DATA-PRIVACY FOCUSED FEDERATED LEARNING FRAMEWORK FOR INDUSTRIAL IOT

R25-039

Weerasinghe K.M– IT21831904

Supervisor: Mr. Amila Nuwan Senerathne
Co-Supervisor: Tharaniwarma Kumaralingam

B.Sc. (Hons) Degree in Information Technology Specialized in Cyber Security

Department of Computer Systems and Engineering

Sri Lanka Institute of Information Technology

Sri Lanka

August 2025

DATA-PRIVACY FOCUSED FEDERATED LEARNING FRAMEWORK FOR INDUSTRIAL IOT

R25-039

Weerasinghe K.M– IT21831904

Supervisor: Mr. Amila Nuwan Senerathne
Co-Supervisor: Tharaniyawarma Kumaralingam

B.Sc. (Hons) Degree in Information Technology Specialized in Cyber Security

Department of Computer Systems and Engineering

Sri Lanka Institute of Information Technology

Sri Lanka

August 2025

DECLARATION

I declare that this is our own work, and this proposal does not incorporate without acknowledgment any material previously submitted for a degree or diploma in any other university or institute of higher learning and to the best of our knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text. Also, I hereby grant to Sri Lanka Institute of Information Technology the nonexclusive right to reproduce and distribute m y dissertation, in whole or in part of print, electronic or other medium. I retain the right to use this context as a whole or part in future works (such as articles or books)

Name	Student ID	Signature
Weerasinghe K.M	IT21831904	

The candidate above is carrying out research for the undergraduate dissertation under supervision of the undersigned.

The above candidates are carrying out research for the undergraduate Dissertation

under my supervision.

Signature of the supervisor: _____

Name of the supervisor: Amila Nuwan Senarathne

Date: _____

I hereby approve of the research carried out by the above candidate.

Signature of the Co-supervisor: _____

Name of the Co-supervisor: Tharaniwarma Kumaralingam

Date: _____

ABSTRACT

The emergence of the Industrial Internet of Things (IIoT) has offered new potential for data-driven intelligence but also posed crucial data protection and reliability problems. Federated Learning (FL) offers a possible alternative by enabling collaborative model training without sharing raw data. However, conventional FL remains vulnerable: central servers can infer sensitive information from individual updates, and unreliable distributed networks enhance the danger of client failures, hurting the learning process.

This paper presents a Data-Privacy Focused Federated Learning Framework with a Secure Aggregation Module (SAM) to address these challenges. The module implements a two-layer cryptographic protocol—pairwise masking to conceal individual updates and Shamir's Secret Sharing (SSS) for threshold-based recovery—to assure both privacy and resilience against client failures. The protocol was developed in Python with gRPC for multi-round communication and PyTorch for training a Convolutional Neural Network (CNN). Experimental results show that the framework securely aggregates updates, tolerates client dropouts up to a verifiable threshold, and yields final models identical to non-secure aggregation baselines. These findings demonstrate the practicality and usefulness of the approach for providing secure, privacy-preserving federated learning in distributed environments.

ACKNOWLEDGEMENT

I convey my profound gratitude to everyone who supported me throughout this research project, through thoughts, words, and action.

I am extremely grateful to my supervisor, Amila Nuwan Senerathne, for his continuous guidance, support, and helpful advise from the beginning phases of the thesis to its completion. His knowledge and feedback were important in shaping this work.

Additionally, I want to thank Tharaniwarma Kumaralingam, my co-supervisor, for his insightful comments and recommendations, which significantly raised the caliber of my thesis.

My deepest gratitude also go to the faculty and staff of the Department of Computer Systems Engineeringat the Sri Lanka Institute of Information Technology, for providing a supportive environment and all the required resources for my research.

Finally, I convey my heartfelt appreciation to my family and friends for their unwa- vering support, patience, and understanding during this journey.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AI – Artificial Intelligence

CNN – Convolutional Neural Network

FL – Federated Learnin

HE – Homomorphic Encryption

HFL – Hierarchical Federated Learning

IIoT – Industrial Internet of Things

ML – Machine Learning

PRG – Pseudorandom Generator

PWM – Pairwise Mask Key

PMS – Personal Mask Seed

SEK – Share Exchange Ke

SAM – Secure Aggregation Module

SSS – Shamir’s Secret Sharing

SMPC – Secure Multi-Party Computation

TUI – Terminal User Interface

1. INTRODUCTION

The Industrial Internet of Things (IIoT) is rapidly expanding and revolutionizing operations across various sectors, including manufacturing, energy, healthcare, and logistics. This transformation is occurring because of the substantial volumes of data generated by interconnected sensors and devices. This information is employed to instruct robust AI and ML models in executing tasks such as predictive maintenance, automated quality control, and process optimization [1].However, the reason that this data is so important relates directly to how sensitive information is. Factory operating data, proprietary process parameters, and patient health metrics are highly confidential, and standard centralized ML algorithms, which involve aggregating this raw data on a central server, generating major privacy and security concerns [2].

To address this critical challenge, Federated Learning (FL) has emerged as a decentralized training paradigm. In the FL method, a shared global ML model is trained jointly across numerous devices without the requirement of submitting the private, local data to a central server. This strategy inherently increases privacy by limiting raw data at the edge. However, the FL process itself is not inherently secure. While raw data remains on-device, the model changes that each client transmits to the central aggregator can still be unsecure. Research has found that these updates can be reverse-engineered by an honest-but-curious server or a malicious actor to deduce sensitive information about the private data on which they were trained [3] [4].

This problem necessitates the implementation of Secure Aggregation, a series of cryptographic algorithms meant to ensure that the central server can only learn the sum of all client updates, not the individual contributions [4]. Modern protocols that are suitable for real-world, unstable networks usually utilize a sophisticated, two-layer strategy to attain both privacy and robustness. The essential strategy for privacy often involves pairwise masking, wherein every pair of clients in a training round secretly agrees on a randomized, high-dimensional vector. One client adds this vector to its real model update, whereas the other eliminates it. When the server averages all the received masked updates, these paired masks are mathematically required to cancel each other out, leaving only the sum of the genuine updates [4]. While this effectively hides individual contributions, it causes an inherent vulnerability: if a single client drops out before sending its update, its masks stay uncancelled in the aggregate, hurting the entire round. To solve this, these protocols have been enhanced with a threshold cryptography technique such as Shamir's Secret Sharing as an effective recovery method. These second layer allows the online clients to securely transmit to the server with just enough information and a threshold of shares to reconstruct and remove the masks of the dropped-out clients, thus ensuring the integrity of the final amount without compromising the privacy of any online participant [5].

While these protocols offer solid guarantees for security and reliability, their design frequently assumes a relatively uniform environment of powerful smart devices, such as smartphones. This assumption does not hold in the IIoT ecosystem, which is characterized by significant device variety, ranging from powerful edge gateways to extremely resource-constrained microcontrollers

and sensors. Applying a unified, high-security cryptographic technique across such a varied range of devices is inefficient and often infeasible. This study tackles this research gap by presenting a Data-Privacy Focused Federated Learning Framework for Industrial IoT. This component's key contribution is the design and implementation of a Secure Aggregation Module (SAM) that utilizes proven secure aggregation techniques for the particular problems of IIoT. This component offers a novel approach called Resource-Aware Masking Precision, which enables the protocol to dynamically adapt its cryptographic security level to match the capabilities of different devices. This establishes a practical trade-off between maximal security (using 64-bit precision for masks) and resource efficiency (using 32-bit precision), enabling a more scalable and practicable implementation across diverse IIoT networks. This paper provides the architecture of this protocol, its implementation using a PyTorch-based CNN trained on the CIFAR-10 benchmark dataset, and an examination of its performance.

1.1 Background Study

Basic Secure Aggregation with masking

Federated Learning keeps raw data from being shared, but it leaves clients' model updates open to inference attacks by the central server [6]. The most effective defense against this threat is Secure Aggregation [5]. Additive masking is the most basic conceptual method for secure aggregation. In this scheme, a client wishing to share a secret value $\boldsymbol{x}$ can split it into multiple random shares, such as $\boldsymbol{s_1, s_2, \dots, s_n}$. These shares are constructed so that their sum equals the original value:

$$\boldsymbol{x = s_1 + s_2 + s_3 + \dots + s_n}$$

These shares are then divided between the other participants by the client. Without ever seeing a client's initial value (x), the aggregator can calculate the total sum of all clients' values by adding up the associated shares it receives.

Although this approach offers great privacy, it has a serious drawback that renders it unsuitable for distributed systems in the real world: it is not resilient to failures. If even one of the clients disconnects or fails to send its shares, the complete aggregation process fails because the final sum cannot be correctly computed. This fragility becomes especially critical in IIoT and mobile environments where network connections are unreliable and client loss of data are common. Because of this constraint, more sophisticated, useful protocols that offer robustness and privacy have been established. Pairwise masking for privacy and threshold cryptography for robustness are the two primary components of such a workable secure aggregation.

Advanced Secure Aggregation

In modern day masking is used, specifically pairwise masking is the main method used to protect the privacy of individual model updates. This solution cleverly hides each client's genuine update by injecting cryptographic "noise" in a manner that this noise is eliminated out during the calculation.

Threshold Cryptography for Robustness

Masking addresses the privacy issue, however by itself, it is not resistant to client dropouts. A second cryptographic layer is necessary to resolve this issue. This recovery technique is based on Shamir's Secret Sharing (SSS) algorithm. A secret can be divided into several parts, known as shares, using this efficient method known as threshold cryptography. A secret S can be divided into n shares, and the scheme is configured with a threshold ***t (t ≤ n).*** Any t or more shares can be put together to reassemble the initial secret S, but any t-1 or less shares disclose no information about it [7].

In a strong secure aggregation technique, SSS is employed as the recovery mechanism. Furthermore, customers exchange secrets about their private keys and masks with SSS, but not their model data. In the event that a client terminates, the server can get a "threshold" of the shares from the remaining online clients. This enables the server to retrieve the proper result by recalculating the masks it would have used when reconstructing the secrets of client and mathematically removing them from the aggregate sum [7].

Navigating IIoT System Diversity

A crucial challenge in implementing advanced cryptographic algorithms to IIoT contexts is the large variety in device capabilities. In comparison to the very unified environment of modern smartphones, IIoT networks comprise a variety of devices, from robust edge servers to low-power microcontrollers with constrained memory, computing power, and energy constraints [8]. Constrained devices may be overwhelmed by cryptographic protocols created with the assumption that all devices have equivalent processing capabilities. This might lead the aggregated model to become inaccurate, which could result in higher client dropout rates, higher latency, or even the most constrained devices being incapable to engage in federated learning. Even though threshold cryptography and pairwise masking offer solid privacy and resilience foundations, the variety of device capabilities in the IIoT make their application challenging. Resource-constrained devices may struggle with the computational and communication needs of typical secure aggregation techniques, underscoring the need for ways that balance security, efficiency, and adaptability across a broad IIoT ecosystem

1.2 Literature review

Federated Learning (FL) allows distributed devices to collaboratively train a shared model without transferring raw data, making it well-suited for the IIoT. However, model updates can still leak sensitive information. Secure aggregation protocols address this by enabling the server to compute aggregated updates without accessing individual contributions. Designing such protocols for IIoT entails balancing privacy, efficiency, and model utility, and research has progressively evolved from basic privacy-preserving methods to attack-resilient and efficiency-optimized solutions.

- Secure Aggregation Techniques

Bonawitz et al. (2017) introduced a pairwise masking approach for mobile federated learning, integrating cryptographic primitives like Shamir's Secret Sharing and double masking to protect privacy while tolerating client failures. While robust, this approach incurs substantial computational and communication overhead, acceptable for mobile devices but problematic for IIoT networks with low-power sensors.

Homomorphic encryption (HE) has been proposed to eliminate masking and ensure updates remain encrypted during aggregation (Sanon et al., 2022). HE improves confidentiality but imposes heavy computation and ciphertext expansion, exceeding the capabilities of many IIoT devices. Similarly, chain-based aggregation techniques reduce per-client complexity but introduce fragility: a broken or compromised link can halt aggregation, which is concerning in unstable or adversarial IIoT networks.

To handle malicious updates, robust aggregation strategies based on the Geometric Median (GM) and Auto-weighted Geometric Median (AutoGM) have been suggested (Li et al., 2021). These methods tolerate outliers but increase computational cost. Additionally, backdoor-resilient systems like SECURE Learning use Oblivious Random Grouping (ORG) and Partial Parameter Disclosure (PPD) to detect anomalies without compromising individual privacy.

Mansouri et al. (2023) provide a comprehensive survey of secure aggregation protocols, noting that masking-based approaches remain the most common but emphasizing that privacy alone is insufficient; additional mechanisms such as differential privacy are often required. Across these studies, a recurring theme emerges there is a trade-off between cryptographic strength and computational/communication efficiency, and most protocols assume uniform client capabilities.

In conclusion, Existing protocols typically enforce identical, high-security operations across all devices. In IIoT environments, however, devices vary from powerful edge controllers to low-power sensors. Applying uniform cryptographic operations can be inefficient or even infeasible, potentially excluding critical data sources from training. This highlights a **pressing gap**: the need for **adaptive secure aggregation algorithms** that account for heterogeneous resources while maintaining privacy and model utility. Addressing this gap is the central motivation for this thesis.

1.3 Research Gap

A critical analysis reveals that, even though secure aggregation protocols have developed to offer privacy and attack resilience in federated learning, they often rely on the assumption of uniform client capabilities, a model that is practically not in line with the diverse reality of Industrial IoT (IIoT) networks. IIoT ecosystems are composed of low-power sensors and actuators coexisting beside large edge gateways, which makes resource-aware, flexible security measures are critical.

A careful examination of literature highlights the following limitations:

Masking-based protocols like Bonawitz et al. (2017) are now considerably more efficient, with recent single-setup techniques lowering user-side processing by up to 99%. Currently, the emphasis is on maintaining long-term security spanning several rounds, including forward and backward secrecy. In contrast, homomorphic encryption remains unfeasible for IIoT devices due to its heavy computation and excessive data expansion, which overload restricted bandwidth.

Aggregation methods like AutoGM are integrity-focused robust aggregation rules, not privacy protocols. Their objective is to defend against Byzantine failures and poisoning assaults. Their fundamental and critical shortcoming in the IIoT context is a serious performance drop when faced with non-IID heterogeneous data. Because they perceive statistical outliers as malevolent, they typically ignore good but rare data from honest customers, causing model accuracy to collapse to the level of random guessing in some circumstances [1].

The trade-off between efficiency and privacy is not merely a design decision; it is a fundamental limitation. It is impossible to optimize privacy, model utility, and efficiency. Most present protocols adopt a static, optimal security approach, failing to provide the tools to dynamically negotiate this trade-off space based on the variable resource availability and security requirements inside a heterogeneous IIoT network.

Client-server designs encounter scalability challenges across thousands of Varied IIoT devices, notably due to slow devices. Hierarchical Federated Learning (HFL) tackles this by leveraging edge servers for local aggregation, lowering central server load and boosting overall scalability.

A critical analysis reveals that, even though secure aggregation protocols have developed to offer privacy and attack resilience in federated learning, they often rely on the assumption of uniform client capabilities, a model that is practically not in line with the diverse reality of Industrial IoT (IIoT) networks. IIoT ecosystems are composed of low-power sensors and actuators coexisting beside large edge gateways, which makes resource-aware, flexible security measures are critical.

A careful examination of literature highlights the following limitations:

Masking-based protocols like Bonawitz et al. (2017) are now considerably more efficient, with recent single-setup techniques lowering user-side processing by up to 99%. Currently, the emphasis is on maintaining long-term security spanning several rounds, including forward and backward secrecy. In contrast, homomorphic encryption remains unfeasible for IIoT devices due to its heavy computation and excessive data expansion, which overload restricted bandwidth.

Aggregation methods like AutoGM are integrity-focused robust aggregation rules, not privacy protocols. Their objective is to defend against Byzantine failures and poisoning assaults. Their fundamental and critical shortcoming in the IIoT context is a serious performance drop when faced with non-IID heterogeneous data. Because they perceive statistical outliers as malevolent, they typically ignore good but rare data from honest customers, causing model accuracy to collapse to the level of random guessing in some circumstances [1].

The trade-off between efficiency and privacy is not merely a design decision; it is a fundamental limitation. It is impossible to optimize privacy, model utility, and efficiency. Most present protocols adopt a static, optimal security approach, failing to provide the tools to dynamically negotiate this trade-off space based on the variable resource availability and security requirements inside a heterogeneous IIoT network.

Client-server designs encounter scalability challenges across thousands of Varied IIoT devices, notably due to slow devices. Hierarchical Federated Learning (HFL) tackles this by leveraging edge servers for local aggregation, lowering central server load and boosting overall scalability.

Protocol / Feature	Goal	Handles Heterogeneous Clients	Low Computation	Low Communication	Byzantine Mitigation	IIoT suitability
Bonawitz et al., 2017 (Original)	Privacy	-	✕	✕	✕	✕
Single-Setup Masking (Modern)	Privacy	-	✓	✓	✕	✓
Homomorphic Encryption	Privacy	-	✕	✕	✕	
AutoGM (Robust Aggregation)	Integrity	✕	✓	✓	✓	✕
Hierarchical Architectures (HFL)	System Efficiency	✓	✓	✓	-	✓

Table 1: Research gap Overview

1.4 Research problem

TThe primary problem is in the practical implementation and cryptographic validation of a secure aggregation protocol for federated learning. While Federated Learning (FL) intrinsically promotes privacy by keeping raw data on local devices, the model updates delivered to a central server remain subject to inference attacks that can reveal sensitive information. Furthermore, in real-world distributed systems, client dropouts due to network instability can interrupt the aggregation process, distorting the global model. Although protocols combining pairwise masking for privacy and threshold secret sharing for robustness exist in theory, a comprehensive implementation that validates their correctness, resilience to failures, and security against adversarial threats within a functioning FL workflow is a critical step for practical deployment.

The research is led by the subsequent research questions:

Addressing these challenges allows us to find lightweight alternatives to secure aggregation which maintain privacy, robustness, and efficiency for IIoT contexts.

I. How can a secure aggregation protocol, ensuring privacy against a central server and providing robustness against client dropouts, be effectively implemented within a multi-round federated learning system?
II. How resilient is the implemented protocol to client failures, and at what dropout rate does the aggregation process break down?
III. Can the integration of a robust aggregation rule within this secure framework effectively mitigate the impact of internal threats, such as backdoor attacks from malicious participants?

2. RESEARCH OBJECTIVE

This research focuses on the practical implementation and validation of a secure, robust, and privacy-preserving framework for Federated Learning (FL). The project centers on the design, implementation, and empirical evaluation of a secure aggregation protocol that is cryptographically sound and resilient to common failures and threats in distributed learning environments.

2.1 Main Objectives

The main objective of this research is to develop and implement a secure aggregation framework for Federated Learning that validates the effectiveness of modern cryptographic techniques in strengthening client privacy and providing resilience against client failures.

Many existing secure aggregation frameworks are built for general federated learning contexts and are not optimized for the problems of Industrial IoT. IIoT systems comprise of highly resource-constrained devices, face demanding reliability requirements, and often experience major communication overhead. These limitations render conventional methods less efficacious in practice. This research fills the gap by developing a scalable, lightweight, and privacy-preserving framework specifically designed for IIoT, thereby enhancing the security and usability of Federated Learning in industrial applications.

2.2 Specific Objectives

Design and implement a robust secure aggregation protocol

This involves developing a multi-round protocol that integrates a two-layer system for privacy and robustness

Masking for Privacy: The key privacy approach will be masking. For each clients in a training round, a shared secret will be constructed to generate a random number mask. One client will add this mask to its model update, while the other subtracts it. This ensures that when the server averages all the masked updates, the masks mathematically negate one another out, enabling the server to learn the proper aggregate sum without ever discovering an individual client's actual contribution.

Threshold Cryptography for Robustness: A recovery layer will be created utilizing threshold cryptography to increase the protection of cryptographic masks. Clients will use Thrshold Secret Sharing (SSS) to distribute shares of secrets connected to their private keys and masks. The original secret can only be reconstructed if enough shares have been aggregated, ensuring that none of the participants has complete control and that the aggregation process is secure and reliable.

2.3 Evaluation of the proposed Framework

Correctness: Verifying that the protocol execution of the aggregation component protects client Key generation and its successful multi round training.

Application: Demonstrate the protocol's viability by training a Convolutional Neural Network (CNN) on the CIFAR-10 benchmark dataset in a federated learning scenario, where each client trains the CNN locally on its own data and the server aggregates updates securely using the suggested framework.

Demonstrate Practical Viability: Apply the secure protocol to a standard machine learning task by training a Convolutional Neural Network (CNN) on the CIFAR-10 dataset to ensure the security measures do not compromise model utility.

2.4 Requirement Analysis

This section articulates the functional and non-functional requirements for the Secure Aggregation Module, which is the key component of the proposed Data-Privacy Focused Federated Learning Framework for Industrial IoT. These requirements are drawn from the project objectives and are aimed at ensuring that the outcome is secure, reliable, efficient, and scalable, in keeping with the requirements of the IIoT environment.

- **Functional Requirements**

	Requirement Description
01	Client Key Generation: The system shall enable each client participating in a training round to create two separate pairs of public/private keys: one for establishing paired secrets, the paired Mask Key (PWM), and one for securely exchanging secret shares, the Share Exchange Key (SEK).
02	Public Key Distribution: The system must feature a way for the server to obtain public keys from all contributing clients and broadcast the whole set of public keys to every client in the round.
03	Secret Sharing: Each client will be able to split its private Pairwise Mask Key (PMK) and Personal Mask Seed (PMS) into a predetermined number of shares (n) using Shamir's Secret Sharing (SSS), with a customizable threshold (t) defined as "total number of clients minus half the clients, rounded up"
04	Secure Share Exchange: Each client shall encrypt the shares destined for every other client using a shared secret. The server will serve as a secure channel to forward these encrypted shares to their correct recipients.

05	Model Update Masking: Model Update Masking: Before sending the final masked vector to the server, each client must apply both its personal mask (from its PMS) and the total of all its pairwise masks to its locally computed model update.
06	Secure Aggregation: By adding up all masked updates received and deducing the reconstructed pairwise masks and the reconstructed personal masks of the online clients, the server will be able to correct the sum of the true model updates
07	Configurable Mask Precision: To facilitate a trade-off between security and computational performance, the system must make it feasible to adjust the bit-length of the cryptographic mask

Table 2: Functional Requirements

- **Non-Functional Requirements**

Category	Requirement Description
Security	Confidentiality: The protocol must ensure that the central server cannot infer any information about an individual client's model update. The server should only learn the final aggregated sum.
Robustness	Fault Tolerance: The aggregation process must succeed and then, the final aggregate must correctly represent the sum of updates from only the online clients.
Performance	Efficiency: components must be efficient enough to run on resource-constrained IIoT devices.
Scalability	Client Scalability: The protocol must be able to scale to participating clients.

Table 3: Non-Functional Requirements

3. METHODOLOGY

The content of this section outlines the approach used to design, construct, and evaluate the Data-Privacy Focused Federated Learning Framework for Industrial IoT. The suggested method is built upon a strong system architecture that combines the new Secure Aggregation Module to provide both privacy and resilience. The methodology comprises the overall system architecture, the systematic process of the secure aggregation protocol, an analysis of its key elements, and a method for its practical evaluation.

3.1 System Architecture

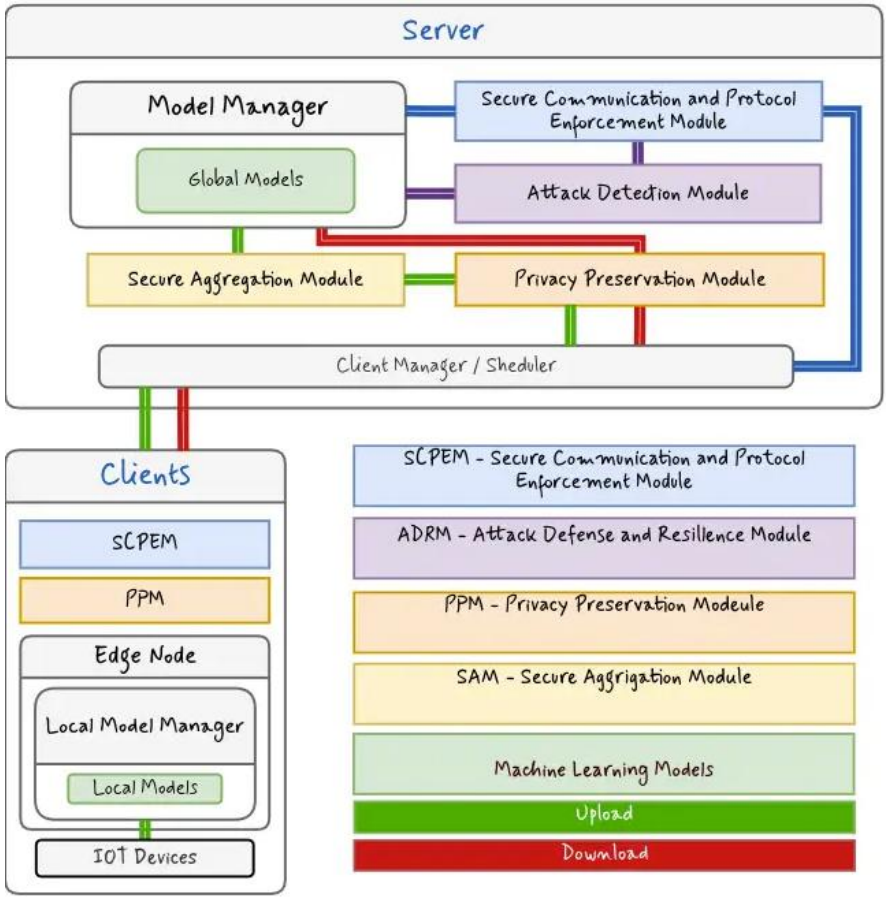


Figure 1: Proposed Overall Framwork

The proposed framework follows a client-server architecture common in Federated Learning. The system consists of a central server and multiple distributed client nodes, representing IIoT devices.

- **Server**: The server orchestrates the entire federated learning process. It contains several key modules:
    - **Model Manager**: Responsible for maintaining the global machine learning model (a CNN), evaluating its performance, and applying aggregated updates.

    - **Client Manager**: Manages the pool of connected clients, tracks their status, and selects participants for each training round.
    - **Secure Aggregation Module (SAM)**: responsible for executing the secure aggregation protocol to compute the sum of true client updates, including handling dropouts.
    - **Secure Command and Control Module (SCPM)**: Manages all communication channels (gRPC) and enforces security policies.
- **Client Nodes (IIoT Devices)**: Each client represents an individual IIoT device. It holds a private local data set and is responsible for:
    - Receiving the global model from the server.
    - Performing local training to compute a model update.
    - Participating in the secure aggregation protocol by generating and applying cryptographic masks to its update before transmission.

For large-scale IIoT deployments, this architecture can be extended hierarchically. As suggested by existing research on FL for IoT [3], clients could be organized into logical clusters that report to local edge aggregators. These edge aggregators would then participate in a higher-level aggregation with the central server. While this hierarchical model is not implemented in the current scope, it represents a clear path for future work to enhance scalability and further reduce communication overhead.

3.2 Individual Component Diagram

The design of the proposed federated learning framework comprises several essential modules tasked with protecting privacy, overseeing the learning lifecycle, and preserving system integrity. The subsequent sections highlight the operation of the Secure Aggregation Module and the fundamental components accountable for system administration and orchestration.

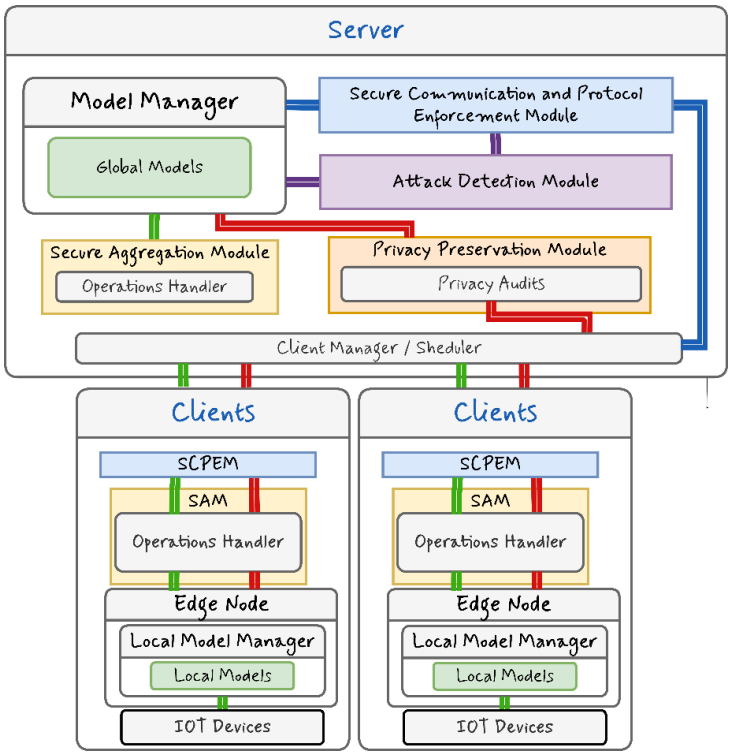


Figure 2: Individual Framework Design with the other components

The Secure Aggregation Module (SAM) is a vital privacy-enhancing element intended to function in conjunction with the Privacy Preservation Module (PPM), allowing the central server to calculate the aggregate of model updates from all participating clients without accessing the raw contributions of any individual client. This protects against disclosure and ensures client data privacy from a possibly curious yet honest server. The system utilizes a dual-protocol cryptographic methodology: Shamir's Secret Sharing (SSS) facilitates fault tolerance by partitioning each client's update into several shares disseminated among peers, enabling recovery provided a threshold of shares is accessible, thereby alleviating the effects of client dropouts. The

Bonawitz protocol, on the other hand, protects core privacy by allowing the server to calculate the total sum directly over cryptographically masked updates. This way, individual contributions are kept concealed and only the total is disclosed. Collectively, these measures ensure a secure and confidential aggregation process, wherein the final model accurately represents authentic client inputs while protecting sensitive information.

The Client Manager is responsible for the complete lifecycle management of client interaction in the federated learning process, providing smooth coordination and efficiency. Client selection and scheduling are two of its most significant jobs. For each training round, participants are carefully recruited based on parameters including network state, resource availability, and connection stability to get the highest performance and dependability. Additionally, it regulates joining and state tracking, overseeing the integration of new clients into the system while continuously monitoring their status whether online, offline, or training to maintain effective participation and support robust federated learning operations.

Orchestration module works as the primary controller for the whole federated learning workflow, directing the process from start to completion. It interfaces with all other system components to ensure a seamless and secure operational flow. Its tasks include starting training rounds, distributing the global model, managing the secure aggregation phase, and enforcing the security and privacy requirements established by the framework throughout the training lifespan.

The Model Manager serves as the repository and controller of the shared global model, assuring its integrity and versioning throughout the training lifecycle. Its functions include secure storage and versioning of model updates, distribution of the current global model to selected clients, and update application, where the final aggregated update produced by the secure aggregation component is applied to the global model. This guarantees that the shared model remains consistent, up-to-date, and indicative of genuine client contributions during training rounds.

Log Manager serves as the system's core record-keeping utility, giving a comprehensive and timestamped log of all system actions. It collects data from every other module, including communication events, client activity, and any anomalies noticed by the security modules. These logs are crucial for real-time monitoring, executing security audits, and debugging the system to maintain operational transparency and reliability

3.3 The Secure aggregation Workflows

The secure aggregation protocol used in federated learning iterations combines ideas from the FastSecAgg protocol by Kadhe et al [9]. and the secure Aggregation protocol by Bonawitz et al [5]. to ensure completeness, efficiency, and clarity. This multi-round procedure establishes the necessary cryptographic secrets and facilitates reliable aggregation in dynamic IIoT contexts.

- Round 0: Advertise Keys

To initiate a training round, the server selects a group of clients. Each client generates:

- Pairwise Mask Key (PWM) – for creating pairwise secrets.
- Share Exchange Key (SEK) – for securely exchanging secret shares.

Server: Online Clients = {C1, C3, C4}, Dropped = {C2}

Clients → send {PMS shares of online, PWM shares of dropped} → Server

- Round 1: Share Keys

Each client uses the broadcast public keys to establish shared secrets with every other client via Diffie-Hellman Concurrently,

- Each client also generates a Personal Mask Seed (PMS).
- The PMK and PMS are split into shares using SSS.
- Shares are encrypted and sent via the server to other clients.

Client → generates {PWM, SEK}

Client → sends Public Keys → Server → broadcasts all Public Keys

- Round 2: Masked Input Collection

Each client decrypts received shares and computes its masked model update:

- Start with True Update.
- Add Personal Mask (from PMS).
- Add Sum of Pairwise Masks (from PWM with other clients).

Shared Secret (ECDH) = f (*Client_1* PubKey, *Client_2* PrivKey)

PMS → split into shares {PMS_1, PMS_2, …}

PWM → split into shares {PWM_1, PWM_2, …} Encrypted Shares → Server → Forwarded to

- Round 3: Announce Online Set & Send Recovery Shares

The server identifies who is online and who has dropped out, then broadcasts the online list.

- Online clients send PMS shares of online clients.
- Online clients send PWM shares of dropped clients.

Shared Secret (ECDH) = f (*Client_1* PubKey, *Client_2* PrivKey)

PMS → split into shares {PMS_1, PMS_2, …}

PWM → split into shares {PWM_1, PWM_2, …} Encrypted Shares → Server → Forwarded to

- Round 4: Final Aggregation

The server reconstructs the missing secrets using recovery shares:

- Rebuild PMS of online clients.
- Rebuild PWM of dropped clients.
- Then it subtracts these masks from the sum of all masked updates.

Reconstructed Masks = (Personal Masks of Online) + (Pairwise Masks of Dropped)

True Aggregated Update = (Masked Updates) – Reconstructed Masks

3.4 Evaluation Methodology

The efficiency of the suggested framework will be verified through a series of experiments utilizing Python and the NumPy library in a simulated setting. The assessment will focus on evaluating scalability, security, and resource efficiency using specific criteria, following best practices from previous research.

- Robustness Against Client Failures: A primary requirement for any practical federated system is its ability to operate reliably despite client dropouts caused by network instability. To validate this, an experiment was designed to measure the Aggregation Success Rate as a function of the Participant Dropout Rate. This test quantifies the protocol's resilience and identifies its operational limits.

- Resilience Against Attacks: To test the framework's resilience against internal threats, an experiment was conducted by introducing a subset of malicious clients into the training process. These clients attempt to corrupt the global model by sending malicious updates designed to make it misclassify specific inputs . The impact of this attack was measured by comparing the backdoor attack's success rate in a standard federated averaging setup versus the rate achieved when the framework's defensive aggregation mechanism is active. This evaluation demonstrates the system's effectiveness in preserving model integrity against targeted poisoning efforts.

3.5 Tools and Techniques

The implementation of the Secure Aggregation Module (SAM) and its related test environment was achieved utilizing a specifically selected set of industry-standard libraries and frameworks. The selection of these tools was influenced by the specific requirements of the protocol, including requirements for high-performance numerical calculation, resilient network connectivity, secure cryptographic operations, and real-time monitoring. This section outlines the primary technologies used in this research.

- **Core Language and Libraries**

Python (3.10+) — Selected as the primary programming language due to its robust environment for computing, machine learning, and cryptography, enabling rapid prototyping and testing of complicated protocols.

NumPy — Used for processing huge multi-dimensional arrays that reflect model updates and cryptographic masks. Its improved vectorized functions allow the high-volume addition and subtraction operations necessary in pairwise masking.

- **Communication Framework**

gRPC + Protocol Buffers — Facilitates client-server communication with minimal latency and robust support for organized multi-round workflows. This was important for executing separate stages of the protocol (key exchange, share distribution, and masked update submission).

- **Cryptographic Libraries**

PyCryptodome – Provided low-level cryptographic primitives, such as Diffie hellman

secret-sharing – Implemented Shamir's Secret Sharing (SSS) for dropout recovery, enabling reconstruction of client secrets when participants fail mid-protocol.

- **Machine Learning Framework**

PyTorch – Used to define and train the SimpleCNN model on CIFAR-10. The protocol assumes that the vectors being securely aggregated are flattened weight updates derived from PyTorch models.

- **Front end and monitoring**

Rich: Developed a terminal-based monitoring interface that tracks aggregation rounds and includes live tables, logs, and panels.

AIOHTTP: Ensured asynchronous, non-blocking data retrieval from the gRPC server by providing a REST API for the monitoring interface

3.6 Commercialization of the project

The proposed framework holds strong commercialization potential due to the growing demand for privacy-preserving data analytics in Industrial IoT environments. Organizations across sectors such as manufacturing, healthcare, logistics, and energy are increasingly adopting IIoT solutions, which generate vast amounts of sensitive operational data. Ensuring the security and privacy of this data during collaborative machine learning is both a regulatory requirement and a competitive necessity.

The key target market for this solution includes organizations and service providers who rely on distributed IoT devices for real-time monitoring and decision-making. By including a Secure Aggregation Module (SAM) that safeguards client updates while preserving robustness against client dropouts, the architecture directly solves one of the most significant difficulties in federated learning adoption. Additionally, the novelty of customizable mask precision allows businesses the ability to balance between computational efficiency and cryptographic strength, making the solution suitable to both high-performance systems and resource-constrained IIoT devices.

The framework's commercialization pathway is strengthened by its clear value proposition: enabling collaborative model training in IIoT environments without compromising data privacy or operational resilience.

4. IMPLEMENTATION

This chapter describes the technological implementation of the Secure Aggregation Module (SAM), which is the core part of the proposed Data-Privacy Focused Federated Learning Framework for Industrial IoT. The implementation was written in Python, employing a suite of solid tools. The architecture emphasizes privacy, robustness aginsts errors, and accuracy of the final aggregated model update.

4.1 System setup and Environment

The implementation and evaluation of the Secure Aggregation Module (SAM) were undertaken within a simulated, multi-process environment to provide controllable and reproducible evaluations, a usual method for testing distributed protocols. The entire system was constructed using Python (version 3.10+), employing a suite of powerful, open-source libraries to meet the complicated requirements of the protocol.

secret-sharing: This library provided a basic and effective implementation of Shamir's Secret Sharing (SSS). It was used to partition client secrets (their private keys and personal random seeds) into numerous shares and to reassemble them on the server-side during the dropout recovery phase, fulfilling the protocol's requirement for a threshold cryptography technique.

NumPy: The NumPy library was necessary for any high-performance numerical calculations. Model updates and cryptographic masks were represented as NumPy arrays, allowing for efficient, vectorized arithmetic. This was important for applying pairwise and personal masks to the model updates, which is the core of the privacy-preserving approach.

PyTorch: While the standalone test environment employs NumPy vectors to represent model updates for simplicity, the system is developed in the broader context of a Federated Learning framework training a Convolutional Neural Network (CNN). PyTorch was used to define the Simple CNN architecture, and the entire secure aggregation pipeline is developed to accommodate the high-dimensional weight vectors produced by such a model.

4.2 Server initialization and core modules

The server works as the central orchestrator and aggregator for the federated learning process. Its configuration is managed using a modular, asynchronous design mediated by the main entry point, server.py.

The server is initialized using an asyncio event loop to perform concurrent tasks, such as maintaining numerous client connections and running the training loop simultaneously. Upon launch, server.py instantiates several crucial singleton classes:

- **Client Manager:** Responsible for tracking the condition of all connected clients. It keeps a persistent record in client_data.json, holding each client's ID, connection state, last heartbeat time, and a reputation score. It also handles the selection of available clients for each training round.
- **Model Manager:** Manages the lifecycle of the global machine learning model (SimpleCNN). It is responsible for loading the starting model, disseminating its state to clients, and applying the final, securely aggregated updates received from the orchestrator.
- **Orchestrator**: This is the central "brain" of the federated learning process. It executes the main training loop, collaborates with the Client Manager to choose clients, and interfaces with the SecureAggregationModule (SAM) to receive incoming model updates.
- **ServerControlPlaneManager (SCPM):** Acts as the communication center for the server. It is responsible for initializing and controlling all network listeners.

The server's communication is handled by SCPM, which launches two distinct listeners managed by specialized handlers:

- **GrpcHandler**: This handler starts a high-performance gRPC server on a secure port (e.g., 50051). Security is enforced using **mutual TLS (mTLS)**. The handler loads the server's private key, its signed certificate, and the Certificate Authority (CA) certificate from the directory. This configuration ensures that only clients presenting a valid certificate signed by the same CA can establish a connection, and all communication is encrypted.
- **ApiHandler**: This handler starts a REST API server (e.g., on port 8080) which serves data to the Text-based User Interface (TUI). This allows for real-time monitoring of the server's status, connected clients, and model performance metrics.

4.3 Client Configuration and core modules

Each client's unique identity is configured through a simple text file, client_id.txt. Upon startup, the client reads this file to establish its ID. If the file does not exist, a new unique ID is generated and saved. This allows for easy simulation of multiple, distinct clients on a single machine. Like the server, the client operates on an asyncio event loop.

The client is configured to communicate exclusively with the server's secure gRPC port. The connection is secured using mTLS. The client loads its own credentials from the directory:

- Its unique private key (client_1.key).
- Its unique certificate is signed by the CA (client_1.crt).
- The public CA certificate (ca.crt) to verify the server's identity.

This configuration ensures that the client only connects to a trusted server and that its own identity is verified by the server, establishing a secure, end-to-end encrypted channel for all communication.

The client's main logic in client.py follows a stateful, asynchronous loop:

1. **Connect**: It establishes a secure gRPC connection to the server.
2. **Heartbeat**: It begins sending periodic heartbeats to the server to signal its availability and maintain its "connected" status in the Client Manager.
3. **Wait for Signal**: The client waits for the server to send a new_round_available = true flag in a heartbeat response.
4. **Perform Training**: Once signaled, the client calls it perform_training_round method. This method fetches the current global model from the server, invokes its local ClientModelManager to perform training on its private dataset, and computes the model update.
5. **Secure and Send**: The client then uses the logic from the SecureAggregationModule to apply cryptographic masks to its update before sending the final result

4.4 Implementation of the front end

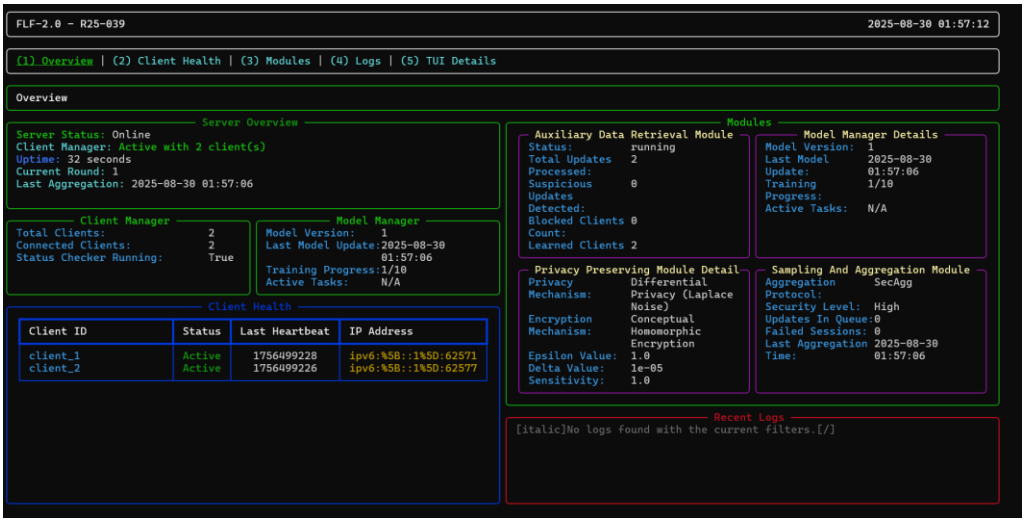


Figure 3: TUI Frontend

This figure displays the custom-built Terminal User Interface (TUI) for server-side monitoring. This dashboard provides a real-time, consolidated view of the entire federated learning framework. It includes panels showing the overall server status, a live list of active clients, and the operational state of the core components. Notably, the **Sampling and Aggregation Module** panel confirms the active protocol and its current parameters, offering an at-a-glance verification of the system's security posture.

4.5 Implementation of Aggregation Module

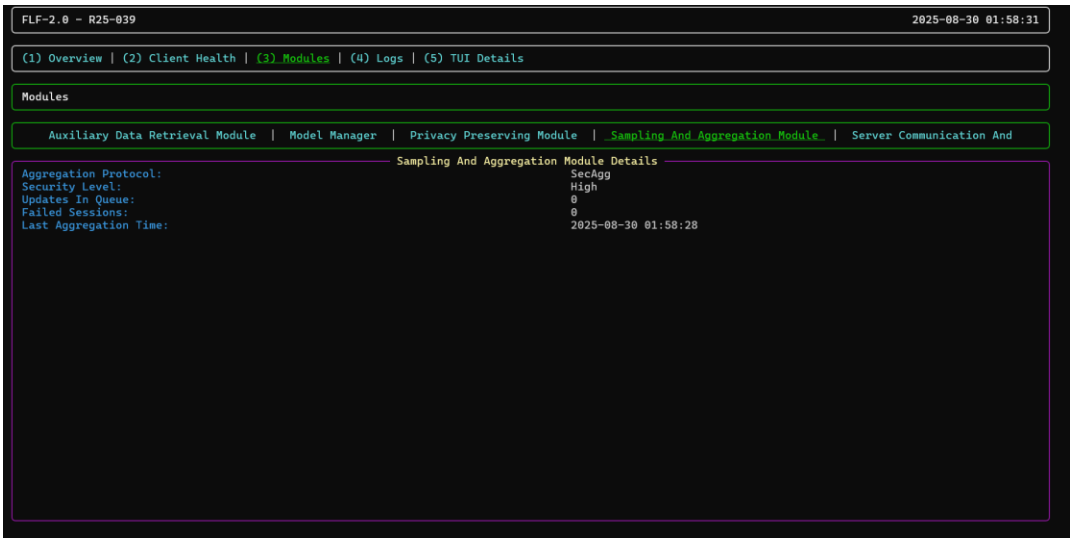


Figure 4: Aggregation module

This figure illustrates the initiation of the secure aggregation process on the server side. Once enough clients join the round, the server coordinates key exchange, share distribution, and the collection of masked updates. The logs displayed in the screenshot confirm the execution of these steps in sequence, demonstrating that the server receives encrypted shares, verifies client participation, and successfully begins the aggregation phase. This provides evidence that the secure aggregation module is functioning as intended, masking individual contributions while preparing for robust and privacy-preserving aggregation

5. RESULTS

This section presents the results obtained from implementing and evaluating the proposed Data-Privacy Focused Federated Learning Framework for Industrial IoT. The experiments were conducted in a simulated environment using Python and the NumPy library, with a focus on measuring scalability, robustness, security, and resource efficiency.

5.1 Threshold Recovery

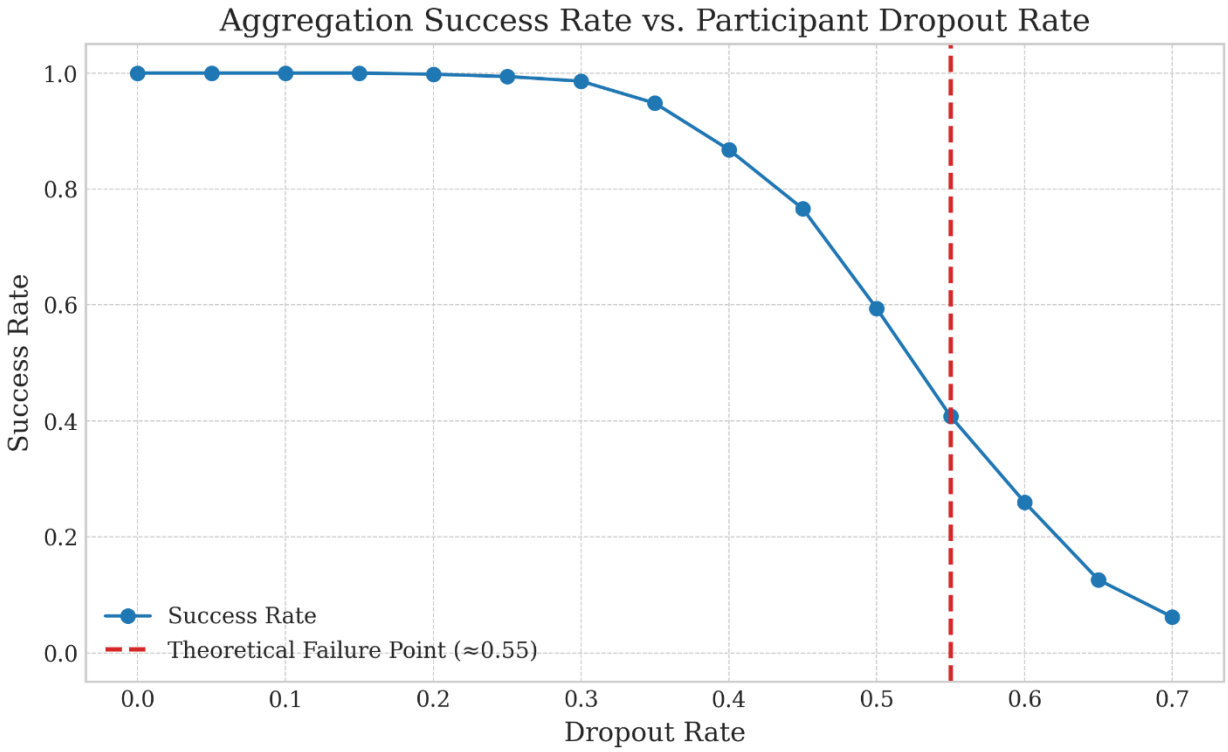


Figure 5: Results of the threshold recovery simulation

The Secure Aggregation Module (SAM) is designed for robustness against client dropouts using a threshold-based recovery mechanism. To validate this, we conducted a simulation within our federated learning framework, which trains a Convolutional Neural Network (CNN) on the CIFAR-10 dataset.

The results, shown in Figure 6, demonstrate high fault tolerance. The simulation confirms a perfect aggregation success rate even when up to 30% of clients fail to participate. This performance aligns with the protocol's theoretical guarantees and confirms its resilience, making it suitable for practical deployment in unstable IIoT environments.

5.2 Resilience Against Backdoor Attacks

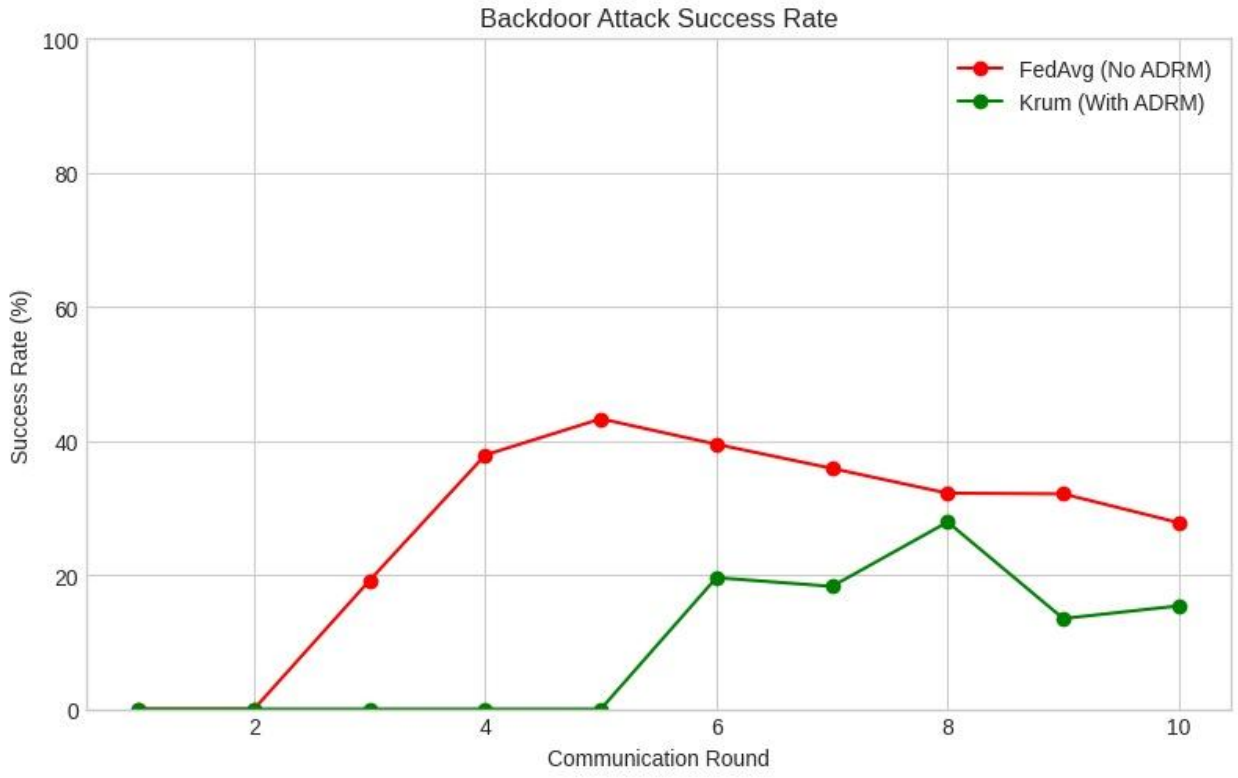


Figure 6: Aggregation and security models effectiveness

To analyze the framework's robustness against internal threats, a simulation was conducted to measure its resilience to backdoor attacks. In this scenario, malicious clients attempt to corrupt the global model by training it to misclassify individual inputs. The efficiency of the framework's security module in conjunction with the Krum robust aggregating technique, was compared against a baseline utilizing ordinary Federated Averaging (FedAvg) without any defense.

The results clearly indicate the important relevance of this defensive aggregation method in protecting model fidelity. The baseline FedAvg protocol showed very vulnerable, with the backdoor attack's success rate climbing rapidly to a peak of over 40%, indicating a serious breach of the global model. In sharp contrast, the framework using Krum in combination with the secure aggregation module successfully suppressed the attack. It kept the success rate at 0% for the initial communication rounds and guaranteed it stayed much lower during the whole training process, never topping 28%. This experiment indicates that adding a robust aggregation method like Krum as part of the secure aggregation process provides a strong resistance against backdoor poisoning attempts, preserving the integrity and stability of the final model for deployment in sensitive IIoT applications.

6. CONCLUSION AND FUTURE WORK

The increasing recognition of data-driven optimization through Federated Learning (FL) demands solid privacy guarantees that traditional protocols fail to deliver. This research dealt with the important problem of building a safe and robust model aggregation system suitable for distributed situations where client failures are common. The essential contribution of this thesis is the design, implementation, and evaluation of a Data-Privacy Focused Federated Learning Framework, containing a core Secure Aggregation Module (SAM). This module effectively provides a strong, multi-round protocol that uses a two-layer cryptographic technique to preserve client data. The experimental results demonstrated that the use of pairwise masking effectively preserves the privacy of individual model updates against a central server. Furthermore, the protocol's threshold cryptography method, founded on Shamir's Secret Sharing, was demonstrated to be a feasible way for offering resistance against client dropouts. The results effectively revealed that the devised protocol is both correct, producing a mathematically accurate aggregate sum—and durable, accepting simulated client failures without compromising the integrity of the training round. This study serves as a proof of concept that it is possible to design a safe and robust aggregation mechanism that does not negatively affect the accuracy of the final machine learning model, hence giving a viable method for deploying privacy-preserving FL in real-world contexts.

6.1 Limitations

While this research effectively achieved its aims, it is crucial to understand its limitations, which provide context for the extent of the findings. All investigations were conducted in a virtual environment on a single system. While this allows for controlled and reproducible findings, it does not represent the real-world difficulties of network latency, packet loss, and the genuine performance characteristics of diverse IIoT hardware. Furthermore, the server-side logic for dropout recovery was conceptual. While the cryptographic mathematics for reconstructing secrets from shares is sound and implemented in the sam.py module, the test server did not execute the complex multi-party communication flow required for the server to securely collect the necessary shares from online clients. Instead, it employed deterministically produced keys to simulate a successful reconstruction.

6.2 Future work

The findings and limitations of this research open up several promising avenues for future work. The most critical next step is to deploy and evaluate the framework on a physical testbed of heterogeneous IIoT devices, such as Raspberry Pis, Jetson Nanos, and ESP32 microcontrollers. This would allow for the collection of real-world performance metrics, including CPU usage, memory footprint, and power consumption, providing a more accurate assessment of the protocol's feasibility on constrained hardware. To create a more comprehensively secure system, the current privacy-preserving protocol could be integrated with a robust aggregation algorithm designed to defend against malicious clients, such as Auto-weighted Geometric Median (AutoGM), to provide a multi-layered defense. A significant engineering effort would be to implement the complete, secure multi-party computation protocol for the unmasking phase, moving beyond the current conceptual simulation. Lastly, future work could explore a dynamic protocol where clients advertise their computational capabilities to the server, allowing the server to intelligently orchestrate the training round by grouping clients or dynamically adjusting cryptographic parameters to further optimize the balance between security and performance in a heterogeneous network.

7. REFERENCES

[1] Shenghui Li, Edith Ngai,Thiemo Voigt,, "Byzantine-Robust Aggregation in Federated Learning," IEEE, 2021.

[2] Mohamad Mansouri, Melek Önen, Wafa Ben Jaballah, Mauro Conti, "SoK: Secure Aggregation Based on Cryptographic Schemes for," petsymposium.org, 2023.

[3] H. Brendan McMahan, Eider Moore, Daniel Ramage, Blaise Ag, "Federated Learning of Deep Networks using Model Averaging," 2016.

[4] Zhang, Zhuosheng and Li, Jiarui and Yu, Shucheng and Makaya, Christian, "SECURELearning: Secure Aggregation in Federated Learning With Backdoor Detectability," Institute of Electrical and Electronics Engineers (IEEE), 2023.

[5] Keith Bonawitz and Vladimir Ivanov and Ben Kreuter and Antonio Marcedone and H. Brendan McMahan and Sarvar Patel and Daniel Ramage and Aaron Segal and Karn Seth, "Practical Secure Aggregation for Federated Learning on User-Held Data," 2016.

[6] Jinhyun So and Ramy E. Ali and Basak Guler and Jiantao Jiao and Salman Avestimehr, "Securing Secure Aggregation: Mitigating Multi-Round Privacy Leakage in Federated Learning," 2023.

[7] Shamir, Adi, "How to share a secret," Association for Computing Machinery, 1979.

[8] Hu, Yujiao and Jia, Qingmin and Yao, Yuan and Lee, Yong and Lee, Mengjie and Wang, Chenyi and Zhou, Xiaomao and Xie, Renchao and Yu, F. Richard, "Industrial Internet of Things Intelligence Empowering Smart Manufacturing: A Literature Review," Institute of Electrical and Electronics Engineers (IEEE), 2024.

[9] Swanand Kadhe and Nived Rajaraman and O. Ozan Koyluoglu and Kannan Ramchandran, "FastSecAgg: Scalable Secure Aggregation for Privacy-Preserving Federated Learning," 2020.

[10] H. Brendan McMahan and Eider Moore and Daniel Ramage and Seth Hampson and Blaise Agüera y Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," 2023.

[11] Sanon, Sogo Pierre and Reddy, Rekha and Lipps, Christoph and Schotten, Hans, "Secure Federated Learning: An Evaluation of Homomorphic Encrypted Network Traffic Prediction," 2023.

[12] Rathee, Mayank and Shen, Conghao and Wagh, Sameer and Popa, Raluca Ada, "ELSA: Secure Aggregation for Federated Learning with Malicious Actors," 2023.

[13] Timothy Stevens, Christian Skalka,Christelle Vincent, John Ring, Samuel Clark, Joseph Near, "Efficient Differentially Private Secure Aggregation for Federated Learning via," 2022.

[14] George, Juliana, "LIGHTWEIGHT CRYPTOGRAPHIC PROTOCOLS FOR PRIVACY-PERVING IOT AGGREGATION IN RESOURCE-CONSTRAINED DEVICES," 2023.

[15] Ahmed Imteaj, Urmish Thakker, Shiqiang Wang, M. Hadi Amini, "A Survey on Federated Learning for," IEEE INTERNET OF THINGS JOURNAL,, 2022.

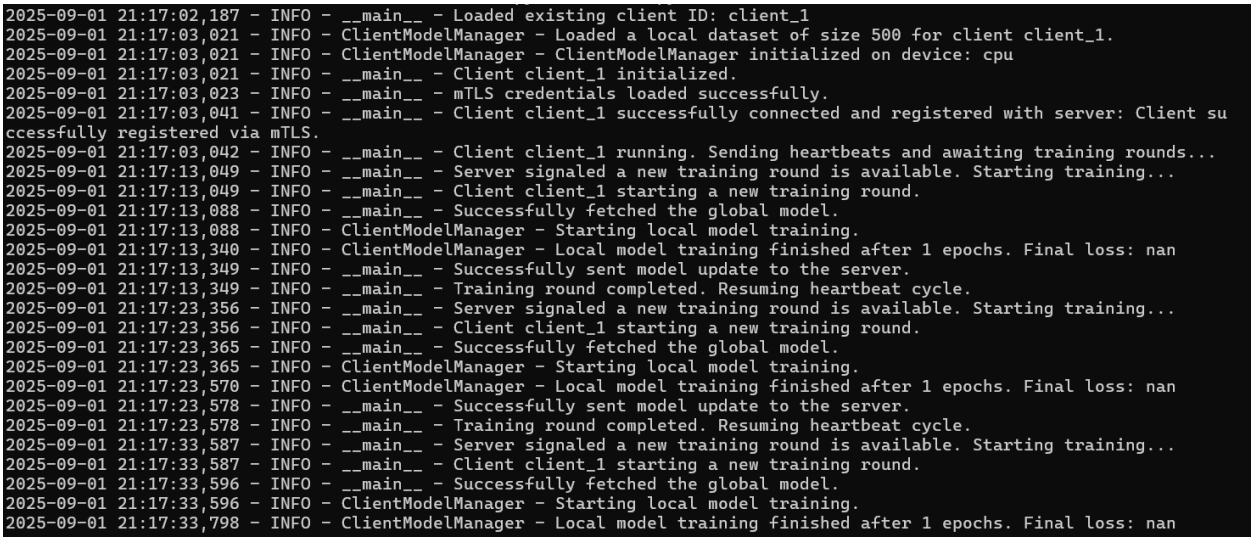
APPENDIX

Module execution

- Server start and initiation of client model manager

```
2025-08-30 01:58:58,938 - INFO - __main__ - Server signaled a new training round is available. Starting training...
2025-08-30 01:58:58,938 - INFO - __main__ - Client client_2 starting a new training round.
2025-08-30 01:58:58,947 - INFO - __main__ - Successfully fetched the global model.
2025-08-30 01:58:58,948 - INFO - ClientModelManager - Starting local model training.
2025-08-30 01:58:59,172 - INFO - ClientModelManager - Local model training finished after 1 epochs. Final loss: 946.7903
2025-08-30 01:58:59,180 - INFO - __main__ - Successfully sent model update to the server.
2025-08-30 01:58:59,180 - INFO - __main__ - Training round completed. Resuming heartbeat cycle.
2025-08-30 01:59:09,196 - INFO - __main__ - Server signaled a new training round is available. Starting training...
2025-08-30 01:59:09,196 - INFO - __main__ - Client client_2 starting a new training round.
2025-08-30 01:59:09,204 - INFO - __main__ - Successfully fetched the global model.
2025-08-30 01:59:09,204 - INFO - ClientModelManager - Starting local model training.
2025-08-30 01:59:09,431 - INFO - ClientModelManager - Local model training finished after 1 epochs. Final loss: 1383.727
3
2025-08-30 01:59:09,440 - INFO - __main__ - Successfully sent model update to the server.
2025-08-30 01:59:09,440 - INFO - __main__ - Training round completed. Resuming heartbeat cycle.
2025-08-30 01:59:19,456 - INFO - __main__ - Server signaled a new training round is available. Starting training...
2025-08-30 01:59:19,457 - INFO - __main__ - Client client_2 starting a new training round.
2025-08-30 01:59:19,465 - INFO - __main__ - Successfully fetched the global model.
2025-08-30 01:59:19,465 - INFO - ClientModelManager - Starting local model training.
2025-08-30 01:59:19,681 - INFO - ClientModelManager - Local model training finished after 1 epochs. Final loss: 884.2057
2025-08-30 01:59:19,689 - INFO - __main__ - Successfully sent model update to the server.
2025-08-30 01:59:19,689 - INFO - __main__ - Training round completed. Resuming heartbeat cycle.
2025-08-30 01:59:29,701 - INFO - __main__ - Server signaled a new training round is available. Starting training...
2025-08-30 01:59:29,701 - INFO - __main__ - Client client_2 starting a new training round.
2025-08-30 01:59:29,710 - INFO - __main__ - Successfully fetched the global model.
2025-08-30 01:59:29,710 - INFO - ClientModelManager - Starting local model training.
2025-08-30 01:59:29,921 - INFO - ClientModelManager - Local model training finished after 1 epochs. Final loss: 992.9814
2025-08-30 01:59:29,929 - INFO - __main__ - Successfully sent model update to the server.
2025-08-30 01:59:29,929 - INFO - __main__ - Training round completed. Resuming heartbeat cycle.
```

- Global model training starts from the terminal (Client initiated)



```
2025-09-01 21:17:02,187 - INFO - __main__ - Loaded existing client ID: client_1
2025-09-01 21:17:03,021 - INFO - ClientModelManager - Loaded a local dataset of size 500 for client client_1.
2025-09-01 21:17:03,021 - INFO - ClientModelManager - ClientModelManager initialized on device: cpu
2025-09-01 21:17:03,021 - INFO - __main__ - Client client_1 initialized.
2025-09-01 21:17:03,023 - INFO - __main__ - mTLS credentials loaded successfully.
2025-09-01 21:17:03,041 - INFO - __main__ - Client client_1 successfully connected and registered with server: Client su
ccessfully registered via mTLS.
2025-09-01 21:17:03,042 - INFO - __main__ - Client client_1 running. Sending heartbeats and awaiting training rounds...
2025-09-01 21:17:13,049 - INFO - __main__ - Server signaled a new training round is available. Starting training...
2025-09-01 21:17:13,049 - INFO - __main__ - Client client_1 starting a new training round.
2025-09-01 21:17:13,088 - INFO - __main__ - Successfully fetched the global model.
2025-09-01 21:17:13,088 - INFO - ClientModelManager - Starting local model training.
2025-09-01 21:17:13,340 - INFO - ClientModelManager - Local model training finished after 1 epochs. Final loss: nan
2025-09-01 21:17:13,349 - INFO - __main__ - Successfully sent model update to the server.
2025-09-01 21:17:13,349 - INFO - __main__ - Training round completed. Resuming heartbeat cycle.
2025-09-01 21:17:23,356 - INFO - __main__ - Server signaled a new training round is available. Starting training...
2025-09-01 21:17:23,356 - INFO - __main__ - Client client_1 starting a new training round.
2025-09-01 21:17:23,365 - INFO - __main__ - Successfully fetched the global model.
2025-09-01 21:17:23,365 - INFO - ClientModelManager - Starting local model training.
2025-09-01 21:17:23,570 - INFO - ClientModelManager - Local model training finished after 1 epochs. Final loss: nan
2025-09-01 21:17:23,578 - INFO - __main__ - Successfully sent model update to the server.
2025-09-01 21:17:23,578 - INFO - __main__ - Training round completed. Resuming heartbeat cycle.
2025-09-01 21:17:33,587 - INFO - __main__ - Server signaled a new training round is available. Starting training...
2025-09-01 21:17:33,587 - INFO - __main__ - Client client_1 starting a new training round.
2025-09-01 21:17:33,596 - INFO - __main__ - Successfully fetched the global model.
2025-09-01 21:17:33,596 - INFO - ClientModelManager - Starting local model training.
2025-09-01 21:17:33,798 - INFO - ClientModelManager - Local model training finished after 1 epochs. Final loss: nan
```