FYP-II

Final Evaluation Report

F22-112-R-CuratioMorbi

Title: Cryptographically secure federated learning for healthcare consortia

Members:

Minahil Irshad	19I-2178
Malaika Waheed	19I-0726
Ibrahim	19I-0508

Supervisor:
Ms. Hina Binte Haq
Co Supervisor:
Dr. Zainab Abaid

Anti-Plagiarism Declaration

This is to declare that the above FYP report produced under the:
Title: F22-112-R-CuratioMorbi

It is the sole contribution of the author(s) and no part hereof has been reproduced on **as its** basis (cut and paste) which can be considered as **Plagiarism.** All referenced parts have been used to argue the idea and have been cited properly. I/We will be responsible and liable for any consequence if violation of this declaration is determined.

Date: Oct-24-2022

Student 1:
Name: Minahil Irshad
Signature: Minahil Irshad

Student 2:
Name: Malaika Waheed
Signature: Malaika Waheed

Student 3:
Name: Ibrahim
Signature: Ibrahim

Supervisor (Faculty)
Name: Maam Hina Binte Haq
Signature:

Authors' Declaration

This states Authors' declaration that the work presented in the report is their own, and has not been submitted/presented previously to any other institution or organization.

Abbreviation Table

FL	Federated Learning

HE	Homomorphic Encryption
SMC	Secure Multi-party Computation
DP	Differential Privacy
ACIQ	Analytical Clipping for Integer Quantization
HIPAA	Health Insurance Portability and Accountability
GDPR	General Data Protection Regulation
NbAFL	Noise before aggregation of Federated Learning Model
SGD	Stochastic Gradient Descent
ECC	Elliptic curve Cryptography

Abstract

Federated learning enables its clients to do collaborative learning while keeping their data on their local systems. This however does not completely minimize the risk of data

breaching. We propose a cryptographically secured federated learning model. We will develop this system especially for a consortium of hospitals to undertake federated learning. We present a new approach towards achieving privacy and anonymity in federated learning. This federated learning paradigm shares learning, and not the data. Hospitals on the network train models locally on their dataset and share their learnings with the server. Server aggregates the weights by taking federated averages to get the weights of the final model. The final model weights are shared with the hospitals so they can use it for the prognosis of the diseases which in our case is scoped to 14 chest diseases only.

Executive Summary

This report provides an overview of our project titled “Cryptographically secure federated learning for healthcare consortia” which is a Centralized Federated Learning setup for prediction of Chest diseases. The system is cryptographically secured to protect from attacks on the shared weight in order to ensure privacy. The report starts with an introduction of the project and is followed by the research gap. Introduction contains the background of the project and the basis on which our research is built upon. The research problem statement identifies the areas of our research and the research gap that exists in a Cryptographically secure federated learning setting. The project scope is defined to narrow down the type of Federated Learning being used, the name and nature of the dataset, and the type of Machine learning model that will be used in the development part of our project. The assumptions that will be considered during the research and development of this project are also mentioned.

A literature review of six papers is given to elaborate upon the work that has been done so far in this domain. A summary of these papers is provided in order to give a brief description of the techniques that have been proposed by them. All these papers have been critically analyzed and the relationship with our project has been identified to give a clear view of the research gap.

The report further contains a use case diagram that specifies the uses and behavior of our system. High level use cases are designed for each use case to give a better insight into the use cases. Software Requirements Specifications have been included in this document to define the requirements and important part of our entire system. This part also enlists the major features and functionalities of our project.

The proposed approach explains the protocols, workflow, constraints and evaluations of our project. Diagrams have been added for better understanding and illustration. This part defines the development of our system, the implementation of our proposed solution to the identified research gap and incorporation of our solution in the system. Finally the report concludes with a “Conclusion and Future Work” paragraph.

References for the papers included in the Literature Review are given under the heading “References”.

Table of Contents

Chapter 1 Introduction

1.1 Problem Domain

Artificial Intelligence has highly transformed the industry. However, certain AI models require large amounts of datasets from multiple sources. The transfer of sheer data to a centralized server generates traffic over the network. It does not work well in applications where real-time decisions need to be made like self-driving cars. Furthermore, legal constraints and rising concerns of privacy prevents organizations from sharing their data. These problems are addressed by the advent of Federated Learning. A technique that enables a large number of users to locally train a model on local datasets. The knowledge (trained weights) of these models is then shared on a network and it is aggregated into a bigger model. All of this is done without sharing any data.

Though FL ensures that data remains anonymous to the silos part of the system, there is still a risk of various privacy attacks during the training process. Our proposed research is based on the existing solution for privacy preservation in a FL model, the overheads they involve and finding out a solution to overcome them. We will develop a **horizontal cross silo Federated Learning model** that will provide prediction for **chest diseases** in a healthcare setup. For the prevention of privacy leakage we propose **Cryptograms** as a solution.

1.2 Research Problem Statement

A Federated Learning model is vulnerable to attacks that can be made on the weights of the model to compromise privacy. Following are the possible attacks:

- ***Data Poisoning Attack:*** The training data of a machine learning model is polluted. The integrity of the model is affected because tampering with the training data impacts the model's ability to output correct predictions.
- ***Adversarial Attack:*** Generate a wrong result by designing a specific input. Aggregator is not familiar with the local update modes thus are not able to detect anomalies

- ***Inference Attack:*** Inference attacks aim to reveal secret information by probing a machine learning model with different input data and weighing the output.

The existing techniques that protect the Federated learning model from these attacks are either computationally expensive, involve complex operations, place communication overhead or are less accurate. The examples include Secure Multi Party Computation, Differential Privacy, and Homomorphic Encryption.

1.3 Project Scope

- Convolutional Neural Networks (CNN) model is used by the hospitals for training the local models on their respective datasets.
- The ChestMNIST dataset will be used by our system. It is a multi-class binary dataset that predicts 14 types of chest diseases.
- Centralized Federated Learning will be used by the system to aggregate the weights sent by the hospitals. All hospitals connected to the system will be headed by the regulatory body.
- Horizontal Federated Learning protocols will be used to aggregate the final model. This means that all hospitals on the system will train their local model on the same features.
- Cross-Silo Federated Learning will be used in our system. In a cross-silo setting, organizations cooperatively train a global model with their local data.
- Cryptograms will be used as the encryption scheme between the hospitals and the regulatory body. This is a relatively new encryption method that will be developed in our project to safeguard the shared weights from adversaries.

Justifications

We use a horizontal federated learning scheme in our contribution as we use only one dataset i.e. ChestMNIST dataset. We divide this dataset into three parts for different clients and since all the parts have the same features space, we are

bound to use horizontal federated learning. As our clients in this case are hospitals which are organizations so the type of federated learning is cross silo.

Assumptions

The following assumptions have been made while developing our project and will therefore not be catered to in the project:

- The end systems at the hospital, regulatory body and doctor have been secured according to ISO 27001.
- The dataset being used by the hospital for training the local model is authentic and accurate.
- All hospitals on the system use the protocols developed in the project to maintain a standard communication.
- No hospital or doctor on the network is involved in any malicious activity.

Chapter 2 Literature Review

2.1 Differentially Private Secure Multi-Party Computation for Federated Learning in Financial Applications

2.1.1 Summary

The proposed approach combines differential privacy with safe multi-party computation for federated learning. FL reduces the risk of information leakage but the adversarial attacks can be done to recover the true form of data. The preservative privacy measures are provided by DP and SMC that add multiple protective layers to make the reverse engineering very difficult to nearly impossible. DP protocol adds noise to the present weights so the data is not recovered. SMC protocol adds accuracy that is reduced due to the DP.

They provided an anti-data-sharing fraud detection system and additional security layers to secure customer data and transactions. Even with adversarial participation in the system, the cryptographic algorithms used make data retrieval extremely challenging and practically impossible. This protocol is meant for the non specialist audience that has a computational background but it does not necessitate prior understanding of security, encryption, privacy, or distributed learning.

2.1.2 Critical Analysis

The approach tends to solve the problem of data privacy with two protocols i.e. Differential privacy and secure multi party computation. Differential Privacy improves the data privacy by adding noise to the present weights which in turn results in the compromising of data accuracy. Secure Multi party computation helps with computation on private distributed data without exposing it. It works on the semi honest adversary who follows the protocol specification, but may attempt to learn honest parties' private information from the messages it receives, or to collude with other parties to learn private information.

2.1.3 SWOT Analysis

STRENGTHS	WEAKNESSES
· Federated learning enables clients in its system to do collaborative learning while keeping their data on their own local systems. · Federated learning along with differential privacy which adds a layer of randomness to the data that makes reversal attacks like database reconstruction and reidentification attacks useless to reveal private data. · To cater to the accuracy loss generated by differential privacy and data collusion an additional layer for accuracy-reducing noise is introduced which is secure multiparty computation. · This privacy-preserving protocol is also for the audience that has no prior knowledge about security, encryption, privacy, distributed learning, federated learning, differential privacy, and secure multi-party computation.	· Although multiparty computation introduces no loss to the collaborative training but differential privacy component does inverse proportion to the epsilon privacy loss parameter. · With more clients the training loss and model accuracy worsens dramatically. · A naive classifier would always result in returning False and would achieve a misleading 99.8% accuracy of prediction. · Smaller selections of privacy loss parameter epsilon are better for privacy but harm the accuracy of the learned model.

OPPORTUNITIES	THREATS
· The dataset used is to distinguish the fraudulent system from genuine ones in financial applications. Different kinds of datasets from different domains can be used for example in health care applications, distinguishing between fatal and harmless diseases. · Researching and implementing different parameters that can help reduce the loss but enhance the accuracy of the system. · Strong encryption techniques that make data reversal attacks impossible.	· Accuracy is compromised if the internal parties are dishonest. · The average calculated would not be error-free.

2.1.4 Relationship to the proposed work

The proposed work is similar to our work in that both of them highly focus on data privacy and data accuracy. The paper discusses a solution using financial applications as an example, whereas our project uses a medical dataset from MNIST to work towards reaching privacy in a medical setting. The technique uses Diffie Hellman key agreement protocol to establish common randomness. Our approach to develop Cryptograms as the encryption scheme works on achieving data privacy uses Diffie Hellman ensuring that the aggregated model's accuracy is not threatened.

2.2 BatchCrypt: Efficient Homomorphic Encryption for Cross-Silo Federated Learning

2.2.1 Summary

The paper presents an advanced approach to Homomorphic Encryption in Federated Learning. It highlights the bottlenecks of the existing privacy preserving techniques for Federated Learning model weights. Secure multi-party computation involves complex Machine Learning algorithms and is computationally expensive. Differential Privacy introduces noise in its system and thus affects the accuracy of its model. Furthermore it exposes gradients to the server which weakens the privacy of the overall FL model. Similarly secure aggregation weakens the privacy by exposing the results to a third party. Homomorphic encryption ensures strong privacy but at the tradeoff of communication and computation overhead.

The proposed approach caters the drawbacks of homomorphic encryption by making amendments to the conventional HE. Instead of encrypting individual gradients a batch of quantized gradients is encrypted. Quantization is required as gradients are signed floating values and cannot be arranged according to their exponents. Quantization is done using the approach dACIQ which is an extended version of ACIQ, a state of art clipping technique for ML over centralized data. This technique allows an optimal clipping threshold before quantization, reducing minimum cumulative error. Quantized value’s true form is translated into two's complement's form with two’s complements codec. Batch Manager is in charge of batching and debatching gradients in their two’s complement form, it remembers data’s original shape before batching and restores it during unbatching.The batching of gradients minimizes computation and communication overhead saving CPU usage and power consumption.

2.2.2 Critical Analysis

The approach is an attempt to solve the issues involved in HE as an FL privacy preserving technique. It preserves the additive factor of Homomorphic Encryption and

can be applied to existing ML algorithms and optimization techniques. Gradients are quantized before encryption to arrange them according to their corresponding model weights. This method allows direct dequantization of gradients even if the number of values aggregated are not known. Batching of gradients reduces the amount of encryption operation by avoiding the possibility of mismatch of plaintext and ciphertext. Furthermore this method is resilient to overflow in addition so the encrypted gradients are not corrupted and accuracy loss is prevented.

Despite reducing computational expense of HE this scheme has certain limitations. It introduces additional steps of quantization of gradients that add up to the complexity of the model. Per layer gradient range and size is sent to the aggregator for calculating layer-wise clipping thresholds before the actual quantization and training at the client side begins. Also, when quantization is applied to ML models the convergence of weights becomes difficult and slow. Another weakness is that since gradients are batched this technique will work best when there is a large amount of dataset and will become inefficient with smaller dataset.

This method caters to the problem of large data transfers and long training time of conventional homomorphic encryption. But the proposed algorithm “Batch Crypt” to quantize, batch encode and perform analytical quantization co-designs these steps in a complex manner. Additionally, given that this method is effective with only large datasets, the proposed solution in overall not the best privacy preserving technique in Federated Learning

STRENGTHS	WEAKNESSES
● BatchCrypt preserves the additive factor of Homomorphic Encryption. ● Batching of gradients reduces the amount of encryption operation by avoiding the possibility of mismatch of plaintext and ciphertext. ● This method is resilient to overflow in addition so the encrypted gradients are not corrupted and accuracy loss is prevented.	● Introduces additional steps of quantization of gradients that add up to the complexity of the model. ● When quantization is applied to ML models the convergence of weights becomes difficult and slow. ● Works best only when the dataset is large.
OPPORTUNITIES	**THREATS**
● Does not involve large data transfers and long training time that reduces the load on the network and time of the process. ● Can be applied to existing ML algorithms and optimization techniques.	● Convergence might become slow due to quantization techniques ● Will show no effective advantage in terms of computational expense if the dataset is small over conventional Homomorphic encryption.

2.2.3 Relationship to the proposed work

We propose the implementation of cryptograms as an encryption algorithm for our Federated Learning model. Similar to BatchCrypt, our approach solves the problem of computational expense and data inflation. Our approach is better in the aspect that it does not require complex steps of quantization unlike the approach of BatchCrypt. Moreover, it is relatively more efficient with all sizes of datasets. The datasets used in this paper are MNIST, CIFAR and Fashion-MNIST. We, on the other hand, are focusing on FL's use in medical science and hence have used ChestMNIST dataset to predict 14 different chest diseases ensuring privacy of the data without compromising on the accuracy of the AI model being used.

2.3 E-Fed: Communication efficient multi-party computation enabled federated learning

2.3.1 Summary

The paper presents an approach CE-Fed against the possibility of extracting private and confidential information from the shared models even-though sharing of raw data is prevented by federated learning. The approach is built on the Secure MultiParty Computation technique and addresses the issues of communication and computation overhead when implemented on a large-scale decentralized federated learning system. A conventional MPC technique works by each dividing a S model into n secret shares (n is the number of clients) and sending a secret share to each client. For example if there are 6 clients in the system there will be an $(6 \times (6 - 1) \times 2) = 60$ exchange of messages.This increases the communication cost.

CE-Fed solves this problem by introducing a hierarchical model aggregation module. The clients located near to each other are grouped together and a group leader is elected based on the latency. All the members of the groups train their datasets and intra-group aggregation is performed using the MPC protocol. The group leader shares the aggregated models with each other instead of all clients sharing their models with each

other to reduce communication overhead. From the group leaders a subset is chosen as the aggregation committee which performs inter group aggregation. Then, the group leaders broadcast the intergroup model to their followers in their group. The whole process of intra-group model aggregation and inter-group model aggregation are repeated until the model converges.

2.3.2 Critical Analysis

CE-Fed is tested on two forms MNIST data setting:

1. Independent and Identically Distributed (IID)
2. Non- Independent and Identically Distributed (IID) setting

This approach reduces the number of message exchanges in both types up to 90%. In IID data setting each client has balanced and identical data points while in non-IID data among the clients is distributed with different proportions. However in contrast with other approaches like Differential Privacy the computation cost is high CE-Fed as still a certain amount of message exchanges needs to be done among the clients.

The CE-Fed model when tested with MNIST datasets had good accuracy. The accuracy is not affected by the addition of more clients making it scalable.

Groups of clients are made according to their location. The clients nearby are grouped together as part of CE-Fed hierarchical aggregation in order to reduce communication expenses. However this approach fails when clients are located in different geographical sites, where different links in the system have significantly different latencies.

As part of the hierarchical aggregation, group leaders are elected which aggregate models in their group of clients and share them with the aggregation committee. These leaders are selected based on their latencies. If a leader does not respond while the intra group aggregation a new leader, one with lowest latency, is elected. This can add delays in the process and intra group aggregation.

STRENGTHS	WEAKNESSES
• This approach reduces the number of message exchanges in both types up to 90%. • The CE-Fed model when tested with MNIST datasets had good accuracy. The accuracy is not affected by the addition of more clients making it scalable.	• Computation cost is high CE-Fed as still a certain amount of message exchanges needs to be done among the clients in contrast with other approaches like Differential Privacy. • This approach fails when clients are located in different geographical sites, where different links in the system have significantly different latencies.
OPPORTUNITIES	**THREATS**
• This approach reduces the communication cost incurred in Multi Party Computation enabled Federated Learning • This approach avoids sharing the model parameter from each client to all the other clients in the network making it less vulnerable to malicious clients.	• Delays in response from leaders selected for intra group aggregation intra group aggregation. • The approach is applicable with the assumption that each client is at a fixed position and the estimated latencies between each client is known. If the assumption fails the leader selection process might become ineffective.

2.3.3 Relationship to the proposed work

Our approach of using cryptograms as a privacy preserving technique in FL is also an attempt to address the computation and communication overheads. Cryptograms improve communication without compromising the accuracy of the system. Unlike CE-Fed they don't have the constraint that clients need to be geographically in the same region. They also don't involve additional steps in FL training like grouping and electing leaders which minimizes the possibility of any kind of delays. The dataset used in this paper is from MNIST and is related to fashion whereas we are working on medical FL settings using ChestMNIST dataset to predict chest-related diseases.Our approach is similar ro CE-Fed in the the aspect that both of these schemes are cross silo.

2.4 The future of digital health with Federated Learning

2.4.1 Summary

The study presents a viewpoint on the potential application of FL in the field of digital health care, outlining both its advantages and disadvantages. Without the centralization of medical data, the challenge of acquiring the data is addressed using the collaborative learning FL offered. The benefit of relocating the model rather than the data is that the connected institutions who must use the data don't have to duplicate the high-dimensional, storage-intensive medical data.FL enables collaborative learning without exchanging the data but the consensus model gradients and parameters are transferred such that the private data remains inside the firewalls of its institute.The use of FL has been successful in the realm of medicine for example in medical imaging, including whole brain segmentation in MRI, segmentation of brain tumors, evaluation of mammograms, and patient prediction of breast cancer and melanoma. Despite the location, establishing FL on a worldwide scale could lead to high-quality clinical decisions. Establishing FL on a global scale could result in high-quality clinical decisions

despite the location. Researchers and AI developers can conduct a thorough study on the vast knowledge extracted out from 67the collection of real medical data. However, the failure cases cannot be studied in detail as it is impossible to look at the real data. Challenges are data quality, bias and standardization. The data heterogeneity poses a challenge to FL algorithms and strategies like FegProx and FedAvg as medical data is diverse in terms of modalities, dimensionality and characteristics. A global optimal solution may not be optimal for an individual local participant. There are risks of privacy preserving measures as FL cannot provide full assurance. FL systems avoid sharing medical data among the involved institutes. But the shared information can still be leaked by model inversion of the model updates themselves or due to reverse engineering adversarial attacks.

2.4.2 Critical Analysis

The paper provides an overview of federated learning, including its types, benefits, challenges, and application examples. The federated learning technique is communication efficient as it brings models to the data rather than sending the whole dataset over the network. FL needs multiple privacy preserving measures as FL strategies aren't enough to cater the risks of information leakages. The protective layers are needed for the data privacy and security such as advanced encryption of model submissions, secure authentication of all parties, traceability of actions, differential privacy, verification systems, execution integrity, model confidentiality and protections against adversarial attacks. No matter how the FL studies can help researchers and AI developers, they cannot deduce the inefficiency of the model on an individual local participant with just the bare knowledge regarding disease as the real data is encrypted.

2.4.3 SWOT Analysis

STRENGTHS	WEAKNESSES
· To avoid privacy leakage, the first	· There is some sacrifice in efficiency

cloud runs ciphertext perturbation before sending it to the second server and the aggregated model is sent back to the clients for decryption individually · Experimental result in benchmark datasets showed that competitive accuracy was achieved · It performed better in terms of efficiency than the FL models secured by homomorphic encryption · The training protocol in each client do not reveal any information as the other clients can learn the local model · Clients interact with first server which cannot learn anything about the encrypted models since it does not have either the master key or the private key · The first server creates random masks in each iteration to communicate with the second server, which holds the master key. This randomness ensures the security of models · Security is much improved than single key homomorphic encryption without any significant loss in accuracy	due to multi-key encryption · The protocol does not deal with a malicious client · The system is only secure when all clients are secured

OPPORTUNITIES	THREATS
· This method can be used in large scale applications as it performed well on benchmark datasets, improving security without any accuracy loss like the models that use differential privacy	· A malicious client can affect the local and hence the global model by training the local model on malicious data, adversely affecting the overall accuracy

2.4.4 Relationship to the proposed work

Our proposed work is in coordination with the federated learning model and the cryptographic tools that provide the necessary protective layers for the data privacy by not compromising on its accuracy that federated learning lacks. As it is discussed that a simple FL model has privacy concerns as the cyber attacks on such systems can result in privacy of the data being compromised. We will be developing Cryptograms on our project that is a new encryption scheme. It will be integrated with the FL model to ensure that the shared model is safe from any adversaries. We will be working to ensure privacy without compromising the accuracy or efficiency of the system.

2.5 Secure Neural Network in Federated Learning with Model Aggregation under Multiple Keys

2.5.1 Summary:

This paper shares a new approach to secure a Neural network. Instead of using simple Federated learning to aggregate the model or using conventional Homomorphic Encryption, the paper suggests a better approach to enhance the security of the network. Homomorphic Encryption only supports a single key. The paper sheds light on a better implementation of Homomorphic Encryption that would secure the Neural network using multiple keys. “Double-Trapdoor Encryption” is adapted in this paper. The implementation depends on non-colluding cloud servers. The client uploads the encrypted local models on the first cloud server, that can be decrypted by the second server using one trapdoor. To avoid privacy leakage, the first cloud server uses ciphertext perturbation before sending data to the second server. The paper discusses why Federated Learning became a hot topic due to two reasons. First is data fragmentation and isolation and the second catering to the public’s concerns regarding data protection and privacy. The paper also discusses that the encryption schemes used to secure federated models can generally be divided into three classes: (i) Multiparty Computation (ii) Differential Privacy and (iii) Homomorphic Encryption. The paper then compares these classes and explains how Homomorphic Encryption is perfect for applications where privacy preserving outsourced computation is needed. The only problem with Homomorphic Encryption is that its efficiency is low and it only supports a single key. The difficulty in using multiple keys has its own complications. If the participants of a Federate Learning network choose different keys, the cloud server must perform model aggregation on encrypted data under different keys.

The idea of multi-key addition relies on adding noise to the models in the form of ciphertext based on the homomorphic feature of BCP scheme in one of the cloud servers. The model and the noise are aggregated on the other cloud server based on the double-trapdoor feature of the BCP scheme. The first server then removes the noise from the models and then sends the aggregation result back to the clients. There are N data clients, two cloud servers and one key generation center (KGC). The scheme is based on the horizontal distribution of the data. That is to say, the data samples of each data party are different, but the features are the same. Each client uses the local data to train the model and then uses the parameters from the KGC to generate their own public and secret

key pair, and uses their public keys to encrypt the model parameters and upload them to the first cloud server. The first server then generates random masks in each iteration and then communicates with the second server that has the master key and sends the model to each of the clients individually. This creates a trap door and the first server can learn nothing about the encrypted model nor the client communicate with the aggregating server directly, which results in a much enhanced security than simple, single-key homomorphic encryption.

2.5.2 Critical Analysis:

The approach attempts to solve the issue of preserving privacy in a federated learning model. Introducing multiple keys increases security for the local nodes and overall security of the system. Normally adding noise drops the accuracy of data substantially but this method proves to ensure comparable accuracy. Even if security is enhanced and accuracy is somewhat comparable to the other encryption methods used to secure Federated Learning models, there is a high computational expense when the system is dealing with multiple keys. From generation of key pairs to adding additional loads of separate encryption and decryption operations for each node, the computational expense is a very large factor.

2.5.3 SWOT Analysis

STRENGTHS	WEAKNESSES
· Federated Learning solves the problem of data governance by training algorithms through collaboration without any data	· Efforts to solve problems by Federated Learning require agreements to define the scope, aim and technologies used which, since it is still novel, can be difficult to list

exchange itself · Federate Learning enables to use the potential of AI and ML without moving the patient data beyond the firewalls of the residing institution · Recent research proves that AI models trained by Federated Learning can achieve comparable performance to the models trained on isolated, single-institutional data · Models that are trained through Federated Learning perform much better against the anomalies as their training involves high data variability due to collaborative learning · Centralizing or sharing data can have regulatory, ethical, and legal challenges related to privacy and data protection, making Federated Learning a viable solution in such applications · In a FL setting, each data controller can define its own data governance and data access processes along with associated privacy policies for the training and validation phase	down · An up-front effort is required for an agreement regarding the data structures, annotations and protocols that will be used to ensure that the information presented to the collaborators is in common understood formats · Institutions will be required to invest in on-premise computing infrastructure or cloud services with standard data formats to ensure that the ML models can be trained and evaluated seamlessly · FL based solutions limits data visibility which implies that it cannot be investigated or visualize all the data on which the model was trained on which means it is not possible to look on an individual failure case to understand why the model performed poorly or gave an unexpected result
OPPORTUNITIES	**THREATS**

· A good implementation of Federated Learning can enable resulting models to make precise, unbiased decisions while respecting data governance and privacy concerns making it a great solution in healthcare · FL models used in digital health will be much more sensitive to an individual's physiology and detection of rare diseases while respecting governance and privacy concerns · Federated Learning has the potential to overcome the limitations of approaches that need a large of pool of centralized data · Control on local data for each controller opens new opportunities for Federated Learning, enabling it for large scale inter-institutional applications or novel research on rare diseases where incident rates are too low making single institutional dataset too small for learning purposes · Federated Learning enables disruptive innovations in the future by enabling to capture high data variability to analyze a problem across different demographics · Patients could get high quality	· Although institutions can remain anonymous to each other in a FL setting, but under certain conditions the models can start memorizing information which makes it necessary to enhance privacy by placing some form of encryption mechanism · FL does not solve the problems that are inherent to machine learning as quality learning would still depend on data quality, bias and standardization · Data coming from different sources will be heterogeneous in nature due to different modalities, dimensionalities, and characteristics, therefore the contributors are required to be on the same page in recording and training data as factors such as demographics, acquisition differences and brand of devices can differ the results even if common protocols are used · FL does not solve all privacy issues and taking measures to solve those can result in performance as a tradeoff · All contributors taking part in a FL setting must trust each other as local training on each node has an affect on the global model, this makes it hard to use FL for applications where

diagnosis and treatments regardless of their treatment location FL solutions are scaled globally · FL solutions can reduce the concerns of being a data donor, since people can be assured that their data will remain with their own institution its access can be revoked · Once a proper FL setting is there in place, it can improve the treatment of patients with better results reducing the costs of expensive tests	inter-organization trust is a problem

2.5.4 Relationship to the proposed work:

Our proposed implementation is an encryption method called Cryptograms. This method tends to resolve the problem of ensuring privacy without any major trade off in the form of accuracy or computational expense.

2.6 Federated Learning with Differential Privacy:Algorithms and Performance Analysis

2.6.1 Summary:

The paper proposes an approach to prevent the information leakage by using DP with an FL model. The DP protocol works by adding noise to the present weights. This uses the DP protocol on noising before aggregated FL model.Firstly, NbAFL is proven to check if

it satisfies DP under distinct protection levels by adding different variances of artificial noises. Secondly, the theoretical convergence bound of loss function of trained FL models in NbAFL is developed. The key points include the tradeoff between a convergence performance and privacy protection levels, for fixed privacy protection level increasing the number of clients participating in FL can improve convergence performance and the communication rounds for protection level are optimal. Thirdly, a random scheduling strategy is proposed that randomly selects the clients for the participation in aggregation. This maintains the above three points discussed.

2.6.2 Critical Analysis

The study provides an overview of the existing DP protocols like local differential privacy and distributed SGD and differential privacy meta learning and their applications. The proposed approach NbAFL strengthens the privacy levels by adapting to different variances of artificial noises. However, the approach provides a strong privacy measure, the addition of noises in the parameters of clients weakens the accuracy of the final aggregated model.

The approach NbAFL is a mechanism to satisfy DP requirements for an FL but the strengthening of privacy comes at the cost of poor convergence performance as stated in the paper. Moreover an optimal number of clients are needed for aggregation to be identified as an attempt to improve the convergence performance of the Federated Learning model.

2.6.3 SWOT Analysis

STRENGTHS	WEAKNESSES
· Federated learning is preserving the private data of its clients from being exposed to external adversaries. · To prevent information leakage differential privacy is used.	· K-random scheduling brings down the little performance loss

· Layer of artificial noise is added namely noise before model aggregation federated learning (NbAFL). · Better convergence performance is achieved for a given protection level. · K-random scheduling obtains better trade offs than the normal selection policy. · Existence of an optimal value of K that achieves the best convergence performance at a fixed privacy level.	
OPPORTUNITIES · Implementing and studying different strategies for scheduling. · Combine DP with other techniques to analyze the effect on performance analysis.	**THREATS** · If the server is not honest, the data would not be secure. · External adversaries may do a model inversion attack to extract the client's private information. · Privacy leakage can also happen in the broadcasting phase by analyzing the global parameter

2.6.4 Relationship to the proposed work

The proposed approach enables the prevention of information leakage in NbAFL.Our proposed implementation of cryptograms also solves the information leakage problem in FL.

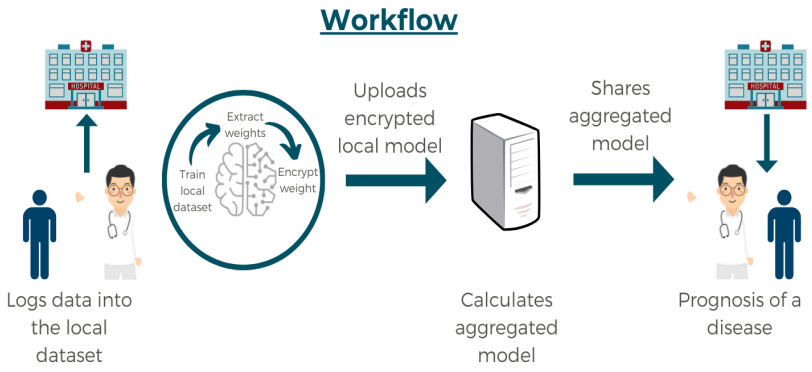
Paper	**Year**	**Mechanism**	**Advantages**	**Disadvantages**
CE-Fed: Communication efficient multi-party computation enabled federated learning.	2022	Multi party computation	Reduce the communication overhead and scalability MPC protects final results from being exposed to the server	Computationally expensive
BatchCrypt: Efficient Homomorphic Encryption Cross-Silo Federated Learning	2020	Paillier Homomorphic Encryption	No learning accuracy loss Allows gradient aggregation to be performed on ciphertexts (without decryption first) Faster rate of data transfer	Large volume of data to enable to enable faster machine codes and massive parallelism Computationally expensive

Differentially Private Secure Multi-Party Computation for Federated Learning in Financial Applications	2020	Differential Privacy and Secure Multi-Party Computation	MPC protects final results from being exposed to the server Use of differential privacy allows to remain anonymity between clients despite that g client's encryption occurred with a colluding counterparty	Adding randomness to the collected data preserves user privacy at the cost of accuracy Computationally inefficient
The future of digital health with Federated Learning	2020	Federated Learning	Addresses data privacy and governance challenges Better than the centralized training system	Communication- Efficiency is slow as a massive number of institutes/ devices are in the network. Systems Heterogeneity includes variability of hardware, network connectivity and power supply.

				Statistical Heterogeneity is the data collected from various sources is non identical. Privacy Concerns like attacks on the shared information.
Secure Neural Network in Federated Learning with Model Aggregation under Multiple Keys	2021	Federated Learning with multi-key Homomorphic Encryption	Better security than conventional encryption schemes Comparable Accuracy to other Federated Learning implementations	Computationally Expensive (increased encryption and decryption operations) Need to rely on a third trusted party to produce the public and private keys

Chapter 3- Proposed Approach

3.1 Workflow



3.2 Cryptograms

Cryptograms is the solution we provide for the problem of ensuring privacy in a FL setting. It is an encryption technique that has not been previously used with Federated Learning. This technique was proposed in a research paper (Shahandashti & Hao, 2016) where it was implemented with E-voting to ensure voting's privacy. We hope to achieve competitive accuracy compared to other techniques, but with a lower computational expense. This technique is based upon the principles of the elliptic curve. An elliptic curve over real numbers may be defined as the set of points (x,y) which satisfy an elliptic curve equation of the form:

$y2 = x3 + ax + b$, where x, y, a and b are real numbers.

The principles of elliptic curve cryptography which have been used for the implementation of cryptograms are following:

1. Prime Galois field GF(p): The field over which elliptic curve is defined is called Prime Galois field. A prime number p determines the order of the field. The field contains integers from 0 to p-1, and arithmetic operations are performed modulo

"p". The value of p which we have used is 0xffffffff00000001000000000000000000000000ffffffffffffffffffffffff.

Point addition: Point addition in elliptic curve cryptography is another core concept. It involves addition of two points to produce a third point on the curve. The addition of points on an elliptic curve is associative, commutative, and has an identity element O (the point at infinity).

- Associative: (P + Q) + R = P + (Q + R)
- Commutative: P + Q = Q + P
- Additive Identity: P+O=P and P+infinity=P
- Additive Inverse: P+(-P)=O

The formula for point addition is

$$x_r = \lambda^2 - x_p - x_q$$
$$y_r = \lambda(x_p - x_r) - y_p$$

where

$$\lambda = \frac{y_q - y_p}{x_q - x_p}$$ (Point addition: P+Q)

$$\lambda = \frac{3x_p^2 + a}{2y_p}$$ (Point doubling: P+P)

Scalar Multiplication: Scalar multiplication in elliptic curve cryptography is done by repeatedly adding a point to itself. A base point is added to a scalar number of times to itself to find a resulting value. This means 3P=2P+P and 2P=P+P.

Cyclic group of an elliptic curve: The points on an elliptic curve together with O have cyclic subgroups. Given a point P(5,1) defined over the elliptical curve $y^2 = x^3 + 2x + 2$(field order=17) the cycle for it is as follows:

$2P = (5,1)+(5,1) = (6,3)$
$3P = 2P+P = (10,6)$
$4P = (3,1)$
$5P = (9,16)$
$6P = (16,13)$
$7P = (0,6)$
$8P = (13,7)$
$9P = (7,6)$
$10P = (7,11)$
$11P = (13,10)$
$12P = (0,11)$
$13P = (16,4)$
$14P = (9,1)$
$15P = (3,16)$
$16P = (10,11)$
$17P = (6,14)$
$18P = (5,16)$
$19P = \mathcal{O}$

Inverse of Point: To negate a point P on an elliptic curve x-coordinate is kept the same while y is negated. So if P=(x,y) then -P=(x,-y).

The Cryptogram Algorithm:

To implement the algorithm an elliptic curve, a base point and a prime number for the Prime Galois field are initialized. We have implemented **curve p-256** for elliptic curve operations. The standard parameters for curve p-256 are used

Name	Value
p	0xffffffff00000001000000000000000000000000ffffffffffffffffffffffff
a	0xffffffff00000001000000000000000000000000fffffffffffffffffffffffc
b	0x5ac635d8aa3a93e7b3ebbd55769886bc651d06b0cc53b0f63bce3c3e27d2604b
G	(0x6b17d1f2e12c4247f8bce6e563a440f277037d812deb33a0f4a13945d898c296, 0x4fe342e2fe1a7f9b8ee7eb4a7c0f9e162bce33576b315ececbb6406837bf51f5)

1. Weights extracted from three clients are stored in w1, w2 and w3

   w1←weights of client 1

   w2←weights of client 2

   w3←weights of client 3

2. A point on the curve is found using P1 = k * G where k is a random variable between 1 and p-1(p is the prime number for the Prime Galois field).
3. Negations/Inverse of points G (Base Point) and P1(Generated point in step 2) are taken and stored as nG and nP1.
4. R1, R2 and R3 are calculated as salt values to add randomness to the weights.

   R1 = r1 * P

R2 = r2 * P

R3 = r3 * nP

r1, r2, r3 are random numbers between 1 and p. They are found within this range to ensure that the points R1, R2, and R3 remain within the field. “r1”, “r2”, and “r3” are multiplied with P using scalar multiplication of ecc. The same salt value is added to all the weights of a particular model individually. Eventually, when the weights are aggregated in later stages at the server, the points R1, R2, and R3 sum up to 0. This property ensures that the accuracy of the Federated Learning model is not adversely affected.

5. The next step is to calculate C1, C2 and C3 that are being referred to as “cryptograms”.

   C1= (r1 + w1[i])* G

   C2= (r2 + w1[i]) * G

   C3= -r3+w3[i]

   The value of i ranges from 0 to the number of weights. Therefore the number of cryptograms calculated are equal to the number of weights of each model.

6. The cryptograms are added by the server to perform aggregation as part of federated learning. The sums of the cryptograms become the reference points for the elliptic curve cyclic group. The number of reference points is the same as the number of weights.

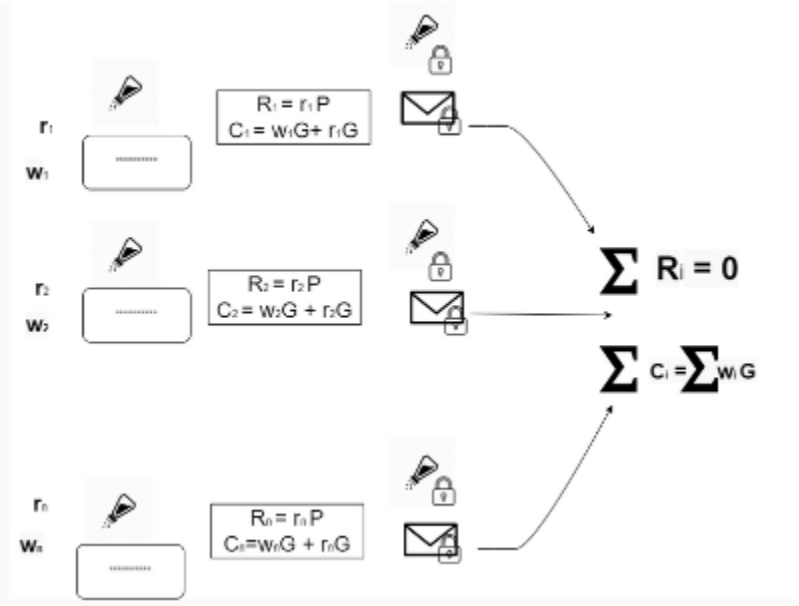
   Reference point[i]=C1[i]+C2[i]+C3[i]

   The value of i ranges from 0 to the number of weights.

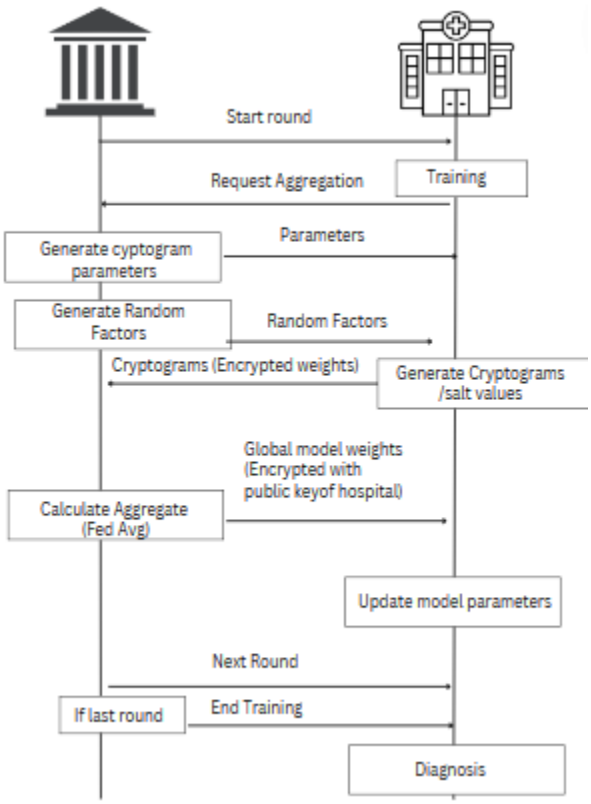
   A point to note here is that aggregation is performed on encrypted weights. This means that the original weights are not revealed to the server.

7. In this algorithm, the cyclic group of elliptic curves starts from the base point G. It continues to generate the cycle by finding 2G, 3G, 4G, and so on. The cycle terminates when the value of point additions equals a particular reference point. If k*G is equal to the reference point, then k represents the aggregated value of

the three weights (one from each model) corresponding to that reference point. This process is repeated for all the reference points, and aggregated weight values are determined for the corresponding weights of each reference point.



3.3 Architecture



- ***Local Dataset Training***

  The clients will train their datasets locally. ChestMNIST dataset is trained locally using the Convolutional Neural Network(CNN) model that consists of 14 diseases. Following the training the weights of models are generated. The clients have a webpage on their end to enable them uploading of dataset, training it locally, encrypting the weights and sharing the encrypted weights with the server.

- ***Encrypting Weights***

  The generated weights are encrypted using cryptograms. Keys are shared between the server and client. The encryption will be done using elliptic curve based implementation.

- ***Aggregation***

  Encrypted weights are received by the server that decrypts the weights and performs aggregation by calculating the Federated Average. The aggregated model is then re encrypted and shared with the clients

- ***Aggregated Model Sharing***

  Clients receive aggregated models from the server which is then used to make predictions. The doctors will have an interface to view the results when they make a query.

This process is done in multiple rounds where local models constantly send their weights to the server until convergence is reached.

3.4 Constraints

Following are the constraints

- Few number of clients (up to 3)
- The accuracy and correctness of dataset being used
- The accuracy of query made by client i-e accuracy of information provided by the doctor
- Only caters 14 diseases related to chest
- Hospitals must maintain separate dataset for chest diseases

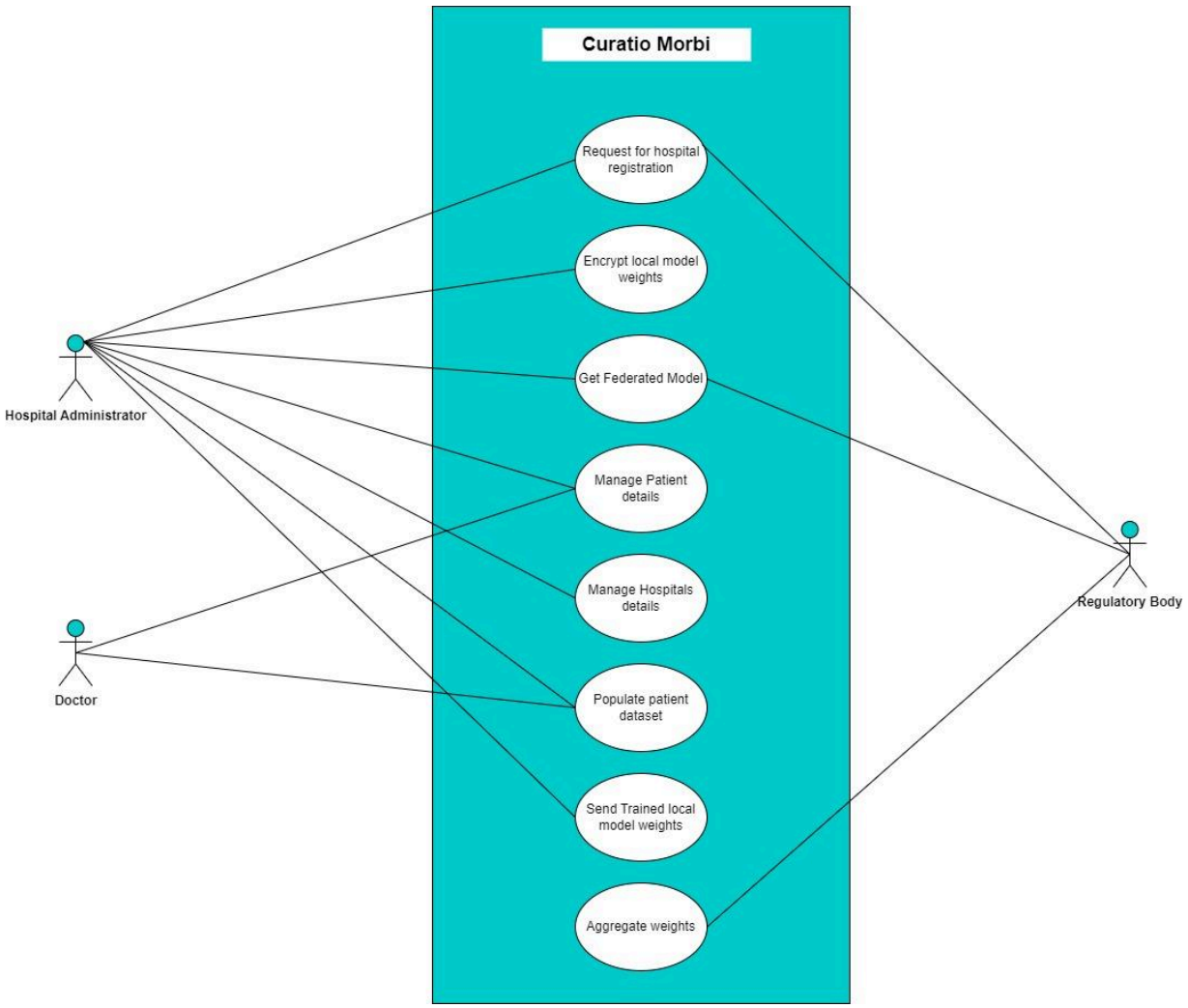
- Hospitals have to follow a protocol while storing the data (i-e features must be same of all the datasets)

3.5 Evaluation

- We will evaluate our developed system by the accuracy of the local models and the accuracy after aggregation is performed.
- We will match predictions made by our model with real or known cases in order to evaluate the correctness of results
- We will compare accuracy after implementation of encryption algorithms with the accuracy of the original model

Chapter 4- Use Cases

4.1 Use Case Diagram



4.2 Use Cases

Use Case ID	Primary Actor	Use Case
UC-1001	Hospital Administration	Request for hospital registration
UC-1002	Hospital Administration	Encrypt Local model weights
UC-1003	Hospital Administration	Get Federated Model
UC-1004	Hospital Administration, Doctor	Manage Patient details
UC-1005	Hospital Administration	Manage Hospital details
UC-1006	Hospital Administration, Doctor	Populate patient dataset
UC-1007	Hospital Administration	Send trained local model weights
UC-1008	Hospital Administration	Aggregate weights

4.3 High Level Use Cases

Use Case ID:	UC-1001
Use Case Name:	Request to register hospital in system
Actors:	Hospital Administration
Type:	Primary
Description:	Hospital Administrative will be able to request regulatory body to become a part of the system.

Use Case ID:	UC-1002
Use Case Name:	Encrypt local model weights
Actors:	Hospital Administration
Type:	Primary
Description:	Hospital Administration will be able to encrypt model weights before sending them on the server.

Use Case ID:	UC-1003
Use Case Name:	Get Federated model
Actors:	Hospital Administration
Type:	Primary
Description:	Hospital Administration will be able to request servers for federated models.

Use Case ID:	UC-1004
Use Case Name:	Manage Patient details
Actors:	Hospital Administration, Doctor
Type:	Primary
Description:	• Hospital Administration will be able to add and view patients in the system.

	• Doctors will be able to view patient details.

Use Case ID:	UC-1005
Use Case Name:	Manage Hospital details
Actors:	Hospital Administration, Doctor
Type:	Primary
Description:	• Hospital Administration will be able to view hospitals in the system. • Doctors will be able to view hospitals in the system.

Use Case ID:	UC-1006
Use Case Name:	Populate patient dataset

Actors:	Hospital Administration, Doctor
Type:	Primary
Description:	• Hospital Administration will be able add new patients' details in the dataset. • Doctors will be able to add new patients' details in the dataset.

Use Case ID:	UC-1007
Use Case Name:	Send trained local model weights
Actors:	Hospital Administration
Type:	Primary
Description:	Hospital Administration trains their model locally and send the weights to regulatory body.

Use Case ID:	UC-1008

Use Case Name:	Aggregate weights
Actors:	Hospital Administration
Type:	Primary
Description:	The weights sent by the hospital administration are aggregated by the regulatory body.

Chapter 5-Software Requirements Specification

5.1 List of Features/Functional Requirements

The major functions/features of the system will be:

- ***Managing the local dataset***

  The hospitals on the system would be able to manage their local datasets. This dataset will be later used to train the local AI model by the hospital.

- ***Encryption of the local model weights***

  The hospitals will encrypt the weights of their trained AI models before sending it to the server. This would protect the shared data (weights of the model) from adversaries.

- ***Local model training***

  The hospitals on the system will be able to train the machine learning model on their locally stored dataset that will be shared later to create the federated machine learning model by the server.

- ***Aggregation of the weights***

  The server of the regulatory authority will aggregate all of the encrypted weights received by the hospitals on the system. These weights will be aggregated to get the federated weights of the final model.

- ***Disease Prediction***

  The doctors at the hospitals will be able to use the system to access the federated model. On the hospitals end, the doctor could send the patients data on the system to get predictions of the diseases from the federated model received by the hospital.

5.2 Quality Attributes/Non-functional Requirements

- ***Performance***

  The various models that the system will use will have at least 80% accuracy.

- ***Usability***

  The web app will have a clean, minimalistic look that will be easy for users to understand and use.

- ***Reliability***

  The system will be reliable in terms of predicting – using various machine learning techniques, our application will give a good accuracy.

- ***Interoperability***

  The system would be made to run on a large range of web browsers, ensuring that it runs on the web browsers having either the latest or older web versions.

- ***Maintainability***

  The system will be developed in a way that its maintenance would be made easy. The system would be compromising independent modules. Each module could be maintained separately which would ensure that performing a change in a specific module would not affect the whole system.

- ***Modifiability***

  The system will be developed in independent modules which would ensure that each module could be modified separately without affecting the whole system.

- ***Testability***

  The system will be trained on a very large dataset with high data variability. This would ensure that the system would perform well in the testing stage.

- ***Reusability***

  Since each module will be independent, therefore with a little change, they can be used in other similar systems. The model trained on the system could be used in other scenarios of federated learning. We will also be able to incorporate different datasets and AI models in our system easily.

- ***Robustness***

  No data is being saved on the server. Only the weights of the model are shared with the server. The weights will be encrypted so attacks by an adversary will be unlikely. If it is assumed the datasets are secured and backed-up by the hospital, our system will be pretty much secured and there will be no chances of any data loss.

5.3 Hardware requirements

I. Hospital

The local model is to be trained at the hospital's end. The hospital management should ensure a high-performance computer to train the model efficiently.

Minimum Requirements:

- Processor: Intel Core i7 (10^{th} generation)
- RAM: 32 GB DDR4
- GPU: NVIDIA GeForce RTX 3070
- Operating System: Windows 10
- Python 3 environment with all the required libraries installed

II. Server

The aggregated model is to be trained at the server's end. Although aggregation of the local weights can be done using a high-performance computer but keeping scalability in mind, a good server will be required. The regulatory body must ensure a high performance server that ensures availability and performance to calculate the aggregated model.

Our codes for homomorphic encryption are written in Python 3.6, relying on Charm, a framework for rapidly prototyping advanced cryptosystems in Python. All tests are performed on a server with an Intel Xeon E5-2660 v3 2.60 GHz CPU and 64GB Memory. Since our work is mainly focused on the model aggregation, all the involved parties are simulated in one machine to omit the network communication impact.

Minimum Requirements:

- Processor: Intel Xeon E5-2660 v3 2.60 GHz
- RAM: 64 GB DD

Chapter 6-Implementation

1) Iteration 1

In [7]:

```
# define a simple CNN model

class Net(nn.Module):
    def __init__(self, in_channels, num_classes):
        super(Net, self).__init__()

        self.layer1 = nn.Sequential(
            nn.Conv2d(in_channels, 16, kernel_size=3),
            nn.BatchNorm2d(16),
            nn.ReLU())

        self.layer2 = nn.Sequential(
            nn.Conv2d(16, 16, kernel_size=3),
            nn.BatchNorm2d(16),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2))

        self.layer3 = nn.Sequential(
            nn.Conv2d(16, 64, kernel_size=3),
            nn.BatchNorm2d(64),
            nn.ReLU())

        self.layer4 = nn.Sequential(
            nn.Conv2d(64, 64, kernel_size=3),
            nn.BatchNorm2d(64),
            nn.ReLU())

        self.layer5 = nn.Sequential(
            nn.Conv2d(64, 64, kernel_size=3, padding=1),
            nn.BatchNorm2d(64),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2))

        self.fc = nn.Sequential(
            nn.Linear(64 * 4 * 4, 128),
            nn.ReLU(),
            nn.Linear(128, 128),
            nn.ReLU(),
            nn.Linear(128, num_classes))

    def forward(self, x):
        x = self.layer1(x)
        x = self.layer2(x)
        x = self.layer3(x)
        x = self.layer4(x)
        x = self.layer5(x)
        x = x.view(x.size(0), -1)
        x = self.fc(x)
        return x
model = Net(in_channels=n_channels, num_classes=n_classes)
# define loss function and optimizer
if task == "multi-label, binary-class":
    criterion = nn.BCEWithLogitsLoss()
else:
    criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(model.parameters(), lr=lr, momentum=0.9)
```

Our project focuses on a classification problem that uses a medical dataset from MNIST, consisting of medical chest scans. After conducting our thorough research, we found that Convolutional Neural Networks (CNN) have shown great performance in similar image-related problems. Moreover, we also studied many research papers as a part of our literature review and noticed that CNN models were being used for similar image related datasets. Therefore, we decided to use CNN as a basic reference point and went forward

with developing our solution, aiming to achieve comparable results for our specific problem.

In [9]:
```
def client_update(client_model, optimizer, train_loader, epoch=5):
    """
    This function updates/trains client model on client data
    """
    model.train()
    for e in range(epoch):
        for batch_idx, (data, target) in enumerate(train_loader):
            data, target = data.cuda(), target.cuda().float()
            optimizer.zero_grad()
            output = client_model(data).float()
            loss = F.cross_entropy(output, target)
            loss.backward()
            optimizer.step()
    return loss.item()
```

In [10]:
```
def server_aggregate(global_model, client_models):
    """
    This function has aggregation method 'mean'
    """
    ### This will take simple mean of the weights of models ###
    global_dict = global_model.state_dict()
    for k in global_dict.keys():
        global_dict[k] = torch.stack([client_models[i].state_dict()[k].float() for i in range(len(client_models))], 0).mean(
    global_model.load_state_dict(global_dict)
    for model in client_models:
        model.load_state_dict(global_model.state_dict())
```

Through extensive training and evaluation, we were able to achieve an impressive accuracy of around 94.7% on this dataset, which was quite comparable to the accuracy levels documented in the research papers for similar datasets from MNIST. Satisfied by these results, we then proceeded to develop our FL model using the same CNN model.

In [9]:
```
def client_update(client_model, optimizer, train_loader, epoch=5):
    """
    This function updates/trains client model on client data
    """
    model.train()
    for e in range(epoch):
        for batch_idx, (data, target) in enumerate(train_loader):
            data, target = data.cuda(), target.cuda().float()
            optimizer.zero_grad()
            output = client_model(data).float()
            loss = F.cross_entropy(output, target)
            loss.backward()
            optimizer.step()
    return loss.item()
```

In [10]:
```
def server_aggregate(global_model, client_models):
    """
    This function has aggregation method 'mean'
    """
    ### This will take simple mean of the weights of models ###
    global_dict = global_model.state_dict()
    for k in global_dict.keys():
        global_dict[k] = torch.stack([client_models[i].state_dict()[k].float() for i in range(len(client_models))], 0).mean(
    global_model.load_state_dict(global_dict)
    for model in client_models:
        model.load_state_dict(global_model.state_dict())
```

In [16]:

```
###### List containing info about Learning #########
losses_train = []
auc_test = []
acc_train = []
acc_test = []
# Runnining FL

for r in range(num_rounds):
    # select random clients
    client_idx = np.random.permutation(num_clients)[:num_selected]
    # client update
    loss = 0
    for i in tqdm(range(num_selected)):
        loss += client_update(client_models[i], opt[i], train_loader[client_idx[i]], epoch=epochs)

    losses_train.append(loss)
    # server aggregate
    server_aggregate(global_model, client_models)

    acc, auc = test(global_model, test_loader)
    auc_test.append(auc)
    acc_test.append(acc)
    print('%d-th round' % r)
    print('average train loss %0.3g | auc test %0.3g | test acc: %0.3f' % (loss / num_selected, auc, acc))

```

```
100%|██████████| 3/3 [00:47<00:00, 15.77s/it]
test  auc: 0.507  acc:0.943
0-th round
average train loss 1.52 | auc test 0.507 | test acc: 0.943
100%|██████████| 3/3 [00:47<00:00, 15.71s/it]
test  auc: 0.499  acc:0.937
1-th round
average train loss 1.45 | auc test 0.499 | test acc: 0.937
100%|██████████| 3/3 [00:46<00:00, 15.42s/it]
test  auc: 0.496  acc:0.933
2-th round
average train loss 1.02 | auc test 0.496 | test acc: 0.933
```

We established a network consisting of 3 clients and trained their local models by randomly dividing the dataset among them. Through 3 iterations of federated learning, we were able to achieve an accuracy of 93.3% for the final aggregated model, which was still pretty comparable to the performance of the isolated model that was trained on the complete dataset.

2) Iteration 2

In iteration 2 of our project, we focused on developing Cryptograms, an encryption scheme that we implemented to ensure privacy on our system. As discussed earlier, we aimed to minimize the computational overhead to ensure privacy without any loss in accuracy of our Federated model. By the end of this iteration, we had successfully completed a basic implementation of Cryptograms using Python.

In [8]:

```
# G(0,1)
gx = 0
gy = 1
G = Point(gx,gy,curve256)
#P(0,-1)
px = 0
py = -1
P = Point(px,py,curve256)

r1 = random.randint(1, 17)
r2 = random.randint(1, 17)
r3 = r1 + r2
w1 = 3
w2 = 1
w3 = 2
print("W:",w1,w2,w3)

# nP = Point(px,(p-py)%p,curve256)
#nG = Point(gx,(p-gy)%p,curve256)

nP = Point(px,(p-py)%p,curve256)
nG = Point(gx,(p-gy)%p,curve256)

#print("NP",np)
R1 = P
R2 = r2 * P
R3 = r3 * nP
#print("R:",R1,R2,R3)


Z1 = (r1 + w1) * G
Z2 = (r2 + w2) * G
#--------------
nr = -r3 + w3
#--------------
Z3 = -nr * nG

Rsum = R1 + R2 + R3
referencePoint = Z1 + Z2 + Z3
```

```
print("Reference point: \n",referencePoint)
basePoint = P

#Cycle of Ps
points = []
points.append(P)
check = False
temp = basePoint
while check == False:
    temp += basePoint
    if(temp == I):
        check = True
    points.append(temp)
print("Length of cycle:",len(points),'\n')
print("Points in the cycle:")
num = 0

iteration = 0
for i in points:
    print("Point ",iteration+1,":")
    print(i)
    if(referencePoint == i):
        num = points.index(i)
    iteration += 1
print("\nReference Point matches the point",num+1,"in the cycle")
```

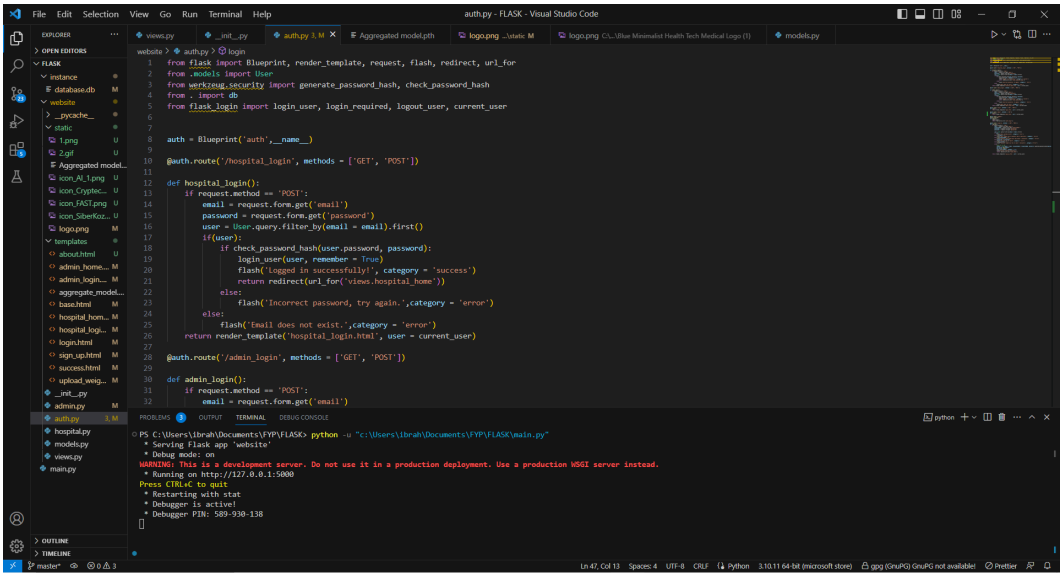
```
W: 3 1 2
Reference point:
 Point(x=0x0000000000000000000000000000000000000000000000000000000000000005, y=0x0000000000000000000000000000000000000000000000000
0000000000000000, curve=EllipticCurve(a=0x0000000000000000000000000000000000000000000000000000000000000002, b=0x0000000000000
0000000000000000000000000000000000000000000000001, field=PrimeGaloisField(prime=17)))
Length of cycle: 12

Points in the cycle:
Point  1 :
Point(x=0x0000000000000000000000000000000000000000000000000000000000000000, y=0x-0000000000000000000000000000000000000000000000000
0000000000000001, curve=EllipticCurve(a=0x0000000000000000000000000000000000000000000000000000000000000002, b=0x0000000000000
0000000000000000000000000000000000000000000000001, field=PrimeGaloisField(prime=17)))
Point  2 :
Point(x=0x0000000000000000000000000000000000000000000000000000000000000001, y=0x0000000000000000000000000000000000000000000000000
0000000000000002, curve=EllipticCurve(a=0x0000000000000000000000000000000000000000000000000000000000000002, b=0x0000000000000
0000000000000000000000000000000000000000000000001, field=PrimeGaloisField(prime=17)))
Point  3 :
Point(x=0x0000000000000000000000000000000000000000000000000000000000000008, y=0x0000000000000000000000000000000000000000000000000
000000000000000b, curve=EllipticCurve(a=0x0000000000000000000000000000000000000000000000000000000000000002, b=0x0000000000000
0000000000000000000000000000000000000000000000001, field=PrimeGaloisField(prime=17)))
Point  4 :
Point(x=0x0000000000000000000000000000000000000000000000000000000000000007, y=0x0000000000000000000000000000000000000000000000000
0000000000000010, curve=EllipticCurve(a=0x0000000000000000000000000000000000000000000000000000000000000002, b=0x0000000000000
0000000000000000000000000000000000000000000000001, field=PrimeGaloisField(prime=17)))
Point  5 :
Point(x=0x000000000000000000000000000000000000000000000000000000000000000a, y=0x0000000000000000000000000000000000000000000000000
0000000000000001, curve=EllipticCurve(a=0x0000000000000000000000000000000000000000000000000000000000000002, b=0x0000000000000
0000000000000000000000000000000000000000000000001, field=PrimeGaloisField(prime=17)))
Point  6 :
Point(x=0x0000000000000000000000000000000000000000000000000000000000000005, y=0x0000000000000000000000000000000000000000000000000
0000000000000000, curve=EllipticCurve(a=0x0000000000000000000000000000000000000000000000000000000000000002, b=0x0000000000000
0000000000000000000000000000000000000000000000001, field=PrimeGaloisField(prime=17)))
Point  7 :
Point(x=0x000000000000000000000000000000000000000000000000000000000000000a, y=0x0000000000000000000000000000000000000000000000000
0000000000000010, curve=EllipticCurve(a=0x0000000000000000000000000000000000000000000000000000000000000002, b=0x0000000000000
0000000000000000000000000000000000000000000000001, field=PrimeGaloisField(prime=17)))
Point  8 :
Point(x=0x0000000000000000000000000000000000000000000000000000000000000007, y=0x0000000000000000000000000000000000000000000000000
0000000000000001, curve=EllipticCurve(a=0x0000000000000000000000000000000000000000000000000000000000000002, b=0x0000000000000
0000000000000000000000000000000000000000000000001, field=PrimeGaloisField(prime=17)))
Point  9 :
Point(x=0x0000000000000000000000000000000000000000000000000000000000000008, y=0x0000000000000000000000000000000000000000000000000
0000000000000006, curve=EllipticCurve(a=0x0000000000000000000000000000000000000000000000000000000000000002, b=0x0000000000000
0000000000000000000000000000000000000000000000001, field=PrimeGaloisField(prime=17)))
Point  10 :
Point(x=0x0000000000000000000000000000000000000000000000000000000000000001, y=0x0000000000000000000000000000000000000000000000000
000000000000000f, curve=EllipticCurve(a=0x0000000000000000000000000000000000000000000000000000000000000002, b=0x0000000000000
0000000000000000000000000000000000000000000000001, field=PrimeGaloisField(prime=17)))
Point  11 :
Point(x=0x0000000000000000000000000000000000000000000000000000000000000000, y=0x0000000000000000000000000000000000000000000000000
0000000000000001, curve=EllipticCurve(a=0x0000000000000000000000000000000000000000000000000000000000000002, b=0x0000000000000
0000000000000000000000000000000000000000000000001, field=PrimeGaloisField(prime=17)))
Point  12 :
Point(x=None, y=None, curve=EllipticCurve(a=0x0000000000000000000000000000000000000000000000000000000000000002, b=0x00000000000
0000000000000000000000000000000000000000000000001, field=PrimeGaloisField(prime=17)))

Reference Point matches the point 6 in the cycle
```

In this particular implementation of Cryptograms, we demonstrated the practical application of elliptic curve cryptography to perform addition on three numbers. By leveraging this technique, we showed how the sum of three numbers can be accurately computed while ensuring the privacy and confidentiality of each individual number. This work served as a base for our future work for FYP 2, as our plan was to further expand this technique and integrate it with our clients in order to obtain the sum of weights contributed by each of the three clients. That value of sum would be then used to compute the respective average weight of the aggregated model.

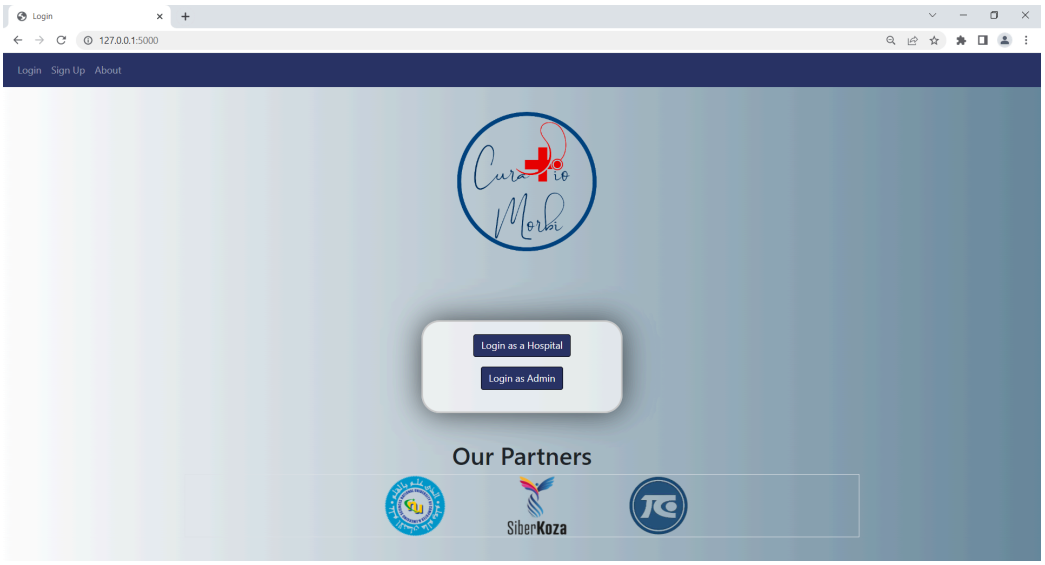
In this way, the weight contributed by a specific client cannot be traced back to that client using this cryptographic technique. This ensures that no privacy loss occurs during the aggregation process, maintaining the confidentiality of each client's contribution.

3) Iteration 3

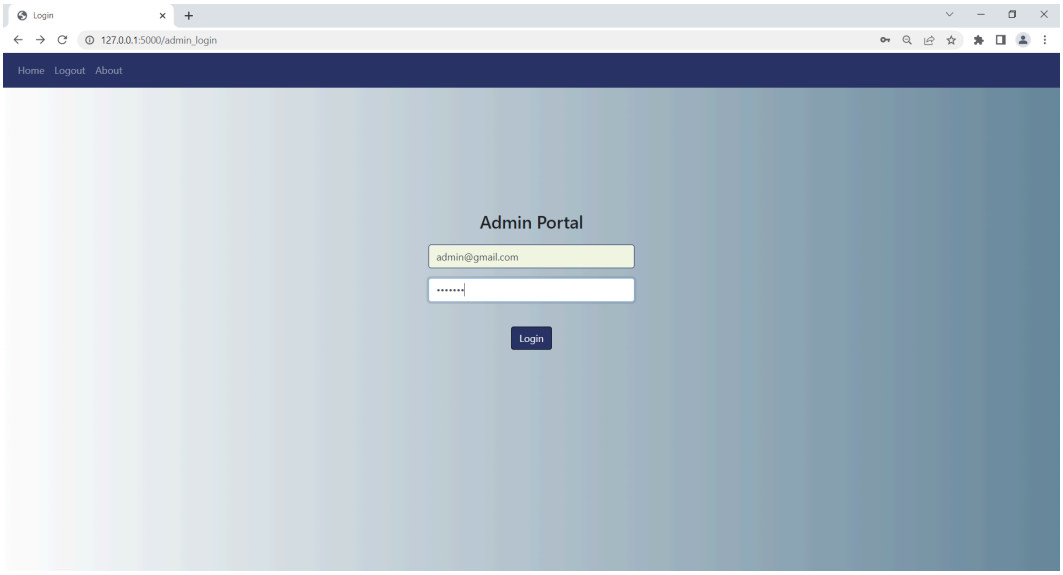
In the third iteration of our project, we focused on developing a FLASK website using Python, that would serve as the front end for our system. Our goal in iteration was to first integrate the Cryptogram encryption scheme with our Federated Learning model and then incorporate the entire system into a user-friendly website.

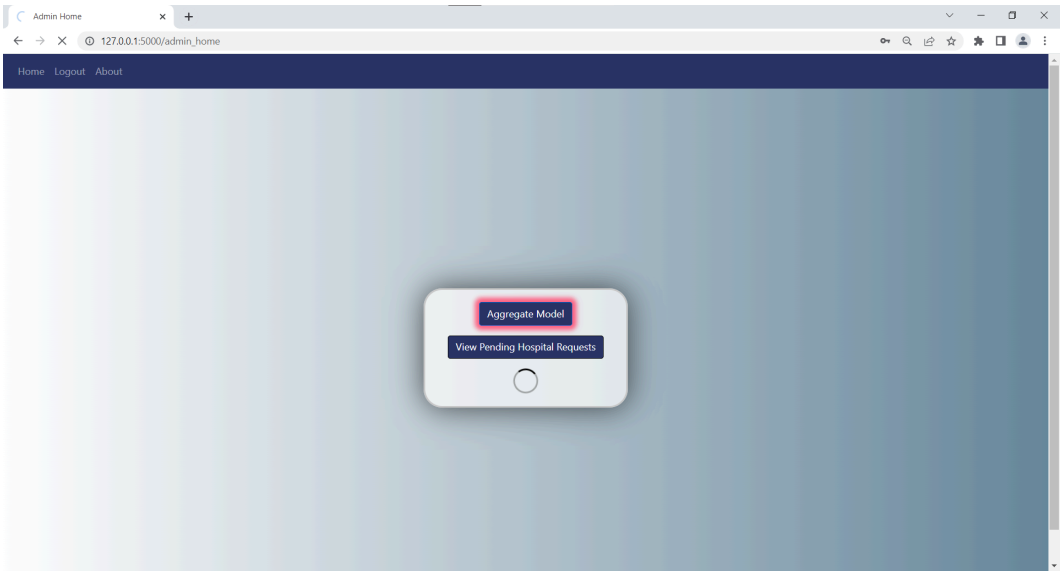


Through this website, the hospitals are able to register themselves and upload their locally trained models onto the website.

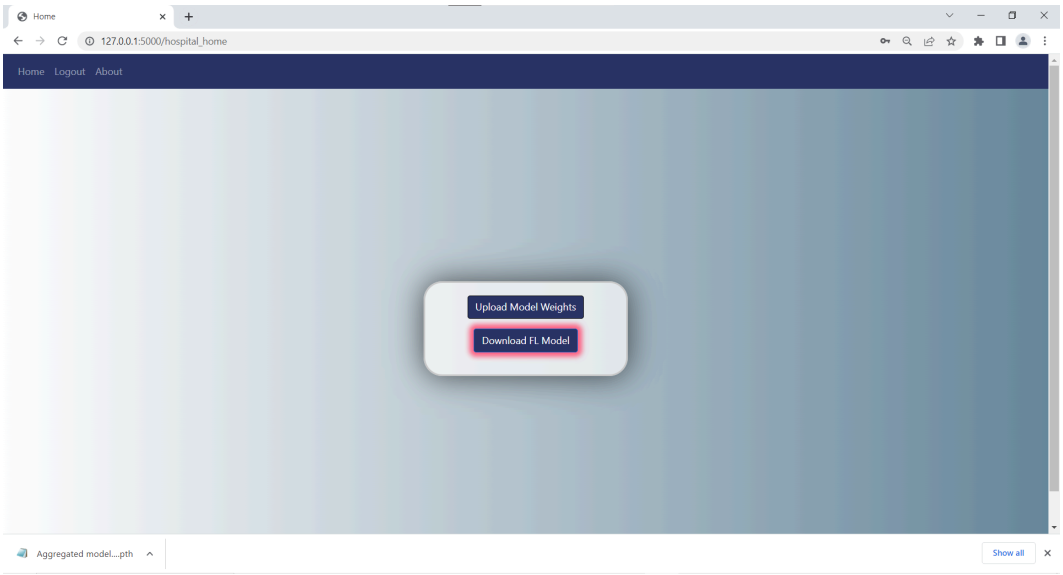


The server can retrieve the hospital models from the database during aggregation combining them to form the aggregated FL model using the Cryptograms cryptographic technique. This allows us to leverage the collective knowledge and learning of multiple hospitals on the system while preserving the privacy of their individual contributions.





Once the aggregation is complete, the hospitals can easily download the aggregated model from the website. This system empowers hospitals to stay updated with the latest aggregated model, enhancing their own local model and ultimately improving the overall accuracy and performance of the system.



4) Iteration 4

By the end of iteration 3, we had successfully completed the major milestones of our Final Year Project (FYP), and even presented our progress to the committee during the mid-evaluation. As we approached the final iteration, our focus shifted towards testing and enhancing the overall project so that its optimal performance and usability could be ensured. During this final stage of our project, we aimed to identify any potential issues and address them to improve the functionality and user-friendliness of the system.

Through these iterative cycles, we aimed to deliver a final product that not only fulfilled our initial project goals but also exceeded expectations in terms of usability, performance, and the overall user experience.

Chapter 7-Result and Discussions

The Federated Learning model has been developed using CNN model. It was run on the ChestMNIST dataset. For the implementation of FL the dataset containing 14 chest diseases was divided into 3 clients (nodes) in our case which are hospitals. The number of iterations performed were 3 as shown under the Implementation heading. The accuracy obtained was approximately 93%. CNN model was applied to ChestMNIST dataset with and without Federated Learning. The number of epochs were kept constant, that is 3 in both the cases. The results obtained in both the cases are similar and therefore it shows that the use of Federated Learning does not affect or drop the accuracy of the model.

Furthermore, from the research conducted on privacy issues of a Federated Learning setting, we concluded that the existing privacy preserving techniques like homomorphic encryption, secure multi party computation and differential privacy do not ensure no loss of accuracy. We have compared our model accuracy with the prior technique Differential Privacy and the results are shared below;

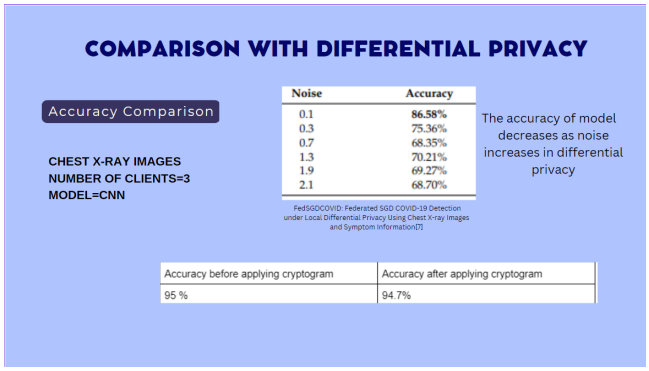


Fig: Shows comparison with Differential Privacy

We have also identified cryptograms as an efficient encryption methodology that solves the problems in the existing methodologies against the intrusive attacks on a FL model.

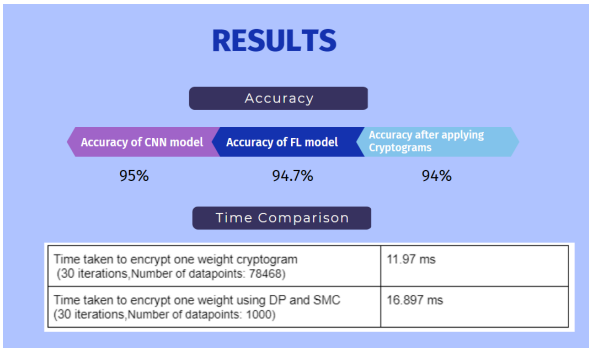


Fig: Shows the accuracy results with Cryptogram Algorithm

The algorithm of cryptograms works on the concepts of Diffie Hellman and Elliptic curve. The random factors generated for the encryption process are eventually canceled out so the accuracy is not compromised and the privacy is also achieved.

Chapter 8-Conclusion and Future Work

Federated Learning (FL) has emerged as a hot topic of discussion in the rapidly evolving tech industry as it combines the power of deep learning with edge computation. It offers a trusted solution that leverages the potential of Artificial Intelligence (AI) in scenarios where data governance and data privacy are the major concerning elements. Our research and development efforts have resulted in the creation of a system that demonstrates how FL can revolutionize inter-organizational collaboration, enabling the sharing of high-quality models without any fear or compromise of privacy.

The applications that can use FL can span across various domains, including but not limited to healthcare, finance, among many others. Our system addresses the problem of data privacy and governance by ensuring that data remains private to local users, with only the models being transmitted to the central server. This mitigates the risk of privacy attacks such as data poisoning, adversarial and inference attacks.

While certain solutions already exist that have attempted to tackle these challenges, they have their limitations and trade-offs. Some of them introduce potential accuracy loss, while the others contribute to greater computational costs or weaker privacy guarantees. Through extensive research and literature review, we had analyzed the shortcomings of the existing techniques and have proposed an innovative solution that uses Cryptograms and caters to all of these factors, ensuring privacy and without any accuracy loss or computational overhead.

Looking towards the future aspect of this project, our work can be laid as a foundation for further advancements and enhancements in Federated Learning. We can expand the capabilities of our existing system, exploring new avenues for inter-organizational collaboration. By continuous iterations and feedback, we can unlock the potential of Federated Learning and collaborative AI that may contribute to a new era of privacy-preserving AI applications.

References

[1] Byrd, D., & Polychroniadou, A. (2020, October). Differentially private secure multi-party computation for federated learning in financial applications. In *Proceedings of the First ACM International Conference on AI in Finance* (pp. 1-9).

[2]Rieke, Nicola, Jonny Hancox, Wenqi Li, Fausto Milletari, Holger R. Roth, Shadi Albarqouni, Spyridon Bakas et al. "The future of digital health with federated learning." *NPJ digital medicine* 3, no. 1 (2020): 1-7.

[3] Chengliang Zhang, Li, S., Xia, J., Wei Wang, W. W., Yan, F., & Liu, Y. (2020). BatchCrypt. BatchCrypt: Efficient Homomorphic Encryption for Cross-Silo Federated Learning

[4]Kanagavelu, R., Wei, Q., Li, Z., Zhang, H., Samsudin, J., Zhang, Y., Sio, R., & Wang, S.(n.d.). CE-Fed: Communication efficient multi-party computation enabled federated learning.

[5] Dickson, B. (2021, April 14). *Inference attacks: How much information can machine learning models leak?* PortSwigger. Retrieved October 23, 2022

[6]Constantin, L. (2021, April 12). *What is data poisoning? Attacks that corrupt machine learning models*. CSO Online. Retrieved October 23, 2022

[7]Wei, Kang, Jun Li, Ming Ding, Chuan Ma, Howard H. Yang, Farhad Farokhi, Shi Jin, Tony QS Quek, and H. Vincent Poor. "Federated learning with differential privacy: Algorithms and performance analysis." *IEEE Transactions on Information Forensics and Security* 15 (2020): 3454-3469.

[8] Shahandashti, S. F., & Hao, F. (2016). DRE-ip: A Verifiable E-Voting Scheme Without Tallying Authorities. *Computer Security – ESORICS 2016*.