

UNIVERSITY OF BAHRAIN

**College of Information
Technology**



**Department of
Computer Engineering**

Applying Homomorphic Encryption Algorithm for Securing Big Data Sharing in Open Banking

A Thesis Submitted in Partial Fulfilment of the Requirements for the
M.Sc.Degree in Cyber Security

Submitted by
Dana Abdulla AlRayes
20190200

Supervised by
Dr. Mazen Ali
Associate Professor

University of Bahrain

March 2024

ABSTRACT

Homomorphic encryption stands out from other encryption techniques by enabling arbitrary computations on encrypted data without revealing the original plaintext. Researchers have extensively studied various encryption methods, highlighting the importance and efficiency of homomorphic encryption in data security. Its deployment in the financial sector is crucial for ensuring the safe utilization of the vast financial transaction data in open banking.

While there have been studies on partial (PHE), somewhat homomorphic encryption (SHE), and fully homomorphic encryption (FHE), they haven't specifically investigated its application within open banking platforms utilizing big data sharing. This study tested a fully homomorphic encryption algorithm on an open banking dataset, using an open-source Linux database and the SEAL homomorphic encryption library.

The study explores the application of fully homomorphic encryption algorithms in securing financial data. The practical deployment demonstrates its effectiveness in ensuring data security, even in the event of a breach. Additionally, the study's results hold implications for the future development of open banking and big data sharing technologies, emphasizing the importance of advanced encryption methods for maintaining data privacy and integrity.

Contents

ABSTRACT	A
TABLE OF CONTENTS	B
LIST OF FIGURES	E
LIST OF TABLES	F
ACKNOWLEDGMENTS	G
DEDICATION	H
LIST OF ABBREVIATIONS	I
1 Introduction	1
1.1 Online Banking Overview	1
1.2 Adoption of Open Banking	5
1.2.1 Data on the Cloud	13
1.3 Open Banking Threats	18
1.4 Research Question	24
1.5 Research Significance	25
1.6 The Study's setting	28
1.7 Research Terminology	31
2 Literature Review	35
2.1 Overview	35
2.1.1 Homomorphic Encryption History	37
2.2 Online Banking and Security Related Work	39
2.3 Discussion	42

2.3.1	Homomorphic Scheme Solutions	43
2.4	Summary	47
3	Methodology	50
3.1	introduction to methodology	50
3.1.1	Overview of research methodology	50
3.1.2	Research Objective and goals	51
3.1.3	Scope and limitations of the study	52
3.1.3.1	Cryptographic methods for Secure Trans- actions	52
3.1.3.2	Access Controls and Customer Informa- tion Protection	53
3.1.3.3	Regulatory Compliance	53
3.1.3.4	Ethical consideration	54
3.2	Research design	55
3.2.1	Explanation of Homomorphic Encryption scheme . .	56
3.2.2	Justifications for selecting Homomorphic Encryption	57
3.2.3	Comparison of other encryption methods	58
3.3	Data collection	59
3.3.1	Types of open banking data	59
3.3.2	Source of Data	61
3.3.3	Considerations of data collection in an open bank- ing environment	61
3.4	Experimental setup	63
3.4.1	selection of environment	63
3.4.2	Tools and technologies used	64
3.4.3	Parameters for experimentation	65
3.5	Homomorphic encryption implementation	69
3.5.1	Description of Homomorphic Encryption scheme . .	72
3.5.2	Step-by-step implementation process	74

3.5.3	Challenges encountered during implementation . . .	75
4	Results and Findings	77
4.1	Security analysis	77
4.1.1	Evaluation of security risks in open banking	77
4.1.2	How Homomorphic encryption addresses security concerns	79
4.1.3	Analysis of potential threats and vulnerabilities . . .	81
4.2	performance evaluation	82
4.2.1	Metrics of security risks in open banking	82
4.2.2	Comparison of encryption overheads	83
4.2.3	Assessing computational efficiency and speed	83
4.3	Testing and validation	88
4.3.1	Testing scenarios and use cases	88
4.3.2	Validation of encrypted data operations	92
4.3.3	Results and findings	93
5	Conclusions and recommendations	94
5.1	Summary of methodology	94
5.2	key findings and conclusion	95
5.3	Recommendations and future work	96
	References	98
	Abstract (in Arabic)	

List of Figures

1.1	Open Banking APIs: How it works	3
1.2	cloud services categories explained	15
3.1	Installation of package from github	64
3.2	Seal on python implementation	64
3.3	Front page GUI	66
3.4	Login page	66
3.5	File upload page	67
3.6	User selecting file for encryption	67
3.7	Uploading the excel file	68
3.8	Encrypting data by using seal	69
3.9	SE Proposed Scheme	73
4.1	Key generaion time	87
4.2	Use case chart	89

List of Tables

3.1	Hardware Specifications	65
3.2	Software Specifications	65

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisor, Dr.Mazen Ali and Co-Supervisor, Dr. Wael Elmedany for their great support, help, and motivation. Their guidance helped me in all the time of my research and during the writing of this thesis.

DEDICATION

This thesis is affectionately dedicated to my parents, who taught me that the best sort of information is knowledge gained for its own sake, and that even the most difficult endeavor can be completed if approached in small steps. Additionally, I would like to express my gratitude to my associate professor, Dr. Mazen Ali, whose guidance and expertise significantly contributed to the success of this thesis. I am also deeply thankful to my friends and family for their unwavering support throughout this academic journey.

LIST OF ABBREVIATIONS

OB	Open Banking
HE	Homomorphic Encryption
PHE	Partially Homomorphic Encryption
FHE	Fully Homomorphic Encryption
API	Application Programming Interface
FTP	File Transfer Protocol
PSD	Payment Service Directive
PSD2	Payment Service Directive
PCI DSS	Payment Card Industry Data Security Standard
Bahrain OBF	Bahrain Open Banking Framework
VAN	Value Added Networks
OBIE	Open Banking Implementation Entity
CBB	Central Bank of Bahrain
PSD	Payment Services Directive
KYC	Know Your Customer
TPP	Third Party Providers
OBF	Open Banking Framework
MIT	Massachusetts Institute of Technology

ARPA	Advanced Research Projects Agency
DARPA	Department Advanced Research Projects Agency
AWS	Amazon Web Services
IaaS	Infrastructure as a Service
SaaS	Software as a Service
PaaS	Platform as a Service
DES	Data Encryption Standard
DES	Data Encryption Standard
RSA	Rivest-Shamir-Adleman
AML	Anti-Money Laundering
KYP	Know Your Partner
DDoS	Distributed Denial of Service
EFT	Electronic Fund Transfer
SE	System Engineering
ECC	Elliptic Curve cryptography
FDX	Financial Data Exchange
LWE	Learning With Errors
ABAC	Attribute-based access control
RBAC	role-based access control
GDPR	General Data Protection Regulation

Chapter 1

Introduction

1.1 Online Banking Overview

Modern banking is never an easy step to take towards traditional products because of the way consumers going to change to access their bank accounts and make transactions. By introducing "open banking," a greater range of third-party financial service providers will be able to access consumers financial data, including account balances, transaction history, and payment data. In this instance, APIs are leveraged to make this happen. Here is one reason why Fintech agreed to a partnership with banks to empower consumers to be in charge of their financial matters and other desired outcomes. New banking services, business models, and monetary opportunities can be developed by developers thanks to open banking and its accompanying suite of technologies, rules, and services. In order to succeed, open banking needs new client expectations and technology-centric policies. According to Del-Fresno (2019), the banking sector has dramatically evolved in different aspects in the last decade, and it has been tough to live without (Anderson). Open banking is transforming the modern banking and financial services landscape. Its market is steadily expanding. According to a survey by Allied Market Research, the market would expand by 24.4% by 2026, reaching approximately \$43 billion (Team, 2022).

The idea behind "open banking" is to increase levels of competition and innovation in the banking sector to better serve the needs of businesses and customers. Open banking is one of the latest banking revolutions. Financial institutions and banks have been adopting open banking in the last couple of years. However, despite the adoption of open banking, it has been linked to many issues and concerns from financial institutions and consumers (Jeng, 2022). In contrast, the emergence of these new competitors forces banks to reevaluate their value offerings in order to preserve their dominant market position. Moreover, open banking seeks to become an industry-wide driver of innovation by providing greater access to data and functionalities to market players and thereby enabling the development of new products and services that were previously out of reach for competitors like Fintechs, fully-digital banks, and Bigtechs (*What is a digital bank and how it will benefit you*, 2022).

Recent open banking initiatives in Europe and the United Kingdom are hastening the development of a cooperative atmosphere, and the subsequent use of application programming interfaces (APIs) to exchange data is enabling Fintechs, incumbents, and other third-party providers to significantly revolutionize the entire financial services sector. These two directives aim to create a unified market across Europe, open the door to new entrants, increase the level of competition and innovation, and ultimately benefit consumers by giving them access to better goods and services at more reasonable costs. Application Programming Interfaces (APIs) have been shown to be the most secure and well-tested technology for acquiring access to consumers' bank accounts, despite the fact that regulators do not specifically stipulate their use (Zachariadis & Ozcan, 2016). The manner that many companies and regular people deal with banks is changing as a result of new open application programming interfaces. Others are being modified to

accommodate the current regulatory climate, in which an increasing number of vendors and partners use open, standardized APIs rather than a variety of proprietary ones.



Figure 1.1: Open Banking APIs: How it works

Historically, the federal government has been the primary source of legislation designed to increase competition in network industries like telecommunications, utilities, and transportation. It correlated with sustained decreases in cost, increased output, and improved worker productivity (Partida, 2022).

For decades, businesses have strived to be customer-focused. There are several companies that, within their industry, mold their competitive advantages around this idea. The emergence of digitalization paved the way for better consumer comprehension via precise analysis of their online actions, allowing for the development of a specialized plan to meet their specific requirements. Expanded access to digital devices and channels is influencing a sea change in the banking industry, one that is always adapting to the needs of its consumers. Customers have constant access to their financial services, and they have higher expectations than ever before for the quality of service they receive. Companies with a big number of customers rely heavily on data analysis to learn about their customers' preferences, online activity, demographics, and other vital factors. Companies can use this data to better cater to each individual customer's needs. As a result of technological advancements, not only are company procedures evolving, but so,

too, are customer behaviors. As a result, services across many industries are adapting to provide better value to their customers. With its emphasis on personalization and its promotion of business model innovation through a collaborative environment, open banking can help the financial services industry achieve a higher level of individualization (Riley, 2021).

Banking APIs are spreading the paradigm at several levels, allowing third parties to have a more central position in the business. Therefore, intriguing new possibilities and shifts in user experience are made possible by the establishment of a collaborative ecosystem between banks and other participants, who are primarily represented by Financial Technology companies. Businesses can now gain insight into customers' spending habits and use that information to develop more personalized products and services. By analyzing debit card transactions, for instance, firms that take use of open banking frameworks can get a bird's-eye perspective of customers' spending across various industries, which in turn encourages the formation of new partnerships aimed at meeting these newly discovered requirements (Mohammed, 2019).

By removing barriers to entry for new entrants, open banking gives customers more leeway in selecting their preferred goods, services, and delivery methods. Open banking removed the prior scalability barriers that prevented other parties such as Financial Technology companies, digital banks, and other businesses from gaining access to the transaction information and bank accounts of customers. Now, more than ever, third parties can use this wealth of data to their advantage, producing more personalized experiences that boost retention, loyalty, and satisfaction among users.

Since the introduction of open banking, clients have been able to easily transfer their money and accounts to new providers, making these consider-

ations the primary focus of modern enterprises. These new rules intend to facilitate the creation of a greater variety of user-friendly, intuitive services by opening up data that was previously only accessible to banks to other parties. Therefore, open banking establishes new pillars for competing in the financial services industry and pushes businesses towards a higher emphasis on customization, transparency, integrity, and adaptability in their offerings. Customers, on the other hand, have access to more financial products and services than ever before, and they can get these products and services from a wider variety of sources than ever before, saving them time and energy (Marzouk, 2021).

With ease and uncomplicated ways of opening bank accounts, money transfers, etc. the risk hacking, phishing and unauthorized access, etc. becomes bigger, and need to mitigate them by enhancing security of open banking by protecting sensitive information even during processing with the application of homomorphic encryption.

1.2 Adoption of Open Banking

Open banking remained one of the results of partnerships of banks and financial technology institutions and was first introduced for the world in 2013 as a part of the European Commission's revised PSD2" European regulation for electronic payment services" proposal. An article by Wolters & Jacobs (2019), states that Payments Services Directive which known as PSD2 started in 2007 with the payment service providers (PSD) . The idea of contributing to the European market by developing single payment by promoting innovation has been used and adapted to energize the development of payment services integration. It guarantees a high level of security for payment transactions on the web against fraud. The PSD2 does not pro-

protect the consumer's data on the cloud as much as securing executed transactions. PSD2 has many everyday uses and acronyms such as ASPS, AISP, EBA, PISP, PSP, PSU, RTS, SCA, TPP and finally the most popular is API; which is an application Programming Interface that enables organizations to open up their data for third party developers, this allows the services and products to leverage their data and functionality to each other by a documented interface (Polasik, Huterska, Iftikhar, & Štěpán Mikula, 2020).

Over the past decade, new competitors with cutting-edge business models have emerged to compete with conventional banks: the financial technology industry. The connection between banks and customers, which is built on soft information, is being challenged by fintech's thanks to the ability to draw on hard information to boost the level of customization of their value package. As a result, FinTech's place a premium on serving their clients, whereas banks prioritize developing new products. However, open data sharing brings up concerns like data privacy, which banks have historically seen as a major obligation. Banks have developed reliable customer relationships, and their internal systems are able to handle massive amounts of sensitive data. With the introduction of open banking comes both new difficulties and new possibilities for the financial sector as a whole to leverage its greatest assets, including its enormous client base and the scalability of banks, in order to innovate and be as adaptable as smaller enterprises.

Traditional banking relied on economies of scale to offer both payment services and liquidity at the same time, as customers with immediate cash needs were more likely to make in-person deposits or use their debit cards. Most transactions today can be settled without physically exchanging currency, instead relying on electronic transfers of value between accounts that once functioned as bank accounts. However, with Fintech's foray into the

market, a plethora of new services have emerged to make it easier to move money across financial institutions. Traditional banks are facing competition from newer institutions that are using technology more effectively and are narrowing their attention to improving just one banking service. The relationship with banks has undergone a dramatic shift thanks to fintech startups, which have created a level playing field in which new realities may thrive without the backing of established players (Hornuf, Klus, Lohwasser, & Schwienbacher, n.d.). However, Fintech is not yet a perfect substitute for conventional banking. Contrarily, cooperation between these organizations is the most promising scenario.

Traditionally, financial institutions like banks have looked for partnerships inside their own industry; but, in the digital age, it is necessary to broaden these partnerships in order to build more effective and transparent synergies. Faster, more easily digestible, and more interactive services with or without the backbone of a "physical" financial network are the unavoidable future of the finance business, and all of this is done to raise and improve the customer experience. Open banking legislation aim to support and sustain this type of ecosystem, and in this scenario institutions must rationally and boldly foresee and facilitate this form of interaction to avoid becoming completely obsolete.

Regulations based on "open banking" have the same overarching goal of fostering and maintaining this environment. Since the implementation of the Revised Payment Service Directive (PSD2), Application Programming Interfaces (APIs) have assumed a position of utmost importance; experts regard them as the most efficient means of connecting banks and third parties. APIs are utilized in a variety of situations, but the emergence of open banking models prompted businesses to reassess APIs' potential and capitalize

on the value they can offer to their organizations, both for established banks and challenger players in the sector. The new open banking environment compels banks to abandon their rigid and reactive organizational structures in order to increase their profitability and significance in the new financial services value chain.

APIs enable banks to accomplish this objective by providing valuable services depending on context, position, and customer technology. The objective of open banking rules is to establish a collaborative ecosystem in which banks, Fintechs, Bigtechs, and other third-party providers may capitalize on their complementary capabilities by exchanging data intelligently. Therefore, APIs can be utilized by third parties within their organizations to provide the end user with a more tailored banking experience. When discussing open banking, it is also essential to understand the core characteristics and diverse commercial applications of application programming interfaces (APIs). API has been used widely in the last decade, and most popular web applications would not have existed without APIs (*CBB launches the Bahrain Open Banking Framework: CBB, n.d.*). In the kingdom of Bahrain, the adoption of open banking started in October 2020, when the Central Bank of Bahrain (CBB) launched the Bahrain open banking framework (Bahrain OBF), followed by the rules previously launched in December 2018 (cag, 2022).

Various empirical studies show that these constraints lead to better outcomes for customers, higher quality financial products and services, and more money spent on research and development. These successes, however, hinge on the financial sector's capacity to modify and grow its business models in order to provide new sources and propositions that make use of the legal environment.

Multiple studies show that there is a considerable risk to open banking busi-

ness models, but there are also promising opportunities.

While open banking refers to the practice of sharing financial information through cloud securely, with the consent from the data owner. there are some key security risks associated with open banking:

1. **Data Breaches:** when a malicious actor gains unauthorized access to open banking platform, which result in compromising sensitive financial and personal data.
2. **Phishing Attacks:** this type of attack may trick the users into revealing their login credentials or provide sensitive information.
3. **API vulnerabilities:** weaknesses in the security of APIs used for data sharing can be easily exploited by the attackers. its crucial to implement secure coding practices and regularity updates and patches APIs to address vulnerabilities.
4. **Third-Party Risks:** in open banking often requires third party service providers, if these providers has security risks, the entire ecosystem will be compromised.
5. **Consent Management:** misuse of consent risks comes when the consent mechanism is not robust.
6. **Insufficient Authentication:** weak authentication of the user account can lead to unauthorized access. strong measure like two factor authentication is crucial to protect user accounts and sensitive financial and personal information.

in the cases like phishing attacks, data breaches, insufficient authentication and APIs vulnerabilities. the solution of Homomorphic encryption can be a good solution to not bother when the data are breached,

or the unauthorized access has been encountered. while the data will be encrypted when in transit or at rest. the main benefit is to have your data encrypted all the time.

When it comes to the new reality, large commercial and investment banks are both the most vulnerable and the most favored. Inevitably, consumers will give their OK to the sharing of their financial information using industry-standard protocols and tools. With the advent of open banking and the use of open application programming interfaces (APIs), these procedures became more efficient for all new entrants in the sector, who may now operate at a pace formerly reserved for incumbents. On the other hand, open banking presents an opportunity for established financial institutions to develop novel revenue streams and propositions that aim to leverage the data of third parties to speed up processes that might otherwise take more time. Through the sharing of information across various parties and channels, the new open banking system paves the way for those with no credit history to apply for loans more quickly, for example. All of these are only a handful of the many potentials uses and partnerships made possible by open banking laws.

The financial services industry's future, and the business models of its players, appear to be in a state of constant flux. This analysis, will explain how open banking has affected both long-standing firms and up-and-coming newcomers to the market. In order to facilitate greater adaptability, simplification of development and design, and expansion of options and ease of innovation, APIs are utilized in the production of new services and products as well as the management of current ones. As a result, the exchange of digital content between parties can be facilitated in a more streamlined manner. Any business or organization can benefit from the construction of an API since it streamlines the process by which they can incorporate third-party technology into their offerings. As more and more people use an API, the

exposed service improves, which is good for the company's profile and the people who use the API. The procedure is fast, automatic, and effective, and it functions, at its core, similarly to how an electrical appliance uses electricity: To "plug in" and gain access to a company's services and/or data, an API requires interaction with other applications that follow the same pattern. In most cases, they connect two unrelated online resources so that one can give the user with access to the services of another.

APIs are commonly utilized in the real world to provide information aggregation services. Here we see the use of third-party application programming interfaces (APIs) to provide consumers with the ability to shop around for the best deal on a hotel room or plane ticket and then make the purchase with zero additional fees. Another common scenario is using a social network's "login with" feature to verify a user's identity before they can access the network's APIs. Companies can improve their service to customers by making better use of APIs, which in turn provides customers with more alternatives and a more engaging user experience. Firms operating within this ecosystem can benefit from increased innovation and reduced technology costs if a heterogeneous and adaptable infrastructure is developed across these systems. However, APIs can be differentiated according to their degree of openness, with some being intended solely for internal use and others being exposed to any third party, regardless of their relationship with the company that exposes them (public APIs). Companies utilize private APIs internally to facilitate communication across departments. The capacity to access data in real time has a positive impact on corporate efficiency and productivity (Zachariadis & Ozcan, 2017).

Private application programming interfaces are widely utilized in the banking and insurance industries to provide onboarding and Know Your Customer (KYC) processes that are both more secure and faster than their public counterparts. APIs may be made available as data or services only to

selected business associates in other situations. The offering of a product or service is one such example of the unusual circumstances under which this occurs.

Private APIs are functionally identical to public APIs; however, third-party API gateway solutions are typically used to restrict access. To be more specific, banks leverage APIs to expand their business offerings to include the sale of products from third parties such as insurance and credit cards. For instance, Starling Bank, a UK-based digital-only bank, lets customers use the Transfer Wise app to make domestic and international transfers in a variety of currencies. Providers were able to improve the quality of their service offerings thanks to the APIs' exposure to sensitive data including transfer and financial data between the parties. Public APIs, on the other hand, are accessible to anybody who accepts the API provider's terms and conditions. The data's potential uses can be governed by the publisher. Software and app developers rely on application programming interfaces (APIs), and their availability can pave the way for new digital platforms on which people can utilize existing APIs for their own purposes.

There are times when this service costs money, but it can be a good source of extra income for banks and other financial institutions. For instance, if a digital payment provider opens their application programming interface (API), e-commerce websites can integrate with their platform for seamless, efficient transactions. Since public APIs make it easier for TPPs to offer banks' services to their customers, they play a crucial role in the evolution from closed to open banking. The use of application programming interfaces (APIs) is making it easier for financial services to interact with one another and other services, which is greatly improving the user experience. Using a GPS and a bank's application programming interface (API) to pinpoint the location of the nearest ATM or branch is one way to better engage with and serve clients.

After the introduction of open banking, APIs gained critical importance in the growth and development of the financial services industry. For instance, the Revised Payment Service Directive (PSD2), like the other open banking directives, capitalizes on the decentralization and dissemination of information, services, and products between well-established and rising companies in accordance with predetermined criteria. As a result, the goal of open banking is to foster an ecosystem in which banks and large conglomerates can reap the benefits of their competitive advantage, larger scale of data, and customer trust while also providing third parties with the opportunity to gain access to customer data and contribute to the development of new and improved applications and services for consumers.

In the United Kingdom, where the policy has been in place for four years, more than 5 million people now use open banking. It took ten months to expand from one million to two million users in 2020, but only four months to go from two million to five million users. The OBIE has made these figures public. We anticipate that this trend will continue and spread globally as open banking technology becomes more prevalent and user-friendly in the near future (Kassab & Laplante, 2022).

1.2.1 Data on the Cloud

Banks may save a remarkable amount of time and money by embracing cloud computing solutions. When it comes to the widespread adoption of cloud computing, it is vital to consider data security and protection. Fully Homomorphic encryption is still the subject of ongoing research. One of its limitations is the time necessary to encrypt and decode sensitive data, as classic encryption systems have demonstrated a level of resistance and great maturity that can be rehabilitated or hybridized for use in the field of protecting sensitive data hosted in the Cloud.

In this century, cloud computing is widely regarded as one of the most

ground-breaking technological advances that has been made. Through the use of the Internet to provide data storage, server space, application and network services, and more. Providers of cloud computing services are the organizations that actually deliver them to clients.

Cloud services have many advantages, but security remains a major obstacle to their widespread use. Information stored in the cloud is always accessible to the cloud service provider. Data might be added, removed, or altered, and it could be distributed to third parties. Customers using the Cloud are understandably wary about trusting the service with information as personal and potentially damaging as their medical records and bank accounts.

Most of the cloud services fall into three main categories:

- Infrastructure As A Service(IaaS): categorized as a basic cloud computing service, virtual and cloud space is rented and paid-as-you-go basis.
- Platform As A Service (PaaS): refers to cloud computing as on-demand supply environment development, testing, managing and delivering software applications. it make it easy for developers to create mobile applications without worrying about managing and setting up underlying servers infrastructure, storage and network needed for development.
- Software As A Service (SaaS): basically is a method for delivering software applications over the internet, on demand and on subscription basis. it is able to host and manage software applications and underlying infrastructure and handle maintenance like software upgrades and security patches. the user is able to connect to the application over the internet easily using any web browser using pc, laptop and even mobile phones.

Cloud Service Models

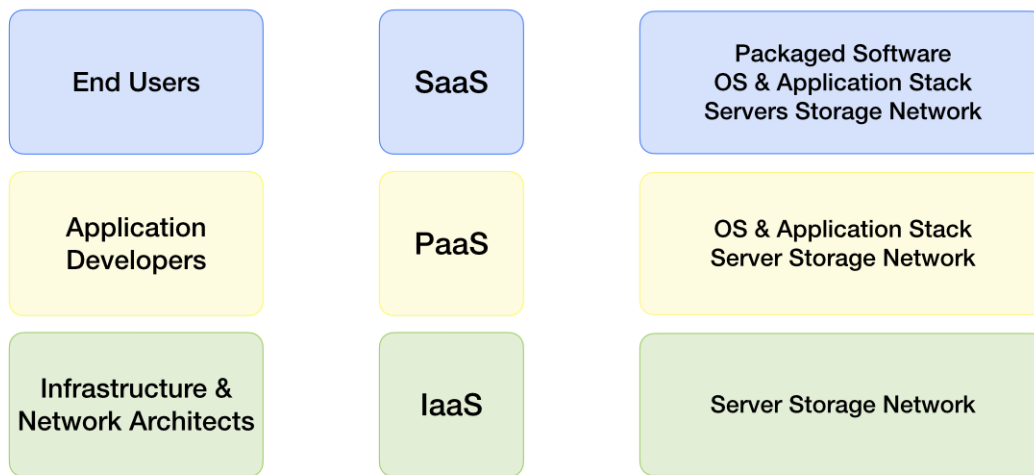


Figure 1.2: cloud services categories explained

Several methods can be modified to lessen concerns over data security and contribute to Cloud users' access to secure, reliable storage of their data. Customers with sensitive data on the Cloud can take extra precautions by encrypting it, but a special type of encryption is required for data that is already being processed. An encryption that lets the cloud service provider perform calculations on top of encrypted data. Homomorphic Encryption is a type of cryptography (HE).

In 1963 the project MAC was the world's first large-scale experiment in personal computing, and it was founded in Massachusetts Institute of Technology (MIT) by the U.S. Department of Defense's Advanced Research Projects Agency (ARPA) and National Science Foundation. The primary idea of the project MAC is to find a solution to develop general-purpose time sharing capabilities, which means to access a single program by multiple users from different locations (Waldrop, 2001). MIT Computer scientists Robert M. Fano and Fernando Jose Corbato were the founding members (DARPA, 2021). Six months after the project MIT was founded, around

200 users could access the program from different departments within MIT (Campanile, n.d.). The spread use of the word cloud started in 2006, when google and amazon started to use “cloud computing,” People were using the web to store files instead of physical storage.

Companies like amazon and google in the past have put a lot of work in research and development for cloud computing. The google cloud platform and the amazon drive both have gained millions of users. The system provided by google and amazon ensures that users can have hardware free backup which is available anywhere and can be accessed by any device at any time (Bahrami, 2015).

Cloud computing has been evolved since it was founded. and it has provided various benefits in terms of cost savings and scalability, but it also poses new security concerns about sensitive data. Following this, and in order to perform actions on cloud-stored encrypted banking data without decrypting them, we will choose for Homomorphic encryption. This latter method involves encrypting the client’s identification with a multiplicative Homomorphic encryption that corresponds to the ”logical and” used for searching for keywords in an encrypted database. In addition, the data regarding a client’s deposits across all of his or her accounts is encrypted using additive Homomorphic encryption, with the cloud server calculating the total amount of the client’s deposits and the client’s private bank deciphering the result.

In both instances, the outcome is identical, as if calculations were performed using raw data. The lack of cloud services usage, not only by banks but by all companies in all industries, creates a strong incentive for researchers and cloud providers to adapt their solutions to the security requirements of banks, thereby encouraging banks to migrate and outsource their computations to the cloud and allowing them to concentrate on their core The cloud move will result in the loss of control over consumer bank data (account

numbers, sum of deposits and loans allocations). There are numerous convincing arguments for migrating apps and data to a private or public cloud, including scalability, agility, cost savings, etc. However, these advantages are accompanied by a new security risk for mission-critical corporate data, which must be managed by any organization whose data has been migrated to the cloud using a secure solution for data encryption and encryption key management (Papisetty, 2017a). This vulnerability is exemplified by the USA PATRIOT Act, which enables rivals and governments to access data through cloud providers without the data owner's consent. Even keeping data in other jurisdictions is insufficient to ensure that customer information is not subject to disclosure obligations. Some security suppliers need the installation of a physical key management server in the data center as a prerequisite for resolving this issue. Other security providers demand that you "trust" them in order to utilize their key management service. This method contradicts both the premise and the criterion of having the keys within your control.

The banking industry is recognized for the quantity and velocity of the data it generates, delivers, and stores. Given the recent financial crisis and accompanying regulatory restrictions, the cost of computers has increased rapidly not only for the entire banking institution but also at the division level. Banks must now reduce their IT expenses, but this should not be done at the expense of data security and integrity. In order to cut IT expenditures while preserving data integrity, it was necessary to investigate unorthodox and creative approaches. With limited resources and funds, however, a standard IT installation process is neither effective nor desired. In recent years, Cloud Computing as a technology alternative for data and business processes has gained traction in various industries, including financial services. Security must be the first focus of your cloud provider. In fact, the majority of Amazon Web Services (AWS) investments are devoted to increasing this

level of security using a shared responsibility paradigm. AWS is responsible for the security of the lower levels (physical buildings, networks, virtualization, etc.), while you are responsible for the security of your applications, operating system, and data over which you have complete control (e.g., administrator access to a virtual machine). This implies that you apply the same security policies to your cloud infrastructure apps. Now it primarily contains three services which are: infrastructure as a service (IaaS), software as a service (SaaS), and platform as a service (PaaS). Furthermore, it has been widely used and has become a valuable solution for companies to access information infrastructure and hardware resources. On the other hand, cloud technology made companies able to utilize services managed by other third-party companies (Acar, Aksu, & Uluagac, 2018).

1.3 Open Banking Threats

Cyber threats have increased in the last decade because most daily bases routines have become digitalized. Many people shop, buy groceries, work, and even open bank accounts through the web without the need to go out and reach to the Bank brunch, this digital life made it risky for the consumers and organizations and made it very easy for cyber criminals to attack if both the consumer and organization if they did not protect themselves against cyber threats (Williams, Chaturvedi, & Chakravarthy, 2020).

In open banking, since 2018, most of the attacks involved APIs and are likely to continue increasing. Banks and financial institutions must have comprehensive cyber security controls, because the days of attackers are shifting from traditional software attacks to different targets like passwords, logins, and APIs. Stealing user information and targeting users' vulnerabilities using that information. Identity Theft has also become a huge issue.

Third party services used by banks also creates vulnerability in the system. Banking and other institutions are using different ways to protect their data, many methods are being used in order to keep the hackers away from breaching their systems and stealing confidential data.

Encryption is one of the methods being used to secure the data, which it is used to protect the transmitted data through the internet, Data Encryption Standard known as (DES), is considered outdated and there are some alternative algorithms. Most financial institutions, and any organization needs to encrypt their data and keep them protected at rest in the cloud or in the server, employ advanced encryption standards and international data encryption algorithms. RSA is one of the most used public key encryption algorithm, also known as Asymmetric cryptographic algorithm. This algorithm works by setting different encryption keys for both sender and receiver (Meneses et al., 2016).

Open banking, which involves sharing financial information with third-party providers through APIs (Application Programming Interfaces), brings about various benefits such as increased competition and innovation. However, it also introduces certain security and privacy concerns. Here are some threats associated with open banking and how homomorphic encryption can help address them:

1. Data Privacy and Confidentiality:

Threat: Open banking involves sharing sensitive financial data with third-party applications, raising concerns about data privacy and confidentiality.

Solution with Homomorphic Encryption: Homomorphic encryption enables computations to be performed on encrypted data without decrypting it. In the context of open banking, this means that sensitive

financial data can be processed by third-party applications without exposing the raw data. Homomorphic encryption helps maintain the confidentiality of the data throughout the computation process.

2. Man-in-the-Middle Attacks:

Threat: Open banking APIs are susceptible to man-in-the-middle attacks where an attacker intercepts and potentially alters the communication between the user, the bank, and third-party applications. **Solution with Homomorphic Encryption:** Homomorphic encryption adds an extra layer of security by encrypting the data in transit. Even if intercepted, the encrypted data remains unintelligible to the attacker. This helps prevent unauthorized access to sensitive financial information during the communication between different entities in the open banking ecosystem.

3. Identity Theft and Authentication Issues:

Threat: Open banking involves the exchange of user credentials and other personal information between various parties, increasing the risk of identity theft and unauthorized access. **Solution with Homomorphic Encryption:** Homomorphic encryption can be used to protect user credentials and personal information. By encrypting this sensitive data, even if it is accessed by unauthorized parties, it remains in an encrypted form, ensuring that user identities and authentication details are not compromised.

4. Data Residency and Compliance:

Threat: Open banking services may operate across borders, leading

to concerns about data residency and compliance with different regulatory frameworks. Solution with Homomorphic Encryption: Homomorphic encryption can assist in addressing data residency concerns by allowing computations on encrypted data without the need to decrypt it. This means that sensitive data can remain encrypted even during processing, helping financial institutions comply with data protection regulations across different jurisdictions.

5. Insider Threats:

Threat: Employees within financial institutions or third-party providers may pose a threat by intentionally or unintentionally accessing and misusing sensitive financial data. Solution with Homomorphic Encryption: Homomorphic encryption helps mitigate insider threats by limiting access to the actual data. Even employees with legitimate access to the systems would only be able to work with encrypted data, reducing the risk of unauthorized data access and misuse.

homomorphic encryption provides a powerful tool for addressing security and privacy concerns in open banking. By allowing secure computations on encrypted data, it enables financial institutions to embrace the benefits of open banking while ensuring the confidentiality and integrity of sensitive financial information. (Behbehani, Rajarajan, Komninos, & Al-Begain, 2022)

Frequently without the bank's knowledge, many TPPs have utilized consumers' login credentials to screen-scrape data from bank client web portals and apps. Screen-scraping is seen as insecure, may fall outside the defined scope of existing AML regulations, and its use in the EU has been severely restricted. Where this popular practice is less regulated, banks must explore

how to regulate it. It is possible for bad actors to use the Cloud for illegal or unethical objectives. An adversary, for instance, may host malicious code or launch an assault against clients using the nearly limitless processing power and resources. There is a risk that service providers will mislead potential consumers by promising them boundless resources. Malicious code developers and spammers can take advantage of the anonymity provided by these registration and usage mechanisms (Bouti & Keller, 2012). Multiple attacks of this type, in which the customer has heightened influence over the infrastructure, have been experienced by IaaS and PaaS providers. SaaS, on the other hand, was subject to fewer attacks since its architecture restricts users to accessing their data via a centralized, hosted application. In an ecosystem, AML transaction monitoring is more complex.

An financial institute may observe money' last hop into their accounts, but has no visibility into or across TPPs and other financial institutes via which the funds have previously transited, possibly including their own systems in circumstances when consumers hop in and out again. To ensure AML/CTF and sanctions compliance, financial institutes must be able to comprehend transaction chains throughout the ecosystem (Pomerleau, Lowery, et al., 2020). Application Programming Interfaces (APIs) or other software interfaces allow customers to manage and communicate with Cloud services. These APIs are utilized by cloud providers for data management, monitoring, and synchronization. Therefore, if the security of the underlying APIs is compromised, it will have a knock-on effect on the security and availability of the general services. Because the APIs are publicly available, security and activity monitoring for the APIs need to be built into their initial design. In unregulated marketplaces, compliance officers may face an increasing but disorderly TPP environment. Without API standards, centralized databases, and explicit AML/CTF compliance guidelines for TPPs, it will be significantly more challenging to acquire complete visibility into

the flow of funds across accounts, borders, FIs, and fintech's. This presents additional options for the placement, stacking, and integration of funds, particularly in instances where TPPs are not held to rigorous AML/CTF and KYC rules. In addition, it is difficult to conduct sanctions screening without visibility into the origins and destinations of funds. Many businesses are aware of the risk posed by dishonest employees working in their provider's data center. These employees need access to the data in order to manage the Cloud services, but they may also exploit the provider's lack of openness. When it comes to questions like how employees are hired or how users gain access to the Cloud's infrastructure, providers don't always have all the answers. If an opponent can gain access to staff accounts, they may steal sensitive information or take over the cloud services completely without anybody noticing. As a result, the service provider must carry out all necessary measures, such as keeping tabs on staff and conducting thorough audits of their operations (Choo, 2011). Open Banking incorporates newly identified AML/CTF concerns, such as money laundering through cryptocurrency exchanges. Some cryptocurrency exchanges are especially designed to entice financial criminals due to their ability to provide anonymity and conceal the source of payments. Know your partner (KYP) processes, whether regulated or not, must be strengthened in order to monitor the crypto realm for AML/CTF and sanctions compliance. Conventional information technology places data under the strict control of the customer in a more compartmentalized environment. However, due to the functional properties of the Cloud environment, data stored there can be jeopardized in more serious ways. It's important to back up your data in the cloud in case anything gets erased or changed. Threats to encryption, such as employing improper encryption algorithms and techniques, and poor key management practices, can also result in a loss of data secrecy if it is stored on untrustworthy media. Lastly, secret information should be accessible only by those who have a legitimate

need to see it. Fraud, phishing, and the use of software vulnerabilities are still actively being developed and used by attackers. The severity of such attacks is sometimes magnified when victims reuse sensitive information like passwords. Intruders can do data manipulation, return false information, eavesdrop on activities and transactions, and reroute clients to malicious websites if they get access credentials to the Cloud. Authentication and authorization flaws are a major security risk for the Cloud. Furthermore, the attacker may utilize the compromised account or service as a springboard for other assaults.

cyber attacks techniques has been changed since 2016, as the attackers has been focusing on attacking financial institutions, Distributed Denial of Service (DDoS) attacks has been increased, and its targeting banks. in order to mitigate this type of attacks, these institutions contracted with DDoS mitigation services. although, this mitigation method can only detect the traditional types of DDoS attacks.

Ransomwares, another method with different effect on targeted institutions. modern Ransomware known as the most devastating forms of cybercrime, it is common way for the attacker to extort money from the victim, victims are willing to pay up big amounts rather than data exposure or data loss (Carter, 2017).

1.4 Research Question

since open banking is build to share financial institutions and banks in the cloud using secure programming interfaces known as (APIs).this big shift in service providing method address some serious significant issues for both the supplier and consumer. data is the main concern in open banking applications, the main concern is delivering services with no security issues, open banking platform should be technically secure and to protect end users

data.

the main problem questions addressed in this thesis can be split into three parts:

1. how to provide a protected cloud storage for big data, which different banks and financial institutions going to share.
2. how to provide a third party application, which consumers can feel safe to use without the risk of losing or stealing their data.
3. What are the performance implications and scalability challenges associated with the implementation of homomorphic encryption in real-time financial transactions within the open banking?

1.5 Research Significance

Cyber threats has been targeting the banking sector and has been thriving specially in the last years, and causing them losses in many different aspects. because of the pandemic effects all of the services have been digitized specially in the banking sector and as well other institutions like fintech.

open banking applications are spreading, because of its benefits for consumers, as it is saving time and them being able to access all bank accounts from different banks in one platform and making payments from their devices. Services like bill payments can be scheduled and sending money can be easily done from your device.

Open banking applications is working by accessing all banks data in one cloud they use third party services to access financial data and provide services to the user. Putting the data on the cloud makes it attractive for the attackers,by allowing them to hack inorder to get information of multiple users of the cloud, they can also get access to customers personal information which can then lead to multiple attacks and the system and the user

both are compromised. To ensure the security of open banking applications, rather than the known security measures usually taken by organization (Al Nafea & Almaiah, 2021).

Homomorphic Encryption Algorithm, was introduced in 1970 when it was in the first stages of implementation. through the years after it has been introduced, many researchers adopted the idea of Homomorphic encryption to be used in modern applications. Fully Homomorphic encryption can be used as a cloud security solution. When constructing the Fully Homomorphic scheme, Craig n included four methods; key generation algorithm, encryption algorithm, decryption algorithm, and evaluation algorithm. Multiply Homomorphic encryption algorithm and additives Homomorphic encryption algorithm are the basic types of homomorphism, as Gentry work to support multiplication and addition in the same time, corresponding to AND and XOR in Boolean algebra, which any computation can be converted into a function containing only AND and XOR. As known in algebra can use several techniques to convert a function, so using any of these techniques, a function can be converted by using only a specific Boolean operation (Halevi, 2017).

Banking and finance technology industry is transformed by using cloud technology, with more flexibility provided for the growing needs in this industry. Using online services, it has become easier and perhaps less hectic for banks to provide services to customers and for data collection. Cloud computing becomes useful for data management, enabling Fintech companies to store and manage the data in a secured, trusted way while implementing fully Homomorphic encryption, allowing the businesses in this industry to take advantage of this approach serving the consumers with banking products online. Many countries around the world are now focusing on fintech as a way of investment to using it in daily life purchases makes it easier to track the number of transactions and the flow of cash.

Google and Amazon started using cloud computing in public services but the use of cloud computing has increased. Although cloud computing has been used rapidly in the previous years, especially in the Fintech industry, there are many problems and challenges with implementing cloud computing; like safety, stability, and performance issues, and the security is the most concerning issue, as their many incidents reported reveals security issues. The best way to assuage these concerns regarding the security of cloud data is to store all the data in the cloud encrypted and perform computations on the encrypted data stored in the cloud (Marston, Li, Bandyopadhyay, Zhang, & Ghalsasi, 2011).

Homomorphic encryption schemes allow that to happen, as the data should be encrypted with Homomorphic encryption before being sent to the cloud, and secure safe storage of data in the cloud, in this order even the cloud computing service providers cannot quickly obtain the information of the plaintext. Fully Homomorphic Encryption enables the users to process the cipher-text data directly instead of the original data. The retrieval method of the Homomorphic is based on cipher-text, which can be searched directly on the cipher-text data. This can ensure query privacy and improve the efficiency of data retrieval, which can be added and multiplied without changing the corresponding plaintext (Chase et al., 2017).

There are plenty of studies in the implementation of Homomorphic encryption in cloud applications, and it shows a promising future about implementing it in the real world. The argument is still there between fully Homomorphic encryption and somewhat Homomorphic encryption to use in cloud computing. Some addressed that there are some disadvantages of using Homomorphic, one of the most considered disadvantages is that Homomorphic requires modification of the application and specialized client-server to make it work efficiently. Fintech organizations cannot take this risk

based on methodologies and studies, which it will cost the organizations and destruct them from the more critical initiatives. Only with Cloud Encryption can data be encrypted and keys managed easily in virtual environments without compromising security. In recent years, there has been a rise in regulations concerning the safety of sensitive information and computational techniques. As the technological landscape expands, hackers are able to target a wider range of digital assets and infrastructure. Data encryption is one of many options for storing and reading sensitive information securely. When asked to conduct a computation using encrypted data, the difficulty of the problem increases. Here, Homomorphic cryptographic methods can be put to good use. Partial Homomorphic encryption facilitates various data manipulations without requiring decryption (Yousuf, Lahzi, Salloum, & Shaalan, 2021).

1.6 The Study's setting

When encrypting, it means changing the form of the text, such as employing codes, so that it cannot be deciphered. In the early days of data encryption, there were only two ways for encrypting data: traditional encryption and asymmetric encryption. It can either transpose or substitute your existing letters or numbers with new ones using any of these two procedures, which are known as transposition and substitution, respectively. Substitution came second after transposition. The Caesar cipher, one of the oldest in use, was created by transposing the letters, as has already been explained. With the use of codes, substitutions of letters and words began to appear later.

During World War II, the German military developed the Enigma machine, which used substitution ciphering and a code book to encrypt its messages. Transposition and substitution were the only methods of tradi-

tional encryption, and this type of encryption has been in use since antiquity, which aided researchers in developing more advanced encryption techniques like public-key encryption and Homomorphic encryption, which will be discussed in the following section (D. K. Sharma, Singh, Noola, Doss, & Sivakumar, 2022).

Asymmetric and symmetric encryption are often known as secret-key and private key encryption. To encrypt and decrypt the plaintext in Symmetric-key encryption, there is only one private or secret key shared by the sender and recipient. In the majority of the symmetric-key algorithms, stream and block ciphers are employed. Key generation, substitution, and transposition are common steps in this sort of encryption. Symmetric-key algorithms include AES (Rijndael), Triple-DES, Blowfish, IDEA, and so on.. Key sizes and substitution and transposition are different for each method. In Symmetric-key encryption techniques, plaintext assaults are more likely because of the use of a shared secret key, which is not always exchanged securely.

With asymmetric key encryption, the public and private keys are generated at the same time and then used to encrypt data. In contrast to symmetric-key encryption, the sender and receiver do not have access to the same private key. To decrypt a given encrypted message, only the recipient of that message must have access to both parties' private keys. Key creation, transposition, and substitution of letters are among the techniques used in this sort of encryption. Because the secret key is never exchanged in an insecure manner between the sender and the receiver, asymmetric key algorithms are impervious to plaintext attacks. It is also more powerful than Symmetric-key encryptions. Asymmetric-key algorithms such as RSA, Diffie-Hellman, and so on are well-known and widely utilized (Dixit, Gupta, & Trivedi, 2018) (Zhang, 2021).

One of the strongest and most secure forms of encryption is Homomorphic encryption. Homomorphic encryption can be described as the conversion of data into ciphertext which can be analyzed and worked with as if it was retaining its original form. Homomorphic encryption enables complex mathematical operations to be performed on encrypted data without compromising the encryption. Encrypting the plaintext and then doing calculations on the encrypted text using a technique known as Homomorphic encryption keeps the plaintext private, so it cannot be decrypted. Because it is found in so many cryptographic tools, Homomorphic Encryption can be considered a foundation stone of modern cryptography.

The antipodal features of an encryption scheme are homomorphism and non-malleability. Homomorphic in mathematics are used to describe the transformation of one data set into another data set keeping the relationships between elements in both sets preserved. Because the data in a Homomorphic encryption scheme retains the same structure, identical mathematical operations will provide equivalent results regardless of whether the action is performed on encrypted or decrypted data. If a function f is nontrivial, anyone can transform an encryption of m into an encryption of the function $f(m)$ using a Homomorphic encryption scheme. As an alternative, non-malleable encryption ensures that no adversary may convert an encrypted copy of message m into another encrypted copy of any “related” message. The main difference between Homomorphic encryption and typical encryption methods is that it enables mathematical computations to be performed directly on the encrypted data, which can make the handling of user data by third parties safer. The Homomorphic encryption was designed to create an encryption algorithm that enables an infinite number of additions to encrypted data (Mohan, Devi, & Prakash, 2017).

1.7 Research Terminology

Having a workable HE system is the ultimate goal of collaborating with HE program. When encrypting, we mean changing the form of the text, such as employing codes, so that it cannot be deciphered. In the early days of data encryption, there were only two ways for encrypting data: traditional encryption and asymmetric encryption. You can either transpose or substitute your existing letters or numbers with new ones using any of these two procedures, which are known as transposition and substitution, respectively. Substitution came second after transposition.

The PHE and SWHE programs were milestones in the right direction. PHE has made it possible for numerous cloud services and systems to gain advantages, even while they only have restricted applications. Banking applications use APIs to send financial data and allow users to access certain services.

Banking applications use the financial data and suggest, display products and services suited to the user using the data. It relies on network instead of focusing on centralization and physical interactions. All sort of finances and history is available to user which can make it easy and simple for the user to view his finances and can form decision about mortgages or loans. Instead of using passwords and usernames for each account which is risky and can be easily attacked they focus on APIs which are far more secure. Open banking can be done from the bank's website or from the mobile application. Wallets and other kind of fintech also use APIs. Banks also utilize AI, big data and blockchain technology to facilitate transactions in the internal network. To automate this system robotic process automation is used which emulates human working and makes it an easier experience for the user. To make this more secure encryption is done. When using CryptDB, you can encrypt your database and still do standard SQL queries. By utiliz-

ing CryptDB, the server may carry out user-submitted queries. The server decrypts the customer's data, processes the customer's queries, and returns the data and its resulting values. Once the consumer receives the encrypted result, they can decipher it. No query is processed on the customer's machine (Hernández, Öztürk, Sittón, & Rodríguez, 2019).

Since the Paillier cryptosystem allows for simple arithmetic operations on integers, it was adopted. Homomorphic encryption is used to securely tally votes in Helios 2.0, a cloud-based open audit voting system. The votes can now be securely tabulated and submitted. After all votes have been cast and counted, the result will be revealed. A modified version of ElGamal has been used in the PHE scheme (Jabbar & Alsaad, 2017).

The homomorphic encryption protocol Porticor is a commercial implementation of this protocol. It was one of the earliest implementations to provide a PHE-based, cloud-based key management system with a high level of security. Porticor was bought up by another company called Intuit. Conversely, most SWHE methods proposed after Gentry's idea are already included into other FHE schemes. As a result, it is not easy to distinguish between SWHE and FHE implementations. Our goal in this part is to examine a few of these applications. After a cryptographic scheme has been designed, it must be put into practice. After the scheme has been implemented, it can be tested in the wild against a real-world service to determine how well it performs. The high cost of the FHE implementations is a key drawback that hinders deploying FHE in real-world services without disrupting customers, despite significant improvements in capacity and performance of the implementations of the new suggested FHE schemes. Squashing was originally optimized and made easier by Gentry et al. They are credited with first implementing the FHE technique. They showed toy,

small, medium, and giant tiers of protection. (Tebaa, Hajji, & Ghazi, 2012)

Encryption algorithms and tools like DES, RSA, AES as well as Homomorphic encryption is used. Multiple layers of security like OTPs are used along with email verification. SMS are also used to alert the user about everything. From login the application to money transfer everything is informed in real time. Bank calls customer if any suspicious activity is suspected or if the login is in a different location. All these security measures are used to make the experience safe for the bank and the user. Different banks in different countries use different payment services which are integrated with the app which makes all payments possible. Even apps like apple pay and google pay use similar principles for EFT (Electronic Fund Transfer). Open banking permits third-party payment service providers and other financial service providers access to the personal and financial data of their customers' banks. Prior to this, the client must authorize access for the sharing of information, typically through an online consent form following an agreement of terms and conditions. The third-party providers can then utilize available APIs to access the relevant shared data. These APIs can handle transactions from one bank to another without requiring the cumbersome processes that consumers previously had to perform. Additionally, APIs can utilize a consumer's transaction history to identify products and services that enhance the customer experience. Examples include a new credit card with a lower interest rate or more cashback than the consumer's current card, or a savings account that generates more interest than the consumer's current means of saving.

Every open banking API call is centered on data, so agile access to data should be the first port of call for every new system. However, despite the fact that every fintech company desires an agile, efficient, and scalable data lake, the majority have a data swamp: fragmented data sources, a mix of

old and new, real-time and streaming data, and a labyrinth of organizational boundaries. To overcome this, an effective integration system is required. Before you can expose the data in your apps via APIs, you must guarantee that you are working with a complete and correct picture of all the data and that the data that users interact with is current, accurate, and comprehensive. This is the benefit of integration. Once integrated, your application data can be directly and securely exposed via APIs. Application integration and data virtualization are typically the two methods for achieving this data integration. Physical integration enables APIs to securely call your backend systems directly. In contrast, data virtualization is precisely what it sounds like. Instead of physically merging all your data sources, you utilize a virtual layer to integrate them. Data virtualization enables teams to combine dozens of distinct data sources into a single virtual data warehouse with virtually identical performance to a single system. Instead of overusing ETL to create a larger data swamp for APIs, data virtualization leaves the data in its current location. This provides a consistent interface for accessing customer data as if it were a single system. Consider both integration options to help you control your data swamp (Kurze, Schopp, & Moxon, 2018).

Chapter 2

Literature Review

2.1 Overview

Evolution of the digitization in banking services, makes it mandatory for banks and financial institutions to achieve the highest possible security control on the financial applications. when the pandemic started, it made banks and financial institutions to rebuild their strategies concerns digital banking, due to the difficulties for the consumers to transport within the high spread of covid-19 different variants, and the challenges arose for the traditional banking with the revenue, traditional banking specially had to make decisions and start to launch these applications to serve consumers with almost all banking services in order to mitigate risks related to covid-19 pandemic. Banking operations that previously needed the consumer to go to the bank now has been changed and being replaced with mechanism that allows transactions to be done anywhere, changing of strategies and shifting to digital banking, has made it easy to third party companies to adopt the develop-

ment of open banking application. Open Banking (OB) is used to describe a different type of ecosystem that are governed by a set of rules, policies and regulation for best consumer operation experience. OB allows third parties to access banking data through the use of Application Programming Interfaces (API's).

The biggest and most advantage change to banking sector adopting open banking is allowing the consumers to not consider visiting the bank as a regular thing to do in their life, as open banking makes the bank visit not a necessary journey for consumers. since Covid-19 pandemic, people prefer no control their financials online rather than visiting the bank in person, also because staying at home has becoming essential for people's health the accelerated transformation and adoption of open banking applications since the pandemic started in 2019 brings many benefits, and risks. while open banking is evolving, also cyber risk was evolving in parallel. while banks were dealing with the difficulties of changing from the traditional banking, the criminal cyber activities are thriving, including phishing attacks and fraud which has been increased. Due to the high increase of threats and cyber-criminal activities, banks, financial institutions and third parties are trying to overcome the obstacles of cyber security in open banking. (MFA) multi factor authentication is the most effective and commonly implemented to secure any authentication of an application can be used recently along with other types of authentications and security applications and standards. data on the cloud is the most critical data need to be secure, one of the proposed methods, and yet not have been totally applied is Homomorphic Encryption Algorithm, its and encryption method that can protect the data on the cloud (Not, 2018).

2.1.1 Homomorphic Encryption History

Homomorphic encryption was first introduced by Ron Rivest, Alderman, and Dertouzos in 1978 after the public key scheme was developed. After nearly 30 years, in 2009 with Gentry's work breakthrough, this constructed fully Homomorphic encryption. This type of encryption allows operating on encrypted private data, while cloud computing, such as the Fintech industry, requires a more practical secure encryption solution. In this case, Homomorphic encryption is the best practice to provide such security to the data stored. In 1978, in tandem with the introduction of the RSA cryptosystem, one of the earliest widely used encryption techniques for transmitting data, the concept of Homomorphic encryption was originally hypothesized. In RSA, information is encrypted with a (public) key in order to render it unintelligible. The recipient uses a different (private) key to decode the information once it has been delivered to them. In 1978, it was questioned if data could be encrypted in such a way that it could still be used for various operations (such as addition and multiplication) without first decrypting the data and so disclosing private information. Over the past three decades, many solutions have been offered that enable a certain function to be performed, but no fully homomorphic system where any transformation is possible has been discovered. For the first time in 2009, a fully Homomorphic encryption (FHE) system was introduced, and the decade of the 2010s saw major improvements in the effectiveness and practicality of FHE systems (Kogos, Filippova, & Epishkina, 2017).

In homomorphic encryption, the encrypted text itself is the object of mathematical operations, rather than the original data. In order to get the intended outcome, it is processed and then decrypted. Homomorphic encryption is distinguished by the fact that decrypting the encrypted cipher text should yield the same result as operating on the unencrypted plain text. Three main types of Homomorphic Encryption are there, but the only difference be-

tween them is the types and frequency of the mathematical operations that are performed on the cipher-text:

- Partially Homomorphic Encryption (PHE): only one operation can be performed unlimited times on the same cipher-text, the only selected mathematical value. (PHE) with multiplicative operations is used as a foundation for RSA encryption, which is used to secure connection through SSL/TLS.
- Somewhat Homomorphic Encryption (SHE): supports additional or multiplication operations with limited complexity and can perform as a set number of times.
- Fully Homomorphic Encryption (FHE): when in the development stage, it can constantly function with privacy by keeping the information secure and accessible at the same time (Not, 2018).

Importantly, organizations are still responsible for protecting user privacy even whether data is encrypted (Homomorphically or otherwise). It would still be personal information at its core, so strict data management and control would be needed to ensure it was used appropriately. The lack of generally acknowledged standards and the inherent limitations of existing solutions make widespread adoption of Homomorphic encryption impractical at the present time. Analysis on completely Homomorphically encrypted data (where any type of operation is available) is several orders of magnitude slower than the same analysis on unencrypted data, and many homomorphic encryption systems only allow only one type of operation (e.g. addition or multiplication, but not both). Therefore, this method can only be applied in situations where a small subset of functions is required (as is the case with HE) or where computing speed and cost are not crucial (in the case of FHE). However, developments in this field have made it possible to finish certain computations in a matter of seconds or minutes, opening the door to the use

of homomorphic encryption for the protection of sensitive data. Start-ups like Ziroh Labs and Inpher have created HE and FHE methods that are computationally viable for real-world use cases, thus this is still an area of active development. The widely adopted standards of legacy encryption technologies permit a high level of interoperability and widespread use. Since there is no generally agreed-upon standard for HE or FHE algorithms, the utility of any particular homomorphic encryption method is severely limited. Homomorphic Encryption Standardization is just one example of an ongoing effort to establish industry benchmarks for this innovation. From the development of a somewhat homomorphic encryption scheme (FHE), both multiplication and addition can be used any number of times and establish secure multi-party computation more efficient, more than any other type of Homomorphic encryption, which it can handle arbitrary computations in the cipher-text. For example, the process begins with some plain text message, mm. It could be the name of someone in a computer system, "jane doe". The goal is to perform some function ff on it, like capitalizing the name. That's possible, but it would be safer to encrypt the message using encenc before performing any functions on it so the person performing the function never learns Jane's name. So, the message is encrypted to some cipher text 163726. Then, it is evaluated, or transformed, into another value using some other function, $f(x) = x + 127$ $f(x)=x+127$, for example. The output, 163853, is another completely encrypted message. This message can then be decrypted to "Jane Doe" (Tebaa et al., 2012) (Dowlin et al., 2017).

2.2 Online Banking and Security Related Work

since the financial industry started to be shifting to cloud computing, securing data on the cloud has been an attraction for the researchers in the last decade, some findings and solutions has been adopted in existing cloud en-

vironment. some of the proposed solutions focuses on the authentication of the user, protecting API's, framework applications, and others focuses on the security of the data itself by applying encryption solutions.

Canada's financial sector is based on Scottish national branch banking system, for that it has preventing failure in the financial sector, as a result of Canada's banking sector resilience it has been recognized for its stability and strong security (Taylor-Kerr, 2020). the Financial Data Exchange (FDX) is creating and working towards a way to create a safe and transparent way of sharing financial tech. Privacy enhancing technologies are a must for financial data exchange as they make it easy for the companies and users to exchange their data. Canada's success plan of blockchain are relying on the fact of deploying wide legislative and regulatory environment (Ducas & Wilner, 2017).

In 2017, proposed theoretical solution, addresses the gap in mobile applications issues. mobile applications has many threats, as the attacker may attack the users of the application, these threats are: malware, wireless network attack, denial of service, and break in, which is when the attacker can abuse the flaws of the application code. the European Union directive known as Payment Service Directive (PSD2) was developed after the Payment Service Directive (PSD), because it has rules flaws regarding payments through the internet, and wireless devices (Wolters & Jacobs, 2019). the main purpose of the directive is to create strong protection for the end user, ensure the security of the electronic transactions. a way to legislate remote payments from service provider and financial institutions, also to ensure the authentication of remote payments in order to create consumer trust. the proposed solution addresses the importance of the asymmetric encryption as an added value security method of data encryption, as it adds authenticity, integrity and non-reputability, to PSD2 as a feature. (Kaipainen, 2017).

another solution proposed an Elliptic Curve cryptography (ECC) solution to secure cloud computing applications, which is a public key encryption based on elliptic curve theory, that can create cryptographic keys faster, smaller and more efficient. ECC works by generating keys through the properties of the elliptic curves equation instead of traditional generation method. researchers addresses that elliptic can provide security with 164-bit key despite other systems that require 1024-bit key. ECC can provide security with low computation power and resources, for that it is becoming widely used solution for mobile applications (Alowolodu, Alese, Adetunmbi, Adewale, & Ogundele, 2013) (Khan & Qazi, 2019).

Financial Blockchain, studies claims it can effectively protect financial data privacy, when sharing data in open banking, several management methods are combined for the sake of global data privacy protection emerged. with some considerations with data privacy protection, for example regulations limitations of the country which the financial institution or banks are in. this solution are based on customer-strategy model which consists of collaborative filtering algorithm along with default data privacy disclosure schemes that are proposed based on a nudge theory, results on a concrete blockchain based financial data privacy management technical realization. the future work of this study is to be tested, also the nudging schemes proposed going to be enhanced (Ma et al., 2018).

Homomorphic encryption experiment on IBM's for quantum computing has been deployed. quantum computing has been evolved in the past years. the algorithm proposed provided a speedup over the existing used algorithm. the experiment shows the solving of linear equation on quantum algorithm, the solution allows the user to encode the circuit on IBM's cloud quantum computing platform. overall results shows that the Homomorphic algorithm applied provides a secure platform and secured data on the cloud (Huang et al., 2016).

A verifiable Homomorphic Encryption algorithm proposed for securing big data computing, the study proposed a an efficient verifiable noise-free fully Homomorphic encryption method, this method uses Lipschitz's quaternions and permits computations over the existed encrypted data with symmetric key. the proposed scheme assures if the data computation was performed in a correct form. as a result, the encryption scheme show the effectiveness of the application across the domain of smart computations on the encrypted data in the cloud (El-Yahyaoui & Ech-Chrif El Kettani, 2017).

related studies and implementations that has been conducted in the field of data privacy and security, shows that Homomorphic encryption is a solid solution has integrity, authenticity and non-repudiation comparing to traditional encryption techniques used nowadays, along with data privacy laws and security framework implemented in the organization. the study proposed is differ by implementing the solution in existing cloud database, and comparing the encryption with one of the traditional encryption methods used with Homomorphic encryption, the outcome will show that the encryption proposed cannot be resolved if the data has been hijacked, because the data will remain encrypted while on transit and at rest, as this type of encryption can perform calculation upon decrypted data.

2.3 Discussion

There has been rapid progress in the creation of numerous Homomorphic encryption algorithms since the birth of Homomorphic encryption and continues to this day. These four basic forms of Homomorphic encryption systems, PHE (Partial Homomorphic encryption), SHE (Somewhat Homomorphic Encryption), LHE (Leveled Homomorphic Encryption), and FHE (Fully Homomorphic encryption), were described in the preceding sections. Additive, multiplicatively, or mixed Homomorphic encryption methods are

all examples of this categorization (Drozdowski, Buchmann, Rathgeb, Margraf, & Busch, 2019). From partial Homomorphic encryption to the other schemes in this category, we have Paillier, ElGamal, Unpadded RSA, Benaloh, and Goldwasser-Micali as starting points. Scheme converters can also be used to convert additive schemes to multiplicatively Homomorphic systems. Banks, wallets and other fintech applications have all used cloud computing and as the risk of being hacked is greater and vulnerability is also great all these companies have used encryptions to make their systems more secure and financial data secure (Yaji, Banger, & Neelima, 2018).

Homomorphic encryption, which eliminates the need to decode data before performing actions on it, is the future of cryptographic methods. The Cloud would benefit greatly from PHE systems since they would provide an efficient data protection mechanism and answers to significant privacy concerns. Consider the Cloud-based analyses of DNA sequence databases conducted by scientists in the medical industry. It would appear that Homomorphic encryption's slow execution speed is one of its primary flaws (Armknecht et al., 2015).

2.3.1 Homomorphic Scheme Solutions

Microsoft appears to have done well in this task in the SEAL library which encrypted with fully Homomorphic encryption developed in C++ , but whether or not PHE methods are adopted broadly for use in Cloud infrastructure has to be seen in the years to come (Titus, Kishore, Stavish, Rogers, & Ni, 2018).

By using Homomorphic encryption, a user can submit two encrypted values to a server, which then sends back the summed encrypted values. The output can be decoded by the user to reveal the sum of the input values. In other words, the total of the encrypted values is the encrypted sum of the original values when using Homomorphic encryption. Homomorphic

systems are most effective when used in accordance with their specific qualities. As a result, it is vital to understand the workings, attributes, and applications of schemes (Sathya, Vepakomma, Raskar, & Bhattacharya, 2018).

Homomorphic encryption techniques that allow easy computations on encrypted data have been known for a long period of time. Examples of encryption systems that do not support both adding and multiplying encrypted ciphertexts include Goldwasser and Micali El Gamal, Cohen and Fischer, and Paillier. Boneh, Goh, and Nissim were the first to devise a system that could simultaneously do both adds and multiplications; however, their system could only manage a single multiplication. DGHV, BGV, and FHEW are all examples of fully Homomorphic encryption, while YASHE and FV use leveled Homomorphic encryption Encoding method (Chase et al., 2017).

The Paillier cryptosystem uses the Paillier scheme, which is a partially homomorphic encryption algorithm. Cryptographic method. When given a secret key and the decrypted versions of m_1 and m_2 , this is often an additive homomorphic scheme, meaning that the sum of m_1 and m_2 may be calculated as $m_1 + m_2$. There are homomorphic features for addition and multiplication in this probabilistic asymmetric encryption system. The constant expansion factor of this algorithm makes it efficient in both encrypting and decrypting, as it uses many bits at a time for one operation (Mohan et al., 2017).

When comparing the generalized Paillier scheme to the original, the expansion factor is reduced from 2 to almost 1. With the Paillier cryptosystem, discrete logarithms can be solved as well as chosen-plaintext attacks can be prevented. An early version of the Paillier method could only perform additions on the ciphertext, but later modifications in the technique proposed that multiplication could also be done on the ciphertexts. Partially Homomorphic encryption schemes such as the paillier system are perhaps the most

well-known and efficient. Paillier, in an interesting turn of events, returns to the normal RSA modulus of $N=pq$. Key creation, Encryption, and Decryption are the three main phases in this technique. This is a reference to and explanation of the underlying steps for each of these steps, which can be found at HELib is a free and open-source C++ library that uses ciphertext packing and GHS (another BGV version) optimizations to efficiently implement the BGV scheme (Papisetty, 2017b).

In addition to the basic Homomorphic operations of the BGV scheme, it also offers higher-level procedures that implement the GHS data-movement methods and simple linear algebra. LWE difficulties prompted Brakerski, Gentry, and Vaikuntanathan to develop a novel FHE technique (BGV). Other light-weight methods are used to refresh encrypted ciphertexts instead of re-encryption . However, they can minimize the accumulation of noise, allowing the scheme to analyze even deeper circuits because they do not completely refresh the ciphertexts like re-encryption does (Abinaya & Santhi, 2021). LWE-based FHE and BGV-based BGV encryption schemes were also initially presented, however they differed greatly in their ability to encrypt data without bootstrapping. With BGV-type cryptosystems, the current secret key and modulus change as we apply operations on ciphertexts, which is a crucial aspect. Bits are encrypted using BGV, an asymmetric encryption method. It is based on lattices, as are many other (slightly) FHE methods. Both integer vectors and integer polynomials can be used in the cryptosystem, however the security of the latter is tied to the decisional LWE (Learning With Errors) problem, while the former is linked to the R-LWE (Ring-Learning With Errors) problem (Zhou, Yang, Zhang, & Wu, 2016). Data for a patient's medical record is encrypted before it is uploaded to the cloud storage system by the patient's healthcare professionals (Patient Controlled Encryption), which uses HE as part of a private cloud medical records storagesystem. The patient has complete control over access to the

records and the sharing of secret keys with certain key providers (features include a hierarchical structure of the record, the ability to search encrypted data, and different options for how to manage key distribution). Homomorphic encryption allows us to perform computations on encrypted data, as we have learned. In this case, too, the cloud can update, set warnings, and do other operations on the encrypted data based on the information the patient receives.

With the Paillier cryptosystem in voting systems, each voter transmits an encrypted version of his or her vote to a bulletin board, along with a confirmation that the vote is legitimate. The Homomorphic property of the encryption technique is then used to combine all of the valid votes and produce an encrypted version of the outcome. Decryption and publication can be done by a group of trustees (who share the secret key of the scheme in a threshold manner) (T. Sharma, 2016).

Homomorphic encryption and proxy re-encryption methods are used in a secure data-prevention (RSD) approach that is robust . Everything from key generation to sharing to data outsourcing to data access is supported by the RSD technique. This model solves the issue of user revocation and rejoining, making the handoff procedure easier to initiate. If the data and the function to be performed on the data are both secret and proprietary, HE can be employed. As an illustration, when it comes to investing, data about companies' stock prices, earnings, and inventories can be useful for making informed judgments. Making trading judgments requires the most up-to-date data, which can be streamed continuously in order to do so (Hani & Dichter, 2016). A corporation may prefer to keep these models private in order to protect their advantage and their investment in financial analysts who have spent a lot of time and money developing these new prediction models for stock price performance.

Homomorphic encryption helps to protect some functions, such as up-

loading encrypted data to the cloud, in which the cloud performs its own computation and then returns the result back to the customer in encrypted form. Single server PIR protocol, an essential application of Homomorphic encryption that uses sub-linear transmission, can be constructed from any additively HE scheme . PIR is a cryptographic technique that allows a user to access a database on the Internet or a network without exposing what he is looking for (Persiano & Yeo, 2022). However, I must remain a secret from both the receiver and the sender. If a data miner wants to find out how often certain values appear in a customer's database, Homomorphic encryption can be used to do so in a secure manner. To put it simply, the issue boils down to summing private inputs securely. Only one flow of communication from each client to the miner is required, and no customer needs to speak with any other customers. To maintain privacy in both user-to-data holder and data holder-to-data holder applications, it is necessary to apply PPDM (Privacy-Preserving Data Mining) (Mendes & Vilela, 2017). Studies on the privacy framework for individuals focus on how to acquire and manage user profiles while preserving privacy.

2.4 Summary

PaaS (Platform as a service) and SaaS (Software as a service) are two of the most popular cloud computing services, however there are four modes of cloud computing deployment: private, public, community, and hybrid . Data injection into the cloud environment is the sole option now available, which is sufficient for gaining access to and processing data stored elsewhere in cyberspace (Tsai, Bai, & Huang, 2014). The enciphered data is still consistent and correct, hence this poses minimal problems. Cloud computing is a notable area of application for HE since it may be used to execute tiny or medium computations depending on the individual's privacy

requirements.

The widespread adoption of cloud computing has opened up a massive market for a technique that would enable any user to do mathematical computations on encrypted data, without having access to the secret decryption key, and still provide an encrypted result of the computation. Data stored in the cloud that has been encrypted using any current "standard" encryption approach is practically unusable without the secret decryption key. Privacy is jeopardized when the data is in plaintext form during computation if the secret decryption key is employed. Transferring encrypted cloud data to a private safe server, where data privacy may be protected, is now the only secure option (Jain, Gyanchandani, & Khare, 2016).

Because all services are supplied via web apps with cloud computing, consumers no longer own the information they place in the cloud. The privacy and security of data and computing resources can be jeopardized if they are moved to the cloud. Most cloud storage solutions have an issue with inadvertent leaks of sensitive information. In the cloud computing world, this flaw is referred to as Data Leakage, Cloud Drain, or Data Loss. Cloud computing has a severe flaw in this area . A cloud data store that isn't secure will deter people from using the mobile cloud. When a user initiates the handoff procedure, there is a risk of data leakage. If a hacker gains access to data being exchanged between a client and a server while it is in transit, this is known as data leakage (Venkatesh & Eastaff, 2018). When it comes to cloud computing, one of the biggest issues is that customers can't properly verify the security and confidentiality of a remote resource. Contracts and trust are still required for security, which is a drawback. The transfer of data, unlike the execution of programs, is well safeguarded by a variety of cryptographic approaches. For example, asymmetric methods can satisfy confidentiality, integrity, and authenticity requirements in the field of encrypted communication as well as classic multi-party computations,

which describe a scenario of two parties who want to execute a function with secret input data on each side and a common output. IBM, Google and Amazon are among the many Cloud service providers that use virtualization on their Cloud platform and on the same server to coexist virtualized storage and treatment space belonging to concurrent organizations. Data from each company must be safeguarded by the security and confidentiality component (Opara, 2019).

Data confidentiality is a major issue in cloud computing, which is a form of shared computer infrastructure. MrCrypt, a viable method that uses Homomorphic encryption to ensure data secrecy by automatically modifying programs, has been presented to address this problem of data confidentiality. Java programs are analyzed to identify the most efficient Homomorphic encryption scheme that supports the necessary operations on each column of input data, and MrCrypt then automatically rewrites the program to execute on encrypted data, and instead of encoding each column using a fully Homomorphic encryption scheme, one can encrypt it using a more efficient scheme that supports only the necessary operation (Millstein & Tetali, n.d.). Cloud computing security can be reduced to the task of determining which primitive operations can be used to secure the computations in each column of the input.

Chapter 3

Methodology

3.1 introduction to methodology

3.1.1 Overview of research methodology

The methodology used in this study is extensive and specifically designed to address the complexities of banking data security. Understanding how critical it is to safeguard financial data, particularly in the banking industry where privacy and legal compliance are critical, our approach incorporates a number of factors. To give a comprehensive grasp of privacy-preserving techniques in the context of banking data, it covers cryptographic protocols, regulatory compliance, technological issues, and ethical factors.

The technique is designed to support a systematic investigation, guaranteeing that every aspect of the research is carefully considered. It uses a combination of theoretical analysis and data collection, integrating quantitative and qualitative research methodologies. In addition to creating strong

privacy-preserving measures, the multi-dimensional approach seeks to assess, techniques' efficiency in actual banking situations.

3.1.2 Research Objective and goals

This study focuses on the security and privacy of banking data, an area with significant risks, strict laws, and changing technology environments. The main objective is to create and assess privacy-preserving systems that handle the particular difficulties presented by banking data and go beyond traditional security measures. In order to accomplish this overall aim, the research is structured into several goals:

1. **Secure Transactions using Cryptographic Protocols:** Examine and evaluate cryptographic protocols created to protect banking transactions. This includes delving deeply into key management systems, secure communication methods, and encryption algorithms.
2. **Access Controls and the Protection of Customer Information:** To safeguard sensitive customer data, investigate and evaluate banking system access control measures. To avoid unauthorised access to vital banking data, this goal includes knowing user authentication, authorization, and data access restrictions.
3. **Regulatory Compliance:** Analyze the laws and regulations that control the security of bank data. This entails a thorough examination of the directives and policies issued by the regulatory agencies in charge of the banking industry. Ensuring the legal and ethical integrity of banking data practices requires strict adherence to these regulations.

3.1.3 Scope and limitations of the study

This study's scope is narrowly focused on the analysis of privacy-preserving strategies utilized by banks. This comprises a thorough examination of access restrictions, encryption mechanisms, and legal compliance that are particular to the security of bank data.

3.1.3.1 Cryptographic methods for Secure Transactions

Examining cryptographic methods intended to safeguard financial transactions is the first step in our technique. To find and assess different encryption algorithms, key management systems, and secure communication routes that are being used or suggested for banking data security, a thorough literature review will be carried out. We'll use quantitative approaches to evaluate the efficacy and efficiency of various cryptographic strategies. We will gather and compare information on transaction times, resource usage, and security strength across multiple protocols. A qualitative investigation of these protocols' scalability and flexibility in the changing banking environment will also be part of this phase.

We will work with banking institutions who are prepared to take part in a pilot implementation of specific cryptographic protocols in order to collect empirical data. To evaluate the performance of the protocols in real-world circumstances, they will be simulated. The viability of incorporating these protocols into current banking systems without interfering with regular business operations will also be examined in this study. One of this phase's constraints is that it depends on banking institutions' willingness to take part in the trial implementation. The particular operational quirks and infrastructure of the participating institutions may have an impact on the results. In addition, after this investigation starts, new cryptographic techniques might be introduced due to the quickly changing cybersecurity scenario.

3.1.3.2 Access Controls and Customer Information Protection

Examining and evaluating banking system access control techniques is the second part of our methodology. The goal of this phase is to comprehend the implementation of data access policies, user authentication, and authorization to protect sensitive client information. The study will begin with a thorough analysis of the models of access control already in use and how well they work for the banking industry. This covers various advanced access control paradigms such as attribute-based access control (ABAC), role-based access control (RBAC), and others. In terms of banking data security, the study will assess each model's advantages and disadvantages. The main goal will be to comprehend how well access control systems work to stop illegal access to vital banking data. The impact of these mechanisms on system performance and user experience will also be investigated in this study.

Qualitative methods will be used to collect information about the human aspects influencing access control in addition to the quantitative study. Investigating user behavior, social engineering threats, and the psychological components of security awareness in banking personnel are all part of this.

One of this phase's constraints is that banking institutions can be reluctant to share specifics about their access control measures for security-related reasons. By protecting the privacy and anonymity of sensitive data, the study will try to overcome this constraint. Additionally, the degree of security awareness and training among bank staff may affect how effective access controls are.

3.1.3.3 Regulatory Compliance

A thorough analysis of the regulatory environment pertaining to banking data security is the third component of our technique. Understanding the regulatory requirements set forth by agencies supervising the banking sector

is the goal of this phase.

The study will commence with a comprehensive examination of regulatory documents, directives, and specifications issued by regulatory entities, including international banking associations, central banks, and financial regulatory authorities. This include, but is not restricted to, agreements such as Basel III, the General Data Protection Regulation (GDPR), and the Payment Card Industry Data Security Standard (PCI DSS).

Through a review of the particular security measures required by these rules, quantitative data will be gathered. This will entail compiling a compliance checklist and evaluating how closely financial institutions follow these guidelines.

To comprehend the advantages and difficulties of regulatory compliance in banking data security, qualitative methodologies will be utilized. Policy-makers, legal professionals, and compliance officers will be interviewed to gain understanding of the real-world effects of following these rules.

The phase's constraints encompass the ever-changing regulatory frameworks. The results could be affected if regulations changed while the study was being conducted. Furthermore, due to competitive concerns, several banking institutions could be reluctant to reveal precise compliance details. By concentrating on broad compliance patterns and anonymizing particular institutional data, the study will overcome this constraint.

3.1.3.4 Ethical consideration

Ethical considerations are critical because of the sensitive nature of banking data and the possible effects of study findings on banking organizations and their clients. The research will be conducted responsibly by adhering to the highest ethical standards.

Informed Consent: Consent will be sought in writing before any bank-

ing institution or individual participant is involved. This includes being open and honest about the nature, goals, and possible results of the study. At any point, participants will be free to revoke their consent without suffering any repercussions.

Confidentiality: The highest level of confidentiality shall be applied to all obtained data, regardless of its quantitative or qualitative nature. To protect privacy, any data that might be used to identify a particular banking institution or participant will be anonymized.

Security Measures: To safeguard the integrity and confidentiality of the data, security measures pertaining to cryptographic protocols and access controls will be put into place during the data collection process. This includes encrypting sensitive data, limiting unauthorised personnel's access to study data, and securely storing data.

Transparency: There will be thorough discussion throughout the research process regarding the approach, methods for collecting data, and methods for analysis. All affiliations and conflicts of interest that could affect the research will be declared.

3.2 Research design

This research is based on experimental design, which is the best method to test the efficiency of homomorphic encryption for securing open banking data. implementation method ensures a rigorous and objective evaluation, which enables researchers to make a reliable conclusion of the effectiveness of homomorphic encryption in real-world scenarios.

3.2.1 Explanation of Homomorphic Encryption scheme

A cryptographic technique called homomorphic encryption makes it possible to do calculations on encrypted material without requiring its decryption. The application of homomorphic encryption offers a safe means to carry out different operations on private, sensitive banking data when it comes to banking data encryption. Key elements of a homomorphic encryption scheme within the research design are explained as follows:

Encryption Data is encrypted before being sent or stored using a public key. A homomorphic encryption strategy makes sure that encrypted data can be directly computed on without the requirement for decryption.

Homomorphic operations On encrypted data, operations like addition and multiplication are supported using homomorphic encryption. Financial data that has been encrypted can be used for banking-related computations such as interest rate applications, balance computations, and statistical studies.

Key generation An encrypted key generation secure procedure is part of the research design. It is possible to generate pairs of public and private keys. The public key is used for encryption, and the secret key is stored safely for decryption.

Decryption The private key is used to decrypt the outcomes of homomorphic computations. This guarantees the confidentiality of sensitive data during the computing process.

Integration with banking system The homomorphic encryption method easily fits into the current banking procedures and systems. Banking apps

and encrypted data can exchange and compute securely thanks to interfaces and protocols

3.2.2 Justifications for selecting Homomorphic Encryption

the data collected was created for previous testing purposes in financial sector, the data input was relevance to the research and accurate; which contains personal informal, bank account numbers, and financial transactions. data was gathered and organized in excel sheets. the size of data was convenient for the testing environment and supports the echo system.

Several significant benefits support the choice of homomorphic encryption for encrypting banking data:

Privacy-preservation: Computations on encrypted data can be carried out without disclosing sensitive information thanks to homomorphic encryption. This guarantees banking data privacy, which is essential for the financial industry.

Secure computation: Complex computations are frequently involved in banking processes. These computations on encrypted data are made possible by homomorphic encryption, which maintains security throughout and offers a safe environment for financial transactions and analysis.

Compliance with regulations: Strict safeguards are mandated by numerous regulatory frameworks to protect confidential financial data. Because homomorphic encryption protects data confidentiality while processing, it complies with these rules.

Secure Data Outsourcing: Banks can safely outsource computations to cloud settings or other third-party service providers while maintaining data protection thanks to homomorphic encryption. This makes it possible to process data securely and efficiently without disclosing private information.

Adaptability to various operations: Homomorphic encryption is adapt-

able for various financial computations since it offers a variety of operations, such as addition and multiplication.

Mitigation of insider threats: By guaranteeing that even administrators or service providers cannot read the actual content of the encrypted data, homomorphic encryption guards against insider threats.

3.2.3 Comparison of other encryption methods

Understanding the advantages and disadvantages of homomorphic encryption requires a comparison of it with alternative encryption techniques as part of the research design. This is a quick analogy:

- **Homomorphic Encryption vs Traditional Encryption:** Conventional encryption techniques usually necessitate data decryption prior to conducting any calculations. Conversely, homomorphic encryption maintains privacy during processing by enabling direct computations on encrypted data. The difference between symmetric and homomorphic encryption
- **Homomorphic Encryption vs Symmetric Encryption:** One key is used for both encryption and decryption in symmetric encryption. With its public and private key pair, homomorphic encryption provides extra protection along with the capacity to operate on encrypted data.
- **Homomorphic Encryption vs Asymmetric Encryption:** Although public and private key pairs are used in both asymmetric and homomorphic encryption, homomorphic encryption is distinguished in situations where safe computations without decryption are necessary by its capacity to execute operations on encrypted data.
- **Performance Comparison:** A performance comparison between homomorphic encryption and alternative encryption techniques should

be part of the research design. It is necessary to take into account elements like resource usage, speed, and computational efficiency.

- **Scalability:** A crucial component of encryption techniques is scalability. In comparison to other encryption techniques, the research design should evaluate how effectively homomorphic encryption scales with rising data volumes and processing demands.

3.3 Data collection

The data collected in the experiment is a dummy data collected from various sources to build a comprehensive dataset. the collected data can be utilized for analysis, research, decision making, or any other purpose. in this thesis it is used in the application of homomorphic encryption techniques.

3.3.1 Types of open banking data

Application Programming Interfaces (APIs) are used in open banking systems to facilitate the interchange of financial data between authorized institutions. The kinds of data that are gathered in these kinds of settings are varied and include client information, operational specifics, and different facets of financial transactions. The principal categories of open banking data are as follows:

- **Transaction data** Information regarding user-conducted financial transactions is included in transaction data. This includes information about the date, time, amount, parties involved, and kind of transaction. It is essential to open banking since it gives information about consumer purchasing habits and financial conduct.

- **Account Information** is information about a user's bank account. Account balances, account numbers, account kinds, and information about related financial items are all included in this. Authorized institutions can provide individualized financial services by having access to this data.
- **Customer Profile Data:** Personal information about the account holder, such as name, address, phone number, and identification credentials, is included in customer profile data. Comprehending the customer's profile is essential for customizing services and guaranteeing adherence to regulations.
- **Credit Information:** Credit information is essential in open. This could contain credit limits, credit ratings, and other pertinent credit-related information.
- **Payment Initiation Data:** This category includes information regarding payments that have been started, such as beneficiary details, payment instructions, and confirmation that the payment has been executed. This is essential to enable smooth money transfers.
- **Transaction Metadata:** Transaction metadata is supplementary context-relevant data about transactions that sheds light on their intended use and improves our comprehension of user behavior.
- **Consent and Authorization Data:** Ensuring compliance with data protection standards requires data pertaining to user consent and authorization. It contains documentation of user consents given for data sharing as well as the length of those consents.

3.3.2 Source of Data

Financial institutions, such as banks, credit unions, and other approved participants in the open banking ecosystem, are the main suppliers of open banking data. Important references consist of:

Banks and financial institutions: they serve as the main repositories of open banking data. Account details, transaction histories, and other pertinent financial data are accessible through them.

Third-Party Providers (TPPs): These companies, which include payment service providers, FinTech firms, and other financial technology companies, act as middlemen in the access and use of open banking data. These organizations frequently use APIs to communicate with financial institutions and to compile or offer financial services.

Regulatory Interfaces: Open banking regulatory organizations and frameworks can be useful sources of information about consent management, regulatory reporting, and compliance.

Data Aggregators and APIs: In open banking, APIs are essential for enabling data interchange. As middlemen, data aggregators and API platforms link different financial institutions and outside suppliers.

3.3.3 Considerations of data collection in an open banking environment

Because banking information is sensitive and the regulatory framework is complex, gathering data in an open banking system has specific difficulties that need to be carefully considered. Important things to think about are:

Data security and Encryption: Strong data security measures, such as encryption during transmission and storage, are essential to prevent against unauthorized access and data breaches given the sensitivity of banking data.

Regularity Compliance: Initiatives related to open banking are sub-

ject to strict regulatory frameworks. To guarantee ethical and lawful data management, data gathering procedures must comply with applicable financial regulatory norms and laws like the General Data Protection Regulation (GDPR).

User Consent and privacy: It is essential to get user consent before collecting and sharing data. Transparent permission procedures that let users know and manage how much of their data is shared are essential for open banking services.

Data Quality and Accuracy: The success of open banking services depends critically on accurate and trustworthy data. Institutions and suppliers need to have policies in place to guarantee the consistency, correctness, and completeness of their data.

API security and Standardization: It is crucial that APIs used for data sharing are secure. Standardization of API protocols guarantees security standards and compatibility between various players in the open banking ecosystem.

Data lifecycle Management: Careful planning is necessary to manage the data lifecycle from collection to destruction. Complying with data protection rules can be facilitated by establishing explicit policies on data retention, archiving, and disposal.

Auditing and Monitoring: Finding anomalies or possible security breaches requires routine auditing and monitoring of data access and usage. This guarantees the safe handling of data at every stage of its lifecycle.

Interoperability: For data to exchange seamlessly, different systems and platforms must be compatible. Interoperability between various parties is facilitated by open standards and clearly defined data formats.

Educating Users: Two crucial elements of responsible data collecting are open communication and user education. Consumers should be aware of the security precautions in place, the advantages of open banking, and how

their data will be utilized.

3.4 Experimental setup

deploying the experimental solution, Homomorphic Encryption is known as a new encryption Algorithm technique. the implementation was tested on different platforms and databases, such as; Amazon Web Services (AWS), Oracle MySQL cloud service, and IBM Db2 Database. the above tested environments was not suitable to deploy Homomorphic Encryption as an Encryption solution, because it was limited to use the traditional encryption methods. for that reason, the open source environments was the perfect solution to deploy and test the presented solution.

3.4.1 selection of environment

The implementation of presented technique was done on a Ubuntu virtual machine using Python3 language. For the running cost analysis and representation of data in form of graphs, the MS Excel 365 is used. Downloaded pyseal and followed git instructions to build the library.



Figure 3.1: Installation of package from github

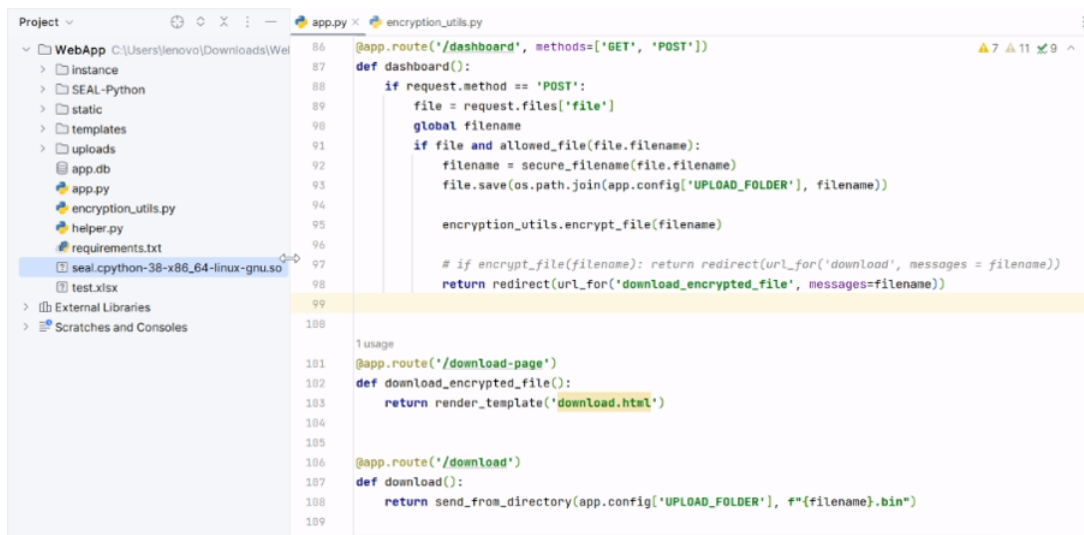


Figure 3.2: Seal on python implementation

3.4.2 Tools and technologies used

For applying homomorphic encryption, for the purpose of testing, after trying several methods; seal was the best solution. used open source operating system. to run seal which is compatible with windows operating system, mac operating system and linux platforms. in order to run SEAL, along

with ubuntu linux, python 3 to run the libraries.

Component Specification

Operating System	Ubuntu 18.04.4 LTS
Memory (RAM)	3.8 GiB
Processor	Intel Core i3-4010U CPU @ 1.70GHz × 3
OS Type	64-bit
Virtualization	VMware Workstation 15.5.6 Pro

Table 3.1: Hardware Specifications

Operating System	Ubuntu 18.04.4 LTS
Language	Python 3.6.9
IDE	PyCharm Community 2020.1
Library	OpenCV 3.2.0
Library	Pycryptodomex 3.9.7
Library	Pyexcel 0.6.1

Table 3.2: Software Specifications

3.4.3 Parameters for experimentation

Getting to start the Encryption method, we need to access the GUI start page, by accessing the local host url.

User Registration In order to generate and manage cryptographic keys in a safe system, the User Registration phase very important. This stage is essential for verifying the user's identity, generating the required cryptographic keys, and guaranteeing the safe start of next processes. Screenshot of the main components of the User Registration phase is given below.

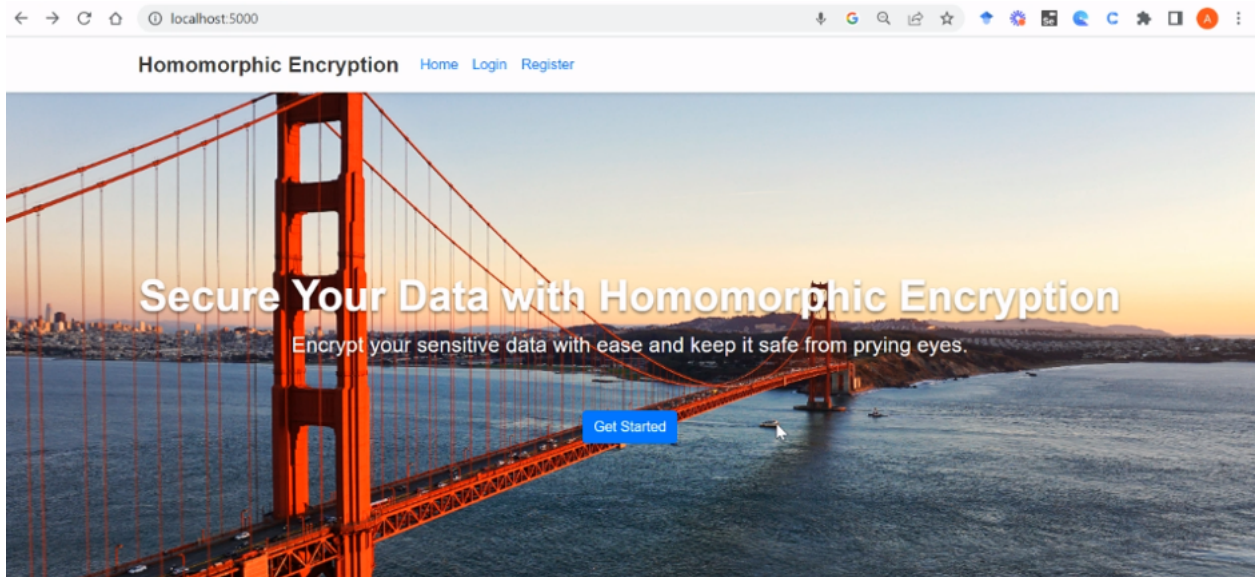


Figure 3.3: Front page GUI

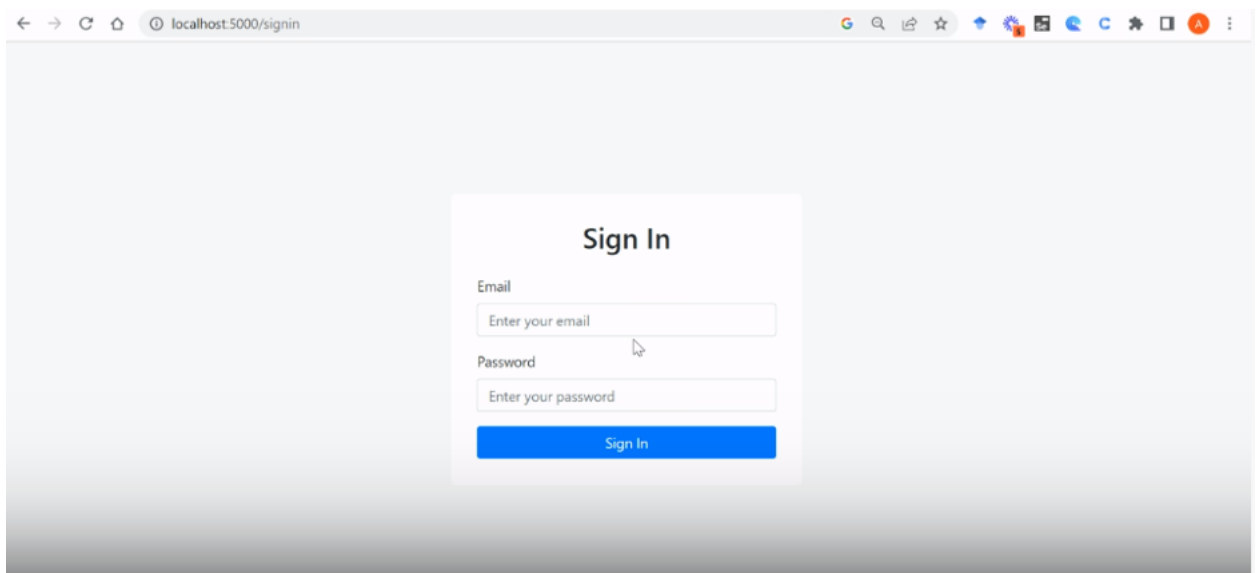


Figure 3.4: Login page

Upload File By choosing the file or files user wish to upload from their local devices to the system, users start the file upload procedure. Depending on how the system is designed, this can be done via a desktop application, online interface, or another method. When user data needs to be safeguarded in a secure system, the file upload and encryption procedure is essential.

Users submit files to the system during this procedure, and the supplied material is then encrypted using cryptographic keys.

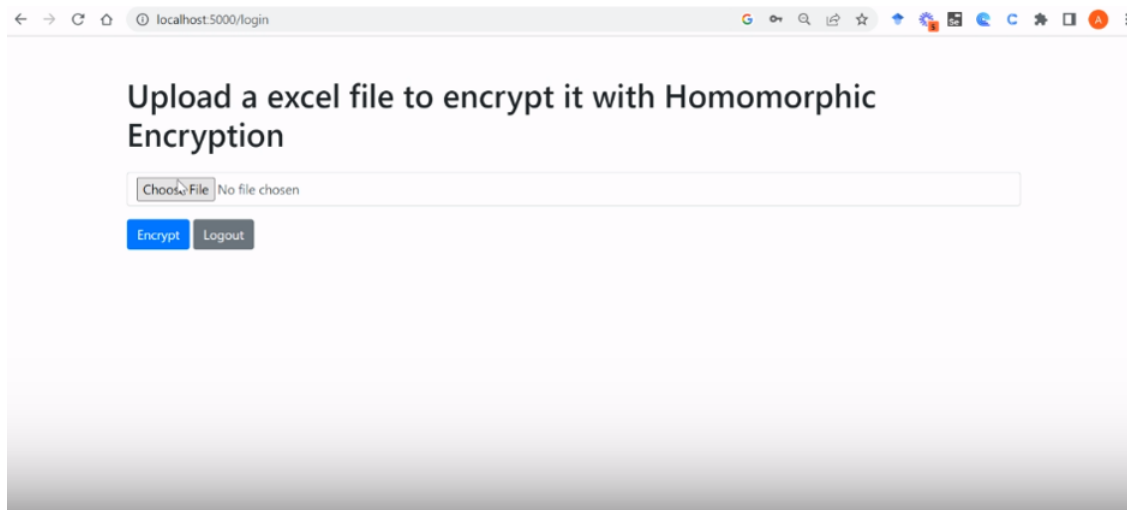


Figure 3.5: File upload page

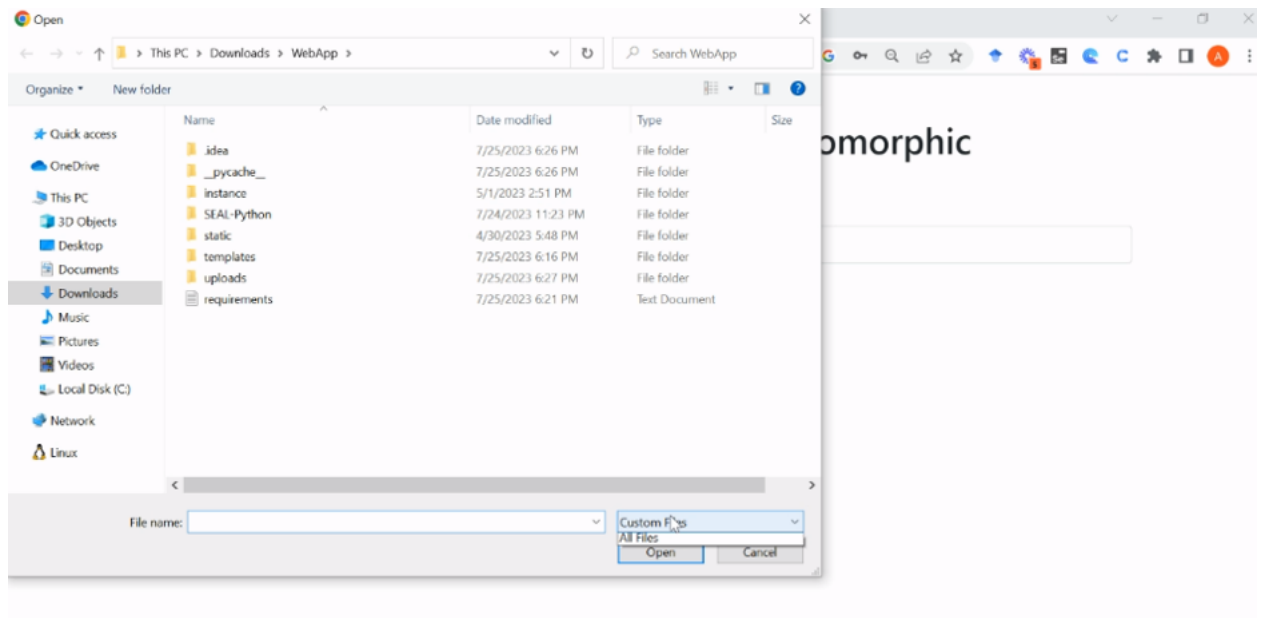


Figure 3.6: User selecting file for encryption

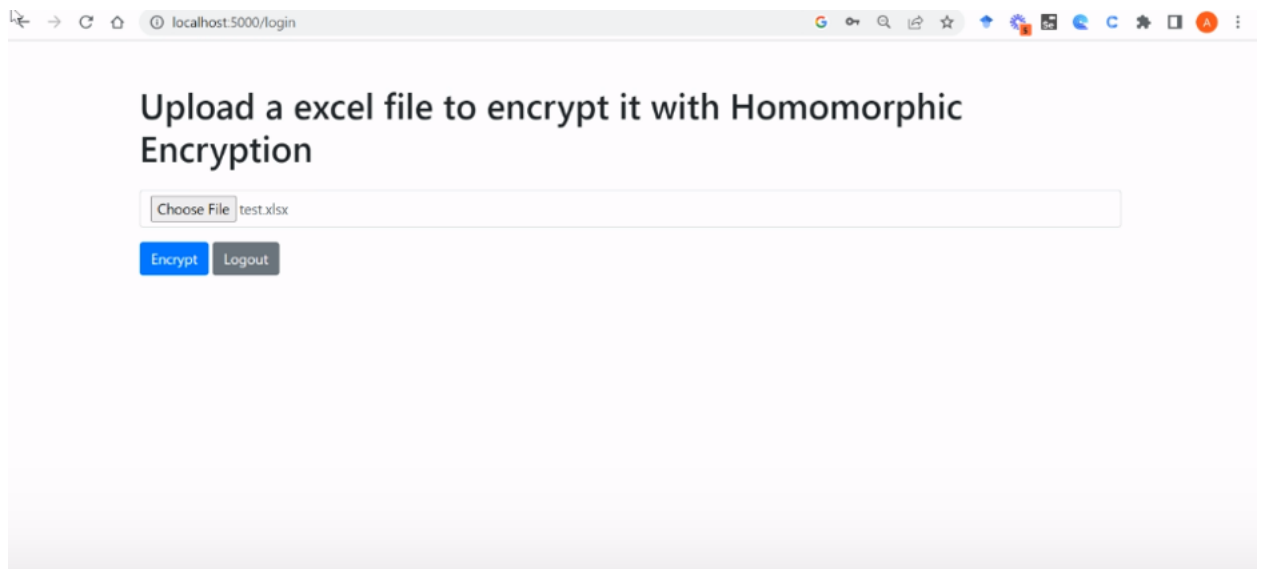


Figure 3.7: Uploading the excel file

Download Encrypted File When a user selects the "Download File" option in a secure system with encrypted files that are protected by cryptographic keys, the encrypted data is retrieved from storage and presented in a downloadable txt file. The main factors and procedures for adding a "Download File" option for encrypted data is for user to be able to view the encrypted data and user it further. First a link is created for a secure download or to start a secure file transfer process. To reduce exposure, this link have a time limit and expire after a certain amount of time.

Encryption process This tool delivers an encrypted file in a text format, which make it easier if you want to share encrypted file to other people. As long as the decryption keys are kept private, encrypted binary data that has been converted to text format doesn't necessarily can be read.

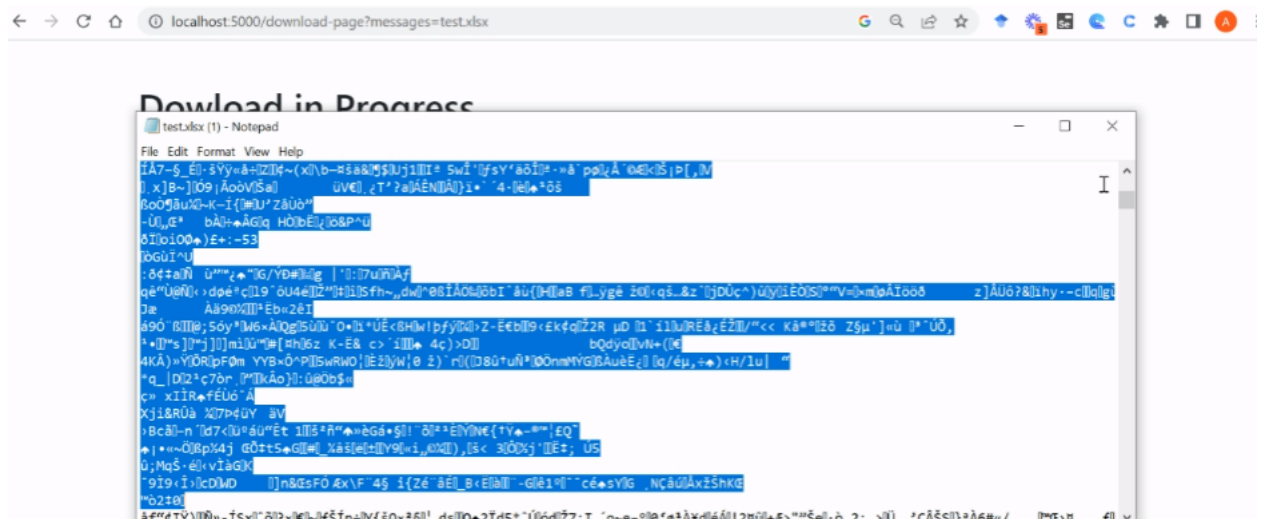


Figure 3.8: Encrypting data by using seal

3.5 Homomorphic encryption implementation

Presented technique uses the banking dataset sample for the analyses and testing purposes, this technique is not limited to the dataset any other such dataset can be used for according to the specifications and requirement of

the client because of the proposed scheme generic nature. The main contribution of presented schemes is as follows:

Implement a privacy-preserving encryption technique that can ensure the user's privacy. Proposed scheme resists Search Pattern leakage attack and traceability issue. Proposed scheme is python based implemented over Linux distribution Ubuntu. Theoretical and experimental analysis shows that the proposed technique is more secure and efficient as compared to state-of-the-art schemes.

Threat model and assumptions:

Threat model To define the threat model, we are considering two entities for a SE scheme:

1. The Cloud Server
2. The owner of the data.

In the proposed scheme, the data is being outsourced. The owner is assumed to be fully trusted and cause no security threat to the system the main concern is with the CS as it acts as an adversary while performing security analysis of the proposed scheme. The CS can launch different successful attacks.

The adversary can have multiple characteristics as:

- Polynomial time adversary:

The polynomial time adversary means that the attacker is limited to execute the polynomials bounded number of operations only in limited time only. These operations can be of encryption, decryption, or any passive attacks.

- standard model:

For the standard model, the Cloud server is not assumed to have unlimited re-sources and time as the ideal adversary or replaced with the random oracle model. A novel privacy preserving technique is presented in this scheme based on complex problems which cannot be solved in a polynomial time and this scheme can provide a high level of security in this standard model.

- Design Goals:

While constructing a scheme, there must be some security and performance goals. These includes but not limited to: probabilistic key, lightweight, and privacy preserving. To resist the distinguish ability attack, each query key must be generated differently. This makes the scheme probabilistic and secure the users from traceability issues. The CS should not know any information about the query key Also, the scheme should be computations friendly and can run at the CS as well as at the user end smoothly.

- Security definitions:

For the proposed scheme, we are considering the definitions of indistinguishability, for query keywords (a.k.a trapdoors) and query index (a.k.a trapdoor index). Chapter 4 shows that the proposed SE scheme complies with the following definitions.

- Keyword-Trapdoor Indistinguishability for SE:

Keyword-Trapdoor Indistinguishability is a process of performing search over cipher text so that the encrypted query keyword should not reveal any relevant data about the unencrypted query keyword. If a same keyword is being search in plaintext repeatedly, the associated cipher

text or trapdoor should not be distinguishable even if an adaptive adversary (with keyword and trapdoor history) is considered. To predict a meaningful and a relevant information of query keyword, the adversary must perform a lot of operations in polynomial time and record the tremendous amount of data.

3.5.1 Description of Homomorphic Encryption scheme

The challenger C generates an encrypted key using the credential provided by the user. The C sends the entries of first row of IT , all object keywords W , and encrypted queries generated by W to the A . The order of appearing keywords is maintained during sharing of keywords with A . The A chooses two keywords which he want to get the encrypted trapdoors i.e. W_0, W_1 and forwards to the C . The C tosses a fair coin b and encrypt the object keyword based on the output of b . This encrypted keyword i.e. trapdoor is sent to the A and the SE scheme provides the trapdoor-index indistinguishability. Here the security parameter λ is considered small enough to be ignored.

Let SE be a searchable scheme with six phases (KeyGen, Build_Obj_Index, Build_Query, Encryp,) with a security parameter, I be the set of images, W be the set of objects, and $A = A_0, A_1$ be the adversaries. Now we are considering to have a probabilistic experimental function $\text{Index_Trap, SE, } A(\lambda)$: Here the probability is dependent over b .

The Privacy Preserving Encrypted Banking Data Scheme is secure if the generated encryption is secure and query object trapdoors are probabilistic. The proposed PPSEI scheme is secure and exhibits the Keyword Trapdoor Indistinguishability and Trapdoor-Index Indistinguishability.

proposed scheme This section presents the Privacy Preserving Search Over Encrypted Images Searchable Encryption Scheme that consists of 6 phases. A detailed explanation and representation of each phase is given

in this section. Following are the Four phases involved in the proposed SE scheme:

1. Key Generation Phase (KeyGen λ Phase)
2. Upload data Phase (upload(I) Phase)
3. Query Generation Phase (Build_query Phase)
4. Encryption Phase (Outcome Phase)

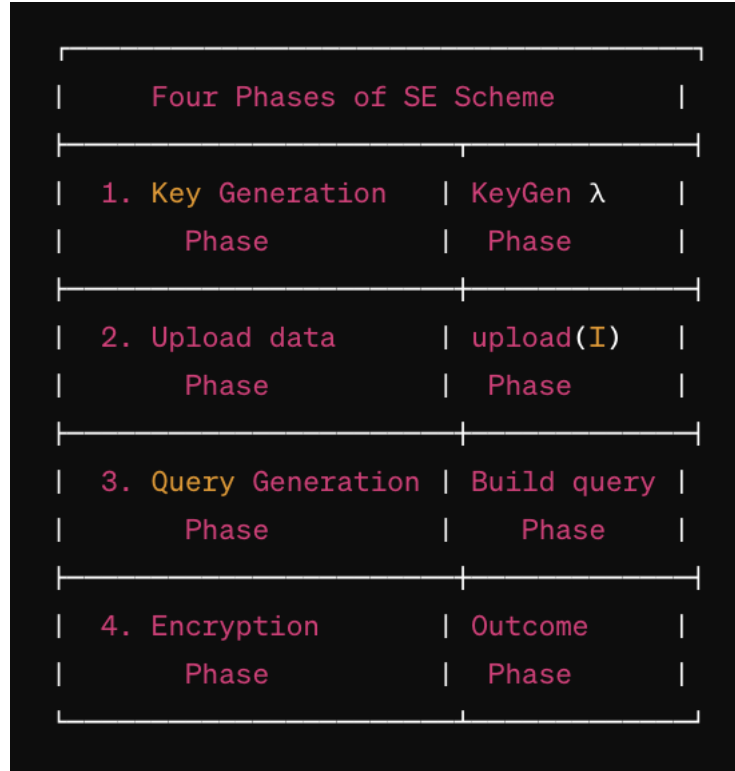


Figure 3.9: SE Proposed Scheme

Scheme Construction These phases are explained in details here.

- Phase 1-[K, ks] $\leftarrow$ KeyGen λ : In key generation phase, master and session keys are generated by giving a security parameter as input argument to the key generation algorithm such that $K, ks \leftarrow 0,1 \lambda$.

- Phase 2- $W \leftarrow$ upload(I): If the data owner want to add more data to the database, this phase will be repeated each time. Manual entries can also

be done by the data owner. In case of manual entries, this phase will not be processed.

- Phase 3-QW $\leftarrow$ nBuild _Query(K, ks ,W,num): For the query generation, following steps are performed. num is the number of images desired by the user.

- Compute $a = HK(W)$

- Compute $b = \text{Encks}(W)$

- Phase 4-A[] $\leftarrow$ Outcome (IT,QW):

In the searching phase, following operations are performed. Later, the output is stored in the array A[] as search outcomes.

- Initialize a dynamic 2D array A[].

- For $1 \leq y \leq \text{number of columns in IT}$

1. Set $a = IT[1][y]$;

2. if $(d == H(a \oplus c))$:

- (a) For $1 \leq r \leq \text{num}$

- * Find the highest RF, return $\text{EncK}(\text{id}(I))$;

- A[] $\leftarrow \text{EncK}(\text{id}(I_i))$ and return A to the client.

3.5.2 Step-by-step implementation process

This sections gives the detailed analysis of each phase involved in proposed scheme.

KeyGen Phase:

In this phase, a master and session keys are generated by the user. The input argument of this phase include a security parameter λ and produces two keys; a session key ks as $ks \in \{0,1\}^\lambda$ and a master key K as $K \in \{0,1\}^\lambda$. Both master and session keys are to be kept secret at the user end and there is no need to share any key with the CS. As, at the CS end, all operations are

being performed without master and/or session key. Master key generation is a deterministic function while session key generation is a probabilistic algorithm. Both phases are executed by the data owner.

Upload Phase:

The data owner executes this phase which is deterministic in nature. User upload data at the CS.

Build Query Phase

Another parameter c is computed by computing the XOR operation of parameters a and b . Then, another parameter d is computed by taking a simple cryptographic hash of parameter b . The computed parameters are then set to a user query parameter QW which contains the parameters c, d . This query parameter QW is then sent to the CS. The data owner executes this phase each time when a query is made. This phase is probabilistic in nature.

Encryption Phase

When the CS receives the query request from the user containing three parameters, the CS returns the encrypted data and download the file. The CS executes this phase which is deterministic in nature.

3.5.3 Challenges encountered during implementation

- **Key generation:**

The randomness source for session key generation needs to be carefully considered when implementing a safe key generation procedure. It is difficult to make sure the security parameter is used efficiently without sacrificing effectiveness. Implementation issues include handling and safely storing produced keys.

- Uploading data:

When working with a variety of data kinds and formats, it may be difficult to guarantee the determinism of the upload process. For effective deployment, potential constraints in data transfer capacity and speed must be addressed.

- Building query:

It might be difficult to control and handle the probabilistic elements of query generation. Careful implementation is needed to guarantee that cryptographic hash functions are resistant to assaults and generate dependable query parameters across a range of query inputs.

Chapter 4

Results and Findings

4.1 Security analysis

4.1.1 Evaluation of security risks in open banking

Security Evaluation of Proposed Scheme

In this section we will evaluate the presented technique data leakage profiles. In best case scenario presented system would not expose any type of details that would compromise privacy protection, outsource the information, or statistics data which would reveal the query generated information performed over the users details. When it comes to security of a cryptographic system, it is considered safe if it protects the individual's privacy and resist the pattern and access pattern attacks. Presented encryption provides security against deterministic pattern based attacks.

The size of data exposed to the external threat is assessed by the information leakage profile. This data might be secured with the encryption or plaintext,

and it can be valuable or irrelevant.

Open banking offers a number of security issues that should be carefully understand these threats in order to put in place efficient security measures considered, even as it promotes convenience and creativity. It is essential to. In open banking, a few major security issues are as follows:

- Unauthorized access:

There is a big risk of unauthorized access to private banking information. Malicious actors may obtain access to user accounts and financial data if authentication procedures are breached.

- Data Breach:

Data breaches are more likely to occur as a result of the extensive gathering and exchange of banking data. Transaction details, private customer information, and other information may become public knowledge as a result of breaches.

- Fraud and Identity Theft:

Open banking's interconnected system makes fraud and identity theft easy. Vulnerabilities could be used by malicious organizations to carry out fraudulent transactions or steal sensitive data.

- API Vulnerabilities:

Since APIs are essential to open banking, they can be used to gain unauthorized access, manipulate data, or cause a denial-of-service attack.

- Insufficient Encryption:

Sensitive data might be exposed to interception or unauthorized access during transmission and storage due to inadequate or badly applied encryption.

- **Lack of User Awareness:**

Users might not be completely aware of the security precautions of open banking. Ignorance might result in dangerous actions like giving login information or becoming a target of social engineering scams.

- **Third Parties Risks:**

There are extra hazards when third-party sources are involved. These suppliers' security protocols have the potential to jeopardize the ecosystem's overall security for open banking.

4.1.2 How Homomorphic encryption addresses security concerns

In open banking, homomorphic encryption is essential for addressing a number of security issues.

Confidentiality of Data:

Computations can be done on encrypted data without having to first decrypt it because of homomorphic encryption. This guarantees the confidentiality of sensitive financial data throughout processes, such as data analysis and query processing.

Secure Query Processing:

The Cloud Service (CS) may handle user requests without access to the

unencrypted data because of the usage of homomorphic encryption in the Query phase. By doing this, we can improve query processing security and guard against unwanted access to user inquiries.

Pattern Leakage Prevention:

Because homomorphic encryption permits actions on encrypted data, it guards against pattern leaking attacks. This improves privacy protection by guaranteeing that the CS cannot determine the true content even when queries contain sensitive patterns.

Data-in-Transit and Data-at-Rest Security:

By guaranteeing the confidentiality of data both during transmission (data-in-transit) and during storage (data-at-rest), homomorphic encryption enhances the overall security posture. This takes care of issues with eavesdropping and unwanted access.

Privacy Preservation:

Homomorphic encryption protects user privacy by permitting calculations on encrypted data. In order to reduce the possibility of data misuse, it makes sure that not even service providers and other third parties engaged in open banking activities have access to the unencrypted data.

Secure Key Generation:

Secure key management is facilitated by the probabilistic algorithm used for session key generation and the deterministic nature of master key production. This improves the open banking system's overall security.

4.1.3 Analysis of potential threats and vulnerabilities

In an open banking system, various threats and weaknesses persist even with the security advantages offered by homomorphic encryption:

Key management risks:

Strong key management is essential to the homomorphic encryption scheme's security. Threats like weak key protection or compromised keys could jeopardize encrypted data's secrecy.

Implementation flaws:

Vulnerabilities may arise from improper application of homomorphic encryption methods. Exploitable weaknesses may result from configuration mistakes, coding flaws, or insufficient testing.

Side-channel attack:

Information could be extracted from the encryption process by using side-channel attacks like power analysis or timing attacks. It is crucial to implement safeguards against side-channel assaults.

Homomorphic encryption overhead:

The performance of open banking operations may be impacted by the computations related to homomorphic encryption. In order to guarantee a Successful user experience, security and efficiency must be balanced.

Social engineering and phishing:

Although homomorphic encryption takes care of technological security issues, user behavior is still a possible weak point. Phishing and social engineering techniques have the ability to trick individuals into disclosing login credentials or private information.

Regulatory compliance:

Maintaining adherence to changing data protection laws is a constant struggle. New compliance risks could be brought about by modifications to regulatory standards or interpretations.

Insider threats:

Insiders who have access to confidential data or systems could be dangerous. To reduce insider risks, strict access controls and monitoring systems must be put in place.

API security:

One major worry that still exists with APIs is security. Unauthorized parties may be able to access sensitive data due to poor access controls or insecure API endpoints.

4.2 performance evaluation

4.2.1 Metrics of security risks in open banking

As discussed, the proposed technique is based on probabilistic generation of key keyword. The key generation involves the encryption and hashing of query keywords that is employed with random number and AES. The AES input parameters including the user's private key, initialization vector. By keeping the iv and key random each time, the generated keyword is random. To test the probabilistic encrypted experimentally, we have generated trapdoors for keyword "person". Due to the limitation of storage, only few rows were allowed to write in excel file. We applied filter on duplicate values

in excel file and found that there is no repeated query trapdoor was found. These results shows that the generated query trapdoors are random even if the same keyword is being searched multiple times.

4.2.2 Comparison of encryption overheads

This section compares and evaluates the effectiveness of the presented approach with the already existing schemes discussed previously in literature review chapter. searchable encryption schemes over encrypted banking data explained in literature review lacks in the security against traceability issues and most these techniques does not offer the sufficient protection to the user data because of the their deterministic nature.

Proposed searchable homomorphic cloud based encryption provides security against the pattern attacks to secure the data privacy of the client and is probabilistic in nature to resists the security attacks. Due to the involvement of the hashed key value in the phase where query keyword is generated and then the same keyword is encrypted with the encryption which of indeterministic nature. This randomization task is achieved by the using the AES Advanced standard encryption. Creating the key generated at the user end random each time even if the data is identical different key is generated every time. This hides the search pattern of the encryption and makes the encryption effectively resistant against the attacks which rely on the traceability issues. This function protects the use's data secrecy and allows key without being traced back.

4.2.3 Assessing computational efficiency and speed

This chapter discusses the performance and computational analysis of the proposed technique. The performance analysis is divided in different parts i.e. first the algorithmic analysis. Then the storage overhead and computa-

tional analysis is given in detail. The implementation of proposed scheme, system and dataset specifications are given in computation analysis section. Furthermore, the implementation is done in Python programming language and pseudo codes are given. The computational overhead of each phase of the proposed scheme is given in details.

1. Trapdoor-Index Indistinguishability for SE

The complexity of an SE scheme is measured by the indistinguishability of trapdoor indexes. This means that the generated query trapdoors are random in such a way that no information is disclosed to the adversaries about keyword or index. The indistinguishability should remain in contact with the keyword even if the same keyword is searched repeatedly. The generated query trapdoor must preserve the users privacy and keyword security in terms of distinguishability and linkability with history (keyword, trapdoor, index) in adaptive adversarial model. Also, by changing a single bit or character in query keyword, the resultant Trapdoor should be completely changed and same is true for reverse.

2. Algorithmic Analysis

To check the performance of the proposed scheme, multiple analysis are presented and discussed. The complexity is based on the data present in collection. The complexity analysis of these phases are given as follows:

The complexity of the schemes are denoted by $O()$, read as "big oh". This is called the asymptotic upper bound. It tells the time complexity required to run the code when the size and number of input parameters increases. It is a relationship among input parameters and the required time to process those input parameters. In the case of

proposed scheme, KeyGen phase are fairly constant functions and requires the same amount of time. Therefore, the time complexity for KeyGen $O(1)$. The initialization phase is linear and independent from number of input. The object detection function's complexity depends on the loops used. For n number of data, the algorithm takes almost linear time to detect objects. Therefore, its complexity is $O(n)$. Key generation algorithm involves the AES encryption and Hash functions which gives the linear time complexity. In conclusion, if the number of data increased, time complexity increases linearly. Therefore, the complexity of key generation phase is $O(n)$. In the key generation function, the function uses Hash functions and one AES encryption function along with one XOR function. The time complexity for each is constant individually. The time complexity will be constant i.e. $O(2h+e)$. Therefore, we can conclude that the time complexity for this function is $O(n)$, here "n" denotes the number of keywords. Higher the number of query keywords, more time it will take to process each query.

3. Storage

The proposed technique presented in is consists of 4 phases. The Key-Gen phase generated 2 keys i.e. the masker key K and a session key k_s . Session key is generated at run time and the user do not need to store session key. Only master key is stored which will of 265 bit. So, the client stores $(256 \text{ bits}) / 8 = 32 \text{ Bytes}$. The random numbers through CSPRNGs are generated and stored at the user end. For the testing purposes, we have generated 180 random numbers and stored in an .xls file. The size of random number file is 5,500 Bytes. In case of a.txt file, this space is only 962 Bytes. At the CS side, the CS stores encrypted index table IT and encrypted data in the database. Let we

have the average size of an encrypted data as I_{avg} and n number of images, then the storage overhead for images would be $n.I_{avg}$. The storage overhead can be calculated as $x(mn) + n.I_{avg}$ at the CS end. Where m and n are rows and cols and x is the number of total data.

4. Computational analysis:

The proposed technique is also tested and analyzed based on the implementation and testing. This section gives the implementation and performance analysis in details. The scheme is implemented in Python language and tested on banking dataset. Before going in to the details of the computation cost analysis, some preliminary details about system specifications and image dataset information are given. Proposed schemes is implemented in python for the analyses on Linux distribution Ubuntu. All algorithms of proposed scheme, are implemented at the client machine. For the testing purpose open source banking dataset is used.

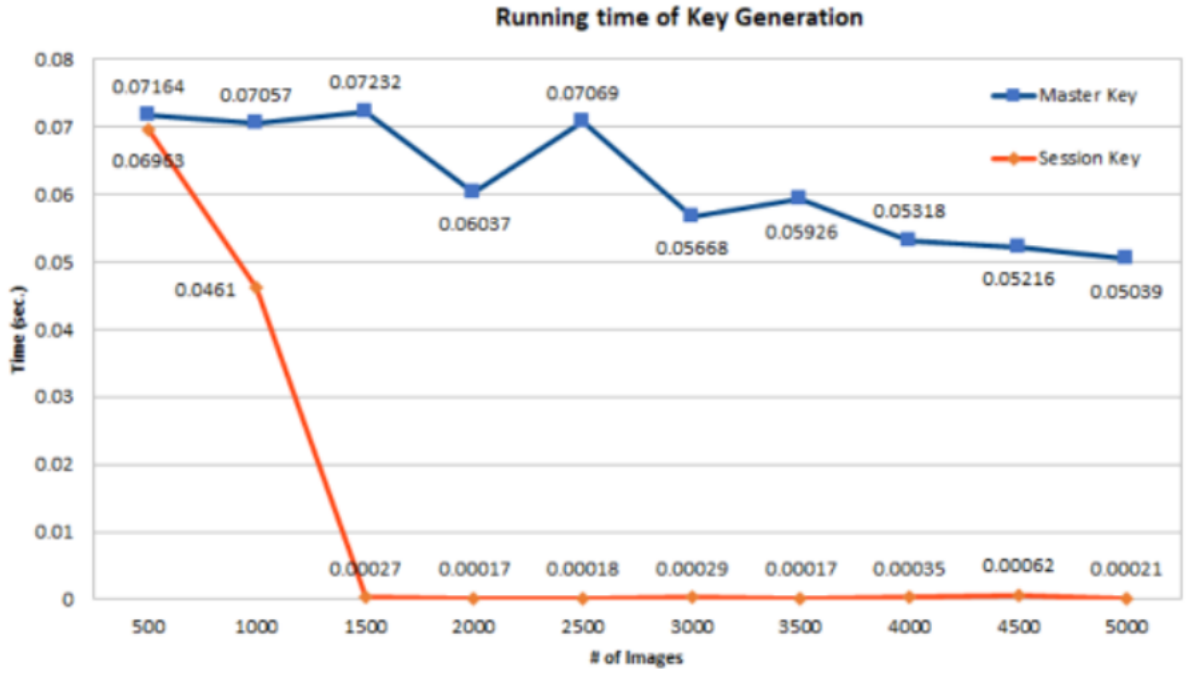


Figure 4.1: Key generaion time

Figure 3.10 shows the running time of both key generation functions. The session key generation function takes higher time at start but time decreases gradually when the library is properly initialized and loaded in memory. The master key generation function takes almost the same amount of time for key generation each time. It is evident that the key generation phase is almost constant and independent of number of input images. The master key takes more time than session key as shown in Figure.

This was a discussion about the performance analysis in terms of computational and running time complexity of the proposed homomorphic technique. The algorithmic analysis tells the time complexity required to process each phase of the proposed scheme. The storage overhead gives the analysis of required space to store security keys, random

numbers and encrypted files. The computational analysis shows the system specifications, dataset specifications and the implementation details. In section below, we have given the conclusion and future directions of the research. Some challenges faced during this research are also discussed in next chapter.

4.3 Testing and validation

4.3.1 Testing scenarios and use cases

Techniques and schemes discussed in literature review follows the deterministic nature and not secure in terms of the search pattern attacks, randomization keyword is also one of the problem which leads to the data leakage exposing the user data. The encryption can resist traceability and pattern attacks. This schemes analyzed with respect to the security point view on the basis of the different data leakage profiles. Then the formal security analyses is conducted in this chapter, reviewing the keyword trapdoor and trapdoor index indistinguishability.

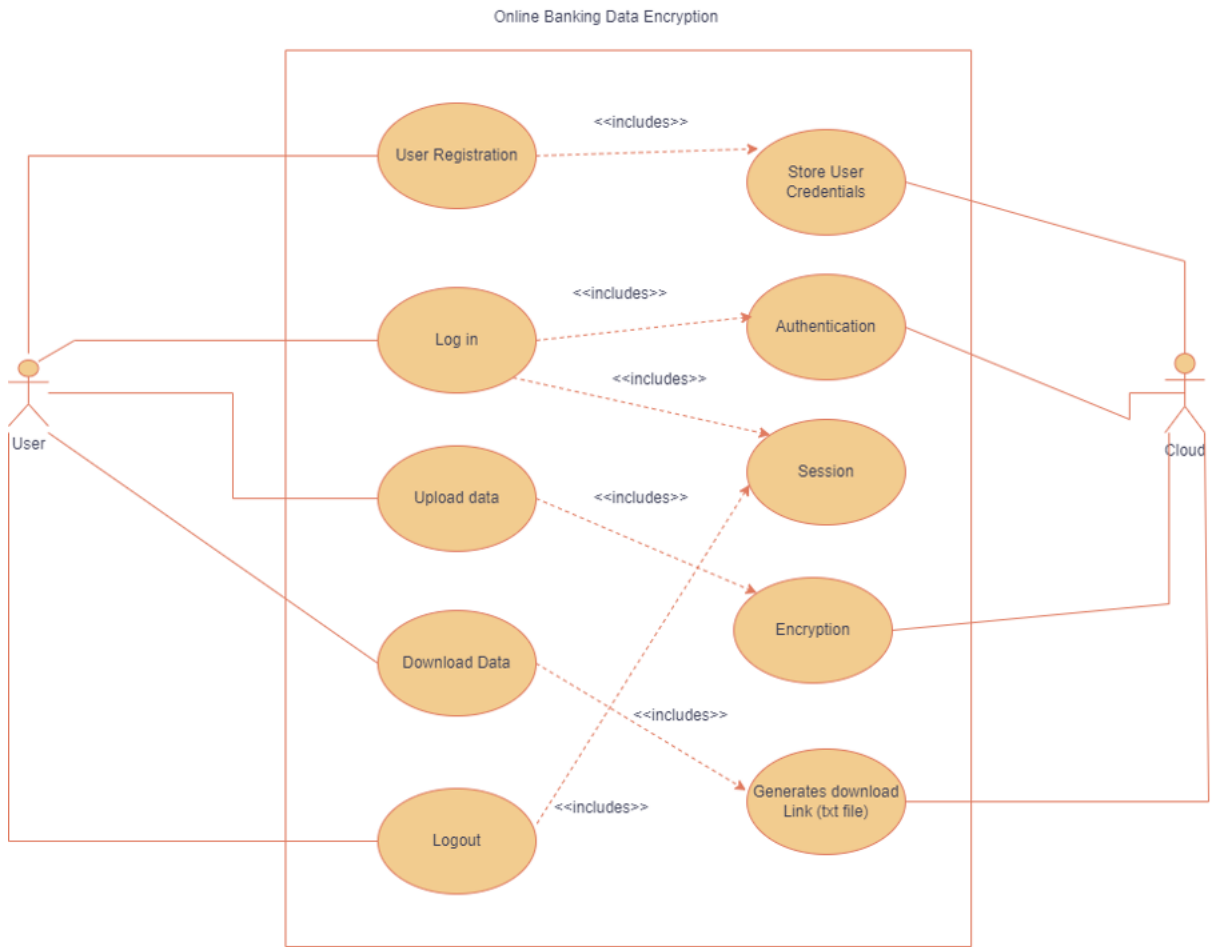


Figure 4.2: Use case chart

1. User Registration:

Description: Gives the user the option to register up a new account in the encrypted online cloud banking facility.

Actor: Unregistered User, Guest

Sequence of Events:

- The user enters the necessary registration information.
- The data is verified by the system.
- The system generates a new user account if it is valid.

Exceptions: The system indicates an error if the data entered is inaccurate or incomplete.

2. Log in:

Description: Enables a user who has registered to access the system.

Actor: Registered user

Sequence of Events:

- The user inputs their password and username.
- Credentials are verified by the system.
- If verified, the user is granted entry to the system.

Exceptions:

An authentication failure notice is displayed by the system if the credentials are entered incorrectly.

3. Log out:

Description: Enables a user who is currently logged in to log out of the system.

Actor: Registered user

Sequence of Events:

- The user starts the logout procedure.
- The user's session is terminated by the system.

4. **Encryption process:**

Description: Gives the user the ability to encrypt banking information.

Actor: Registered user

Sequence of Events:

- The encryption option is chosen by the user.
- The user is prompted to upload a document via the system.
- The document is encrypted by the system using a safe algorithm.

Exceptions: The system displays an error message if the encryption procedure is unsuccessful.

5. **Upload a DOCX file:**

Description: Uploading a DOCX file to the cloud storage is done by this feature.

Actor: Registered user

Sequence of Events:

- The user decides to upload a DOCX file.
- The user is prompted by the system to choose the file.
- The uploaded DOCX file is safely stored by the system.

6. Download file:

Description: Gives the user the ability to download a TXT-formatted encrypted document.

Actor: Registered user

Sequence of Events:

- The user decides to download a TXT file.
- The user is select to download file.
- After the encryption, the system offers a link for a safe download.
- The document is encrypted and then converted by the system to TXT format for download.

Exceptions: The system shows an error message if the document cannot be found or if the conversion is unsuccessful.

4.3.2 Validation of encrypted data operations

Leakage Profiles

Leakage profile defines the amount of information leaked to the adversary. This information can be encrypted or unencrypted, significant or insignifi-

cant. We will analyze the information obtained from the six polynomial time algorithms of proposed SE scheme to the CS. The adversary A can launch any possible attack in standard model. This is because we are not restricting A to replace the technique with different weak construction. The only restriction applies to the time of execution of different process and steps by the A i.e. the A is restricted to polynomial.

4.3.3 Results and findings

In the proposed scheme, if somehow the attacker has access to the key generation process accidentally, the queries generated in future with this function are still secure. This is because each generated key is independent. Encryption do not leak any data related to outsourced data to the cloud. The assumptions and leakages discussed here are interrelated and interdependent to each other's. Therefore, to achieve the best level of security, it is assumed that all the security assumptions are followed strictly.

Chapter 5

Conclusions and recommendations

5.1 Summary of methodology

in summary, homomorphic encryption stands out as a viable solution to tackle pressing issues to secure data processing and the protection of sensitive financial and personal data. with its capacity to execute computations on encrypted data without requiring of decryption, the provided solution offers resilient framework for securing sensitive information across diverse sectors such as cloud computing and finance. Following swift enhancement in technologies concerning cloud computing, machine learning, and big data, clients are relying on outsourcing their data to the cloud storage.

Cloud storage solves the problem of the computational overhead but comes with the security issues which could compromise the integrity and confidentiality of the data. Encryption over data is proposed to mitigate this problem and provide protection to data being processed or saved over cloud server. However the present schemes are not secure enough to provide full security

of the data. Especially, while retrieving from the CSP, user sends the query request to the CSP. These queries are not well protected and/or randomized. Therefore, they are prone to traceability issues and do not provide security from search pattern leakage attacks. We proposed the encryption in this thesis which provides the secure encryption algorithm and protects user's privacy against patterns attacks. Techniques analysis shows that the presented technique is more secure and efficient than other already discussed schemes. In this section, we have presented an overview of our research, summary of contributions, and discuss some challenges and future directions.

5.2 key findings and conclusion

While open banking is build to share different financial and personal data from different financial institutions, providing security for both of financial institutions and consumers has been a big issue, the solution provided in this thesis is to use a new type of encryption; which is homomorphic encryption. this type of encryption works different than the traditional encryption methods. the main idea is to conduct computations on the encrypted data whiteout the need for decryption. the different types of data which is shared in a third party application will remain secure even if the data are hacked or leaked, while in transit or at rest, because it both ways the data will remain encrypted.

Real-time implementation of Homomorphic encryption might introduce both performance implications and scalability challenges. while homomorphic encryption enhances data security and privacy, it comes with computational overhead that might impact the speed and scalability of financial transactions.

The research presented in this thesis discussed the security issues associated with currently available techniques in the domain of encrypted banking data. From the literature review presented in Chapter 2, it is clear that the current techniques are prone to search pattern leakage attacks that leads to the user traceability issue. As a results the user privacy is at risk. We have scheme that preserves the user's privacy by generating the indeterministic/probabilistic key as discussed in Chapter 3. The security analysis presents that the proposed technique is secure in terms of leakage profiles and provides security in terms of indistinguishability. The performance analysis, clearly shows that the technique is efficient and can be utilized in real word applications.

5.3 Recommendations and future work

In the rapidly evolving landscape of financial technology, the integration of advanced cryptographic techniques, such as homomorphic encryption, holds significant promise for fortifying the security and privacy of real-time financial transactions within the realm of open banking. However, the adoption of homomorphic encryption introduces computational complexities that necessitate careful consideration. As computational overhead becomes a focal point, it poses potential challenges to the speed and efficiency of financial transactions, thereby prompting the need for strategic recommendations and avenues for future research.

some of the challenges:

Computational Overhead

Homomorphic encryption introduces significant computational overhead, as these computations need to be performed on encrypted data. this can impact the speed of the data processing and impact the speed

and efficiency of financial transactions, potentially leading to some delays on the system.

Complexity of integration

when integrating Homomorphic encryption to existing banking systems, can involve legacy infrastructure and diverse technology stacks, which can be a complex while implemented. Ensuring seamless comparability without disrupting the existing used workflow is a substantial challenge.

Scalability

the volume of financial transactions is increasing, so the scalability of homomorphic encryption becoming a concern. to make sure that the encryption can handle the growing users and transaction is a significant challenge.

This discussion delves into recommendations aimed at mitigating the challenges posed by homomorphic encryption for future work, encompassing algorithmic optimizations, the exploration of hardware acceleration, and considerations of hybrid encryption approaches. Furthermore, it underscores the importance of continuous monitoring, user education, and transparent communication to foster user acceptance in the face of potential delays. Looking ahead, the exploration of post-quantum homomorphic encryption, standardization initiatives, and the integration with emerging technologies like blockchain are identified as key areas for future research.

References

- Abhinaya, B., & Santhi, S. (2021). A survey on genomic data by privacy-preserving techniques perspective. *Computational Biology and Chemistry*, 93, 107538.
- Acar, A., Aksu, H., & Uluagac, M., A. Selcuk and Conti. (2018). A survey on homomorphic encryption schemes: Theory and implementation. , 51(4). Retrieved from <https://doi.org/10.1145/3214303> doi: 10.1145/3214303
- Al Nafea, R., & Almaiah, M. A. (2021). Cyber security threats in cloud: Literature review. In *2021 international conference on information technology (icit)* (pp. 779–786).
- Alowolodu, O., Alese, B., Adetunmbi, A., Adewale, O., & Ogundele, O. (2013). Elliptic curve cryptography for securing cloud computing applications. *International Journal of Computer Applications*, 66(23).
- Armknacht, F., Boyd, C., Carr, C., Gjøsteen, K., Jäschke, A., Reuter, C. A., & Strand, M. (2015). A guide to fully homomorphic encryption. *Cryptology ePrint Archive*.
- Bahrami, M. (2015). Cloud computing for emerging mobile cloud apps. In *2015 3rd ieee international conference on mobile cloud computing, services, and engineering* (pp. 4–5).
- Behbehani, D., Rajarajan, M., Komninos, N., & Al-Begain, K. (2022). Detecting open banking api security threats using bayesian attack graphs. In *2022 14th international conference on computational intelligence*

- and communication networks (cicn)* (pp. 789–796).
- Bouti, A., & Keller, J. (2012). Securing cloud-based computations against malicious providers. , 46(2). Retrieved from <https://doi.org/10.1145/2331576.2331583> doi: 10.1145/2331576.2331583
- cag, D. (2022, Jan). *The rise of open banking: How big data is changing fintech*. Retrieved from <https://fintechmagazine.com/financial-services-finserv/rise-open-banking-how-big-data-changing-fintech>
- Campanile, B. (n.d.). Robert fano, an italian computer scientist from project mac to the internet. *ComPuter network Histories*.
- Carter, W. (2017). Forces shaping the cyber threat landscape for financial institutions.
- Cbb launches the bahrain open banking framework: Cbb*. (n.d.). Retrieved from <https://www.cbb.gov.bh/media-center/cbb-launches-the-bahrain-open-banking-framework/>
- Chase, M., Chen, H., Ding, J., Goldwasser, S., Gorbunov, S., Hoffstein, J., ... others (2017). Security of homomorphic encryption. *HomomorphicEncryption.org, Redmond WA, Tech. Rep*.
- Choo, K.-K. R. (2011). Cyber threat landscape faced by financial and insurance industry. *Trends and Issues in Crime and Criminal Justice [electronic resource]*(408), 1–6.
- Dixit, P., Gupta, A. K., & Trivedi, V. K., Munesh Chandra andYadav. (2018). Traditional and hybrid encryption techniques: a survey. In *Networking communication and data knowledge engineering* (pp. 239–248). Springer.
- Dowlin, N., Gilad-Bachrach, R., Laine, K., Lauter, K., Naehrig, M., & Wernsing, J. (2017). Manual for using homomorphic encryption for bioinformatics. *Proceedings of the IEEE*, 105(3), 552–567.
- Drozdowski, P., Buchmann, N., Rathgeb, C., Margraf, M., & Busch, C.

- (2019). On the application of homomorphic encryption to face identification. In *2019 international conference of the biometrics special interest group (biosig)* (pp. 1–5).
- Ducas, E., & Wilner, A. (2017, December). The security and financial implications of blockchain technologies: Regulating emerging technologies in canada. *International Journal: Canada's Journal of Global Policy Analysis*, 72(4), 538–562. Retrieved from <https://doi.org/10.1177/0020702017741909> doi: 10.1177/0020702017741909
- El-Yahyaoui, A., & Ech-Chrif El Kettani, M. D. (2017). A verifiable fully homomorphic encryption scheme to secure big data in cloud computing. In *2017 international conference on wireless networks and mobile communications (wincom)* (p. 1-5). doi: 10.1109/WINCOM.2017.8238186
- Halevi, S. (2017). Homomorphic encryption. In *Tutorials on the foundations of cryptography* (pp. 219–276). Springer.
- Hani, Q. B., & Dichter, J. P. (2016). Data leakage prevention using homomorphic encryption in cloud computing. In *2016 IEEE Long Island Systems, Applications and Technology Conference (LISAT)* (pp. 1–5).
- Hernández, E., Öztürk, M., Sittón, I., & Rodríguez, S. (2019). Data protection on fintech platforms. In *International conference on practical applications of agents and multi-agent systems* (pp. 223–233).
- Hornuf, L., Klus, M. F., Lohwasser, T. S., & Schwienbacher, A. (n.d.). How do banks interact with fintech startups? *Small Business Economics*, 57(3), 1505–1526.
- Huang, H.-L., Zhao, Y.-W., Li, T., Li, F.-G., Fu, Y.-T. D. X.-Q., Zhang, S., ... Bao, W.-S. (2016, December). Homomorphic encryption experiments on IBM's cloud quantum computing platform. *Frontiers of Physics*, 12(1). Retrieved from <https://doi.org/10.1007/s11467-016-0643-9> doi: 10.1007/s11467-016-0643-9

- Jabbar, I., & Alsaad, S. N. (2017). Design and implementation of secure remote e-voting system using homomorphic encryption. *Int. J. Netw. Secur.*, 19(5), 694–703.
- Jain, P., Gyanchandani, M., & Khare, N. (2016). Big data privacy: a technological perspective and review. *Journal of Big Data*, 3(1), 1–25.
- Jeng, L. (Ed.). (2022). *Open banking*. Oxford University Press. Retrieved from <https://doi.org/10.1093/oso/9780197582879.001.0001> doi: 10.1093/oso/9780197582879.001.0001
- Kaipainen, L. (2017). Effects of psd2 on security architecture of mobile banking: a review of literature.
- Kassab, M., & Laplante, P. A. (2022). Open banking: What it is, where it's at, and where it's going. *Computer*, 55(1), 53–63.
- Khan, I. A., & Qazi, R. (2019). Data security in cloud computing using elliptic curve cryptography. *International Journal of Computing and Communication Networks*, 1(1), 46–52.
- Kogos, K. G., Filippova, K. S., & Epishkina, A. V. (2017). Fully homomorphic encryption schemes: The state of the art. In *2017 ieee conference of russian young researchers in electrical and electronic engineering (eiconrus)* (pp. 463–466).
- Kurze, C., Schopp, M., & Moxon, P. (2018). Data virtualization: a standardized front door to company-wide data opens the way for (digital) business success. In *Digital marketplaces unleashed* (pp. 699–709). Springer.
- Ma, S., Guo, C., Wang, H., Xiao, H., Xu, B., Dai, H.-N., ... Wang, T. (2018). Nudging data privacy management of open banking based on blockchain. In *2018 15th international symposium on pervasive systems, algorithms and networks (i-span)* (p. 72-79). doi: 10.1109/I-SPAN.2018.00021
- Marston, S., Li, Z., Bandyopadhyay, S., Zhang, J., & Ghalsasi, A. (2011).

- Cloud computing—the business perspective. *Decision support systems*, 51(1), 176–189.
- Marzouk, E. (2021). Consumer adoption of open banking products and services: The influence psychological ownership of data on consumer adoption of open banking products.
- Mendes, R., & Vilela, J. P. (2017). Privacy-preserving data mining: methods, metrics, and applications. *IEEE Access*, 5, 10562–10582.
- Meneses, F., Fuertes, W., Sancho, J., Salvador, S., Flores, D., Aules, H., . . . Nuela, D. (2016). Rsa encryption algorithm optimization to improve performance and security level of network messages. *IJCSNS*, 16(8), 55.
- Millstein, M. L. R. M. T., & Tetali, S. D. (n.d.). Mrcrypt: Static analysis for secure cloud computations.
- Mohammed, B. (2019, 12). Api “application programming interface” banking: A promising future for financial institutions (international experience). , 18, 31-43.
- Mohan, M., Devi, M. K., & Prakash, V. J. (2017). Homomorphic encryption-state of the art. In *2017 international conference on intelligent computing and control (i2c2)* (pp. 1–6).
- Not, F. O. (2018). Open banking.
- Opara, C. (2019). Cloud computing in amazon web services, microsoft windows azure, google app engine and ibm cloud platforms: A comparative study. *Diss. Near East University*.
- Papisetty, S. D. (2017a). Homomorphic encryption: Working and analytical assessment: dghv, helib, paillier, fhe and he in cloud security..
- Papisetty, S. D. (2017b). *Homomorphic encryption: Working and analytical assessment: Dghv, helib, paillier, fhe and he in cloud security*.
- Partida, D. (2022, Jun). *How apis are enabling open banking: Nordic apis* —. Retrieved from <https://nordicapis.com/how-apis-are>

- enabling-open-banking/
- Persiano, G., & Yeo, K. (2022). Limits of preprocessing for single-server pir. In *Proceedings of the 2022 annual acm-siam symposium on discrete algorithms (soda)* (pp. 2522–2548).
- Polasik, M., Huterska, A., Iftikhar, R., & Štěpán Mikula. (2020). The impact of payment services directive 2 on the paytech sector development in europe. *Journal of Economic Behavior Organization*, 178, 385-401. Retrieved from <https://www.sciencedirect.com/science/article/pii/S0167268120302328> doi: <https://doi.org/10.1016/j.jebo.2020.07.010>
- Pomerleau, P.-L., Lowery, D. L., et al. (2020). Countering cyber threats to financial institutions. In *A private and public partnership approach to critical infrastructure protection*. Springer.
- Riley, N. (2021, Jul). *Apis in banking: Four use cases (and why you should care)*. Retrieved from <https://www.csiweb.com/what-to-know/content-hub/blog/apis-in-banking-four-use-cases-and-why-you-should-care/>
- Sathya, S. S., Vepakomma, P., Raskar, R., Ramesh and Ramachandra, & Bhattacharya, S. (2018). *A review of homomorphic encryption libraries for secure computation*. arXiv. Retrieved from <https://arxiv.org/abs/1812.02428> doi: 10.48550/ARXIV.1812.02428
- Sharma, D. K., Singh, N. C., Noola, D. A., Doss, A. N., & Sivakumar, J. (2022). A review on various cryptographic techniques & algorithms. *Materials Today: Proceedings*, 51, 104–109.
- Sharma, T. (2016). E-voting using homomorphic encryption scheme. *International Journal of Computer Applications*, 141(13), 14–16.
- Taylor-Kerr, A. J. (2020). Adopting open banking in canada: An analysis of current global frameworks.
- Team, T. I. (2022, Sep). *Open banking: Definition, how it works, and*

- risks. Investopedia. Retrieved from <https://www.investopedia.com/terms/o/open-banking.asp>
- Tebaa, M., Hajji, S. E., & Ghazi, A. E. (2012). Homomorphic encryption method applied to cloud computing. In *2012 national days of network security and systems* (p. 86-89). doi: 10.1109/JNS2.2012.6249248
- Titus, A. J., Kishore, S., Stavish, T., Rogers, S. M., & Ni, K. (2018). *Py-seal: A python wrapper implementation of the seal homomorphic encryption library*. arXiv. Retrieved from <https://arxiv.org/abs/1803.01891> doi: 10.48550/ARXIV.1803.01891
- Tsai, W., Bai, X., & Huang, Y. (2014). Software-as-a-service (saas): perspectives and challenges. *Science China Information Sciences*, 57(5), 1–15.
- Venkatesh, A., & Eastaff, M. S. (2018). A study of data storage security issues in cloud computing. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 3(1), 1741–1745.
- Waldrop, M. M. (2001). The origins of personal computing. *Scientific American*, 285(6), 84–91.
- What is a digital bank and how it will benefit you.* (2022, Jul). Retrieved from <https://www.outlookindia.com/business/digital-bank-news-what-is-a-digital-bank-and-how-it-will-benefit-you-news-212287>
- Williams, C. M., Chaturvedi, R., & Chakravarthy, K. (2020, September). Cybersecurity risks in a pandemic. *Journal of Medical Internet Research*, 22(9), e23692. Retrieved from <https://doi.org/10.2196/23692> doi: 10.2196/23692
- Wolters, P. T., & Jacobs, B. P. (2019). The security of access to accounts under the psd2. *Computer law & security review*, 35(1), 29–41.
- Yaji, S., Bangera, K., & Neelima, B. (2018). Privacy preserving in

- blockchain based on partial homomorphic encryption system for ai applications. In *2018 ieee 25th international conference on high performance computing workshops (hipcw)* (pp. 81–85).
- Yousuf, H., Lahzi, M., Salloum, S. A., & Shaalan, K. (2021). Systematic review on fully homomorphic encryption scheme and its application. *Recent Advances in Intelligent Systems and Smart Applications*, 537–551.
- Zachariadis, M., & Ozcan, P. (2016). The API economy and digital transformation in financial services: The case of open banking. *SSRN Electronic Journal*. Retrieved from <https://doi.org/10.2139/ssrn.2975199> doi: 10.2139/ssrn.2975199
- Zachariadis, M., & Ozcan, P. (2017). The api economy and digital transformation in financial services: The case of open banking.
- Zhang, Q. (2021). An overview and analysis of hybrid encryption: The combination of symmetric encryption and asymmetric encryption. In *2021 2nd international conference on computing and data science (cds)* (pp. 616–622).
- Zhou, T., Yang, X., Zhang, W., & Wu, L. (2016). Efficient fully homomorphic encryption with circularly secure key switching process. *International Journal of High Performance Computing and Networking*, 9(5-6), 417–422.

ملخص الدراسة

برز التشفير الهومومورفي بين مختلف تقنيات التشفير، حيث يسمح بإجراء العمليات الترتيبية على البيانات المشفرة دون الكشف عن النص الأصلي. لسنوات عديدة، كرس الباحثون جهودهم لإيجاد طرق لحماية البيانات. تم إجراء العديد من الأنواع والدراسات للعثور على أفضل طريقة لحماية البيانات. تم اختبار طرق التشفير في مجالات مختلفة. قام الباحثون بإجراء دراسات نظرية حول مدى أهمية وكفاءة التشفير الهومومورفي. إن الحاجة إلى تطبيق التشفير الهومومورفي في القطاع المالي ضرورة لضمان الاستخدام الآمن لبيانات الخدمات المصرفية المفتوحة الكبيرة

لسنوات عديدة، كرس الباحثون جهودهم لإيجاد طرق لحماية البيانات. تم إجراء العديد من الدراسات للعثور على أفضل طريقة لحماية البيانات. تم اختبار طرق التشفير في مجالات مختلفة. لقد قام عدد قليل الباحثين بفحص استخدام التشفير الهومومورفي جزئيًا ونوعيًا وبالكامل. ومع ذلك، لم تجري هذه الدراسات تحقيقات تجريبية تركز بشكل خاص على تطبيق هذه الطريقة التشفيرية ضمن منصات الخدمات المصرفية المفتوحة التي تستخدم مشاركة البيانات الكبيرة للخدمات المصرفية المفتوحة.

تم اختبار خوارزمية التشفير الهومومورفي بالكامل على مجموعة بيانات الخدمات المصرفية المفتوحة في هذه الدراسة، باستخدام قاعدة بيانات مفتوحة المصدر تعتمد على نظام التشغيل لينكس ومكتبة خوارزمية التشفير الهومومورفي.

تستكشف الدراسة تطبيق خوارزمية التشفير الهومومورفي بالكامل في تأمين البيانات المالية. يظهر النشر التطبيقي كفاءته العملية في ضمان أمان البيانات، حتى في حالة الاختراق. بالإضافة إلى ذلك، قد تكون نتائج الدراسة لها تأثيرات على تطوير تقنيات الخدمات المصرفية المفتوحة ومشاركة البيانات الكبيرة في المستقبل، مؤكدة أهمية الخوارزميات الهومومورفية المتقدمة في هذا المجال.

جامعة البحرين

قسم هندسة الحاسوب



كلية تقنية المعلومات

تطبيق خوارزمية التشفير الهومومورفي لتأمين مشاركة البيانات

الكبيرة في الخدمات المصرفية المفتوحة

أطروحة مقدمة كجزء من متطلبات الحصول على

درجة الماجستير في الأمن السيبراني

إعداد

دانة عبدالله الرئيس

20190200

إشراف

د. مازن علي

أستاذ مشارك

جامعة البحرين

مملكة البحرين

مارس 2024