National College of Ireland

Homomorphic Encryption In Open Banking

MSc Research Project
Masters In Cybersecurity

Priyanka Srirangam
Student ID: X21125708

School of Computing
National College of Ireland

Supervisor: Ross Spelman

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Priyanka Srirangam
Student ID:	X21125708
Programme:	Masters In Cybersecurity **Year:** 2022
Module:	Research Internship
Supervisor:	Ross Spelman
Submission Due Date:	13/12/2023
Project Title:	Application of Homomorphic Encryption In Open Banking
Word Count:	8200 **Page Count** 19

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Priyanka Srirangam
Date:	13/12/2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not	□

sufficient to keep a copy on computer.	

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Homomorphic Encryption In Open Banking

Priyanka Srirangam
X21125708@Student.ncirl.ie

Abstract

Open banking is the practice of enabling secure sharing of financial information of a customer between a data owning financial entity and an innovative financial service provider through a set of well-defined APIs and third-party aggregators. The third-party aggregators use customer data for training AI models to offer competent digital solutions for fraud detection, lending analysis etc. Prior to this, aggregators are required to minimize customer data to eliminate re-identification or data leakage concerns. But the inadequate and possibly confusing regulations around the minimization techniques leave financial institutions to adopt an individualised risk-based approach to evaluate the minimization procedures in place. Thus, usage of homomorphic encryption is proposed in this research work, to encrypt customer data before arriving at the 3rd party aggregators. This will help enhance data security and assuage privacy concerns surrounding data minimization techniques. Homomorphic encryption is a form of encryption which allows for computations to be performed on encrypted data. The result of such computation is same as that of normal operation on plain data. The complexity of computations that can be performed are continuously improving, with latest applications in machine learning. Key focus of the paper is to show a high-level design of an open banking ecosystem embedded with homomorphic encryption. A simple implementation is included to prove that it is feasible to encrypt and decrypt API payloads using available homomorphic encryption libraries. Furthermore, the research includes evaluation of security and performance of the proposed approach, as well as utility of the homomorphically encrypted data for predictive AI models. The paper then concludes with a view of current challenges with the implementation and future areas worthy of research.

1 Introduction

Data is the most valuable resource in the current world run by Information technology. Data security refers to the protection of personal data from unauthorized access, attacks and exploitations so that data integrity is maintained. This ensures that data remains accurate, reliable and available to authorized parties only. In the current AI driven era, there is a lot of focus on security, privacy and ethical usage of customer data. Financial data, when used appropriately has the ability to enrich and simplify a customer's journey. This is achieved through a wide range of financial service offerings, starting from a personal loan application of worth a few thousand Euros to evaluation of credit risk for much larger financial transactions. Thus, came open banking, a technological innovation in the financial services industry where financial institutions can openly share the data of a consenting customer with each other, in order to ease a customer's journey through a financial service. This would not only mean reduced paperwork or manual uploading of information into websites by a customer, but also possible automation of decisioning, fraud detection, credit risk checks etc.

to name a few, using predictive AI models. To understand this interaction better, please refer to figure 1. There are 3 key entities in the open banking ecosystem as depicted in the figure.

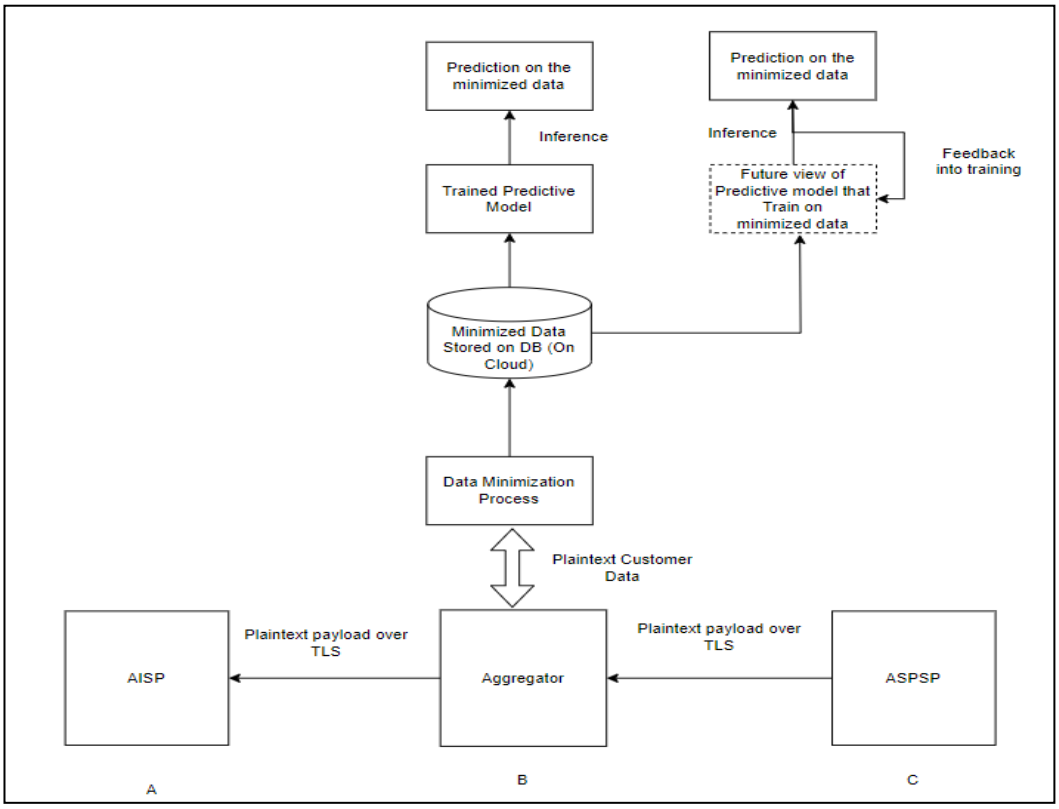

Figure 1: Depiction of present-day open banking ecosystem

A: AISP/PISP (Account Information Service Provider/Payment Information Service Provider) - Destination financial institution which receives the customer data and implements financial service leveraging customer's data i.e. data controller. For the purpose of this research, only AISP aspect has been taken into account and only APIs related to AISP are considered for encryption. Thus, only AISP is referenced in the rest of the documentation.
B: Aggregators - 3rd party companies which leverages customer data to facilitate the AISP/PISP, through API layering and prediction models.
C: ASPSP (Account Servicing Payment Service Providers) - source of the customer data i.e. data owner.

Open banking requires open sharing of data between A & C through RESTful APIs. Some of the key APIs include Accounts, Transactions, Balance, Party etc. As evident from the figure, in the process data sharing, B also receives a copy of the data acting as an intermediary. B uses acquired data to train internal machine learning models, after applying a minimization process, which is elaborated in the next section. Regulatory bodies require aggregators, which act as bridging agents between financial institutions, to employ data minimisation techniques such as anonymisation, pseudonymisation etc. to remove personally identifiable information either by redaction or tokenization process.[1]

[1] https://documents1.worldbank.org/curated/en/099425002082230437/pdf/P1705050aeb8e704f088260f228802b73b8.pdf

1.1 Challenges of data minimization

Utility of minimized data is reduced due to modification or removal of important personal information[2]. Also, such techniques can allow for reidentification of customer with the help of cross reference of additional data. An article[3] provides an interesting statistic by one study, that three key characteristics such as gender, birthday, and zip code can be used to identify 87% of American population and another study[4] states the re-identification percentage to be approximately 69%. Neither of the numbers are insignificant. An article by IaaP[5] further discusses the confusion in the EU data protection regulations surrounding anonymization and pseudonymization techniques, and the necessity for subsequent deletion of original data to avoid data link back etc. The article clearly elaborates on the absence of a one-size-fits-all minimization technique. One other article by IaaP[6], discusses the necessity for a pragmatic anonymization approach to be employed by EDBP regulators and how in the meantime, institutions are required to follow a risk-based approach for each use case. While there are tools and guidelines available to data controllers to determine the best suited data minimization approach for a given use case, there is no guarantee that the approach offers total data security. Individual financial institutions are left to go through legal procedures along with the 3rd party institutions to understand the risk of data minimization techniques employed and their completeness. Such risk-based approach will never serve to address the data security concerns at the root.

1.2 Role of Homomorphic Encryption

Homomorphic encryption is an encryption scheme where several mathematical operations can be performed on encrypted data without ever having to decrypt. The results of the operations when decrypted match the results of computations without encryption. This unique property makes Homomorphic Encryption, one of the most desirable encryption schemes of the present time, with machine learning requiring access to substantial amount of raw data for training. If the training can be successfully completed on encrypted data, the privacy and security achieved in the process is no match to any anonymization or pseudonymization process.

Where the regulations stand to be incomplete and unclear, technology could come to rescue and such is homomorphic encryption, a trending privacy preserving technology. With several years of research backing homomorphic encryption and its incredible ability to run mathematical operations on encrypted data never revealing the underlying plaintext, this technology could be utilized in open banking ecosystem to enhance data security. It is proposed that the open banking API payloads be encrypted using homomorphic encryption. The proposal is that the payload shared among the 3 entities can be homomorphically

---

2 https://edpb.europa.eu/sites/default/files/webform/public_consultation_reply/digitaleurope_submission_to_edpb_consultation_guidelines_interplay_of_psd2_and_gdpr.pdf

3 https://dataprivacylab.org/projects/identifiability/paper1.pdf

4 https://crypto.stanford.edu/~pgolle/papers/census.pdf

5 https://iapp.org/news/a/a-guide-to-the-eus-unclear-anonymization-standards/

6 https://iapp.org/news/a/the-definition-of-anonymization-is-changing-in-the-eu-heres-what-that-means/

encrypted end to end. The private key in this context need only be share between A & C. While B remains ambivalent to the private key, it would still have access to encrypted data, public key and the context. These 3 factors will enable the aggregator to be able to play the role it does today and act successfully as an intermediary. Such intermediary functions usually include maintenance of the customer data for continued access[7], services such as credit checks, fraud detection based on predictive algorithms etc. Because of homomorphic nature of the data, as part of this research, we will show that encrypted data can be used to train AI models and provide meaningful inferences of real time encrypted data. And all of this can be achieved without comprising data security since data remains encrypted on the aggregator premises. It is important to note that Homomorphic Encryption (HE) comes with certain costs. One such cost is ciphertext sizes. HE results in large ciphertexts and to circumvent this problem, it is proposed here to use selective encryption of only PII attributes in a payload.

1.3 Research Questions

Primary goal of the paper is to conclude that homomorphic encryption can be applied to payloads in the Open Banking Ecosystem to significantly enhance the security of the personal data in the context of aggregators. The research also stresses on the necessity of the utility of encrypted data, due to the open banking usage. Aggregators should still be able to train and run predictive analytics models on the homomorphically encrypted data. While security and performance aspects of homomorphic encryption have been thoroughly discussed in several previous works, few observations based on this use case will also be noted. To that effect, below are the few research questions that are addressed as part of the thesis.

- Can homomorphic encryption be applied in the context of Open Banking?
- Is it possible to build and implement a high-level design for such a system?
- Are there libraries available today which can implement such encryption and decryption of API payloads?
- Is it possible to implement such solution in a high-level language such as Python?
- Is it still possible to run a sample prediction algorithms on open banking data post encryption?

1.4 Document Structure

The research paper reviews the latest work in this area showing great progress that has been made with respect to improved performance of the homomorphic encryption algorithms, the increasing use cases being researched on application of homomorphic encryption in the financial services and other critical areas such as medical services. Finally, a quick review of literature around successful training and inference on homomorphically encrypted data for several simple to moderately complex machine learning algorithms is carried out. After the literature review, a very high-level architecture of open banking ecosystem with the usage of

[7] https://truelayer.com/blog/compliance-and-regulation/explaining-changes-to-the-90-day-rule-for-open-banking-access/

homomorphically encrypted payload is presented. The paper in conjunction with configuration manual shows how an open banking API payload can be homomorphically encrypted, and presents the evaluation of several aspects of security, performance and utility of the encrypted data. In the end, several open questions, constraints and scope for future work are discussed.

2 Related Work

In the following section, a number of articles written by several experts in the field are reviewed to offer insights into the research work that follows.

2.1 Brief survey of homomorphic encryption

While homomorphic encryption has been around for decades, only in the early 2000s, first plausible algorithm for a fully homomorphic encryption based on lattice-based cryptography was proposed by Gentry (Gentry, 2009). This proposed construct supports both additions and multiplications along with an innovative approach to reduce noise to evaluate more complex circuits. Because noise in encrypted outputs of operations is a serious concern associated with homomorphic encryption, the technique of bootstrapping proposed by Gentry became vital to a number of subsequent works in the field. Reportedly, implementation of early Gentry cryptosystem took approximately 30 minutes to complete a simple bit operation (Gentry & Halevi, 2011). Further enhancements to this system radically improved the execution time, bringing it down to several milliseconds at the moment. Further in 2010, Dijk, Gentry, Halevi and Vaikuntanathan proposed a 2nd fully homomorphic encryption scheme (Gentry, et al., 2010) based on Gentry's constructs but seeking simplification in the system by enabling operations over integers. These schemes are collectively known as the first generation of homomorphic encryption schemes. The 2nd generation schemes still widely used in the majority of encryption libraries today (SEAL[8], OpenFHE[9], HELib[10]) such as BGV (Brakerski, et al., 2011), BFV (Fan & Vercauteren, 2012) schemes are based on hardness on Ring With Learning errors problem (RLWE) (Lyubashevsky, et al., 2010) in mathematics. These encryption schemes utilize the bootstrapping technique proposed by Gentry to scale up to a fully homomorphic encryption scheme with no limit on number of circuits to be evaluated. In a later work 2013, setting in motion the 3rd generation of FHE, Gentry, Sahai, Waters propose usage of "approximate eigenvector" to improve performance of multiplications over "relinearization", making the scheme faster and more comprehensible (Gentry, et al., 2013). This paper also put forth the idea of identity based FHE. While this scheme was based LWE, further ring variants led to development of FHEW[11] and TFHE[12] schemes, both of which are still key players in the field of FHE. Zama AI using Concrete library based on TFHE, is one of the fastest growing homomorphic encryption libraries. TFHE introduced an impressive technique known as programmable bootstrapping, which

8 https://www.microsoft.com/en-us/research/project/microsoft-seal/
9 https://www.openfhe.org/
10 https://homenc.github.io/HElib/
11 https://github.com/lducas/FHEW/tree/0959af8daf6635a5e69013f6db7120c6d39e2319
12 https://tfhe.github.io/tfhe/

returns any function of the output ciphertext with controlled noise level[13]. This technique enables this scheme to make leaps and bounds in the field of AI using homomorphic encryption. Finally, one of the key schemes in the fourth and current generation is CKKS which enables floating point operations (Cheon, et al., 2013). Due to its inherent similarities to the way machine learning algorithms work with real values, this scheme plays a key role in the machine learning world. Li and Miccaiancio discuss in their paper (Li & Micciancio, 2020) the possible passive attacks and opportunities to retrieve secret keys in CKKS scheme, thus further work is deemed necessary to protect against this. A slight variation of RLWE based schemes proposed later on where the depth of circuits that can be evaluated without bootstrapping, is predetermined, known as leveled FHE (Brakerski, et al., 2014) is still very much in use today. It can be stated conclusively that homomorphic encryption has made immense progress through the four generations, with respect to simplification of techniques, improvements in performance and security aspects, proving itself to be a viable technology in the current machine learning dominated world where data privacy is invaluable. In the proposed research use case, a selective payload encryption approach is adopted to circumvent some of the cost and size implications with encryption technique.

2.2 Application of homomorphic encryption in financial and healthcare services

The research work by Cao (Ruiwen, 2022) evaluates usage of homomorphic encryption in the context of credit score evaluation and reviews few performance measures to improve overall computation overhead by implementing a weighted distance approach. The application use case is quite promising, and the core notion is quite similar to proposed research in trying to achieve higher levels of security around customer financial information by keeping the data private and evaluating credit scores through homomorphic operations. It is to be noted that the author indicates the communication overhead of the approach can be alleviated through sufficient network bandwidth allocations. Similarly another paper by Munjal and Bhatia (Munjal & Bhatia, 2022) alludes to prevalence of cloud technologies, the role third party service providers play in today's healthcare in achieving effective patient care by the way of Electronic Health Records (EHR) and the manner in which homomorphic encryption has enhanced the data security in the healthcare context. This in turn aligns with the proposed research paper's problem statement in being able to offer better data security for the data living on 3rd party service provider's end. Both financial and medical history records of an individual can be regarded as two equally critical sets of personal information that deserve utmost security. This paper also refers to the pitfalls of anonymization and pseudonymization techniques in fulfilling computation necessities while maintaining data privacy effectively. The paper then reviews a number of successful use cases underpinned by FHE schemes over a decade and more, such as a SafeBioMetrics system using BGV (Bocu & Costache, 2018) , usage of contrast enhancing RDH with FHE to build a safe and high visual quality framework for medical pictures (Yang, et al., 2019), a self-serviced medical device using Homomorphic Encryption (Li, et al., 2020), usage of BGV (Brakerski, et al., 2014) to calculate

13 https://whitepaper.zama.ai

average/minimum/maximum heart rate(s) (Soyata & Kocabas, 2020), maintaining healthcare data integrity using FHE schemes in CFHET-PPF (Sendhil & Amuthan, 2021) etc. The number of successful use cases in the field of healthcare surrounding HE is highly promising.

2.3 Predictive analytics using homomorphically encrypted data

The paper (Brand & Pradel, 2023) proposes a practical approach to training machine learning models using homomorphically encrypted data, by showing that a Support Vector Machine (SVM) learning model[14] can be trained using leveled homomorphic encryption in a scalable fashion. The paper reviews important work carried out by predecessors on usage of HE in the inference phase of machine learning, such as neural networks trained on plaintext data was shown to draw successful inferences on encrypted data by Gilad-Baschrach (Gilad-Bachrach, et al., n.d.), work based on polynomial approximation of ReLU function (Hesamifard, et al., n.d.), work by Boemer et al. to implement MP2ML (Boemer, et al., 2020) for ML inference over encrypted domain with the help CKKS FHE scheme etc. The authors as with most machine learning models using HE, opted to use CKKS scheme, adopted a carefully assessed client assisted model to offload heavy-lifting to Cloud. The paper shows extreme promise in optimizing the training over homomorphically encrypted data by employing novel techniques. This offers insight into the evaluation of utility in machine learning training and inference of homomorphically encrypted data in the proposed research area. Another paper by (Bos, et al., 2014) presents a working implementation of a prediction model for heart attacks based on few body measurements using homomorphically encrypted data. The paper avidly reviews various use cases where data concern is of utmost priority in healthcare, before delving into usage of their own previously proposed leveled homomorphic encryption scheme in the paper (Naehrig, et al., n.d.). Another important contribution by the authors is an automatic parameter selection module for determining appropriate parameters for correct and secure evaluation of complex circuits, which could prove to be useful in future research work of the proposed use case. Authors showed that such module made for efficient evaluation of models based on logistic regression and Cox proportional Hazard regression.

3 Research Methodology

This section briefs the research methodology used in order to arrive at the proposed system design, implementation and evaluation.

3.1 Transactions and Party API data Review & Classification

Based on the sample payloads provided by UK Open banking website [15], a quick review of attributes in various API payloads (such as accounts, transactions, party, balance etc.) is carried out. Party and Transaction API payloads are particularly investigated further, as the selected use cases for the proposed encryption application. These APIs have varying range of

[14] https://scikit-learn.org/stable/modules/svm.html
[15] https://openbankinguk.github.io/read-write-api-site3/v3.1.11/profiles/account-and-transaction-api-profile.html

attributes of interest, such as individual names, transaction references, amounts etc. An attempt to manually categorize the API attributes based on an understanding of personally identified information was made to select attributes for encryption. Certain attributes may not necessarily be personally identifiable but could be categorized as sensitive, such as balance after a transaction. So, as an illustration balance has been included PII attributes. Example PII attributes include name, full legal name, account role from Party API payload and transaction reference, information from Transaction API payload. It is important to identify specific attributes that need to undergo encryption so as to limit the performance impacts such as encryption time and encrypted payload size. Further information such as account number, account co-owner etc. from Accounts payload would fall under PII category too.

3.2 Identifying FHE algorithm for the encryption

BGV (Brakerski, et al., 2011) and BFV (Fan & Vercauteren, 2012) schemes as reviewed earlier, are homomorphic encryption algorithms based on variants of Learning With Errors problem, support SIMD operations[16] where the plaintext is a vector of values. Thus, they are all suitable to be used in the context of open baking. While CKKS (Cheon, et al., 2013) is more suitable for real numbers, BFV (Fan & Vercauteren, 2012) and BGV (Brakerski, et al., 2011) are preferred for integer. This very fact makes BFV (Fan & Vercauteren, 2012) or BGV (Brakerski, et al., 2011) more suitable to encrypt ASCII encoded characters of a string value. To elaborate, a string is composed of a few characters and there is no direct API to encode a character in any of the HE libraries. So, array of characters is first converted into an array of integer ASCII codes, and the resulting array is encrypted homomorphically. The resulting ciphertext is encryption of the string. On decryption, the ciphertext can be decrypted to output the array which in turn can be decoded to retrieve the original string. On the other hand, CKKS (Cheon, et al., 2013) is more suited for encrypting real numbers such as transaction amount.

3.3 Generating Homomorphic Encryption Context and Keys

Once the algorithm has been chosen, it was important to determine the process to generate the encryption context and keys. The context is a special object which holds information such as algorithm type, encryption parameters such as plaintext modulus and polynomial modulus. Polynomial modulus degree denoted as n (a power of 2 such 1024) is the number of slots per plaintext of elements to be encoded in a single ciphertext. Another important parameter is t, the Plaintext modulus such that encrypted operations happen are in t modulus. Value of t must be such that it is a prime number satisfying condition t-1 should be divisible by 2^n. Example values of n and t are 1024 and 65537. Luckily for the implementation purposes of this research, the high-level libraries such as the ones to be reviewed in next section offer an easy way to set these parameter values. With simple method calls to the underlying encryption backbone library (such as SEAL or OpnFHE), the encryption context and keys can be generated.

[16] https://www.sciencedirect.com/topics/computer-science/single-instruction-multiple-data

3.4 Applying the proposed HE on a dataset using a program

A number of high-level libraries were investigated to review feasibility and ease of use. Microsoft SEAL (Microsoft, 2023) is an open-source homomorphic encryption library written in programming language C++[17], which allows for computations to be performed on encrypted data. It supports BFV (Fan & Vercauteren, 2012), BGV (Brakerski, et al., 2011) and CKKS (Cheon, et al., 2013) algorithms. Open FHE (OpenFHE, 2023) is another powerful open-source library written in C++, which supports BFV, BGV, CKKS, DM/FHEW (Badawi & Polyakov, 2023) algorithms, and allows for switching between few combinations of these algorithms. Pyfhel (Pyfhel, 2023) is a Python library which provides Python interfaces for homomorphic operations such as encryption, decryption, addition, multiplication with the backbone of SEAL and OpenFHE libraries. Due to underlying C++ backends, the Pyfhel library requires C++ support to run programs. It supports BFV, BGV and CKKS. Due to ease of use, Pyfhel has been used as part of this research work to develop the encryption module. Finally, TenSEAL (TenSEAL, 2023) is a Python Library which allows homomorphic encryption operations on tensors. This also utilises SEAL as its encryption backend. It mainly supports BFV, CKKS algorithms along with addition, subtraction, and multiplication operations. TenSEAL has been used for evaluating utility of the encrypted data in the evaluation section later. Concrete AI was investigated for this purpose too, but the library mainly supports evaluation of machine learning models over encrypted data at present. The training aspect is still a work in progress and thus the choice was made to use TenSEAL.

3.5 Evaluation

As part of evaluation of the proposed solution, three main factors have been investigated – security, performance and utility. For security, side channel attack and lattice-based attacks are mainly reviewed. A case is also made to show that existing security infrastructure surrounding open banking still remains and the attack surface only applies to the encrypted data on the aggregator side. Performance wise, execution time of the implementation programs is reviewed. A very insightful view of encrypted payload size increment from base plaintext API payload is also offered. And finally, utility of the encrypted data is reviewed. Utility implies the ability to successfully use the (encrypted) data on the aggregator side to train AI models. Proving this is key to the success of proposed work, as explained in introduction, for open banking use case. In the evaluation section, we show that TenSEAL library can be used to train a logistic regression model using encrypted transaction amount data and further inferences can be made using the encrypted data as well.

4 Design Specification

In this section, the proposed architecture of open banking with homomorphic encryption is reviewed. A brief listing of scope limitation is also provided.

[17] https://cplusplus.com/

4.1 Overview

Image below depicts a high-level system design with Homomorphic encryption in place, with source (ASPSP - Account Servicing Payment Service Providers), destination (AISP – Account Information Service Provider) and aggregators in view. In the following diagram, as in actual open banking ecosystem, an aggregator is at the centre of the entire transaction. This entity holds the customer information on behalf of an AISP, as mentioned before.

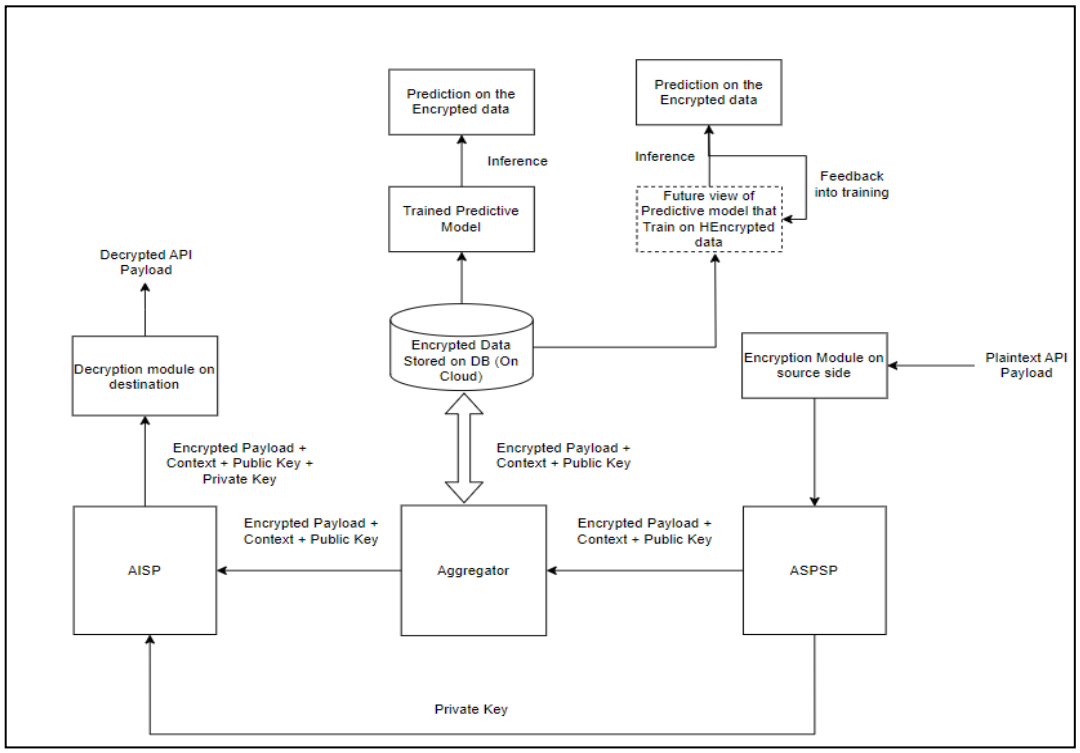

Figure 2: Depiction of proposed open banking ecosystem with Homomorphic Encryption

In the proposed design, the data at the aggregator is always in an encrypted state. This would imply that the source (data owner entity – ASPSP) would entail a new homomorphic encryption module which would accept a plaintext API payload, create the encryption context (algorithm type, polynomial modulus n, plaintext modulus t), save the output public key, evaluation key (used to encrypt the secret key to evaluate multiplication or more deeper circuits in HE), rotation key and secret key. The source would also classify the payload attributes as PII and non-PII so as to minimize the encryption time and encrypted payload size. The context, the payload and the selected attributes are fed into the encryption program implemented as part of the research, resulting in partially encrypted payload. The output encrypted payload along with context (without private key) is to be shared with the aggregator. The destination (i.e. AISP) needs the private key, so as to be able to decrypt the payload if necessary. It is important to highlight the necessity of sharing the private key, so as to keep the open nature of data exchange intact between source and destination while limiting the view of aggregator.

On the aggregator side, the payload and the context can be stored for further usage such as forwarding the data onto AISP as necessary, to train the predictive models for inference

purposes etc. The data on the aggregator premise always remains in an encrypted state with only the source and the destination with the knowledge of the private key as intended.

On the destination (AISP) side, a new decryption module would need to exist which accepts the encrypted payload, context from the aggregator whenever requested. With the help of private key, the encrypted payload can be decrypted and used as necessary. The continued access provision of open banking up to 90 days would mean the same private key can be used for that period. Alternatively, the key can be rotated for better security. The only time the private key needs complete update would be when the source encrypts the payload and set up a new encryption context.

The proposed design achieves several of the intended goals by enabling selective encryption of API payload end to end. The selective nature of encryption can be changed to complete payload encryption depending on future improvements in the field of homomorphic encryption around cipher text size and run time. The encrypted data due to its homomorphic nature, can still be used to train prediction models. Such trained model can also run inferences on encrypted data and can be further trained continually. This ensures that PII data is never exposed to any component of the aggregator ecosystem and remains secure throughout the lifecycle 100%. A further food for thought is the necessity of a decryption module at the destination side. With the advent of privacy preserving technologies such as multi-party computations, it is possible to utilize encrypted data even on receiving end of an open banking transaction, ensuring complete security of data.

4.2 Scope Limitations

The above system design and the research work by extension doesn’t delve into certain aspects of the proposal due to limited time nature of the current research work. One of them being key exchange between AISP and ASPSP without the aggregator's awareness. While it is an important aspect to be investigated, the focus was on proving the encryption/decryption process for the payload. But there are provisions of such key exchange between two systems already in the security world for example TLS set up. Another aspect which is out of scope for this research is parameter consideration based on factors such as payload attribute sizes. While there is an evaluation and review of performance of the encryption process and predictive algorithms on the encrypted data, changes in homomorphic encryption parameters such as plaintext and polynomial moduli has not been explored further in this paper. Also, usage of evaluation and rotation keys has been left out of scope for this work.

5 Implementation

The solution includes an encryption and a decryption program using the Pyfhel (Pyfhel, 2023) library in Python using a Visual Studio Code IDE. Complete details of the code and implementation dependencies can be found in the configuration manual. Here pseudocode for encryption and decryption programs have been provided.

5.1 Pseudocode for Encryption Module

- Read the API payload JSON and run the following steps for each attribute.
    - Declare a list of attributes to be treated as PII attributes.
    - Drill down to attributes and call encryption routine if an attribute is present in piiAttributesList.
    - kick off encryption of integer/string encryption method depending upon attribute type. Convert the ciphertext instance to a base64 string and return.
        - Create an instance of HE context with appropriate parameters.
        - If a given data type is integer, encrypt using BFV and the above created context.
        - If a given data type is String, obtain an array of characters from the string. Convert the characters to corresponding ASCII coded Integers. Such an integer can be fed through the encryption routine to the above sub step.
        - For the purpose of simplification, skip encryption of dates. This can be explored further by converting the date to an integer and feeding it to step ii as well.
        - Under each encryption, quickly show the integer and string can be decrypted from the ciphertext.
    - Once the inline decryption is proved, add the encrypted value back to the dictionary against a field.
- Save the final JSON with a combination of encrypted and plaintext attributes.

Thus provided an input JSON, the application converts it to (selectively) homomorphically encrypted payload and outputs the JSON.

5.2 Pseudocode for Decryption Module

- Read the API payload JSON and run the following steps for each attribute.
    - Declare a list of attributes to be treated as PII attributes.
    - Drill down to attributes and call decryption routine if an attribute is present in piiAttributesList.
    - kick off decryption of integer/string encryption method depending upon attribute type. Convert the ciphertext instance to a base64 string and return.
        - Create an instance of HE context with the context, private key and public key attributes on the payload. Remove these 3 attributes from the JSON once context is set up.
        - Decrypt each PII attribute using the HE context and keys.
        - Save each attribute back to the JSON.
- Save the final JSON.

6 Evaluation

In the following section, the detailed approach of evaluation of proposed design is reviewed. The section ends with discussing the evaluation results.

6.1 Security

One of the key points to note before delving into other security aspects is that the application of Homomorphic Encryption to the API payloads is proposed as an addendum to the existing security measures in place in open banking ecosystem, such as transport layer security using certificates signed by qualified Trust Service Providers[18]. Security of proposed implementation is directly related to security of underlying homomorphic encryption scheme. An encryption scheme is considered indistinguishable under chosen plaintext attack also known as IND-CPA if the attacker is unable to distinguish between results of encryption of two different messages of her choice. As the encryption scheme in use is BFV, hardness of Ring Learning with Errors problem comes into play. Most HE schemes including BFV enjoy security against IND-CPA thanks to the hardness assumption of underlying mathematical construct RLWE (Fauzi, et al., 2022). But authors in the same paper highlight the fact that the FHE schemes don't enjoy similar level of security against IND-CCA1 (where the attacker possesses decryption oracle for a brief time) and IND-CCA2 (where the attacker possesses decryption oracle indefinitely). Another important note, due to the small error width in the RLWE-based encryption schemes, usage of non-cyclomatic rings is not recommended as they are associated with certain weaknesses already identified (Chase, et al., 2017). Most researched libraries support 2-power cyclomatic rings, so this choice was already made for users in the favour of higher security. Next, we review few other possible attack types. A side channel attack is one where the physical cryptosystem is exploited to leak information. Due to the time complex computations involved in Homomorphic Encryption, an attacker could possibly trace information on the systems used for computations leading to malicious exploitation (Aydin & Aysu, 2023). HE based solutions are susceptible to Lattice attacks, where an attacker exploits vulnerabilities in the lattice structure to acquire information on private key. In certain cases, it is also possible to obtain plaintext from ciphertext without private key as show in a paper (Chunsheng, 2012).

6.2 Performance

Performance costs of the overall solution can be measured using few attributes such as overall execution time, sizes of ciphertext, the additional content that needs to be exchanged such as context and keys etc. The implementation of encryption and decryption programs using Pyfhel and BFV scheme naturally incurs a certain execution time overhead. Due to the latest the improvements in the underlying encryption library, the observed execution time is still within milliseconds. Important note is that instead of encrypting the entire payload, the research proposes a selective encryption of only PII attributes to keep execution time impacts to a minimum. As the review of history of homomorphic encryption indicates, the encryption and decryption times have significantly improved over the years and continue to do so, making the technology more viable for practical us cases. On the size of ciphertext, below

[18] https://www.digicert.com/faq/signature-trust/what-is-a-trust-services-provider

graph represents an analysis of variation between plaintext and encrypted payload types for Party & Transaction APIs. As it can be observed, there is a significant rise in the payload size post encryption. And this challenge of ciphertext size is known and an active under research area in Homomorphic encryption. Another important aspect to consider is that the payload presented in the research includes encryption context and keys, which in reality need not be repeated for every payload. For one open banking interaction, one set of context, keys can be forwarded to the aggregator and destination institution. Excluding context and keys from the payload reduces the size approximately by one-third. To further emphasise this point, the graph includes sizes of payloads without the context and keys alongside ones with context and keys and plaintext payload.

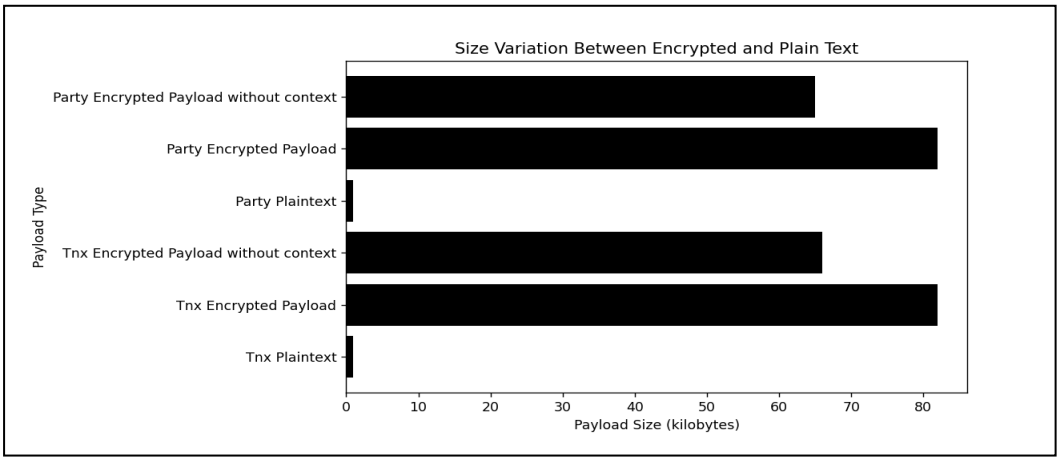

Figure 3: Payload Sizes Analysis chart

6.3 Utility of the encrypted data

In this research, for evaluating utility of the encrypted data, TenSEAL library has been used. This library enables homomorphic encryption on tensors, by offering Python APIs on the underlying C++ based SEAL. It mainly supports BFV and CKKS schemes. For this evaluation, an existing program[19] provided by TenSEAL library (TenSEAL, 2023) to illustrate the training and inference of encrypted data, has been further modified to aid with the evaluation of this use case specific data. The use case is to evaluate if a given transaction amount exceeds a certain limit i.e. 500. This is a very basic model which could be further enhanced for fraudulent transaction detection. The input data sets had a varying list of random transaction amounts under 1500, along with a column labelling the items exceeding the limit (0,1). Approximately 12 such datasets with different number of records starting from 4000 to 100,000 are used for both training and testing. The program splits each input data set into training and test data set by shuffling and delimiting the data points. First a logistic regression model (single node and single layer neural network) is trained and evaluated using plaintext data. This same trained model is used for evaluation using encrypted data later on. The program aims to show that the accuracy of the model is not impacted by the usage of encrypted data set. Since the input data set is floating point, CKKS encryption scheme is used in the program. It is observed that the over the 12 datasets, the accuracy of evaluation

[19] https://github.com/OpenMined/TenSEAL/tree/main/tutorials

between plaintext and encrypted data doesn't vary by much, in fact in 75% of tests the accuracy of the model over encrypted data was higher than plaintext data. As evident from the line graph below, the accuracy of the model evaluation remains quite high in the range of 0.9 (maximum being 1.0 i.e. 100%).

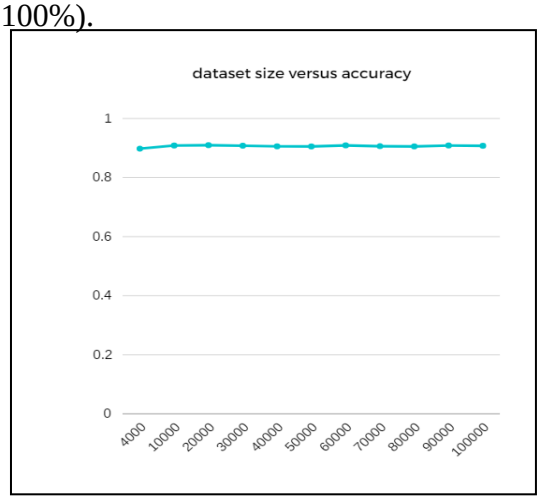

Figure 4: Accuracy of evaluation over encrypted data sets with x representing number of records in a data set (used for both training and inference) and y representing the accuracy of results on data set on a scale of 0 to 1.

As evident from the above graph, the inference using encrypted data on a trained model is highly efficient. Following this, the training of the model using encrypted data has been reviewed as well. For this, aforementioned model and implementation provided by TenSEAL (TenSEAL, 2023) has been modified and a subset of same test data with amounts and "over the limit" labels as before has been used. It is observed that all 4 sets of data used showed average accuracy of over 90% in 6 epochs. But the training over large sets encrypted data proved to be extremely resource heavy, for instance – data set with 50000 records executed for over an hour.

6.4 Discussion

While the security of the proposed solution is underpinned by security of homomorphic encryption in general, there are certain aspects such as key exchange which would need to be thoroughly investigated to ensure security of proposed solution. But as mentioned before, this has been considered out of scope for the present work. The encryption parameters such as plaintext and polynomial moduli would also play a key role strengthening the encryption system. On the performance aspect of the solution, there is enough evidence to state that ciphertext size stands to be an open challenge still, while execution time of the programs has been shown to be reasonable ranging in few milliseconds, with the selective encryption approach. The selective encryption approach could be challenged as it involves identifying attributes with PII data and only encrypting these. As such attributes increase, the ciphertext size could be a growing concern. It is also possible to improve the solution further by feeding a number of integers as a vector to the encryption algorithm at once and reducing multiple ciphertexts into only one, similar to how the string encryption was achieved. Finally, utility of homomorphically encrypted data for training and inference on machine learning models is

an active area of research. A simple use case of training and testing a regression model using transaction amount was presented. As explained in research methodology, transaction amount can be considered as a sensitive attribute rather than a PII attribute. But it has been chosen for model testing and training for simplicity and due to limited time available. As can be observed in the above evaluation details, the accuracies with encrypted inference (only) and encrypted training and inference have been shown to be over 90% for most data sets used with simple logistic regression model. It is important to note that the time required for training rises significantly with increase in the number of elements in the dataset. Also, the real-life use cases around machine learning would involve much more complex neural network circuits, so the cost implications around training and evaluation of these would be significant. While the present research work has been successful in showing that it is feasible to homomorphically encrypt standard open banking payloads, much more thought is required to make this a practical solution.

7 Conclusion and Future Work

This section concludes the thesis and highlights challenges encountered during the research process. In the end, several interesting future works have been discussed.

7.1 Conclusion

The journey through this research work has been quite intriguing, starting with limitations of data minimization techniques to understanding current homomorphic encryption landscape and the latest advances in it. The concern expressed in the introduction, around data security in the open banking ecosystem would have to be alleviated appropriately as we intend to encourage its usage. This research thesis offers an initial insight into what is achievable using homomorphic encryption to make data in open banking ecosystem more secure. Continued research in the field of Homomorphic encryption and increasing usage of the technology in other fields such as Healthcare are bringing the technology so much closer to being practical in its performance and ease of use. At the time of this research work, there are still non-negligent cost and size overheads to implementing HE in the open banking context. But these parameters have significantly improved through the four generations of the homomorphic encryption algorithms as evident from the literature review, trending towards more practical.

There is also active effort required to standardize the Homomorphic encryption technology across the industries, to encourage safe and consistent usage. A noteworthy article[20] by Alan Turing institute reviews the key role homomorphic encryption could play in the financial services sector in tracking money laundering and detecting fraudsters. The article applauds the work being carried out by Homomorphic Encryption Standardization project[21] to this end. The proposed system design with homomorphic encryption in Open Banking might not only assuage the initial concern of data security and privacy in the 3rd party aggregator environment, but it might also offer an end-to-end payload encryption

[20] https://www.turing.ac.uk/blog/homomorphic-encryption-future-secure-data-sharing-finance
[21] https://homomorphicencryption.org/

optionality for open banking interactions. This is not too different from payload encryption in transaction processing with entities such as MasterCard[22] and Visa[23]. Such an end-to-end encryption could act as an additional layer of security on top of the existing TLS layer among the entities, while leaving the doors open for multi-party computations among all the concerned. Finally, it is prudent to call out the community support among Homomorphic Encryption developers, enthusiasts, open groups and cryptographers in offering guidance when approached, as it has been a key factor in successful completion of the research work.

7.2 Challenges

In Homomorphic Encryption, parameter sizes limit the maximum size of an input value that can be safely encrypted by a scheme, thus certain encryption schemes may not accept certain input sizes with smaller parameters. At the same time, larger parameters could imply larger encryption/decryption time and ciphertext size. Thus, it was bit challenging to arrive at a feasible combination of parameters for the encryption program. Another noteworthy aspect is string encryption. After extensive investigation to find a high-level API to encrypt strings in HE and failing to find any, a unique approach was adopted to treat string types separately. Each sting was broken down into an array of characters, followed by ASCII code conversion of each character and then the entire array was fed into the encryption method as BFV encryption scheme accepts a vector of integers. One of the libraries investigated at the beginning of the research work was Concrete by Zama AI, which is a Python library based on TFHE. But it proved to be slightly challenging to work with compared with Pyfhel as it is available only on Docker for windows. Also, Concrete doesn't allow for training on encrypted data at the moment unlike TenSEAL, while there is active research being conducted to achieve the same. APIs offered by the high-level language libraries to save the ciphertext, security context, keys in a feasible format are limited in terms of options. Another point to note is most of the core homomorphic encryption libraries such as SEAL, OpenFHE, HELib are based on C++, so a C++ compiler is a prerequisite to be able to execute any of the libraries.

7.3 Future Work

There are several open questions which can be researched as part of future work for this paper, some of which include but are not limited to the following. Key exchange between source and destination financial institutions needs to be investigated further, as this is an important aspect for successful implementation of homomorphic encryption in open banking. At the same, the need for destinations to have access to cleartext customer data could be challenged too, since the encrypted data is proved to be useful for important use cases such as machine learning model training and inference. Parameters such as ciphertext size, execution time, algorithm parameters (plaintext modulus and polynomial modulus), key sizes etc. can not only be optimized to make the implementation more practicable, but the automatic

[22] https://developer.mastercard.com/platform/documentation/security-and-authentication/securing-sensitive-data-using-payload-encryption/
[23] https://developer.visa.com/pages/encryption_guide

selection of parameters as mentioned in literature review of the paper (Naehrig, et al., n.d.) could be investigated further to enhance the practicality of HE libraries. Another area of research would be optimized storage of homomorphically encrypted payloads on aggregator platforms. Multiparty computation (Damgard, et al., 2022) between AISP/PISP and ASPSP for payment transactions is a curious area that can take the application of homomorphic encryption a step further. MPC would prove to play a key role in the future of secure interaction at the advent of quantum computing. Finally, investigation of homomorphic encryption library in different high level programming languages other than Python such as Java would be deemed useful for the industry considering majority of enterprise applications are based on Java as of 2023[24]. Finally, while it is understandable that the necessity for lower-level optimisations offered by C++ with respect to operations is prudent for the performance of homomorphic encryption, more work could be carried out on how to implement and optimize the schemes with other prevalent compilers such as Java.

References

Aydin, F. & Aysu, A., 2023. *Leaking Secrets in Homomorphic Encryption with,* s.l.: s.n.

Badawi, A. A. & Polyakov, Y., 2023. *Demystifying Bootstrapping in Fully Homomorphic Encryption,* s.l.: Cryptology ePrint Archive.

Bocu, R. & Costache, C., 2018. A homomorphic encryption-based system for securely managing personal health metrics data. *IBM Journal of Research and Development,* Volume 62.

Boemer, F. et al., 2020. *MP2ML: A Mixed-Protocol Machine Learning Framework for Private Inference,* s.l.: Cryptology ePrint Archive.

Bos, W. J., Lauter, K. & Naehrig, M., 2014. Private predictive analysis on encrypted medical data. *Journal of Biomedical Informatics,* Volume 50, pp. 234-243.

Brakerski, Z., Gentry, C. & Vaikuntanathan, V., 2011. *Fully Homomorphic Encryption without Bootstrapping.* [Online]
Available at: eprint.iacr.org

Brakerski, Z., Gentry, C. & Vaikuntanathan, V., 2014. (Leveled) Fully Homomorphic Encryption without Bootstrapping. *ACM Transactions on Computation Theory,* 6(3).

Brand, M. & Pradel, G., 2023. *Practical Privacy-Preserving Machine Learning using Fully Homomorphic Encryption,* s.l.: Cryptology ePrint Archive.

Chase, M. et al., 2017. *SECURITY OF HOMOMORPHIC,* s.l.: Microsoft.

Cheon, H. J., Kim, A., Kim, M. & Song, Y., 2013. *Homomorphic Encryption for Arithmetic of Approximate Numbers,* s.l.: s.n.

Chunsheng, G., 2012. *Attack on Fully Homomorphic Encryption over the Integers,* s.l.: Cornell University.

Damgard, I., Pastro, V., Smart, N. & Zakarias, S., 2022. *Multiparty Computation from Somewhat Homomorphic,* s.l.: Springer.

Fan, J. & Vercauteren, F., 2012. *Somewhat Practical Fully Homomorphic.* [Online]
Available at: https://eprint.iacr.org/2012/144

Fauzi, P., Hovd, N. M. & Raddum, H., 2022. *On the IND-CCA1 Security of FHE Schemes,* s.l.: Cryptography 2022.

Gentry, C., 2009. *Fully Homomorphic Encryption Using Ideal Lattices,* s.l.: s.n.

Gentry, C. & Halevi, S., 2011. Implementing Gentry's Fully-Homomorphic Encryption Scheme. *Advances in Cryptology – EUROCRYPT 2011.*

Gentry, C., Halevi, S., Dijk, M. v. & Vaikuntanatha, V., 2010. Fully Homomorphic Encryption over the Integers. *Advances in Cryptology – EUROCRYPT 2010.*

Gentry, C., Sahai, A. & Waters, B., 2013. *Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based,* s.l.: s.n.

Gilad-Bachrach, R. et al., n.d. *CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy.* s.l., Machine Learning Research.

[24] https://www.orangemantra.com/blog/why-java-remains-a-top-enterprise-software-framework/

Hesamifard, E., Takabi, D. & Ghasemi, M., n.d. *Deep Neural Networks Classification over Encrypted Data.* s.l., 2019.
Li, B. & Micciancio, D., 2020. *On the Security of Homomorphic Encryption on Approximate,* s.l.: Cryptology ePrint Archive.
Li, D., Liao, X., Wu, J. & Le, J., 2020. Privacy-preserving self-serviced medical diagnosis scheme based on secure multi-party computation. *Computers & Security,* Volume 90.
Lyubashevsky, V., Peikert, C. & Regev, O., 2010. On Ideal Lattices and Learning with Errors over Rings. *Advances in Cryptology – EUROCRYPT 2010.*
Microsoft, 2023. *Microsoft SEAL.* [Online]
Available at: https://www.microsoft.com/en-us/research/project/microsoft-seal/
[Accessed 2023].
Munjal, K. & Bhatia, R., 2022. A systematic review of homomorphic encryption and its contributions in healthcare industry. *Complex & Intelligent Systems.*
Naehrig, M., Lauter, K. & Bos, J. W., n.d. *Improved Security for a Ring-Based Fully,* s.l.: Cryptology ePrint Archive.
OpenFHE, 2023. *open FHE.* [Online]
Available at: https://www.openfhe.org/
[Accessed 2023].
Pyfhel, 2023. *PYthon For Homomorphic Encryption Libraries.* [Online]
Available at: https://pyfhel.readthedocs.io/en/latest/
Ruiwen, C., 2022. *Credit Scoring Model of Small and Medium-sized Enterprises over Encrypted Data.* s.l., IEEE 10th Joint International Information Technology and Artificial Intelligence Conference.
Sendhil, A. & Amuthan, A., 2021. Contextual fully homomorphic encryption schemes-based privacy preserving framework for securing fog-assisted healthcare data exchanging applications. *International Journal of Information Technology.*
Soyata, T. & Kocabas, O., 2020. Towards Privacy-Preserving Medical Cloud Computing Using Homomorphic Encryption. In: *Virtual and obile Healthcare.* s.l.:s.n., p. 33.
TenSEAL, 2023. *TenSEAL.* [Online]
Available at: https://github.com/OpenMined/TenSEAL
Yang, Y., Xiao, X., Cai, X. & Zhang, W., 2019. A Secure and High Visual-Quality Framework for Medical Images by Contrast-Enhancement Reversible Data Hiding and Homomorphic Encryption. *IEEE Access,* Volume 7.