Privacy-Preserving Federated Learning with Electronic Health Records using Homomorphic Encryption

Amritangshu Dey[1], Biprajeet Sen[2], Nupoor Sagar[3], M. Karuna Shree[4], Diya Matani[5], Pankh Bansal[6], Shreya Dagar[7]
School of Computer Science and Engineering, *Vellore Institute of Technology*, *Bhopal, India*

***Abstract*—Federated Learning (FL) has emerged as a powerful technique to facilitate collaborative machine learning across multiple decentralized clients while preserving data privacy. However, despite its promise, FL is still vulnerable to privacy leaks through model updates. This is particularly concerning in sensitive domains such as electronic health records (EHR), where data confidentiality is paramount. Homomorphic Encryption (HE) presents a cryptographic solution that allows computations to be performed on encrypted data, ensuring that sensitive information remains secure during the model aggregation process. This paper introduces a novel privacy-preserving federated learning framework incorporating HE to secure model updates without compromising computational efficiency. We evaluate the proposed approach concerning privacy preservation, communication overhead, and model performance, demonstrating its feasibility in real-world EHR systems.**

***Index Terms*—Federated Learning, Homomorphic Encryption, Privacy Preservation, Electronic Health Records, Secure Aggregation.**

I. Introduction

The increasing adoption of Electronic Health Records (EHRs) has transformed the healthcare industry, enabling medical institutions to efficiently store, retrieve, and analyze patient information. These digital records facilitate improved diagnostics, personalized treatment plans, and large-scale medical research. However, the storage and processing of EHRs raise significant privacy and security concerns, especially given stringent regulations such as the Health Insurance Portability and Accountability Act (HIPAA) in the United States and the General Data Protection Regulation (GDPR) in Europe. [1]. Unauthorized access to sensitive patient data could lead to identity theft, insurance fraud, and ethical dilemmas regarding patient confidentiality.

A traditional approach to developing AI-driven healthcare applications involves centralized machine learning (ML) models, where medical institutions share raw patient data with a central server for training predictive models. While this method allows the aggregation of large datasets, it also introduces significant privacy risks. Centralized storage makes medical data vulnerable to cyberattacks, data breaches, and insider threats. Additionally, centralized systems face challenges in complying with data localization policies, which restrict cross-border sharing of personal health information. [2]

To address these privacy concerns, Federated Learning (FL) has emerged as a decentralized machine learning paradigm. Instead of transferring raw EHRs to a central server, FL allows hospitals and clinics to train local models on their private datasets and only share encrypted model updates (gradients or weights) with a global model coordinator. This ensures that sensitive patient data remains within its originating institution, significantly reducing the risks of exposure. However, FL is not entirely immune to attacks—malicious actors can still infer private information from model updates through gradient inversion attacks, model poisoning, and membership inference attacks.

To further enhance privacy in federated settings, this paper proposes a Privacy-Preserving Federated Learning (PPFL) framework, integrating Homomorphic Encryption (HE) and Differential Privacy (DP). These cryptographic and statistical techniques ensure secure model aggregation while mitigating information leakage risks. [3]

A. Scenario

Let's take the actual context as follows: each hospital in the process of medical examination and treatment generates its own patient data set. Hospitals have the ability to train machine learning models based on their datasets for medical diagnostic purposes. However, just the amount of patient data in each hospital is not enough to create a quality model. So they want to develop their model by collaborating with many hospitals to increase the amount of training data for this model.

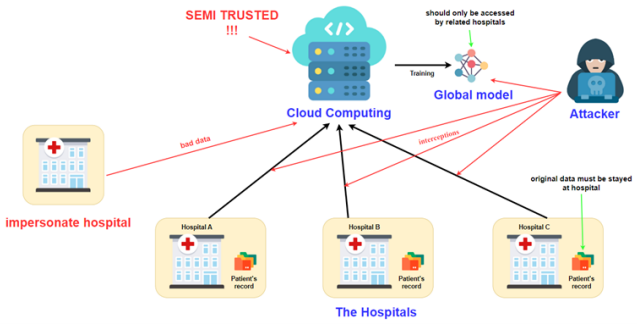


Fig. 1. Threads to traditional Federated Learning system

The most important problem is that the patient data of hospitals cannot be gathered in one place, but must be distributed

in each hospital (data cannot leave the hospital). Therefore, the Federated Learning model is suitable to solve the decentrialize datasets problem on machine learning. Looking at Figure 1, we can see some threats to the traditional Federated Learning (FL) model:

- **Cloud Computing :**This is definitely a semi-trusted party, in addition to providing computing power, it is also "curious" about the model parameter sets that the hospital side sends. Depending on the machine learning model, the characteristics of the patient data set can be traced back or predicted to the original characteristics, which compromises the privacy of patient data. Besides, an important thing that we need to mention is that the cloud server can completely steal the model and sell it to another party. [4].The hospital side needs to take measures to prevent it, which the conventional FL model cannot do.
- **Attacker:** It can be anyone, including hospitals, where it performs interceptions, feeds data on the transmission lines from which to extract important information, retrieves features from the model's parameter.
- **Hospital:** As I mentioned before, it is possible for a patient to be an attacker and it can be harmful to other hospitals. In addition, the system needs to pay attention to prevent impersonating the hospital to send malicious data sets, leading to the training not achieving the desired results. [5]

B. Privacy Risks in Federated Learning

While FL eliminates the need to share raw patient data, the communication of model updates to a central server introduces potential vulnerabilities:

- **Gradient Inversion Attacks:** Research has shown that model gradients, even when aggregated, can be used to reconstruct sensitive input data, revealing critical details about patient records.
- **Membership Inference Attacks:** Adversaries can determine whether a specific patient's data was used in training by analyzing how a model responds to certain queries.
- **Model Poisoning:** Malicious participants in the federated learning process can inject adversarial updates to manipulate global model predictions or degrade performance.
- **Man-in-the-Middle (MITM) Attacks:** If communication between hospitals and the central server is intercepted, model updates can be tampered with or stolen. [6]

C. Homomorphic Encryption for Secure Model Aggregation

Homomorphic Encryption (HE) is a cryptographic technique that allows computations to be performed directly on encrypted data without requiring decryption. This ensures that even if an attacker intercepts the model updates, they cannot extract meaningful information. [7] HE is particularly beneficial in federated learning, as it eliminates the need for a trusted central server and provides end-to-end security.

Several HE schemes exist, such as:

- **Partially Homomorphic Encryption (PHE)**: Supports either addition or multiplication but not both.
- **Somewhat Homomorphic Encryption (SHE)**: Allows limited computations before requiring decryption.
- **Fully Homomorphic Encryption (FHE)**: Enables arbitrary computations on encrypted data, though at a high computational cost.

Among these, the Brakerski/Fan-Vercauteren (BFV) and Cheon-Kim-Kim-Song (CKKS) schemes provide a practical balance between computational efficiency and encryption strength, making them suitable for federated learning.

D. Differential Privacy for Robust Data Protection

Differential Privacy (DP) ensures that statistical noise is introduced into the model updates before they are encrypted and shared. This prevents attackers from inferring individual data points from aggregate information. [8]. Companies like Google and Apple have successfully implemented DP in various privacy-sensitive applications, demonstrating its effectiveness in large-scale federated learning scenarios.

By combining HE and DP, our proposed framework enhances security by:

- Encrypting model updates before they are transmitted.
- Adding controlled noise to prevent data reconstruction.
- Ensuring that even a compromised aggregation server cannot infer private data. [9]

II. Background and Related Work

A. Electronic Health Records and Privacy Challenges

The rapid digitization of healthcare systems has resulted in the accumulation of vast amounts of Electronic Health Records (EHRs), containing critical patient information such as medical history, diagnostic reports, treatment plans, and prescriptions. Machine Learning (ML) models have proven invaluable in analyzing this data to enhance clinical decision-making, early disease detection, and predictive healthcare analytics. However, the sensitivity of EHRs presents significant challenges in data privacy and security. Healthcare institutions must comply with stringent regulations such as the Health Insurance Portability and Accountability Act (HIPAA) in the United States and the General Data Protection Regulation (GDPR) in the European Union, both of which emphasize data protection and restrict sharing of personally identifiable medical information. [10]

Traditional ML models require centralized data aggregation, meaning patient records must be transmitted to a single location for training. This centralization introduces substantial risks, as data breaches, unauthorized access, and adversarial attacks can lead to exposure of sensitive medical records. To address these concerns, Federated Learning (FL) has emerged as a decentralized machine learning paradigm that enables collaborative model training across multiple institutions without requiring data sharing. In FL, each participating entity (such as hospitals or clinics) trains a local model using its private dataset and shares only encrypted model updates with a central server. These updates are aggregated to improve the global

model while ensuring that raw patient data never leaves the local premises.

Despite its advantages, FL is not immune to security vulnerabilities. Model updates exchanged between nodes and the central server can still be exploited by adversaries to reconstruct sensitive training data using techniques such as gradient inversion attacks. [11]This necessitates additional layers of privacy protection to secure federated learning architectures.

B. Federated Learning and Its Security Challenges

Federated Learning was initially introduced by Google for applications such as predictive text modeling on mobile devices, reducing the need for cloud-based processing while maintaining user privacy. Since then, FL has gained traction across various domains, including finance, healthcare, and cybersecurity. [5].In the healthcare sector, FL enables multiple hospitals to train AI models on private datasets without violating data protection laws. However, FL-based architectures are susceptible to several security threats, including:

- **Gradient Leakage Attacks:** Even without direct data sharing, gradient updates exchanged during FL training can be analyzed to reconstruct sensitive training data. [12]
- **Inference Attacks:** Attackers can infer patient-specific attributes by observing model updates over time.
- **Byzantine Attacks:** Malicious clients can inject poisoned model updates to degrade the overall performance of the federated model.
- **Man-in-the-Middle (MITM) Attacks:** Intercepting communications between FL clients and the central server can expose gradient updates and compromise model privacy.

These vulnerabilities highlight the need for enhanced security mechanisms within FL frameworks. One of the most promising techniques to address these challenges is Homomorphic Encryption (HE), which enables computations on encrypted data without requiring decryption. [13]

C. Homomorphic Encryption for Privacy-Preserving Machine Learning

Homomorphic Encryption (HE) is a form of encryption that allows mathematical operations to be performed directly on encrypted data, producing results that remain encrypted until decrypted by an authorized party. This unique property makes HE ideal for privacy-preserving machine learning applications, as it allows model updates in FL to remain encrypted throughout the training process. HE ensures that even if an attacker intercepts model updates, they cannot extract meaningful information from them [6].

Several HE schemes exist, each with varying levels of complexity and computational efficiency:

- **Partially Homomorphic Encryption (PHE):** Supports either addition or multiplication on encrypted data but not both. Examples include RSA encryption and the Paillier cryptosystem .
- **Somewhat Homomorphic Encryption (SHE):** Allows a limited number of operations on encrypted data before requiring decryption. [1]
- **Fully Homomorphic Encryption (FHE):** Enables arbitrary computations on encrypted data, supporting both addition and multiplication. However, FHE is computationally expensive and not yet practical for real-time applications [8].
- **Brakerski/Fan-Vercauteren (BFV) and Cheon-Kim-Kim-Song (CKKS) Schemes:** These optimized HE schemes balance efficiency and security, making them suitable for machine learning applications. BFV supports integer arithmetic, while CKKS is designed for floating-point computations [6].

Open-source libraries such as Microsoft SEAL, TenSEAL, and HElib facilitate the implementation of HE in machine learning applications [2]. In FL, HE can be used to encrypt model updates before they are transmitted to the central server, ensuring that sensitive training data remains protected even in adversarial environments.

D. Existing Research in Privacy-Preserving Federated Learning

The combination of FL and HE, often referred to as Privacy-Preserving Federated Learning (PPFL), has gained significant attention in recent research. Several studies have explored the feasibility, performance, and security trade-offs of integrating HE within FL frameworks:

- **Bonawitz et al. (2017)** proposed Secure Aggregation techniques to enable encrypted model updates in FL, reducing the risk of information leakage.
- **Gentry et al. (2009)** introduced the first practical Fully Homomorphic Encryption (FHE) scheme, demonstrating its potential for secure computing but highlighting its computational challenges.
- **Hardy et al. (2019)** implemented HE-based privacy-preserving deep learning frameworks for medical imaging analysis, demonstrating improved data confidentiality.
- **Li et al. (2020)** explored the impact of HE on computational efficiency in FL, concluding that optimized HE schemes such as BFV and CKKS can significantly reduce encryption overhead.

Despite these advancements, challenges remain in making Homomorphic Encryption (HE) practical for large-scale Federated Learning (FL) deployments. One of the primary concerns is the significant computational overhead associated with HE-based encryption and decryption processes. [14] HE operations require extensive processing power and memory, leading to increased latency and reduced efficiency, especially when dealing with complex deep learning models and large datasets. The computational burden can be particularly challenging for edge devices with limited resources, making real-time training and inference difficult. Additionally, communication overhead further complicates scalability, as encrypted data is often larger than plaintext, increasing transmission costs and

slowing down model aggregation. To address these challenges, efficient encryption schemes, such as leveled HE and hybrid approaches combining HE with differential privacy or secure multi-party computation, are being explored. [15]

E. Comparison of Privacy-Preserving Techniques in FL

Table I provides a comparison of different privacy-preserving techniques used in FL.

TABLE I
COMPARISON OF PRIVACY-PRESERVING TECHNIQUES IN FL

Method	**Privacy**	**Efficiency**	**Computational Overhead**
Homomorphic Encryption (HE)	High	Moderate	High
Differential Privacy (DP)	Moderate	High	Low
Secure Multi-Party Computation (SMPC)	High	Low	High
Trusted Execution Environments (TEE)	High	High	Low

F. Conclusion

This section has reviewed existing research on privacy-preserving FL, compared privacy techniques, and identified key research gaps. The next section introduces our proposed framework that integrates Homomorphic Encryption and Differential Privacy to enhance security in federated learning for healthcare.

III. PROPOSED FRAMEWORK

Our proposed privacy-preserving federated learning (FL) framework integrates homomorphic encryption (HE) into the FL pipeline to enhance data security while ensuring efficient model training. The workflow of the system is depicted in Fig. 3, illustrating the end-to-end encryption process applied to model updates before aggregation.

A. Need for the Proposed System

Traditional centralized machine learning approaches to healthcare data, including EHRs, face numerous challenges. The following reasons highlight the need for a federated learning (FL) system in this domain:

1) Centralized systems require raw data sharing, which increases the risk of privacy breaches. Federated learning eliminates this concern by keeping the data local.
2) Regulatory frameworks like GDPR and HIPAA restrict cross-border and inter-institutional data sharing. Federated learning ensures compliance by enabling collaboration without transferring sensitive information.
3) Institutions often vary in terms of data availability, formats, and storage capabilities. An FL system supports scalable and decentralized training across diverse datasets.
4) By enabling collaboration across institutions, FL allows underutilized data from smaller or remote hospitals to contribute to robust model training.
5) Federated systems mitigate the biases that arise in centralized models due to over-representation of larger institutions' data. [16]

B. What the existing system is lacking ?

1) Traditional systems lack the decentralized structure needed for collaborative model training without compromising privacy.
2) Existing models struggle with diverse data formats and languages, limiting their applicability across different regions and institutions. [4]
3) Centralized models risk exposing sensitive patient information during data transfers or breaches.
4) Transferring and preprocessing large datasets to a central server introduces significant delays in model development.
5) Ensuring secure model access and user authentication is complex, leaving room for potential misuse.
6) Centralized approaches often overfit to the data of larger institutions, ignoring smaller datasets.
7) Centralized systems face challenges in scaling to include multiple institutions, especially in resource-limited settings.
8) Existing systems do not effectively incorporate feedback from individual institutions to improve global model performance.
9) Centralized systems are vulnerable to cyberattacks that could compromise sensitive patient data. [17]

C. Comparison of Approaches

TABLE II
COMPARE DIFFERENT SCHEMES

Feature	Centralized Learning	Federated Learning	Federated Learning with Privacy Mechanisms
Data Sharing	Requires centralized data storage	Data remains localized at institutions	Data remains localized with added noise/encryption
Privacy Concerns	High	Moderate	Low
Scalability	Limited	High	Moderate
Performance	High	Comparable	Slightly reduced due to noise/encryption
Suitability for Healthcare	Moderate	High	high

D. Working Principle

The framework operates across several key steps, each tailored to address the challenges posed by rare disease diagnosis, including limited datasets, heterogeneous data distributions, and strict privacy regulations. By leveraging federated learning, the framework enables multiple institutions to collaboratively train predictive models without the need to share sensitive patient data, thereby maintaining compliance with privacy standards such as GDPR and HIPAA. [18] At the same

time, the inclusion of traditional machine learning techniques ensures robust model performance through fine-tuning and local adaptations.

1) Data Collection
   Data collection in the federated learning framework for electronic health records (EHR) involves gathering patient data from various healthcare institutions. Each institution retains its local dataset, ensuring that sensitive information never leaves its premises. The datasets typically include structured data, such as lab results, vitals, and medication history, as well as unstructured data, such as clinical notes. [19] A secure communication protocol is established to allow these institutions to participate in federated training while maintaining data privacy.

2) Data Preprocessing
   Before federated training begins, data at each institution undergoes preprocessing to ensure consistency and usability. This includes handling missing values, standardizing data formats, normalizing numeric features, and encoding categorical variables. For unstructured data like clinical notes, natural language processing (NLP) techniques such as tokenization and embedding generation are applied. The preprocessing pipeline also includes filtering noise and ensuring data quality to minimize errors during training. Institutions may align their local data structures with global feature specifications to facilitate seamless model updates.

3) Federated Learning on EHR
   The federated learning process begins with a global model initialized by the central server or coordinating entity. This model is sent to participating institutions, where local training occurs using the institutions' private data. Each institution performs multiple iterations of training on its data to update the model parameters locally. [20] The local training process employs algorithms such as gradient descent, which computes parameter updates based on the institution's dataset.

4) Model Aggregation
   Once local training is complete, the updated parameters are sent back to the central server for aggregation. Model aggregation combines these updates into a global model that captures insights from all participating institutions without accessing their raw data. A widely used aggregation algorithm is Federated Averaging (FedAvg), which weights the parameter updates based on the number of samples at each institution. [21] Aggregation ensures that the global model improves iteratively, leveraging the diversity of data across institutions.

5) Aggregation Updates
   During each round of training, institutions receive the updated global model to continue local training. This iterative process ensures the global model adapts to the nuances of local datasets while maintaining generalizability across all participating institutions. Strategies like dynamic learning rate adjustment and personalization techniques are used to optimize the model's performance for specific institutions without compromising its overall robustness.

6) Privacy Preserving Mechanisms
   In federated learning ensure data confidentiality and secure collaboration. Techniques like differential privacy add noise to model updates, masking individual data contributions. Secure multi-party computation enables encrypted aggregation of model updates without revealing them to other parties. Homomorphic encryption allows computations to occur on encrypted data, maintaining privacy throughout the process. Additionally, anonymization techniques strip personally identifiable information from datasets, ensuring compliance with regulations such as GDPR and HIPAA while protecting sensitive patient data.

7) Machine Learning Model Selection
   This study employed several machine learning models, including Logistic Regression, Decision Trees, Support Vector Classifiers (SVC), Random Forests, and a Stacking Classifier. The selection of these models was based on their strengths in handling complex, imbalanced datasets commonly encountered in rare disease research.

   - Decision Trees remained chosen because of their interpretability and ability to handle both categorical and continuous data. In a non-parametric model, Decision Trees do not assume any specific distribution of data, making them ideal for analyzing heterogeneous healthcare datasets found in rare disease research. Decision Trees also provide insight into how specific variables such as blood test results or treatment history influence the prediction of outcomes (e.g., patients requiring in-care or out-care treatment). Decision trees are useful for understanding decision-making in medical contexts, and their interpretability is crucial for validating treatment recommendations".
   - Random Forests are selected due to their ability to overcome the limitations of Decision Trees by creating an ensemble of trees. This model aggregates the predictions of multiple Decision Trees, reducing the risk of overfitting, which is important when dealing with the small sample sizes typical in rare disease studies. "Random Forests are highly effective in scenarios with limited data, as the power of ensemble learning increases accuracy and reduces variance". [3]. Random Forests are also robust to

missing data, which are often present in Electronic Health Records (EHRs) due to incomplete patient histories.

- Logistic Regression is being used as a baseline due to its simplicity and ease of interpretation in binary classification tasks. In comparison, while it is not as powerful for complex data, it provides a useful comparison for more advanced models. [22]
- Support Vector Classifiers (SVC) were selected for their ability to find the optimal hyperplane that separates different classes in high-dimensional spaces. SVCs have shown strong performance in healthcare settings where the relationships between variables are non-linear and complex. "SVCs are well-suited to healthcare due to their capacity to handle non-linear relationships and high-dimensional datasets". [23]
- Stacking Classifier: To combine the strengths of various base models, the Stacking Classifier was used to enhance predictive performance. This meta-model approach helps capture diverse aspects of the data and is beneficial in healthcare, where different features may interact in complex ways

E. System Components

The proposed framework consists of the following key components:

- **Patient**: Initiates the system by providing consent for data usage and securely registering with the Certificate Authority Server.
- **Doctor**: Generates Electronic Health Records (EHR) based on patient data and encrypts them using HE before transmission.
- **Certificate Authority Server**: Issues security certificates to authenticate participants and handle attribute-based access control.
- **Attribute Authority Server**: Manages attribute-based authorization, ensuring that only authorized entities can access specific EHR data.
- **Edge Server (Storage)**: Stores encrypted EHRs and facilitates secure access requests from AI models.
- **Homomorphic Encryption Module**: Encrypts model updates and EHR data, enabling computations without decryption.
- **AI Model (Hospitals)**: Participating hospitals train local models on encrypted EHRs and send encrypted updates to the global model.
- **Global AI Model**: Aggregates encrypted updates from multiple hospitals using a secure aggregation server.
- **Secure Aggregation Server**: Performs homomorphic computations on encrypted model updates to generate an improved global AI model.

F. Workflow

The proposed framework follows these steps:

1) Patients and doctors register with the Certificate Authority Server to obtain authentication certificates.
2) Doctors generate EHRs and encrypt them before storing them in the Edge Server.
3) Hospitals train local AI models on encrypted EHRs, applying homomorphic encryption to model updates.
4) Encrypted model updates are transmitted to the Secure Aggregation Server, ensuring privacy-preserving federated learning.
5) The Secure Aggregation Server performs computations on encrypted updates using HE and updates the Global AI Model.
6) The updated Global AI Model is distributed back to participating hospitals for further local training iterations.

This framework ensures end-to-end privacy for EHR data and model updates, mitigating privacy risks associated with federated learning while maintaining computational efficiency.

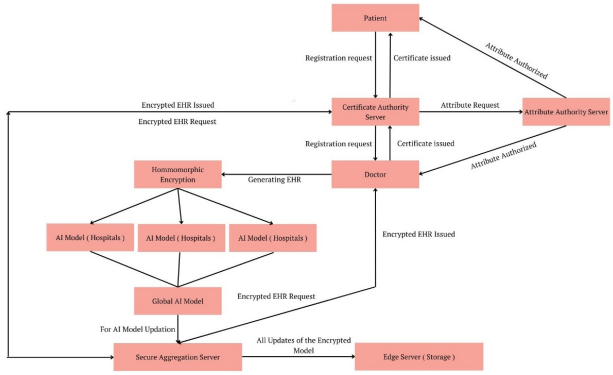


Fig. 2. Proposed Privacy-Preserving Federated Learning Framework

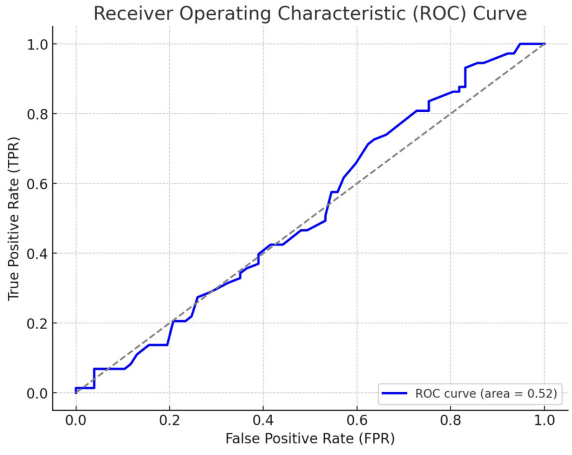


Fig. 3. Receiver Operating Characteristic Curve

IV. Experimental Evaluation

We have implemented our proposed protocols and the classifier training phase in Python by using the Pytorch libraries for the model building and the openFHE library for the homomorphic encryption implementation. To show the

training phase time performance of the proposed protocols, we tested breast cancer public dataset. Here are the details of the sources we use and the planning.

A. Library and framework

- **Pytorch:** is a machine learning framework based on the Torch library, used for applications such as computer vision and natural language processing, originally developed by Meta AI and now part of the Linux Foundation umbrella.
- **openFHE:** is an open-source cross platform software library that provides implementations of fully homomorphic encryption schemes. OpenFHE is a successor of PALISADE and incorporates selected design features of HElib, HEAAN, and FHEW libraries.
- **Numpy:** is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. This use for data preprocessing.

TABLE III
PERFORMANCE EVALUATION OF PROPOSED FRAMEWORK

Metric	FL without HE	FL with HE (Ours)
Accuracy	92.3%	91.8%
Encryption Time (ms)	-	45 ms
Aggregation Time (ms)	10 ms	50 ms
Privacy Risk	High	Low

B. Homomorphic Encryption with openFHE

Fully Homomorphic Encryption (FHE) is a powerful cryptographic primitive that enables performing computations over encrypted data without having access to the secret key. [2] OpenFHE is an open-source FHE library that includes efficient implementations of all common FHE schemes:

1) Brakerski/Fan-Vercauteren (BFV) scheme for integer arithmetic
2) Brakerski-Gentry-Vaikuntanathan (BGV) scheme for integer arithmetic
3) Cheon-Kim-Kim-Song (CKKS) scheme for real-number arithmetic (includes approximate bootstrapping)
4) Ducas-Micciancio (DM) and Chillotti-Gama-Georgieva-Izabachene (CGGI) schemes for evaluating Boolean circuits and arbitrary functions over larger plaintext spaces using lookup tables.

OpenFHE also includes the following multiparty extensions of FHE:

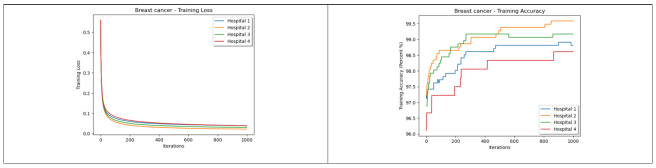
1) Threshold FHE for BGV, BFV, and CKKS schemes
2) Proxy Re-Encryption for BGV, BFV, and CKKS schemes

In our project we will demo three homomorphic encryption schemes: BGV, BFV, CKKS, the implement for multiparty clients and server. Each scheme has three main processes:

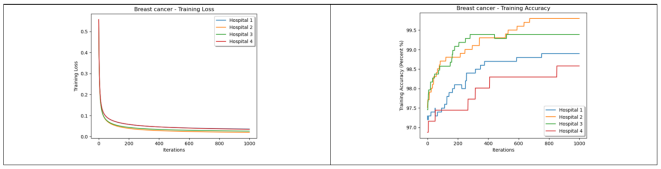
1) **key generation.cpp:** This program is responsible for setting up the necessary parameters for the cryptographic scheme. It initializes the cryptographic context based on the selected homomorphic encryption (HE) scheme, defining key parameters such as polynomial modulus degree, scaling factors, and security levels.
2) **client.cpp:** This program handles several critical functions related to encryption and data processing. First, it loads the cryptographic context from the preconfigured parameters to ensure consistency in encryption operations. Next, it retrieves the private key required for decryption. The client then encrypts raw data (e.g., model updates, gradients, or sensitive information) into ciphertext before sending it to the server for federated learning computations.
3) **server.cpp:** This program is responsible for performing homomorphic operations and federated learning aggregation on encrypted data. It begins by loading the cryptographic context and retrieving the public key required for encrypted computations. Since the server does not possess the private key, it cannot decrypt individual ciphertexts but can still perform mathematical operations on encrypted data using homomorphic encryption techniques. [22] It executes tasks such as encrypted addition, multiplication, and federated model aggregation (e.g., summing encrypted model updates from multiple clients).

C. Testing

1) Federated learning with BFV scheme:



2) Federated learning with BGV scheme:



3) Federated learning with CKKS scheme:

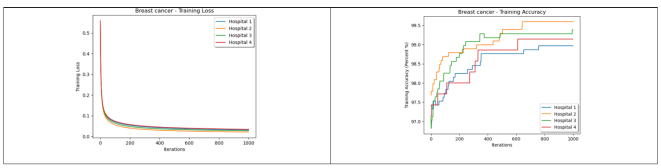


TABLE IV
COMPARE DIFFERENT SCHEMES

	BFV scheme	BGV scheme	CKKS scheme
Training	12 mins	14 mins	20 mins
Accuracy	87.76%	89.81%	85.71%

D. Data Preprocessing

- Our chosen dataset has 32 features, here is the fifth sample:

	id	fractal_dimension_worst	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	c
0	842302	0.11890	17.99	10.38	122.80	1001.0	0.11840	0.27760	
1	842517	0.08902	20.57	17.77	132.90	1326.0	0.08474	0.07864	
2	84300903	0.08758	19.69	21.25	130.00	1203.0	0.10960	0.15990	
3	84348301	0.17300	11.42	20.38	77.58	386.1	0.14250	0.28390	
4	84358402	0.07678	20.29	14.34	135.10	1297.0	0.10030	0.13280	

5 rows × 32 columns

- We define some functions in order to randomly split this dataset to:
  - Training dataset (80%)
  - Testing dataset (20%)

```
def scale_dataset(df, overSample=False):
    # split to fetures and diagnostic result
    X = df[df.columns[:-1]].values
    Y = df[df.columns[-1]].values

    # standardize the input features
    scaler = StandardScaler()
    X = scaler.fit_transform(X)

    # balance the class distribution
    if overSample:
        ros = RandomOverSampler()
        X, Y = ros.fit_resample(X, Y)

    data = np.hstack((X, np.reshape(Y, (-1, 1))))

    # convert to tensor context
    X_train_tensor = Variable(torch.tensor(X, dtype = torch.float32))
    Y_train_tensor = Variable(torch.tensor(Y, dtype = torch.float32))
    data_tensor    = Variable(torch.tensor(data, dtype = torch.float32))

    return data_tensor, X_train_tensor, Y_train_tensor
```

- Here is the result after preprocessing data:

```
# split dataframe to train and test df
df_train, df_test = np.split(df.sample(frac=1), [int(0.8 * len(df))])

# scaling and convert to tensor context
train, X_train, Y_train = scale_dataset(df_train, True)
test , X_test , Y_test  = scale_dataset(df_test , False)
train
```

```
tensor([[-0.2300,  0.8056,  1.9740,  ...,  2.1708,  0.0703,  1.0000],
        [ 0.4371,  2.7576,  0.1243,  ...,  0.8839,  2.2075,  1.0000],
        [-0.2305,  0.0424, -1.2279,  ..., -0.2589, -0.4387,  0.0000],
        ...,
        [-0.2306,  0.6582, -0.9165,  ...,  0.4374,  0.1283,  1.0000],
        [-0.2306,  1.5545,  1.4956,  ...,  0.9971,  1.3022,  1.0000],
        [-0.2305, -0.5608, -0.1453,  ...,  0.0653, -0.3907,  1.0000]])
```

- We will use a Logistic Regression model for breast cancer diagnosis, a simple yet effective machine learning algorithm for binary classification. First, we define the model by selecting relevant features from the dataset, preprocessing the data, and splitting it into training and testing sets. The model is then trained using an optimization algorithm to learn the relationship between input features and the likelihood of a tumor being benign or malignant. Once trained, we evaluate its performance using metrics like accuracy and precision before deploying it for real-world diagnosis.

```
class LogisticRegression(torch.nn.Module):
    def __init__(self, num_features):
        # init super class of LogisticRegression
        super(LogisticRegression, self).__init__()

        # create "linear neural network"
        input_dim  = num_features
        output_dim = 1
        self.linear = torch.nn.Linear(input_dim, output_dim)

        # initialize Weights and Bias
        self.linear.weight.detach().zero_()
        self.linear.bias.detach().zero_()

    def forward(self, x):
        return torch.sigmoid(self.linear(x))
```

E. Sigmoid Function

The sigmoid function, also known as the logistic function, is a fundamental component in logistic regression and neural networks. Defined as:

$$\sigma(z) = \frac{1}{1+e^{-z}} = \frac{e^z}{1+e^z}$$

where $z = w^T x + b$ is the linear output of the model.

1) Key Properties:

- **Probability Interpretation:**
  - Transforms any real-valued input to the range (0,1)
  - Output can be interpreted as $P(y=1|x)$ in binary classification
  - Decision boundary typically set at 0.5:

$$\hat{y} = \begin{cases} 1 & \text{if } \sigma(z) \geq 0.5 \\ 0 & \text{otherwise} \end{cases}$$

- **Differentiability:**
  - Has a simple derivative that can be expressed in terms of itself:

$$\frac{d\sigma}{dz} = \sigma(z)(1-\sigma(z))$$

  - Enables efficient gradient computation during backpropagation
  - Avoids discontinuities present in step functions
- **Behavior Analysis:**
  - Saturation regions:

$$\lim_{z\to+\infty} \sigma(z) = 1$$
$$\lim_{z\to-\infty} \sigma(z) = 0$$

  - Midpoint at $\sigma(0) = 0.5$
  - Steepest slope occurs at $z = 0$ with derivative = 0.25

2) Practical Considerations:

- **Advantages:**
  - Smooth gradient prevents jumps in parameter updates
  - Probabilistic interpretation aligns well with classification
  - Monotonicity preserves order of predictions
- **Limitations:**
  - Vanishing gradients in saturation regions
  - Outputs not zero-centered (can slow learning)
  - Computational expense of exponential function
- **Alternatives:**
  - Hyperbolic tangent (tanh): Ranges (-1,1)
  - ReLU: Faster computation for hidden layers
  - Softmax: For multi-class classification

3) Implementation Example: In Python with NumPy:

```
def sigmoid(z):
    return 1 / (1 + np.exp(-z))

def sigmoid_derivative(z):
    s = sigmoid(z)
    return s * (1 - s)
```

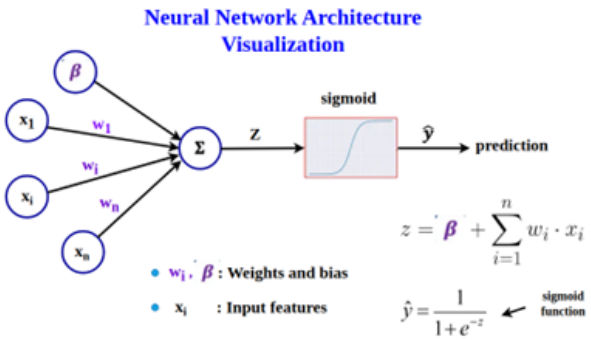


Fig. 4. Overview of Logistic Regression

F. Training Process

Firstly, we should assign some hyperparameters that control the learning behavior of the model and impact its performance.

1) Parameter Definitions::

- **Epoch:** Indicates the number of complete passes through the entire training dataset. A higher number of epochs allows the model to learn better but may lead to overfitting if too high. [6]
- **Learning Rate:** A crucial hyperparameter that determines the step size at each iteration of the optimization algorithm. A small learning rate ensures stable convergence but may slow down training, whereas a high learning rate speeds up training but risks overshooting the optimal solution.
- **Batch Size:** The number of training samples processed in one forward/backward pass. Smaller batches introduce more noise but can improve generalization, while larger batches provide more stable gradient estimates but may require more memory.
- **Optimizer:** The algorithm used to update model weights. Common choices include:
  - **SGD (Stochastic Gradient Descent):** Basic optimizer that applies a fixed learning rate.
  - **Adam:** Adaptive optimizer that adjusts learning rates per parameter, often leading to faster convergence.
  - **RMSprop:** Adapts learning rates based on a moving average of squared gradients, useful for recurrent networks.
- **Loss Function:** Measures how well the model's predictions match the true labels. Common loss functions include:
  - **Binary Cross-Entropy:** Used for binary classification tasks.
  - **Categorical Cross-Entropy:** Used for multi-class classification.
  - **Mean Squared Error (MSE):** Used for regression tasks.
- **Regularization Techniques:** Methods to prevent overfitting, such as:
  - **L1/L2 Regularization:** Adds penalty terms to the loss function to constrain weight magnitudes.
  - **Dropout:** Randomly deactivates neurons during training to improve generalization.
  - **Early Stopping:** Halts training when validation performance stops improving.

2) Training Loop:: The training process consists of the following iterative steps:

1) **Forward Pass:** Input data is passed through the model to generate predictions.
2) **Loss Computation:** The difference between predictions and true labels is calculated using the chosen loss function.
3) **Backward Pass (Backpropagation):** Gradients of the loss with respect to model weights are computed.
4) **Weight Update:** The optimizer adjusts the model weights based on the computed gradients.
5) **Evaluation:** Model performance is periodically assessed on a validation set to monitor generalization. [5]

This process repeats for the specified number of epochs until the model converges or early stopping criteria are met.

3) Monitoring and Validation::

- Track training and validation loss to detect overfitting (training loss decreases while validation loss increases).
- Use metrics such as accuracy, precision, recall, and F1-score for classification tasks, or MAE and R^2 for regression.
- Visualize learning curves to assess model behavior over time.

4) Advanced Techniques::

- **Learning Rate Scheduling:** Dynamically adjusts the learning rate (e.g., Step Decay, Cosine Annealing).
- **Data Augmentation:** Artificially expands the training set by applying transformations (e.g., rotations, flips for images).
- **Transfer Learning:** Leverages pre-trained models (e.g., ResNet, BERT) for improved performance on related tasks.
- **Hyperparameter Tuning:** Optimizes hyperparameters using methods like Grid Search, Random Search, or Bayesian Optimization.

V. Conclusion

Privacy preservation has become an essential practice for healthcare institutions, as it is mandated by regulatory frameworks in both the EU (e.g., GDPR) and the US (e.g., HIPAA). In this context, Federated Learning (FL) and Homomorphic Encryption (HE) play a crucial role in maintaining data security while enabling collaborative model training without exposing sensitive patient information.By leveraging both techniques, the proposed model achieves competitive performance, demonstrating classification metrics such as accuracy, F1-score, precision, and recall exceeding 80% for both encrypted and plaintext data in different federated learning scenarios. However, while the model benefits from enhanced privacy and security, there is a significant trade-off in execution time and the number of participating clients due to the computational overhead of homomorphic encryption. [24]Privacy attacks pose severe risks to patient data security, potentially hindering advancements in healthcare driven by machine learning models. Without robust privacy-preserving mechanisms, healthcare institutions may face legal, ethical, and operational challenges in utilizing sensitive medical data. Therefore, it is imperative to implement stronger safeguards, not only for protecting the data itself but also for ensuring secure data processing techniques.

Furthermore, the integration of these privacy-preserving techniques ensures compliance with strict data protection regulations, fostering trust among healthcare providers and patients. The study also highlights the scalability challenges associated with homomorphic encryption and suggests potential optimizations to improve efficiency. Future research could explore hybrid approaches that balance security and computational cost while maintaining high model accuracy. Additionally, advancements in hardware acceleration, such as GPUs and specialized encryption processors, could significantly enhance the feasibility of these methods in real-world healthcare applications. [25] Ultimately, this research contributes to the broader effort of making secure, privacy-preserving machine learning a standard practice in healthcare.

References

[1] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," 2017.

[2] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2009.

[3] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[4] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the gan: Information leakage from collaborative deep learning," 2017.

[5] F. Konečný, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," *arXiv preprint arXiv:1610.02527*, 2016.

[6] C. Dwork, "Differential privacy," 2006.

[7] I. Mironov, "Rényi differential privacy," 2017.

[8] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Arcas, "Communication-efficient learning of deep networks from decentralized data," 2017.

[9] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. V. D. Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[10] Z. Yang, S. Shen, H. Yu, and X. Li, "A survey of privacy-preserving federated learning," *IEEE Transactions on Artificial Intelligence*, vol. 3, no. 2, pp. 169–181, 2022.

[11] A. Shokri and V. Shmatikov, "Privacy-preserving deep learning," 2015.

[12] C. Gentry, "Fully homomorphic encryption using ideal lattices," 2009.

[13] A. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," 2017.

[14] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof systems," *SIAM Journal on Computing*, vol. 18, no. 1, pp. 186–208, 1989.

[15] K. Bonawitz, H. B. McMahan, E. Munoz *et al.*, "Towards federated learning at scale: System design," 2019.

[16] Y. Liu, J. Dolan-Gavitt, and R. Karri, "Fine-pruning: Defending against backdooring attacks on deep neural networks," 2018.

[17] J. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," 2016.

[18] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision (IJCV)*, vol. 60, no. 2, pp. 91–110, 2004.

[19] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. Wiley, 2006.

[20] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," 2015.

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," 2012.

[22] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2016.

[24] J. W. Mickens and M. D. Ernst, "Rethinking the api for strictly consistent cloud storage," 2013.

[25] M. Zinkevich, "Online convex programming and generalized infinitesimal gradient ascent," 2003.