

# A Federated Learning Service Ecosystem for Secure and Flexible Model Sharing in Multi-Cloud Environments

MSc Research Project  
Cloud Computing

Dhyanesh Naik  
Student ID: X22206124

School of Computing  
National College of Ireland

Supervisor: Rashid Mijumbi

**National College of Ireland**  
**MSc Project Submission Sheet**



**School of Computing**

**Student Name:** DHYANESH PRADIP NAIK

**Student ID:** X22206124

**Programme:** : Cloud Computing **Year:** 2024

**Module:** MSc Research Project

**Supervisor:** Rashid Mijumbi

**Submission**

**Due Date:** 12/08/2024

**Project Title:** A Serverless Federated Learning Service Ecosystem for Secure and Flexible Model Sharing in Multi-Cloud Environments

**Word Count:** 8630

**Page Count:** 22

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

**Signature:** Dhyanesh Pradip Naik

**Date:** 12th August 2024

**PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST**

Attach a completed copy of this sheet to each project (including multiple copies)	<input type="checkbox"/>
<b>Attach a Moodle submission receipt of the online project submission</b> , to each project (including multiple copies).	<input type="checkbox"/>
<b>You must ensure that you retain a HARD COPY of the project</b> , both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	<input type="checkbox"/>

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

<b>Office Use Only</b>		
Signature:		
Date:		
Penalty Applied (if applicable):		

# A Serverless Federated Learning Service Ecosystem for Secure and Flexible Model Sharing in Multi-Cloud Environments

Dhyanesh Naik  
X22206124

## Abstract

This research work proposes a relatively new serverless federated learning (FL) model-sharing mechanism for securing flexible multi-cloud environments. The research presents solutions to important issues in decentralized machine learning such as securing fine-grained access control and privacy-preserving aggregation. This proposed framework scales across multiple AWS accounts for secure data and model sharing using a decentralized architecture that resembles the federated learning process. It uses attribute-based encryption (ABE) enabled by encapsulating the attributes into Advanced Encryption Standard (AES) to allow peer-to-peer model sharing across different AWS accounts without the need for central authorities. A masked-ring protocol is implemented for decentralized model aggregation to protect user privacy during training. We have implemented and evaluated our architecture on AWS Lambda to prove that it can be used on real-world serverless platforms. The framework is highly effective and scalable as demonstrated by experiments on multiple cloud accounts by training convolutional neural networks (CNN) on a subset of the MNIST datasets for local training and local model generation. The experimental framework is designed using AWS Lambda functions to distribute the dataset across clouds, begin local model training, encrypt, and use local masks before saving to S3 buckets. The host function can then be triggered to access these local models to perform masked ring aggregation for unmasking and decrypting the aggregated model. This research is part of the work that will advance privacy-encapsulated collaboration learning in multi-cloud, keeping focused on the pragmatic balance between security resourcing vs flexibility alongside performance.

## 1 Introduction

Federated learning is an emerging paradigm for collaborative machine learning using multiple decentralized datasets without sharing raw data. Yet running secure and efficient federated learning across multi-cloud serverless environments is not simple. In this work, we introduce a new serverless federated learning framework capable of securely sharing knowledge from models across multiple cloud environments to address the challenges attributed to decentralized machine learning.

## 1.1 Research Problem and Background

The main research problem addressed by this work is: how to design and deploy a secure, scalable, flexible multi-cloud serverless federated learning system with detailed model sharing without central control points. Most of them work by allowing you to train models without ever sharing your raw data, but they are not built in a way that perform well on the multi-cloud where there is no central entity. However, this constraint prevents the broad applicability of federated learning in a typical cloud-based organization seen throughout today's enterprises (Chadha et al., 2021).

The challenges faced by decentralized servers are that they lack fine-grained access control for access management of training data and models resulting from distributed ML learning on clouds. This limits their capacity to share these deployments among other trusted parties in the cloud, who are not involved in the training, but have the potential and requirement to do model inference for more validation of the generated model (Li et al., 2022). The next issue with decentralized environments is the process of aggregating the model without compromising on privacy (Chen et al., 2024). This had to be considered with much importance before sharing the model across clouds. This research will look to address those issues as well. Also, though serverless architecture presents an attractive option for organizations because of their scalability and low-cost offerings, their implementation is more complex with unique challenges like the difficulty in maintaining a stateless execution environment and managing resource policies (He et al., 2020). Also, it is quite intimidating and challenging to create a framework that can operate well across various cloud platforms while also focusing on security, scalability and cost efficiency (Zhang et al., 2023).

## 1.2 Motivation

This work targets a problem arising from business more and more using multi-cloud strategies, driving the need for privacy-preserving machine learning approaches. Federated learning promises a collaborative way of training these models without exchanging any information. This is great, but when it comes to the real world – deploying this in multi-cloud serverless environment, we have quite a few challenges to talk about. The purpose of this work is to enable businesses to access a pool of data and trained ML models for informed decisions in an ad hoc manner without running the risk that any individual organization's privacy compliances will be breached by developing a secure, efficient, and privacy-preserving federated learning framework designed specifically for multi-cloud settings. Examples of such potential applications are found in healthcare, finance, and smart cities requiring privacy-preserving collaborative research (Su et al., 2021).

## 1.3 Research Question

How can we design and implement a secure, efficient, and flexible multi-cloud serverless federated learning system that supports detailed model sharing without relying on centralized servers, while addressing the challenges of decentralized key management, privacy-preserving aggregation, and multi-cloud adaptability?

## 1.4 Research Objective

In this work, we present a serverless multi-cloud federated learning framework that enables collaborative model training, supports enhanced access control using AES encryption wrapped over user-defined attribute policies (ABE), offers privacy-preserving aggregated models through masked ring aggregation, and is deployable across AWS cloud accounts and usable by practitioners in real-world serverless environments.

## 1.5 Research Contributions

The research contributions are:

- The implementation of a novel multi-cloud secure model-sharing model that utilizes federated learning architecture for decentralized computing.
- An enhanced attribute-based access control using AES by creating unique keys for each or a group combination of attributes that can control access to data or models.
- Additional security is provided by random masking methods to secure the local model updates to guarantee the participating client's privacy.
- With masked ring aggregation, the participating clients can create a ring structure for the model to be aggregated by the host.
- Implementation and evaluation of the proposed framework on AWS Lambda across multi-cloud accounts.
- This approach can be analysed in terms of security, scalability and efficiency

## 1.6 Thesis Organization

The rest of this thesis is organized into chapters as: Chapter 2 gives an exhaustive review on the literature in federated learning in cloud environments and serverless computing, model sharing and ingrained access controls, and decentralized aggregation techniques. This section also covers the research gaps in previous approaches and how the proposed research overcomes it. In chapter 3, research methodology includes the research methods involved in building a secure model sharing techniques using federated learning in a serverless environment. The design specification of the proposed system, including its architecture, components and requirements is explained in Chapter 4. In Chapter 5, implementation phase of the federated learning based secure model sharing is implemented in AWS with all its phases of development is covered. Chapter 6 assesses the results in a way of practical deployment of the research in AWS with multiple clouds. The thesis concludes with Chapter 7 which summarizes the main contributions and highlights the future work for expanded research scope.

# 2 Related Work

## 2.1 Federated Learning in Multi-cloud Environments

(Merseedi and Zeebaree. 2024) presented a comprehensive review of cloud architectures for distributed multi-cloud computing with an emphasis on hybrid and federated cloud environments. It considers the technologies needed to integrate and manage diverse cloud resources from different providers alike. These include private and public cloud architectures together with the key factors to consider such as data residency, security, and workload orchestration. It also focuses on federated clouds architectures that ease workload coordination across multiple clouds by organisations. The authors evaluate current studies and industry practices finding existing research gaps, areas of future exploration or innovation. The last sections however mainly review the existing works without suggesting original approaches or solutions in most cases and does not have performance evaluations or comparisons between different type of multi-clouds strategies that exist today.

A federated learning model by (Singh et al., 2022) is proposed as a workable solution to this problem, using privacy-preserving serverless cloud computing techniques. The authors' method of introducing blockchain-enabled dew servers in each home area network (HAN) eases data storage and training models locally, thus overcoming the shortcomings of current centralized frameworks. Their approach helps to mitigate the effects on training results of outliers with advanced perturbation and normalization procedures. Compared with existing

methods, the proposed model shows reduced computation and communication costs, lower attack probability and better test accuracy results. Experimental results from benchmark datasets attest to the effectiveness of the proposed model.

(Li et al., 2022) addresses the drawbacks of the earlier study in terms of its generalizability by presenting a federated cloud/edge (FCE) framework for distributed cloud environments to execute large-scale AI tasks. These authors use a weighted graph model that is a distributed cloud system, and they apply software-defined networking (SDN) for monitoring and routing purposes. They propose a hierarchical federated aggregation algorithm which is used to build a global model across multiple cloud federations. The paper proves that the FCE framework can lead to 41.3% reduction in total AI processing time and reaches 87% model accuracy in telemedicine experiments. Moreover, the system is tolerant to faults, hence gracefully degrades during node failures. The present research is unquestionably significant with respect to advancing distributed AI processing but overlays an emphasis on performance improvements but does not address privacy concerns pointed out in (Singh et al., 2022) work.

The main goal of the proposed model, Multi-FedLS (Brum et al., 2023) is the implementation of cross-silo federated learning (FL) applications on multi-cloud environments. The authors propose to use multiple cloud providers for reducing both execution time and financial costs of FL applications. The framework is composed of four main components: Pre-Scheduling, Initial Mapping, Fault Tolerance and Dynamic Scheduler. In Pre-Scheduling, they perform experiments for better execution times and communication delays. The Initial Mapping module provides an optimal scheduling map for server and client VMs. Checkpoint mechanisms are implemented by fault tolerance module along with anomaly detection while the Dynamic Scheduler selects new VMs if there are failures. This approach was assessed with a real-world tumor classification application and two FL benchmark datasets. The results state that multi-FedLS can save up to 56.92% costs compared to only using on-demand VMs with just 5.44% increase in execution time.

Similar to (Brum et al., 2023), this paper too proposes a more detailed multi-FedLS framework, primarily focusing on the pre-scheduling module (Brum et al., 2022). The researchers conduct tests aimed at assessing communication delays between various cloud regions as well as expected training and evaluation times for FL clients. The two main cloud providers considered in this research namely, Amazon Web Services (AWS) and Google Cloud Platform (GCP) with different VM types and regions were analyzed in terms of their performance. According to the authors, there are considerable variations in communication times between cloud providers and regions which underlines the importance of careful VM choice and placement across multi-cloud environments. While this paper gives insights into what it takes to implement FL across various cloud suppliers, it does not tackle the problem of data dissimilarity or propose any specific methods for enhancing FL model accuracy under multi-cloud settings.

(Stefanidis et al., 2023) propose MulticloudFL, a FL system for multi-cloud environments that can adapt itself. The authors have made two main contributions: a client selection mechanism based on data size, local loss function values and the use of deep learning non-convex optimization local loss functions. Their approach for selecting clients helps to find faulty data and therefore increases the accuracy of models overall. This system also works with both IID (Identically and Independent Distributed) data scenarios as well as non-IID ones, thus reflecting more realistic multi-cloud deployments. The authors evaluate their ideas through various kinds of experiments where they show that prediction accuracy can be enhanced with respect to application and resource monitoring metrics. So, this work deals explicitly with heterogeneity in handling data, anomaly detection, model improvement among other subjects in the context of multi-cloud FL systems. However, it does not discuss very much about privacy concerns that may arise from such client selection method or scalability issues if applied on large scale multi-cloud setups.

In this section, different techniques of federated learning in multiple cloud environments are discussed. Although research such as (Merseedi and Zeebaree. 2024) provide an extensive literature review, the research tends to not present practical solutions or benchmark results. (Singh et al., 2022) and (Li et al., 2022) propose blockchain and hierarchical aggregation strategies, respectively, but neither of them considers the privacy issue. The works of (Brum et al., 2023) are centered on cost and time factors in multi-cloud FL but do not consider the heterogeneity of data or the enhancement in the model accuracy. (Stefanidis et al., 2023) have proposed an adaptive system but the way they have selected their client can be problematic from privacy point of view.

## 2.2 Secure Model Sharing and Access Control

Initial efforts to promote protection of data privacy in cloud computing revolved around the use of encryption to ensure that the content remained confidential when shared among untrusted cloud service providers. However, conventional methods of encryption did not have capabilities for controlling access at granular level. In order to solve this problem, (Sahai and Waters. 2005) suggested attribute-based encryption (ABE) which could make it possible for data to be encrypted under different access policies depending on attributes. To improve the expressiveness of access policies, (Bethencourt et al., 2007) and (Goyal et al., 2006) developed key-policy and ciphertext-policy versions respectively while retaining the same functionality as ABE. Even though it was possible to control access more precisely through the use of ABE there were efficiency issues when revoking users' rights especially over stored cloud data.

Xu et al., (2018) significantly enhanced revocable-storage attribute-based encryption (RS-ABE) efficiency while maintaining ciphertext delegation essentiality in their constructive works. They came up with a new time encoding mechanism that related to (Sahai and Waters. 2005) identity-based encryption scheme as part of their originality. This helped to reduce computation costs and storage requirements by a logarithmic factor with respect to the overall system lifetime, a big improvement in efficiency for systems that run for long period of times. But still, the method still used pairing operations based on bilinear algebraic structures. Although pairings have been known to provide powerful cryptographic primitives, their use requires heavy computation which becomes problematic in environments limited by resources such as IoT devices or sensors at the edge.

In order to better cope with the particular issues presented by edge computing environments in IoT, (Zhou et al., 2021) have extended attribute-based encryption (ABE) access controls. This study describes diversity and decentralization as the two main features where processing and data sharing take place between different devices at network edges. Therefore, they suggested a new scheme called ciphertext-policy attribute-based proxy re-encryption (CP-ABPRE) that should allow for secure sharing and access of information in such complex networks. They made it possible to detect any potentially harmful actions by any one given node situated at an edge through their methods of embedding unique identifiers into each entity accountable in certain scenarios. Even though it greatly improved safety levels associated with the system's trustworthiness through this feature, there were also drawbacks like additional computation time needed during accountability check-ups. Such trade-offs between increased safety precautions and more demanding computational needs pose typical challenges when designing secure systems meant for limited-resource environments.

In order to tackle the issues of access control in decentralized cloud storage systems, (Gajmal et al., 2024) have put forward a novel idea combining attribute-based encryption (ABE) with blockchain technology. In their design they used the distributed nature and tamper-proof properties of blockchain for key management with implementation on Inter-Planetary File System (IPFS) in data storage across different locations. This integration was primarily intended to bolster security across the board and enhance resilience against central authorities



which could serve as single points of failure or attack. According to initial assessments carried out on their approach, it was found out that there were fewer cases of unauthorized entry than before, thus indicating potential gains in terms of both safety measures and control at different levels. But the incorporation of blockchain also posed new challenges with regard to scalability as blockchains can only handle limited transactions per second and they might encounter higher latency as more nodes join the network. Consequently, these problems around scalability may affect performance as well as responsiveness of an access control system particularly when implemented widely such as having large number users making frequent requests for file access on cloud servers distributed across multiple sites.

Jiang et al., (2020) have done a great job in enhancing cross-domain object detection for self-driving cars with the help of blockchain and distributed learning. They proposed a method which used blockchains to share models between edge nodes while ensuring security. This was a possible because the team built their own domain adaptive YOLO (You Only Look Once) model for multiple edge nodes to be trained in distributed fashion. To enable distributed training and model sharing, they used smart contracts on the blockchain to handle logistics in a tamper-proof decentralized manner. The approach could improve object detection algorithms on a range of domains or scenes. By distributing learning, they can harness computing power from other edges which helps enhance generalization and model performance. The integration with blockchains also brought about transparency as well as auditability into the model sharing process. Nonetheless, while this approach enhanced cross-domain resilience while providing secure frameworks for sharing models, there were still some privacy issues which were not completely taken care of by this strategy with respect to underlying training data and parameters of the model such as information leakage during exchange among others.

The work proposed in (Neyigapula et al., 2023) concentrated more directly towards privacy and security concerns when it comes to AI model exchange over cloud platforms. For this purpose, they presented an exclusive secure key exchange protocol that could enable encrypted machine learning models' sharing between different clouds systems. The proposed method made use of already existing cryptographic techniques i.e., public key cryptography along with Diffie-Hellman key exchange protocol to establish safe communication channels through which encrypted models could be transmitted securely from one cloud system environment onto another cloud system environment without any risk or threat being imposed upon those shared resources throughout their transfer process. This technique guarantees high confidentiality since all shared files are protected against unauthorized access throughout their transmission period across multiple locations within clouds. However, the main drawback of their architecture was that it relied heavily on centralized key management infrastructure, which could compromise overall security and reliability associated with multi-cloud environment used for sharing models.

This section discusses the progression of secure model sharing as well as access control mechanisms. The first attempts at data protection were mainly on encryption, therefore there was no fine control over the access of data. The expressivity was enhanced through ABE but there was a problem of efficiency especially when it came to revoking users. In (Xu et al., 2018) the efficiency was improved but the efficiency came at the cost of computationally expensive pairing operations. (Zhou et al., 2021) enhanced ABE for edge computing to enhance the security but at the same time, it increases the complexity of computation. (Gajmal et al., 2024) proposed an ABE framework in 2024 and integrated it with blockchain to improve security, though at the cost of scalability. (Jiang et al., 2020) implemented blockchain to self-driving car model sharing while enhancing the transparency but not solving the privacy problem completely.

### 2.3 Decentralized Aggregation and Key Management

Bonawitz et al., (2017) introduce a system for securely aggregating high-dimensional data through communication-efficiency and failure-resilience, e.g., user-provided model updates in federated learning. The aim is to allow the server to calculate the sum of data vectors held by users without getting to know each individual contribution made by them. Security is guaranteed under both honest but curious adversaries as well as active ones where also it remains secure even though some users may drop out at any given point arbitrarily. A major breakthrough employed herein involves double-masking technique that safeguards user's information even if masks can be reconstructed by servers. This protocol performs well with large datasets and client pools in terms of runtime or communication overhead; for example, it causes  $1.73\times$  expansion over sending clear text when dealing with input values of 16 bits long, 210 users and 220-dimensional vectors while in case of 214 users along with 224-dimensional vectors there is a  $1.98\times$  expansion recorded over sending clear text. With this approach, the authors propose a practical approach applicable to secure aggregation within real-world federated learning scenarios.

To maintain privacy while ensuring that learning is federated, (Jeon et al., 2021) suggest a decentralized aggregation protocol that uses the Alternating Direction Method of Multiplier (ADMM) algorithm. They address the challenges to privacy in distributed aggregation by creating a protocol that limits communication between participants during each round of aggregation so as to reduce risks of divulging private information. This particular protocol guarantees privacy against an honest-but-curious adversary. The secure aggregation protocol makes it possible to perform efficient federated training with guaranteed privacy. In their work, they conducted experiments on computer vision models and NLP models over benchmark datasets consisting of 9 and 15 distributed sites respectively, achieving comparable learning performance as traditional centralized federated learning methods would do in these cases although having assumed synchronous network model requirements accompanied by computation and communication overheads scaling up linearly with number-of-participants growth rates.

In their study, (Fan et al., 2020) developed a decentralized privacy-preserving data aggregation (DPPDA) method for smart grids based on the blockchain. A block is formed by each mining node selected from among the smart meters in residential areas using a leader election algorithm. In this scheme, the Boneh-Lynn-Shacham short signature and SHA-256 are responsible for ensuring data confidentiality and integrity as far as billing and power regulation are concerned while the Paillier cryptosystem is used by the mining node to aggregate users' power consumption data. It can be seen that this scheme achieves decentralization without relying on any trusted third party or certificate authority thereby safeguarding user privacy. Experiments have shown its computation and communication overheads to be lower than those of existing schemes in some cases while being limited only to one-dimensional power consumption data aggregation methods.

Lin et al., (2017) propose a collaborative key management protocol in ciphertext policy attribute-based encryption (CKM-CP-ABE) for cloud data sharing to overcome the limitations of relying on a trusted third party. Without introducing additional infrastructure, this new approach allows private keys to be generated, issued and stored in a distributed manner. In order to achieve fine-grained and immediate attribute revocations, the authors introduce attribute groups that can be used to build a private key update algorithm. The collaborative mechanism also takes into consideration two aspects, the key escrow problem and key exposure not fully addressed in previous research. It was found that this protocol is secure against semi-trusted entities and efficiency evaluation shows that performed little better performance than representative CP-ABE schemes for cloud-based data sharing on mobile devices, but only applies when sharing data encrypted under CP-ABE policies.

Zhou et al., (2021) developed a system for managing keys in block-chain-based intelligent transport systems (ITS) based on threshold. Shamir’s secret sharing and Chinese remainder theorem (CRT) are used to suggest two different methods of attaining key sharing among stakeholders. If, and only if at minimum ‘t’ number of participants take part, then ‘n’ number of shareholders jointly recover their secret by this plan. This ensures safety as well as fault tolerance in data sharing through ITSs. According to complexity analysis, recovery of secrets can be done more efficiently with CRT than with Shamir’s method. The proposed scheme allows for decentralized key management suitable for use in blockchain technology applied to intelligent transport systems. In future works, one might consider investigating designs where thresholds are computed for access structures that have greater complexity than those described by (t, n)-thresholds.

This section focuses on Decentralized aggregation and Key Management in Federated Learning. Some of the recent solutions that have been proposed include double-masking, ADMM, blockchain for smart grids, collaborative key management for cloud data sharing and threshold-based security for ITS. These methods represent an advancement of the decentralized solutions, but the performance of these methods is a trade-off between the security, efficiency and the flexibility to apply to various situations. However, these approaches have their potential shortcomings including scalability problems, privacy issues, and restricted universality to various situations. Collectively, these studies suggest that more research is still required in decentralized architectures in federated learning.

## 2.4 Summary

The literature analysis exposes a few important research gaps in FL within multi-cloud environments. First, while the training phase of FL in multi-cloud settings has been solved, sharing secure models after training has not seen much exploration. The majority of current methods just concentrate on the training process and ignore phases that follow it where trained models are shared as services. Secondly, there is no fine-grained access control mechanisms for sharing trained models among various clouds or accounts in decentralized manner. Most solutions rely on centralized key management or trusted third parties which do not fit well with decentralized multi-cloud architectures. In addition, sharing trained FL models lacks flexibility and granular access controls, thus limiting their applicability across different domains.

To fill these gaps, several innovations has been proposed by this serverless FL service ecosystem model. The system provides a complete framework for supporting FL starting from customized model training to granular model sharing, while still preserving privacy and fitting into multi-cloud collaboration environment. It uses mask-based ring aggregation scheme for achieving privacy preserving aggregation without destroying distributed nature of training process itself (Hu et al., 2023). Moreover, this approach introduces distributed secret key generation and management scheme integrated with ABE which ensures full decentralization during service. This overcomes limitations associated with existing solutions where keys are managed centrally. Moreover, through attribute-based access control (ABAC), more flexible secure sharing among both participants and non- participants can be allowed, enabling closer integration between them at different levels during their lifecycle. The proposal addresses all these issues thereby providing wider ranging safer alternative for carrying out federated learning within multi-cloud environments.

## 3 Research Methodology

To address the gap described above and improve on this line of work, we propose a mixed-methods approach in designing an effective serverless federated learning framework for secure formative model sharing run over multi-account cloud environments. The methodology

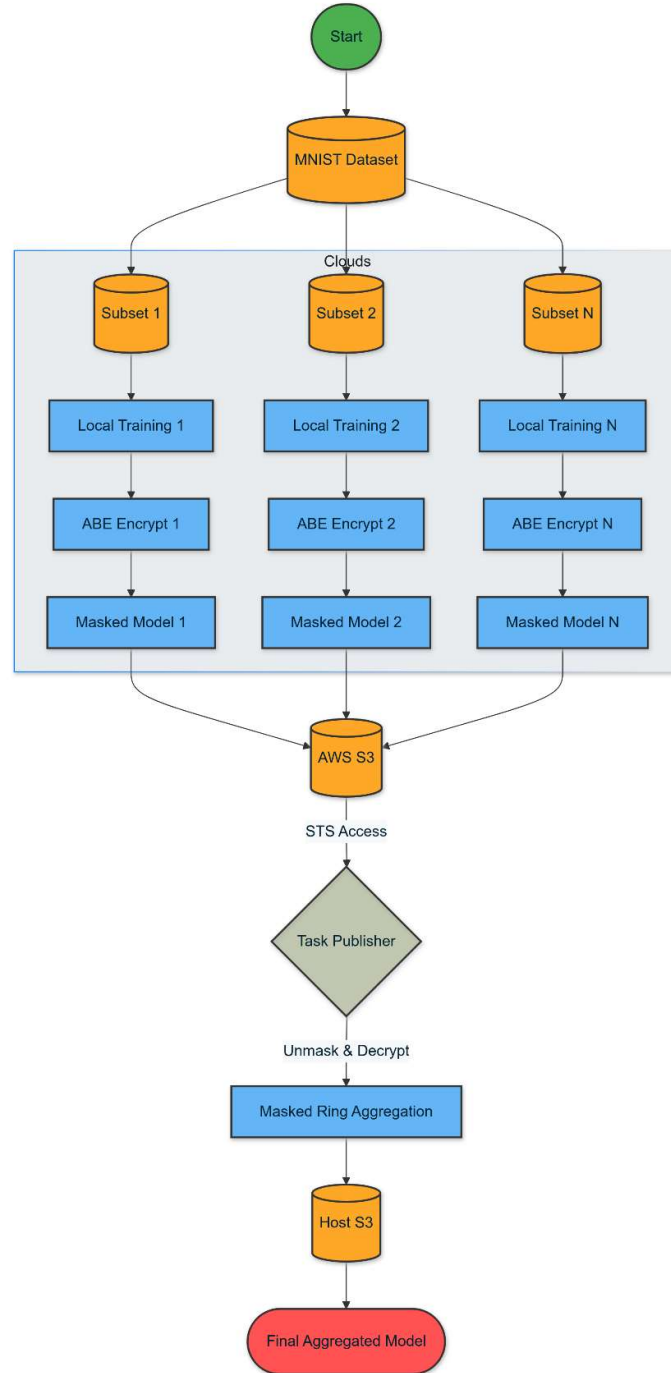
consists of several main stages: architecture design, implementation, experimental setup, data collection, and analysis as in figure 1.

The first phase concentrates on the architecture of a serverless federated learning framework as a solution for decentralized model aggregation, fine-grained access control, and distributed key management in multi-cloud scenarios. A key ingredient in this design is Attribute-Based Access Control (ABAC) combined with AES encryption, allowing secure peer-to-peer model sharing without any central authority. This method presents a higher degree of control and flexibility in terms of access controls compared to its static role-based counterparts, which is why it especially fits well with the dynamic nature that multi-cloud environments operate within.

The MNIST dataset for handwritten digits classification is taken as our benchmark model to be generated for distribution across the cloud. The dataset is preprocessed, normalized, and divided into multiple subsets to simulate distributed data across different cloud environments. This division allows us to realistically model the challenges of federated learning in a multi-party setting.

The cloud infrastructure setup is multiple AWS accounts configuration to mimic a multi-cloud environment. We use AWS S3 for storing our data and models, and as a serverless computing platform with AWS Lambda. We chose this infrastructure to illustrate the viability and scalability of our framework in a real-world cloud setting. Additionally, the setup covers setting up Security Token Service (STS) access for secure cross-account interactions that is very important to keep data privacy and accessibility in a multi-cloud environment. We adopt to train local models within AWS SageMaker, with code used to create all necessary components for training machine learning models like convolutional neural networks (CNN) across distributed MNIST subsets. In this phase, we configure the training parameters to be used for model generation, before commencing with the execution of the entire process on individual data subsets across the multiple accounts in AWS cloud environments. We have applied Attribute Based Access Control with AES encryption as the core component for its fine-grained access policies based on attributes of the accessing entities and protected resources. When combine with AES encryption, it guarantees only permitted parties can access and decrypt the model parameters shared from one place to another, resulting in desirable security on decentralized learning.

We also use random masking techniques in the research to further obscure model parameters before they are shared. Authorized parties then unmask and decrypt the shared information, guaranteeing that during federated learning its integrity is maintained while remaining confidential. Task Publisher – the component responsible for distributing learning tasks and orchestrating the federated model aggregation process across multi-accounts on AWS cloud. This component is important for coordinating the decentralized learning workflow and ensuring participants are in sync with other participants in the closed cloud environment. Our approach heavily relies on the Masked Ring Aggregation protocol provided as a serverless function in our host-node model. The protocol renders safe aggregation of model updates by multiple parties in privacy preserving manner, enabling collaborative learning. Ultimately, we form a final aggregated model that combines the individual learners to collective knowledge learned in the process. This model is stored securely at the same time made available to only authorized parties, showing our full-fledged serverless federated learning framework.



**Figure 1: Proposed Federated Learning based Secure Model Sharing**

It entails a comprehensive methodology, through which we have attempted to tackle the intricate issues surrounding secure and efficient federated learning within multi-cloud environments. Our methodology has specific import for organizations wishing to balance distributed data and computing resources with security grounds. The capability addresses a real pain point and rising trend in enterprise AI development—agile, privacy-preserving collaboration on sensitive data across organizational (and now geographical) boundaries that may otherwise hinder progress, effectively democratizing collaborative innovation outside traditionally siloed fields.

## 4 Design Specifications

We present a serverless federated learning framework that addresses the challenges of secure and efficient model sharing when running multiple clouds. Fundamentally, the architecture leans on AWS Lambda for efficient and economical execution of federated learning tasks at scale, completing an adaptable architectural backbone to enable distributed machine learning across a number of diverse cloud platforms.

### 4.1 Data Preparation

The Framework starts with the Dataset Preparation Module which serves to normalize the distribution and processing of MNIST dataset across different simulated cloud environments. The module performs data preprocessing operations (e.g. normalization and augmentation) so that distributed subsets have been transformed in a consistent way. Data prepared in the framework is sent to AWS S3 buckets of the participating clients as it moves along where corresponding subsets are associated with different cloud accounts in AWS cloud environment. This distributed designed closely relates to the real-world scenario of data silos across organizations in a multi-party collaboration, forming foundations for running federated learning.

### 4.2 Local Model Training Module

This forms the core of the proposed framework and is deployed on AWS Sagemaker for local model training. This uses existing machine learning libraries (ex. TensorFlow or PyTorch) optimized for producing finely tuned models with control over their hyperparameters. This localized module takes a subset of the MNIST dataset as input and chooses some global model parameters, which outputs an updated local model. To facilitate training several local models, we use SageMaker capabilities to automatically search the hyperparameters, that will optimize our successful model during any one-shot iteration. This guarantees that each participant in the federated learning process can individually make large beneficial contributions to a model, even for different local datasets. This module connects to other parts of the framework via SageMaker integration with AWS services. Such as with the model parameters and input data taken from S3 buckets, to saving the trained models back in s3 for use in future stages of federated learning (FL) process.

### 4.3 Attribute-based Access Control (ABAC) with AES

The second most important component which is addressed by our framework is security and access control using Attribute-Based Access Control (ABAC) with AES Module. This component covers the most fine-grained and flexible access control in place of well-known Attribute-Based Encryption (ABE). AES encryption is used to encrypt model parameters and define access policies according to the attributes of requesting entities. It gives you the flexibility to decide who can access, and decrypt shared models which is important in multi-cloud and multi-account clouds.

### 4.4 Random Masking Layer

The Random Masking Module masks the data further to ensure even more privacy protection in form of an additional layer. This module injects a random noise mask into the model parameters before sharing updates of them. It is implemented taking the local update of the model after encryption and returning it in masked form. The model update mask is designed to preserve the statistical properties of the model update, but not allow recovering individual data points from it. This module is used in conjunction with the Unmasking Module, which undoes

the masking after aggregation to provide good performance of final model that respects privacy for all records.

## **4.5 Masked Ring Aggregation Module**

The Masked Ring Aggregation Module is the very core of federated learning. This is a new concept that would allow models to be aggregated across multiple assets without using as central server. This module is implemented in a manner where each function corresponds to one node on the virtual ring and together, they aggregate masked model updates from all clouds involved. The ring also guarantees that an individual participant cannot directly read the plain mirrored updates of other participants. To secure the intermediate result, they were saved in S3 temporarily and access control has been set for both individual buckets. It also handles the unmasking and decryption process, before starting the local model aggregation

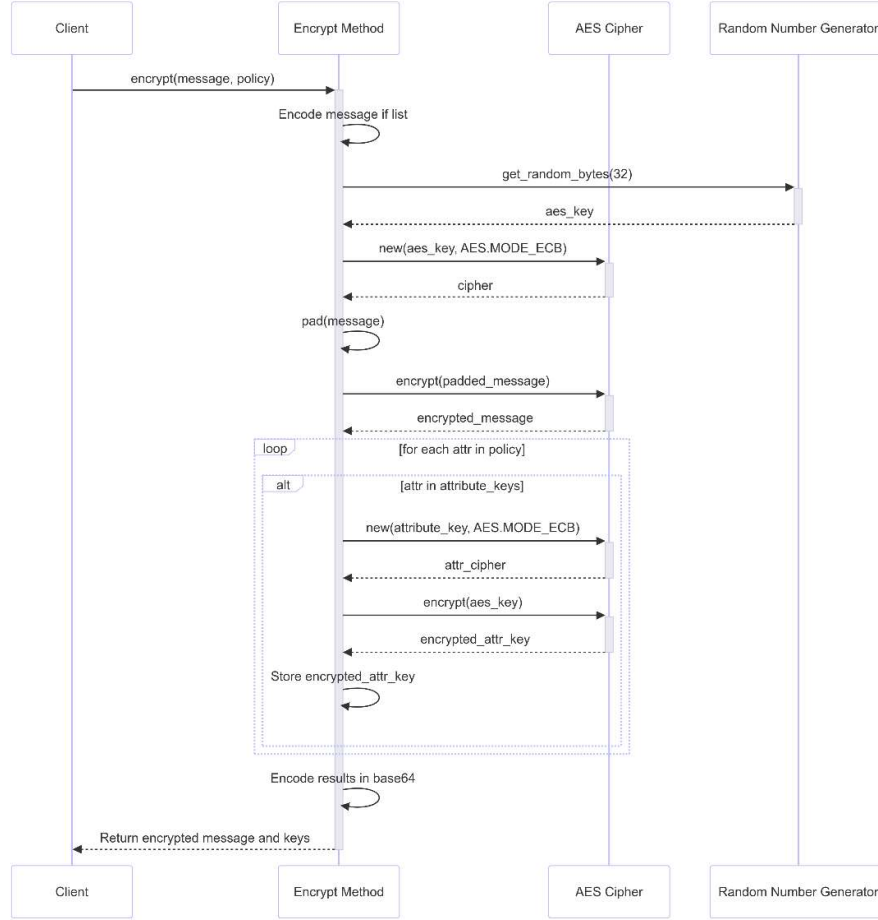
## **4.6 Task Publisher**

The Task Publisher Module coordinates the entire federated learning workflow. It receives global updates on models which it distributes, triggers local training processes and initiates the aggregation protocol. The final aggregation and updating of the global model stored in S3 are also done by this layer along with other layers. Task Publisher is failure-proof and reassuring, using retrying mechanisms to handle errors or network disruptions occurred during the outbound direction of federated learning.

All the modules are interconnected to build a powerful, secure and flexible Serverless Federated Learning Framework. The applications communicate over the event bridge using a mix of direct Lambda invocation, S3 object movement and SageMaker model training. Such design makes having a high scalable and fault-tolerant system where every component should be able to function autonomously while contributing towards the federated learning goal. The loosely coupled, modular design of the framework allows flexibility in extending/customizing its components. Individual components such as the machine learning algorithm (CNN) in Local Model Training Module or aggregation protocols can be modified by researchers and practitioners without impacting overall system architecture. This enables support to a diversified range of federated learning applications more than the classical MNIST classification.

## **4.7 ABAC-AES Encryption – Sequence Diagram**

For this application to federated learning — local model encryption — the ABAC (Attribute-Based Access Control) AES Encryption plays a secure way of safeguarding important parameters. When a local client participant wants to share its model updates, it invokes encryption with the model parameters as the message and an access control policy on who is allowed to consume this encrypted data. This newly generated AES will now be able to encrypt the model's parameters, followed by keys for each attribute in this policy. Then, this way only participants matching these attributes will be able to decrypt and access the new model updates. Encrypting features and not identities leave room for participants to swap out of the system in an ever-changing federated learning standard. This approach improves fine-grained control over what to share for a model, which is important for security and privacy in multi-party machine learning collaborations on different clouds. Figure 2 presents the encryption process as a sequence diagram for detailed understanding.

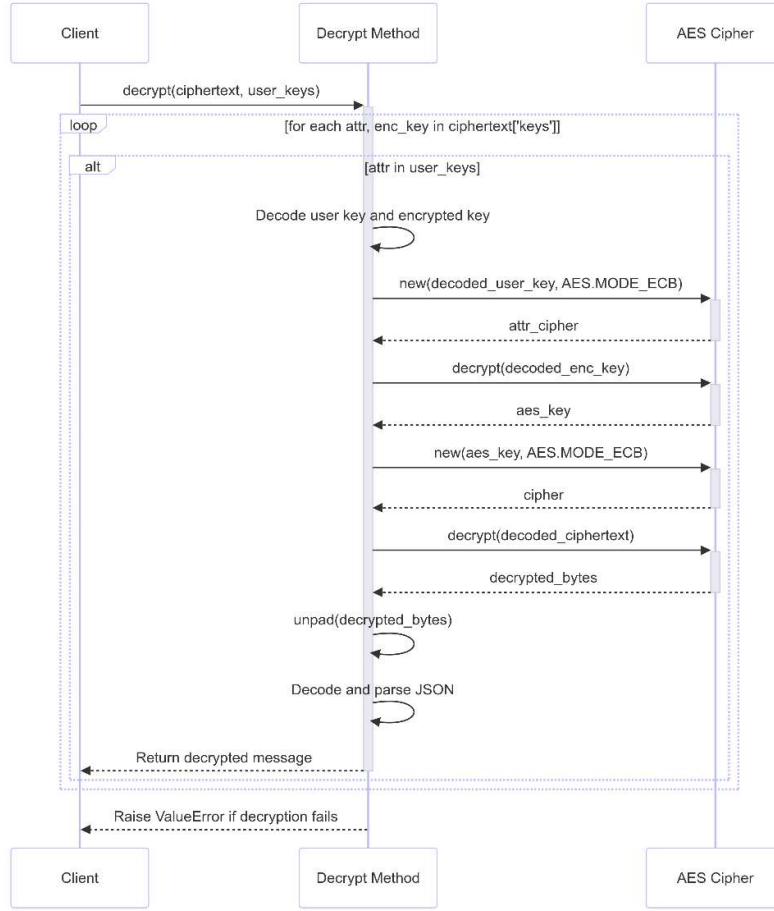


**Figure 2: ABAC-AES Encryption Process**

#### 4.8 ABAC-AES Decryption – Sequence Diagram

When a client gets an encrypted model, it requests the decrypted value of a given tensor in its scope, by giving both ciphertext and their user keys. So, this iterates through every plaintext attribute key in the ciphertext and tries to decrypt it using corresponding user keys of a client. If this is successful, it will derive the primary AES key for encrypting model parameters. This key is then used to decrypt the real model data and unpad it, as well parse from JSON format. As a result of this, encrypted model updates can only be decrypted by the intended owners who have appropriate attribute keys (associated with role/clearance/organization unit). By doing so, the system ensures security and privacy in the multi-party, federated learning environment with different cloud providers by fine-grained access control to share model only among authorized participants  $U$  while preventing adversaries from accessing sensitive information of a shared global model. The sequence modeling diagram is presented in Figure 3 for the decryption process.





**Figure 3: ABAC-AES Decryption Process**

## 5 Implementation

This stage of the research discusses on how the proposed research framework is implemented on AWS covering the configuration, development, and calling of each module discussed in the previous section.

### 5.1 AWS Services

This serverless federated learning framework will for the most part be implemented using an array of AWS services as in Figure 4. We will use Amazon S3 (Simple Storage Service) as the main storage service for this purpose such as datasets, model parameters, masks, and user keys. AWS Lambda-based serverless approach is implemented for different modules of the framework such as dataset preparation and masked ring aggregation module. For this demo, we will use Amazon SageMaker to train the local model on the participating client nodes, encrypt our models with ABAC-AES, and apply masks on them. The configuration of AWS Security Token Services (STS) ensures temporary access to other client AWS accounts and access its resources like S3. The AWS Identity and Access Management (IAM) will be heavily utilized to configure permissions across the accounts.

## 5.2 S3 Bucket Setup

The first step in the implementation process is to create multiple S3 buckets across the client AWS account to capture the essence of working in a multi-cloud environment. Every participating client account will have its own bucket named cloud-2-bucket, cloud-3-bucket and so on for storing the MNIST subset, local model parameters, user keys, and masks associated with each local model. The bucket policies must be configured in such a way that the host account has access write/read permissions to it. The task publisher that holds the masked ring aggregation module must also have access permission to work with cross-account S3 storages before it can read or write from them.

## 5.3 SageMaker Configuration

We'll set up SageMaker for each of the AWS accounts participating in Local Model Training Module. In SageMaker, a Jupyter Notebook environment with proper preconfigured python package libraries to run the model training on the fly can be selected along with the EC2 instances. This includes making SageMaker notebook instances for development and experimentation, defining SageMaker training jobs for the real model training. We will need to create IAM roles that give permission to SageMaker so it can access the specific S3 buckets, read input data and write out the models. The generated local model of the client-node will be saved back to S3 after applying ABAC-AES encryption and masking on it, post-model training.

## 5.4 Cross-Account Access Setup

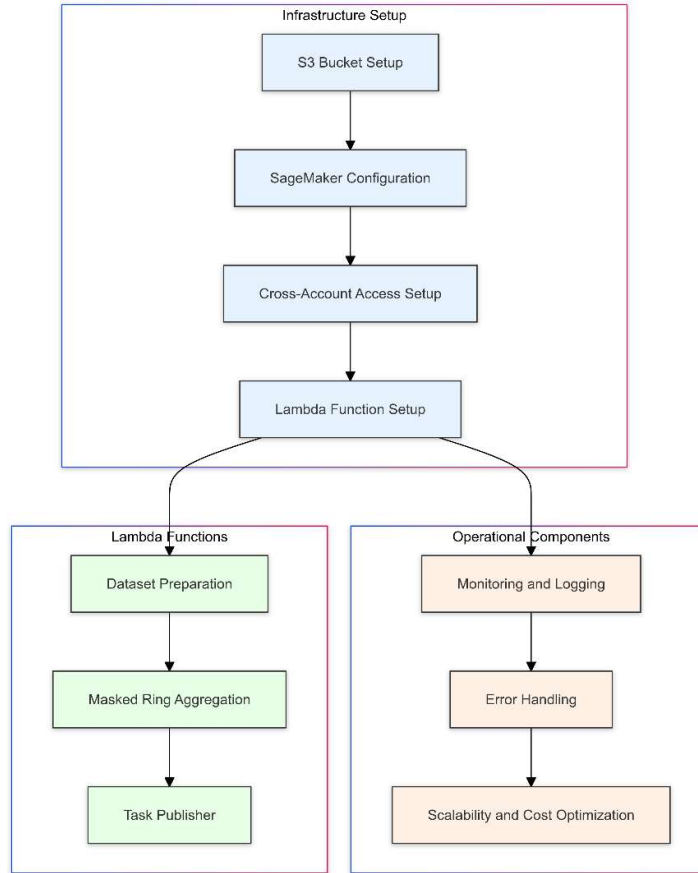
To make this work securely, we will use AWS STS. We have to do this so we can create IAM roles within each account trust with the other participating accounts allowing our service entities access across the multi-account cloud structure. These roles will have policies added that contain the necessary permissions for cross-account S3 access and Lambda invocations. In particular — the Task Publisher module will need to play these roles to orchestrate the federated learning process across accounts.

## 5.5 Monitoring and Logging

Since we want to persistently keep watch of the various services invoked and run in the federated learning process, we'll provision AWS CloudWatch for logging and monitoring. Also configuring CloudWatch Logs to start collecting logs from Lambda functions and SageMaker training jobs for S3 bucket access. CloudWatch Alarms will be set up to notify of any irregularities in the system or failure.

## 5.6 Error Handling

Finally, we want to have the safety net of performing smooth error handling in our Lambda functions and SageMaker training jobs so that everything breaks gracefully enforce more and everywhere. In the context of Masked Ring Aggregation, we will implement a failure recovery mechanism that is able to detect, and bypass failed nodes in our ring so as to allow for partially functional participants yet still have full completion.



**Figure 4: AWS Infrastructure for Secure Serverless Federated Learning**

## 5.7 Scalability and Cost Optimization

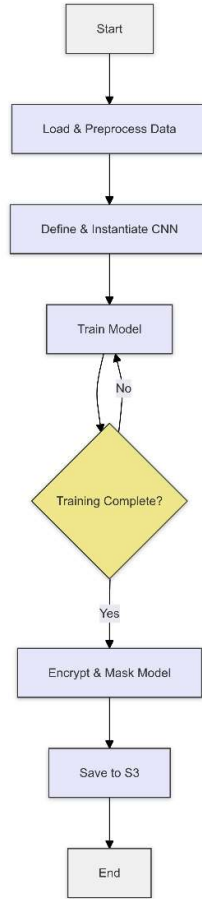
The serverless architecture of our solution using Lambda and SageMaker means it scales without us needing to worry about the infrastructure. But restrictions from Lambda concurrency limits and SageMaker instance types will have to be considered so that we can handle the required scale of federated learning. Depending on how many accounts are participating and the size of models being trained/aggregated, we will also likely need to request limit increases for certain AWS services.

In this example multi-account scenario, we will use AWS Cost Explorer for checking and managing spending in the given accounts. We will leverage managed spot training as much is appropriate (for SageMaker) so that we can cut compute costs. We will further equip S3 buckets to incorporate lifecycle policies which can then transition the objects between different storage classes and expire as per data retention requirements.

## 5.8 CNN Local Model Training on SageMaker

- As shown in Figure 5, it starts with loading a subset of the MNIST dataset from an S3 bucket. A PyTorch Dataloader is created for better batching during training after the raw data has already been preprocessed and normalized.
- This architecture consists of two Convolutional layers, 2 dropouts for regularization and batch normalization followed by a fully connected layer with ReLU activation, and a max pooling `block.

- `train_model`: This is a function to be used for training the model — optimizing parameters using Adam and CrossEntropyLoss as the loss function while iterating through a specified number of epochs.
- The actual code instantiates a CNN model, loads training data from S3, and calls the `train_model` function to fit the network.
- After training, we encrypt the parameters of the models with a custom ABE scheme encapsulated within AES and then mask it for two-level protection.
- The encrypted and masked model parameters are then saved back to S3 for later use or redistribution in a secure, federated learning environment.



**Figure 5: CNN Model Training for Local Model Creation**

## 6 Evaluation

The complete evaluation of the results implementing the aforesaid framework on both the host and the client AWS cloud accounts by setting up the configuration environment and executing the lambda codes for MNIST dataset distribution and masked-ring aggregation. AWS SageMaker implementation is done for local model training, encryption and masking. For convenience's sake, three cloud accounts will be considered, one acting as the host and two others acting as client-1 and client-2 respectively.

## 6.1 Step-1: Lambda function to create MNIST subsets across the cloud accounts

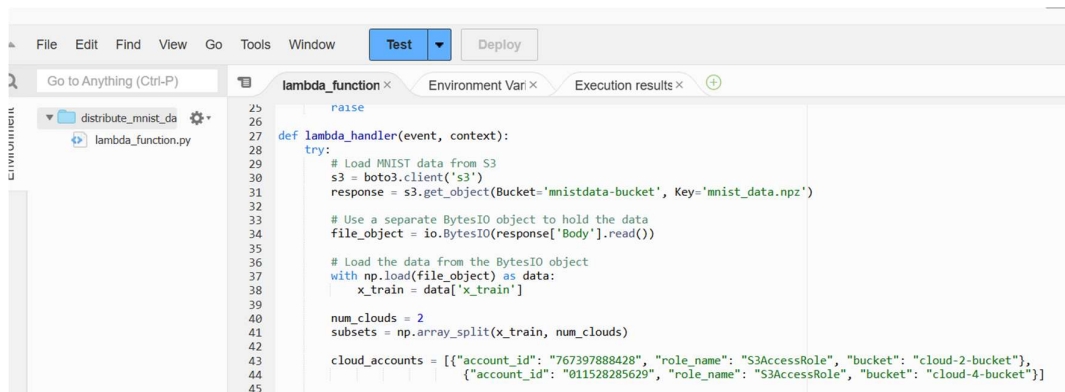


Figure 6: MNIST distribute lambda function with cloud account details

## 6.2 Step-2: Distribution of MNIST subsets

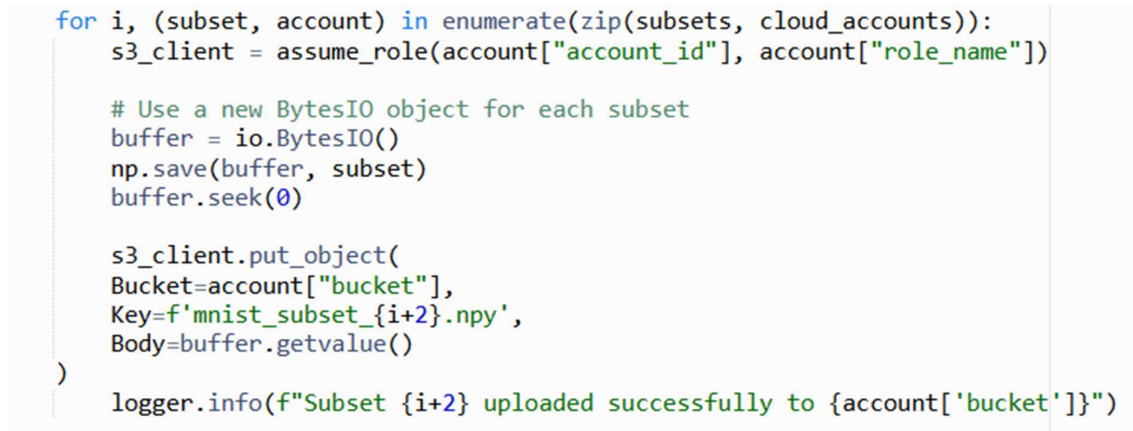


Figure 7: Code Snippets showing the process behind the MNIST subset distribution

## 6.3 Step-3: MNIST subset on Client-2 Account

10.05.37 (UTC+05:30)					
<input type="checkbox"/>	<a href="#">mnist_subset_4.npy</a>	npz	August 11, 2024, 00:35:37 (UTC+05:30)	22.4 MB	Standard

Figure 7: mnist\_subset\_4 is saved in .npy format

## 6.4 Step-4: SageMaker Local Model Training

For model training on SageMaker, create a SageMaker studio domain and use a Jupyter notebook to create the model training file. This implements the CNN model for training using PyTorch package libraries, ABAC-AES model for encryption using pycryptodome, and masking for further layer of security.

```
# CNN Model (unchanged)
class CNN(nn.Module):
    def __init__(self):
        super(CNN, self).__init__()
        self.conv1 = nn.Conv2d(1, 32, 3, 1)
        self.conv2 = nn.Conv2d(32, 64, 3, 1)
        self.dropout1 = nn.Dropout2d(0.25)
        self.dropout2 = nn.Dropout2d(0.5)
        self.fc1 = nn.Linear(9216, 128)
        self.fc2 = nn.Linear(128, 10)
```

Figure 8: CNN model used for local model training on MNIST subset

```
# Generate a random AES key
aes_key = get_random_bytes(32)
cipher = AES.new(aes_key, AES.MODE_ECB)

# Pad the message before encryption
padded_message = pad(message, AES.block_size)
encrypted_message = cipher.encrypt(padded_message)

encrypted_keys = {}
for attr in policy:
    if attr in self.attribute_keys:
        print("Before Encryption", self.attribute_keys[attr])
        attr_cipher = AES.new(self.attribute_keys[attr], AES.MODE_ECB)
        encrypted_keys[attr] = attr_cipher.encrypt(aes_key)
        print("After Encryption", encrypted_keys[attr])

return {
    'message': base64.b64encode(encrypted_message).decode('utf-8'),
    'keys': {k: base64.b64encode(v).decode('utf-8') for k, v in encrypted_keys.items}
```

Figure 9: AES Encryption Block

```
Loaded data shape: (30000, 28, 28)
Processed images shape: (30000, 1, 28, 28)
Labels shape: (30000,)
[2024-08-11 21:58:32.669 pytorch-1-12-cpu-py3-ml-
INFO utils.py:28] RULE_JOB_STOP_SIGNAL_FILENAME:
[2024-08-11 21:58:32.852 pytorch-1-12-cpu-py3-ml-
INFO profiler_config_parser.py:111] Unable to find
fig.json. Profiler is disabled.
Epoch 1/88 completed
Epoch 2/88 completed
Epoch 3/88 completed
Epoch 4/88 completed
Epoch 5/88 completed
Epoch 6/88 completed
Epoch 7/88 completed
```

Figure 10: Model Training Process Output




<input type="checkbox"/>	 <a href="#">masked_model_4.json</a>	json	August 12, 2024, 16:05:59 (UTC+05:30)	34.4 MB	Standard
<input type="checkbox"/>	 <a href="#">masks_4.json</a>	json	August 12, 2024, 16:05:59 (UTC+05:30)	34.4 MB	Standard
<input type="checkbox"/>	 <a href="#">user_keys_4.json</a>	json	August 12, 2024, 16:06:00 (UTC+05:30)	129.0 B	Standard

Figure 11: On client-2 S3 storage, masked model, masks, and user keys are saved

## 6.5 Step-5: Masked-ring Aggregation Lambda on Host Cloud for model aggregation

```
def assume_role(account_id, role_name):
    sts_client = boto3.client('sts')
    assumed_role_object = sts_client.assume_role(
        RoleArn=f"arn:aws:iam::{account_id}:role/{role_name}",
        RoleSessionName="AssumeRoleSession"
    )
    credentials = assumed_role_object['Credentials']
    return boto3.client(
        's3',
        aws_access_key_id=credentials['AccessKeyId'],
        aws_secret_access_key=credentials['SecretAccessKey'],
        aws_session_token=credentials['SessionToken']
    )
```

Figure 12: assume\_role allows to access resources like S3 on other client accounts

```
s3_client = assume_role(account['account_id'], account['role_name'])
masked_model = get_s3_object(s3_client, account['bucket'], account['model_key'])
masks = get_s3_object(s3_client, account['bucket'], account['masks_key'])
user_keys = get_s3_object(s3_client, account['bucket'], account['keys_key'])

logger.info(f"Retrieved data from account {account['account_id']}")

startTime = datetime.datetime.now()

unmasked_model = unmask_model(masked_model, masks)

endTime = datetime.datetime.now()
print("UnMasking Time", account['bucket'])
print(endTime - startTime)

startTime = datetime.datetime.now()

decrypted_model = decrypt_model(unmasked_model, user_keys['user_keys'])
```

Figure 13: Masked model unmasking and decryption followed by aggregation

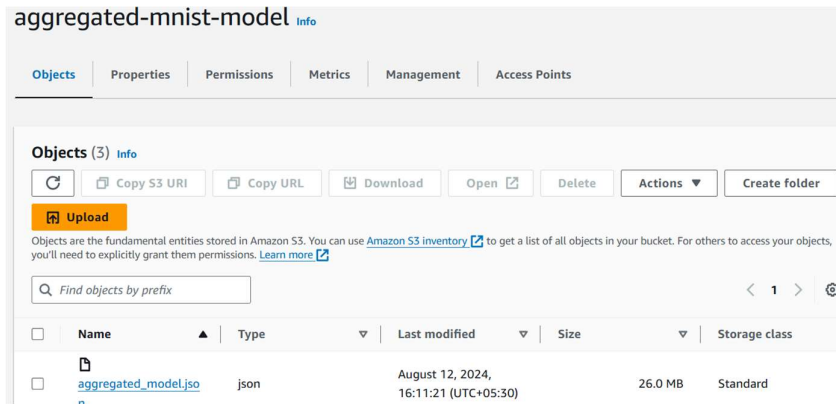


Figure 14: Aggregated MNIST model with the whole MNIST dataset is saved to host cloud in JSON format.

## 6.6 Discussions

These experimental processes presented clearly analyze the implementation of the secure model sharing framework with a demonstration of what happens in the client cloud, and host cloud. How the MNIST subsets distributed over client cloud accounts can be used to train the local model, encrypt, and mask it before saving to S3 storage. Since we consider two clients here, the masked ring aggregator must access the saved models with their masks and user keys, before performing unmasking, decryption, and masked ring aggregation of the host cloud. One important thing to note here is that, before any form of communication, or model/sharing to happen between the participating clients and host, AWS S3 access roles must be defined on the

client cloud for cross-account for host account access, and client S3 account storage must have permissions/trust relationship to allow access to AWS lambda execution roles. The host cloud must have STS policies installed to gain temporary access over the client clouds as well. These configurations to the permissions, roles, and policies on the client and host help to strength the security by preventing authorised access

## 7 Conclusion and Future Work

In this work, a serverless federated learning framework has been proposed that securely and efficiently shares models among multiple cloud accounts. The framework tackles some of the most fundamental challenges in decentralized machine learning: fine-grained access control (solved using attribute-based encryption with AES), privacy-preserving aggregation (e.g., via masked-ring protocols) and scalability across AWS accounts. The experimental results show that the proposed approach is both effective and efficient in mining CNN models training and sharing of MNIST datasets on a distributed cloud account. Future work may investigate the usage of this framework on different machine learning models and datasets, as well as incorporate further security features such as differential privacy. The performance and scalability of the framework should be further studied using different network conditions and larger scale deployments. Moreover, broadening the framework to include a wider range of cloud platforms and serverless computing ecosystems will make it more robust (or practical) in actual use case.

## References

- Bellavista, P., Della Penna, R., Foschini, L. and Scotece, D., 2020, June. Machine learning for predictive diagnostics at the edge: An IIoT practical example. In *ICC 2020-2020 IEEE International Conference On Communications (ICC)* (pp. 1-7). IEEE.
- Benomar, Z., Longo, F., Merlino, G. and Puliafito, A., 2021, December. Deviceless: A serverless approach for the Internet of Things. In *2021 ITU Kaleidoscope: Connecting Physical and Virtual Worlds (ITU K)* (pp. 1-8). IEEE.
- Bhattacharjee, A., Chhokra, A.D., Kang, Z., Sun, H., Gokhale, A. and Karsai, G., 2019, June. Barista: Efficient and scalable serverless serving system for deep learning prediction services. In *2019 IEEE International Conference on Cloud Engineering (IC2E)* (pp. 23-33). IEEE.
- Calabrese, M., Cimmimo, M., Fiume, F., Manfrin, M., Romeo, L., Ceccacci, S., Paolanti, M., Toscano, G., Ciandrini, G., Carrotta, A. and Mengoni, M., 2020. SOPHIA: An event-based IoT and machine learning architecture for predictive maintenance in industry 4.0. *Information*, 11(4), p.202.
- Chadha, M., Jindal, A. and Gerndt, M., 2020, December. Towards federated learning using FaaS fabric. In *Proceedings of the 2020 sixth international workshop on serverless computing* (pp. 49-54).
- Chen, Y., Ning, Y., Slawski, M. and Rangwala, H., 2020, December. Asynchronous online federated learning for edge devices with non-iid data. In *2020 IEEE International Conference on Big Data (Big Data)* (pp. 15-24). IEEE.
- Christou, I.T., Kefalakis, N., Zalonis, A. and Soldatos, J., 2020, May. Predictive and explainable machine learning for industrial internet of things applications. In *2020 16th*



*international conference on distributed computing in sensor systems (DCOSS)* (pp. 213-218). IEEE.

Golec, M., Gill, S.S., Wu, H., Can, T.C., Golec, M., Cetinkaya, O., Cuadrado, F., Parlikad, A.K. and Uhlig, S., 2024. Master: Machine learning-based cold start latency prediction framework in serverless edge computing environments for industry 4.0. *IEEE Journal of Selected Areas in Sensors*.

He, C., Li, S., So, J., Zeng, X., Zhang, M., Wang, H., Wang, X., Vepakomma, P., Singh, A., Qiu, H. and Zhu, X., 2020. FedML: A research library and benchmark for federated machine learning. arXiv preprint arXiv:2007.13518.

Lee, K.C., Villamera, C., Daroya, C.A., Samontanez, P. and Tan, W.M., 2021, November. Improving an IoT-Based Motor Health Predictive Maintenance System Through Edge-Cloud Computing. In *2021 IEEE International Conference on Internet of Things and Intelligence Systems (IoTals)* (pp. 142-148). IEEE.

Li, T., Sahu, A.K., Talwalkar, A. and Smith, V., 2020. Federated learning: Challenges, methods, and future directions. *IEEE signal processing magazine*, 37(3), pp.50-60.

Liu, X., Wen, J., Chen, Z., Li, D., Chen, J., Liu, Y., Wang, H. and Jin, X., 2023. FaaSLight: General application-level cold-start latency optimization for function-as-a-service in serverless computing. *ACM Transactions on Software Engineering and Methodology*, 32(5), pp.1-29.

Magadán, L., Suárez, F.J., Granda, J.C. and García, D.F., 2020. Real-time monitoring of electric motors for detection of operating anomalies and predictive maintenance. In *Science and Technologies for Smart Cities: 5th EAI International Summit, SmartCity360, Braga, Portugal, December 4-6, 2019, Proceedings* (pp. 301-311). Springer International Publishing.

Martinez, M.M. and Pandey, S.R., 2022, March. Predictive function placement for distributed serverless environments. In *2022 25th Conference on Innovation in Clouds, Internet and Networks (ICIN)* (pp. 86-90). IEEE.

McGrath, G. and Brenner, P.R., 2017, June. Serverless computing: Design, implementation, and performance. In *2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW)* (pp. 405-410). IEEE.

Nastic, S., Rausch, T., Scekcic, O., Dustdar, S., Gusev, M., Koteska, B., Kostoska, M., Jakimovski, B., Ristov, S. and Prodan, R., 2017. A serverless real-time data analytics platform for edge computing. *IEEE Internet Computing*, 21(4), pp.64-71.

Nizam, H., Zafar, S., Lv, Z., Wang, F. and Hu, X., 2022. Real-time deep anomaly detection framework for multivariate time-series data in industrial IoT. *IEEE Sensors Journal*, 22(23), pp.22836-22849.

Pelle, I., Czentye, J., Dóka, J., Kern, A., Gerő, B.P. and Sonkoly, B., 2020. Operating latency sensitive applications on public serverless edge cloud platforms. *IEEE Internet of Things Journal*, 8(10), pp.7954-7972.

Poojara, S.R., Dehury, C.K., Jakovits, P. and Srirama, S.N., 2022. Serverless data pipeline approaches for IoT data in fog and cloud computing. *Future Generation Computer Systems*, 130, pp.91-105.

- Sethunath, M. and Peng, Y., 2022. A joint function warm-up and request routing scheme for performing confident serverless computing. *High-Confidence Computing*, 2(3), p.100071.
- Shahrad, M., Fonseca, R., Goiri, I., Chaudhry, G., Batum, P., Cooke, J., Laureano, E., Tresness, C., Russinovich, M. and Bianchini, R., 2020. Serverless in the wild: Characterizing and optimizing the serverless workload at a large cloud provider. In *2020 USENIX annual technical conference (USENIX ATC 20)* (pp. 205-218).
- Su, Z., Wang, Y., Luan, T.H., Zhang, N., Li, F., Chen, T. and Cao, H., 2021. Secure and efficient federated learning for smart grid with edge-cloud collaboration. *IEEE Transactions on Industrial Informatics*, 18(2), pp.1333-1344.
- Bagai, R., 2024. Comparative Analysis of AWS Model Deployment Services. *arXiv preprint arXiv:2405.08175*.
- Wang, B., Ali-Eldin, A. and Shenoy, P., 2021, June. Lass: Running latency sensitive serverless computations at the edge. In *Proceedings of the 30th international symposium on high-performance parallel and distributed computing* (pp. 239-251).
- Wang, B., Ali-Eldin, A. and Shenoy, P., 2021, June. Lass: Running latency sensitive serverless computations at the edge. In *Proceedings of the 30th international symposium on high-performance parallel and distributed computing* (pp. 239-251).
- Wang, I., Liri, E. and Ramakrishnan, K.K., 2020, November. Supporting IoT applications with serverless edge clouds. In *2020 IEEE 9th International Conference on Cloud Networking (CloudNet)* (pp. 1-4). IEEE.
- Wang, J., Zhang, L., Duan, L. and Gao, R.X., 2017. A new paradigm of cloud-based predictive maintenance for intelligent manufacturing. *Journal of Intelligent Manufacturing*, 28, pp.1125-1137.
- Wu, S., Tao, Z., Fan, H., Huang, Z., Zhang, X., Jin, H., Yu, C. and Cao, C., 2022. Container lifecycle-aware scheduling for serverless computing. *Software: Practice and Experience*, 52(2), pp.337-352.
- Zhang, C., Li, S., Xia, J., Wang, W., Yan, F. and Liu, Y., 2020. {BatchCrypt}: Efficient homomorphic encryption for {Cross-Silo} federated learning. In *2020 USENIX annual technical conference (USENIX ATC 20)* (pp. 493-506).