Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

FedQP: Large-Scale Private and Flexible Federated Query Processing

HUSSAIN M. J. ALMOHRI[1], (Senior Member, IEEE), LAYNE T. WATSON[2], (Life Fellow, IEEE)
[1]Department of Computer Science, Kuwait University, Kuwait City, Kuwait (e-mail: almohri@cs.ku.edu.kw)
[2]Departments of Computer Science, Mathematics, and Aerospace and Ocean Engineering, Virginia Tech, Blacksburg, VA, USA (e-mail: ltw@ieee.org)

Corresponding author: Hussain M. J. Almohri (e-mail: almohri@cs.ku.edu.kw).

“This work was supported by Kuwait University under Project RS02/23. ”

ABSTRACT State-of-the-art federated learning coordinates stochastic gradient descent across clients to refine shared model parameters while protecting individual datasets. Current methods require a uniform data model and are vulnerable to privacy attacks such as model inversion. The key challenge is in designing algorithms that efficiently aggregate client input, minimize data exposure, and maximize model adaptability. We propose a new scheme for aggregating local, diverse, and independent data models, addressing data inconsistency and model-inversion attacks. The proposed scheme, federated query processing `FedQP`, enables clients to build local models independently, without coordinating with the server or other clients. The server communicates with selected clients only to predict future values. Our scheme only requires one communication round between each client and the server for model initialization without the need for a training phase.
This article presents the methodology, design, and evaluation of a performance-efficient implementation of `FedQP`. We show that the caching algorithm in `FedQP` reduces redundant client communications during query processing. In addition, we present a privacy analysis that shows that our method outperforms prominent gradient-based approaches to federated learning.
Our experiments show that FedQP consistently achieves higher classification accuracy in non-IID settings and demonstrates stronger resilience to reconstruction attacks compared to gradient-based methods such as federated averaging (`FedAvg`).

INDEX TERMS Federated learning, privacy-preserving systems, secure machine learning, distributed learning

I. INTRODUCTION

Federated learning [1] (FL) enables multiple independent data owners to collaborate and produce a unified machine learning model that represents their datasets while avoiding the disclosure of raw data [2]. State-of-the-art FL models propose a mediator server that coordinates model generation [1]. Untrustworthy users can submit queries to the data model without direct access to the model parameters or the underlying datasets.

Federated learning is useful in many applications where data owners are reluctant to share data due to privacy restrictions. For example, a network intrusion detection system benefits from modeling data collected from various networks that experience distinct events or features. Network administrators (regarded data owners) have conflicting goals; they wish to keep their data private while utilizing a model that represents data from other networks. More data with unique characteristics can produce a more enriched network anomaly detection model that exhibits improved accuracy in post-deployment monitoring [3]. The unified model is expected to increase the prediction accuracy compared to isolated local models with incomplete datasets. However, building a unified model requires accessing raw data points. To resolve this conflict, federated learning aims to minimize dataset exposure while building a unified global model that represents various private and independent datasets.

Federated learning faces three major concerns. First, federated learning models and system architectures must provide reliable and practical model fusion or aggregation methods. The technical challenge lies in designing a solid mathematical method that can combine the outputs of hundreds of participating clients, resulting in a unified model.

Second, the model fusion method must minimize the exposure of data that leads to direct or indirect access to private client data. For example, a model fusion technique is to produce a single global model, which benefits from iterations of stochastic gradient descent (SGD), as shown by McMahan et al. [4]. Unfortunately, gradient-based federated learning (based on parameter averaging) is vulnerable to membership inference attacks [5]. For example, Nasr et al. show that the SGD algorithm, through its error minimization objective, allows each training point to influence the model in a way that leaves a detectable footprint. Thus, a participant's dataset is potentially exposed when submitting the local updates or local model parameters, defeating the purpose of federated learning.

Third, federated learning must be communication efficient with a fast query processing system. System efficiency is a problem when building a unified model and responding to queries. The prominent single-model federated learning systems require many rounds of communication with the client at the time of model creation and infusion, which must be repeated when a subset of client data is updated.

To fix privacy attacks in single-model federated learning, differential privacy [6] (systematically adding noise to local updates) and secure multiple party computation [7] (using a variation of homomorphic encryption [8]) solutions have been proposed. These solutions improve privacy, protect local datasets from exposure, and add significant encryption overhead. BatchCrypt attempts to solve the problem by reducing the frequency of encryption [9]. However, encrypting federated learning updates imposes significant execution overhead, particularly when performed frequently or at fine-grained levels, which can limit the scalability and practicality of secure aggregation schemes.

To address the limitations of traditional federated learning, we present federated query processing (FedQP), an efficient and privacy-preserving framework that improves on the earlier scheme in [10]. FedQP separates the roles of clients into data-owning clients, who build and maintain local models, and query clients, who submit prediction queries without contributing training data. Data-owning clients independently train models using local data and share only statistical summaries (centroids) with a central unification server. This server coordinates prediction tasks by selecting a subset of relevant clients based on centroid proximity to respond to each query. Without accessing raw data, models, or intermediate parameters, the server aggregates these responses to generate a final prediction for the query client. This architecture ensures model agnosticism, strong privacy guarantees, and scalability across diverse client datasets.

Unlike gradient-based federated learning methods that require coordination across differentiable models, FedQP supports fully local training and model-agnostic query fusion. This enables the use of training algorithms that are not based on the gradient descent algorithm, such as decision trees or ensembles, making FedQP applicable to a wider range of systems.

The architecture of `FedQP` improves the previous federated interpolation in two important ways. First, federated interpolation intrinsically works well only with regression tasks. Thus, the method in [10] (referred to as `FedInt`) uses rounding to estimate the classification results. Here, we use a two-stage federated classification to select a subset of the clients predicted to have the best answer to a query. Then, we aggregate the individual client responses as the query's final predicted value. Second, the method in [10] required round-trip communication between the server and every client for each query. `FedInt` had no solution to select the right subset of clients to respond to a query. In addition, it could not reduce redundant communication when processing repeated queries. In contrast, federated query processing eliminates unnecessary communications by selecting the relevant clients when predicting a query. It also adds a query cache, where local client responses are reused for subsequent communications.

Although existing federated learning approaches like `FedAvg` [1] protect raw data by keeping them on client devices, they can be vulnerable to model inversion attacks when sharing local client gradients [11], [12]. An adversary can reconstruct training data by analyzing the gradients exchanged during model updates, as demonstrated by recent work on deep leakage from gradients. `FedQP` fundamentally restructures the federated learning architecture to reduce the possibility of model inversion. Instead of sharing model parameters or gradients, clients only share dataset centroids — statistical summaries that characterize their data distribution without revealing individual data points. Additionally, `FedQP`'s query processing mechanism ensures that even when responding to prediction requests, clients maintain complete control over their models and never expose parameters that could leak training data. This architectural change provides stronger privacy guarantees than traditional federated approaches, while maintaining competitive accuracy through our novel centroid-based aggregation scheme.

Our work provides a practical and efficient federated learning alternative. It avoids traditional encryption techniques while achieving an accuracy comparable to single global models with centralized data access. The key contributions of this paper are as follows.

1) `FedQP`, an encryption-free private model interpolation method that avoids model or data sharing, utilizing query aggregation instead of model aggregation to deliver privacy-preserving predictions (Section III),
2) a scalable system architecture for implementing `FedQP` in a client-server setting, capable of handling large client participation,
3) algorithms for initializing `FedQP` and processing model queries, which include mechanisms for selecting relevant clients to improve query accuracy and efficiency,
4) experimental evaluation showing the high accuracy of `FedQP` using a real-world network intrusion dataset (Section IV),
5) simulations (Section IV-C) demonstrating efficient

query caching significantly reduce communication costs without compromising prediction accuracy, and

6) a comprehensive theoretical (Section III-E1 and empirical (Section IV-D) privacy analysis using a real-world dataset.

Our privacy analysis using the UCI Adult Census dataset [13] validates the theoretical privacy guarantees discussed in Section III-E1. Compared to federated averaging (`FedAvg`) [1], Federated Proximal optimization (`FedProx`) [14], and stochastic controlled average (SCAFFOLD) [15], our centroid-based approach demonstrates stronger empirical privacy preservation. `FedQP` achieves higher reconstruction error (1.371) and information loss (2.818) compared to `FedAvg` (1.150 and 2.509), `FedProx` (1.107 and 2.511), and notably outperforms SCAFFOLD (0.802 and 0.539). Even with access to centroids, as demonstrated by our empirical metrics, the attacker faces significant challenges in reconstructing the original dataset, particularly given the unknown dataset size and the exponential growth of search space with dimensionality.

II. BACKGROUND AND RELATED WORKS

Federated query processing is inspired by Shepard's function interpolation and federated interpolation [10]. To provide the required background, this section presents the definition of original federated learning, focusing on the federated averaging aggregation scheme (Section II-A) and Shepard's interpolation method (Section II-C). At the end of the section, we present a categorized overview of related works (Section II-D).

A. FEDERATED LEARNING

The pioneering work by McMahan et al., Federated Averaging (`FedAvg`) [1], defines federated learning with the objective:

$$\min_{\theta} F(\theta) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(\theta)$$

where

$$F_k(\theta) = \frac{1}{n_k} \sum_{i \in \mathcal{D}_k} L(f_\theta(x_i), y_i)$$

for K clients participating in the learning process. Here, θ represents the parameters of the global model that is being optimized and $F(\theta)$ is the global objective function, representing the weighted sum of the client objectives. n is the total number of data samples in all clients such that $n = \sum_{k=1}^{K} n_k$. Each client k has n_k data samples, and $F_k(\theta)$ is the local objective function of a client k, calculated as the average loss in the local data set of the client $\mathcal{D}_k$. The loss function $L(f_\theta(x_i), y_i)$ measures the discrepancy between the predicted label $f_\theta(x_i)$ by the model f_θ with parameters θ and the actual label y_i for the data point x_i.

`FedAvg` promises to preserve user privacy by ensuring that raw data never leaves the client's device. However, federated learning is susceptible to model inversion attacks [11], [12], [16]–[18]. In such attacks, adversaries can reconstruct the original data from shared model updates. Model inversion exploits the information in the gradients to infer sensitive data characteristics, thus posing a significant privacy risk. For example, deep leakage from gradients attempt to reconstruct the input X_i that the client C_i wants to hide from the kth iteration gradient ∇_k from C_i. The attack is possible by initially assuming a value X_i^* and iteratively adjusting X_i^* until the gradient ∇_k is achieved [17].

To solve some of the privacy problems in federated learning, Bonawitz et al. introduced the use of cryptographic masking to protect individual clients from model inversion by the curious server [16]. However, cryptographic approaches suffer from increased data size, execution penalties, and the possibility of implementation errors. Furthermore, Zhao et al. showed that their attack, LOKI, can bypass some of the state-of-the-art measures [19].

B. HOW DOES INTERPOLATION IMPROVE PRIVACY?

An alternative is to hide the data points and model parameters from the aggregate server while enabling collaborative federated learning. To this end, federated interpolation [10] (`FedInt`) is a horizontal federated learning [20] method that provides an alternative to prominent gradient updates methods (such as `FedAvg`). Figure 3 shows a comparison between `FedAvg` and `FedInt`, where `FedInt` improves the privacy of the system by hiding local model parameters, thus preventing model attacks.

As can be seen in Figure 2, clients do not collaborate in creating a unified model. Clients independently collect data and train local models. They do not have to agree on specific model parameters. The only agreement is on the target value (for example, classifying a network request as benign or denial of service). The model fuser server has no information about its dataset besides a single data centroid. In contrast, as shown in Figure 1, the aggregate server in `FedAvg` has full information on gradual iterations to build the central model, potentially exposing client datasets.

In this article, we take a step forward and improve federated interpolation's accuracy and efficiency, as explained later in Section III. In the next section, we provide the background information on Shepard's model interpolation, which inspired federated interpolation.

C. CHALLENGES WITH INTERPOLATION

Interpolation has a long and rich history in mathematics and continues to be developed, with recent work including the mathematical software SHEPPACK [21] and DELAUNAYSPARSE [22], and in high dimensional data modeling [23]. For many applications interpolation is superior to current machine learning models [23], and still dominates approximation in computational science and engineering.

The central problem in function approximation is to take a given function $f(x)$ and construct a simple function $s(x)$ that approximates $f(x)$ [24]. Polynomial interpolation represents

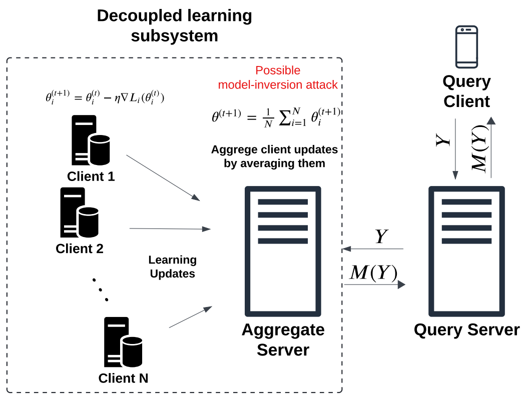

Decoupled learning subsystem
Possible model-inversion attack
$\theta_i^{(t+1)} = \theta_i^{(t)} - \eta \nabla L_i(\theta_i^{(t)})$
$\theta^{(t+1)} = \frac{1}{N}\sum_{i=1}^{N} \theta_i^{(t+1)}$
Aggrege client updates by averaging them
Client 1
Client 2
Learning Updates
Client N
Aggregate Server
Y
M(Y)
Query Client
Query Server

FIGURE 1: Federated averaging process.

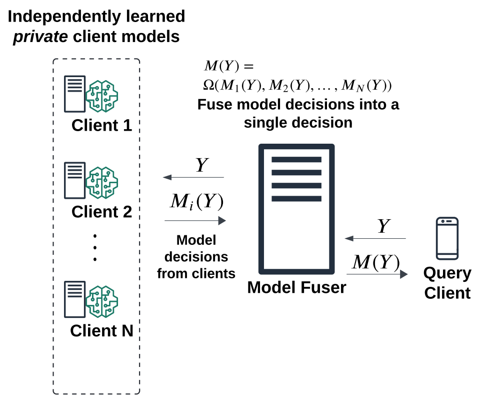

Independently learned *private* client models
$M(Y) = \Omega(M_1(Y), M_2(Y), \ldots, M_N(Y))$
Fuse model decisions into a single decision
Client 1
Client 2
Client N
Y
$M_i(Y)$
Model decisions from clients
Model Fuser
Y
M(Y)
Query Client

FIGURE 2: Federated interpolation process.

FIGURE 3: Federated averaging (`FedAvg`) compared with federated interpolation (`FedInt`) and query processing. In `FedAvg`, the learning process is coordinated by the central server, potentially exposing private information about datasets. In `FedInt`, the learning is completely hidden from the server. During query processing, a subset of clients must be online to provide their answers. The server aggregates these answers using an Ω function of the local responses.

the first attempt to solve the approximation problem. Interpolation differs from regression in that regression is not required to reproduce the given data points, which interpolation does.

Shepard [21], [25] proposed to create a surface based on a weighted average of values at data points. Shepard's method is particularly useful for scattered data points in high dimensions, which can be used in machine learning problems. This particular property of Shepard's algorithm inspired the federated interpolation scheme presented in [10]. This is because individual clients are only motivated to collaborate when each client has a distinct and significant dataset that can influence the direction of prediction taken by a unified global model.

There are three major challenges in utilizing Shepard's interpolation for federated learning. First, any interpolation cannot be used directly in federated learning, as the produced function $f(x)$ would reveal the raw data points, which defeats the purpose of federated learning. To overcome this issue, in an earlier version of this work, we proposed to modify Shepard's method to interpolate client models instead [10]. The scheme presented in [10] successfully applies interpolation to federated learning.

The second challenge is that interpolation methods can only be used with regression tasks. To be used with a classification problem, the results must be rounded to the nearest integers. As labels must be numeric values (chosen arbitrarily) with most machine learning algorithms, the algorithm in [10] may not work well with multi-label classification problems. We extend the methodology in [10] to overcome the classification limitations by introducing majority voting when forecasting individual model queries.

Third, federated interpolation requires a round of communication for every model query. To reduce the communication needed, we introduce a client selection mechanism in which the server only consults a subset of the available models. We also introduce a new model caching mechanism. In this case, the predictions for a previously seen model query are recorded in a cache that can be reused without communication overhead.

D. RELATED FEDERATED LEARNING METHODS

Since federated learning is a collaborative learning approach, two fundamental problems arise. First, a server has to delegate the task of aggregating client input and the production of a model. Second, the federated learning system must ensure maximum privacy for client datasets.

An important approach is to use cryptographic schemes such as multiparty computation or fully homomorphic encryption to share model parameters (client input) with the aggregating server. The server averages encrypted model parameters and produces a common model representing the client's datasets. Some previous work used the central server to produce a unified global model, only delegating the iterations of the gradient descent algorithm to the participating clients(e.g., [26], [27]).

Others elect only to perform model aggregation (e.g., [10], [28]), leaving the learning task to the participating clients. PerFedRec++ provides an interesting approach in which clients pre-train their models individually, addressing the problem of non-Independent and Identically Distributed (non-IID) datasets [29]. Federated query processing benefits from a similar approach. However, federated query processing provides full client pre-processing. As a result, clients do not have to have complete agreement on their datasets while isolating their data points from the federated learning server.

FedProto, by Tan et al. [30], is a federated interpolation approach similar to the work in [10]. FedProto uses prototype-based learning, operates under the assumption of cross-client federated learning, and applies to image classification problems. The prototypes of local client datasets are averaged on the server to produce a global model. In contrast, we focus on tabular data classification problems that have received considerably less attention from the research community. In addition, our proposed scheme does not use simple averaging. Instead, each client is asked to produce a local model value for a given query and a norm distance of the query from the centroid of the local dataset. Clients communicate with the server per submitted query and do not share any local dataset or model information.

Mansour et al. developed three approaches to federated learning, including the idea of *model interpolation* [28] implicit in the earlier work of Shepard. With model interpolation, as proposed in [28], the unified global model is formed by fitting the entire available dataset to find the best coefficients (weights) of the terms in the interpolation. In general, the proposed method requires optimization of the unified model using client data, violating data privacy. Federated query processing (Section III) goes beyond simple model interpolation and eliminates all client participation in building the unified global model, except for providing the hyperparameters of the local models and centroid values representing the data collected by participating clients.

Yurochkin et al. introduce a new beta Bernoulli process through which a *meta-model* of optimized local models is produced [31]. Their approach aims to learn a shared set of parameters through the participating local datasets. Similarly to our federated interpolation, the model in [31] has the advantage of requiring a low bandwidth. In contrast, federated query processing is simpler and faster to compute and makes no particular assumption on the parameters of local models. The advantage of using interpolation to aggregate local models is that federated query processing is model agnostic and does not require wholly disjoint or independent datasets.

Similarly to the work by Yurochkin et al. [31], Claici et al. proposed to minimize the Kullback-Leibler (KL) divergence for fusing multiple heterogeneous local models, to recover a mean-field approximation to the posterior distribution [32]. Their proposed scheme results in averaging the parameters of models produced by participating clients. The work in [32] moves a step towards our goals. However, federated query processing avoids minimization on the server side. Our approach approximates a mixture of local models based on the values provided by a specific model query.

Robustness is a key challenge in federated learning. Recent studies have focused on improving robustness through the randomization of aggregation functions and adaptive runtime strategies. Nabavirazavi et al. [33] examine the impact of stochastic switching between robust aggregation rules such as Krum and the trimmed mean. Their results suggest that randomizing aggregation complicates adversarial attacks and can enhance robustness, albeit with convergence predictability trade-offs. Although our FedQP architecture does not use parameter aggregation, their findings underscore the importance of using uncertainty to defend against model poisoning.

Alsobeh and Shatnawi [34] propose a security framework for IoT systems, integrating model checking, self-adaptation, and threat detection using BIP components. Although IoT-focused, it helps integrate run-time validation in federated systems. Our query processing architecture could evolve by adding dynamic trust assessment or adaptive privacy budgeting during query fusion. Recent AI-enhanced runtime monitoring offers promising avenues for integrating intelligent observability into federated learning systems. AlSobeh et al. [35] introduce an adaptive monitoring framework that uses statistical model verification and execution trace analysis to detect inconsistencies and refactor the code.

III. FEDERATED QUERY PROCESSING

Federated query processing (`FedQP`) is a scheme to process queries to federated and independent models, supporting regression and classification tasks. In this section, we give an overview of federated interpolation [10] (Section III-A), and provide the construction of the proposed federated query processing (Section III-B) as a generalization of federated interpolation.

A. FEDERATED INTERPOLATION

The core concept of federated interpolation `FedInt` is to hide the data and model parameters of the clients who collaborate to build a joint machine learning model. `FedInt` repurposes Shepard's interpolation algorithm [21], [25], [36] by creating a unified model that *interpolates* local client models. In `FedInt`, clients can train their models using different parameters or algorithms. The clients only agree on the target value and the features of the dataset. The scheme aims to address the privacy exposure from state-of-the-art federated averaging `FedAvg` (Section II-A) due to model inversion attacks (as discussed in Section II-A).

Formally, consider data points as $\big(x, f(x)\big)$ where $x \in \mathbb{R}^p$ is a real p-dimensional vector representing the input features, and the function $f(x) \in \mathbb{R}^m$ is a real m-dimensional vector representing the output or label associated with x. The ith client has a dataset $\big\{\big(x_{ij}, f(x_{ij})\big)\big\}_{j \in I_i}$, where x_{ij} represents the jth data point of client i and forms a point cloud around the centroid:

$$\bar{x}_i = \frac{1}{|I_i|} \sum_{j \in I_i} x_{ij}. \qquad (1)$$

Here, $\mathbb{R}^p$ and $\mathbb{R}^m$ denote real vector spaces of dimensions p and m, respectively, where $\mathbb{R}^p$ represents the space of input features, and $\mathbb{R}^m$ represents the space of function outputs or labels. I_i is the index set for data points belonging to client i, and $|I_i|$ represents the number of data points for client i. These point clouds can overlap (meaning their convex hulls intersect), but ideally, the centroids are well-separated.

Each client i trains a local model $M_i(x) \approx f(x)$ built from its dataset using common learning algorithms. The problem is to estimate $f(Y)$ for a given query Y using predictions from client models M_i. The solution involves interpolating the predictions of the local client models M_i for Y. The response of a model M_i is weighted based on the distance between the given query Y and the client's centroid, as described below.

Assume that the data $f(x)$ is noisy, so that a trivial solution—finding the x_{ij} closest to Y (where each client i returns the minimum distance $\|x_{ik} - Y\| = \min_j \|x_{ij} - Y\|$ and the corresponding value $f(x_{ik})$, or an average in case of a tie for minimum distance) and using $f(Y) \approx f(x_{ij})$ for this closest x_{ij}—is inadequate. Given a query $Y \in \mathbb{R}^p$, define

$$w_i = \frac{1}{\|Y - \bar{x}_i\|_s}, \qquad (2)$$

where $\|\cdot\|_s$ denotes the s-norm for some choice of s (e.g., $s = 1, 2, 3, \infty$), and approximate the response at Y by

$$f(Y) \approx \sum_i \left(\frac{w_i}{\sum_j w_j} \right) M_i(Y). \qquad (3)$$

Here, w_i is a weight that is inversely proportional to the s-norm distance between the query Y and the client i's centroid $\bar{x}_i$. The symbol $\|\cdot\|_s$ represents the s-norm, which is a distance metric depending on the value of s. The centroid $\bar{x}_i$ represents the average location of client i's data points and is calculated to reflect the central tendency of its dataset. For $s = 2$, the norm represents the Euclidean distance. The choice of $s > 2$ places more emphasis on larger components of $Y - \bar{x}_i$, and modifying w_i to w_i^k for an integer $k > 1$ further increases the weighting effect for centroids $\bar{x}_i$ that are closer to Y. A common variant of this approach excludes centroids from the sum if $\|Y - \bar{x}_i\|_s$ is larger than a certain threshold, though this may result in a zero response for certain queries Y.

Federated interpolation balances the participation of models with centroids closer to the provided argument Y. Models with farther centroids receive lower weights and thus have less opportunity to influence the value $f(Y)$. This model can produce improved output from one that is generated by simply averaging over individual client models (as in [1]).

B. COMPONENTS OF FEDERATED QUERY PROCESSING

A client i is an independent entity that owns a dataset and operates a one-to-one service that communicates only with the `FedQP` server S. Each client dataset i can have distinct features. However, all clients have to agree on the same target variable (such as benign or malicious network activity).

1) Client Model Development

Federated query processing requires a bootstrapping phase in which at least one client must produce a local model using its local dataset. Each client i independently trains a model M_i using its local dataset $\mathcal{D}_i$. The model M_i is a function that maps the input data x to $M_i(x)$, predicting the target value given the observation in x.

2) Centroid Computation

Next, the server S requests the computation of a *centroid* from each client that has completed the training of the local model. Each client must provide at least a single centroid used to compare client datasets *without* requiring exposure of the entire or even parts of the client datasets. Unlike in `FedInt`, clients can submit more than one centroid. Thus, each client computes k_i centroids $\{\bar{x}_{i1}, \bar{x}_{i2}, \ldots, \bar{x}_{ik_i}\}$ that summarize different regions of its data points. Each centroid $\bar{x}_{ij}$ is computed as described in Section 3.1. Once the centroids are computed, each client sends its vector of centroids $\{\bar{x}_{i1}, \bar{x}_{i2}, \ldots, \bar{x}_{ik_i}\}$ to the central server. The server maintains a list of all the centroids of all the clients. In Section III-D, we describe Algorithm 3 for an optional partitioning of the client dataset to produce more than one centroid when the client chooses to do so. However, a single client centroid suffices to produce query responses in Algorithm 1.

The bootstrapping phase of `FedQP` ends with the communications of the centroids with the server S. The bootstrapping phase requires only two communication rounds (per client), one for initializing the bootstrapping and the other for receiving the required values from each client.

3) Query Processing

The server S allows *query clients* to submit queries to predict their target values. A client can submit Y and receive a prediction according to responses from local models. When a new query Y is submitted to the server, the server S computes the Euclidean distance between Y and each centroid $\bar{x}_{ij}$:

$$d(Y, \bar{x}_{ij}) = \|Y - \bar{x}_{ij}\|_2 = \sqrt{\sum_{l=1}^{d} (Y_l - \bar{x}_{ij})^2}.$$

4) Relevant Client Selection

The reason for computing the centroid distances with the query Y is to select the most relevant clients to produce their predictions for Y. In `FedInt`, all clients must participate in responding to a single query. In contrast, `FedQP` avoids unnecessary communication with less relevant clients and only selects a subset of clients to respond. It also prevents polluting the final prediction with values from datasets remote to the query Y, thus improving accuracy. The server selects a subset $\mathcal{S}$ of the clients with the closest centroids to Y. The subset $\mathcal{S}$ is determined by:

$$\mathcal{S} = \{(i, j) \mid \bar{x}_{ij} \text{ is among the } k \text{ closest centroids to } Y\}.$$

5) Fusing Predictions

The server communicates the query Y to the selected clients in $\mathcal{S}$, and receives their predictions $M_i(Y)$. After receiving the responses from the relevant clients, in the final step, the server S must produce a single prediction for Y and return it to the querying client. The server fuses these predictions using a function Ω. The server can define Ω in several ways. These include a weighted average of the predictions

$$\Omega(\{M_i(Y)\}_{i\in\mathcal{S}}) = \sum_{i\in\mathcal{S}} \frac{w_i}{\sum_{i\in\mathcal{S}} w_i} M_i(Y),$$

where w_i are the weights assigned to each client's prediction. The weighted average function in FedQP is suitable for regression tasks. However, for classification tasks, the statistical mode and maximum can be used (results in Section IV). A statistical function f of the returned values that could be defined as a majority vote (statistical mode), the maximum of the predictions, or the median:

$$\Omega(\{M_i(Y)\}_{i\in\mathcal{S}}) = f(\{M_i(Y)\}_{i\in\mathcal{S}}).$$

FedInt has poor performance for classification tasks. This is because interpolation works intrinsically well with regression tasks. The simple way to round out the regression results for classification fails because of the discrepancy with the encoding of the labels. Thus, the function Ω in FedQP resolves the issue by providing a mathematically sound fusion of values that works for classification.

Algorithm 1 summarizes the steps for processing a query in FedQP. The algorithm is executed asynchronously for each request from a client. Implementation details include query pipelining for efficient query processing. However, the system implementation details for handling queries are beyond the scope of this work. Furthermore, line 10 in Algorithm 1 requires query communications with selected clients $\mathcal{S}$.

Note that the choice of aggregation function can affect the robustness of the unified model. As analyzed by Pillutla et al. [37] and Taheri et al. [38], simply replacing the arithmetic mean with the geometric mean or using probabilistic weighting can help reduce vulnerability to malicious input from clients. Later in Section IV-I, we provide empirical evidence that choosing functions such as statistical mode can reduce the effect of malicious input on FedQP.

6) Communication Efficiency

FedQP uses two methods to reduce communication overhead between the server and the client models. One is the selection of clients with Algorithm 1. Even with client selection, every query requires at least one client communication before FedQP can produce the response. The second method is to avoid unnecessary communication repetition using a query cache. Algorithm 2 reuse answers to previous queries when possible. The cache stores a table of client query values with their corresponding computed response as the output of Algorithm 1.

Algorithm 2 starts with a cache threshold ϵ, which controls when the algorithm accepts a cache entry for a specific query Y. If an entry with a distance below ϵ was found, the algorithm would use the cache entry as a response to Y. Euclidean distance determines the difference between Y and the values in the cache.

Algorithm 1 Query Processing in FedQP. The aggregation function Ω can be instantiated as a task-specific fusion operator, such as weighted averaging (regression), majority voting (classification), or trimmed mean (robust inference).

```
Require: Query $Y$, centroids $\{\bar{x}_{ij}\}$ from all clients, number
    of selected clients $k$, blending function $\Omega$
 1: $\mathcal{S} \leftarrow \emptyset$
 2: for each client $i$ and its centroids $\{\bar{x}_{ij}\}$ do
 3:     for each centroid $\bar{x}_{ij}$ do
 4:         Compute distance $d(Y, \bar{x}_{ij}) = \|Y - \bar{x}_{ij}\|_2$
 5:         Insert $(i, j, d(Y, \bar{x}_{ij}))$ into $\mathcal{S}$
 6:     end for
 7: end for
 8: Sort $\mathcal{S}$ by distance $d(Y, \bar{x}_{ij})$
 9: $\mathcal{S}_k \leftarrow$Select the top $k$ clients and centroids
10: Communicate query $Y$ to each client in $\mathcal{S}_k$.
11: for each selected client $\bar{x}_i$ in $\bar{x}_s$ do
12:     Receive prediction $M_i(Y)$ from client $\bar{x}_i$.
13: end for
14: $\hat{Y} \leftarrow \Omega(\{M_i(Y)\}_{i\in\mathcal{S}_k})$
15: return $\hat{Y}$
```

Algorithm 2 Caching Scheme for Federated Query Processing

```
Require: Query $Y$, cache $\mathcal{C}$, cache threshold $\epsilon$
Ensure: Cached or computed prediction $\hat{Y}$
 1: Normalize query norm$(Y) \leftarrow Y/\|Y\|$.
 2: $\mathcal{C} \leftarrow$ Retrieve cache values.
 3: for each cached query $Q$ in $\mathcal{C}$ do
 4:     Normalize cached query norm$(Q) \leftarrow Q/\|Q\|$
 5:     Compute distance $d \leftarrow \|$norm$(Y) -$ norm$(Q)\|_2$
 6:     if $d < \epsilon$ then
 7:         Return cached result $\mathcal{C}(Q)$
 8:     end if
 9: end for
10: Process $Y$ using Algorithm 1.
11: Store result in cache $\mathcal{C}(Y) \leftarrow \hat{Y}$
12: return $\hat{Y}$
```

C. ANALYSIS OF THE CACHING MECHANISM

Caching queries reduces client communication and is crucial for frequent and repeated queries, but it burdens the query server, necessitating complexity analysis.

The query cache has two operations. First, store$(Y, M(Y))$ receives an unseen query and its classification response and stores it in the cache database. Second, lookup(Y), which receives a query and searches the cache database for a cached response. The store operation requires constant time to complete as it will only append the parameters $Y, M(Y)$ to the cache database. lookup in general has the complexity of search in terms of the maximum number of cached queries n, which takes $n \log_n$ to complete.

System performance may be affected by large cache sizes n. Although a large cache reduces communication overhead, it increases search time. Cache optimization can be improved with multiple levels or parallel databases based on query frequency.

`FedQP` can employ several caching strategies, depending on the system's requirement. When `FedQP` is used in a low-throughput environment, a single cache with a maximum number of cache entries can provide improved communication performance while maintaining low overhead for the aggregation server. The maximum number of cache entries directly affects the communication performance. When the system aims to reduce client connectivity (in case of unreliable or slow clients), the maximum cache capacity can be increased. Otherwise, when most clients are reliable, the maximum cache capacity can be reduced to improve the performance on the aggregation server.

`FedQP` can leverage multiple cache levels in high-throughput environments. Dedicated cache servers reduce compute power for aggregation. The system initially uses a single cache and adds a second level for frequently used queries as processing continues. In this two-cache setup, `FedQP` uses high computation capacity specifically for the second cache, optimizing resources for frequent queries.

Several other caching strategies can be envisioned for `FedQP`, which require further analysis and are left as future work.

D. CLIENT DATA PARTITIONING

Clients can optionally partition their data and submit multiple centroids, each representing a subset of their dataset. As centroids increase, clients expose more information about their datasets. Thus, a balance of accuracy and privacy can be achieved by submitting an appropriate number of centroids. The number of centroids is application-dependent.

To systematically partition the dataset, the client executes Algorithm 3. The client can partition the dataset into $P \geq 1$ partitions. The Algorithm 3 receives the number of partitions P needed and slices the dataset. The algorithm randomly selects P locations in the dataset and divides the data before and after the selected location. Then, the centroids for each partition are computed (lines 3–6). The algorithm assesses the quality of partitions (line 7) using a threshold as a minimum difference value between any two centroids. If the difference is below the minimum, the algorithm repeats the process. The loop also has a maximum number of iterations to guarantee a halt.

E. PRIVACY ANALYSIS

We analyze Algorithm 1, demonstrating that `FedQP` minimally affects the privacy of the client's training data. The goal is to fully conceal training datasets from both the server and query clients seeking model predictions. `FedQP` only requires sharing client centroids (e.g., arithmetic means). Further, a client model M_i of the training dataset never leaves the client's machine as Algorithm 1 only requests the output of each M_i when processing a query. In the following subsections, we briefly explore the privacy implications of our design from a model reconstruction perspective. Later in Section IV-D, we provide empirical results for reconstructing a real-world dataset using the following. exposed centroids.

Algorithm 3 Centroid Computation for Federated Learning Client

```
Require: Dataset $\mathcal{D}$, number of partitions $P$, threshold $\epsilon$,
    maximum iterations $T$
 1: $t \leftarrow 0$
 2: repeat
 3:     Randomly select $P-1$ locations in $\mathcal{D}$ to split into $P$
        partitions $\{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_P\}$
 4:     for $j = 1$ to $P$ do
 5:         Compute centroid $\bar{x}_j$ using Eq. 1 in Section III-A.
 6:     end for
 7:     Compute minimum difference $d_{\min} \leftarrow \min_{i \neq j} \|\bar{x}_i -
        \bar{x}_j\|$
 8:     $t \leftarrow t + 1$
 9: until $d_{\min} \geq \epsilon$ or $t \geq T$
10: return $\{\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_P\}$
```

1) Privacy Guarantees from Centroid-based Design

Consider a client's dataset D_i containing $|I_i|$ data points $x_{ij}, f(x_{ij}) j \in I_i$ represented by centroid $\bar{x}_i$. Assume that each data point is $x_{ij} \in \mathbb{R}^p$. An adversary (e.g., the coordinating server) with access to $\bar{x}_i$ wants to recover each individual data point x_{ij} in the training dataset. Consider the simple case where the dataset has a single dimension ($p = 1$). The centroid $\bar{x}_i$ provides the arithmetic mean of the real numbers in the vector $x_{i1}, x_{i2}, \ldots, x_{i|I_i|}$. Even if the adversary knows $|I_i|$, reconstructing the dataset requires finding numbers that sum to $\bar{x}_i \times |I_i|$ (when arithmetic mean is used as a centroid). This is akin to the subset sum problem with the additional constraint that exactly $|I_i|$ numbers must be used. Since the general subset sum problem is NP-complete [39], this variant with an additional cardinality constraint is at least as hard. Moreover, in practice, the adversary does not know $|I_i|$ since the training client is not required to expose it, making reconstruction even more difficult, since the adversary must also determine the correct size of the dataset.

2) Information Leakage Given the Centroid

Information leakage from a client's dataset can be assessed by how precisely any individual point can be reconstructed from its centroid. For a dataset D_i with $|I_i|$ points in $\mathbb{R}^p$ and centroid $\bar{x}_i$, at least one point lies within distance r of the centroid, where r is the radius of the smallest enclosing ball covering all points in D_i. The adversary does not know this radius, only that each point x_{ij} fits the constraint $\|\bar{x}_i - x_{ij}\| \leq d_{\max}$, where $d_{\max}$ is the maximum distance in the feature space. The adversary can guess a ball of radius $d_{\max}$ centered on $\bar{x}_i$ to contain the point. The volume of this ball in $\mathbb{R}^p$ grows exponentially with dimension p, so the uncertainty in reconstructing a point increases exponentially with dimensionality.

3) Privacy Implications of Exposing from Multiple Centroids

The privacy-utility trade-off in `FedQP` is evaluated based on how many partitions s a client makes in their dataset D_i. In one extreme, the dataset has a single centroid, offering maximum privacy according to Section III-E. Conversely, when each data point is a partition, it fully exposes the dataset via centroids. For intermediate s values, where $1 < s < |I_i|$, the dataset D_i divides into s subsets $\{D_{i1}, D_{i2}, \ldots, D_{is}\}$, each with its centroid $\bar{x}_{ij}$. As s grows, each centroid covers fewer points, simplifying an adversary's approximation of original data points within partitions. For a partition D_{ij} with k points, the adversary's search reduces from the whole feature space to a ball of radius $d_{\max}$ centered at $\bar{x}_{ij}$ with only k points. This greatly reduces the uncertainty of reconstruction compared to a single centroid, showing that finer partitions increase utility at the expense of privacy.

IV. EXPERIMENTAL RESULTS

We evaluated `FedQP` from multiple perspectives. We examine the accuracy of the learning algorithm in Section III. A federated learning query response may be at a disadvantage compared to (i) a local model or (ii) a theoretical global model built from the entire datasets of *all* clients. To show the efficacy, we stress-test our algorithm using a real-world intrusion detection dataset as outlined in Section IV-B.

In Section IV-C, we provide a simulation to evaluate the efficiency of client communication. Two scenarios assist in these tests. One is a system in which the datasets are volatile and regularly updated, while the federated learning algorithm responds to live queries. In the second scenario, the models are relatively more stable, and the query patterns vary from showing new unseen input to a more stable series of queries with repeated values.

We evaluated the level of privacy preservation by `FedQP` and compared it with existing solutions. As described in Section IV-D, `FedQP` is resilient against reconstruction attacks. We also evaluated the impact of the number of centroids on model accuracy (Section IV-E) and client privacy (Section IV-G).

We evaluated the resilience of our method against poisoning attacks using four aggregation functions (Sections IV-I), the effect of aggregation methods on accuracy (Section IV-F), and the effect of various norms on accuracy (Section IV-H.

A. ADVERSARIAL ASSUMPTIONS AND EXPERIMENTAL SETUP

1) Adversarial Model.

We consider a semi-honest adversary (e.g. the coordinating server or an external observer) who seeks to compromise privacy by reconstructing or inferring training data from shared components. The adversary has no access to raw training examples, local model parameters, or gradients. Instead, their view is limited to:

- The set of published client centroids $\{\bar{x}_i\}$,
- The number of centroids per client (known), and
- The number of samples per centroid $|I_i|$.

2) System Configuration.

The following global settings are used in all experiments.

- Datasets: Experiments are carried out on real-world data (except cache experiments).
- Client Partitioning: We simulate data splitting based on the feature space or through stochastic partitioning.
- Number of centroids: Each client publishes between 1 and 10 centroids, depending on the experimental configuration.
- Distance norm: the ℓ_2 norm is used as the default distance function in interpolation, unless explicitly varied.

3) Classification Accuracy Evaluation.

For utility evaluation, we report standard multiclass classification metrics that include accuracy, precision, recall, and F1-score (Section IV-D). Performance is measured under varying centroid counts and degrees of non-IID partitioning.

4) Robustness Evaluation.

To test robustness against poisoned clients (Section IV-H), we inject between 10% and 40% malicious clients who return fixed and incorrect predictions for any query. We evaluated the impact of different fusion strategies (mode, median, trimmed mean, and randomization) on prediction accuracy under attack.

The attacker does not use the internals of the model, auxiliary training data, or side channel information. However, an approximate reconstruction may still reveal statistical patterns. Our metrics are designed to quantify this leakage from multiple analytical perspectives.

B. IS FEDQP ACCURATE?

To examine the accuracy of `FedQP`, we use the ToN IoT intrusion detection data set [40]. The dataset is a recent recording of heterogeneous data sources from Internet of Things (IoT) devices. Various operating systems and software stacks are used in the IoT network from which the data was collected.

1) Description of the Dataset

The learning task is to classify the type of traffic (see Table 1) as normal or a specific type of attack. The values of the types of traffic and the number of data points for each value are given in Table 2.

The dataset includes 461043 samples, of which 449972 are unique. Unique samples are used in the experiments. From the 45 available features, 10 were selected to exclude incomplete data and redundant feature values. The 10 features (in Table 1) are subjectively selected to represent the true distinctive features of the network requests. The remaining 35 features were dropped because of (i) extreme redundancy in values and (ii) negligible impact on prediction accuracy. A detailed discussion of the reduction of the dimensionality of the dataset is beyond the scope of this work as our focus is on showcasing the accuracy of `FedQP`.

TABLE 1: Client Dataset Intervals and Data Points

Features	Description
src_ip	Client IP
src_port	Client Port
dst_ip	Server IP
dst_port	Server Port
proto	Network Protocol (TCP/UDP)
service	Network Service (such as SSH)
duration	Connection duration in seconds
src_bytes	Bytes transferred from client
dst_bytes	Bytes transferred from server
type	Traffic type label (such as normal, DDoS)

TABLE 2: Number of Points for Each Traffic Type

Traffic Type	Number of Points
normal	288929
backdoor	20000
password	20000
ddos	20000
dos	20000
scanning	20000
injection	20000
ransomware	20000
xss	20000
mitm	1043

2) Creating Federated Learning Clients

To simulate a federated set of clients, we partition the ToN IoT dataset based on network request duration (in seconds), captured as

$$end\_time - start\_time,$$

corresponding to a single network request. The setup includes five clients with duration intervals as given in Table 3, producing distinct data subsets for each client. The clients are created to mimic distinct client scenarios. The intervals were chosen to roughly balance the number of unique data points in which subset.

TABLE 3: Client Dataset Intervals and Data Points

Interval	Data Points
$[0.1, 0.5)$	943
$[0.5, 2)$	393
$[2, 5)$	208
$[5, 20]$	189
$[20, \infty)$	254

3) Accuracy Measurement Approach

The experiments measure the (i) accuracy of each client (independent of the datasets from other clients), (ii) the accuracy of a unified single-client dataset (all data points are trained and tested with no client split), and (iii) the accuracy of a combined federated interpolation model with the synthetic clients as described earlier.

For a fair accuracy comparison, we tested the accuracy of models trained for each client on test data from other clients. Thus, for each client dataset D_i, we first generate a model M_i trained on all the data points in D_i. In a real-world scenario, the client C_i, if working alone, would use M_i to answer queries and predict future values. Thus, we test M_i against the data points in each $D_j \in \mathcal{D}$ (where $\mathcal{D} = \bigcup D_j$, for $j = 1, \ldots, n$). The results show how each model can perform independently when tested with a subset of each client's dataset (including its dataset).

We also evaluate a global unified client scenario. The global client is the single-client scenario in which all the client datasets are combined into a single dataset for which a model is generated. This scenario is added as an extreme benchmark, showing how each client's model can be compared to the case where all clients expose their datasets to a single aggregator to create a unified model.

Finally, we evaluate and compare the performance of both `FedAvg` and `FedQP` in which a unified global model is created. As explained below, one can see that `FedQP` has a similar accuracy to the unified global model, although it does not have access to any data points. However, `FedAvg` falls short of `FedQP` in classification accuracy.

4) Experimental Setup

Each client owns a non-overlapping dataset (based on packet arrival time). A client i independently trains a random forest classifier M_i, which responds to queries according to the local training data. In all experiments, the dataset is split into a 75% training and a 25% testing subsets with random data shuffling before the split. The classifier uses the parameters in Table 4.

5) Classification Results

The learning task is to classify network requests as benign or a specific attack (see Table 1). We use precision, recall, and the F1 score to measure the performance of each client's model. Table 5 shows the results of each performance indicator (averaged over the performance for all classes in the dataset).

We performed the experiments in four major settings. First, the unified model is trained on all combined data sets, assuming a scenario in which one party owns all the data. As expected, the model trained in this scenario outperforms all other models as it can access all data points. Second, the table shows the average performance of the model trained separately for each client dataset when cross-tested with the training sets of all other clients. In this scenario, each client builds an independent local model without access to any data from other clients. In our results, the model performance varies between clients, with precision as low as 0.75 and as high as 0.94 (close to the performance of the unified model).

Third, we compare `FedQP` with two other aggregation approaches. `FedAvg` is similar to `FedQP`, except using the arithmetic mean of the responses of the clients as opposed to the statistical mode. Note that `FedAvg` averages predictions of participating clients instead of averaging model parameters. This `FedAvg` is consistent with our federated interpolation approach. `FedAvg` of gradient iterations is inherently inapplicable to random forest models as they are not produced using

Parameter	Default Value
`n_estimators`	100 (number of trees in the forest)
`criterion`	`'gini'` (function to measure split quality)
`max_depth`	None (no limit on tree depth)
`min_samples_split`	2 (minimum samples to split a node)
`min_samples_leaf`	1 (minimum samples to be a leaf node)
`max_features`	`'sqrt'` (number of features to consider for best split)
`bootstrap`	True (use bootstrap samples for training)

TABLE 4: Parameters of the random forest classifier for training client models as used in `scikit-learn` [41].

TABLE 5: Performance of random forest classification for various training models. The unified model includes all the datasets trained on a single model. Single clients refer to independent local models. `FedAvg` averages independent client model responses and `MulTreeFed` uses multiple local trees in a single model. Clients are numbered 1–5. The performance indicators are values in $[0, 1]$ with 1 being the best performance.

Training Model	Classes	Precision	Recall	F1-Score
Unified	10	0.95	.93	0.94
Single Client 1	8	0.75	0.75	0.74
Single Client 2	6	0.78	0.8	0.78
Single Client 3	7	0.94	0.92	0.93
Single Client 4	7	0.83	0.75	0.72
Single Client 5	7	0.76	0.76	0.76
`FedAvg`	10	0.89	0.90	0.89
`MulTreeFed`	10	0.91	0.92	0.91
`FedQP`	**10**	**0.95**	**0.94**	**0.94**

the gradient descent algorithm. The next comparison is with multiple trees federation `MulTreeFed`. In this case, we use the trees from local clients. When aggregating the models, we add trees from each client in a combined multiple-tree model.

Table 5 shows the results of `FedQP`. In this scenario, each client trains its model. Then, the `FedQP` model responds to queries from test sets of all clients. The performance results show that `FedQP` performs as well as the unified model (where one model is trained on all data points) and slightly outperforms the unified model in the recall. `FedQP` also outperforms `MulTreeFed` and `FedAvg`.

C. CAN FEDQP REDUCE CLIENT COMMUNICATIONS?

As explained in Section III, to produce a prediction $M(Y)$ `FedQP` requires client communications when the server S receives a query Y. In this section, we investigate whether query processing (as in Algorithm 1) is a communication bottleneck that makes `FedQP` impractical. We designed a simulation model of Algorithm 2, investigating the impact of query cache in reducing communication rounds with the clients.

Recall that the query cache stores the results of previous queries of the federated model to be reused in future client requests. The cache algorithm uses the Euclidean or cosine distance functions to find *nearly identical* previous queries from the cache. In the rest of this section, we explain the method for generating a synthetic simulation dataset and present the results of cache hits when executing Algorithm 2.

1) Generating a Synthetic Dataset

The simulation aims to measure the algorithm's performance under several client behavioral assumptions. A simulation instance includes N as the number of clients, each possessing a single local dataset D_i. All clients agree on a single learning task. Client datasets are synthetic and are arbitrarily set to 10 dimensions $x_1, \ldots, x_{10}$. We randomly generate the value of x_i for each row $x_i \sim \mathcal{N}(1, 0.5)$.

Query clients simulate the submission of queries. Clients may submit queries that are (i) previously unseen, (ii) identical to values of a row in a training dataset, or (iii) similar to the values of a row in a training dataset with small perturbations: $x_{ij} = x_j + \epsilon_{ij}$.

2) Cache Hit Results

We compare four client query arrival behaviors (burst submission, quadratic increase, constant rate, and sinusoidal rate) and two distance metrics (Euclidean and cosine) to find nearly identical cache records to respond to clients.

The client query submission behaviors capture various scenarios to simulate query processing. These scenarios are important because `FedQP` depends on live processing of queries from data-owning clients. In the tested scenarios, clients may show

1) Burst-submission pattern, where a large number of clients simultaneously issue queries to the server, triggering multiple concurrent connections with data-owning clients.
2) Quadratically increasing query rate, modeled by a quadratic function to represent a gradual but accelerating growth in query volume over time.
3) Constant query rate, where queries arrive at the server at a fixed interval throughout the simulation.
4) Sinusoidal query rate, representing periodic fluctuations in query arrivals, simulating moderately varying load conditions.

Recall that the original training data points are randomly drawn from the Gaussian distribution. We add a slight random noise to one dataset dimension to simulate the cache hits and submit it as a new and unseen client query. With a fresh cache, the first query results in a cache miss. As the system receives new queries, cache hits start to occur.

Table 6 shows the result of simulating Algorithm 2. 1000 client queries are used to simulate the cache. The cache algorithm uses the Euclidean distance to generate the results in Table 6.

To see the impact of the distance measurement method, we added cosine and Manhattan distance to the available distance measures. The Manhattan distance had poor performance, with almost 1000 queries generating cache misses. However, the cosine distance performed the best, compared to the other two distance measures, as shown in Figure 4.

3) Simulating Cache with IoT Dataset

The previous experiment showed the effect of using cache on the number of communications needed. As the number of cache hits grows, the number of communications with client models decreases.

Here, we investigate whether using Algorithm 2 affects the accuracy of a model. In this experiment, we compare the accuracy of `FedQP` and a unified system (where all the data points are visible for the server) using a query cache. The question is whether a query cache significantly reduces the model's accuracy. The query cache uses three thresholds: 10%, 20%, and 30% difference between the query and a cached query. The cached queries are in a first-in-first-out list, where the first match with the threshold is used as the predicted response.

Table 7 shows the results of the experiment. The Accuracy Difference for each configuration with cache is defined as the difference in F1 score between the "No Cache" case and the cache enabled case, calculated as follows:

$$\text{Accuracy Difference} = \text{F1-score (No Cache)} - \text{F1-score (Cache)}$$

These results show a slight loss in accuracy due to the cache predictions described above. This is because cache predictions use a numerical distance between a submitted and cached query. Thus, in some cases, the difference between the cached query and the submitted query results is in different classes. However, the reduced accuracy is not significant. Depending on the application, the cache may be adjusted for better performance or accuracy.

4) Lessons Learned

Simulating cache hits shows that `FedQP` can utilize the distance measurement on the dataset to benefit from a cache of previous query responses. Caching these responses does not result in any further privacy exposure. The server in `FedQP` has access to queries and responses from data-owning clients and does not learn new information by using the cache. In addition, increasing relatively smaller cache thresholds can

FIGURE 4: Various threshold values and distance calculation methods for reusing query responses from the cache, avoiding communications with clients.

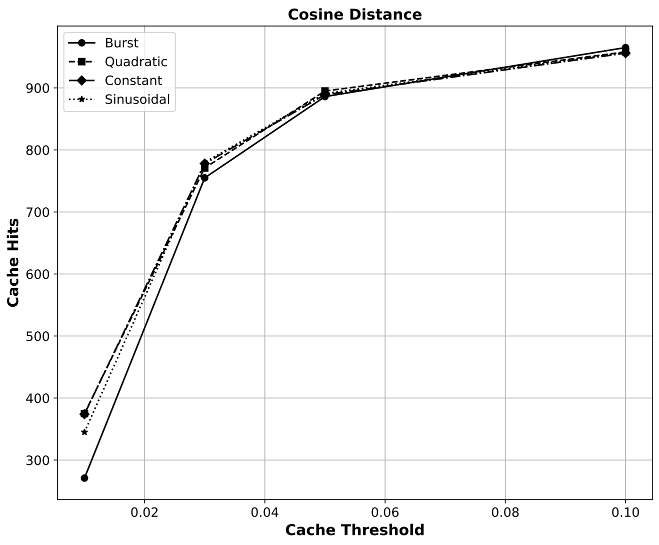


FIGURE 5: Cache hits under cosine distance of a new query and a cache item.

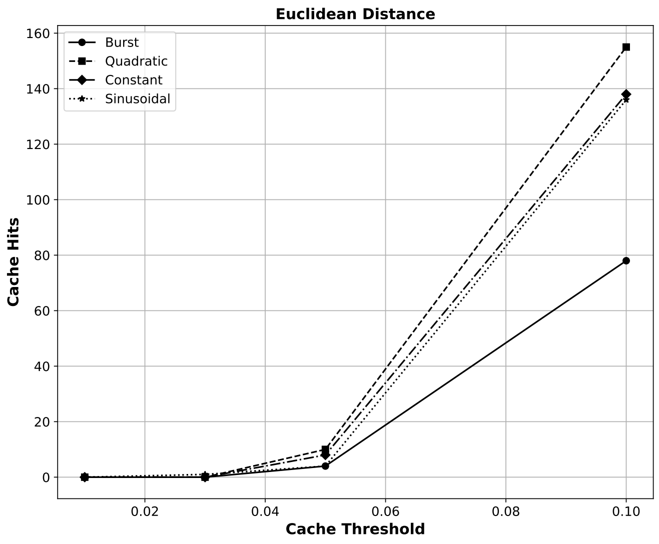


FIGURE 6: Cache hits under Euclidean distance of a new query and a cache item.

profoundly impact the number of cache hits, thus reducing redundant communications for query processing. However, distance metrics have an even greater impact on the outcome. As observed in Figure 4, the cosine distance with a cache threshold < 0.02 increases the cache hit by threefold compared to the Euclidean distance.

Our simulation is limited to the scenarios tested, the distance measures used in randomly generated datasets, and random differences along a single dimension. The results can potentially vary with various datasets. However, our simulation setup overcomes limitations by normalizing the data

TABLE 6: Comparison of Cache Hit Rates Across Different Query Submission Behaviors

Query Submission Behavior	Cache Hits	Total Queries
Burst submission	93	1000
Quadratic increase	123	1000
Constant rate	125	1000
Sinusoidal rate	125	1000

Number of cache hits observed for different query submission behaviors in a federated learning simulation. The threshold for cache hit is a distance ≤ 0.05. The scenarios include burst submission, quadratic increase, constant rate, and sinusoidal rate. The cache hit rate reflects the efficiency of the caching mechanism in recognizing and reusing previously computed results for similar queries.

TABLE 7: Impact of cache on accuracy in FedQP and a unified setting.

Configuration	Precision	Recall	F1-Score	Cache Hit Rate (%)	Accuracy Difference
FedQP (No Cache)	0.91	0.93	0.91	N/A	0.00
Unified (No Cache)	1.00	1.00	1.00	N/A	0.00
FedQP (10% Cache)	0.91	0.93	0.91	9.7%	0.00
Unified (10% Cache)	0.99	0.99	0.99	2.7%	0.01
FedQP (20% Cache)	0.91	0.93	0.91	20.3%	0.00
Unified (20% Cache)	0.97	0.97	0.97	7.0%	0.03
FedQP (30% Cache)	0.90	0.92	0.90	34.8%	0.01
Unified (30% Cache)	0.95	0.95	0.95	11.6%	0.05

points and covering common client query submission behaviors. Further studies with live implementations of FedQP can help investigate the benefits of caching with real datasets. This investigation is beyond the scope of this work as the article focuses on the design of the methodology from the privacy perspective.

D. PRIVACY-PRESERVATION RESULTS

We evaluated privacy preservation by comparing our method with FedAvg [1], FedProx [14], and SCAFFOLD [15]. We evaluated these methods using exposed training client data for model inversion attacks. This evaluation aims to compare our method with gradient-based methods from an adversary's perspective. However, a detailed study of attacks and the resiliency of various approaches is beyond the scope of this work.

The compared methods represent different approaches to federated learning. FedAvg aggregates local model updates through weighted averaging of client gradients, establishing the foundation for federated optimization. FedProx is a minor generalization of FedAvg (adding a proximal term to local updates) with the objective of better convergence guarantees. SCAFFOLD employs control variates to correct for client drift, maintaining variance reduction through server-client control signals that track the optimization trajectory, making it particularly effective for heterogeneous data distributions.

Our evaluation employs three distinct metrics to assess the preservation of privacy. The reconstruction error measures the Euclidean distance between the original and reconstructed data points, normalized by the dimensionality of the data. Information loss quantifies the mutual reduction in information between the original and reconstructed datasets, calculated using the Kullback-Leibler divergence [42] of their respective probability distributions. The distribution divergence metric evaluates how well the statistical properties of the original dataset are preserved in the reconstruction, computed as the Jensen-Shannon divergence [43] between the empirical distributions of the original and reconstructed datasets. For all metrics, higher values indicate stronger privacy preservation, as they represent greater difficulty in reconstructing the original data.

We used the US Adult Income dataset [13] for privacy analysis simulations. The Adult Income dataset, also known as the Census Income dataset, contains demographic and employment information from the 1994 US Census database. It consists of 48,842 instances with 14 attributes including age, work class, education, marital status, occupation, and other socioeconomic factors, with the prediction task of determining whether an individual's annual income exceeds $50,000. For our experiments, we used the continuous numerical features (age, final weight, education-num, capital-gain, capital-loss, and hours-per-week) resulting in a 6-dimensional feature space. The dataset exhibits natural heterogeneity in its distribution, making it particularly suitable for evaluating federated learning approaches.

In our simulations, we attempt to reconstruct the models given the available information. In case of our FedQP, the adversary only has the centroid and the number of points in the dataset. In other methods, the adversary knows the gradient iterations and the number of points in the dataset.

Table 8 and Figure 8 show the results of our simulations. Our model outperforms the gradient-based models in the reconstruction error and information loss metrics, while achieving similar results in distribution divergence. Our results show that FedQP improves privacy protection even when under attack from a powerful adversary with the knowledge of the centroid and the number of data points in the dataset.

TABLE 8: Comprehensive Privacy Preservation Analysis

Metric	FedQP	FedAvg	FedProx	SCAFFOLD
Reconstruction Error	**1.371**	1.150	1.107	0.802
Information Loss	**2.818**	2.509	2.511	0.539
Distribution Divergence	0.456	0.535	0.509	0.389

Note: Higher values indicate better privacy preservation for all metrics

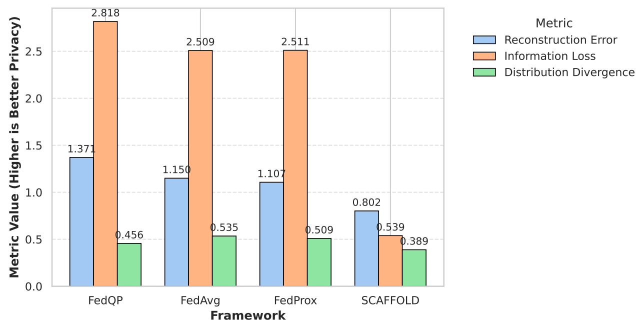


FIGURE 7: A bar chart of privacy preservation metrics for FedQP, FedAvg, FedProx, and SCAFFOLD.

E. IMPACT OF CENTROIDS ON ACCURACY

We evaluated the impact of the number of centroids per client. The experiments were performed using a federated setup with five clients, each working with a separate partition of the dataset (selected uniformly).

The number of centroids used to represent each client's dataset has a significant effect on the overall accuracy of FedQP. As shown in Table 9, increasing the number of centroids results in better accuracy. However, the improvement plateaued after using five centroids. Note that the experiments are conducted on the US Adult Income dataset.

Centroids	Accuracy
1	0.8432
2	0.8675
3	0.8724
4	0.8768
5	**0.8801**
6	0.8791
7	0.8782
8	0.8767
9	0.8754
10	0.8748

TABLE 9: Impact of Number of Centroids on FedQP Accuracy. The results are based on weighed average aggregation. As the centroids increase, there is an improvement in accuracy, which can plateau after five centroids.

These results demonstrate that increasing the number of centroids from 1 to 5 leads to a consistent improvement in accuracy, reaching a peak of 0.8801 with 5 centroids. Beyond this point, we observe a slight decline in performance, suggesting that too many centroids may introduce unnecessary complexity without additional benefits. This pattern indicates that there is an optimal number of centroids to represent client data distributions in federated learning environments.

F. COMPARISON OF AGGREGATION METHODS

FedQP supports multiple aggregation methods for combining client predictions. Table 10 presents the accuracy achieved with different aggregation approaches while keeping the number of centroids fixed at 1.

TABLE 10: Performance Comparison of Different Aggregation Methods in FedQP

Aggregation Method	Accuracy
weighted average	**0.8432**
mode	0.8276
maximum	0.7914
median	0.8364

Note: All experiments conducted with a single centroid per client

The results show that the weighted average aggregation method achieves the highest accuracy (0.8432), closely followed by the median method (0.8364). The mode method delivers reasonable performance (0.8276), while the maximum method lags behind (0.7914). This suggests that weighted averaging, which considers the distance between client centroids and queries, provides the most effective mechanism for prediction aggregation in FedQP.

These experimental findings validate the effectiveness of our approach and highlight the importance of carefully selecting both the number of centroids and the aggregation method when implementing FedQP in practice.

G. PRIVACY-UTILITY TRADE-OFF ANALYSIS

The relationship between privacy preservation and the number of centroids is crucial for understanding the trade-offs involved in implementing FedQP. Figure 8 illustrates how different privacy metrics are affected as the number of centroids increases.

TABLE 11: Effect of Number of Centroids on Privacy Metrics

Privacy Metric	Single Centroid	Ten Centroids
Reconstruction Error	**2.0134**	0.8768
Information Loss	**2.5892**	1.7643
Distribution Divergence	**0.5921**	0.3804
Neighbor Privacy	**1.0000**	1.0000

Note: Higher values indicate better privacy preservation for all metrics.

As shown in Figure 8, with fewer centroids, the system achieves higher privacy scores with multiple metrics. We observe that the reconstruction error shows the most significant decline (56.4%) as the number of centroids increases from 1 to 10, indicating that adversaries gain a better ability to

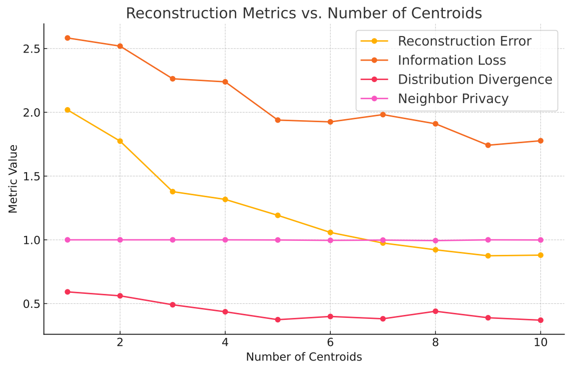


FIGURE 8: Impact of Number of Centroids on Privacy Preservation Metrics. Higher values indicate better preservation of privacy.

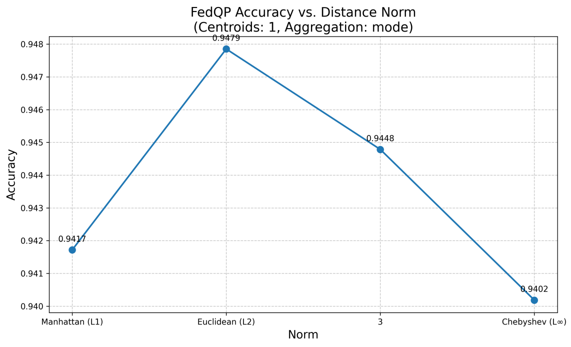


FIGURE 9: A comparison of FedQP's accuracy using various norms for calculating the distance of a query with the client centroids.

reconstruct the original data points when more centroids are exposed. Information loss decreases by 31. 9%, reflecting the reduced uncertainty in the statistical properties of the reconstructed data. Distribution divergence follows the same pattern with a 35.8% reduction, while Neighbor Privacy remains consistently high across all configurations.

H. COMPARISON OF VARIOUS NORMS

Recall that FedQP relies on calculating the distance of a query submitted with the centroids of participating clients to produce a unified query response. Calculating the distance (Section III-A) can be done using various norms (such as Manhattan and Euclidean norms).

We conducted experiments to measure the impact of the norm on the final model performance. The results are shown in Figure 9. The results do not show a significant change in the accuracy achieved. although higher order norms achieve a slightly lower accuracy (accuracy is measured using subset accuracy, which requires that the predicted label set for each sample exactly matches the ground truth).

I. ROBUSTNESS AGAINST MALICIOUS CLIENTS

Robustness of federated learning aggregation is a subject of previous studies [33], [44]. For example, Nabavirazavi et al. suggest using randomization to reduce the impact of poisonous input [33]. Although, FedQP is fundamentally different from gradient-based methods, robustness of FedQP can potentially benefit from randomization.

To test the robustness of FedQP against model attacks, we conducted experiments with three types of attacks and varying percentages of malicious clients (0-40%). We evaluated four different aggregation functions that work intrinsically as defense mechanisms: mode, median, trimmed mean, and randomization (random selection of a subset of clients). We used the US Adult Income dataset with 50 clients. The data set was divided into 50 subsets with sizes nearly equal.

We used two types of attacks. First, in a flip attack, the attacker randomly flips the binary decision of the model. Second, in a centroid attack, the attacker sends false centroids to the aggregator server to influence its decision.

As shown in Figure 10, FedQP demonstrates resilience to prediction flipping attacks, where malicious clients invert their prediction probabilities (for example, changing 0.2 to 0.8). The median defense mechanism maintains its performance even with malicious clients 40%, experiencing only a minor accuracy degradation of 0.84 to 0.81.

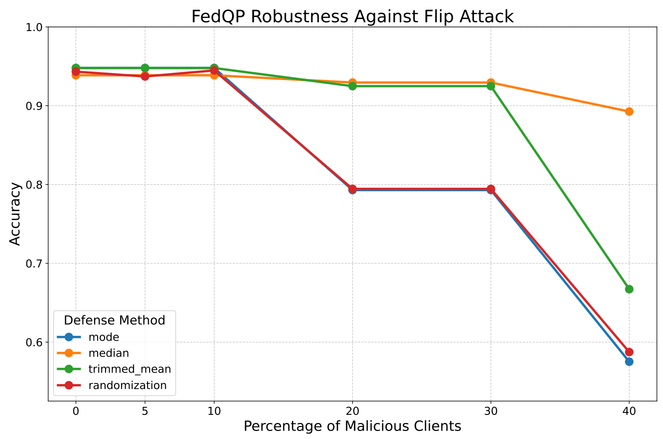


FIGURE 10: FedQP robustness against prediction flipping attacks. The horizontal axis shows the percentage of malicious clients, while the vertical axis shows classification accuracy.

Figure 11 shows the performance of FedQP in centroid manipulation attacks, where malicious clients report false centroids to manipulate client selection. With this attack vector that targets the fundamental FedQP client selection mechanism, the median aggregation method maintains stable performance up to 30% malicious clients. This attack is potentially more damaging as it compromises the distance-based weighting central to FedQP's design, yet the system maintains reasonable accuracy with appropriate defenses.

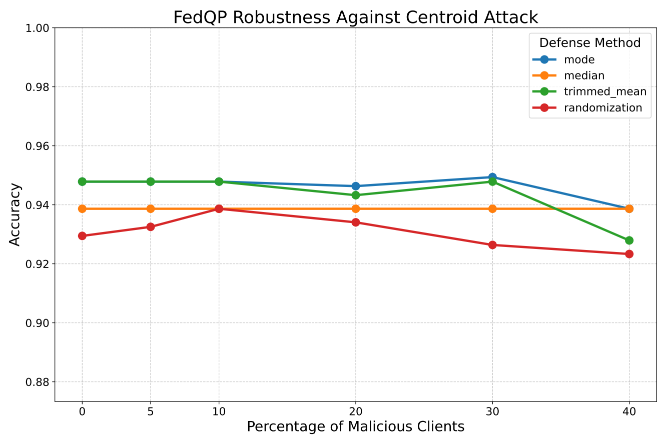


FIGURE 11: FedQP robustness against centroid manipulation attacks. This attack targets client selection rather than prediction values.

V. LIMITATIONS

FedQP demonstrates strong robustness, scalability, and privacy-preserving capabilities. However, we acknowledge several limitations of the current design.

First, although FedQP avoids parameter sharing and gradient aggregation, it remains susceptible to adversarial behavior at the prediction fusion stage. If a sufficiently large fraction of responding clients are malicious and collude to return misleading predictions for a specific query, they may bias the interpolated result, particularly when robust fusion (e.g., median or trimmed mean) is not enabled.

Second, FedQP does not incorporate explicit trust scoring, anomaly detection, or behavior profiling for participating clients. All selected responses are treated equally unless a robust aggregation strategy is applied. As such, adaptive attacks that exploit query patterns or poison only selective centroids could reduce output fidelity over time.

Finally, while our evaluation includes non-IID data, variable centroid granularity, and adversarial conditions, we do not yet explore dynamic client dropout, client drift across rounds, or integration with fully asynchronous communication models.

Addressing these limitations is a key direction for future work, including dynamic trust-weighted fusion, adaptive privacy budgeting during querying, and integration with secure multiparty computation frameworks.

Additional research is needed to analyze FedQP's compatibility with time-series data, streaming queries, and multitask learning settings. Formal privacy guarantees under differential privacy or local differential privacy assumptions are also promising areas for exploration.

VI. CONCLUSION

This paper introduces FedQP, a federated query processing framework designed to aggregate heterogeneous local models from distributed clients while preserving privacy. Unlike traditional federated learning approaches, FedQP enables secure, model-agnostic inference by leveraging client-side centroids and relevance-based query processing. Clients participate without sharing raw data or model parameters, thereby reducing exposure risks.

Experimental evaluations demonstrate that FedQP achieves predictive performance comparable to centralized models and significantly improves over individual client models. Furthermore, by incorporating relevance-based client selection and caching, the framework minimizes communication overhead.

FedQP accommodates both gradient-based and non-gradient models, such as decision trees, and is extensible to various data modalities. These features make it well-suited for deployment in heterogeneous, privacy-sensitive environments.

The underlying methodology of FedQP can be extended to support non-tabular data modalities, such as images or videos. Although its general applicability to these domains requires further investigation, the framework is designed to be both data-agnostic and model-agnostic. Clients are only required to agree on (i) a common subset of features, (ii) a shared target variable, and (iii) the learning task type (e.g., classification or regression). Aggregation for deep models, such as those using convolutional architectures, remains open to further experimentation.

Future work includes integrating dynamic client trust scoring, asynchronous communication support, and advanced defense mechanisms against adaptive attacks.

REFERENCES

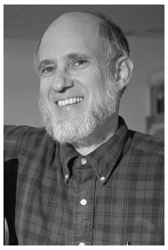
[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in Artificial Intelligence and Statistics, pp. 1273–1282, PMLR, 2017.
[2] K. Martineau, "What is federated learning?," Aug 2022. https://research.ibm.com/blog/what-is-federated-learning.
[3] K. Pykes, "A guide to monitoring machine learning models in production." https://developer.nvidia.com/blog/a-guide-to-monitoring-machine-learning-models-in-production/, 2023. Accessed: 2024-01-15.
[4] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in Proceedings of COMPSTAT'2010: 19th International Conference on Computational Statistics, pp. 177–186, Springer, August 2010.
[5] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning," in Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP), pp. 1–15, IEEE, 2018.
[6] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, CCS '15, (New York, NY, USA), pp. 1310–1321, ACM, Association for Computing Machinery, 2015.
[7] R. Xu, N. Baracaldo, Y. Zhou, A. Anwar, and H. Ludwig, "Hybridalpha: An efficient approach for privacy-preserving federated learning," in Proceedings of the 12th ACM workshop on artificial intelligence and security, pp. 13–23, ACM, 2019.
[8] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, "A survey on homomorphic encryption schemes: Theory and implementation," ACM Computing Surveys (Csur), vol. 51, no. 4, pp. 1–35, 2018.
[9] C. Zhang, S. Li, J. Xia, W. Wang, F. Yan, and Y. Liu, "BatchCrypt: Efficient homomorphic encryption for Cross-Silo federated learning," in 2020 USENIX Annual Technical Conference (USENIX ATC 20), pp. 493–506, USENIX, USENIX Association, July 2020.

[10] H. J. Almohri and L. T. Watson, “Model-agnostic federated learning for privacy-preserving systems,” in 2023 IEEE Secure Development Conference (SecDev), (Los Alamitos, CA, USA), pp. 99–105, IEEE, IEEE Computer Society, oct 2023.
[11] M. Fredrikson, S. Jha, and T. Ristenpart, “Model inversion attacks that exploit confidence information and basic countermeasures,” in Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, CCS ’15, (New York, NY, USA), pp. 1322–1333, ACM, Association for Computing Machinery, 2015.
[12] J. Layne, Q. E. A. Ratul, E. Serra, and S. Jajodia, “Analyzing robustness of automatic scientific claim verification tools against adversarial rephrasing attacks,” ACM Trans. Intell. Syst. Technol., vol. 15, Nov. 2024.
[13] B. Becker and R. Kohavi, “Adult.” UCI Machine Learning Repository, 1996. DOI: https://doi.org/10.24432/C5XW20.
[14] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, “Federated optimization in heterogeneous networks,” Proceedings of Machine learning and systems, vol. 2, pp. 429–450, 2020.
[15] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, “Scaffold: Stochastic controlled averaging for federated learning,” in International conference on machine learning, pp. 5132–5143, PMLR, 2020.
[16] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, “Practical secure aggregation for privacy-preserving machine learning,” in Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS ’17, (New York, NY, USA), pp. 1175–1191, ACM, Association for Computing Machinery, 2017.
[17] L. Zhu, Z. Liu, and S. Han, “Deep leakage from gradients,” in Advances in Neural Information Processing Systems, vol. 32, Curran, Curran Associates, Inc., 2019.
[18] Y. Zhang, D. Zeng, J. Luo, X. Fu, G. Chen, Z. Xu, and I. King, “A survey of trustworthy federated learning: Issues, solutions, and challenges,” ACM Trans. Intell. Syst. Technol., vol. 15, Oct. 2024.
[19] J. Zhao, A. Sharma, A. Elkordy, Y. H. Ezzeldin, S. Avestimehr, and S. Bagchi, “LOKI: Large-scale data reconstruction attack against federated learning through model manipulation,” in 2024 IEEE Symposium on Security and Privacy (SP), (Los Alamitos, CA, USA), pp. 34–34, IEEE, IEEE Computer Society, may 2024.
[20] Q. Yang, Y. Liu, T. Chen, and Y. Tong, “Federated machine learning: Concept and applications,” ACM Trans. Intell. Syst. Technol., vol. 10, Jan. 2019.
[21] W. I. Thacker, J. Zhang, L. T. Watson, J. B. Birch, M. A. Iyer, and M. W. Berry, “Algorithm 905: Sheppack: Modified shepard algorithm for interpolation of scattered multivariate data,” ACM Trans. Math. Softw., vol. 37, Sept. 2010.
[22] T. H. Chang, L. T. Watson, T. C. H. Lux, A. R. Butt, K. W. Cameron, and Y. Hong, “Algorithm 1012: DELAUNAYSPARSE: Interpolation via a Sparse Subset of the Delaunay Triangulation in Medium to High Dimensions,” ACM Trans. Math. Softw., vol. 46, nov 2020.
[23] T. C. Lux, L. T. Watson, T. H. Chang, Y. Hong, and K. Cameron, “Interpolation of sparse high-dimensional data,” Numerical Algorithms, pp. 1–33, 2020.
[24] L. T. Watson, “Polynomial interpolation.” Online, 2019. https://courses.cs.vt.edu/cs3414/F19/.
[25] M. A. Iyer, L. T. Watson, and M. W. Berry, “SHEPPACK: A Fortran 95 Package for Interpolation Using the Modified Shepard Algorithm,” in Proceedings of the 44th Annual Southeast Regional Conference, ACM-SE 44, (New York, NY, USA), pp. 476–481, ACM, Association for Computing Machinery, 2006.
[26] W. Liu, L. Chen, Y. Chen, and W. Zhang, “Accelerating federated learning via momentum gradient descent,” IEEE Transactions on Parallel and Distributed Systems, vol. 31, no. 8, pp. 1754–1766, 2020.
[27] Z. Wu, Q. Ling, T. Chen, and G. B. Giannakis, “Federated variance-reduced stochastic gradient descent with robustness to byzantine attacks,” IEEE Transactions on Signal Processing, vol. 68, pp. 4583–4596, 2020.
[28] Y. Mansour, M. Mohri, J. Ro, and A. T. Suresh, “Three approaches for personalization with applications to federated learning.” arXiv, 2020.
[29] S. Luo, Y. Xiao, X. Zhang, Y. Liu, W. Ding, and L. Song, “Perfedrec++: Enhancing personalized federated recommendation with self-supervised pre-training,” ACM Trans. Intell. Syst. Technol., vol. 15, Nov. 2024.
[30] Y. Tan, G. Long, L. LIU, T. Zhou, Q. Lu, J. Jiang, and C. Zhang, “FedProto: Federated prototype learning across heterogeneous clients,” in Proceedings of the AAAI Conference on Artificial Intelligence, no. 8 in 36, pp. 8432–8440, AAAI, Jun. 2022.
[31] M. Yurochkin, M. Agarwal, S. Ghosh, K. Greenewald, and N. Hoang, “Statistical model aggregation via parameter matching,” Advances in Neural Information Processing Systems, vol. 32, pp. 10956–10966, 2019.
[32] S. Claici, M. Yurochkin, S. Ghosh, and J. Solomon, “Model Fusion with Kullback-Leibler Divergence,” in International Conference on Machine Learning, pp. 2038–2047, PMLR, 2020.
[33] S. Nabavirazavi, R. Taheri, and S. S. Iyengar, “Enhancing federated learning robustness through randomization and mixture,” Future Generation Computer Systems, vol. 158, pp. 28–43, 2024.
[34] A. Alsobeh and A. Shatnawi, “Integrating data-driven security, model checking, and self-adaptation for iot systems using bip components: A conceptual proposal model,” in Proceedings of the 2023 International Conference on Advances in Computing Research (ACR’23) (K. Daimi and A. Al Sadoon, eds.), (Cham), pp. 533–549, Springer Nature Switzerland, 2023.
[35] A. AlSobeh, A. Shatnawi, B. Al-Ahmad, A. Aljmal, and S. Khamaiseh, “Ai-powered aop: Enhancing runtime monitoring with large language models and statistical learning.,” International Journal of Advanced Computer Science & Applications, vol. 15, no. 11, 2024.
[36] T. C. H. Lux, L. T. Watson, T. H. Chang, Y. Hong, and K. Cameron, “Interpolation of sparse high-dimensional data,” Numerical Algorithms, 2020.
[37] K. Pillutla, S. M. Kakade, and Z. Harchaoui, “Robust aggregation for federated learning,” IEEE Transactions on Signal Processing, vol. 70, pp. 1142–1154, 2022.
[38] R. Taheri, F. Arabikhan, A. Gegov, and N. Akbari, “Robust aggregation function in federated learning,” in International Conference on Information and Knowledge Systems, pp. 168–175, Springer, 2023.
[39] J. C. Lagarias and A. M. Odlyzko, “Solving low-density subset sum problems,” J. ACM, vol. 32, p. 229–246, Jan. 1985.
[40] A. Alsaedi, N. Moustafa, Z. Tari, A. Mahmood, and A. Anwar, “TON_IoT telemetry dataset: A new generation dataset of iot and iiot for data-driven intrusion detection systems,” IEEE Access, vol. 8, pp. 165130–165150, 2020.
[41] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in python,” J. Mach. Learn. Res., vol. 12, p. 2825–2830, Nov. 2011.
[42] S. Kullback and R. A. Leibler, “On information and sufficiency,” The annals of mathematical statistics, vol. 22, no. 1, pp. 79–86, 1951.
[43] M. L. Menéndez, J. Pardo, L. Pardo, and M. Pardo, “The jensen-shannon divergence,” Journal of the Franklin Institute, vol. 334, no. 2, pp. 307–318, 1997.
[44] S. Nabavirazavi, R. Taheri, M. Shojafar, and S. S. Iyengar, “Impact of aggregation function randomization against model poisoning in federated learning,” in 22nd IEEE international conference on trust, security and privacy in computing and communications, TrustCom 2023, pp. 165–172, Institute of Electrical and Electronics Engineers Inc., 2024.

HUSSAIN M. J. ALMOHRI received the B.S. degree in Computer Science from Kuwait University and the M.S. and Ph.D. degrees in Computer Science from Kansas State University and Virginia Tech, respectively. He is currently an Associate Professor in the Department of Computer Science at Kuwait University, where he has also served as Chair. His research interests include systems and network security, secure cloud infrastructures, intrusion detection, and the application of optimization techniques in cybersecurity.

He has been the principal investigator on several nationally funded research projects supported by Kuwait University and the Kuwait Foundation for the Advancement of Sciences (KFAS) since 2013. He has supervised multiple theses and undergraduate capstone projects in cybersecurity and applied systems. His work has appeared in venues such as IEEE TDSC, IEEE TIFS, ACM TOMS, and ACM TAAS.

He is a Senior Member of both the IEEE and the ACM, and has served on the program committees of security conferences including ACSAC and SecDev. He is also a regular reviewer for leading journals such as IEEE TDSC, and IEEE TIFS.



LAYNE T. WATSON (F '93) received the B.A. degree (magna cum laude) in psychology and mathematics from the University of Evansville, Indiana, in 1969, and the Ph.D. degree in mathematics from the University of Michigan, Ann Arbor, in 1974.

He has worked for USNAD Crane, Sandia National Laboratories, and General Motors Research Laboratories and served on the faculties of the University of Michigan, Michigan State University, and University of Notre Dame. He is currently a professor of computer science, mathematics, and aerospace and ocean engineering at Virginia Polytechnic Institute and State University. He serves as senior editor of Applied Mathematics and Computation, and associate editor of Computational Optimization and Applications, Evolutionary Optimization, Engineering Computations, and the International Journal of High Performance Computing Applications. He is a fellow of the National Institute of Aerospace and the International Society of Intelligent Biological Medicine. He has published well over 300 refereed journal articles and 200 refereed conference papers. His research interests include fluid dynamics, solid mechanics, numerical analysis, optimization, parallel computation, mathematical software, image processing, and bioinformatics.