FATE vs. SecretFlow: A Practical Comparison for Privacy-Preserving Machine Learning

Privacy-Preserving Data Analytics

Vlad Ionita[1]
Associate Professor: Dr. Zeki Erkin[1]
Supervisor: Dr. Roland Kromes[1]
[1]EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 22, 2025

Name of the student: Vlad Ionita
Final project course: CSE3000 Research Project
Thesis committee: Dr. Zeki Erkin, Dr. Roland Kromes, Dr. Xucong Zhang

Abstract

Secure multi-party computation (SMPC) is a cryptographic technique that enables multiple parties to work together on data without sharing their private information with each other. This paper investigates how two open-source frameworks, SecretFlow and FATE, implement SMPC and other privacy-preserving techniques and evaluates their usability and scalability in applied settings. The goal is to better understand the trade-offs these frameworks offer for researchers and developers seeking to build privacy-preserving machine learning pipelines. Using a small-scale experimental setup, both frameworks were deployed in controlled environments and evaluated based on setup complexity, documentation quality, modularity, architecture, and the secure computation methods used. The results show that SecretFlow offers greater SMPC flexibility, modularity, and developer usability, making it well-suited for research and rapid prototyping. FATE, while more complex to integrate, provides comprehensive workflow coordination and is better suited for production environments. Experiments were conducted on a single physical node with a small dataset, enabling reproducibility but limiting generalizability to large-scale, real-world deployments. This work provides practical insights into the usability and architectural design of the frameworks, along with a protocol-level comparison to guide developers in selecting tools for privacy-sensitive machine learning.

1 Introduction

Due to the increase of data in the current digital age, ensuring its privacy has become critically important. With growing regulatory pressures such as the General Data Protection Regulation (GDPR) [1] and increasing awareness of data misuse, organizations are actively exploring ways to perform analytics without exposing sensitive information. Privacy-preserving data analytics aims to solve this problem by allowing valuable insights to be extracted from data without compromising individual privacy. Among the leading techniques enabling this are federated learning [2, 3] and secure multi-party computation (SMPC) [4].

These approaches have led to the development of privacy-preserving frameworks, such as FATE (Federated AI Technology Enabler) [5] and SecretFlow [6], which are designed to support data processing in sectors like healthcare [7], finance [8], and smart cities [9], where privacy concerns are particularly significant. FATE implements vertical federated learning and supports various machine learning algorithms, such as logistic regression, gradient boosting decision trees, and deep neural networks, over encrypted or partitioned data. It has been integrated into smart city infrastructures, as illustrated in FLIBD [9], where federated learning was deployed over Apache Spark for secure Internet of Things (IoT) data management. SecretFlow, in contrast, builds on both federated learning and secure computation techniques such as homomorphic encryption [10] and SMPC, offering a modular and performant architecture [11]. Despite their shared goal of enabling privacy-preserving computation, no academic studies have systematically compared these tools in terms of setup complexity, scalability, and quality of documentation.

This study focuses on comparing two open-source frameworks in the domain of privacy-preserving machine learning: SecretFlow and FATE. Both frameworks offer secure computation support for federated analytics, but differ in architecture, underlying protocols, ease of deployment, and target applications.

SecretFlow is a Python-based framework developed by Ant Group [6] that supports a modular backend architecture for secure computation. It provides multiple SMPC protocols

and is extensible to incorporate other secure computation techniques such as homomorphic encryption and differential privacy [12]. Its design emphasizes flexibility, making it particularly well-suited for research-oriented deployments and cross-domain integrations.

FATE, developed by WeBank [5], is designed for production-level industrial federated learning, with strong support for vertical federated learning (VFL). It incorporates a variety of secure computation techniques such as Paillier homomorphic encryption [13], protocols for private set intersection [14], and SecureBoost [15], a tree-boosting algorithm tailored for privacy-preserving learning. Its modular architecture allows multiple collaborative roles such as guest, host, and arbiter, making it especially suitable for applications in sectors like finance and healthcare where data privacy is important.

These two frameworks were chosen for this study because they represent contrasting design approaches: SecretFlow emphasizes modularity and extensibility, making it well-suited for academic research and experimental development, while FATE focuses on production readiness, coordination, and comprehensive workflow support for regulated environments. Comparing them provides insights into how different priorities in framework design affect usability, scalability, and the implementation of privacy-preserving protocols.

Therefore, the aim of this research is: *how do privacy-preserving machine learning frameworks such as SecretFlow and FATE implement secure computation techniques, and how do they compare in terms of ease of integration and scalability for collaborative data analysis tasks?*

To answer this, we evaluated both frameworks using the publicly available Diagnostic Wisconsin Breast Cancer dataset [16] and a standard federated logistic regression task relevant to privacy-preserving machine learning. Our assessment covers important aspects such as architectural design, usability, documentation quality, and implementation of secure computation techniques. The evaluation focuses on practical integration workflows and the extent to which each framework supports secure, modular, and developer-friendly implementation of privacy-preserving machine learning tasks.

To the best of our knowledge, this is the first systematic, protocol-level comparison of the SecretFlow and FATE frameworks, with a focus on their architecture, usability, and secure computation implementations. Our contribution lies in providing a detailed, hands-on evaluation that highlights the practical challenges and advantages of each framework in a realistic development scenario.

The results show that SecretFlow offers greater flexibility in implementing SMPC, along with improved modularity and developer usability, making it well-suited for research environments and rapid prototyping. In contrast, FATE provides more comprehensive workflow coordination and integration capabilities, making it a better fit for production-oriented deployments, although with a steeper learning curve. All experiments were conducted on a single physical node using a small dataset, which ensures reproducibility but limits the generalizability of the findings to large-scale, distributed environments. Nonetheless, this work offers practical insights and actionable comparisons to guide developers and researchers in selecting suitable frameworks for privacy-sensitive machine learning applications.

The remainder of this paper is structured as follows. We present in Section 2 the methodology used and provide background on relevant secure computation techniques. In Section 3, we analyze the secure computation approaches and architectures of the selected frameworks. We evaluate their integration complexity and scalability through a series of practical experiments in Section 4. In Section 5 we reflect on the responsible research aspects, including ethical considerations, reproducibility, and limitations. In Section 6 we discuss the implications of our findings and important observations. Finally, we conclude the paper in Section 7

and propose directions for future work.

2 Background and Methodology

In this chapter, we provide the background needed for understanding and evaluating privacy-preserving machine learning frameworks, focusing on secure multi-party computation and federated learning as core techniques that enable collaborative analytics without exposing sensitive data. Our methodology combines qualitative analysis with small-scale empirical testing to assess factors such as usability, integration complexity, and practical deployment.

2.1 Secure Computation in Collaborative Analytics

Privacy-preserving machine learning utilizes techniques such as secure multi-party computation (SMPC) [4], homomorphic encryption (HE) [10], and differential privacy (DP) [12], often combined within modern frameworks to offer layered privacy guarantees. Although all are supported by the evaluated tools, this subsection highlights SMPC and federated learning, which are central to enabling secure collaborative analytics.

In privacy-preserving collaborative data analysis, SMPC is a foundational technique that allows multiple parties to jointly compute a function over their private inputs without revealing the inputs to one another. This is particularly relevant in scenarios where regulatory constraints such as the GDPR [1] prohibit raw data sharing, but collaborative insights are needed across institutional boundaries.

SMPC achieves this goal by relying on cryptographic primitives such as secret sharing, oblivious transfer, and homomorphic encryption to protect data throughout the computation process. A common implementation of SMPC is based on Shamir's Secret Sharing [17], where a secret $s \in \mathbb{F}$ (a finite field) is split into n shares $s_1, s_2, ..., s_n$, distributed among n parties, such that any subset of size $t+1$ (the threshold) can reconstruct the secret, but any set of t or fewer shares reveals nothing about s.

Federated learning (FL) [2, 3] is another essential technique in privacy-preserving analytics. Unlike SMPC, which focuses on secure computation over shared inputs, FL enables decentralized model training by keeping the raw data local and only exchanging model parameters or updates. This approach is particularly well-suited for cross-institutional machine learning and Internet of Things (IoT) networks.

In a basic FL setting, a global model w is trained collaboratively by multiple parties (clients), each holding local datasets D_i. The training follows an iterative procedure where the server broadcasts the current global model w_t and each client i computes local model updates Δw_i using its private data:

$$\Delta w_i = \nabla \mathcal{L}(w_t, D_i), \tag{1}$$

where $\nabla \mathcal{L}(w_t, D_i)$ is the local loss function on client i's dataset. After computing the updates, the server aggregates them using weighted averaging, where each client's contribution is proportional to the size of its dataset $|D_i|$. The global model is then updated as follows [2]:

$$w_{t+1} = w_t - \eta \cdot \sum_{i=1}^{N} \frac{|D_i|}{\sum_{j=1}^{N} |D_j|} \cdot \Delta w_i. \tag{2}$$

Here, η is the learning rate, and N is the number of participating clients in that communication round. This weighted scheme ensures that clients with larger datasets have proportionally greater influence on the global model.

FL is commonly categorized into two types:

- Horizontal FL: Parties hold datasets with the same feature space but different samples. Example: Two banks in different regions with similar data but no overlapping users.
- Vertical FL: Datasets share the same sample identifiers but differ in features. Example: A hospital and an insurance company collaborating on the same patients but with different data attributes.

Both approaches often incorporate secure computation techniques such as SMPC to ensure privacy throughout the model training process.

2.2 Comparison Approach and Evaluation Methodology

The comparison in this paper combines qualitative analysis with limited empirical evaluation to assess the practical usability and technical capabilities of SecretFlow and FATE. The qualitative analysis focuses on aspects such as setup complexity, quality and clarity of documentation, architectural modularity, and the flexibility of deploying each framework in local and cloud-based environments. The secure computation mechanisms are compared based on their official documentation and the use cases they are designed to support.

For the empirical evaluation, small-scale experiments were conducted using the SURF Research Cloud [18] to simulate a controlled, single-node deployment. Both frameworks were installed and configured according to their respective setup guides, and a standard federated logistic regression task was implemented using the Diagnostic Wisconsin Breast Cancer dataset [16], which is well-suited for testing federated learning systems. Main evaluation metrics included ease of environment setup, clarity and completeness of documentation, implementation effort, and the availability of built-in secure computation components.

The outcomes of these analyses provide insights into each framework's strengths and limitations, helping to identify whether they are better suited for research prototyping or production-scale deployments.

3 Framework Analysis: Secure Computation and Architecture

In this section, we provide a comparative analysis of SecretFlow and FATE, focusing on their secure computation mechanisms and system architectures. Both frameworks aim to enable privacy-preserving machine learning (PPML) in collaborative environments, but differ in the techniques and abstractions they employ. Secure computation and system architecture are foundational to PPML frameworks, shaping not only how data confidentiality is maintained, but also how effectively the system scales and performs in practical deployments. The following comparison highlights how SecretFlow and FATE approach these core dimensions.

3.1 SecretFlow

As shown in Figure 1, the framework is structured into four distinct layers that collectively enable privacy-preserving data intelligence and machine learning:

- A top-level machine learning workflow layer that integrates various components such as data processing, distributed model training, and hyperparameter tuning.

- An algorithm layer which offers machine learning and data analysis capabilities tailored for both horizontally and vertically partitioned datasets.
- An intermediate device flow layer modeling higher-level algorithms as device object flow and directed acyclic graph (DAG).
- A foundational abstract device layer consists of plain devices and secret devices, encapsulating various cryptographic protocols to support privacy-preserving computation.

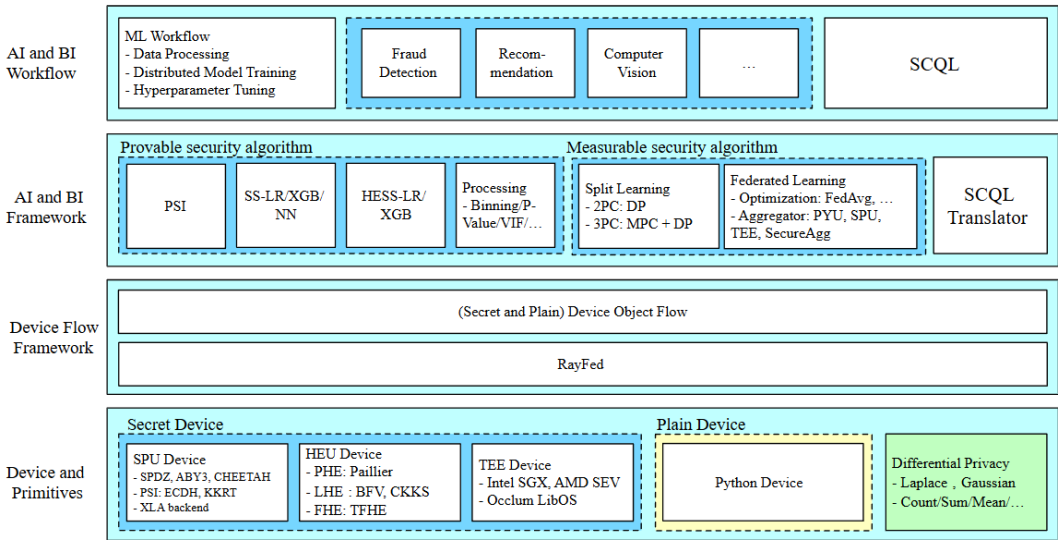

Figure 1: SecretFlow system architecture [19]

SecretFlow's Secret Device is composed of three distinct modules: the Secure Processing Unit (SPU), Homomorphic Encryption Unit (HEU), and Trusted Execution Environment (TEE) backends. Each module is responsible for specific aspects of secure computation, allowing for tailored configurations based on the requirements of a particular application.

In this research, we will focus primarily on the SMPC capabilities of SecretFlow, specifically its SPU module. According to the official documentation [20], the SPU consists of a frontend compiler and a backend runtime. The frontend transforms standard machine learning programs into an intermediate representation designed for multi-party computation. This representation is then executed by the backend, which runs the corresponding MPC protocols. Through a series of code optimizations, the SPU enables efficient execution of privacy-preserving computations. It allows machine learning models from different frameworks to be adapted with minimal changes for secure, privacy-preserving execution.

SecretFlow employs SMPC protocols to enable collaborative computations over distributed datasets without revealing individual data entries. Its primary protocol is ABY^3 [21], which supports three-party computation under the semi-honest model. ABY^3 combines arithmetic, binary, and Yao 3PC [22] sharing to enable efficient, privacy-preserving machine learning with low communication overhead and a constant number of communication rounds. It assumes that all three parties are non-colluding and is well-suited for domains like

healthcare and finance, where data privacy is critical. However, ABY^3's security guarantees are limited to semi-honest adversaries, making it less appropriate for hostile environments.

While ABY^3 is the default protocol used in SecretFlow for secure ML workflows, the framework also supports additional SMPC protocols, including Semi2k-SPDZ [23] and Cheetah [24], through its SPU backend:

- **Semi2k-SPDZ**: This protocol is a semi-honest version of SPDZ [25] that uses a trusted setup phase to generate offline randomness, often assumed to be provided by a trusted first party. As such, it is not secure in the malicious model and is mainly intended for testing and debugging. Semi2k-SPDZ retains the two-phase SPDZ structure: an offline (preprocessing) phase to generate correlated randomness, and a more efficient online phase, where actual computations occur. However, reliance on trusted randomness restricts its suitability for production environments.

- **Cheetah**: This is a high-performance 2-party computation (2PC) protocol optimized for private matrix operations, including deep learning inference. Unlike Semi2k-SPDZ, Cheetah does not require a trusted third party, instead using homomorphic encryption and packed secret sharing to reduce communication overhead. It is designed for semi-honest adversaries and demonstrates significant improvements in throughput and latency for matrix-heavy computations. Although Cheetah offers high performance, its applicability is limited to two-party scenarios, and it may not provide adequate security in adversarial environments.

These alternatives offer different trade-offs in terms of security assumptions, computational complexity, number of required parties, and performance characteristics. The choice of protocol can be tailored based on specific deployment scenarios and trust models.

SecretFlow's SPU also supports Private Set Intersection (PSI) protocols, which enable multiple parties to collaboratively identify common elements in their datasets without revealing any non-intersecting elements. SecretFlow currently implements several semi-honest PSI protocols suited for different data scales and performance needs:

- **ECDH-PSI** [26, 27]: Based on elliptic curve Diffie-Hellman key exchange, this protocol is suitable for small datasets. It encodes data points as elliptic curve elements and masks them with private keys from both parties. After a mutual exchange, one party can compute the intersection. Although simple and computationally efficient, ECDH-PSI has higher communication efficiency but is less performant on large datasets.

- **KKRT-PSI** [28]: A two-party PSI protocol based on Oblivious Transfer (OT) extensions and cuckoo hashing, KKRT-PSI is optimized for large datasets and achieves sublinear communication complexity. It is significantly faster than ECDH-based PSI but requires higher communication overhead and its security guarantees are weaker compared to protocols designed for malicious adversaries.

- **BC22PCG-PSI** [29]: This protocol utilizes pseudorandom correlation generators (PCG) and vector oblivious linear evaluation (VOLE) to enable efficient two-party PSI. It improves communication efficiency while maintaining strong privacy guarantees, making it well-suited for scenarios with strict bandwidth limitations.

As a general guideline, Oblivious Transfer-based protocols such as KKRT and BC22 offer higher computational efficiency, while ECDH-based methods are simpler and more

communication-efficient. The choice of protocol depends on the trade-offs between computation and communication overheads and the dataset size.

From an architectural perspective, the SPU abstracts its underlying secure computation protocols through a unified execution environment, enabling developers to switch between protocols based on their security and performance requirements. The SPU compiler plays a central role in this design by translating high-level machine learning operations into an intermediate representation optimized for secure multi-party computation. To reduce user learning costs and maximize compatibility, the compiler adopts XLA (Accelerated Linear Algebra) [30] as its source language, leveraging the fact that many AI frameworks like TensorFlow [31] and JAX [32] can compile Python code into XLA IR.

For platform-independent optimizations and code generation, the compiler builds on the MLIR [33] infrastructure, reusing existing optimization and lowering passes. The SPU compiler also extends the type system with security-related type hints, guiding the generation of efficient, MPC-friendly bytecode that balances privacy guarantees and performance.

3.2 Federated AI Technology Enabler (FATE)

According to FATE's documentation [34], it supports horizontal, vertical, and transfer learning federated training scenarios, allowing institutions to collaboratively train models without exchanging raw data, thus maintaining data locality and regulatory compliance.

At its core, FATE is built on FederatedML [35], a comprehensive library of federated and privacy-preserving machine learning algorithms [36]. This library is designed with a modular and decoupled architecture to ensure scalability and extensibility across various federated scenarios. The library supports several essential algorithms tailored for distributed environments while maintaining data privacy. These include Federated Logistic Regression (LR), which enables binary classification across horizontally or vertically partitioned datasets, and Federated Gradient Boosting Decision Trees (GBDT), an ensemble method effective for both classification and regression on complex data. Additionally, Federated Deep Neural Networks (DNN) support collaborative training of deep learning models for tasks such as image analysis and natural language processing, all within a privacy-preserving framework.

Beyond model training, FATE-ML also includes comprehensive modules for federated statistical analysis (e.g., PSI, union, Pearson correlation), feature engineering (e.g., feature binning, sampling, selection), and evaluation metrics for binary, multiclass, and regression tasks. These components collectively provide a full-stack solution for developing secure, federated machine learning pipelines.

The system is deployed using a client-server model, where a central coordinator orchestrates the training process across multiple parties. Each participant runs a local FATE runtime that securely executes its portion of the computation, ensuring compliance with privacy constraints. At the center of FATE's privacy-preserving capabilities is its secure computation layer, which supports multiple cryptographic protocols, such as homomorphic encryption and secure multi-party computation, each optimized for specific trust assumptions and performance needs.

As described in this study, our analysis focuses on the SMPC capabilities of FATE. This framework supports several SMPC protocols and cryptographic primitives, including:

- **SPDZ** [25]: A multi-party computation protocol that combines additive secret sharing with somewhat homomorphic encryption to enable secure arithmetic operations, even under malicious adversarial models. It offers security against against an active adversary that can corrupt up to $n-1$ of the n players. SPDZ separates computation

into an offline (preprocessing) phase, which is computationally intensive, and a more efficient online phase, where actual computations occur. SPDZ supports both arithmetic and Boolean circuits, but may incur higher communication costs compared to protocols designed for the semi-honest model.

- **Feldman Verifiable Secret Sharing (VSS)** [37]: This is a publicly verifiable extension of Shamir's secret sharing scheme that ensures correctness and reliability in the presence of dishonest parties. It operates in the information-theoretic security model and allows each participant to verify that their received share is consistent with a committed polynomial without revealing the secret. Feldman VSS tolerates up to t corruptions in a (n, t)-threshold setting and is non-interactive during verification, making it efficient for distributed systems. Its use in SMPC protocols guarantees malicious parties cannot bias computations or inject incorrect shares without detection.

- **Oblivious Transfer (OT)**: While not a complete protocol on its own, OT is a fundamental cryptographic primitive used in many SMPC protocols. FATE implements OT based on the construction by Hauck and Loss [38], which enables secure selection of inputs without revealing the receiver's choice or the sender's unchosen messages. This construction is optimized for elliptic curve cryptography (ECC), reducing interaction and computational overhead compared to traditional OT protocols and improving performance in secure computation tasks.

These components collectively enable FATE to support secure, distributed computations while offering flexibility across semi-honest and malicious threat models. Their modular implementation allows developers to choose the appropriate protocol stack based on the regulatory context, latency constraints, and security requirements.

Similarly to SecretFlow, FATE also provides built-in support for Private Set Intersection (PSI), a critical step in vertical federated learning where overlapping records must be aligned across institutions without revealing non-intersecting data. It includes an optimized PSI mechanism based on the Elliptic Curve Diffie-Hellman (ECDH) [26, 27] key exchange that utilizes Curve25519 [39], which was chosen for its strong security guarantees and efficient elliptic curve operations, offering a good trade-off between cryptographic strength and performance. This PSI mechanism is tightly integrated into FATE's data preprocessing pipeline to ensure that only mutually shared samples are retained for downstream model training, in line with privacy-by-design and data minimization principles.

FATE additionally provides a job directed acyclic graph (DAG) executor and task scheduler that breaks down federated learning workflows into directed acyclic graphs of federated operators. Each operator encapsulates a secure computation logic and executes on distributed nodes with built-in synchronization and security primitives.

To provide further insight, Figure 2 in Appendix A.2 presents a detailed comparison of the primary secure computation protocols utilized by these frameworks, highlighting their security assumptions, communication overhead, and typical use cases.

4 Integration and Scalability Evaluation

In this section, we present an integration and scalability evaluation of SecretFlow and FATE, focusing on their practical deployment within a research cloud environment and their support for multiparty learning at scale. While both frameworks aim to provide secure and flexible platforms for federated machine learning, they differ in ease of integration, architectural

complexity, and scalability strategies. Through hands-on deployment within a research cloud environment and an analysis of framework documentation, we assesses the operational readiness of each system, their support for distributed execution, and the reliability of their integration tooling.

4.1 Integration Experience

Since both SecretFlow and FATE recommend a minimum requirement of 8 cores for CPU and 16GB of RAM for memory [40, 41], they were integrated and tested within the SURF Research Cloud [18], a secure and configurable academic computing platform that provides virtual machines (VMs) for reproducible computational experiments. The SURF environment used was a desktop VM with Ubuntu 22.04, a single A10 GPU, 11 core CPU, and 88GB of RAM, which enabled controlled testing conditions for federated machine learning workflows.

An important aspect of this study was assessing how straightforward it is to integrate each framework into a real-world cloud-based research environment. Both SecretFlow and FATE provide official installation guides, Docker images, and configuration templates. However, notable differences in usability and integration effort were observed.

SecretFlow provides clear setup instructions readily available in English, making it accessible to a wide audience. It was integrated by following its starting guide [42], which first prompted us to the installation guide [40], followed by the deployment guide [43]. The framework was successfully installed by following the steps outlined in the documentation, and the sample tutorial they gave was also executed by following their instructions. The deployment guide also outlines four modes of deployment: debug, simulation, production, and production on Kuscia [44], but for this research, only the simulation mode was used. Additionally, the starting page offers a page with useful tutorials [45], as well as a user guide [46] that contains documentation about several common utility functions/classes such as preprocessing, private set intersection, MPC machine learning, and federated learning.

The official FATE GitHub repository [47] outlines two modes of deployment: standalone [48] and cluster [49]. While the standalone deployment guide is available in English, the cluster deployment documentation is primarily in Chinese, which may pose a barrier for non-Chinese-speaking users. To set up the framework, we opted to follow the quick start guide provided on the official documentation website [50], rather than the standalone guide from GitHub.

One notable difference between these two guides is that the quick start guide does not include the creation of a separate Python environment, whereas the standalone guide explicitly recommends this step. Therefore, we first created a new virtual environment as advised in the standalone guide, and then proceeded to successfully install the framework using the installation steps outlined in the quick start guide.

It is also worth mentioning that the quick start guide from the documentation website uses an older version of the FATE Client (v2.1.1), in contrast to the more recent version (v2.2.0) referenced in the GitHub standalone guide. An almost identical quick start guide was also found on FATE's GitHub repository [51], but with version 2.2.0 of the FATE Client, indicating a possible inconsistency in updates on the official documentation website. For the purposes of our experiments, we used FATE Client version 2.1.1, as specified in the quick start guide we followed.

4.2 Small-Scale Functional Tests

Following integration, we conducted small-scale functional tests within the SURF Research Cloud environment using official tutorials provided by the developers of both frameworks. We executed the documented examples to assess how each framework communicates setup procedures, configurations, and execution steps. Our aim was to validate how easily a practitioner can replicate common privacy-preserving machine learning tasks using the available documentation. Both SecretFlow and FATE were tested on a common task: federated logistic regression, a widely used method in collaborative data analysis under privacy constraints.

For SecretFlow, we followed the official tutorial provided in the documentation on mixed federated logistic regression [52]. The pipeline simulates a realistic federated learning scenario involving five virtual parties: Alice, Bob, Carol, Dave, and Eric. The Diagnostic Wisconsin Breast Cancer dataset [16] from scikit-learn was used, with all features normalized prior to training. The data was partitioned in a mixed fashion: both vertically and horizontally. Specifically, Alice retained all the label values and features 1 through 10, while Bob held features 11 through 20. Features 21 through 30 were divided among Carol, Dave, and Eric. Additionally, the dataset was horizontally split into three subsets of 200 samples each, representing separation across time periods or organizational boundaries.

The tutorial utilizes SecretFlow's MixDataFrame abstraction [53] to represent this mixed partitioning and securely aggregate updates across different feature and sample holders.

To protect data confidentiality, homomorphic encryption units (HEUs) were instantiated with 2048-bit Paillier encryption. In each HEU setup, Alice was the secret key keeper, while Bob and one of the other parties acted as evaluators. Additionally, secure aggregators were created for each horizontal partition group to perform privacy-preserving aggregation of model updates.

The model was trained using SecretFlow's FlLogisticRegressionMix [54], which combines encrypted gradient computation with secure aggregation. Training was performed over 3 epochs with a batch size of 64 and a learning rate of 0.1.

Evaluation metrics, including the area under the receiver operating characteristic curve (AUC) and accuracy, were used to assess model performance. These metrics were calculated by revealing the predicted outputs and comparing them to the ground truth labels. The model achieved a high AUC of 0.9876, indicating excellent discriminative ability, and an accuracy of 94.20%, reflecting a strong overall performance in classifying the breast cancer samples correctly.

Executing the tutorial allowed us to verify several important aspects of SecretFlow's functionality. The framework successfully simulated and orchestrated complex federated learning scenarios across multiple virtual parties, demonstrating its ability to manage both vertical and horizontal data partitioning. It also showcased seamless integration of secure computation techniques, particularly secure aggregation and homomorphic encryption, within a single workflow. Furthermore, the setup process and model execution were clear and reproducible, reflecting a high degree of usability and reliability in a controlled, simulated multi-party environment. Overall, this test served as a practical validation of SecretFlow's SMPC and federated learning capabilities in realistic data sharing scenarios.

After successfully installing the FATE framework, we proceeded by following the official quick start guide provided on their documentation website [50]. Although the guide originally demonstrates a tree-based classification model applied to a vertically partitioned version of the Diagnostic Wisconsin Breast Cancer dataset [16], we modified the experiment to use FATE's Secure Shared Heterogeneous Logistic Regression (SSHELR) component [55], in order to align the experimental setup more closely with our SecretFlow-based implemen-

tation.

During initial execution, we encountered compatibility issues related to Python package dependencies. To resolve this, we used specific versions of essential libraries: PyTorch (v2.1.0), NumPy (v1.26.4), and Transformers (v4.36.2). Once the environment was stabilized, we prepared the dataset for training by converting local CSV files into the FATE pipeline's internal dataframe format, using designated metadata to ensure proper data labeling and ID matching between the guest and host parties.

We then constructed a FATE pipeline consisting of a Reader, PSI (Private Set Intersection), SSHELR, and Evaluation component. The logistic regression model was trained using 3 epochs, a batch size of 64, and a learning rate of 0.1. Both the training and validation sets were the same and derived from the post-PSI output. After compilation and execution, the pipeline produced a trained model along with performance metrics. The model summary confirmed that the training completed all 3 epochs without converging early, and the coefficients and intercept were learned successfully.

Similarly to the SecretFlow experiment, we used the same evaluation metrics. The evaluation component reported high model performance, with an AUC of 0.9933 and a binary classification accuracy of 97.01% on both the training and validation sets. The trained model was saved and subsequently reloaded to perform predictions using a deployed inference pipeline.

The experiment validated several core strengths of the FATE framework: it effectively supported vertically partitioned data processing through its PSI and SSHELR components, offered a modular and extensible pipeline structure, and demonstrated reliable execution from training to inference. Despite some initial setup challenges, the framework proved capable of delivering accurate, privacy-preserving logistic regression results in a reproducible and transparent manner. These characteristics underscore FATE's advanced development and suitability for secure collaborative machine learning tasks in environments where data is distributed across organizational boundaries.

The source code for the experiments conducted with SecretFlow and FATE is publicly available in the following GitHub repositories: SecretFlow Repository and FATE Repository.

4.3 Scalability Analysis

To assess scalability, we reviewed the official documentation of both frameworks to understand their support for large-scale, multi-node deployments.

SecretFlow features a modular design [56] that facilitates deployment across multiple nodes, enabling users to scale federated learning workflows from local prototypes to distributed environments. Its architecture supports horizontal scaling by incorporating additional worker nodes capable of independently handling partitions of data or model computations. To reduce communication overhead and latency, SecretFlow leverages asynchronous task scheduling and optimized communication protocols. The framework is compatible with both vertical and horizontal federated learning [57], allowing flexible adaptation to mixed partitioning schemes. In addition, comprehensive benchmarks such as overall performance tests and secure gradient boosting comparisons are documented on the official website, demonstrating reasonable scalability in multi-party, large dataset scenarios [58, 59]. While public disclosures of resource usage and multi-node deployment metrics are limited, SecretFlow's modular and extensible architecture, combined with benchmark evidence, suggests its capability to effectively manage the computational and communication demands of large-scale, privacy-preserving machine learning environments.

FATE, by contrast, is explicitly designed for production-grade, large-scale federated learning. It offers two main deployment modes: standalone for local testing and cluster mode for distributed, multi-institutional environments. In cluster mode, FATE adopts a service-oriented architecture in which each party runs dedicated services such as FATE-Flow [60], FederatedML, and proxy components, all coordinated through a central registry [61]. Its distributed computing framework, EggRoll [62], handles data processing, communication, and storage across nodes. A notable advantage is that the same algorithm implementations work seamlessly in both standalone and cluster modes, allowing for straightforward scaling from development to deployment [63]. For orchestration, FATE integrates with Kubernetes [64] via KubeFATE [65], enabling dynamic resource allocation based on workload demands. While up-to-date comprehensive cluster setup guides are primarily available in Chinese and quantitative performance benchmarks are sparse, the system architecture and flexibility indicate that it is well-equipped to scale with increasing data volumes, number of participants, and computational complexity in real-world federated learning applications. This is further exemplified by its integration into smart city infrastructures, such as the FLIBD project [9], where federated learning was deployed over Apache Spark for secure Internet of Things (IoT) data management.

5 Responsible Research

In this section, we reflect on the ethical aspects and reproducibility of the research, emphasizing transparency, responsible methodology, and limitations throughout the process.

5.1 Ethical Considerations

This study adheres to the FAIR (Findable, Accessible, Interoperable, and Reusable) principles for responsible data use and research transparency. All experiments utilized the publicly available and well-documented Diagnostic Wisconsin Breast Cancer dataset [16], which meets several FAIR criteria. The dataset is assigned a persistent identifier (F1), indexed in searchable repositories (F4), and described with detailed metadata (F2, F3), facilitating its discoverability and reuse.

In terms of accessibility, the dataset can be retrieved using standard protocols without restrictions (A1, A1.1), and its metadata remains accessible even if the dataset were to be removed in the future (A2). The dataset's structure and format follow broadly recognized biomedical data conventions, supporting interoperability with machine learning tools and workflows (I1, I2). Moreover, the dataset includes clear documentation of its origin and collection process, and adheres to standard licensing for public use (R1.1, R1.2), enabling reproducibility and reuse in various research contexts (R1.3).

No personally identifiable information (PII) is included in the dataset, aligning with ethical guidelines for human subject research. Additionally, both SecretFlow and FATE are designed for privacy-preserving machine learning, supporting ethical data handling and responsible use of sensitive information. All experiments were conducted objectively, without conflict of interest, and under uniform conditions to ensure fairness and transparency.

5.2 Use of Large Language Models (LLMs)

Although this paper was not written with the use of LLMs, the Writefull plugin in Overleaf, which takes advantage of LLM technology, was used to assist in grammar correction and

spellchecking. Moreover, LLMs were used to look up explanations for unknown terminology found in the frameworks' documentation, as well as to aid in formatting and figure creation in LaTeX. A list of the prompts used can be found in Appendix A.1.

5.3 Reproducibility of Research

All experiments carried out in this study were performed according to the tutorials provided in the official documentation of SecretFlow and FATE. The source code for these experiments is publicly accessible in the following GitHub repositories: SecretFlow Repository and FATE Repository.

5.4 Research Limitations

This research faced several limitations, which are important to recognize for an accurate interpretation of the findings. Firstly, due to limited computational resources, many framework components such as the Secure Processing Unit (SPU) were not fully deployed, but instead evaluated based on technical documentation and published benchmarks. This may limit the depth and realism of our practical assessments. Moreover, the comparison reflects the state of the publicly available versions of both frameworks as of June 2025. Ongoing development may introduce new features or improvements not captured in this study, potentially affecting the relevance of our conclusions over time.

Furthermore, while both frameworks support a range of secure computation techniques, such as homomorphic encryption (HE) and differential privacy (DP), this study primarily focused on the secure multi-party computation (SMPC) mechanisms. As such, our findings may not generalize to use cases where HE or DP is more appropriate. Additionally, although the Diagnostic Wisconsin Breast Cancer dataset provides a more realistic binary classification task than simpler benchmark datasets, it remains limited in both size and feature diversity. This restricts the ability to fully assess how each framework handles large-scale, heterogeneous data in federated settings.

As a result, the conclusions drawn from this research, particularly regarding scalability, integration complexity, and real-world performance, may be biased by these constraints. Readers should interpret the findings with caution, especially when applying them to broader or more demanding deployment scenarios.

6 Discussion

In this section, we summarize the main findings of the framework comparison by examining their architectural design choices, supported secure computation protocols, and integration experiences. We reflect on how these factors impact scalability, maintainability, and usability in practical deployments. Furthermore, the discussion highlights the distinct strengths and limitations of each framework, offering insights into their suitability for different privacy-preserving machine learning applications.

6.1 Architectural and Protocol-Level Differences

SecretFlow and FATE adopt distinct architectural strategies that influence their scalability, maintainability, and extensibility for privacy-preserving machine learning. SecretFlow emphasizes modularity and developer-friendly design, exposing privacy-preserving logic

through a high-level Python API. Its Secure Processing Unit (SPU) serves as the core execution environment for multiple secure multi-party computation (SMPC) protocols, including ABY[3] and semi-honest variants optimized for performance. By decoupling cryptographic primitives from application logic, SecretFlow enables flexible experimentation and simplifies the development of secure ML pipelines.

In contrast, FATE's architecture is built around a layered service-oriented design that accommodates diverse secure computation techniques-including SMPC, homomorphic encryption (HE), and differential privacy (DP). Its SMPC capabilities are primarily based on a secret sharing scheme, with a plug-in architecture that allows users to select the protocols most aligned with their threat model or regulatory context. This makes FATE well-suited for production environments demanding flexibility and long-term compliance.

Overall, SecretFlow is optimized for rapid prototyping and research-oriented development, while FATE targets enterprise-scale federated learning deployments with more complex compliance and integration needs.

6.2 Integration Experience and Documentation Quality

During the integration process, a notable difference emerged in the quality of documentation and the onboarding experience. SecretFlow provides a well-structured and up-to-date documentation library, complete with clear tutorials, architecture diagrams, and regularly maintained examples. This significantly reduced integration time and debugging overhead.

In contrast, FATE's documentation posed recurring challenges. Although the framework is feature-rich and comprehensive, the coexistence of legacy and current documentation created confusion, particularly when older pages were not correctly linked to the latest website. Furthermore, some parts of the documentation were outdated or lacked sufficient detail, complicating the configuration of federated components and secure computation services. This hindered the early stages of experimentation and required considerable effort to resolve.

Nevertheless, both frameworks were ultimately integrated successfully, though SecretFlow's streamlined developer tools and documentation offered a more accessible experience, especially for academic or exploratory use cases.

6.3 Scalability and Practical Suitability

Scalability in both frameworks is shaped by their architectural assumptions. SecretFlow's SPU is designed for extensibility and parallel execution, but full deployment of its multi-party runtime requires substantial infrastructure that was beyond the scope of this study. However, documentation and community benchmarks suggest that it scales well with increasing data volume and parties.

FATE, on the other hand, provides comprehensive support for deployment across distributed environments and includes system components for job scheduling, service orchestration, and monitoring. Its integration into production environments in sectors like healthcare [7] and smart cities [9] indicates its suitability for large-scale, compliance-critical applications. FATE's native support for vertically partitioned data and long-term deployment scenarios reflects its advanced development stage.

These differences illustrate how architectural and operational trade-offs shape the practical suitability of each framework for specific privacy-preserving machine learning scenarios.

7 Conclusions and Future Work

This research set out to explore the question: *How do privacy-preserving machine learning frameworks such as SecretFlow and FATE implement secure computation techniques, and how do they compare in terms of ease of integration and scalability for collaborative data analysis tasks?* Through a detailed architectural and protocol-level investigation, it becomes evident that both frameworks provide reliable and effective support for secure computation, yet differ in their architectural design, developer experience, and integration reliability.

The study concludes that SecretFlow offers greater secure multi-party computation (SMPC) protocol flexibility, architectural modularity, and overall usability. It supports a wider range of SMPC protocols, along with various private set intersection (PSI) techniques. Its layered design and Python-based APIs facilitate rapid prototyping and extensibility, making it especially suitable for research and iterative development. From a usability perspective, SecretFlow stands out with well-maintained, clearly structured documentation, and practical tutorials that align with real-world applications. In contrast, FATE provides a comprehensive production-ready environment with reliable workflow coordination and lifecycle management, particularly for vertical federated learning scenarios. However, its SMPC protocol diversity is more limited, and its fragmented documentation and steeper learning curve can hinder ease of integration. Although SecretFlow is more accessible for researchers and developers, FATE's production-grade reliability may still appeal to enterprise settings with established deployment infrastructure and specialized requirements.

Both frameworks were tested in simulated federated learning scenarios on a single physical node. SecretFlow simulated five-party collaboration in a mixed federated setting, while FATE evaluated a vertical logistic regression task involving host and guest roles. These setups allowed for direct and reproducible comparison of framework behavior, but fall short of capturing the full complexity of real-world, distributed deployments. The experiments were also limited to a relatively small benchmark dataset, which, while useful for controlled evaluation, lacks the diversity and scale typically encountered in practical federated learning applications.

This study contributes a systematic comparison of the SMPC capabilities of SecretFlow and FATE, alongside a qualitative assessment of their integration experiences and documentation quality. It provides practical insight into how these frameworks perform under realistic development scenarios, offering a contextual evaluation of their architectural strengths, usability trade-offs, and suitability for privacy-preserving machine learning. Furthermore, the research includes a comparative summary of widely used secure computation protocols to support informed framework selection in privacy-sensitive applications. While the focus has been on SMPC mechanisms, the study does not cover techniques such as differential privacy or fully homomorphic encryption in depth, which presents a direction for future work.

Future work should expand the scope by evaluating both frameworks under realistic, multi-node federated deployments using large, horizontally or vertically partitioned datasets. This would enable benchmarking performance, scalability, and privacy-utility trade-offs under practical constraints. Additionally, a more thorough comparison of the full architectural stack, including homomorphic encryption modules, deployment workflows, and secure aggregation mechanisms, would offer a more comprehensive understanding of the frameworks' suitability for various collaborative learning settings. Exploring hybrid configurations that combine strengths from both frameworks may also be a promising direction.

In conclusion, both SecretFlow and FATE represent capable and evolving platforms for privacy-preserving machine learning. Each offers a unique balance between usability, flexibil-

ity, and scalability. As demand for collaborative analytics under strict privacy regulations grows, continued investment in open documentation, developer experience, and protocol extensibility will be crucial to support responsible and effective secure computation at scale.

Acknowledgments

We acknowledge the Dutch Research Council (NWO) in The Netherlands for awarding this project access to the LUMI supercomputer, owned by the EuroHPC Joint Undertaking, hosted by CSC (Finland) and the LUMI consortium through the "Computing Time on National Computer Facilities" call.

References

[1] Paul Voigt and Axel Von dem Bussche. *The EU General Data Protection Regulation (GDPR): A Practical Guide*. Springer International Publishing, 2017.

[2] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282. PMLR, 2017.

[3] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated Machine Learning: Concept and Applications. *ACM Trans. Intell. Syst. Technol.*, 10(2):12:1–12:19, 2019.

[4] Yehuda Lindell. Secure Multiparty Computation (MPC). *IACR Cryptol. ePrint Arch.*, page 300, 2020.

[5] WeBank AI Department. FATE: Federated AI Technology Enabler. `https://fate.fedai.org`, 2023. Accessed: May 2025.

[6] Ant Group. SecretFlow: A Unified Privacy-Preserving Computing Framework. `https://www.secretflow.org.cn/en/`, 2025. Accessed: May 2025.

[7] Zelei Liu, Yuanyuan Chen, Yansong Zhao, Han Yu, Yang Liu, Renyi Bao, Jinpeng Jiang, Zaiqing Nie, Qian Xu, and Qiang Yang. Contribution-Aware Federated Learning for Smart Healthcare. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, pages 12396–12404. AAAI Press, 2022.

[8] Blessing Guembe, Ambrose Azeta, Victor Osamor, and Raphael Ekpo. A Federated Machine Learning Approaches For Anti-Money Laundering Detection. In *Proceedings of the International Conference on Information Systems and Emerging Technologies (ICISET)*, 2023.

[9] Aristeidis Karras, Anastasios Giannaros, Leonidas Theodorakopoulos, George A. Krimpas, Gerasimos Kalogeratos, Christos Karras, and Spyros Sioutas. FLIBD: A Federated Learning-Based IoT Big Data Management Approach for Privacy-Preserving over Apache Spark with FATE. *Electronics*, 12(22):4633, 2023.

[10] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 169–178. ACM, 2009.

[11] Ant Group. SecretFlow: A Unified Privacy-Preserving Computing Framework. `https://github.com/secretflow/secretflow/blob/main/README.md`, 2025. Accessed: May 2025.

[12] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith. Calibrating Noise to Sensitivity in Private Data Analysis. In *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings*, volume 3876 of *Lecture Notes in Computer Science*, pages 265–284. Springer, 2006.

[13] Pascal Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 1999.

[14] WeBank AI Department. FATE Private Set Intersection. `https://fate.readthedocs.io/en/latest/2.0/fate/components/psi/`. Accessed: May 2025.

[15] Kewei Cheng, Tao Fan, Yilun Jin, Yang Liu, Tianjian Chen, Dimitrios Papadopoulos, and Qiang Yang. SecureBoost: A Lossless Federated Learning Framework. *IEEE Intell. Syst.*, 36(6):87–98, 2021.

[16] William H. Wolberg, Olvi L. Mangasarian, and W. Nick Street. Breast Cancer Wisconsin (Diagnostic). `https://doi.org/10.24432/C5DW2B`, October 1995.

[17] Adi Shamir. How to Share a Secret. *Commun. ACM*, 22(11):612–613, 1979.

[18] SURF. SURF Research Cloud. `https://www.surf.nl/en/surf-research-cloud`, 2024. Accessed: May 2025.

[19] Ant Group. SecretFlow Architecture Image. `https://github.com/secretflow/secretflow/blob/main/docs/_static/secretflow_arch.svg`, 2023. Accessed: May 2025.

[20] Junming Ma, Yancheng Zheng, Jun Feng, Derun Zhao, Haoqi Wu, Wenjing Fang, Jin Tan, Chaofan Yu, Benyu Zhang, and Lei Wang. SecretFlow-SPU: A Performant and User-Friendly Framework for Privacy-Preserving Machine Learning. In *Proceedings of the 2023 USENIX Annual Technical Conference, USENIX ATC 2023, Boston, MA, USA, July 10-12, 2023*, pages 17–33. USENIX Association, 2023.

[21] Payman Mohassel and Peter Rindal. ABY^3: A Mixed Protocol Framework for Machine Learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 35–52. ACM, 2018.

[22] Payman Mohassel, Mike Rosulek, and Ye Zhang. Fast and Secure Three-party Computation: The Garbled Circuit Approach. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, pages 591–602. ACM, 2015.

[23] Ronald Cramer, Ivan Damgård, Daniel Escudero, Peter Scholl, and Chaoping Xing. SPD$\mathbb{Z}_{2^k}$: Efficient MPC mod 2^k for Dishonest Majority. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part II*, volume 10992 of *Lecture Notes in Computer Science*, pages 769–798. Springer, 2018.

[24] Zhicong Huang, Wen-jie Lu, Cheng Hong, and Jiansheng Ding. Cheetah: Lean and Fast Secure Two-Party Deep Neural Network Inference. In *31st USENIX Security Symposium, USENIX Security 2022, Boston, MA, USA, August 10-12, 2022*, pages 809–826. USENIX Association, 2022.

[25] Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty Computation from Somewhat Homomorphic Encryption. In *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, volume 7417 of *Lecture Notes in Computer Science*, pages 643–662. Springer, 2012.

[26] Catherine Meadows. A More Efficient Cryptographic Matchmaking Protocol for Use in the Absence of a Continuously Available Third Party. In *Proceedings of the 1986 IEEE Symposium on Security and Privacy, Oakland, California, USA, April 7-9, 1986*, pages 134–137. IEEE Computer Society, 1986.

[27] Bernardo A. Huberman, Matthew K. Franklin, and Tad Hogg. Enhancing privacy and trust in electronic communities. In *Proceedings of the First ACM Conference on Electronic Commerce (EC-99), Denver, CO, USA, November 3-5, 1999*, pages 78–86. ACM, 1999.

[28] Vladimir Kolesnikov, Ranjit Kumaresan, Mike Rosulek, and Ni Trieu. Efficient Batched Oblivious PRF with Applications to Private Set Intersection. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 818–829. ACM, 2016.

[29] Dung Bui and Geoffroy Couteau. Improved Private Set Intersection for Sets with Small Entries. In *Public-Key Cryptography - PKC 2023 - 26th IACR International Conference on Practice and Theory of Public-Key Cryptography, Atlanta, GA, USA, May 7-10, 2023, Proceedings, Part II*, volume 13941 of *Lecture Notes in Computer Science*, pages 190–220. Springer, 2023.

[30] OpenXLA. XLA Architecture. `https://openxla.org/xla/architecture`, 2024. Accessed: May 2025.

[31] Martín Abadi, Ashish Agarwal, Paul Barham, and et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. `https://www.tensorflow.org/`, 2015. Accessed: May 2025.

[32] The JAX Authors. JAX: High Performance Array Computing. `https://docs.jax.dev/en/latest/index.html`, 2024. Accessed: May 2025.

[33] The MLIR Community. MLIR: Multi-Level Intermediate Representation. `https://mlir.llvm.org/`, 2020. Accessed: May 2025.

[34] WeBank AI Department. FATE README. https://fate.readthedocs.io/en/develop/_build_temp/README.html, 2025. Accessed: June 2025.

[35] WeBank AI Department. FATE-ML. https://fate.readthedocs.io/en/latest/2.0/fate/components/, 2025. Accessed: June 2025.

[36] Yang Liu, Tao Fan, Tianjian Chen, Qian Xu, and Qiang Yang. FATE: An Industrial Grade Platform for Collaborative Learning With Data Protection. *J. Mach. Learn. Res.*, 22:226:1–226:6, 2021.

[37] Paul Feldman. A Practical Scheme for Non-interactive Verifiable Secret Sharing. In *28th Annual Symposium on Foundations of Computer Science, Los Angeles, California, USA, 27-29 October 1987*, pages 427–437. IEEE Computer Society, 1987.

[38] Eduard Hauck and Julian Loss. Efficient and Universally Composable Protocols for Oblivious Transfer from the CDH Assumption. *IACR Cryptol. ePrint Arch.*, page 1011, 2017.

[39] Daniel J. Bernstein. Curve25519: New Diffie-Hellman Speed Records. In *Public Key Cryptography - PKC 2006, 9th International Conference on Theory and Practice of Public-Key Cryptography, New York, NY, USA, April 24-26, 2006, Proceedings*, volume 3958 of *Lecture Notes in Computer Science*, pages 207–228. Springer, 2006.

[40] Ant Group. SecretFlow Installation Guide. https://www.secretflow.org.cn/en/docs/secretflow/v1.12.0b0/getting_started/installation#installation, 2025. Accessed: June 2025.

[41] WeBank AI Department. FATE Single-Node Deployment Guide. https://github.com/FederatedAI/FATE/tree/master/deploy/standalone-deploy, 2024. Accessed: May 2025.

[42] Ant Group. SecretFlow Getting Started Guide. https://www.secretflow.org.cn/en/docs/secretflow/v1.12.0b0/getting_started, 2024. Accessed: June 2025.

[43] Ant Group. SecretFlow Deployment Guide. https://www.secretflow.org.cn/en/docs/secretflow/v1.12.0b0/getting_started/deployment#deployment, 2025. Accessed: June 2025.

[44] Ant Group. Kuscia Architecture. https://www.secretflow.org.cn/en/docs/kuscia/main/reference/architecture_cn, 2025. Accessed: June 2025.

[45] Ant Group. SecretFlow Tutorials. https://www.secretflow.org.cn/en/docs/secretflow/v1.12.0b0/tutorial#tutorial, 2024. Accessed: June 2025.

[46] Ant Group. SecretFlow User Guide. https://www.secretflow.org.cn/en/docs/secretflow/v1.12.0b0/user_guide#user-guide, 2024. Accessed: June 2025.

[47] WeBank AI Department. FATE GitHub Repository. https://github.com/FederatedAI/FATE, 2024. Accessed: May 2025.

[48] WeBank AI Department. FATE Single-Node Deployment Guide. https://github.com/FederatedAI/FATE/tree/master/deploy/standalone-deploy, 2024. Accessed: June 2025.

[49] WeBank AI Department. FATE Cluster Deployment Deployment Guide. https://github.com/FederatedAI/FATE?tab=readme-ov-file#cluster-deployment, 2024. Accessed: June 2025.

[50] WeBank AI Department. FATE Quick Start Guide. https://fate.readthedocs.io/en/latest/2.0/fate/quick_start/, 2020. Accessed: June 2025.

[51] WeBank AI Department. FATE Quick Start GitHub Guide. https://github.com/FederatedAI/FATE/blob/master/doc/2.0/fate/quick_start.md, 2024. Accessed: June 2025.

[52] Ant Group. SecretFlow Mix Federated Learning - Logistic Regression. https://www.secretflow.org.cn/en/docs/secretflow/v1.12.0b0/tutorial/mix_lr, 2023. Accessed: June 2025.

[53] Ant Group. SecretFlow DataFrame. https://www.secretflow.org.cn/en/docs/secretflow/v1.12.0b0/user_guide/preprocessing/DataFrame, 2023. Accessed: June 2025.

[54] Ant Group. SecretFlow Federated Learning Logistic Regression Mix. https://www.secretflow.org.cn/en/docs/secretflow/v1.12.0b0/source/secretflow.ml.linear#secretflow.ml.linear.FlLogisticRegressionMix, 2022. Accessed: June 2025.

[55] WeBank AI Department. FATE Heterogeneous SSHE Logistic Regression. https://fate.readthedocs.io/en/latest/2.0/fate/components/logistic_regression/#heterogeneous-sshe-logistic-regression, 2020. Accessed: June 2025.

[56] Ant Group. SecretFlow Architecture. https://www.secretflow.org.cn/en/docs/secretflow/v1.12.0b0/developer/design/architecture, 2023. Accessed: May 2025.

[57] Ant Group. SecretFlow Federated Learning. https://www.secretflow.org.cn/en/docs/secretflow/v1.12.0b0/user_guide/federated_learning, 2024. Accessed: May 2025.

[58] Ant Group. SecretFlow Benchmark Results. https://www.secretflow.org.cn/en/docs/secretflow/v1.12.0b0/developer/benchmark/overall_benchmark, 2024. Accessed: June 2025.

[59] Ant Group. Performance Analysis: SecureBoost vs XGBoost. https://www.secretflow.org.cn/en/docs/secretflow/v1.12.0b0/developer/benchmark/sgb_benchmark, 2024. Accessed: June 2025.

[60] WeBank AI Department. FATE-Flow GitHub Repository. https://github.com/FederatedAI/FATE-Flow, 2024. Accessed: June 2025.

[61] WeBank AI Department. FATE Fate Cluster Deployment Guide (Old). https://fate.readthedocs.io/en/develop-1.6/_build_temp/cluster-deploy/README.html, 2020. Accessed: June 2025.

[62] WeBank AI Department. EggRoll GitHub Repository. https://github.com/FederatedAI/eggroll, 2024. Accessed: June 2025.

[63] WeBank AI Department. Federated Learning Algorithms In FATE (Old). https://fate.readthedocs.io/en/develop/_build_temp/README.html, 2020. Accessed: June 2025.

[64] The Kubernetes Authors. Kubernetes: Production-Grade Container Orchestration. https://kubernetes.io/, 2025. Accessed: June 2025.

[65] WeBank AI Department. KubeFATE GitHub Repository. https://github.com/FederatedAI/KubeFATE, 2024. Accessed: June 2025.

A Appendix

A.1 LLM Queries

The following LLM prompts were used to format text and create figures in LaTeX:

- "How can I reference a section in LaTeX?"
- "Generate a figure in LaTeX."
- "How to add a link in LaTeX?"
- "How can I rotate a figure 90 degrees clockwise in LaTeX?"
- "How can I reference a figure in LaTeX?"

A.2 Protocol Comparison Table

Protocol	Security Mode	Num. Parties	Computation Complexity	Communication Overhead	Use Case	Drawbacks
ABY3	Semi Honest	3	Moderate: bit-level operations, fixed-point arithmetic	Moderate bandwidth due to replicated secret sharing	Training and inference in ML, efficient fixed-point arithmetic	Limited to 3 parties, fixed trust assumption
Semi2k SPDZ	Semi Honest	2+	Moderate - High: expensive MAC generation, requires offline preprocessing	High bandwidth due to pre-shared randomness and authentication tags	General-purpose MPC, supports active security in SPDZ variant	Communication-heavy, requires pre-processing and MAC generation
Cheetah	Semi Honest	2	Low: SIMD-optimized, minimal interactive operations	Very low, batch-friendly vectorized design	Efficient 2-party ML inference	Optimized for 2-party use only, not as flexible for n-party setups
SPDZ	Malicious	2+	High: expensive FHE-based preprocessing, complex arithmetic	Very high, significant bandwidth for MACs and ciphertexts	General-purpose secure computation under strong adversaries	Preprocessing-heavy, slower runtime, high bandwidth needs
Feldman VSS	Semi Honest	n	Low: polynomial interpolation/evaluation	Low, public verifiability adds small overhead	Secret sharing schemes, threshold cryptography	Public verifiability but no hiding of shares from dealer
ECDH PSI	Semi Honest	2	Low: scalar EC operations, linear with input size	Low due to lightweight cryptographic primitives	Two-party PSI such as user ID matching	Relies on elliptic curve assumptions, not optimized for large-scale datasets
KKRT PSI	Semi Honest	2	Low: OT extension and hashing, efficient for large inputs	Low - moderate depending on input set size	High-speed PSI on medium to to large datasets	Performance may degrade with very large input sets
BC22PCG PSI	Malicious	2	Low - Moderate: pseudorandom code-based, higher per-element cost	Moderate, includes additional checks for malicious behavior	High-security PSI with better fault tolerance	More complex implementation, slightly higher overhead than OT-based PSI

Figure 2: Comparison table of secure computation protocols