivation Functions (KDFs) and Key Exchange Protocols are supported.

Hashing: It is how this library provides for an implementation of secure hash functions, including SHA-256 and SHA-512, intended for generating fixed-size hash values from arbitrary data.

Utilities: Some provide safe random number generation, encode/decode functions, multiple encoding formats, and management of cryptographic contexts.

That is, the cryptography library will be well designed to be very strong, user friendly, and well-matched for working with during the development of applications performing cryptographic operations securely. Updates in such pertinent information keep getting published continuously to update compatibility with current security policies and best practices

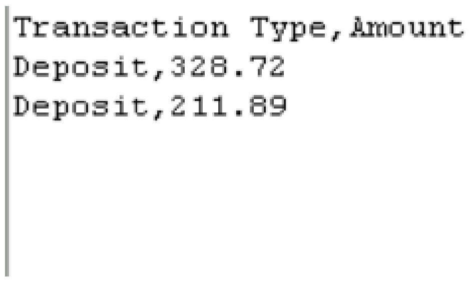
In conclusion these two algorithms are the ones on which the experiment with the data set was done. Both being homomorphic and somehow related due to the fact that Damgård-Jurik is an extension of the Paillier it can be seen how homomorphic encryption works on two different algorithms taken from the LightPHE library and also the Adavance Encryption Standard algorithm taken from cryptography library.

3.2 The data set on which the tests have been done

Moving to the data set, the data set used for the experiment is stored into a csv file which is build on the user decision. Basically the user decides how many transactions wants to generate before the algorithms are run. Clicking the button "Generate Transactions" will generate the number of transactions specified by the user.

After the transactions are generated in order to be used by the algorithms they are taken out from the csv file. The encryption is performed on them and then the computation with the already encrypted data. The result is then displayed on the GUI of the application where the user can read the output.

The number of transactions that can be generated by the user mimic a bank account deposit transactions as can be seen in this picture:



```
Transaction Type,Amount
Deposit,328.72
Deposit,211.89
```

Figure 3.2: Picture taken from the GUI of the experiment

The number of transactions is not limited and the user can generate as many as he wants, in order to see that the time to run the algorithm increase as the number of transaction increases. The experiment from this paper has been run on 100, 500, 1000, 2000 and 5000 such transactions such that the experiment will be able to provide a clear view for a person who thinks about using the LightPHE library in order to build a cloud platform.

The idea behind the dataset on which the algorithms are ran is to mimic a bank account. A bank that provides an application for their users in a cloud system and wants to let the user to see how much money he deposited over an year or over the whole time.

This type of dataset is very useful for testing the performance of homomorphic encryption due to the fact that computation on encrypted data has to be done and most of the homomorphic encryption algorithms are built to work with addition computation.

In conclusion, the homomorphic encryption algorithms presented in this paper that are type of the experiment and the Advanced encryption Standard algorithm are ran on this data that is also generate in the experiment randomly at each start of the experiment and deleted when the GUI of the experiment is closed.

3.3 Experiment design

This part of the chapter is designated to the experiment design. The idea behind the experiment is simple. The user can decided on the size of the data set and then generate the data set which will always be generated randomly and will be deleted after the experiment is closed.

The workflow of the experiment can be explained as follows.

Algorithm 1 Experiment Workflow

```
1: User specifies the size of the dataset by introducing the number of transactions.
2: Generate a random dataset which will be stored temporarily.
3: Provide a button "View CSV" to display the generated transactions for user verification.
4: User selects one of the following computation methods:
     • Compute Total Amount (Traditional): Direct sum computation from the CSV file.
     • Compute Total Amount (Homomorphic-Paillier): Total sum computation using Paillier encryption.
     • Compute Total Amount (Homomorphic-Damgard-Jurik): Total sum computation using Damgård-Jurik encryption.
     • Compute Total Amount (AES): Total sum computation using Advanced Encryption Standard (AES).
5: Perform the selected computation and display results on the GUI:
     • Total sum result.
     • Time taken by the algorithm.
6: A button "Show Computation Time Graph" to display a diagram showing the time taken by each algorithm.
7: Delete the generated dataset upon experiment closure.
```

The user can also see the generated csv file by pressing a button called ”View CSV”, when pressed the transactions will be display and can be checked by the user.

Then there are four buttons, the first one ”Compute total Amount(Traditional)” will display the total sum computed directly from the csv file, this was added to the experiment as an extra check such that the user is able to check the results. Another button is ”Compute Total Amount(Homomorphic-Paillier)” this button will do the total sum with encrypted data using Paillier algorithm. ”Compute Total Amount(Homomorphic-Damgard-Jurik)” is similar with the previous button mentioned but this one will use Damgård-Jurik algorithm. ”Compute Total Amount(AES)” will use Advanced Encryption Standard algorithm in order to compute the total amount.

The results will be then printed on the GUI such that the user is able to see the time taken for each algorithm. If the user wants more details the ”Show Computation Time Graph” button can be clicked and a diagram with each algorithm pops up and a clear view for the time taken is presented.

To conclude, the idea behind the experiment is simple, the data is taken from the file then all algorithms can be run but not in parallel such that the time will be the best in each case. The results are then printed on the GUI and a diagram with the results can also be generated to get a better view of each algorithm time needed to perform the computation.

Chapter 4

Experimental Setup

This chapter presents technical aspects of the experiment and the results of it. In this chapter details about the hardware on which the tests have bun run will be presented and also details about software used, algorithms and libraries will be presented here. The results of the tests that have been run on a different size of the data set will be also presented into this chapter.

4.1 Hardware and software used

The hardware used for runing the experiment is a Acer Nitro AN515-58 notebook with the following specifications:
Processor: 12th Gen Intel(R) Core(TM) i5-12450H 2500 Mhz, 8 Core(s), 12 Logical Processor(s)
Installed RAM: 16.0 GB(15.7 GB usable)
Graphics Processing Unit: NVIDIA GeForce RTX 3050 Laptop GPU
Storage: SSD with 512 GB capacity not partioned

Moving to the software specification on which the experiment has been run. The operating system is a Microsoft Windows 10 Pro with the version: 10.0.19045 on 64 bits.

Other software details include the fact that the test has been written and run using Visual Studio Code version 1.89.1, the user setup one, using python version 3.11.9.

Other important aspects regarding the software are the versions of the packages installed using pip version 24.0.
LightPHE package version 0.0.6
cryptography package version 42.0.7

This are the important details about the hardware and software on which the experiment took place. With these specifications the algorithms have been run on all the different sizes of the data set.

4.2 Algorithms and libraries used

This chapter presents details about each algorithm and libraries used in order to create the experiment. First library used is ”tkinter”. The library servers as a Graphical User Interface standard and provides an easy way to create simple and functional interfaces. It is part of standard Python distribution and does not need to be installed separately. From this library for the experiment creation the following widgets have been used: filedialog to make the user able to display the csv file, scroledtext also for making the user able to see the same file.

Next library used in order to create the experiment is ”csv”. This library comes in help when build the data set. It makes part of the standard Python module and does not need to be installed separately. Without this library the creation of the csv file which stores the data set would be impossible.

Another library that is used is the ”random” library which is also a standard Python library that does not require extra installation. This library is used to generate the random transactions from the csv file and also to generate the random key for the Advanced Encryption Standard Algorithm.

”Time” is also a library used in the creation of the experiment. This library comes in the standard Python module so it does not need to be installed separately. This library is used during the experiment to be able to calculate the time for each algorithm to run.

The ”os” library, another Python standard library is used in order to be able to use the ”urandom” function that generates the 256-bit key which is the Advanced Encryption Standard key and also to generate a random 128-bit key which is used in the process of encryption when implementing the Advanced Encryption Standard algorithm.

Another library is the ”base64” this plays a crucial role in the Advanced Encryption Standard algorithm encryption and decryption processes by ensuring that the data that is being encrypted can be safely encoded into a printable string format and subsequently decoded back into binary for decryption.

The ”cryptography” package as it was mentioned before is used to be able to implement the Advanced Encryption Standard algorithm. This is an extra library that was installed using pip command in the command prompt and is a package that includes many different algorithms related to cryptography. The ones used in this experiment are the ones related to the Advanced Encryption Standard.

The ”LightPHE” package which was also mentioned before is a partially homomorphic encryption library that is fully implemented in python and is used here in order to be able to implement Paillier algorithm and Damgård-Jurik algorithm.

The last library that is imported in the experiment is graph display which is not actually a library is a python file in which the code for displaying the graphs of the time taken for each algorithm to run is written. This file is imported here to be able to have a cleaner code in the main python file.

In the graph display python file the library ”matplotlib” is also used. This library is used in order to be able to draw the graph with the algorithms.

The algorithm used in order to be able to build the experiment are Paillier algorithm Damgård-Jurik algorithm and Advanced Encryption Standard Algorithm.

Paillier algorithm is a homomorphic encryption algorithm which is taken from the LighPHE library and is able to perform the addition computation. The time taken for this addition computation is the one that will be tested in this experiment.

Damgård-Jurik algorithm is also a homomorphic encryption algorithm which was developed from Paillier algorithm and is also implemented in the LightPHE library. It is able to perform addition on encrypted data and the time taken to perform this addition will be tested in this experiment.

Advanced Encryption Standard is a standard encryption algorithm that is used in many domains, in this experiment it is implemented with the help of cryptography package and it will be use to encrypt the data set, then to decrypt it and compute the total amount and then to encrypt it again. Simulating what homomorphic encryption does. The time taken for all operations is what will be tested in this experiment.

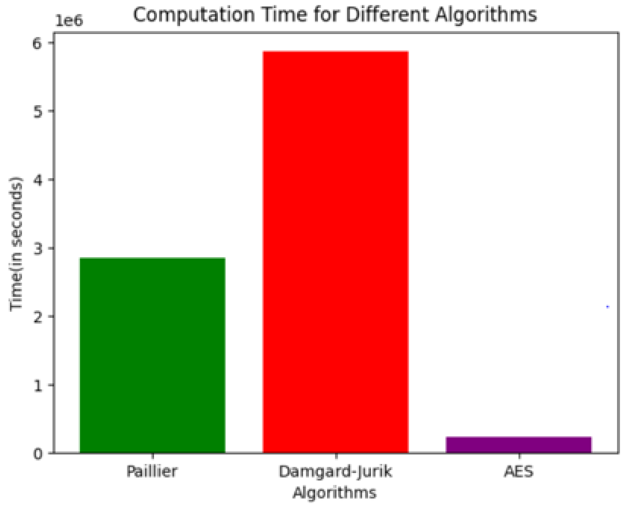
To conclude, these libraries together with the algorithms taken from them are building the experiment which is done in order to show the potential of the Light-PHE library compared with other homomorhpic encryption libraries that are already on the market.

4.3 Results

In this part the results of the tests done during this experiment are presented. The test were ran on 100, 500, 1000, 2000 and 5000 transactions generated randomly. For each size of the csv file I will explain the results obtained during the experiment. The results may be different on a different hardware or software and might differ on the hardware and software that was used in first place because the csv file transactions are generated randomly every time.

In this diagram 4.1, we can see the time taken for the two homomorphic encryption algorithms and also for the Advanced Encryption Standard Algorithm on a CSV file with 100 and 500 transactions.

100 Transactions diagram:



500 Transactions diagram

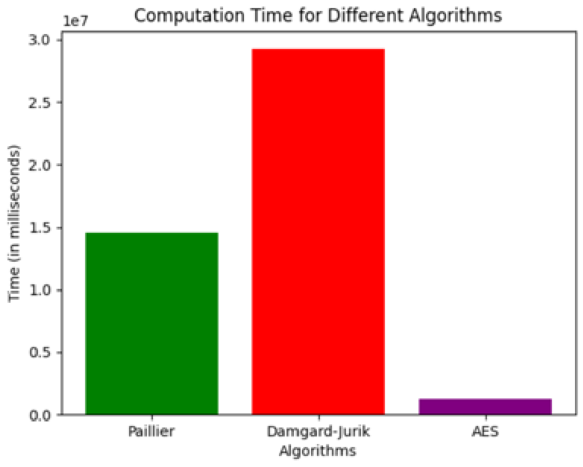
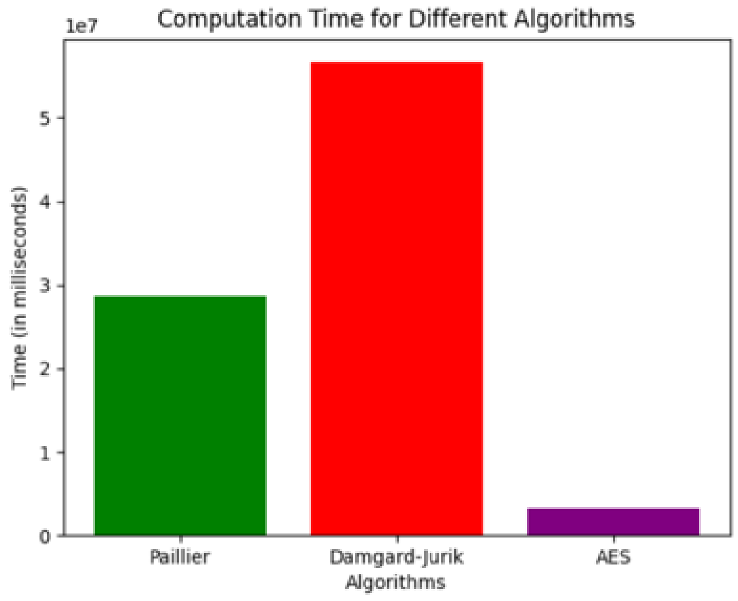


Figure 4.1: Picture taken from the GUI of the experiment

In this diagram 4.2, we can see the time taken for the two homomorphic encryption algorithms and also for the Advanced Encryption Standard Algorithm on a CSV file with 1000 and 2000 transactions.

1000 Transactions diagram:



2000 Transactions diagram:

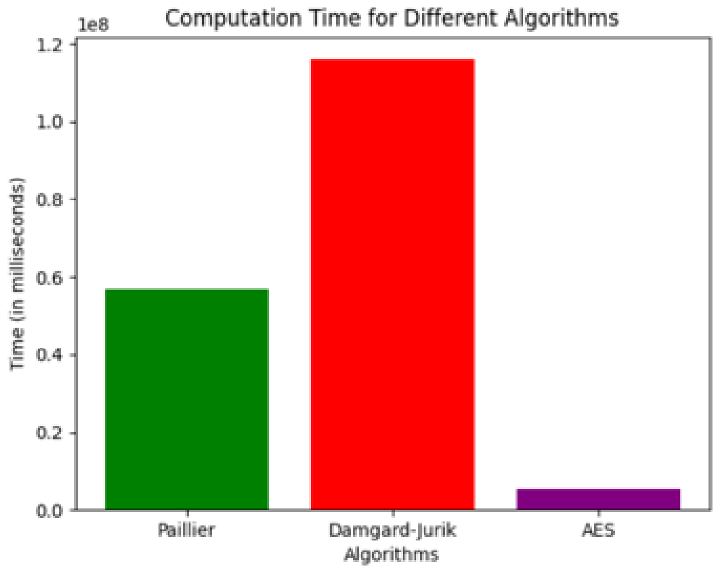


Figure 4.2: Picture taken from the GUI of the experiment

In this diagram 4.3, we can see the time taken for the two homomorphic encryption algorithms and also for the Advanced Encryption Standard Algorithm on a CSV file with 5000 transactions.

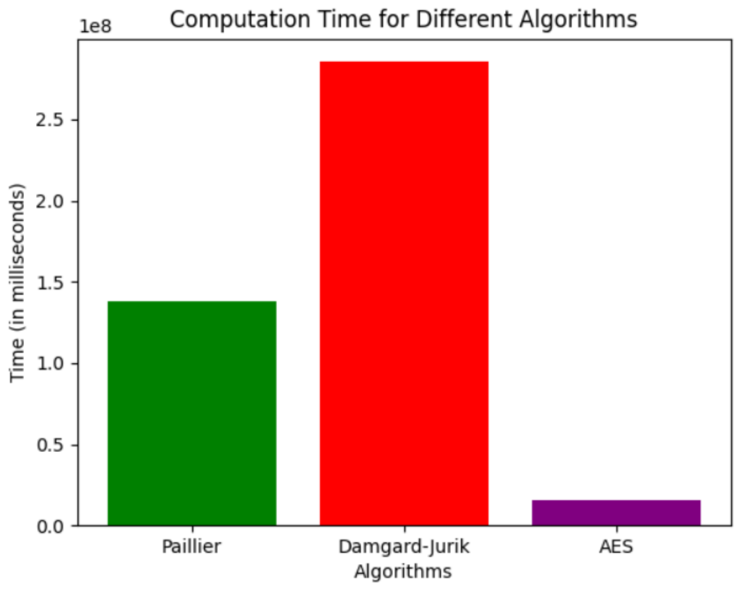
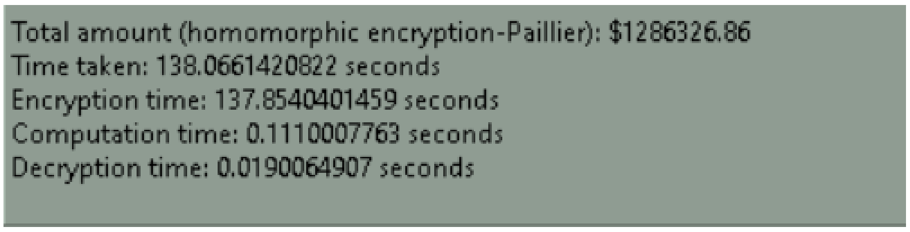


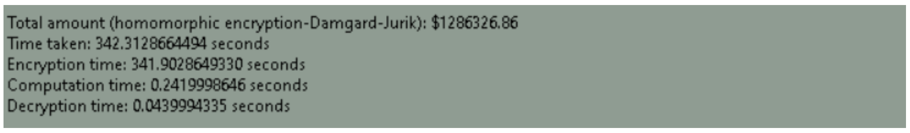
Figure 4.3: Picture taken from the GUI of the experiment

These tests show that the time taken by the homomorphic encryption algorithms is bigger than the time taken for Advanced Encryption Standard. The time that is presented on the diagram is the total time taking encryption into consideration which is bigger for the homomorphic encryption as we can see from these pictures taken for a 5000 transactions data set:



```
Total amount (homomorphic encryption-Paillier): $1286326.86
Time taken: 138.0661420822 seconds
Encryption time: 137.8540401459 seconds
Computation time: 0.1110007763 seconds
Decryption time: 0.0190064907 seconds
```

Figure 4.4: Picture taken from the GUI of the experiment



```
Total amount (homomorphic encryption-Damgard-Jurik): $1286326.86
Time taken: 342.3128664494 seconds
Encryption time: 341.9028649330 seconds
Computation time: 0.2419998646 seconds
Decryption time: 0.0439994335 seconds
```

Figure 4.5: Picture taken from the GUI of the experiment

From this pictures it can be taken out that actually the time needed for the computations is not that big. The difference between these two homomorphic encryption algorithms and Advanced Encryption Standard comes when talking about the encryption time. But if we take into consideration that data encrypted using homomorphic encyrption does not need to be decrypted if many computations are done with that data the times will be better using homomorphic encryption

4.3.1 Results Analysis

The experimental results presented in this chapter showed the performance of two homomorphic encryption algorithms Paillier and Damgård-Jurik compared to the Advanced Encryption Standard (AES) on random-size transaction datasets. The following are the observations and interpretations of these results:

Computation time for encrypted data includes adding transactions with a homomorphic encryption algorithm. This can be seen from the results in the graphs, and the computation time for encrypted calculations scales linearly with the number of transactions.

Encryption has privacy while performing operations on encrypted information without prior decryption. Even though AES outperforms for lower-level, basic activities in the process of encryption and decryption, homomorphic encryption finds its best application at times when operations are to be executed in a manner that does not decrypt the information until completely processed such as secure data analytics and privacy-preserving machine learning.

The hardware specifications that come as a consequence of the CPU, RAM, and GPU influence encryption algorithm performance.

In this work, a test was conducted using an Acer Nitro laptop with an Intel Core i5 processor and an NVIDIA GeForce RTX 3050 GPU. The software dependencies are Python versions, libraries such as LightPHE, cryptography, and development environments like Visual Studio Code, all of which go hand in hand to give the experiment a good outcome with functionality.

Chapter 5

Conclusion and Future Work

Conclusion of the experimental work in the chapter and further discussion of the achievable benefits that can be reached through the use of the LightPHE library by users who want to implement a cloud platform with homomorphic encryption. The section also provides a sketch of a future work effort that should extend the experiments further out and become more valuable.

From this experiment, the deductions show that LightPHE was not as fast in performance compared to most of its market competitors. Still, it has an edge due to its implementation simplicity and ease of installation. It presents such a good platform for algorithm integration, plus the library is so user-friendly that it makes it a beautiful choice, especially for beginners in the development of cloud platforms.

Some critical observations from the results are:. The main strong points of LightPHE are performance and ease of implementation. While it runs slower compared to its other competitors, its ease of installation and algorithm implementation make it accessible to a user who is new in the field of homomorphic encryption.

The security features of the platform on which homomorphic encryption should be one of the clouds are seriously taken into account because, from this experiment, the results are better. This allows computations over encrypted data without the need for a decryption scheme and, in turn, provides data confidentiality to be well preserved. While AES might look faster on fundamental encryption and decryption tasks, the properties of homomorphic encryption become even more efficient over many computations because it doesn't keep repeating the overhead of decrypting and re-encrypting.

The time for this computation using encrypted data operations increases linearly with the number of transactions. This implies that, even with a growing volume of data, the pattern in performance remained constant and within the manageable zone.

The efficiency of encryption algorithms is hardware-based and requires CPU, RAM, and GPU specifications. The experiment used an Acer Nitro laptop with an Intel Core i5 processor and an NVIDIA GeForce RTX 3050 GPU to give the LightPHE library a reasonable testing environment. These specifications had an impact on the performance results. Therefore, hardware considerations should be taken into account for future implementation.

Homomorphic encryption is beneficial in those application domains that demand secure data analytics and privacy-preserving machine learning. Practical support in computation on encrypted data without revealing sensitive information gives powerful advantages in these domains. The realized gain in maintaining privacy while performing computations without decryption provides testament to the potential of homomorphic encryption for real-world applications.

5.1 Future work

Future work that can be done in order to improve the experiment can be to test other algorithms from the LightPHE library. There are some algorithms that can be implemented for addition operation such that the user who want to use the platform can get even a better view of the library.

Other future work could be to test also the multiplication homomorphic algorithms from the library. There are some algorithms that are capable of doing multiplication. Maybe this can be tested on the same data set by converting the total amount into different types of money.

To conclude, there is future work that can be done in order to improve this experiment and make it more user friendly and more precise such that a user who desires to build a cloud platform for himself can get a better view on how strong homomorphic encryption implemented using LightPHE is.

Bibliography

[BDCdVF$^+$20] Enrico Bacis, Sabrina De Capitani di Vimercati, Sara Foresti, Stefano Paraboschi, Marco Rosa, and Pierangela Samarati. Securing resources in decentralized cloud storage. *IEEE Transactions on Information Forensics and Security*, 15:286–298, 2020.

[CGSM19] Pratibha Chaudhary, Ritu Gupta, Abhilasha Singh, and Pramathesh Majumder. Analysis and comparison of various fully homomorphic encryption techniques. In *2019 International Conference on Computing, Power and Communication Technologies (GUCON)*, pages 58–62, 2019.

[DPOB22] Trinh Viet Doan, Yiannis Psaras, Jörg Ott, and Vaibhav Bajpai. Towards decentralised cloud storage with ipfs: Opportunities, challenges, and future directions, 2022.

[FGJS17] Nelly Fazio, Rosario Gennaro, Tahereh Jafarikhah, and William E. Skeith. Homomorphic secret sharing from paillier encryption. In Tatsuaki Okamoto, Yong Yu, Man Ho Au, and Yannan Li, editors, *Provable Security*, pages 381–399, Cham, 2017. Springer International Publishing.

[GMM21] Bulbul Gupta, Pooja Mittal, and Tabish Mufti. A review on amazon web service (aws), microsoft azure & google cloud platform (gcp) services. EAI, 3 2021.

[HLGD20] Van-Hoan Hoang, Elyes Lehtihet, and Yacine Ghamri-Doudane. Privacy-preserving blockchain-based data sharing platform for decentralized storage systems. In *2020 IFIP Networking Conference (Networking)*, pages 280–288, June 2020.

[HWY$^+$18] Chanying Huang, Songjie Wei, Kedong Yan, Gongxuan Zhang, and Anmin Fu. A privacy-preserving attribute-based authentication scheme for cloud computing. In *2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 260–265, Nov 2018.

[JM12] Yashpalsinh Jadeja and Kirit Modi. Cloud computing - concepts, architecture and challenges. In *2012 International Conference on Computing, Electronics and Electrical Technologies (ICCEET)*, pages 877–880, March 2012.

[LSH20] Xuanzhe Liu, Sam Xun Sun, and Gang Huang. Decentralized services computing paradigm for blockchain-based data governance: Programmability, interoperability, and intelligence. *IEEE Transactions on Services Computing*, 13(2):343–355, March 2020.

[MBRNKK23] Shaik Mahaboob Basha, Vissa Rishik, V Jaya Naga Krishna, and S. Kavitha. Data security in cloud using advanced encryption standard. In *2023 International Conference on Inventive Computation Technologies (ICICT)*, pages 1108–1112, 2023.

[NHT21] Nazmun Nahar, Farah Hasin, and Kazi Abu Taher. Application of blockchain for the security of decentralized cloud computing. In *2021 International Conference on Information and Communication Technology for Sustainable Development (ICICT4SD)*, pages 336–340, Feb 2021.

[RRK18] Gowri Sankar Ramachandran, Rahul Radhakrishnan, and Bhaskar Krishnamachari. Towards a decentralized data marketplace for smart cities. In *2018 IEEE International Smart Cities Conference (ISC2)*, pages 1–8, Sep. 2018.

[SPA$^+$22] Pranshu Sood, Parikshit Palsania, Shivang Ahuja, Shivam Kumar, Kiran Khatter, and Atul Mishra. Decentralised collaborative docupad using blockchain. In *2022 IEEE Delhi Section Conference (DELCON)*, pages 1–8, Feb 2022.

[SR19] Jayachander Surbiryala and Chunming Rong. Cloud computing: History and overview. In *2019 IEEE Cloud Summit*, pages 1–7, 2019.

[SSC22] Sonali Sharma, Shilpi Sharma, and Tanupriya Choudhury. A study and analysis of decentralized cloud based platform. In *2022 12th International Conference on Cloud Computing, Data Science Engineering (Confluence)*, pages 124–128, Jan 2022.

[SSMT20] Meet Shah, Mohammedhasan Shaikh, Vishwajeet Mishra, and Grinal Tuscano. Decentralized cloud storage using blockchain. In *2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184)*, pages 384–389, June 2020.

[SWN22] Vasily Sidorov, Ethan Yi Fan Wei, and Wee Keong Ng. Comprehensive performance analysis of homomorphic cryptosystems for practical data processing, 2022.

[UDDG21] S. Uthayashangar, T. Dhanya, S. Dharshini, and R. Gayathri. Decentralized blockchain based system for secure data storage in cloud. In *2021 International Conference on System, Computation, Automation and Networking (ICSCAN)*, pages 1–5, July 2021.

[Wei11] Joe Weinman. The future of cloud computing. In *2011 IEEE Technology Time Machine Symposium on Technologies Beyond 2020*, pages 1–2, June 2011.

[WWR$^+$12] Cong Wang, Qian Wang, Kui Ren, Ning Cao, and Wenjing Lou. Toward secure and dependable storage services in cloud computing. *IEEE Transactions on Services Computing*, 5(2):220–232, April 2012.