

Article

Secure Federated Learning Algorithms for Vertical and Combined Data Partitioning

Amir Anees ^{1,*} , Ding Ming ², Gnana Bharathy ³  and Lois Holloway ⁴ 

¹ South West Sydney Local Health District and Ingham Institute, South Western Sydney Clinical Campus, School of Clinical Medicine, UNSW, Sydney 2033, Australia

² Data61, CSIRO, Eveleigh 2015, Australia; ming.ding@data61.csiro.au

³ Australian Research Data Commons, University of Technology Sydney, Ultimo 2007, Australia; gnana.bharathy@uts.edu.au

⁴ South Western Sydney Clinical Campus, School of Clinical Medicine, UNSW, Sydney 2033, Australia; lois.holloway@health.nsw.gov.au

* Correspondence: a.anees@unsw.edu.au

Abstract

With the growing need for collaborative machine learning across institutions holding sensitive data, ensuring data privacy without compromising model performance has become an important challenge. This work introduces secure federated learning algorithms that use encryption and masking techniques to protect the privacy of data during collaborative model training. Three federated learning algorithms were developed: one for vertical federated learning and two combining horizontal and vertical data partitioning. The proposed algorithms are designed such that participating clients communicate only with the server, even when data exchange between clients is required. This exchange occurs through the server with the help of encryption and masking. The performance of the algorithms, evaluated in terms of accuracy and loss, shows competitive results. The accuracy remains unchanged compared to the centralised scenario for the vertical federated learning algorithm and one of the combined federated learning algorithms, and it remains highly competitive with the other combined federated learning algorithm. The privacy analyses conducted as part of this work demonstrate no risk of data leakage ensuring that no party involved can infer sensitive information.

Keywords: federated learning; vertical data partitioning; combined data partitioning; gradient inversion attack; data privacy; secure data sharing



Academic Editor: Qiang Duan

Received: 4 May 2026

Revised: 26 May 2026

Accepted: 29 May 2026

Published: 3 June 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

The growing need for privacy-preserving and collaborative machine learning has led to the development of Federated Learning (FL) [1]. In many real-world applications, organisations possess sensitive data that cannot be freely shared due to privacy regulations (such as HIPAA and GDPR), competitive concerns or logistical constraints (such as communication and computational) [2,3]. However, isolated datasets often lack the size, diversity or completeness required to train effective machine learning models [4]. FL addresses this challenge by enabling multiple organisations to collaboratively train a machine learning model without transferring their raw data to a centralised server. Instead, each participating organisation computes local updates such as gradients or model parameters, which are then aggregated at the server to improve a shared global

model. This process continues for several rounds until the global model is converged to a local minima.

Depending on how data is distributed across organisations, FL is generally categorised into three types: Horizontal FL (HFL) [5], Vertical FL (VFL) [6] and Combined FL [7]. In HFL, organisations share the same feature space but have different data points. For example, several hospitals collaborate to train a disease prediction model, where each hospital has the same patient characteristics but hold data for different groups of patients. In VFL, organisations share the same data points but have different feature spaces, such as a hospital and an insurance company holding partial information about the same set of patients. In combined settings, data can differ in both data points and features. An example from healthcare is a scenario where local clinics (hospitals and laboratories) collect patient health records, while a national registry holds long-term patient outcomes. In this case, the local clinics have horizontally partitioned data among themselves (similar features, different patients), while the data is vertically partitioned between the clinics and the registry (input features at the clinics, outcome features at the registry) which require a federated setup that can bridge both types of partitions. The different data partitioning scenarios in FL are illustrated in Figure 1.

1.1. Motivations

Although FL inherently preserves privacy by ensuring that local data never leaves the participating organisations, it is still susceptible to various security threats. One such threat is the Gradient Inversion Attack (GIA) [8], which can be employed to infer sensitive local data based on the gradients shared by the clients during the training process. In this attack, the server (or an eavesdropper between the server and the participating clients) with access to the gradients can use optimisation techniques to reverse-engineer the data that lead to those gradients. The server generates synthetic data and computes its corresponding synthetic gradients. The server then iteratively minimises the difference between the synthetic gradients and the gradients sent by the clients. By optimising the synthetic data over several iterations, the server fine-tunes it so that it closely resembles the original local data. Furthermore, over time, more sophisticated versions of the GIA have emerged [9–11]. These newer methods operate under softer assumptions, with fewer constraints on the adversary's knowledge making them even more effective and difficult to mitigate. These advanced attacks can better handle noise and incomplete data posing a more significant challenge to the privacy guarantees of FL systems.

While privacy-preserving techniques have received substantial attention in the context of HFL, VFL and CFL settings remain underexplored [12], specifically when facing advanced GIAs. This gap is difficult to address because, unlike HFL where only model parameters or gradients are shared, VFL and CFL involve sharing intermediate outputs (such as matrix computations), which can leak private information. Additionally, the fact that different parties own different parts of the data, with some holding input features and others holding the output feature, makes protecting privacy even more challenging.

Further, the earlier works [7,13] on VFL and CFL relied on architectures where clients also communicated with one another in addition to the server. This was effective in certain scenarios; however, it introduced the need for additional client-to-client communication channels which increases architectural and implementation complexity and makes practical deployment more challenging.

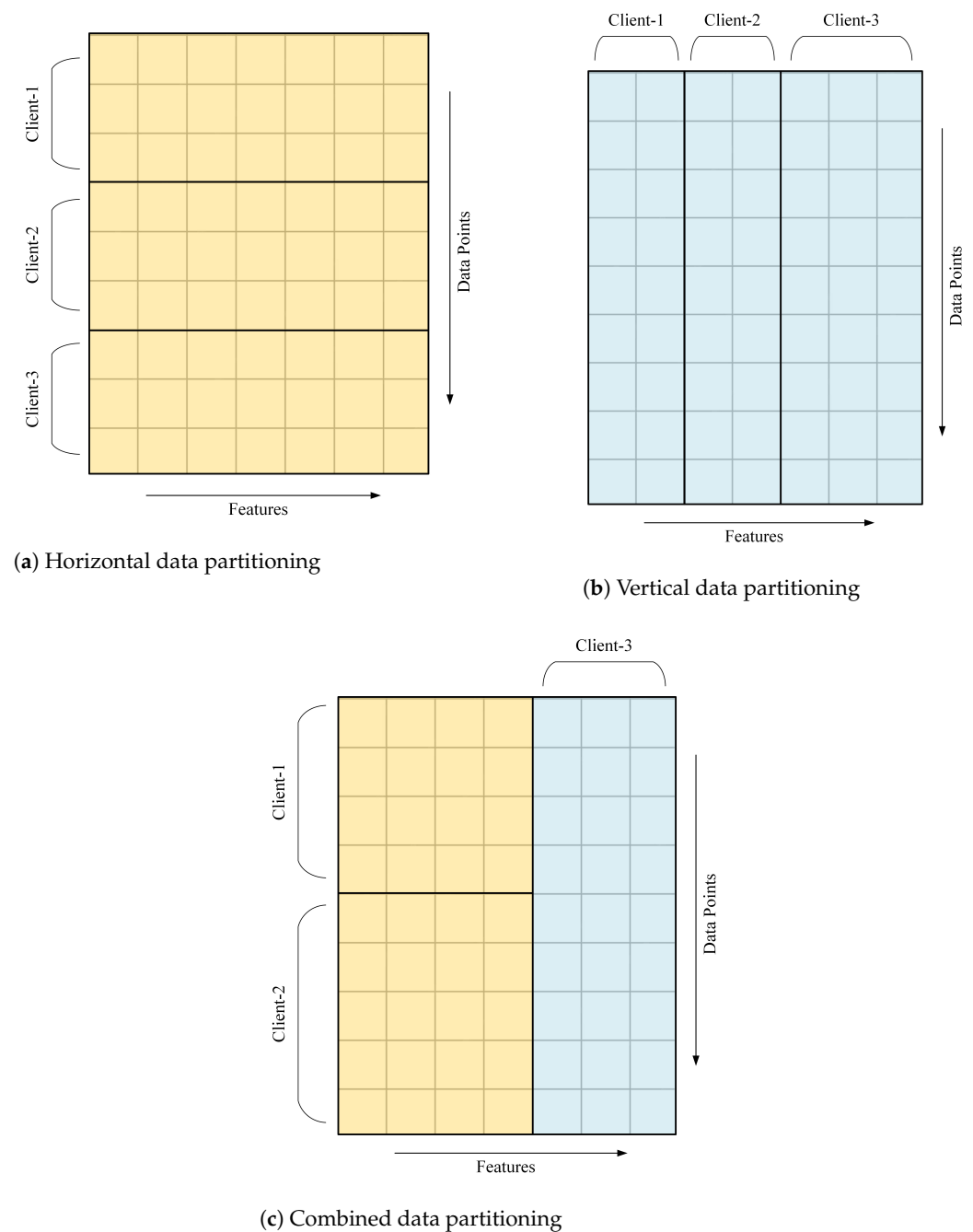


Figure 1. Overview of different data partitioning scenarios in federated learning with three clients: (a) Horizontal data partitioning; (b) Vertical data partitioning; (c) Combined data partitioning.

1.2. Contributions

The algorithms proposed in this work for VFL and CFL are specifically designed to mitigate the risks posed by GIA and enhance the privacy guarantees of these systems. The main contributions of this work are as follows:

1. A secure VFL algorithm is proposed incorporating masking techniques along with private/public key cryptography to enhance data security and authentication. This approach aims to protect sensitive data during the exchange of intermediate results between participating organisations. The masking does not affect model accuracy or the quality of aggregated results.
2. Two distinct secure CFL algorithms are proposed building on the same cryptographic and masking methods used in the VFL approach. These algorithms secure the in-

put and outcome features across all parties in combined settings ensuring privacy preservation during the collaborative model training process.

3. A key design feature of the proposed algorithms is the implementation of a client-server communication structure, where clients communicate solely through the server without any direct communication between clients. From an implementation perspective, this approach is particularly significant in real-time scenarios, where controlled communication through the server helps manage and safeguard sensitive data exchanges.

The rest of the paper is organised as follows: Section 2 presents the literature review, Section 3 explains the proposed frameworks, Section 4 presents the privacy analyses, Section 5 presents the simulation results and Section 6 concludes the paper.

2. Related Work

The increasing awareness of privacy risks in FL posed by GIA has led to the development of various defence mechanisms. This section reviews existing approaches aimed at mitigating GIA attacks with an emphasis on techniques based on differential privacy, homomorphic encryption, secure multi-party computation and masking methods due to their strong theoretical guarantees, extensive use in recent literature and relevance to the threat model addressed in this work. Though other emerging approaches exist, the mentioned techniques represent the dominant lines of defence currently explored and serve as a basis for comparing the proposed method.

Differential privacy (DP) [14] provides a formal framework for quantifying and preserving the privacy of individual data points within a dataset. In general, DP ensures that the inclusion or exclusion of a single data point does not significantly affect the output of a computation. In FL settings, DP is typically applied by adding noise to the gradients in HFL or to the intermediate results in VFL and CFL before they are sent to the server or other participants [15]. This noise addition helps mask the exact contribution of each client's data making it much harder for attacks like GIA to accurately reconstruct sensitive information. For HFL, an adaptive FL framework combining DP mechanisms has been proposed to enhance both the communication efficiency and privacy preservation of FL systems [16]. This approach integrates an adaptive learning rate strategy to improve model convergence and robustness under varying hyperparameter settings. Experimental results demonstrated that the adaptive strategy not only enhances convergence under limited communication budgets but also improves the model's resilience against noise across DP budgets. In [17], an adaptive DP mechanism is designed to address two key limitations of DP in VFL: the uniform treatment of features and privacy budgets. It dynamically allocates privacy budgets based on the impact of each organisation's features on the global model, as well as adjusting the budget throughout the training process to balance privacy and utility. Experimental results demonstrated that this method improves privacy by reducing feature inference attacks by 25% and enhances model accuracy by 15% compared to traditional budget allocation methods. DP ensures strong privacy protection but introduces a trade-off in terms of model performance. Adding noise to gradients or intermediate results can degrade accuracy and slow convergence compared to scenarios without noise. A balance between privacy and performance must be achieved as insufficient noise may expose sensitive data and excessive noise can reduce model utility.

Homomorphic encryption (HE) [18] is a form of encryption that allows computations to be performed on encrypted data without the need to decrypt it first. This property ensures that sensitive data remains private during processing. The results of computations on the encrypted data can be decrypted to reveal the final outcome; however, the data itself is never exposed in an unencrypted form. In the context of FL, HE can be used to protect the privacy of the data being processed [19]. When a client in an FL setup trains a model using its local data, it can encrypt its gradients before sending them to the server. The server can then perform necessary aggregation on the encrypted updates without accessing the actual gradients. Once the aggregation is complete, the result is sent back to the client, where it can be decrypted to obtain the final model update. A privacy-preserving FL system based on HE is proposed in [20] to protect shared model parameters. The paper introduced a method for using different HE private keys for each node within the same FL system which further enhanced security. The proposed solution's computational and communication costs are analysed, and the algorithm's performance is evaluated through simulations in cloud computing-based FL scenarios. The proposed system maintained the privacy of sensitive data by enabling encryption on the gradients. On the other hand, implementing HE comes with challenges. The encryption process can be computationally intensive and performing operations on encrypted data can be much slower compared to working with plaintext data. Therefore, HE introduces a trade-off between privacy and performance, and careful consideration of the associated computational overhead is required for its implementation.

Secure Multi-Party Computation (SMPC) [21] is a cryptographic technique that allows multiple parties to collaboratively compute a function over their inputs while keeping those inputs private. In SMPC, no party learns anything about the other parties' inputs except for the output of the computation. This is achieved through protocols (including HE) that ensure the privacy of the data and allow computations to be carried out securely despite the participation of potentially untrusted parties. In [22], SMPC-based FL is proposed to improve privacy and efficiency. The algorithm enhanced security by eliminating the need for a trusted third party and mitigating the risk of key disclosure making it well-suited for FL environments. It also addressed participant dropouts by encrypting model parameters and ensuring secure aggregation. The algorithm used compressed sensing to reduce communication and computation overhead while maintaining strong privacy protection. Experimental results showed that it outperformed existing approaches in efficiency. However, the paper acknowledged that computational efficiency can be improved and the local operations required for privacy preservation impose a significant load on participants.

Masking [23] is another technique that could be used to obscure sensitive data by replacing it with random noise or "masks" making the data unreadable to unauthorised parties. In FL, the participating clients add carefully constructed random masks to the gradients before transferring them to the server (similar to DP, where random noise is added). This addition of masks helps the FL framework resist the GIA on the server. The masks used by the clients are generated in such a way that their sum is equal to zero. Therefore, unlike DP, there is no loss of accuracy in using masking in FL. An efficient secure aggregation protocol based on masking is proposed in [24]. Besides the server, that is responsible for the aggregation of gradients, there is another third party, whose sole responsibility is to generate random masks and distribute them to the participating clients. The participating clients add these masks to the gradients before sending them to the server. The third party generates random masks in each round of training of FL which could add to the computational and communication complexity of the overall system.

Beyond the four techniques discussed above, other approaches have received growing attention in the privacy-preserving FL literature. Split learning with privacy mechanisms [25] partitions the neural network at a cut layer and protects intermediate activations using differential privacy or masking, though it does not directly address the vertical data partitioning setting. Knowledge distillation-based methods such as FedKD [26] and the complementary distillation approach for VFL proposed by Gao et al. [27] transfer knowledge via model outputs rather than gradients, offering a complementary form of privacy protection with a different threat surface. Recent advances in secure VFL include SecureVFL [28], which uses blockchain and replicated secret sharing for decentralised multi-party VFL and the end-to-end framework of Jin et al. [29], which combines multi-server SMPC with differential privacy to cover both input and output privacy.

The proposed algorithms of this work offer significant advantages over existing privacy-preserving techniques. Unlike DP, there is no loss in performance ensuring that model accuracy and convergence are maintained. In contrast to HE, the proposed approach avoids high computational complexity making it more efficient. Unlike SMPC, the clients only need to communicate directly with the server eliminating the need for complex communication channels. Finally, unlike masking, there is no requirement for a third party to generate and distribute masks simplifying the process and reducing reliance on external parties. Table 1 summarises and compares key privacy-preserving techniques in FL across multiple dimensions such as accuracy impact, computational cost, communication overhead and reliance on third parties.

The criteria underlying the qualitative ratings in Table 1 are defined as follows. Accuracy impact is rated *None* if the technique introduces no degradation relative to centralised training and *Medium* if a measurable accuracy loss is introduced; DP is rated *Medium* because noise addition to gradients or intermediate results provably degrades model utility. Computational cost is rated relative to unprotected baseline training: HE is rated *High* due to the multiplicative overhead of homomorphic operations, while masking, DP and the proposed approach are rated *Low*. Communication overhead is rated *High* for HE because encrypted ciphertexts are substantially larger than plaintext tensors; all other techniques transmit values of the same dimensionality as the unprotected baseline and are rated *Low*. Third-party required is rated *Yes* only for the masking baseline, which relies on a dedicated external dealer for mask generation and distribution each round; the proposed approach eliminates this requirement by delegating mask generation to the V-client.

Table 1. Comparison of privacy-preserving techniques in FL.

Technique	Accuracy Impact	Comp. Cost	Comm. Overhead	Third-Party Required
Differential Privacy	Medium	Low	Low	No
Homomorphic Encryption	Low	High	High	No
Masking	None	Low	Low	Yes
Proposed	None	Low	Low	No

3. Proposed Algorithms

The proposed algorithms aim to enhance privacy preservation in FL. The secure VFL algorithm builds on the foundation of previous work [13] and the secure CFL algorithms extend the ideas developed in [7]. The proposed algorithms incorporate encryption and masking techniques and eliminate the need for direct client-to-client communication enhancing both security and deployment practicality.

To assist readers unfamiliar with the terminology established in our prior work, Table 2 summarises the key terms used throughout this paper and their roles across the three proposed algorithms.

3.1. Secure Vertical Federated Learning

Figure 2 provides a visual overview of the proposed secure VFL framework. In this setup, data is vertically partitioned among multiple participating clients. Each client possesses a distinct subset of the feature space corresponding to all data points and only one designated client holds the output feature in addition to its subset of features. Clients are connected only to the central server without any direct communication among themselves.

Let $X \in \mathbb{R}^{n \times m}$ represent the input feature matrix, where m is the number of data points and n is the number of input features. There are a number of distributed participating clients: clients that only hold input features are referred to as Horizontal-clients (H-clients) and a client having the output feature is referred to as Vertical-client (V-client). Although it may not be intuitive to refer to clients holding only input features as H-clients in the vertical setting, this terminology is adopted for consistency across the paper to align with the terminology used later in the description of the combined FL algorithms.

Each H-client will have a subset of X , labeled as $X_{hj} \in \mathbb{R}^{n_{hj} \times m}$, $\forall j \in \mathcal{J} = \{1, 2, \dots, J\}$, where n_{hj} is the number of input features held by the H-client h^j and J is the total number of H-clients. Let V be the V-client, $X_v \in \mathbb{R}^{n_v \times m}$ be the input features, $y \in \mathbb{R}^{n_l \times m}$ be the output feature held at the V-client side, where n_v is the number of input features held by the V-client, n_l is the number of neurons in the last layer of the neural network ($n_l = 1$ in case of binary classification and more than 1 otherwise) and $n = n_v + [n_{h1} + n_{h2} + \dots + n_{hJ}]$.

3.1.1. Secure Channel Setup and Mask Distribution

Before the collaborative machine learning training can begin among the participating clients, the clients must first establish secure communication channels through the server. Additionally, the V-client is responsible for generating and distributing random masks to the H-clients via the server.

Let S be the server, $\text{pub}_v, \text{priv}_v$ be the public and private keys of the V-client generated via RSA algorithm [30], cert_v be the V-client's digital certificate issued by a trusted certificate authority (CA), and $K_{\text{AES},j}, \forall j \in \mathcal{J}$ the symmetric key generated by the H-client, h^j for AES encryption [31].

Table 2. Summary of terminology used across the proposed algorithms. Terminology is consistent with prior published works [7,13].

Term	Role	Rationale
Horizontal-client (H-client, h_j)	Holds a subset of input features (Secure VFL) or a subset of data points with shared input features (Secure V-OutFed, Secure H-OutFed)	Named for consistency with horizontal partitioning in the CFL setting, where H-clients hold different data points with the same feature space
Vertical-client (V-client, V)	Holds the output feature; also holds a subset of input features in all three algorithms	Named for its role in the vertical partitioning dimension, it is the client that bridges the vertical split by holding the output label
Server (S)	Aggregates intermediate representations; relays encrypted gradients and masks between clients; performs non-linear activation in Secure VFL and Secure V-OutFed	Central coordinating entity; has no access to raw client data

The V-client generates its public-private key pair $(\text{pub}_v, \text{priv}_v)$ and sends its public key and certificate cert_v to the server, which forwards them to all H-clients:

$$S \leftarrow (\text{pub}_v, \text{cert}_v) \Rightarrow h_j \leftarrow (\text{pub}_v, \text{cert}_v), \quad \forall j \in \mathcal{J}. \quad (1)$$

Each H-client h_j verifies cert_v , generates a random symmetric key $K_{\text{AES},j}$, encrypts it using pub_v and sends the encrypted key to the server for forwarding to the V-client:

$$S \leftarrow (h_j, \text{Enc}_{\text{pub}_v}(K_{\text{AES},j})), \quad \forall j \in \mathcal{J} \Rightarrow V \leftarrow (h_j, \text{Enc}_{\text{pub}_v}(K_{\text{AES},j})), \quad \forall j \in \mathcal{J}. \quad (2)$$

The V-client decrypts each encrypted key using priv_v , obtaining the symmetric key $K_{\text{AES},j}$ for secure communication with each H-client:

$$K_{\text{AES},j} = \text{Dec}_{\text{priv}_v}(\text{Enc}_{\text{pub}_v}(K_{\text{AES},j})), \quad \forall j \in \mathcal{J}. \quad (3)$$

The V-client generates $J + 1$ masks, satisfying:

$$M_V + \sum_{j=1}^J M_j = 0, \quad (4)$$

where $M_j \in \mathbb{R}^d$ is the mask for H-client h_j , $M_V \in \mathbb{R}^d$ is the mask for the V-client and d denotes the dimensionality of the gradients or intermediate information. The V-client encrypts each mask M_j using the symmetric key $K_{\text{AES},j}$:

$$C_j = \text{Enc}(K_{\text{AES},j}, M_j), \quad \forall j \in \mathcal{J} = \{1, 2, \dots, J\}, \quad (5)$$

and sends the encrypted masks to the H-clients. Each H-client decrypts its received mask:

$$M_j = \text{Dec}(K_{\text{AES},j}, C_j), \quad \forall j \in \mathcal{J} = \{1, 2, \dots, J\}. \quad (6)$$

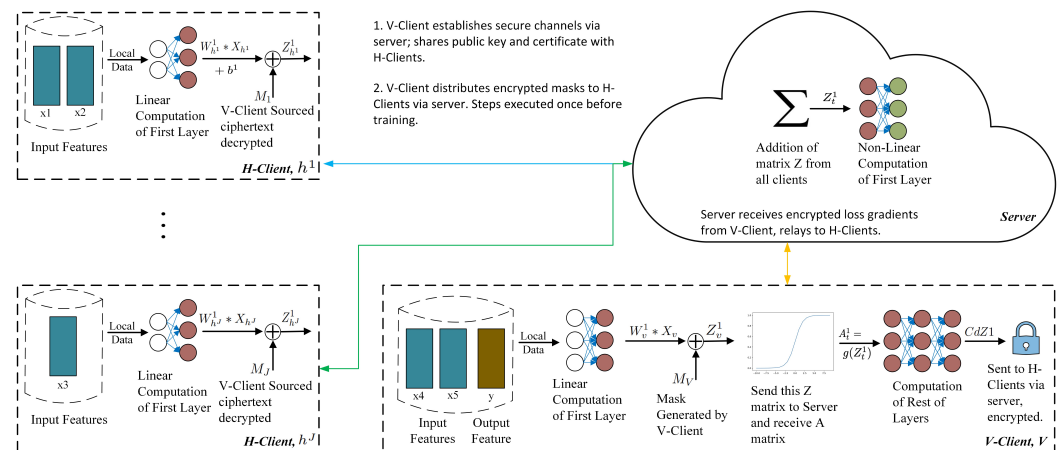


Figure 2. Encrypted collaborative training framework for neural network-based secure VFL algorithm.

3.1.2. Collaborative Neural Network Training

The machine learning model employed in this work is a neural network, which consists of interconnected layers of nodes (neurons) where each layer performs linear transformations followed by nonlinear activation functions to capture both linear and nonlinear relationships. During training, the model undergoes backpropagation, where gradients of the loss function with respect to the model parameters are computed and used to update the weights of the network.

The pseudocode for the secure VFL is presented in Algorithm 1. The V-client initiates training by computing $Z_v^1 \in \mathbb{R}^{n_1 \times m}$ based on its input features and adds its mask M_V as follows:

$$Z_v^1 = W_v^1 * X_v + M_V, \quad (7)$$

where $W_v^1 \in \mathbb{R}^{n_1 \times n_v}$ is the weight matrix for the V-client's first layer, $M_V \in \mathbb{R}^{n_1 \times m}$, n_1 is the number of neurons in the first hidden layer and $*$ represents matrix multiplication.

We now analyse the privacy implications of this design choice formally. Since the weight matrix W_v^1 is updated every round through gradient descent, the masked transmissions in two consecutive rounds r and $r - 1$ are

$$\begin{aligned} Z_{v,r}^1 &= W_{v,r}^1 X_v + M_V, \\ Z_{v,r-1}^1 &= W_{v,r-1}^1 X_v + M_V. \end{aligned}$$

An adversary observing both values can difference them to cancel the mask:

$$Z_{v,r}^1 - Z_{v,r-1}^1 = (W_{v,r}^1 - W_{v,r-1}^1) X_v = \Delta W_{v,r}^1 X_v. \quad (8)$$

The mask M_V cancels exactly, so the adversary observes $\Delta W_{v,r}^1 X_v$ rather than $W_{v,r}^1 X_v$ directly. Recovering X_v from this product remains an underdetermined problem whenever $n_1 < n_v$, as the nullspace argument of Section 4 applies equally here. Also, the nullspace argument applies directly only when the first hidden layer has fewer neurons than the number of local input features held by the target client. However, over R rounds the adversary accumulates a system of R equations:

$$\Delta W_{v,r}^1 X_v = \delta_r, \quad r = 1, \dots, R, \quad (9)$$

where $\delta_r = Z_{v,r}^1 - Z_{v,r-1}^1$ is directly observable. As R increases, the rank of the stacked coefficient matrix may grow, progressively shrinking the nullspace and making recovery of

X_v increasingly tractable. This constitutes a meaningful, round-accumulating privacy risk under mask reuse.

To mitigate this risk, we propose *periodic re-masking*: every K rounds, the V-client generates a fresh set of masks satisfying Equation (4), encrypts them with the existing symmetric keys $\{K_{\text{AES},j}\}_{j \in \mathcal{J}}$, and distributes them to the H-clients via the server following the procedure of Section 3.1.1. This resets the adversary's accumulated equation system, bounding it to at most K equations at any point during training. Masks are regenerated every $\lfloor \eta_r / K \rfloor$ rounds, where η_r is the total number of training rounds and K is the number of remaskings (recommended default: $K = 10$), ensuring the remasking schedule adapts proportionally to the length of training rather than being fixed arbitrarily. A smaller K reduces communication overhead at the cost of longer mask exposure, while a larger K provides more frequent remasking at the expense of additional communication rounds.

The V-client designates one H-client (the first H-client) to add the bias term $b^1 \in \mathbb{R}^{n_1}$. Each H-client h_j computes its own Z_{hj}^1 and adds its respective mask M_j :

$$Z_{hj}^1 = W_{hj}^1 * X_{hj} + b^1 + M_j, \quad \forall j \in \mathcal{J}, \quad b^1 = 0 \quad \text{for } j \neq 1, \quad (10)$$

where $M_j \in \mathbb{R}^{n_1 \times m}$, $\forall j \in \mathcal{J}$. The V-client and H-clients send their computed Z^1 matrices to the server, which computes the sum Z_t^1 for the first layer, $Z_t^1 = Z_v^1 + \sum_{j=1}^J Z_{hj}^1$. The server calculates the activations using ReLU [32] or sigmoid [33], and sends the results back to the V-client, which then continues forward propagation for the subsequent layers.

During backward propagation, the V-client computes gradients and updates its parameters. The V-client then encrypts the gradients $dZ^1 \in \mathbb{R}^{n_1 \times m}$, which is the partial derivative of the loss function with respect to Z_t^1 , and sends them to the H-clients through the server using symmetric encryption with keys $K_{\text{AES},j}$:

$$CdZ1_j = \text{Enc}(K_{\text{AES},j}, dZ^1), \quad \forall j \in \mathcal{J}. \quad (11)$$

The server forwards the encrypted gradients to the respective H-clients, which decrypt the matrices and use them to update their parameters:

$$dW_{hj}^1 = \frac{1}{m} (dZ^1 * X_{hj}^T), \quad W_{hj}^1 = W_{hj}^1 - \alpha_{hj} \cdot dW_{hj}^1, \quad \forall j \in \mathcal{J}, \quad (12)$$

where α_{hj} is the learning rate. The first H-client also updates the bias term:

$$db^1 = \frac{1}{m} \sum dZ^1, \quad b^1 = b^1 - \alpha_c \cdot db^1. \quad (13)$$

This process repeats for each training round until convergence, and the final model parameters are shared among all clients. The final model parameters will be the same as training when the data is centralised, given the initial weights and bias are the same in both settings.

Algorithm 1: Flow of computations in secure VFL algorithm

V-client setup the secure communication channel, generates and distribute masks to **H-clients** via **Server**
V-client initialise W_v^1
H-client 1 initialise b^1
for each *H-client* $j = \{1, 2, \dots, J\}$ **do**
 | Initialise W_{hj}^1
end
for each round $r = 1, 2, \dots$ **do**
 $Z_v^1 = W_v^1 * X_v + M_V$ # computation at V-client
 $Z_{h1}^1 = W_{h1}^1 * X_{h1} + b^1 + M_1$ # computation at H-client 1
 for each *H-client* $j = \{2, 3, \dots, J\}$ **do**
 | $Z_{hj}^1 = W_{hj}^1 * X_{hj} + M_j$
 end
 $Z_t^1 = Z_v^1 + \sum_{j=1}^J Z_{hj}^1$ # computation at Server
 Nonlinear activation on Z_t^1 # computation at Server
 Forward, backward propagation, and model updates # computation at V-client
 V-client sends $CdZ1_j$ (encrypted gradient) to each H-client
 for each *H-client* $j = 1, 2, \dots$ **do**
 | **H-client** updates its model parameters based on dZ^1 (unencrypted gradient)
 end
end

3.2. Secure Vertical Output Federated Learning

Figure 3 presents the data partitioning setup for Secure Vertical Output Federated Learning algorithm (Secure V-OutFed), the first CFL scenario. The dataset is horizontally partitioned across multiple H-clients, each holding a subset of data points with a common set of input features (x_1 and x_2 in the figure). In addition, the V-client possesses extra input features (x_3 and x_4) for all the data points across H-clients and retains the output feature. This setting differs from vertical partitioning, where each H-client holds different features for the same data points; here, the H-clients hold different data points with the same feature space. Further, the term ‘H-clients’ may not have been intuitive in the vertical partitioning scenario, it is now appropriate as these clients hold different data points but share the same feature space aligning with the horizontal partitioning concept in FL.

Let $X \in \mathbb{R}^{n \times m}$ be the data matrix and $y \in \mathbb{R}^{n_l \times m}$ be the matrix of output features as considered before. In this data partitioning scenario, each H-client holds a subset of X , i.e., $X_{hj} \in \mathbb{R}^{n_h \times m_{hj}}, \forall j \in \mathcal{J}$, where n_h is the number of input features and m_{hj} is the number of data points held by the H-client hj , such that $m = \sum_{j=1}^J m_{hj}$. Across the H-clients, the number of input features, n_h , remains the same whereas the number of data points, m_{hj} , is different. Let $X_v \in \mathbb{R}^{n_v \times m}$ be the subset of X held by the V-client, where n_v is the number of input features for the V-client, such that $n = n_h + n_v$.

The V-client will establish secure communication channels with the H-clients via the server and distribute masks using the procedures outlined in Section 3.1.1. However, there will only be two masks to begin with, generated as follows:

$$M_V + M_H = 0, \quad (14)$$

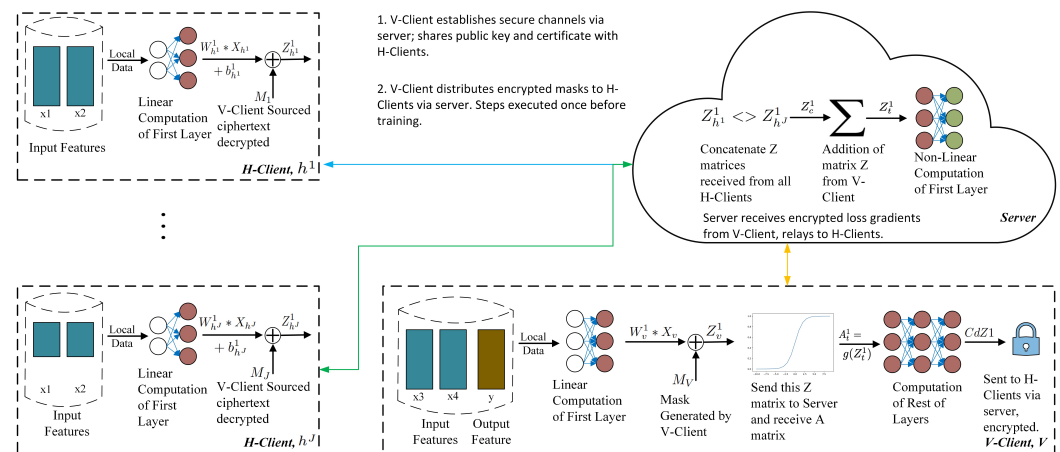


Figure 3. Secure CFL framework with encrypted collaboration between H- and V-clients for neural network training. The output feature is with the V-client.

where M_H is the mask split between all of the H-clients as follows:

$$\text{Mask for } h^j = M_{h^j}^1 \in \mathbb{R}^{n_1 \times m_{h^j}}, \forall j \in \mathcal{J}. \quad (15)$$

Collaborative Neural Network Training

Similar to the previous case, the machine learning model employed in this work is a neural network. The pseudocode for the secure V-OutFed is shown in Algorithm 2.

The V-client initiates training by computing Z_V^1 using Equation (7). At the same time, each H-client will compute the $Z_{h^j}^1$ matrix and will add its mask as follows:

$$Z_{h^j}^1 = W_{h^j}^1 * X_{h^j} + b_{h^j}^1 + M_{h^j}^1, \forall j \in \mathcal{J}. \quad (16)$$

The V-client and H-clients send their computed Z^1 matrices to the server. The server will first horizontally concatenate $Z_{h^j}^1, \forall j \in \mathcal{J}$ and then add this to Z_V^1 , giving: $Z_t^1 = [Z_{h^1}^1, Z_{h^2}^1, \dots, Z_{h^j}^1] + Z_V^1$. The server calculates the activations, and sends the results back to the V-client, which then continues forward propagation for the subsequent layers.

To update the model parameters ($W_{h^j}^1$ and $b_{h^j}^1$) for the H-clients, the V-client will send the specific parts of dZ^1 matrix (gradient), to each of the H-clients via the server in encrypted form.

$$CdZ1_j = \text{Enc}(K_{\text{AES},j}, dZ_{h^j}^1), \forall j \in \mathcal{J}. \quad (17)$$

The server forwards the encrypted gradients to the respective H-clients, which decrypt the matrices and use them to update their parameters:

$$dW_{h^j}^1 = \frac{1}{m_{h^j}} (dZ_{h^j}^1 * X_{h^j}^T), \quad W_{h^j}^1 = W_{h^j}^1 - \alpha_{h^j} \cdot dW_{h^j}^1, \forall j \in \mathcal{J}. \quad (18)$$

$$db_{h^j}^1 = \frac{1}{m_{h^j}} \sum dZ_{h^j}^1, \quad b_{h^j}^1 = b_{h^j}^1 - \alpha_c \cdot db_{h^j}^1, \forall j \in \mathcal{J}. \quad (19)$$

This process repeats for each training round until convergence, and the final model parameters are shared among all clients. As in the case of the secure VFL, the final model parameters in the secure V-OutFed will be the same as training when the data is centralised.

Algorithm 2: Flow of computations in secure V-OutFed algorithm

V-client setup the secure communication channel, generates and distribute masks to **H-clients** via **Server**

V-client initialise W_v^1

for each H-client $j = \{1, 2, \dots, J\}$ **do**

Initialise W_{hj}^1

Initialise b_{hj}^1

end

for each round $r = 1, 2, \dots$ **do**

$Z_v^1 = W_v^1 * X_v + M_v$ # computation at V-client

for each H-client $j = \{1, 2, \dots, J\}$ **do**

$Z_{hj}^1 = W_{hj}^1 * X_{hj} + b_{hj}^1 + M_j$

end

$Z_t^1 = Z_v^1 + [Z_{h1}^1, Z_{h2}^1, \dots, Z_{hJ}^1]$ # computation at Server

Nonlinear activation on Z_t^1 # computation at Server

Forward, backward propagation, and model updates # computation at V-client

for each H-client $j = 1, 2, \dots$ **do**

V-client sends CdZ_{hj}^1 (encrypted gradient) to H-client, h_j

H-client, h_j updates its model parameters based on dZ_{hj}^1 (unencrypted gradient)

end

end

3.3. Secure Horizontal Output Federated Learning Algorithm

Figure 4 presents the data partitioning setup for Secure Horizontal Output Federated Learning algorithm (Secure H-OutFed), the second CFL scenario. The data partitioning remains the same as described in Section 3.2, except that the output feature resides with the H-clients instead of the V-client; that is, each H-client holds a subset of y , i.e., $y_{hj} \in \mathbb{R}^{n_t \times m_{hj}}, \forall j \in \mathcal{J}$.

The V-client will establish secure communication channels with the H-clients via the server and distribute masks using the procedures outlined in Section 3.1.1. However, in this case, there is no need to have a mask for the V-client, the masks sum would be given as:

$$\sum_{j=1}^J M_{hj}^W = 0, \sum_{j=1}^J M_{hj}^b = 0. \quad (20)$$

Collaborative Neural Network Training

As in the previous cases, a neural network is used as the machine learning model. The pseudocode for Secure H-OutFed is provided in Algorithm 3.

The server initialises the global model (excluding the first-layer weights and bias of the V-client) and sends it to each H-client. Upon receiving the model, each H-client computes Z_{hj}^1 as $Z_{hj}^1 = W_h^1 * X_{hj}, \forall j \in \mathcal{J}$. At the same time, the V-client independently computes its contribution as $Z_v^1 = W_v^1 * X_v + b^1$. In the secure H-OutFed, the bias term is added by the V-client, unlike in secure V-OutFed where it was added by the H-clients. This design ensures privacy because here the V-client sends the Z matrices to the H-clients, whereas in secure V-OutFed, the H-clients sent the Z matrices to the V-client.

The V-client then sends the parts of encrypted Z_v^1 , i.e., $CZ_{vj}^1 \in \mathbb{R}^{n_1 \times m_{hj}}$ to the respective H-client $h_j, \forall j \in \mathcal{J}$, where CZ_{vj}^1 is given as:

$$CZ_{vj}^1 = \text{Enc}(K_{\text{AES},j}, Z_{vj}^1), \forall j \in \mathcal{J}. \quad (21)$$

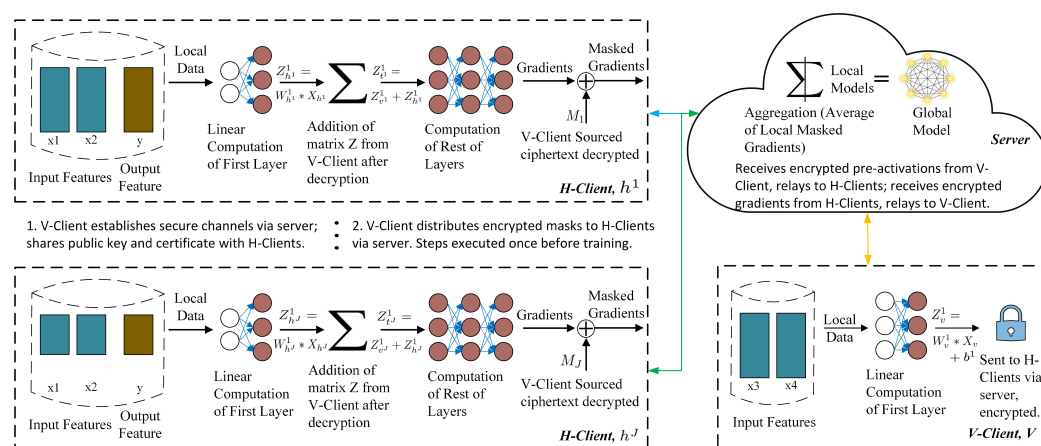


Figure 4. Secure CFL framework with encrypted collaboration between H- and V-clients for neural network training. The output feature is with the H-clients.

The H-clients decrypt and combine them with their own computations to obtain $Z_{vj}^1 = Z_{vj}^1 + Z_{hj}^1, \forall j \in \mathcal{J}$. Each H-client then applies a nonlinear activation and proceeds with local forward and backward propagation.

To update the model parameters (W_v^1 and b^1) for the V-client, each of the H-clients sends $dZ_{hj}^1 \in \mathbb{R}^{n_1 \times m_{hj}}$ (gradient) in encrypted form to the V-client. The V-client will then first decrypt this and then update its model parameters as follows:

$$dW_{vj}^1 = dZ_{hj}^1 * X_{vj}^T, W_{vj}^1 = W_{vj}^1 - \alpha_{vj} \cdot dW_{vj}^1, \forall j \in \mathcal{J}. \quad (22)$$

$$W_v^1 = \sum_{j=1}^J W_{vj}^1. \quad (23)$$

$$db_{vj}^1 = \sum_{m_{vj}} dZ_{vj}^1, b_{vj}^1 = b_{vj}^1 - \alpha_{vj} \cdot db_{vj}^1, \forall j \in \mathcal{J}. \quad (24)$$

$$b^1 = \sum_{j=1}^J b_{vj}^1. \quad (25)$$

The above completes one iteration for all H-clients and the V-client. This process repeats for several iterations, during which the H-clients train local models using their data and the V-client's features. Once trained, H-clients send their models with the addition of V-client-generated masks to the server. The server aggregates the masked models into a global model via weighted averaging, where each client's contribution is weighted by its number of data points relative to the total, as follows:

$$\tilde{W} = \sum_{j=1}^J \frac{m_{hj}}{m} \tilde{W}_{hj} = \sum_{j=1}^J \frac{m_{hj}}{m} (W_{hj} + M_{hj}^W), \quad (26)$$

$$\tilde{b} = \sum_{j=1}^J \frac{m_{hj}}{m} \tilde{b}_{hj} = \sum_{j=1}^J \frac{m_{hj}}{m} (b_{hj} + M_{hj}^b), \quad (27)$$

where m_{hj} is the number of data points at H-client h^j and $m = \sum_{j=1}^J m_{hj}$ is the total number of data points across all H-clients. In the case of equally distributed data, this reduces to simple averaging by $\frac{1}{J}$. These global model parameters are sent back to the H-clients. The training continues until the global model is optimised, and it can be shown that the global model is equivalent to one trained with the V-client's features at the H-clients.

Algorithm 3: Flow of computations in secure H-OutFed algorithm

```

V-client setup the secure communication channel, generates and distribute masks
to H-clients via Server
Server initialise model parameters except  $W_v^1$  and  $b^1$ 
V-client initialise  $W_v^1$  and  $b^1$ 
for each round  $r = 1, 2, \dots$  do
    Server sends model parameters to each H-clients
    for each iteration  $i = \{1, 2, \dots\}$  do
         $Z_v^1 = W_v^1 * X_v + b^1$  # computation at V-client
        for each H-client  $j = \{1, 2, \dots, J\}$  do
             $Z_{hj}^1 = W_h^1 * X_{hj}$ 
             $Z_{vj}^1 = Z_v^1 + Z_{hj}^1$  Forward, backward propagation, and model updates
            H-client,  $h_j$  sends  $CdZ_{vj}^1$  (encrypted gradient) to V-client
        end
        V-client updates  $W_v^1$  and  $b^1$  based on  $dZ^1$  (unencrypted gradient)
    end
    for each H-client  $j = \{1, 2, \dots, J\}$  do
        H-client,  $h_j$  applies masks to the model parameters
        H-client,  $h_j$  sends masked model parameters to server
    end
    Server aggregates masked model parameters
end

```

4. Privacy Analyses

4.1. Threat Model

We adopt the *honest-but-curious* (also referred to as semi-honest) adversarial model throughout this work. In this model, every participating party, the server or any client, follows the prescribed protocol faithfully at every step, but may attempt to infer sensitive information from all messages it legitimately receives during protocol execution.

Formally, the *view* of an honest-but-curious adversary $\mathcal{A}$ is defined as the complete transcript of values received by that party during a protocol run. In our setting:

- The **server's view** consists of the masked intermediate matrices Z_v^1 and $\{Z_{hj}^1\}_{\forall j \in \mathcal{J}}$ from all clients, and the encrypted gradient matrices $\{CdZ_{vj}^1\}_{\forall j \in \mathcal{J}}$.
- An **H-client** h_j 's **view** consists of its own local data X_{hj} , its mask M_j , and the encrypted gradient CdZ_{vj}^1 received from the V-client via the server.
- The **V-client's view** consists of its own local data X_v , the output feature y , all masks $\{M_v, M_1, \dots, M_J\}$, the symmetric keys $\{K_{AES,j}\}_{\forall j \in \mathcal{J}}$, and the non-linear activation outputs received from the server.

All privacy claims and analyses presented in this paper are explicitly bounded to this semi-honest model. The extension to a fully malicious adversarial model, where parties may deviate arbitrarily from the protocol is an important open problem discussed in Section 5.

The semi-honest assumption is well-justified in the federated settings we target. Participants such as hospitals, insurance companies and national health registries are regulated entities subject to strict legal frameworks including HIPAA [34] and GDPR [35]. In such environments, active protocol deviation carries significant legal and reputational consequences, making the assumption of correct protocol execution realistic and well-grounded.

The honest-but-curious model is furthermore the standard baseline adopted in the broader federated learning privacy literature [8] and our analysis is consistent with this convention.

The privacy analyses are conducted to demonstrate that the sharing of intermediate information between different participating parties does not leak local data nor can any party infer sensitive information from the other. The analysis assumes an honest-but-curious server (and makes the same assumption about the clients), meaning they follow the prescribed protocol but may attempt to infer sensitive information from the received messages. In this work, the honest-but-curious server or client is referred to as the adversary.

One of the intermediate pieces of information shared with other parties is the Z matrix (from clients to the server and from clients to other clients). A critical privacy concern is whether the adversary can reconstruct the private data X_v , y or X_{hj} , $\forall j \in \mathcal{J}$ from the received Z matrices. Capturing client-side private information from the observed Z matrices is mathematically infeasible.

Suppose that the adversary attempts to infer X_v from the observed Z_v^1 . The adversary would need to solve the system:

$$Z_v^1 - M_V = W_v^1 X_v. \quad (28)$$

However, the adversary has no access to either W_v^1 or M_V , both of which are randomly and independently generated on the client side. The presence of the random mask M_V , unknown to the adversary, ensures that Z_v^1 appears random and independent of the true value of $W_v^1 X_v$. Without knowledge of M_V , it is mathematically impossible to isolate $W_v^1 X_v$ from Z_v^1 , and hence to retrieve any meaningful information about X_v .

Moreover, even if the mask were absent, the system $Z_v^1 = W_v^1 X_v$ would remain underdetermined, provided that W_v^1 has fewer rows than columns (i.e., the number of neurons in the first hidden layer is less than the number of input features). Therefore, the system has infinitely many solutions for X_v , formally described by:

$$X_v = W_v^{1+}(Z_v^1) + (I - W_v^{1+}W_v^1)B, \quad (29)$$

where W_v^{1+} denotes the Moore–Penrose pseudoinverse [36] of W_v^1 and B is an arbitrary matrix of appropriate dimensions. This further demonstrates that, even in the absence of the mask, unique recovery of X_v is not possible.

The adversary can use the Moore–Penrose pseudoinverse of W_v^1 , denoted by W_v^{1+} , to obtain a least-squares solution:

$$X_v^{\text{particular}} = W_v^{1+} Z_v^1 \quad (30)$$

This is a particular solution, and it is the minimum-norm solution to the underdetermined system. However, since there are infinitely many solutions to the equation $Z_v^1 = W_v^1 X_v$, the general solution can be expressed as:

$$X_v = X_v^{\text{particular}} + X_v^{\text{nullspace}}, \quad (31)$$

where $X_v^{\text{nullspace}}$ is any solution that lies in the nullspace (kernel) of W_v^1 . In other words, it satisfies: $W_v^1 X_v^{\text{nullspace}} = 0$. The solutions in the nullspace can be written as:

$$X_v^{\text{nullspace}} = (I - W_v^{1+}W_v^1)B \quad (32)$$

Thus, the full general solution is given as Equation (29). An identical argument holds for the Z_{hj}^1 matrices sent by the H-clients. Therefore, the transmission of the Z

matrices by the clients with masking ensures that the private local data remains secure and unrecoverable by the adversary.

The other important intermediate piece of information shared with other parties is dZ^1 . It will be in encrypted form, $CdZ_j^1, \forall j \in \mathcal{J}$ sent to the server. Without encryption, the server would have access to excessive information: the Z matrices from all clients and the dZ^1 matrix. The server could then modify the dZ^1 matrix, such as setting certain columns of this matrix to zero before sending it to the other clients. These modifications could expose sensitive details which could lead to leaking local data through the Z matrices of the clients in subsequent rounds compromising privacy. The encryption of dZ^1 ensures that the server cannot manipulate or extract critical information preserving the privacy of the local data.

In the above setting, the server only receives the encrypted dZ^1 ; however, the participating clients receive dZ^1 in unencrypted form. Therefore, it is essential to analyse whether a curious client can infer another client's local data from the received dZ^1 . To infer the original local data from dZ^1 , it would require solving an underdetermined system as dZ^1 depends on both the shared representations and the unknown local model parameters. Further, since dZ^1 aggregates information across mini-batches and depends on the specific way the local model learns, it would be very difficult for a client to reverse it and recover the original data without strong prior knowledge of the target client's data distribution and model structure.

The server also sends the results of the non-linear activation to the V-client. The V-client as an adversary tries to extract the $Z_{hj}^1, \forall j \in \mathcal{J}$ matrices of the H-clients that could be used later on in conjunction with dZ^1 . The server uses ReLU activation function, defined as $\text{ReLU}(x) = \max(0, x)$, which is a one-way non-invertible function. However, this non-invertibility is partial: when $a_i > 0$, the input z_i can be exactly recovered as $z_i = a_i$, allowing a potential attacker to reconstruct those parts of the original signal. For an output $a_i = \text{ReLU}(z_i)$, if $a_i = 0$, then z_i could be any value less than or equal to zero resulting in an ambiguity and the function loses information about the negative part of the input. Formally, there exists no invertible operation $\text{ReLU}^{-1}(a_i)$ that can uniquely recover z_i from a_i , as the inverse is undefined for $a_i = 0$.

For stronger privacy, alternative activation functions such as SELU [37] may be considered as they blur information across both positive and negative ranges and do not offer exact inversion on either side. It is defined as:

$$\text{SELU}(z) = \lambda \begin{cases} z & \text{if } z > 0 \\ \alpha(e^z - 1) & \text{if } z \leq 0 \end{cases} \quad (33)$$

where $\lambda \approx 1.0507$ and $\alpha \approx 1.6733$.

One of the other privacy issues that could arise is from collusion. Collusion refers to a situation where multiple entities in an FL system (e.g., server and clients) secretly cooperate to infer private data from other parties [38]. It can significantly weaken privacy guarantees as the combined information from different parties may help uncover masked or intermediate data. Table 3 summarises the proposed algorithms' resilience to different types of collusion. For each scenario, we specify the joint view of the colluding coalition and the system of equations they can construct and whether recovery of private data is possible.

Table 3. Privacy assessment under different collusion scenarios.

Collusion Scenario	Applied Protection	Privacy Risk
Server alone tries to infer	Fully protected (mask + under-determined + encryption)	Low (Safe)
Clients alone try to infer others	Protected by independent masking and no shared info	Low (Safe)
V-client compromised	Masks are compromised	High (data privacy at risk)
All H-clients compromised	V-client's data can be inferred	High (data privacy lost)
One V-client + one H-client honest	Still protected (masks unknown, underdetermined system)	Low (Safe)

4.1.1. Scenario 1: Server Alone Attempts to Infer Private Data (Low Risk)

The server's view consists of the masked intermediate matrices $\{Z_v^1, Z_{h_1}^1, \dots, Z_{h_J}^1\}$ and the encrypted gradient matrices $\{CdZ_j^1\}_{\forall j \in \mathcal{J}}$. The server does not have access to any mask M_V or M_j , nor to any symmetric key $K_{\text{AES},j}$, nor to any weight matrix W_v^1 or $W_{h_j}^1$. To infer X_v from Z_v^1 , the server must solve:

$$Z_v^1 - M_V = W_v^1 X_v. \quad (34)$$

However, it has access to neither M_V nor W_v^1 . The presence of the unknown random mask M_V ensures that Z_v^1 is statistically independent of $W_v^1 X_v$ from the server's perspective. Even if the mask was absent, the system $Z_v^1 = W_v^1 X_v$ is underdetermined whenever $n_1 < n_v$, admitting infinitely many solutions as established in Section 4. An identical argument applies to each $Z_{h_j}^1$. Furthermore, the encrypted gradient matrices CdZ_j^1 are protected by AES encryption under keys unknown to the server, so no information about dZ^1 is accessible. The server therefore cannot reconstruct any client's private data.

4.1.2. Scenario 2: Clients Alone Attempt to Infer Each Other's Data (Low Risk)

Each H-client h_j independently holds its own local data X_{h_j} , mask M_j , symmetric key $K_{\text{AES},j}$ and the decrypted gradient dZ^1 (or $dZ_{h_j}^1$ in the combined setting). No H-client has access to another H-client's mask, weight matrix or local data. The V-client holds all masks $\{M_V, M_1, \dots, M_J\}$ and keys $\{K_{\text{AES},j}\}$, but in this scenario is assumed to be honest. For H-client h_j to infer X_{h_k} ($k \neq j$) from the received dZ^1 , it would need to solve an underdetermined system in which dZ^1 depends on the aggregated intermediate representation Z_t^1 and on the V-client's internal model parameters, neither of which is accessible to h_j . Furthermore, dZ^1 aggregates information across all clients and all data points in the mini-batch, making it infeasible to isolate the contribution of any individual client's data without knowledge of the other clients' weight matrices and local representations. Each client therefore cannot infer another client's private data when acting independently.

4.1.3. Scenario 3: V-Client Compromised (High Risk)

The V-client's view, in addition to its own local data X_v , output feature y and weight matrices includes all masks $\{M_V, M_1, \dots, M_J\}$ and all symmetric keys $\{K_{\text{AES},j}\}_{\forall j \in \mathcal{J}}$. A compromised V-client can therefore:

1. Remove masking from H-client transmissions: For each H-client h_j , the compromised V-client can subtract the known mask M_j from the observed $Z_{h_j}^1$:

$$Z_{h_j}^1 - M_j = W_{h_j}^1 X_{h_j} + b^1, \quad (35)$$

directly obtaining the weighted data representation $W_{h_j}^1 X_{h_j} + b^1$. While this does not immediately yield X_{h_j} , recovery still requires solving an underdetermined system in $W_{h_j}^1$ and X_{h_j} , both unknown to the V-client, it completely removes the masking layer of protection and exposes the H-client's data representation to any further inference attack the adversary may mount.

2. Decrypt all gradient communications: Using the known symmetric keys $\{K_{\text{AES},j}\}$, the compromised V-client can decrypt any CdZ_j^1 it has forwarded, recovering the plaintext gradient dZ^1 for all H-clients. Combined with knowledge of $Z_{h_j}^1 - M_j = W_{h_j}^1 X_{h_j} + b^1$ and the decrypted dZ^1 , the adversary has access to a richer system of equations that may further constrain the recovery of X_{h_j} , if the adversary also possesses side knowledge about the data distribution or model initialisation.

This scenario represents a fundamental trust boundary of the proposed framework: the security of all H-clients' data relies critically on the V-client remaining honest, since the V-client is the sole generator and custodian of all masks and symmetric keys. This limitation is inherent to the design choice of having the V-client manage mask distribution, which was motivated by the goal of eliminating the need for a trusted third party. One mitigation is to distribute the mask generation responsibility to a threshold of clients using a secret sharing scheme [39] so that no single compromised party can reconstruct all masks. This is identified as an important direction for future work.

4.1.4. Scenario 4: All H-Clients Compromised (High Risk)

If all J H-clients collude, their joint view includes $\{X_{h_j}, W_{h_j}^1, Z_{h_j}^1, M_j, dZ_{h_j}^1\}_{\forall j \in \mathcal{J}}$. From this joint view, the colluding coalition can compute:

$$Z_v^1 = Z_t^1 - \sum_{j=1}^J Z_{h_j}^1, \quad (36)$$

where Z_t^1 is the aggregated matrix computed and broadcast by the server. This gives the colluding coalition direct access to the V-client's masked intermediate representation $Z_v^1 = W_v^1 X_v + M_V$. To recover X_v from Z_v^1 , the coalition must additionally determine W_v^1 and M_V , neither of which is in their view. The system:

$$Z_v^1 - M_V = W_v^1 X_v \quad (37)$$

remains underdetermined in both M_V and X_v simultaneously, with infinitely many joint solutions. However, the coalition also receives dZ^1 in plaintext, which encodes the gradient of the loss with respect to Z_t^1 and thus carries information about the V-client's labels y and its internal model state. Over multiple rounds, the accumulated system of observations, Z_v^1 values and corresponding dZ^1 values, may provide sufficient constraints to mount a gradient inversion style attack [8] against the V-client's data. This scenario therefore constitutes a high privacy risk, particularly under mask reuse (see Section 3.1.2). The periodic re-masking strategy proposed in Section 3.1.2 directly mitigates this risk by limiting the number of consistent observations available to the colluding coalition.

4.1.5. Scenario 5: At Least One V-Client and One H-Client Remain Honest (Low Risk)

Suppose the colluding coalition consists of the server and all H-clients except h_1 , while the V-client and h_1 remain honest. The coalition's view includes $\{Z_{h_j}^1, M_j\}_{j \geq 2}$ and the server's aggregated outputs, but does not include M_V or M_1 . The coalition knows only the sum $M_V + M_1 = -\sum_{j=2}^J M_j$, which is computable from the masks they hold. However, knowing the sum $M_V + M_1$ does not allow the coalition to determine the individual values of M_V or M_1 : there are infinitely many pairs (M_V, M_1) satisfying the same sum, and since the masks are generated as independent random vectors, no pair is more likely than any other from the adversary's perspective. The system of equations available to the coalition therefore remains underdetermined and the privacy of both the honest V-client and the honest H-client h_1 is maintained.

5. Results and Discussions

Our simulations focus on two main evaluation aspects: (i) model performance measured by accuracy and loss across training rounds and (ii) computational time comparing scenarios with and without incorporating encryption and masking mechanisms. The simulations are conducted using the open-source FL tool, FLOWER [40], on three publicly available datasets to evaluate the proposed algorithms across diverse data characteristics: Pima Diabetes [41], Skin Segmentation [42] and Patient Care Data [43]. The Pima Diabetes dataset comprises 768 data points with 8 numerical features, where the task is to predict binary diabetes status (500 negative, 268 positive). The Skin Segmentation dataset includes 245,057 data points with 3 features to distinguish between skin and non-skin pixels (194,198 skin, 50,859 non-skin). The Patient Care Data consists of 3309 data points and 9 features representing a binary classification task between in-care and out-of-care patients (1992 in-care, 1317 out-of-care). Each dataset undergoes normalisation as a pre-processing step. The simulation code and experimental details are available on the GitHub page (<https://github.com/AustralianCancerDataNetwork/FlowerSimulations/tree/main/SecureFlowerSimulations> (accessed on 28 May 2026)).

Although all three datasets involve binary classification tasks, they differ substantially in scale, class balance and feature dimensionality, providing a meaningful range of evaluation conditions. The Pima dataset represents a small-scale, class-imbalanced medical setting (500 negative, 268 positive samples). The Patient Care dataset represents a moderate-scale clinical classification task with a more balanced class distribution (1992 vs. 1317 samples). The Skin Segmentation dataset, at 245,057 samples, is substantially larger and primarily evaluates the scalability of the proposed algorithms. Together, the three datasets span several orders of magnitude in size and provide a reasonable empirical basis for the tabular binary classification setting. All empirical claims in this paper are explicitly bounded to this setting.

Table 4 outlines the distribution of input features, data points, and output features across clients for the three proposed secure FL algorithms applied to the Pima, Patient, and Skin datasets. There is one V-client and two H-clients participated in simulations. In the Secure VFL setup, features were vertically partitioned across clients, with all data points accessible to each client. Only the V-client held the output feature and H-clients contributed complementary subsets of input features. In the Secure V-OutFed configuration, the V-client again possessed the output feature and full data access, but the H-clients held non-overlapping subsets of the data points introducing an unbalanced horizontal partitioning. Secure H-OutFed used the same data partitioning as V-OutFed; however, in this case, the output feature was distributed to the H-clients instead of the V-client. Across all datasets and algorithmic settings, the machine learning model remained consistent: a neural network with two hidden layers, each comprising five neurons and sigmoid activation functions. The training was conducted over 100 rounds using a fixed learning rate of 0.5.

Table 4. Description of data distribution between the clients across three proposed secure FL algorithms for three datasets used.

Algorithms	Dataset	V-Client	H-Client, h^1	H-Client, h^2
Secure VFL	Pima	2 input features, all data points, output feature	3 input features, all data points	3 input features, all data points
	Patient	3 input features, all data points, output feature	3 input features, all data points	3 input features, all data points
	Skin	1 input feature, all data points, output feature	1 input feature, all data points	1 input feature, all data points
Secure VOutFed	Pima	2 input features, all data points, output feature	6 input features, 300 data points	6 input features, 468 data points
	Patient	3 input features, all data points, output feature	6 input features, 1000 data points	6 input features, 2309 data points
	Skin	1 input feature, all data points, output feature	2 input feature, 10,000 data points	2 input feature, 235,057 data points
Secure HOutFed	Pima	2 input features, all data points	6 input features, 300 data points, output feature	6 input features, 468 data points, output feature
	Patient	3 input features, all data points	6 input features, 1000 data points, output feature	6 input features, 2309 data points, output feature
	Skin	1 input feature, all data points	2 input feature, 10,000 data points, output feature	2 input feature, 235,057 data points, output feature

The neural network architecture used throughout this work consists of two hidden layers with five neurons each, sigmoid activations, a learning rate of 0.5 and 100 training rounds was selected through preliminary experimentation conducted prior to the main evaluation. Several configurations were tested on all three datasets; the chosen architecture was the most conservative configuration that achieved stable convergence across all datasets and all three algorithms within a reasonable training budget. Larger architectures did not yield meaningfully different convergence behaviour on these relatively low-dimensional tabular datasets, while smaller architectures showed instability on the larger Skin Segmentation dataset. The chosen configuration therefore represents a stable, dataset-appropriate baseline for the comparative evaluation conducted in this work.

It is important to note that for Secure VFL and Secure V-OutFed, the reported performance is provably architecture-independent by construction. As established in Section 3, both algorithms produce mathematically identical outputs to centralised training given the same initialisation, regardless of the architecture used. This follows from the algebraic structure of the intermediate representation exchange: the masks cancel exactly in the aggregated Z_t^1 matrix and gradient propagation proceeds identically to the centralised case. Consequently, any architecture that converges under centralised training will yield identical results under Secure VFL and Secure V-OutFed. For Secure H-OutFed, where a small performance gap exists, the competitive results observed across three datasets spanning several orders of magnitude in size suggest that this gap is not an artefact of the specific architecture chosen. A formal architectural change for Secure H-OutFed is acknowledged as a limitation and is identified as a direction for future work.

A systematic sensitivity analysis of the training hyperparameters (learning rate, number of neurons, number of training rounds) is acknowledged as a limitation of the current

experimental evaluation. For Secure VFL and Secure V-OutFed, such an analysis is not required as these algorithms are mathematically equivalent to centralised training under any hyperparameter configuration, and sensitivity behaviour transfers directly from the centralised baseline. For Secure H-OutFed, a formal sensitivity analysis is identified as a direction for future work.

Figure 5a–f illustrate the accuracy and loss for the Pima, Skin, and Patient datasets across the proposed secure FL algorithms and a centralised baseline. The performance of Secure VFL and Secure V-OutFed exactly matches that of the centralised model, demonstrating no loss in accuracy or increase in loss despite distributed data and added privacy mechanisms. Secure H-OutFed also achieves competitive performance with slightly higher loss and marginally lower accuracy. The presented results correspond to a specific data distribution scenario as detailed in Table 4. To evaluate the robustness of the proposed algorithms, additional simulations under a range of data distribution settings were conducted in prior work [7] including independent and identically distributed (IID) and non-IID partitions. Readers are referred to that study for comprehensive analysis across these variations; results are not repeated here to avoid redundancy.

The proposed secure VFL and secure V-OutFed algorithms demonstrate practical scalability in computation and communication. The computation per H-client involves a local linear transformation of complexity $\mathcal{O}(n_1 \cdot n_{hj} \cdot m)$ for secure VFL and $\mathcal{O}(n_1 \cdot n_{hj} \cdot m_{hj})$ for secure V-OutFed. The V-client performs more intensive operations: it computes its local representation Z_1^v with cost $\mathcal{O}(n_1 \cdot n_v \cdot m)$, followed by forward propagation, loss computation, and backward propagation. If the total number of parameters in the computation layers is p , the overall cost at the V-client becomes $\mathcal{O}(n_1 \cdot n_v \cdot m + p \cdot m)$. On the server side, aggregation of masked intermediate outputs and relaying gradients involves a cost of $\mathcal{O}(n_1 \cdot m \cdot (J + 1))$, linear in the number of clients J . Communication per client per round is also linear in n_1 and m . Therefore, though the framework scales linearly with data dimensions and client count, the V-client and server may experience computational or memory bottlenecks as the network or dataset grows substantially.

The proposed secure H-OutFed algorithm also exhibits practical scalability in computation and communication. The V-client handles its input features locally and transmits masked intermediate outputs to the server. Its computation primarily consists of a local linear transformation with complexity $\mathcal{O}(n_1 \cdot n_v \cdot m)$. Each H-client undertakes more computationally intensive tasks including computing local representations at a cost of $\mathcal{O}(n_1 \cdot n_{hj} \cdot m_{hj})$, followed by forward propagation, loss evaluation and backward propagation. The overall computational complexity at an H-client is given by $\mathcal{O}(n_1 \cdot n_{hj} \cdot m_{hj} + p \cdot m_{hj})$. On the server side, aggregation of masked gradients or local model parameters received from the H-clients involves a computational cost of $\mathcal{O}(p \cdot J)$. This cost linearly increases with the number of clients and the model size. Communication per client per round is also linear in p , as each client transmits masked gradients or parameters of the same dimensionality. However, similar to the other two algorithms, the server in secure H-OutFed may face computational or memory bottlenecks for large models or high client counts.

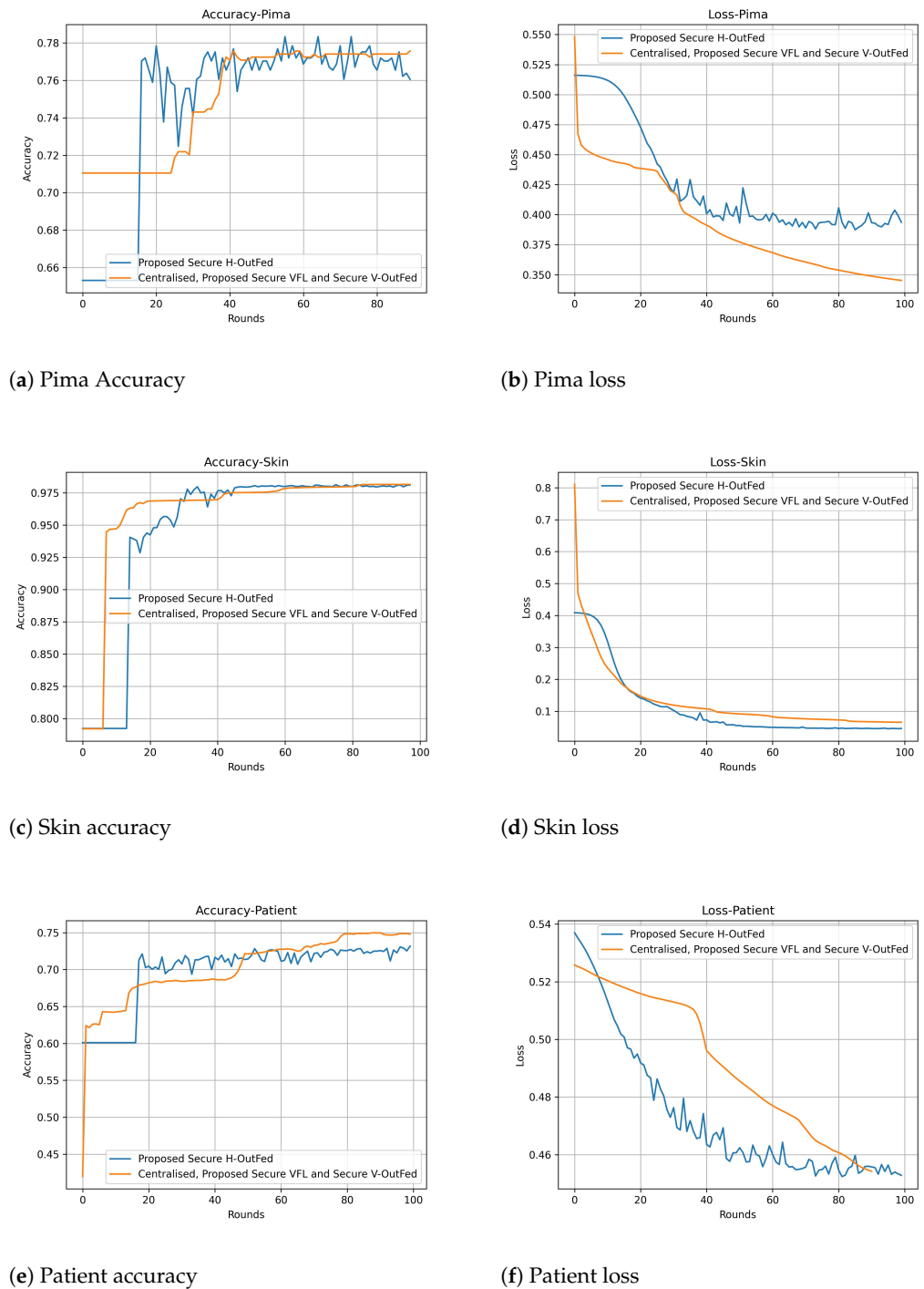


Figure 5. Accuracy and loss comparisons for the Pima, Skin and Patient datasets under Secure VFL, Secure VOutFed, Secure HOutFed and centralised training. Secure VFL and Secure VOutFed achieve performance identical to centralised training and Secure HOutFed shows competitive results with minimal degradation.

Table 5 presents the computation time (in seconds) required by the proposed secure federated learning algorithms across three datasets of varying sizes: Pima (768 data points), Patient (3309 data points) and Skin (245,057 data points). The experiments were conducted on a local Windows 11 machine equipped with a 13th Gen Intel(R) Core(TM) i7-13700H processor (2.90 GHz) and 16 GB of RAM. As expected, the computation time increases with the dataset size and the level of security applied. In the experiments reported in Table 5, the combined use of encryption and masking increased runtime by approximately 3.5%

to 43.9% relative to the unprotected baseline, depending on the dataset and algorithm. The percentage overhead was largest for the smaller Pima dataset, where fixed encryption and protocol costs form a larger share of total runtime, and smallest for the larger Skin Segmentation dataset, where the total computation is dominated by model training. For all three algorithms, the base computation time (i.e., without encryption or masking) is the lowest and the combined use of encryption and masking incurs the highest time cost. In addition, Secure H-OutFed generally exhibits slightly higher computational overhead due to its more complex interaction structure. The increase in time caused by enabling masking or encryption individually is moderate, however, their combined use results in a more noticeable overhead. Despite this, the additional computation cost remains practical even for large datasets like Skin. It is important to note that these values may exhibit minor variations across runs due to randomness in model initialisation and other system-level factors.

Though the proposed secure VFL and CFL algorithms demonstrate promising results in controlled local simulations, deploying them in real-world environments presents several additional challenges. Specifically, the VFL framework without encryption and masking has previously been implemented on the NECTAR cloud platform [44] with the server hosted on NECTAR and clients distributed across different physical locations. In such distributed deployments, communication overhead becomes significantly more pronounced compared to local machine setups. A key bottleneck lies in the transmission of intermediate representation matrix Z , whose column size scales with the number of data points. For large-scale datasets, Z could become exceptionally large, making its secure transmission over the network computationally and logistically intensive. The same issue could be seen in the proposed CFL algorithms (e.g., V-OutFed and H-OutFed), where multiple parties share and aggregate masked or encrypted outputs and gradients. One possible way to tackle this is to transmit compressed representations of the matrix Z or to aggregate partial summaries instead of full outputs.

The correctness of the proposed algorithms holds mathematically for any data type and any differentiable model architecture, since it depends only on the algebraic structure of the intermediate representation exchange and not on the specific form of the input. In this sense, the framework is architecturally general. However, extending it to imaging data in the VFL (and CFL) setting raises non-trivial design challenges. Since VFL partitions data by feature rather than by sample, applying the framework to image data would require splitting images by pixel regions across clients. While mathematically valid, this does not correspond to a realistic real-world VFL scenario, institutions holding imaging data are far more likely to possess entirely different images than disjoint pixel subsets of the same images, which is a horizontal rather than a vertical partitioning scenario. A more realistic imaging-compatible VFL scenario would involve a multimodal dataset pairing imaging features (e.g., medical scans) held by one client with tabular clinical features held by another client for the same patients. Such an extension would also require replacing the fully connected neural network with an architecture combining convolutional and fully connected layers to handle spatial structure, representing a separate model design contribution. Designing and evaluating the proposed framework in such a multimodal VFL setting is identified as an important direction for future work.

Table 5. Illustration of computation time (seconds) required for the proposed secure FL algorithms across three datasets.

Algorithms	Dataset	Without Encryption and Masking	With Masking, no Encryption	With Encryption, no Masking	With Encryption and Masking
Secure VFL	Pima	17.88	18.41	20.53	22.02
	Patient	36.73	38.20	40.12	42.49
	Skin	886.71	896.23	912.39	917.52
Secure V-OutFed	Pima	17.54	18.99	21.62	22.98
	Patient	39.41	39.94	41.61	43.50
	Skin	894.88	901.51	916.09	928.64
Secure H-OutFed	Pima	19.33	21.42	24.73	27.81
	Patient	41.11	44.62	47.18	49.43
	Skin	899.03	912.48	920.91	932.62

The proposed algorithms demonstrate strong performance in heterogeneous data settings as reflected in their accuracy and loss outcomes discussed earlier. However, practical scalability in such settings also depends on the heterogeneous computational and communication capacities of the participating entities. In secure VFL and secure V-OutFed, the V-client bears the majority of the computational burden: handling forward propagation, loss computation and backpropagation. Therefore, it is important that the V-client possesses sufficient computational power and network bandwidth to prevent bottlenecks. On the other hand, in secure H-OutFed, each H-client is actively involved in local model training and must have adequate resources to remain a feasible participant in training.

This work assumes an honest-but-curious server: one that correctly follows the protocol but may attempt to infer sensitive information. However, if the server turns out to be malicious, there are other risks to consider. A malicious server could still drop or block updates from certain clients compromising their participation. Further, it could send incorrect or inconsistent gradients/updates to different clients, leading to model divergence or performance degradation [45]. To resist the impact of a potentially malicious server, mechanisms such as verifiable computation or zero-knowledge proofs [46] can be used, which allow clients to ensure that the server is executing the intended operations without tampering with the updates. Furthermore, another strategy to handle an untrusted or potentially malicious server is to run critical computations within a Trusted Execution Environment (TEE), such as Intel SGX [47]. TEEs help maintain integrity and confidentiality even when the underlying system cannot be fully trusted.

The current work focuses on tabular data, but the same foundational idea where clients compute and share intermediate results can be extended to other data types, such as images. The underlying framework built around neural networks is also flexible enough to accommodate simpler models like logistic regression. However, there is still room to explore how well this approach generalises to other machine learning models such as decision trees or random forests, which have different computational and structural characteristics. Moreover, exploring unsupervised machine learning models could offer new opportunities and challenges for secure and collaborative model training in future work.

6. Conclusions

We proposed a set of secure federated learning algorithms for vertical and combined horizontal–vertical data partitioning scenarios. The proposed methods ensured privacy

through a combination of encryption and masking techniques, which allow clients to communicate exclusively with the server, even when indirect client-to-client communication is in place. The simulation results showed that the algorithms achieved strong performance with accuracy and loss metrics that are either identical or highly comparable to the centralised data partitioning baseline. Detailed privacy analyses confirmed that sensitive information cannot be inferred from any participating party. Although the current work focuses on tabular data with fully connected neural networks, the correctness of the proposed algorithms holds mathematically for any differentiable model architecture. Extending the framework to other data modalities, including imaging data in multimodal VFL settings is an important direction for future work, as discussed in Section 5. Future work includes exploring more diverse model types, unsupervised learning settings, and robustness against malicious server behaviour to further broaden the applicability of secure federated learning.

Author Contributions: Conceptualization, A.A. and L.H.; methodology, A.A.; software, A.A.; validation, A.A., D.M., G.B. and L.H.; formal analysis, A.A.; investigation, A.A.; resources, L.H. and G.B.; writing—original draft preparation, A.A.; writing—review and editing, A.A., D.M., G.B. and L.H.; supervision, D.M., G.B. and L.H.; project administration, L.H.; funding acquisition, L.H. and G.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research/project is supported by the Australian Research Data Commons (ARDC) 2020 Platforms program and the SWSLHD radiation oncology trust funds. The ARDC is funded by National Collaborative Research Infrastructure Strategy (NCRIS).

Data Availability Statement: The data and code used in this study are openly available in a public repository at: <https://github.com/AustralianCancerDataNetwork/FlowerSimulations/tree/main/SecureFlowerSimulations> (accessed on 1 January 2020).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; y Arcas, B.A. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the Artificial Intelligence and Statistics*; PMLR: Cambridge, MA, USA, 2017; pp. 1273–1282.
2. Almanifi, O.R.A.; Chow, C.O.; Tham, M.L.; Chuah, J.H.; Kanesan, J. Communication and computation efficiency in federated learning: A survey. *Internet Things* **2023**, *22*, 100742. [\[CrossRef\]](#)
3. Zhao, Z.; Mao, Y.; Liu, Y.; Song, L.; Ouyang, Y.; Chen, X.; Ding, W. Towards efficient communications in federated learning: A contemporary survey. *J. Frankl. Inst.* **2023**, *360*, 8669–8703. [\[CrossRef\]](#)
4. Gong, Z.; Zhong, P.; Hu, W. Diversity in machine learning. *IEEE Access* **2019**, *7*, 64323–64350. [\[CrossRef\]](#)
5. Yang, Q.; Liu, Y.; Cheng, Y.; Kang, Y.; Chen, T.; Yu, H. Horizontal federated learning. In *Federated Learning*; Springer: Cham, Switzerland, 2020; pp. 49–67.
6. Khan, A.; ten Thij, M.; Wilbik, A. Vertical federated learning: A structured literature review. *Knowl. Inf. Syst.* **2025**, *67*, 3205–3243. [\[CrossRef\]](#)
7. Anees, A.; Field, M.; Holloway, L. Development of federated learning neural networks with combined horizontal and vertical data partitioning. *Appl. Soft Comput.* **2026**, *192*, 114734. [\[CrossRef\]](#)
8. Zhu, L.; Liu, Z.; Han, S. Deep leakage from gradients. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 14774–14784.
9. Geiping, J.; Bauermeister, H.; Dröge, H.; Moeller, M. Inverting gradients-how easy is it to break privacy in federated learning? *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 16937–16947.
10. Zhu, J.; Blaschko, M. R-gap: Recursive gradient attack on privacy. *arXiv* **2020**, arXiv:2010.07733.
11. Jeon, J.; Lee, K.; Oh, S.; Ok, J. Gradient inversion with generative image prior. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 29898–29908.
12. Jiang, X.; Zhou, X.; Grossklags, J. Comprehensive analysis of privacy leakage in vertical federated learning during prediction. *Proc. Priv. Enhancing Technol.* **2022**, *2*, 263–281. [\[CrossRef\]](#)
13. Anees, A.; Field, M.; Holloway, L. A neural network-based vertical federated learning framework with server integration. *Eng. Appl. Artif. Intell.* **2024**, *138*, 109276. [\[CrossRef\]](#)
14. Zhao, Y.; Chen, J. A survey on differential privacy for unstructured data content. *ACM Comput. Surv. (CSUR)* **2022**, *54*, 207. [\[CrossRef\]](#)

15. El Ouadrhiri, A.; Abdelhadi, A. Differential privacy for deep and federated learning: A survey. *IEEE Access* **2022**, *10*, 22359–22380. [CrossRef]
16. Wu, X.; Zhang, Y.; Shi, M.; Li, P.; Li, R.; Xiong, N.N. An adaptive federated learning scheme with differential privacy preserving. *Future Gener. Comput. Syst.* **2022**, *127*, 362–372. [CrossRef]
17. Errounda, F.Z.; Liu, Y. Adaptive differential privacy in vertical federated learning for mobility forecasting. *Future Gener. Comput. Syst.* **2023**, *149*, 531–546. [CrossRef]
18. Munjal, K.; Bhatia, R. A systematic review of homomorphic encryption and its contributions in healthcare industry. *Complex Intell. Syst.* **2023**, *9*, 3759–3786. [CrossRef] [PubMed]
19. Xie, Q.; Jiang, S.; Jiang, L.; Huang, Y.; Zhao, Z.; Khan, S.; Dai, W.; Liu, Z.; Wu, K. Efficiency optimization techniques in privacy-preserving federated learning with homomorphic encryption: A brief survey. *IEEE Internet Things J.* **2024**, *11*, 24569–24580. [CrossRef]
20. Park, J.; Lim, H. Privacy-preserving federated learning using homomorphic encryption. *Appl. Sci.* **2022**, *12*, 734. [CrossRef]
21. Knott, B.; Venkataraman, S.; Hannun, A.; Sengupta, S.; Ibrahim, M.; van der Maaten, L. Crypten: Secure multi-party computation meets machine learning. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 4961–4973.
22. Chen, L.; Xiao, D.; Yu, Z.; Zhang, M. Secure and efficient federated learning via novel multi-party computation and compressed sensing. *Inf. Sci.* **2024**, *667*, 120481. [CrossRef]
23. Li, A.; Sun, J.; Zeng, X.; Zhang, M.; Li, H.; Chen, Y. Fedmask: Joint computation and communication-efficient personalized federated learning via heterogeneous masking. In *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*; Association for Computing Machinery: New York, NY, USA, 2021; pp. 42–55.
24. Stock, J.; Heitmann, H.; Schug, J.N.; Demmler, D. DealSecAgg: Efficient Dealer-Assisted Secure Aggregation for Federated Learning. In *Proceedings of the 19th International Conference on Availability, Reliability and Security*; Association for Computing Machinery: New York, NY, USA, 2024; pp. 1–11.
25. Lee, J.; Seif, M.; Cho, J.; Poor, H.V. Exploring the privacy-energy consumption tradeoff for split federated learning. *IEEE Netw.* **2024**, *38*, 388–395. [CrossRef]
26. Wu, C.; Wu, F.; Lyu, L.; Huang, Y.; Xie, X. Communication-efficient federated learning via knowledge distillation. *Nat. Commun.* **2022**, *13*, 2032. [CrossRef]
27. Gao, D.; Wan, S.; Fan, L.; Yao, X.; Yang, Q. Complementary knowledge distillation for robust and privacy-preserving model serving in vertical federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*; AAAI Press: Washington, DC, USA, 2024; Volume 38, pp. 19832–19839.
28. Fan, M.; Zhang, Z.; Li, Z.; Sun, G.; Yu, H.; Kang, J.; Guizani, M. SecureVFL: Privacy-preserving multi-party vertical federated learning based on blockchain and RSS. *Digit. Commun. Netw.* **2025**, *11*, 837–849. [CrossRef]
29. Jin, S.; Rachuri, S.R.; Wang, Y.; Nascimento, A.C.; Cai, Y. Secure and Privacy-Preserving Vertical Federated Learning. *arXiv* **2026**, arXiv:2604.13474. [CrossRef]
30. Milanov, E. The RSA algorithm. *RSA Lab.* **2009**, *1*, 1–11.
31. Daemen, J.; Rijmen, V. AES Proposal: Rijndael. 1999. Available online: https://www.google.com.hk/url?sa=t&source=web&rct=j&opi=89978449&url=https://www.cryptosoft.de/docs/Rijndael.pdf&ved=2ahUKewity8Cqq-qUAXZkq8BHYPkBVQFnoECBoQAQ&usq=AOvVaw3ky3f_rdWRFqREHesr9v-V (accessed on 28 May 2026).
32. Banerjee, C.; Mukherjee, T.; Pasilio, E., Jr. An empirical study on generalizations of the ReLU activation function. In *Proceedings of the 2019 ACM Southeast Conference*; Association for Computing Machinery: New York, NY, USA, 2019; pp. 164–167.
33. Narayan, S. The generalized sigmoid activation function: Competitive supervised learning. *Inf. Sci.* **1997**, *99*, 69–82. [CrossRef]
34. Act, A. Health insurance portability and accountability act of 1996. *Public Law* **1996**, *104*, 1–16.
35. GDPR. General Data Protection Regulation (EU) 2016/679, 2016. Available online: <https://www.legislation.gov.uk/eur/2016/679/contents> (accessed on 28 May 2026).
36. Barata, J.C.A.; Hussein, M.S. The Moore–Penrose pseudoinverse: A tutorial review of the theory. *Braz. J. Phys.* **2012**, *42*, 146–165. [CrossRef]
37. Klambauer, G.; Unterthiner, T.; Mayr, A.; Hochreiter, S. Self-normalizing neural networks. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 972–981.
38. Byrd, D.; Mugunthan, V.; Polychroniadou, A.; Balch, T. Collusion resistant federated learning with oblivious distributed differential privacy. In *Proceedings of the Third ACM International Conference on AI in Finance*; AAAI Press: Washington, DC, USA, 2022; pp. 114–122.
39. Shamir, A. How to share a secret. *Commun. ACM* **1979**, *22*, 612–613. [CrossRef]
40. Beutel, D.J.; Topal, T.; Mathur, A.; Qiu, X.; Fernandez-Marques, J.; Gao, Y.; Sani, L.; Li, K.H.; Parcollet, T.; de Gusmão, P.P.B.; et al. Flower: A friendly federated learning research framework. *arXiv* **2020**, arXiv:2007.14390.

41. Smith, J.W.; Everhart, J.E.; Dickson, W.; Knowler, W.C.; Johannes, R.S. Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In *Proceedings of the Annual Symposium on Computer Application in Medical Care*; American Medical Informatics Association: Washington, DC, USA, 1988; p. 261.
42. Eraqi, H.M.; Abouelnaga, Y.; Saad, M.H.; Moustafa, M.N. Driver distraction identification with an ensemble of convolutional neural networks. *J. Adv. Transp.* **2019**, *2019*, 4125865. [[CrossRef](#)]
43. Patel, H.J.; Saini, J.R. A Patient Care Predictive Model using Logistic Regression. *Int. J. Adv. Comput. Sci. Appl.* **2021**, *12*, 623–630. [[CrossRef](#)]
44. Gacenga, F.; An-Vo, D.A. Using ARDC Nectar VMs, Jupyter Hub and GitHub to Deploy Agricultural Modelling Applications on the AgReFed Platform. In *Proceedings of the eResearch Australasia 2023 Conference*, Brisbane, Australia, 16–20 October 2023.
45. Gouisssem, A.; Abualsaud, K.; Yaacoub, E.; Khattab, T.; Guizani, M. Collaborative byzantine resilient federated learning. *IEEE Internet Things J.* **2023**, *10*, 15887–15899. [[CrossRef](#)]
46. Ebrahimi, E.; Sober, M.; Hoang, A.T.; Ileri, C.U.; Sanders, W.; Schulte, S. Blockchain-based federated learning utilizing zero-knowledge proofs for verifiable training and aggregation. In *Proceedings of the 2024 IEEE International Conference on Blockchain (Blockchain)*; IEEE: New York, NY, USA, 2024; pp. 54–63.
47. Zheng, W.; Wu, Y.; Wu, X.; Feng, C.; Sui, Y.; Luo, X.; Zhou, Y. A survey of Intel SGX and its applications. *Front. Comput. Sci.* **2021**, *15*, 153808. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.