Starlit: Privacy-Preserving Federated Learning to Enhance Financial Fraud Detection[⋆]

Aydin Abadi[⋆⋆][1] Bradley Doyle[2] Francesco Gini[2] Kieron Guinamard[2] Sasi Kumar Murakonda[⋆⋆⋆][2]
Jack Liddell[2] Paul Mellor[2] Steven J. Murdoch[†][1] Mohammad Naseri[‡][1,3] Hector Page[2]
George Theodorakopoulos[§][4] Suzanne Weller[¶][2]

[1] University College London
[2] Privitar
[3] Flower Labs
[4] Cardiff University

Abstract. Federated Learning (FL) is a data-minimization approach enabling collaborative model training across diverse clients with local data, avoiding direct data exchange. However, state-of-the-art FL solutions to identify fraudulent financial transactions exhibit a subset of the following limitations. They (1) lack a formal security definition and proof, (2) assume prior freezing of suspicious customers' accounts by financial institutions (limiting the solutions' adoption), (3) scale poorly, involving either $O(n^2)$ computationally expensive modular exponentiation (where n is the total number of financial institutions) or highly inefficient fully homomorphic encryption, (4) assume the parties have already completed the identity alignment phase, hence excluding it from the implementation, performance evaluation, and security analysis, and (5) struggle to resist clients' dropouts. This work introduces *Starlit*, a novel *scalable* privacy-preserving FL mechanism that overcomes these limitations. It has various applications, such as enhancing financial fraud detection, mitigating terrorism, and enhancing digital health. We implemented *Starlit* and conducted a thorough performance analysis using synthetic data from a key player in global financial transactions. The evaluation indicates *Starlit*'s scalability, efficiency, and accuracy.

1 Introduction

Sharing data is crucial in dealing with crime. Collaborative data analysis among law enforcement agencies and relevant stakeholders can significantly enhance crime prevention, investigation, and overall public safety. For instance, in the United Kingdom, Cifas, a non-profit fraud database, and fraud prevention organization that promotes data sharing among its members, reported that its members detected and reported over 350,000 cases of fraud in 2019. This collective effort prevented fraudulent activities amounting to £1.5 billion [66]. The National Data Sharing Guidance, developed by the UK Home Office and Ministry of Justice in 2023, further underscores the importance of data sharing in dealing with crime [63].

Typically, inputs for collaborative data analysis come from different parties, each of which may have concerns about the privacy of their data. Federated Learning (FL) [73] and secure Multi-party Computation (MPC) [75], along with their combination, are examples of mechanisms that allow parties to collaboratively analyze shared data while maintaining the privacy of their input data.

FL is a machine learning framework where multiple parties collaboratively build machine learning models without revealing their sensitive input to their counterparts [73,42]. Vertical Federated Learning (VFL) is a

[⋆] Our solution, *Starlit*, has been awarded the joint first prize of UK–US Privacy-Enhancing Technologies Challenge Prize [58]. *Starlit* has been acknowledged by both the White House and UK Government websites [60,59].

[⋆⋆] aydin.abadi@ucl.ac.uk
[⋆⋆⋆] sasi.murakonda@privitar.com
[†] s.murdoch@ucl.ac.uk
[‡] mohammad@flower.dev
[§] theodorakopoulosg@cardiff.ac.uk
[¶] suzanne.weller@privitar.com

vital variant of FL, with various applications, e.g., in dealing with crime [5] and healthcare [44]. VFL refers to the FL setting where datasets distributed among different parties (e.g., banks) have some intersection concerning users (e.g., have certain customers' names in common) while holding different features, e.g., customers' names, addresses, and how they are perceived by a financial institution. Horizontal Federated Learning (HFL) is another important variant of FL where participants share the same feature space while holding different users, e.g., customers' attributes are the same, but different banks may have different customers.

Advanced privacy-preserving FL-based solutions aiming to detect anomalies and deal with financial fraud may face a new challenge. In this setting, datasets for financial transactions might be partitioned both vertically and horizontally. For instance, a third-party Financial Service Provider (FSP) may have details of financial transactions including customers' names, and involved banks, while each FSP's partner bank may have some details/features of a subset of these customers. Thus, existing solutions for VFL or HFL cannot be directly applied to deal with this challenge.

1.1 Our Contributions

In this work, we introduce *Starlit*, a pioneering scalable privacy-preserving federated learning mechanism that can help enhance financial fraud detection. By devising and utilizing *Starlit* in the context of financial fraud, we address all limitations of the state-of-the-art FL-based mechanisms, proposed in [5,46,32]. Specifically, we (1) formally define and prove *Starlit*'s security (in the simulation-based paradigm), (2) do not place any assumption on how suspicious accounts of customers are treated by their financial institutions, (3) make *Starlit* scale linearly with the number of participants (i.e., its overhead is $O(n)$) while refraining from using fully homomorphic encryption, (4) include all phases of *Starlit* in the implementation, performance evaluation, and security analysis, and (5) make *Starlit* resilient against dropouts of clients.

Starlit offers two compelling properties not found in existing VFL schemes. These include the ability to securely:

– Identify discrepancies among the values of shared features in common users between distinct clients' datasets. For instance, in the context of banking, FSP and a bank can detect if a certain customer provides a different home address to each.
– Aggregate common features in shared users among different clients' datasets, even when these features have varying values. For instance, this feature will enhance FSP's data by reflecting whether FSP and multiple banks consider a certain customer suspicious, according to the value of a flag independently allocated by each bank to that customer's account.

We have implemented *Starlit* and evaluated its performance using synthetic data which comprises about four million rows. This synthetic data was provided by a major organization globally handling financial transactions. *Starlit* stands out as the first solution that simultaneously provides the features mentioned above. We identify several potential applications for *Starlit*, including mitigating terrorism, enhancing digital health, and aiding in the detection of benefit fraud (see Section 12).

To develop *Starlit*, we use a combination of several tools and techniques, such as SecureBoost (for VFL), Private Set intersection (for identity alignment and finding discrepancies among different entities' information), and Differential Privacy to preserve the privacy of accounts' flags (that indicate whether an account is deemed suspicious). Moreover, based on our observation that each dataset's sample (or row), such as a financial transaction, can be accompanied by a random identifier, we allow a third-party feature collector to efficiently aggregate clients' flags without being able to associate the flags values with a specific feature, e.g., customer's name.

Summary of our Contributions. In this work, we:

• Introduce *Starlit*, a novel scalable privacy-preserving federated learning mechanism, with various real-world applications.
• Formally define and prove *Starlit*'s security using the simulation-based paradigm.
• Implement *Starlit* and conduct a comprehensive evaluation of its performance.

1.2 Primary Goals and Setting

This paper focuses on a real-world scenario in which a server, denoted by Srv, wants to train a machine-learning model to detect anomalies using its data, and complementary data held by different clients $C = \{\mathrm{C}_1, ..., \mathrm{C}_m\}$. For instance, Srv can be a Financial Service Provider (FSP) such as SWIFT[5], Visa[6], PayPal[7], CHIPS[8], and SEPA[9]—facilitating financial transactions and payments between various clients in set C, such as banks, eBay, and Amazon—that aims to detect anomalous transactions.

In this setting, Srv may maintain a database of samples/rows between interacting clients, but it does not possess all the details about the users included in each sample. For instance, in the context of financial transactions, FSP holds a dataset containing samples (i.e., transactions) between the ordering account held by bank C_i and the beneficiary account held by bank C_j.

Each sample may contain a customer's name, the amount sent, home address, and information about C_i, and C_j. Each client in C maintains a dataset containing certain customers' account information, including customers' details, their transaction history, and even local assessments of their known financial activities. However, each C_j may not hold all users (e.g., customers) that Srv is interested.

While Srv is capable of training a model to detect anomalous transactions using its data, it could enhance the analytics by considering the complementary data held by other clients concerning the entities involved in the transactions. The ultimate goal is to enable Srv to collaborate with other clients to develop a model that is significantly better than the one developed on Srv's data alone, e.g., to detect suspicious transactions and ultimately to deal with financial fraud.

However, a mechanism that offers the above feature must satisfy vital security and system constraints; namely, (i) the privacy of clients' data should be preserved from their counterparts, and (ii) the solution must be efficient for real-world use cases. The aforementioned setting is an example of FL on vertically and horizontally partitioned data in which each Srv's transaction is associated with a sender C_i (e.g., ordering bank), and receiver C_j, e.g., beneficiary bank. Our solution will enhance Srv's dataset with two primary types of features using the datasets of C_i and C_j:

- **Discrepancy Feature**: This will enhance Srv's data by reflecting whether there is a discrepancy between (i) the (value of the) feature, such as a customer's name and address, it holds about a certain user U under investigation and (ii) the feature held by sending client C_i and receiving client C_j about the same user. For each user, this feature is represented by a pair of binary values $(b_{u,i}, b_{u,j})$, where $b_{u,i}$ and $b_{u,j}$ represents whether the information that Srv holds matches the one held by the sending and receiving clients respectively.
- **Sample's Flag Feature**: This will enhance Srv's data by reflecting whether Srv and a client have the same view of a certain user, e.g., a customer is suspicious. This feature is based on a pair of binary private flags for a certain user, where one flag is held by the sending client and the other one is held by the receiving client. In the context of banking, banks often allocate flags to each customer's account for internal use. The value of this flag is set based on the user's transaction history and determines whether the bank considers the account holder suspicious.

To preserve the privacy of the participating parties' data (e.g., data of non-suspicious customers held by banks) while aligning Srv's dataset with the features above, we rely on a set of privacy-enhancing techniques, such as Private Set Intersection (PSI) and Differential Privacy (DP). Briefly, to enable Srv to find out whether the data it holds about a certain (suspicious) user matches the one held by a client, we use PSI. Furthermore, to enhance Srv's data with the flag feature, each client uses local DP to add noise to their flags and sends the noisy flags to a third-party flag collector which feeds them to the model training phase.

[5] `https://www.swift.com`
[6] `https://www.visa.co.uk/about-visa.html`
[7] `https://www.paypal.com/uk/home`
[8] `https://www.theclearinghouse.org/payment-systems/chips`
[9] `https://finance.ec.europa.eu/consumer-finance-and-payments/payment-services/single-euro-payments-area-sepa_en`

2 Related Work

In this section, we briefly discuss the privacy-preserving FL-based approaches used to deal with fraudulent transactions. We refer readers to Appendix A for a survey of related work. Lv *et al.* [39] introduced an approach to identify black market fraud accounts before fraudulent transactions occur. It aims to guarantee the safety of funds when users transfer funds to black market accounts, enabling the financial industry to utilize multi-party data more efficiently. It involves data provided by financial and social enterprises. The approach utilizes *insecure* hash-based PSI for identity alignment.

This scheme differs from *Starlit* in a couple of ways: (i) *Starlit* operates in a multi-party setting, where various clients contribute their data, in contrast to the aforementioned scheme, which has been designed for only two parties, and (ii) *Starlit* deals with the data partitioned both horizontally and vertically, whereas the above scheme focuses only on vertically partitioned data.

Recently, Arora *et al.* [5] introduced an approach that relies on oblivious transfer, secret sharing, DP, and multi-layer perception. The authors have implemented the solution and conducted a thorough analysis of its performance.
Starlit versus the Scheme of Arora et al. The latter assumes that the ordering bank never allows a customer with a dubious account to initiate transactions but allows the same account to receive money. In simpler terms, this scheme exclusively addresses frozen accounts, restricting its applicability. This setting will exempt the ordering bank from participating in MPC, enhancing the efficiency of the solution.

In the real world, users' accounts might be deemed suspicious (though not frozen), yet they can still conduct financial transactions within their bank. The bank may handle such accounts more cautiously than other non-suspicious accounts. In contrast, *Starlit* (when applied to financial transactions context) does not place any assumption on how a bank treats a suspicious account.

Furthermore, unlike the scheme proposed in [5], which depends on an ad-hoc approach to preserve data privacy during training, our solution, *Starlit*, employs SecureBoost—a well-known scheme extensively utilized and analyzed in the literature. Thus, compared to the scheme in [5], Starlit considers a more generic scenario and relies on a more established scheme for VFL.

Recently, another approach has been developed by Qiu *et al.* [46]. It uses neural networks and shares the same objective as the one by Arora *et al.* However, it strives for computational efficiency primarily through the use of symmetric key primitives. The scheme incorporates the elliptic-curve Diffie-Hellman key exchange and one-time pads to secure exchanged messages during the model training phase. This scheme has also been implemented and subjected to performance evaluation.
Starlit versus the Scheme of Qiu et al. The latter scheme requires each client (e.g., bank) to possess knowledge of the public key of every other client and compute a secret key for each through the elliptic-curve Diffie-Hellman key exchange scheme. Consequently, this approach imposes $O(n)$ modular exponentiation on each client, resulting in the protocol having a complexity of $O(n^2)$, where n represents the total number of clients. In contrast, in *Starlit*, each client's complexity is independent of the total number of clients and each client does not need to know any information about other participating clients. Moreover, the scheme proposed in [46] assumes the parties have already performed the identity alignment phase, therefore, the implementation, performance evaluation, and security analysis exclude the identity alignment phase.

Furthermore, the scheme in [46] fails to terminate successfully even if only one of the clients neglects to transmit its message. In this scheme, each client, utilizing the agreed-upon key with every other client, masks its outgoing message with a vector of pseudorandom blinding factors. The expectation is that the remaining clients will mask their outgoing messages with the additive inverses of these blinding factors. These blinding factors are generated such that, when all outgoing messages are aggregated, the blinding factors cancel each other out.

Nevertheless, if one client fails to send its masked message, the aggregated messages of the other clients will still contain blinding factors, hindering the training on correct inputs. In contrast, *Starlit* does not encounter this limitation. This is because the message sent by each client is independent of the messages transmitted by the other clients.

Kadhe *et al*. [32] proposed an anomaly detection scheme, that uses fully homomorphic encryption (computationally expensive), DP, and secure multi-party computation. The authors have also implemented their solution and analyzed its performance.
Starlit versus the Scheme of Kadhe et al. The latter heavily relies on fully homomorphic encryption. In this scheme, all parties need to perform fully homomorphic operations. This will ultimately affect both the scalability and efficiency of this scheme. In contrast, *Starlit* does not use any fully homomorphic scheme.

All of the above solutions share another shortcoming, they lack formal security definitions and proofs of the proposed systems.

3 Informal Threat Model

Starlit involves three types of parties:

- Server (Srv). It wants to train a model to detect anomalies using its data, and complementary data held by different clients. The data Srv maintains is partitioned vertically and horizontally across different clients. Each sample in the data includes various features, e.g., a user's name, sender client, and receiver client.
- Clients ($C_1, ..., C_n$). They are different clients (e.g., nodes, devices, or organizations) that contribute to FL by providing local complementary data to the training process.
- Flag Collector (FC). It is a third-party helper that aggregates some of the features held by different clients. FC is involved in *Starlit* to enhance the system's scalability.

We assume that all the participants are honest but curious (a.k.a. passive adversaries), as it is formally defined in [24]. Hence, they follow the protocol's description. But, they try to learn other parties' private information. We consider it a privacy violation if the information about one party is learned by its counterpart during the model training (including pre-processing). We assume that parties communicate with each other through secure channels.

4 Preliminaries

4.1 Notations and Assumptions

Table 1 summarizes the notations used in this paper. Let $\mathcal{G}$ be a multi-output function, $\mathcal{G}(inp) \rightarrow (outp_1, ..., outp_n)$. Then, by $\mathcal{G}_i(inp)$ we refer to the i-th output of $\mathcal{G}(inp)$, i.e., $outp_i$.

4.2 Private Set Intersection (PSI)

PSI is a cryptographic protocol that enables mutually distrustful parties to compute the intersection of their private datasets without revealing anything about the datasets beyond the intersection.

The fundamental functionality computed by any n-party PSI can be defined as $\mathcal{G}$ which takes as input sets $S_1, ..., S_n$ each of which belongs to a party and returns the intersection $S_\cap$ of the sets to a party. More formally, the functionality is defined as: $\mathcal{G}(S_1, ..., S_n) \rightarrow (S_\cap, \underbrace{\perp, ..., \perp}_{n-1})$, where $S_\cap = S_1 \cap S_2, ..., \cap S_n$. In this work, we denote the concrete PSI protocol with $\mathcal{PSI}$.

4.3 Local Differential Privacy

Local Differential Privacy (LDP) entails that the necessary noise addition for achieving differential privacy is executed locally by each individual. Each individual employs a random perturbation algorithm, denoted as M, and transmits the outcomes to the central entity. The perturbed results are designed to ensure the protection of individual data in accordance with the specified ϵ value. This concept has been formally stated in [20]. Below, we restate it.

Table 1: Notation table.

Symbol	Description
Srv	Server
FSP	Financial Service Provider
FC	Feature Collector
C$_i$	A client or bank
ϵ	Parameter that quantifies the privacy guarantee provided by a differentially private mechanism.
PSI	Private Set Intersection
DP	Differential Privacy
ML	Machine Learning
FL	Federated Learning
VFL	Vertical Federated Learning
RR	Randomized Response
AUPRC	Area Under the Precision-Recall Curve
GOSS	Gradient-based One Side Sampling
H	Hour
Pr	Probability
S_i	A private set
$\mathcal{V}$	Set of flag values
$\pi(v), v \in \mathcal{V}$	Prior probability of value v (FSP's prior knowledge)
$d_p(\hat{v}, v) : \mathcal{V} \times \mathcal{V} \to \mathcal{R}$	Privacy metric (attacker's error when estimating v as $\hat{v}$)
$f(v'\|v) : \mathcal{V} \times \mathcal{V} \to a \in \{0, 1\}$	Privacy mechanism
$\|\|$	Concatenation
$\mathcal{L}$	Leakage function of *Celestial* and *Starlit*
$\mathcal{L}_1$	FSP –side leakage in *Starlit*
$\mathcal{L}_2$	FC –side leakage in *Starlit*
$\mathcal{L}_{i+2}$	C$_i$–side leakage in *Starlit*
$\mathcal{W}$	Leakage function in (V)ML
$\mathcal{F}$	Functionality of *Celestial*
prm_i	Input parameter of a party to (V)FL
$\|S\|$	Size of set or database S

Definition 1 *Let X be a set of possible values and Y the set of noisy values. M is ϵ-locally differentially private (ϵ-LDP) if for all $x, x' \in X$ and for all $y \in Y$:*

$$Pr[M(x) = y] \leq e^{\epsilon} \cdot Pr[M(x') = y] \tag{1}$$

For a binary attribute, i.e., $X = \{0, 1\}$, this protection means that an adversary who observes y cannot be sure whether the true value was 0 or 1.

As proposed by Wang *et al.* [65], we consider two generalized mechanisms on binary attributes for achieving LDP. The first one uses the Randomized Response (RR) and the second one relies on adding Laplace noise with post-processing (applying a threshold of 0.5) for binarizing the values. For either mechanism, each individual employs a 2×2 transformation matrix $P = [p_{ij}]$ to perturb their true value, where the element at position (i, j) represents the probability of responding with value j if the true value is i. To satisfy the definition of DP at privacy level ϵ, we need to have $p_{00}/p_{01} \leq \epsilon$.

Randomized Response. In addition to the requirement of satisfying ϵ-LDP, Wang *et al.* [65] propose selecting the matrix parameters to maximize the probability of retaining the true value, i.e., to maximize $p_{00} + p_{11}$. This yields the following transformation matrix:

$$Q := \begin{pmatrix} \frac{e^{\epsilon}}{1+e^{\epsilon}} & \frac{1}{1+e^{\epsilon}} \\ \frac{1}{1+e^{\epsilon}} & \frac{e^{\epsilon}}{1+e^{\epsilon}} \end{pmatrix} \tag{2}$$

Laplace Noise with Post-Processing. The Laplace mechanism is a DP mechanism proposed by the original DP paper [21]. To achieve ϵ-DP, this mechanism adds noise drawn from the Laplace distribution with parameter $\frac{1}{\epsilon}$ to the true value. This creates continuous values, instead of binary ones. Consequently, we need to make the output binary. It is demonstrated in [65] that using a threshold of 0.5 maximizes $p_{00} + p_{11}$, i.e., if the continuous value is above 0.5, we set the final value to 1; otherwise, we set it to 0. This leads to the following transformation matrix:

$$Q' := \begin{pmatrix} 1 - \frac{1}{2}e^{-\frac{\epsilon}{2}} & \frac{1}{2}e^{-\frac{\epsilon}{2}} \\ \frac{1}{2}e^{-\frac{\epsilon}{2}} & 1 - \frac{1}{2}e^{-\frac{\epsilon}{2}} \end{pmatrix} \tag{3}$$

Note that although we list the matrices only for binary attributes here, both mechanisms generalize to the case of categorical variables with more than two values.

Mechanisms for Optimal Inference Privacy. Any randomization mechanism for obfuscating the flags while sharing can offer certain protection against inference attacks by Srv. Given a value of ϵ, there can be many mechanisms that satisfy the constraint of DP, of which two can be found using Equation 2 (for RR) and Equation 3 (for Laplace).

These mechanisms assign an equal probability of converting a 0 to 1 and 1 to 0. They need not be the optimal transformation matrices that provide maximum inference privacy, i.e., maximum protection against Srv's ability to infer the flag values.

As one of the key contributions of this work, we developed a framework to explore the entire space of transformation matrices and find optimal mechanisms that maximize inference privacy, under the given constraints on utility and local differential privacy.

The main advantage of formulating the construction of a privacy mechanism as an optimization problem is that we can automatically explore a large solution space to discover optimal mechanisms that are not expressible in closed form (such as the Laplace or Gaussian mechanism). Section 7 presents further details about the game construction and solution.

4.4 Federated Learning

Unlike traditional centralized methods, where data is pooled into a central server, FL allows model training to occur on individual devices/clients contributing private data. This preserves the privacy of the data to some extent by avoiding direct access to them. The process involves training a global model through collaborative learning on local data, and only the model updates, rather than raw data, are transmitted to the central server.

This decentralized paradigm is particularly advantageous in scenarios where data privacy is paramount, such as in healthcare or finance, as it enables machine learning advancements without compromising sensitive information. Algorithm 1 presents the overall workflow of FL.

Algorithm 1 : Federated Learning's General Procedure

```
1: Server:
2: Initialize global model $\theta$
3: for each round $k = 1, 2, 3, ..., K$ do
4:     Broadcast $\theta$ to all participating devices
5:     Clients:
6:     for each client $i$ (where $1 \leq i \leq n$) in parallel do
7:         Receive global model $\theta$
8:         Compute local update $g_i$ using local data
9:         Send $g_i$ to the server
10:    Server:
11:    Aggregate local updates: $G_k = \sum_{i=1}^{n} g_i$
12:    Update global model: $\theta_{k+1} = \text{UpdateModel}(\theta_k, G_k)$
```

SecureBoost: A Lossless Vertical Federated Learning Framework. SecureBoost, introduced in [18], stands out as an innovative FL framework designed to facilitate collaborative machine learning model training among multiple parties while safeguarding the privacy of their individual datasets. It accomplishes this by leveraging homomorphic encryption to execute computations on encrypted data, ensuring the confidentiality of sensitive information throughout the training procedure. There are two main technical concepts and phases involved in SecureBoost:

- **Secure Tree Construction:** SecureBoost builds boosting trees, a specific type of machine learning model, by utilizing a non-federated tree boosting mechanism called XGBoost [16] and a partially homomorphic encryption scheme, such as Paillier encryption [45], allowing various operations such as majority votes and tree splits to be performed without exposing the underlying plaintext data to the system's participants.
- **Entity Alignment:** To enable collaborative training, SecureBoost conducts entity alignment to recognize corresponding user records across diverse data silos. This process is typically executed through an MPC (such as PSI), guaranteeing the confidentiality of individual identities.

SecureBoost has been implemented in an open-sourced FL project, called FATE.[10] As discussed above, (V)FL is an interactive process within which parties exchange messages. Thus, there is a possibility of a leakage to these parties. To formally define the leakage to each party in this process, below we introduce a leakage function $\mathcal{W}$.

$$\mathcal{W}(prm_1, ..., prm_n) \rightarrow (l_1, ..., l_n) \tag{4}$$

This function receives the input parameter prm_i from each party in (V)FL and returns leakage l_i to the i-th party, representing the information that (V)FL exposes to that specific party. Note that prm_i is a set, containing all (intermediate) results possibly generated over multiple iterations. This leakage will be considered in *Starlit*'s formal definition (in Sections 6 and 9) and proof (in Appendix B).

4.5 Flower: A Federated Learning Implementation Platform

We implement *Starlit* within *Flower*, which was introduced in [11]. This framework offers several advantages, including scalability, ease of use, and language and ML framework agnosticism.

Flower comprises three main components: a set of clients, a server, and a strategy. Federated learning is often viewed as a combination of global and local computations. The server handles global computations and oversees the learning process coordination among the clients. The clients perform local computations, utilizing data for training or evaluating model parameters.

The logic for client selection, configuration, parameter update aggregation, and federated or centralized model evaluation can be articulated through strategy abstraction. The implementation of the strategy represents a specific FL algorithm. Flower provides reference implementations of popular FL algorithms such as FedAvg [40], FedOptim [47], or FedProx [37].

4.6 Security Model

In this paper, we consider static adversaries. We use the simulation-based paradigm of secure multi-party computation [24] to define and discuss the security of the proposed scheme. Since we focus on the static passive (semi-honest) adversarial model, we will restate the security definition in this adversarial model.

Two-party Computation. A two-party protocol Γ problem is captured by specifying a random process that maps pairs of inputs to pairs of outputs, one for each party. Such process is referred to as a functionality denoted by $\mathcal{F} : \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^* \times \{0,1\}^*$, where $\mathcal{F} := (\mathcal{F}_1, \mathcal{F}_2)$. For every input pair (x, y), the output pair is a random variable $(\mathcal{F}_1(x, y), \mathcal{F}_2(x, y))$, such that the party with input x wishes to obtain $\mathcal{F}_1(x, y)$ while the party with input y wishes to receive $\mathcal{F}_2(x, y)$. When $\mathcal{F}$ is deterministic, then $\mathcal{F}_1 = \mathcal{F}_2$. The above functionality can be easily extended to $n > 2$ parties.

[10] `https://github.com/FederatedAI/FATE`

Security in the Presence of Passive Adversaries. In the passive adversarial model, the party corrupted by such an adversary correctly follows the protocol specification. Nonetheless, the adversary obtains the internal state of the corrupted party, including the transcript of all the messages received, and tries to use this to learn information that should remain private.

Loosely speaking, a protocol is secure if whatever can be computed by a party in the protocol can be computed using its input and output only. In the simulation-based model, it is required that a party's view in a protocol's execution can be simulated given only its input and output. This implies that the parties learn nothing from the protocol's execution. More formally, party i's view (during the execution of Γ) on input pair (x, y) is denoted by $\mathsf{View}_i^{\Gamma}(x, y)$ and equals $(w, r^i, m_1^i, ..., m_t^i)$, where $w \in \{x, y\}$ is the input of i^{th} party, r_i is the outcome of this party's internal random coin tosses, and m_j^i represents the j^{th} message this party receives. The output of the i^{th} party during the execution of Γ on (x, y) is denoted by $\mathsf{Output}_i^{\Gamma}(x, y)$ and can be generated from its own view of the execution.

Definition 1. *Let $\mathcal{F}$ be the deterministic functionality defined above. Protocol Γ securely computes $\mathcal{F}$ in the presence of a passive probabilistic polynomial-time (PPT) adversary $\mathcal{A}$, if for every $\mathcal{A}$ in the real model, there exist PPT algorithms* $(\mathsf{Sim}_1, \mathsf{Sim}_2)$ *such that:*

$$\{\mathsf{Sim}_1(x, \mathcal{F}_1(x, y))\}_{x,y} \overset{c}{\equiv} \{\mathsf{View}_1^{\mathcal{A},\Gamma}(x, y)\}_{x,y}$$

$$\{\mathsf{Sim}_2(y, \mathcal{F}_2(x, y))\}_{x,y} \overset{c}{\equiv} \{\mathsf{View}_2^{\mathcal{A},\Gamma}(x, y)\}_{x,y}$$

Definition 1 can be easily extended to $n > 2$ parties.

5 System Design

Starlit consists of two main phases: (i) feature extraction and (ii) training. During the feature extraction phase, the two types of features (discussed in Section 1.2) are retrieved in a privacy-preserving manner, the data is aligned, and then passed onto a third party, called "Feature Collector (FC)". The use of FC drastically simplifies the training phase from n-party down to 2-party VFL, which will enable the system to scale to a large number of banks.

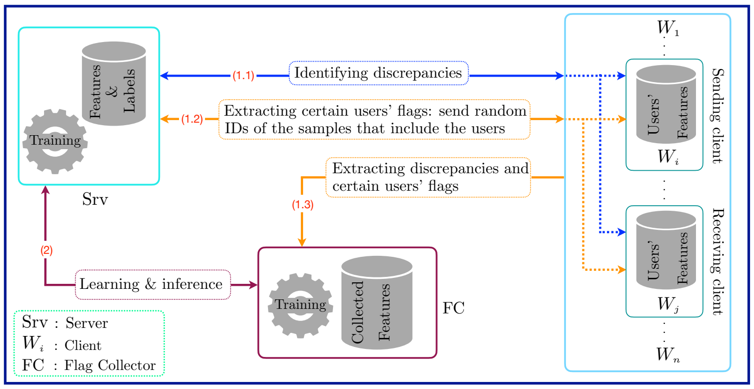


Fig. 1: Outline of parties' interactions in *Starlit*.

Figure 1 outlines the interaction between the parties in *Starlit*. In Phase 1, each client initially engages with Srv to identify discrepancies in specific user features. Additionally, in the same phase, each client interacts with Srv to extract flags for certain users. Subsequently, each client combines the results of discrepancy extraction with the outcomes of flag extraction, sending the pair along with a random ID (known also to Srv) to FC. Moving on to Phase 2, FC and Srv collaborate to train the VFL model using FC's collected features, Srv's local data, and SecureBoost.

This procedure may still leave the chance of an inference attack during model training/deployment. To address this issue, we use LDP, where any flag values that leave the client are obfuscated via a randomization strategy. Note that this protection is an additional layer on top of what is already offered by the SecureBoost protocol, which only shares encrypted (aggregated) gradient information.

6 Formal Security Definition

In this section, we introduce a generic formal definition, that we call *Celestial*. It establishes the primary security requirements of privacy-preserving (V)FL schemes such as *Starlit*. *Celestial* involves three types of parties, (i) a service provider Srv, (ii) a feature collector FC, and (iii) a set of clients $\{\mathrm{C}_1, ..., \mathrm{C}_n\}$ contributing their private inputs. Informally, *Celestial* allows Srv to generate a (global) model given its initial model and the inputs of $\mathrm{C}_1, ..., \mathrm{C}_n$. To achieve a high level of computational efficiency and scalability, in *Celestial*, we involve a third-party FC that assists Srv with computing the model (by interacting with C_is and retrieving the features they hold). The functionality $\mathcal{F}$ that *Celestial* computes takes an input initial model θ from Srv, a set S_i from every C_i, and no input from FC. It returns to Srv an updated model θ'. It returns nothing to the rest of the parties.[11] Hence, $\mathcal{F}$ can be formally defined as follows.

$$\mathcal{F}(\theta, S_1, ..., S_n, \bot) \rightarrow (\theta', \underbrace{\bot, ..., \bot}_{n}, \bot) \tag{5}$$

Since (i) FC interacts with $\mathrm{C}_1, ..., \mathrm{C}_n$ and collects some features from them and (ii) Srv generates the model in collaboration with $\mathrm{C}_1, ..., \mathrm{C}_n$ and FC, there is a possibility of leakage to the participating parties. Depending on the protocol that realizes $\mathcal{F}$ this leakage could contain different types of information. For instance, it could contain (a) each C_i's local model outputs and corresponding gradients (a.k.a. intermediate results) when using gradient descent [64] in VFL, (b) the output of entity aligning procedure, (c) information about features, or (d) nothing at all. We define this leakage as an output of a leakage function defined as follows:

$$\mathcal{L}(inp) \rightarrow (l_1, l_2, ..., l_{n+2}) \tag{6}$$

$\mathcal{L}(inp)$ takes all parties (encoded) inputs, denoted as *inp*. It returns leakage l_1 to Srv, l_2 to FC, and leakage l_i to C_{i-2}, for all i, where $3 \leq i \leq n+2$.

We assert that a protocol securely realizes $\mathcal{F}$ if (1) it reveals nothing beyond a predefined leakage to a certain party and (2) whatever can be computed by a party in the protocol can be obtained from its input and output only. This is formalized by the simulation paradigm. We require a party's view during the execution of the protocol to be simulatable given its input, output, and the leakage that has been defined for that party.

Definition 2 (Security of *Celestial*). *Let $\mathcal{F}$ be the functionality presented in Relation 5. Also, let $\mathcal{L}$ be the above leakage function, presented in Relation 6. We assert that protocol Γ securely realizes $\mathcal{F}$, in the presence of a static semi-honest adversary, if for every non-uniform PPT adversary $\mathcal{A}$ for the real model, there exists a non-uniform PPT adversary (or simulator)*

Sim *for the ideal model, such that for every party P, where $P \in \{Srv, C_1, ..., C_n, FC\}$, the following holds:*

$$\{\mathsf{Sim}_{Srv}^{\mathcal{F},\mathcal{L}_1}(\theta, \theta')\}_{inp} \overset{c}{\equiv} \{\mathsf{View}_{Srv}^{\mathcal{A},\Gamma}(inp)\}_{inp} \tag{7}$$

$$\{\mathsf{Sim}_{FC}^{\mathcal{F},\mathcal{L}_2}(\bot, \bot)\}_{inp} \overset{c}{\equiv} \{\mathsf{View}_{FC}^{\mathcal{A},\Gamma}(inp)\}_{inp} \tag{8}$$

$$\{\mathsf{Sim}_{C_i}^{\mathcal{F},\mathcal{L}_{i+2}}(S_i, \bot)\}_{inp} \overset{c}{\equiv} \{\mathsf{View}_{C_i}^{\mathcal{A},\Gamma}(inp)\}_{inp} \tag{9}$$

where $1 \leq i \leq n$.

[11] For the sake of simplicity, we have restricted the learning of the global model to only Srv. This approach can be easily generalized to allow each C_i to learn the model as well, by mandating Srv to transmit the global model to every C_i.

7 Flag Protection

In this section, we initially present a game for flag protection. Then, we explain how to construct a concrete optimization problem to realize the game.

7.1 The Game

The problem of finding a Privacy Mechanism (PM) that offers optimal flag privacy to a client given the knowledge of the adversary (e.g., Srv or FC), is an instance of a Bayesian Stackelberg game. In a Stackelberg game the *leader*, in our case the client, plays first by choosing a PM (a transformation matrix), and commits to that by running it on the actual values of the flags; and the *follower*, in our case Srv, plays next estimating the flag value, knowing the PM that the client has committed to. It is a Bayesian game because Srv has incomplete information about the true flag values and plays according to its prior information about these values. Inspired by similar work in location privacy protection games [52,51], we now proceed to define the game for a single flag value, but the transformation matrix computed will be used for each value:

- **Step 0.** Nature selects a flag value $v \in \mathcal{V}$ for the client according to a probability distribution $\pi(.)$, the *flag profile*. That is, flag value v is selected with probability $\pi(v)$. This encodes the relative proportions of the flag values in the dataset.
- **Step 1.** Given v, the client runs the PM $f(v'|v)$ to select a replacement value $v' \in \mathcal{V}$.
- **Step 2.** Having observed v', Srv selects an estimated flag value $\hat{v} \sim g(\hat{v}|v'), \hat{v} \in \mathcal{V}$. Srv knows the probability distribution $f(v'|v)$ used by the PM, and the client's flag profile $\pi(.)$, but not the true flag value v.
- **Step 3.** The game outcome is the number $d_p(\hat{v}, v)$, which is the client's privacy for this iteration of the game. This number represents Srv's error in estimating the true value of the flag.

Fig. 2: Bayesian game for a single flag value.

The above description is common knowledge to Srv and the client. Srv tries to minimize the expected game outcome (the error in the estimation of the flag value) via its choice of g, while the client tries to maximize it via its choice of transformation matrix f. As changing the flag values distorts the data for training the ML algorithm, we impose upper bounds $p^{\max}(v', v)$ on the probabilities $f(v'|v)$. Finally, and independently of the above considerations, we want the PM to be ϵ-differentially private.

7.2 Optimization Problem

We now explain how to build a concrete optimization problem that encodes the above description and that we can solve to obtain the optimal PM $f()$, given $\pi(), d_p, p^{\max}(v', v)$, and ϵ. Srv knows $f(v'|v)$ implemented by PM. Thus, it can form a posterior distribution $\Pr(v|v')$ on the true flag value, conditional on the observation v'. Then, Srv chooses $\hat{v}$ to minimize the conditional expected privacy, where the expectation is taken under the posterior distribution:

$$\text{Choose } \hat{v} \text{ that satisfies } \arg\min_{\hat{v}} \sum_{v} \Pr(v|v') d_p(\hat{v}, v). \tag{10}$$

Recall that variables v, v', and $\hat{v}$ take values in $\mathcal{V}$, the set of flag values, so the range of any minimization or summation involving any of these variables will be the set $\mathcal{V}$. If there are multiple minimizing values of $\hat{v}$, then Srv may randomize among them. This randomization is expressed through $g(\hat{v}|v')$, and in this case (10) would be rewritten as $\sum_{v,\hat{v}} \Pr(v|v') g(\hat{v}|v') d_p(\hat{v}, v)$.

It is important to note that the value of this equation would be the same as the value computed in Relation (10) for any minimizing value of $\hat{v}$. As $\pi(v)$ and $f(v|v')$ are known to Srv, it holds that:

$$\Pr(v|v') = \frac{\Pr(v, v')}{\Pr(v')} = \frac{f(v'|v)\pi(v)}{\sum_v f(v'|v)\pi(v)} \tag{11}$$

Thus, for a given v', the client's conditional privacy is given by Relation (10). The probability that v' is reported is $\Pr(v')$. Hence, the unconditional expected privacy of the client is:

$$\sum_{v'} \Pr(v') \min_{\hat{v}} \sum_{v} \Pr(v|v') d_p(\hat{v}, v) = \sum_{v'} \min_{\hat{v}} \sum_{v} \pi(v) f(v'|v) d_p(\hat{v}, v) \tag{12}$$

To facilitate computations, we define:

$$x_{v'} \min_{\hat{v}} \sum_{v} \pi(v) f(v'|v) d_p(\hat{v}, v). \tag{13}$$

Incorporating $x_{v'}$ into Relation (12), the unconditional expected privacy of the client can be rewritten as

$$\sum_{v'} x_{v'} \tag{14}$$

which the client aims to maximize by choosing $f(v'|v)$. The minimum operator makes the problem non-linear, undesirable, but Relation (13) can be transformed into a series of linear constraints:

$$x_{v'} \leq \sum_{v} \pi(v) f(v'|v) d_p(\hat{v}, v), \forall \hat{v} \tag{15}$$

Maximizing the result in Relation (14) under Relation (13) is equivalent to maximizing Relation (14) under Relation (15). For every v', there must be some $\hat{v}$ for which Relation (15) holds as strict equality; Otherwise, we could increase one of the $x_{v'}$, so the value of Relation (14) would increase. From Relations (14) and (15), the linear program for the client is constructed by choosing $f(v'|v), x_{v'}, \forall v, v'$ to solve the following linear programming problem.

$$\textbf{Maximize} \sum_{v'} x_{v'} \tag{16}$$

subject to

$$x_{v'} - \sum_{v} \pi(v) f(v'|v) d_p(\hat{v}, v) \leq 0, \forall \hat{v}, v' \tag{17}$$

$$f(v'|v) \leq p^{\max}(v', v), \forall v, v' \tag{18}$$

$$\sum_{v'} f(v'|v) = 1, \forall v \tag{19}$$

$$f(v'|v) \geq 0, \forall v, v' \tag{20}$$

$$\frac{f(v'|v_1)}{f(v'|v_2)} \leq \exp(\epsilon), \forall v', v_1, v_2 \tag{21}$$

Constraints (19) and (20) reflect that $f(v'|v)$ is a probability distribution function for each v, while (21) enforces ϵ-differential privacy.

Alternative Quality-Privacy Tradeoffs. The above formulation encodes the privacy-accuracy tradeoff in one particular way – maximize inference privacy, subject to a differential privacy constraint and an accuracy-related constraint on the probabilities $f(v'|v)$. The general framework is flexible to accommodate other tradeoffs.

For example, instead of introducing constraints $p^{\max}(v', v)$ on $f(v'|v)$, we can introduce an Accuracy Loss (AL) matrix with entries $AL_{v'v}$ that quantify the loss in accuracy when replacing value v with v'. Then, instead of Relation (18), we can upper bound the total expected accuracy loss that is caused by a given transformation matrix f with the following constraint:

$$AL(f) := \sum_{v} \pi(v) \sum_{v'} f(v'|v) AL_{v'v} \leq AL^{\max}.$$

Alternatively, in a more radical departure from the original formulation, rather than aiming to maximize the client's privacy (inference privacy) subject to AL constraints, we could instead aim to minimize the accuracy loss $AL(f)$ subject to a lower bound on inference privacy, i.e., $\sum_{v'} x_{v'} \geq PR^{\min}$.

In general, the main benefit of formulating the construction of the transformation matrix as an optimization problem is that we can automatically explore a large solution space to discover optimal probability distributions $f(v'|v)$ that are not expressible in closed form (such as the Laplace or Gaussian mechanism), so human intuition would not be able to find them.

8 Starlit's Phases in Detail

8.1 Privacy-Preserving Feature Extraction

In this section, we elaborate on the two primary privacy-preserving mechanisms that we designed to extract features.

Finding Features' Discrepancies. Let $T = \{t_{u,1}, ..., t_{u,m}\}$ be a subset of features that Srv holds for a user U. Consider the scenario where Srv wants to check with a pair of clients $(\mathrm{C}_i, \mathrm{C}_j)$ if there is a discrepancy between some of the features in T that Srv, C_i, and C_j hold, without revealing and being able to learn anything else. This approach could provide information about anomalous transactions.

In the domain of financial transactions, we analyzed synthetic data provided to us and identified key features possessed by FSP for each transaction (with FSP acting as Srv). These features include: (i) $customer_{name}$, (ii) $countryCity_{zipcode}$, and (iii) $street_{name}$ for both the ordering and beneficiary banks. Each bank, per user, maintains various features such as $customer_{name}$, $countryCity_{zipcode}$, and $street_{name}$ (with an associated flag).

Diverse parties may hold varying perspectives on the value of these features. Discrepancies can arise from various factors. For instance, a user may have supplied divergent information to different parties. In the given scenario, a customer might hold accounts with both the ordering and beneficiary banks but could have provided inconsistent details, such as their address, to these banks. Additionally, there is a possibility that the values maintained by Srv have been tampered with, potentially by external entities [10,67]. Thus, incorporating a feature that signals disparities between a client's data and Srv's data can enhance the accuracy of models.

To detect discrepancies while preserving privacy we use PSI, a method that safeguards the privacy of non-suspicious users' data maintained by the involved parties. The PSI outcomes serve as additional features in the FL model. Specifically, Srv and each client C_i participate in an instance of PSI, receiving a set of strings from Srv and the client. The PSI returns the intersection to C_i. For each user, C_i adds a binary feature b to its dataset (if not already present). If a user's details are in the intersection, b is set to 1; otherwise, it is set to 0. Figure 3 presents this procedure in detail. Hence, we not only employ PSI (as a subroutine in SecureBoost) for entity alignment, but we also leverage it to enhance the accuracy of the final model. Note that the outcome of the protocol in Figure 3 will be transmitted to FC in the second phase (collecting flags of suspicious users), presented below.

Collecting Flags of Users. Each user's sample may be accompanied by a flag whose value is independently computed and allocated by a client. For instance, in the context of financial transactions, for each user's

- **Parties:** Srv and C_i.
- **Input:**
  - ◇ Srv's input, for each user U, is a set T_{Srv} of strings (taken from a dataset DS_{Srv}), where each string has the form $t_{u,1}||t_{u,2}||...||t_{u,m}$ and $t_{u,1}$ is a user's unique ID.
  - ◇ C_i's input, for each user U, is a set T_{C_i} of strings (from its dataset DS_{C_i} of all users), where each string has the form $t_{u,1}||t_{u,2}||...||t_{u,m}$.
- **Output:** Updated dataset DS_{C_i}.

---

1. Srv and C_i invoke an stance of PSI protocol: $\mathcal{PSI}(T_{\mathrm{Srv}}, T_{\mathrm{C}_i}) \rightarrow T_{\cap}$.
2. Given $T_{\cap}$, C_i parses each element of $T_{\cap}$ as $t_{u,1}||t_{u,2}||...||t_{u,m}$.
3. If binary feature b is not in DS_{C_i}, then C_i adds b to each user's feature.
4. C_i sets b as follows. For every $t_{u,j} \in DS_{\mathrm{C}_i}$:
   - Sets $b = 1$, when $t_{u,j} \in S_{\cap}$.
   - Sets $b = 0$, otherwise.
5. C_i returns DS_{C_i}.

Fig. 3: PSI-based method to identify discrepancies.

account that a bank holds, there is a flag indicating whether the bank considers the account suspicious. This flag type offers extra information crucial for anomaly detection. Nevertheless, these flags are treated as private information and cannot be directly shared with Srv.

To align the flags with the Srv's dataset without revealing them, we rely on the following observation and idea. The key observation is that each user's sample, which is held by Srv and includes both sender and receiver clients, can be assigned an ID selected uniformly at random from a sufficiently large domain. In certain cases, such as financial transactions, each sample (representing a transaction) already comes with a random ID. As a random string, this ID divulges no specific information about a user's features. For each user's sample, Srv can generate this ID and share this ID (along with a unique feature in the sample) with the clients involved in that sample. Accordingly, if each client groups each ID with a set of binary flags and sends them to FC, FC cannot glean significant information about the user's features linked to those IDs. Based on this observation, we rely on the following idea to extract the flags.

For each user's sample, Srv sends the random ID and a unique feature of the user (e.g., their name or account number) to the related clients. The clients then use their sample information to group each ID with the correct user's flags. It sends this group to FC. When sending a flag for a user to FC, each client also sends to FC the flag b that it generated in Figure 3 (to detect discrepancies). Consequently, FC uses a set (that includes an ID and flags for each user) to create a dataset of flags. This dataset will then be used as the input data for the ML model.

The above private information retrieval mechanism is *highly computationally efficient.* This approach still may reveal certain information to the involved parties. Specifically (a) each client gains knowledge of some of their users that are in Srv's dataset, and (b) FC acquires information about which IDs originate from certain clients, enabling the calculation of the number of transactions between each pair of clients.

However, the privacy of sensitive information is preserved, as (i) each client remains unaware of details about other participating clients or users' features held at other clients and (ii) FC cannot identify the user involved in a sample. FC only has IDs and a set of flags for each ID. Consequently, FC cannot glean any information about a specific account.

As evident during the feature extraction, each client independently computes its message and sends it to FC without the need to coordinate with other clients. Hence, even if some clients choose not to send their messages, this phase is completed. This is in contrast to the solution proposed in [46] which cannot withstand clients' dropouts.

Extension. There is an alternative method for collecting flags, which involves employing an efficient *threshold privacy-preserving voting* mechanism introduced by Abadi and Murdoch [1]. This voting scheme enables the result recipient (e.g., FC or Srv) to ascertain whether, at the very least, a predefined threshold of the involved parties (e.g., clients) sets a user's flag to 1. Importantly, this process does not disclose any additional information, such as individual votes or the count of 1s or 0s, beyond the result to the result recipient. This scheme operates with high efficiency, as it avoids the need for public key cryptography. Integrating this scheme in *Starlit* has the potential to enhance the accuracy of the global model, as there is no longer a requirement to safeguard the flags with DP. A more in-depth analysis is needed to ensure that the system using this voting scheme can withstand potential client dropouts.

For the sake of simplicity, we have presented a solution focused on a single flag per sample. This solution can be readily generalized to situations where multiple flags are linked to a single sample. In this scenario involving multiple flags, when a client receives a sample's ID and the unique user's feature, it retrieves a vector of flags associated with that particular sample. Following applying either DP or the voting-based mechanism to the flags, the client then transmits the resultant outcome to FC.

8.2 Model Training and Inference

Following the feature extraction phase, Srv and FC jointly possess all the necessary data for training the anomaly detection model. Srv retains a dataset of samples, while FC possesses certain features of samples, i.e., discrepancies and flags (protected by DP).

This represents the VFL setting, where only Srv holds the labels to predict. This configuration allows for the utilization of various off-the-shelf protocols suitable for training an ML model, such as those presented in [15,18,22,27,38,49,55,62,70,72,76]. We use the SecureBoost algorithm (discussed in Section 4.4), which involves the exchange of encrypted (aggregate) gradients between Srv and FC during the training phase. Srv can decrypt the gradients to determine the best feature to split on. Once the model is trained, each party owns the part of the tree that uses the features it holds. Hence, when using the distributed inference protocol in [18], Srv coordinates with the FC to determine the split condition to be used.

9 Security of Starlit

In this section, we initially present formal definitions of the leakage that each party attains during the execution of *Starlit*. Subsequently, we formally state the security guarantee of *Starlit*.

Definition 3 (Srv–Side Leakage). *Let $\mathcal{L}$ be the leakage function defined in Relation 6 and inp be the input of all parties (as outlined in Section 6). Let DS_{C_i} be a dataset of users held by each C_i, and v_i be each dataset's size, i.e., $v_i = |DS_{C_i}|$, where $1 \leq i \leq n$. Moreover, let $\mathcal{W}(prm_1, prm_2) \rightarrow (l_1, l_2)$ be SecureBoost's leakage function (defined in Relation 4), where prm_1 is provided by Srv and prm_2 is given by FC. $\mathcal{W}$ returns l_1 to Srv and l_2 to FC. Then, leakage to Srv is defined as: $\mathcal{L}_1(inp) := \Big(v_1, ..., v_n, \mathcal{W}_1(prm_1, prm_2)\Big)$.*

Definition 4 (FC–Side Leakage). *Let $\mathcal{L}$ be the leakage function defined in Relation 6 and inp be the input of all parties. Also, let $s_i = |S_{C_i}|$, where S_{C_i} is a set of triples each of which has the form (ID, b, w), where ID represents a random ID of a sample, b is a binary flag for a feature's inconsistency (as described in Figure 3), w is another binary flag of the same sample (as described in Section 8.1). Moreover, let $\mathcal{W}(prm_1, prm_2) \rightarrow (l_1, l_2)$ be SecureBoost's leakage function (defined in Relation 4), where prm_1 is provided by Srv and prm_2 is given by FC. $\mathcal{W}$ returns l_1 to Srv and l_2 to FC. Then, leakage to FC is defined as: $\mathcal{L}_2(inp) := \Big(s_1, ..., s_n, \mathcal{W}_2(prm_1, prm_2)\Big)$.*

Definition 5 (C_i–Side Leakage). *Let $\mathcal{L}$ be the leakage function defined in Relation 6 and inp be the input of all parties. Moreover, let DS_{Srv} be Srv's dataset while DS_{C_i} be C_i's dataset. Also, let S_{Srv} be a set of pairs each of which has the form $(ID, feat_u)$, where ID represents a random ID of a sample and $feat_u$ refers to user U's unique feature, held by both Srv and C_i). Then, leakage to C_i is defined as: $\mathcal{L}_{i+2}(inp) := \Big((DS_{Srv} \cap DS_{C_i}), |DS_{Srv}|, S_{Srv}\Big)$.*

Theorem 1. *Let $\mathcal{F}$ be the functionality defined in Relation 5. Moreover, let $\mathcal{L}_1(inp), \mathcal{L}_2(inp)$, and $\mathcal{L}_{i+2}(inp)$ be the leakages defined in Definitions 3, 4, and 5 respectively. If PM is ϵ-differentially private and provides optimal flag privacy (w.r.t. Game presented in Figure 2), the SecureBoost and $\mathcal{PSI}$ are secure, then Starlit securely realizes $\mathcal{F}$, w.r.t. Definition 2.*

We prove the above theorem in Appendix B.

10 Implementation of Starlit

We carry out comprehensive evaluations to study *Starlit*'s performance from various aspects, including privacy-utility trade-off, efficiency, scalability, and choice of parameters. In the remainder of this section, we elaborate on the analysis.

10.1 The Experiment's Environment

We implement *Starlit* within an FL framework, called Flower (discussed in Section 4.5). We use Python programming language to implement *Starlit*. Experiments were run using AWS ECS cloud with docker containers with 56GB RAM and 8 Virtual CPUs. The FATE SecureBoost implementation uses multiprocessing to operate on table-like objects. We set the partitions setting to 5, which means operations on tables are performed with a parallelism of 5.

We adjusted and used the Python-based implementation of the efficient PSI introduced in [36]. We have run experiments to evaluate the performance of this PSI.

We conducted the experiments when each party's set's cardinality is in the range $[2^9, 2^{19}]$. Briefly, our evaluation indicates that the PSI's runtime increases from 0.84 to 367.93 seconds when the number of elements increases from 2^9 to 2^{19}. Appendix C presents further details on the outcome of the evaluation. Each instance of the PSI, for each account, takes as input string: $account_{number}||\ customer_{name}||street_{name}\ ||countryCity_{zipcode}$. The output of the PSI is received by the participating bank. To implement *Starlit*, we had to overcome a set of challenges, including the use of Flower and FATE. In Appendix F, we discuss these challenges in detail and explain how we addressed them.

10.2 Dataset

Our experiment involves the utilization of two synthetic datasets:

- Dataset 1: Synthetic dataset that simulates transaction data obtained from the global payment network of FSP (acting as Srv).
- Dataset 2: Synthetic dataset related to customers (or users), inclusive of their account information and flags, derived from the partner banks (or clients) of FSP.

Furthermore, the sizes of the datasets are as follows.

◇ FSP's training datasets, in total, contain about 4,000,000 rows.
◇ The banks' dataset includes around 500,000 rows.
◇ FSP's test dataset comprises about 700,000 rows.

Outline of Dataset 1. Each row (or sample) in this dataset corresponds to an individual transaction, signifying a payment from a sending bank to a receiving bank. Each transaction encapsulates details such as the originators and beneficiaries, sender and receiving banks, and payment corridor. The dataset spans approximately a month's worth of transactions and involves fifty institutions.

It contains various fields such as (a) MessageId: a globally unique identifier, (b) Sender: a bank sending the transaction, (c) Receiver: a bank receiving the transaction, and (d) OrderingAccount: an account identifier for the originating ordering entity. Appendix D provides detailed explanations of the fields contained in Dataset 1.

Outline of Dataset 2. The dataset comprises databases from various banks, encompassing information about their customers' accounts, e.g., the flags associated with each account. Initially, the data was unpartitioned, with all the banks' information consolidated into a single table.

This dataset contains various fields, for instance: (a) Account: an identifier for the account, (b) Name, name of the account, (c) Street: street address associated with the account, (d) CountryCityZip: remaining address details associated with the account, and (e) Flags: enumerated data type indicating potential issues or special features that have been associated with an account. Appendix E elaborates on the fields that Dataset 2 contains.

11 Empirical Results

Our evaluation of *Starlit* includes various perspectives (a) privacy-utility trade-off, discussed in Section 11.1, (b) efficiency and scalability, explored in Section 11.2, and (c) the choice of concrete parameters, covered in Section 11.3.

In this study, our focus does not lie on feature exploration or hyperparameter tuning to enhance model accuracy. Instead, we employ a straightforward approach, utilizing example features extracted from FSP, as provided in the data, in conjunction with four binary values derived from the banks' data. The features extracted from FSP for model training encompass the following: settlement amount, instructed amount, hour, sender hour frequency, sender currency frequency, sender currency amount average, and sender-receiver frequency. Additionally, we incorporate four binary flags, indicating the agreement between FSP and the banks on sender and receiver address details, as well as whether the sending and receiving accounts share the same flag for a given account.

11.1 Privacy-Utility Trade-off

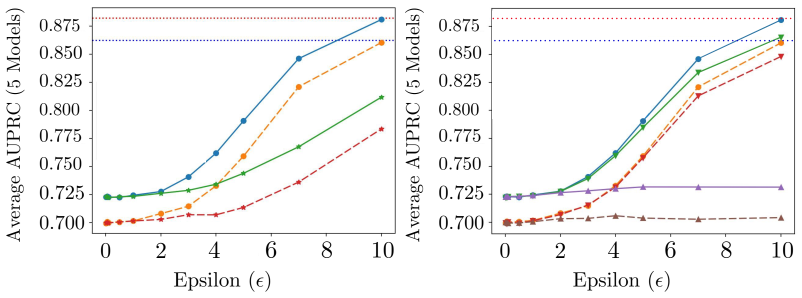
Baseline. To analyze the trade-off between utility and privacy, we establish a benchmark using a *centralized* model constructed within FSP. In this centralized model, all data from banks is revealed in plaintext. The same set of features listed above is extracted. We train a standard XGBoost model with 30 trees. We employ a 5-fold cross-validation with the average precision score as the metric. It is important to note that default values are utilized for all hyperparameters during the model training process.

Evaluation Procedure. The "Area Under the Precision-Recall Curve" (AUPRC) refers to a metric employed to evaluate the performance of an ML classification model. The unit of AUPRC is a value in the range $[0, 1]$, representing the area under the precision-recall curve. It measures the trade-off between precision and recall and provides a summary of the model's performance across different threshold values for classification. A higher AUPRC indicates better model performance, with 1 being the ideal value representing perfect precision and recall.

Starlit. In the evaluation of *Starlit*'s implementation, for analyzing AUPRC that can be achieved at a given level of privacy, we modify the flag values that banks send using DP and construct XGBoost models with these noisy features.

We use the same parameters as in the baseline model (30 trees and 5-fold cross-validation) and measure the average precision score for the final model on training and test data, averaging over 5 runs to account for the randomness of the privacy mechanism and the training process.

SecureBoost does the same computation as XGBoost while constructing the trees albeit on encrypted gradients. Hence, the additional cost will not be on accuracy but rather on performance (which we discuss in section 11.2). We also observed this to be the case from our experimental results.



(a) Rand. Response vs Laplace. (b) Rand. Response vs Asym. matrices.

Fig. 4: Plot(a) compares the effect on AUPRC of the model when using RR and Laplace mechanism with post-processing for achieving LDP. Plot(b) compares the effect on AUPRC when using RR and privacy mechanisms at the same value of ϵ but with the constraint of 10% less probability of converting 0 to 1 (1 to 0) than what is recommended by RR. In Plot(a), red dotted line: non-private-train, blue dotted line: non-private-test, solid blue line: RR-train, solid orange line: RR-test, solid green line: Laplace-train, and solid red line: Laplace-test. In Plot(b), red dotted line: non-private-train, blue dotted line: non-private-test, solid blue line: RR-train, solid orange line: RR-test, solid green line: 10% less 0-¿1 than RR-train, solid red line: 10% less 0-¿1 than RR-test, solid purple line: 10% less 1-¿0 than RR-train, and solid brown line: 10% less 1-¿0 than RR-test.

Key Takeaways. Figure 4 provides a summary of our utility-privacy trade-off analysis. Plot(a) in this figure compares the effect on AUPRC of the model when using Randomized Response (RR) and Laplace mechanism with post-processing for achieving LDP. Consistent with the optimality results presented in [65,33], RR offers a superior utility-privacy trade-off when compared to the Laplace mechanism. Both RR and the Laplace mechanism yield symmetric transformation matrices, meaning an equal probability for converting a 0 to 1 and a 1 to 0.

Plot(b) in Figure 4 illustrates the impact on AUPRC when employing RR and privacy mechanisms. This comparison is conducted at the same ϵ value, with the additional constraint of reducing the probability of converting 0 to 1 (and 1 to 0) by 10% compared to the recommendations provided by RR. These recommendations are determined using our game framework. The results demonstrate that even a slight increase in the probability of converting a zero flag to a non-zero value has a significant impact on the model's performance. This observation aligns with intuition, considering the substantial proportion of zero flag values in the dataset.

11.2 Efficiency and Scalability

Baseline. SecureBoost's training was configured with 10 trees, each with a depth of 3, a dataset sampling rate of 40%, and a "Gradient-based One Side Sampling" (GOSS) sampling of 0.1. Efficiency results for this baseline are provided in Table 2. This baseline is used to investigate various configurations' impact on efficiency.

Table 2: Efficiency metrics of the baseline. H represents time in hours and GB refers to gigabytes.

Efficiency Metric	Unit	Tree's depth	Result
AUPRC	–	3	0.4715
The total training time	H	3	1.1
The peak training memory usage	GB	3	12.38
The network disk volume usage	GB	3	4.98
The network file volume usage	GB	3	993

Starlit. We analyzed *Starlit*'s efficiency with different SecureBoost configurations. The evaluation's results are illustrated in Table 3. SecureBoost offers various options that can be employed to enhance efficiency in different settings. For instance, both direct sampling[12] and GOSS sampling offers a means to reduce network and memory overhead by decreasing the volume of data processed in each round of training.

The tree depth is also a crucial parameter for improving accuracy while maintaining an appropriate level of efficiency in terms of training time and memory consumption. Also, the integration of *Starlit* with FATE and Flower enables the splitting of large messages into chunks, facilitating more efficient processing. Furthermore, *Starlit* utilizes numerous Flower rounds, with a significant portion of the final rounds remaining empty due to the requirement of a pre-set round number by Flower. This situation has an impact on the network metrics.

Table 3: *Starlit*'s Runtime using various training parameters. In the table, H represents time in hours and GB refers to gigabyte. The row highlighted in yellow corresponds to the choice of parameters where AUPRC is at the highest level.

Efficiency Metric	Unit	Sampling Approach		Tree's depth	Max Message Size	Result
		Direct Sampling Rate	GOSS			
AUPRC	–	40%	0.1	3	100MB	0.4715
		100%	0.1	3	100MB	0.5786
		40%	Disabled	3	100MB	0.47
		40%	0.3	3	100MB	0.5965
		40%	0.1	5	100MB	0.652
		40%	0.1	3	1GB	0.4715
The total training time	H	40%	0.1	3	100MB	1.1
		100%	0.1	3	100MB	2.21
		40%	Disabled	3	100MB	2.83
		40%	0.3	3	100MB	1.5
		40%	0.1	5	100MB	1.13
		40%	0.1	3	1GB	1
The peak training memory usage	GB	40%	0.1	3	100MB	12.38
		100%	0.1	3	100MB	17.48
		40%	Disabled	3	100MB	18.39
		40%	0.3	3	100MB	13.66
		40%	0.1	5	100MB	16.4
		40%	0.1	3	1GB	12.22
The network disk volume usage	GB	40%	0.1	3	100MB	4.98
		100%	0.1	3	100MB	14.51
		40%	Disabled	3	100MB	16.61
		40%	0.3	3	100MB	7.84
		40%	0.1	5	100MB	5.1
		40%	0.1	3	1GB	4.34
The network file volume usage	GB	40%	0.1	3	100MB	993
		100%	0.1	3	100MB	1270
		40%	Disabled	3	100MB	1256
		40%	0.3	3	100MB	1035
		40%	0.1	5	100MB	1316
		40%	0.1	3	1GB	927

Furthermore, *Starlit*'s scalability is attributed to its design choice, which maintains the model training phase's independence from the number of banks. In contrast, the feature extraction phase's computational complexity scales linearly with the number of banks and the suspicious accounts identified by FSP.

Starlit extensively employs CPU resources, primarily driven by the CPU-intensive nature of the underlying SecureBoost training. This demand arises from the encryption necessary for securing the gradient in the algorithm. Refer to Figure 7 in Appendix G for the CPU utilization details of *Starlit*.

[12] This direct sampling approach is taken to reduce the training time. It works by randomly using only the stated fraction of the training set.

11.3 Choice of Parameters

Our ultimate selection of parameters was informed by (i) the privacy-utility analysis in Section 11.1, specifically in the selection of ϵ and (ii) the efficiency analysis in Section 11.2, pertaining to the determination of model hyper-parameters, such as AUPRC, the number of trees, and sampling rate.

For the centralized solution, as far as possible, we matched the parameters used in the centralized XG-Boost model with those used in the federated solution. Specifically for the number of trees, tree depth, and L2 regularization parameter. The parameters are set as follows: (a) in both the centralized and federated settings: number of trees = 10, L2 regularization = 0.1, and (b) in the federated setting: $\epsilon = 10$.

11.4 Contrasting Starlit with the Baseline

In this section, we provide a brief comparison of *Starlit*'s performance with the baseline scenario, drawing insights from the information presented in Tables 2 and 3.

Starlit and the baseline achieve the same level of AUPRC when *Starlit*'s (i) direct sampling rate is 40%, (ii) the tree's depth is 3, and (iii) GOSS is not disabled. This is indicated in Figure 5. In the case where direct sampling rate = 40% and tree's depth = 3, regarding:

- ◇ *The total training time*: *Starlit* and the baseline have similar performance, except for the case when GOSS in *Starlit* is disabled. In this case, *Starlit* underperforms the baseline by a factor of 2.5.
- ◇ *The peak training memory usage*: *Starlit* and the baseline have similar performance except for the cases where (i) GOSS in *Starlit* is disabled and (ii) GOSS is 0.3. In these cases, *Starlit* underperforms the baseline by at most a factor of 1.4.
- ◇ *The network disk volume usage*: *Starlit* and the baseline have similar performance when GOSS = 0.1 and max message size = 100 MB. However, when max message size = 1 GB, *Starlit* outperforms the baseline by a factor of 1.1. In the rest of the cases, *Starlit* underperforms the baseline by at most a factor of 3.3.
- ◇ *The network file volume usage*: *Starlit* and the baseline have similar performance when direct sampling rate = 40%, GOSS = 0.1, and max message size = 100 MB. However, when max message size = 1 GB, *Starlit* outperforms the baseline by a factor of 1.07. In the rest of the cases, the baseline outperforms *Starlit* by at most a factor of 1.27.

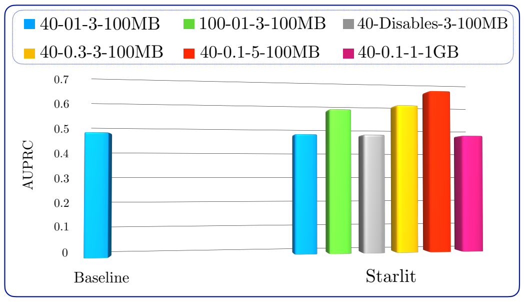


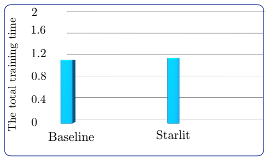
Fig. 5: Comparing the AUPRC of the baseline and different settings of *Starlit*. A bar's label for *Starlit* is a concatenation of (1) direct sampling rate, (2) GOSS, (3) tree's depth, and (4) maximum message size.

As illustrated in Table 3, *Starlit* achieves its highest AUPRC level (i.e., 0.652) when the tree's depth is set to 5. Remarkably, in this instance, *Starlit*'s AUPRC surpasses even the baseline setting (i.e., 0.652 versus 0.4715).

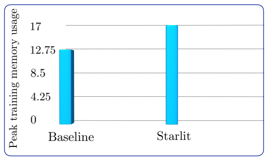
Having identified the parameter that yields the highest AUPRC in *Starlit*, i.e., when the tree's depth is set to 5, we proceed to compare *Starlit*'s other efficiency metrics for only that parameter(s).

- *The total training time*: As Figure 6a demonstrates, *Starlit*'s training time is almost the same as the baseline's training time, i.e., 1.13 compared to 1.10.
- *The peak training memory usage*: As Figure 6b illustrates, under this metric, *Starlit* underperforms the baseline by a factor of 1.3, i.e., 16.4 versus 12.38.
- *The network disk volume usage*: As Figure 6c shows, *Starlit* and the baseline demonstrate similar performance under this metric, i.e., 5.1 compared to 4.98.
- *The network file volume usage*: As Figure 6d demonstrates, under this metric, *Starlit* underperform the baseline by a factor of 1.3, i.e., 1316 versus 993.

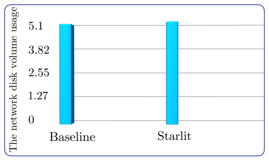
Hence, when the tree's depths in *Starlit* and baseline are set to 5 and 3 respectively, then *Starlit* can attain a superior AUPRC level compared to the baseline. However, in this setting, *Starlit* would impose approximately 1.3 times higher cost than the baseline does.



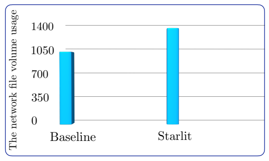
(a) Comparison between the total training time of the baseline and *Starlit*.



(b) Comparison between peak training memory usage of baseline and *Starlit*.



(c) Comparison between network disk volume usage of the baseline and *Starlit*.



(d) Comparison between network file volume usage of the baseline and *Starlit*.

Fig. 6: In this figure, *Starlit*'s: sampling rate = 40, GOSS = 0.1, tree's depth = 5, and max message size = 100MB.

12 Further Applications of Starlit

Starlit demonstrates notable adaptability across a diverse array of collaborative analysis tasks, as outlined in the subsequent sections. Additionally, *Starlit*'s components may have other applications, as discussed in Appendix G.1.

12.1 Mitigating Terrorism

The effective response to the challenge of counter-terrorism calls for the establishment of collaborative partnerships among diverse crime agencies at both national and international levels. While various countries may occasionally exchange some of their intelligence in plaintext, factors such as mishandling or leakage of (top) secret data [48,77] can dissuade them from doing so.

The facilitation of such collaboration in a *privacy-preserving* manner can be achievable through the utilization of *Starlit*. By leveraging *Starlit*, crime agencies can engage in joint efforts, sharing their knowledge to create a unified intelligence model [57]. This collaborative approach involves the integration of data from different crime agencies, allowing for a more comprehensive understanding of potential threats. Specifically, each crime agency can augment its own dataset with flags assigned by counterpart agencies to individuals deemed suspicious, whether citizens or tourists. These flags are based on factors such as a person's history of violence or associations with known terrorist organizations.

In this framework, *Starlit* can also empower crime agencies to identify disparities in the information provided by certain individuals across different countries' crime agencies. The exchange of information facilitated by *Starlit* in real-time contributes to a more comprehensive and accurate understanding of individuals with potential security implications.

12.2 Enhancing Digital Health

While digital healthcare offers numerous advantages, addressing one of its primary challenges, namely data privacy, is crucial [23]. *Starlit* holds the promise of delivering advantages to the digital health sector. For example, it can empower hospitals, local medical practitioners, and wearable devices that gather diverse patient data. This data may encompass crucial indicators/flags revealing a patient's susceptibility to chronic diseases or their ongoing management of such conditions [29]. *Starlit* enables the identification of common patients among each of these entities and the active party responsible for developing a global model. They can then construct a model using the shared data and flags.

Within this framework, *Starlit* is capable of pinpointing disparities in features attributed to overlapping patients among different entities. For example, it can ascertain whether each involved party prescribed distinct medications for the same patient, afflicted with the same disease, at varying points in time. This unique capability plays a pivotal role in the detection of medication errors and streamlines the process of medication reconciliation [9], all while upholding the vital principle of privacy.

12.3 Detecting Benefit Fraud

This process aims to empower different entities, such as government agencies, social service organizations, and different countries' banks to collaboratively develop a model to deal with benefit fraud [26], which is against the law in certain countries. In this particular scenario, *Starlit* serves as the enabling technology, facilitating a government organization to develop a collaborative model with both national and international banks. Each participating bank can assign a distinctive flag to each customer's account, indicating whether the account is receiving social benefits from the respective country or is deemed suspicious.

Much like its original application, *Starlit* excels in identifying disparities in the information provided by a customer to different entities. In this context, technology plays a crucial role in preventing fraud or misuse, ensuring that resources are allocated with fairness and appropriateness as top priorities.

13 Conclusion and Future Work

In this work, we introduced *Starlit*, a scalable privacy-preserving and demonstrated its applications in dealing with financial fraud, mitigating terrorism, and improving digital health. We formally defined and proved the security of *Starlit* in the simulation-based model. To formally capture the security of *Starlit*, we have defined a set of leakage functions that may hold independent significance. We implemented *Starlit* and conducted a comprehensive analysis of its performance and accuracy, using synthetic data provided by one of the key players facilitating financial transactions worldwide.

In any secure FL, the output inevitably discloses certain information about participating parties' private inputs. This fact may dissuade some parties with sensitive and valuable inputs from engaging in the FL process, particularly when they lack interest in the outcome. Future research could enhance *Starlit* by rewarding active contributors, which could also bridge the gap between the data market [25,34,61] and FL. Another avenue is strengthening *Starlit*'s security against fully malicious parties.

References

1. Abadi, A., Murdoch, S.J.: Payment with dispute resolution: A protocol for reimbursing frauds victims. In: ACM Asia CCS (2023)
2. Afriyie, J.K., Tawiah, K., Pels, W.A., Addai-Henne, S., Dwamena, H.A., Owiredu, E.O., Ayeh, S.A., Eshun, J.: A supervised machine learning algorithm for detecting and predicting fraud in credit card transactions. Decision Analytics Journal (2023)
3. Agrawal, R., Evfimievski, A., Srikant, R.: Information sharing across private databases. In: Proceedings of the 2003 ACM SIGMOD international conference on Management of data. pp. 86–97 (2003)
4. AL-Abri, H.H., Kumar, B., Mani, J.: Improving fraud detection mechanism in financial banking sectors using data mining techniques. In: Progress in Advanced Computing and Intelligent Engineering (2021)

5. Arora, S.S., Beams, A., Chatzigiannis, P., Meiser, S., Patel, K., Raghuraman, S., Rindal, P., Shah, H., Wang, Y., Wu, Y., Yang, H., Zamani, M.: Privacy-preserving financial anomaly detection via federated learning & multi-party computation. CoRR **abs/2310.04546** (2023)
6. Askari, S.M.S., Hussain, M.A.: IFDTC4.5: intuitionistic fuzzy logic based decision tree for e-transactional fraud detection. J. Inf. Secur. Appl. (2020)
7. Authority, F.C.: FCA glossary (2021), `https://www.handbook.fca.org.uk/handbook/glossary/G3566a.html`
8. Aziz, R.M., Mahto, R., Goel, K., Das, A., Kumar, P., Saxena, A.: Modified genetic algorithm with deep learning for fraud transactions of ethereum smart contract. Applied Sciences (2023)
9. Barnsteiner, J.H.: Medication reconciliation (2008), `https://www.ncbi.nlm.nih.gov/books/NBK2648/`
10. Bergin, T., Layne, N.: Special report: Cyber thieves exploit banks' faith in swift transfer network. Reuters (2016)
11. Beutel, D.J., Topal, T., Mathur, A., Qiu, X., Parcollet, T., Lane, N.D.: Flower: A friendly federated learning research framework. CoRR (2020)
12. Bonawitz, K.A., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H.B., Patel, S., Ramage, D., Segal, A., Seth, K.: Practical secure aggregation for privacy-preserving machine learning. In: CCS. ACM (2017)
13. Bonneau, J., Preibusch, S.: The password thicket: Technical and market failures in human authentication on the web. In: WEIS (2010)
14. Cao, L.: AI in finance: challenges, techniques, and opportunities. ACM CSUR (2022)
15. Ceballos, I., Sharma, V., Mugica, E., Singh, A., Roman, A., Vepakomma, P., Raskar, R.: Splitnn-driven vertical partitioning. arXiv preprint arXiv:2008.04137 (2020)
16. Chen, T., Guestrin, C.: Xgboost: A scalable tree boosting system. In: ACM SIGKDD (2016)
17. Chen, W., Zheng, Z., Ngai, E.C.H., Zheng, P., Zhou, Y.: Exploiting blockchain data to detect smart ponzi schemes on ethereum. IEEE Access (2019)
18. Cheng, K., Fan, T., Jin, Y., Liu, Y., Chen, T., Papadopoulos, D., Yang, Q.: Secureboost: A lossless federated learning framework. IEEE Intelligent Systems (2021)
19. Confirmation of Payee Team: Confirmation of payee- response to consultation cp20/1 and decision on varying specific direction 10 (2020), `https://t.ly/xiJQM`
20. Duchi, J.C., Jordan, M.I., Wainwright, M.J.: Local privacy, data processing inequalities, and statistical minimax rates. arXiv:1302.3203 (2013)
21. Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating noise to sensitivity in private data analysis. In: Theory of Cryptography (2006)
22. Fang, W., Zhao, D., Tan, J., Chen, C., Yu, C., Wang, L., Wang, L., Zhou, J., Zhang, B.: Large-scale secure xgb for vertical federated learning. In: ACM CIKM (2021)
23. Filkins, B.L., Kim, J.Y., Roberts, B., Armstrong, W., Miller, M.A., Hultner, M.L., Castillo, A.P., Ducom, J.C., Topol, E.J., Steinhubl, S.R.: Privacy and security in the era of digital health: what should translational researchers know and do about it? American journal of translational research (2016)
24. Goldreich, O.: The Foundations of Cryptography - Volume 2, Basic Applications. Cambridge University Press (2004)
25. Golob, S., Pentyala, S., Dowsley, R., David, B., Larangeira, M., De Cock, M., Nascimento, A.: A decentralized information marketplace preserving input and output privacy (2023)
26. Government, U.: Benefit fraud (2023), `https://www.gov.uk/benefit-fraud`
27. Hardy, S., Henecka, W., Ivey-Law, H., Nock, R., Patrini, G., Smith, G., Thorne, B.: Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. arXiv preprint arXiv:1711.10677 (2017)
28. Hassan, M.U., Rehmani, M.H., Chen, J.: Anomaly detection in blockchain networks: A comprehensive survey. IEEE Communications Surveys & Tutorials (2022)
29. Hosseini, M.M., Hosseini, S.T.M., Qayumi, K., Hosseinzadeh, S., Tabar, S.S.S.: Smartwatches in healthcare medicine: assistance and monitoring; a scoping review. BMC Medical Informatics Decis. Mak. (2023), `https://doi.org/10.1186/s12911-023-02350-w`
30. Jacomme, C., Kremer, S.: An extensive formal analysis of multi-factor authentication protocols. ACM Transactions on Privacy and Security (TOPS) (2021)
31. Jung, E., Le Tilly, M., Gehani, A., Ge, Y.: Data mining-based Ethereum fraud detection. In: IEEE International Conference on Blockchain (2019)
32. Kadhe, S.R., Ludwig, H., Baracaldo, N., King, A., Zhou, Y., Houck, K., Rawat, A., Purcell, M., Holohan, N., Takeuchi, M., Kawahara, R., Drucker, N., Shaul, H., Kushnir, E., Soceanu, O.: Privacy-preserving federated learning over vertically and horizontally partitioned data for financial anomaly detection. CoRR **abs/2310.19304** (2023)

33. Kairouz, P., Oh, S., Viswanath, P.: Extremal mechanisms for local differential privacy. Advances in neural information processing systems (2014)
34. Koch, K., Krenn, S., Marc, T., More, S., Ramacher, S.: KRAKEN: a privacy-preserving data market for authentic data. In: DE. ACM (2022)
35. Kolesnikov, V., Kumaresan, R.: Improved OT extension for transferring short secrets. In: CRYPTO (2013)
36. Kolesnikov, V., Kumaresan, R., Rosulek, M., Trieu, N.: Efficient batched oblivious PRF with applications to private set intersection. In: CCS (2016)
37. Li, T., Sahu, A.K., Zaheer, M., Sanjabi, M., Talwalkar, A., Smith, V.: Federated optimization in heterogeneous networks. In: MLSys (2020)
38. Liu, Y., Zhang, X., Wang, L.: Asymmetrical vertical federated learning. arXiv preprint arXiv:2004.07427 (2020)
39. Lv, B., Cheng, P., Zhang, C., Ye, H., Meng, X., Wang, X.: Research on modeling of e-banking fraud account identification based on federated learning. In: IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress, DASC/PiCom/CBDCom/CyberSciTech 2021, Canada, October 25-28, 2021. IEEE (2021)
40. McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Artificial intelligence and statistics. PMLR (2017)
41. McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: AISTATS (2017)
42. McMahan, H.B., Moore, E., Ramage, D., y Arcas, B.A.: Federated learning of deep networks using model averaging. CoRR **abs/1602.05629** (2016)
43. Murdoch, S.J., Abadi, A.: A forward-secure efficient two-factor authentication protocol. arXiv preprint arXiv:2208.02877 (2022)
44. Nguyen, D.C., Pham, Q., Pathirana, P.N., Ding, M., Seneviratne, A., Lin, Z., Dobre, O.A., Hwang, W.: Federated learning for smart healthcare: A survey. ACM Comput. Surv. (2023)
45. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: EUROCRYPT. pp. 223–238 (1999)
46. Qiu, X., Pan, H., Zhao, W., Ma, C., de Gusmão, P.P.B., Lane, N.D.: Efficient vertical federated learning with secure aggregation. CoRR (2023)
47. Reddi, S.J., Charles, Z., Zaheer, M., Garrett, Z., Rush, K., Konečný, J., Kumar, S., McMahan, H.B.: Adaptive federated optimization. CoRR (2020)
48. Reuters: What is known about latest leak of u.s. secrets (2023), `https://www.reuters.com/world/us/what-is-known-about-latest-leak-us-secrets-2023-04-10/`
49. Romanini, D., Hall, A.J., Papadopoulos, P., Titcombe, T., Ismail, A., Cebere, T., Sandmann, R., Roehm, R., Hoeh, M.A.: Pyvertical: A vertical federated learning framework for multi-headed splitnn. arXiv preprint arXiv:2104.00489 (2021)
50. Saheed, Y.K., Baba, U.A., Raji, M.A.: Big data analytics for credit card fraud detection using supervised machine learning models. In: Big Data Analytics in the Insurance Market (2022)
51. Shokri, R.: Privacy games: Optimal user-centric data obfuscation. arXiv preprint arXiv:1402.3426 (2014)
52. Shokri, R., Theodorakopoulos, G., Troncoso, C., Hubaux, J.P., Le Boudec, J.Y.: Protecting location privacy: optimal strategy against localization attacks. In: ACM CCS (2012)
53. Sinigaglia, F., Carbone, R., Costa, G., Zannone, N.: A survey on multi-factor authentication for online banking in the wild. Computers & Security (2020)
54. Srokosz, M., Bobyk, A., Ksiezopolski, B., Wydra, M.: Machine-learning-based scoring system for antifraud cisirts in banking environment. Electronics (2023)
55. Sun, J., Yang, X., Yao, Y., Zhang, A., Gao, W., Xie, J., Wang, C.: Vertical federated learning without revealing intersection membership. arXiv preprint arXiv:2106.05508 (2021)
56. Suzumura, T., Zhou, Y., Barcardo, N., Ye, G., Houck, K., Kawahara, R., Anwar, A., Stavarache, L.L., Klyashtorny, D., Ludwig, H., Bhaskaran, K.: Towards federated graph learning for collaborative financial crimes detection. CoRR (2019)
57. The Royal Society: From privacy to partnership: The role of privacy enhancing technologies in data governance and collaborative analysis, `https://royalsociety.org/-/media/policy/projects/privacy-enhancing-technologies/From-Privacy-to-Partnership.pdf?la=en-GB&hash=4769FEB5C984089FAB52FE7E22F379D6`
58. The UK and US Goverments: The UK and US governments, UK-US prize challenges accelerating the adoption and development of privacy-enhancing technologies (2022), `https://tinyurl.com/4ntm2xv2`

59. The UK Government: At summit for democracy, the United Kingdom and the United States announce winners of challenge to drive innovation in Privacy-Enhancing Technologies that reinforce democratic values (2023), `https://www.gov.uk/government/news/at-summit-for-democracy-the-united-kingdom-and-the-united-states-announce-winners-of-challenge-to-drive-innovation-in-privacy-enhancing-technologies`
60. The White House: At summit for democracy, the United States and the United Kingdom announce winners of challenge to drive innovation in Privacy-Enhancing Technologies that reinforce democratic values (2023), `https://www.whitehouse.gov/ostp/news-updates/2023/03/31/us-uk-annouce-winners-innovation-pets-democratic-values/`
61. Tian, Z., Liu, J., Li, J., Cao, X., Jia, R., Ren, K.: Private data valuation and fair payment in data marketplaces. CoRR (2022)
62. Tian, Z., Zhang, R., Hou, X., Liu, J., Ren, K.: Federboost: Private federated learning for gbdt. arXiv preprint arXiv:2011.02796 (2020)
63. UK Home Office and Ministry of Justice: Data sharing for the criminal justice system guidance (2023), `https://assets.publishing.service.gov.uk/media/652cefa56b6fbf000db7567a/data-sharing-guidance-criminal-justice-system.pdf`
64. Wan, L., Ng, W.K., Han, S., Lee, V.C.S.: Privacy-preservation for gradient descent methods. In: Proceedings of the 13th ACM KDD. ACM (2007)
65. Wang, Y., Wu, X., Hu, D.: Using randomized response for differential privacy preserving data collection. In: EDBT/ICDT Workshops. vol. 1558, pp. 0090–6778 (2016)
66. White, O., Madgavkar, A., Townsend, Z., Manyika, J., Olanrewaju, T., Sibanda, T., Kaufman, S.: Financial data unbound: The value of open data for individuals and institutions. McKinsey Global Institute (2021)
67. Winder, D.: This is how hackers accessed 34,942 paypal accounts. Forbes (2023)
68. Wu, N., Zhang, N., Wang, W., Fan, L., Yang, Q.: Fadman: Federated anomaly detection across multiple attributed networks. CoRR (2022)
69. Wu, X., Du, S.: An optimized association rules mining framework and its application in chinese social insurance fund data auditing. SSRN (2022)
70. Wu, Y., Cai, S., Xiao, X., Chen, G., Ooi, B.C.: Privacy preserving vertical federated learning for tree-based models. arXiv preprint arXiv:2008.06170 (2020)
71. Xiuguo, W., Shengyong, D.: An analysis on financial statement fraud detection for chinese listed companies using deep learning. IEEE Access **10**, 22516–22532 (2022)
72. Xu, R., Baracaldo, N., Zhou, Y., Anwar, A., Joshi, J., Ludwig, H.: Fedv: Privacy-preserving federated learning over vertically partitioned data. In: Runhua (2021)
73. Yang, Q., Liu, Y., Chen, T., Tong, Y.: Federated machine learning: Concept and applications. ACM Trans. Intell. Syst. Technol. (2019)
74. Yang, W., Zhang, Y., Ye, K., Li, L., Xu, C.Z.: FFD: A federated learning based method for credit card fraud detection. In: Big Data (2019)
75. Yao, A.C.: Protocols for secure computations (extended abstract). In: FSC (1982)
76. Zhang, C., Li, S., Xia, J., Wang, W., Yan, F., Liu, Y.: {BatchCrypt}: Efficient homomorphic encryption for {Cross-Silo} federated learning. In: USENIX ATC (2020)
77. Zurcher, A.: How trump, biden and clinton secret files cases compare (2023), `https://www.bbc.co.uk/news/world-us-canada-64230040`

A Related Work

In this section, we discuss the approaches used to deal with fraudulent financial transactions. This includes:

- A centralized approach, where a client (e.g., bank) either trains its model based on the history of financial transactions it holds, or a centralized party (server) collects data in plaintext from different sources without considering the privacy of different parties. This approach is covered in Section A.1.
- A decentralized approach that involves multiple entities, such as different banks and financial service providers (e.g., Visa or SWIFT), where parties collaboratively train their models while preserving the privacy of input data. This approach is discussed in Section A.2.
- Other none AI-based schemes, which is covered in Section A.3.

A.1 Centralized Approaches Without Privacy Support

Afriyie *et al.* [2] studied the performance of three different machine learning models, logistic regression, random forest, and decision trees to classify, predict, and detect fraudulent credit card transactions. They conclude that random forest produces maximum accuracy in predicting and detecting fraudulent credit card transactions. Askari and Hussain [6] aim to achieve the same goal as Afriyie *et al.* (i.e., to classify, predict, and detect fraudulent credit card transactions) using "Fuzzy-ID3" (Interactive Dichotimizer 3). Saheed *et al.* [50] examined machine learning models for predicting credit card fraud and proposed a new model for credit card fraud detection by relying on principal component analysis and supervised machine learning techniques such as K-nearest neighbor, ridge classifier, and gradient boost.

Srokosz *et al.* [54] designed a mechanism to improve the rules-based fraud prevention systems of banks. The proposed mechanism is a rating system that uses unsupervised machine learning and provides early warnings against financial fraud. The proposed system basically distinguishes between rogue and legitimate bank account login attempts by examining customer logins from the banking transaction system. The suggested method enhances the organization's rule-based fraud prevention system. Al-Abri *et al.* [4] proposed a data mining mechanism based on logistic regression to detect irregular transactions, implemented the solution, and analyzed its performance. We refer readers to [14] for a survey of AI-based mechanisms used in traditional financial institutions.

Researchers have also focused on financial reports/statements of companies and developed financial statement fraud detection mechanisms for Chinese listed companies using deep learning, e.g., in [69,71]. Their threat model and solution considered companies as misbehaving actors who want to convince investors, auditors, or governments that they have followed the regulations and that their financial statements are valid.

Researchers also proposed data mining-based mechanisms to provide a detection model for Ponzi schemes on the Ethereum blockchain, e.g., see [31,17]. Very recently, Aziz *et al.* [8] proposed an optimization strategy for deep learning classifiers to identify fraudulent Ethereum transactions and Ponzi schemes. We refer readers to [28] for a survey of anomaly detection in the blockchain network. Note that the proposed solutions for blockchain cannot be directly applied to the conventional banking system as it is in a different setting.

A.2 Distributed Approaches With Privacy Support

Generally, FL can be categorized into three classes [73], according to how data is partitioned:

- Horizontal Federated Learning (HFL).
- Vertical Federated Learning (VFL).
- Federated Transfer Learning (FTL).

HFL refers to the FL setting where participants share the same feature space while holding different samples. On the other hand, VFL refers to the FL setting where datasets share the same samples while holding different features. FTL refers to the FL setting where datasets differ in both feature and sample spaces with limited overlaps. For the remainder of this section, our focus will mainly be on FL-based schemes developed to deal with fraudulent transactions.

HFL-Based Approaches. Suzumura *et al.* [56] developed an ML-based privacy-preserving mechanism to share key information across different financial institutions. This solution builds ML models by leveraging FL and graph learning. This would ultimately allow for global financial crime detection while preserving the privacy of financial organizations and their customers' data. Given that federated graph learning involves collaborative training on a shared graph structure (common set of features) distributed across multiple parties, this proposed solution aligns with the characteristics of HFL. Moreover, Yang *et al.* [74] proposed an FL-based method using "Federated Averaging" [41] to train a model that can detect credit card fraud. Similar to the scheme in [56], this method falls under the HFL category.

Unlike the schemes explained above, *Starlit* deals with the data that have been partitioned both vertically and horizontally.

VFL-Based Approaches. Simultaneously with our solution, another approach has been developed by Qiu *et al.* [46]. This alternative relies on neural networks and aims to deal with financial fraud. It strives for computational efficiency primarily through the use of symmetric key primitives. The scheme incorporates the elliptic-curve Diffie-Hellman key exchange and one-time pads to secure exchanged messages during the model training phase. This solution has also been implemented and subjected to performance evaluation.
Starlit versus the Scheme of Qiu et al. The latter scheme requires each client (e.g., bank) to possess knowledge of the public key of every other client and compute a secret key for each through the elliptic-curve Diffie-Hellman key exchange scheme. Consequently, this approach imposes $O(n)$ modular exponentiation on each client, resulting in the protocol having a complexity of $O(n^2)$, where n represents the total number of clients. In contrast, in *Starlit*, each client's complexity is independent of the total number of clients and each client does not need to know any information about other participating clients. Moreover, the scheme proposed in [46] assumes the parties have already performed the identity alignment phase, therefore, the implementation, performance evaluation, and security analysis exclude the identity alignment phase.

Furthermore, the scheme in [46] fails to terminate successfully even if only one of the clients neglects to transmit its message. In this scheme, each client, utilizing the agreed-upon key with every other client, masks its outgoing message with a vector of pseudorandom blinding factors. The expectation is that the remaining clients will mask their outgoing messages with the additive inverses of these blinding factors. These blinding factors are generated such that, when all outgoing messages are aggregated, the blinding factors cancel each other out. Nevertheless, if one client fails to send its masked message, the aggregated messages of the other clients will still contain blinding factors, hindering the training on correct inputs. The solution proposed in [12], based on threshold secret sharing, can address this issue. However, incorporating such a patch would introduce additional computation and communication overheads. In contrast, *Starlit* does not encounter this limitation. This is because the message sent by each client is independent of the messages transmitted by the other clients.

Lv *et al.* [39] introduced a VFL-based approach to identify black market fraud accounts before fraudulent transactions occur. This approach aims to guarantee the safety of funds when users transfer funds to black market accounts, enabling the financial industry to utilize multi-party data more efficiently. The scheme involves data provided by financial and social enterprises, encompassing financial features extracted from a bank such as mobile banking login logs, and account transaction information. Social features include the active cycle corresponding to the mobile phone number, the count of malicious apps, and the frequency of visits to malicious sites. The approach utilizes *insecure* hash-based PSI for identity alignment.

This scheme differs from *Starlit* in a couple of ways: (i) *Starlit* operates in a multi-party setting, where various clients contribute their data, in contrast to the aforementioned scheme, which involves only two parties and cannot be directly applied to the multi-client setting, and (ii) *Starlit* deals with the data that has been partitioned both horizontally and vertically, whereas the above scheme focuses only on vertically partitioned data.

FL-Based Solutions to Detect Fraudulent Financial Transactions when Datasets are Vertical and Horizontally Partitioned. Recently, to combat the global challenge of organized crime, such as money laundering, terrorist financing, and human trafficking, the UK and US governments launched a set of prize challenges [58]. This competition encouraged innovators to develop technical solutions to identify suspicious bank account holders while preserving the privacy of honest account holders by relying on FML and cryptography approaches. This underscores the importance the distributed privacy-preserving financial data analytics for governments.

Very recently, in parallel with our work, Arora *et al.* [5] introduced an FL-based approach for detecting anomalous financial transactions. This methodology was specifically devised for the aforementioned prize challenge and relied on oblivious transfer, secret sharing, differential privacy, and multi-layer perception. The authors have implemented the solution and conducted a thorough analysis of its performance and accuracy.
Starlit versus the Scheme of Arora et al. The latter assumes that the ordering bank never allows a customer with a dubious account to initiate any transactions, but it permits the same account to receive money. In

simpler terms, this scheme exclusively deals with frozen accounts, limiting its applicability. This approach will exempt the ordering bank from participating in MPC, which ultimately results in lower overhead (compared to the case where the ordering bank had to be involved in MPC).

In the real world, users' accounts might be deemed suspicious (though not frozen), yet they can still conduct financial transactions within their bank. However, the bank may handle such accounts more cautiously than other non-suspicious accounts. In contrast, in the context of dealing with financial transactions, *Starlit* does not place any assumption on how a bank treats a suspicious account.

Furthermore, unlike the scheme proposed in [5], which depends on an ad-hoc approach to preserve data privacy during training, *Starlit*, employs SecureBoost, which is a well-known scheme extensively utilized and analyzed in the literature. Thus, compared to the scheme in [5], *Starlit* considers a more generic scenario and relies on a more established scheme for VFL.

Kadhe *et al*. [32] proposed a privacy-preserving anomaly detection scheme, that relies on fully homomorphic encryption (highly computationally expensive), DP, hash tables, and secure multi-party computation. Similarly, the authors have implemented their solution and analyzed its performance.
Starlit versus the Scheme of Kadhe et al. The scheme in [32] heavily relies on fully homomorphic encryption. In this setting, all parties need to perform fully homomorphic operations, by performing either computation on the outputs of the homomorphic encryption or encrypting and decrypting ciphertexts. This will affect both the scalability and efficiency of this scheme. In contrast, *Starlit* refrains from using any *fully* homomorphic scheme.

All of the above solutions share another shortcoming, they lack formal security definitions and proofs of the proposed systems.

VFL-Based Solutions to Detect Anomaly in Other Contexts We *et al*. [68] proposed a privacy-preserving solution specifically designed to detect anomalies within multiple attributed networks in a privacy-preserving manner, e.g., to detect abnormal IPs or predict cyber attacks. The authors have evaluated the proposed solution using various real datasets, including Computer networks, Car-hailing, Bicycle-sharing, Subway traffic, and Point-of-Interest datasets. However, this solution lacks generality and has been designed specifically for a very specific setting.

A.3 Solutions Beyond AI

There have also been efforts to deal with banking fraud, by using alternative (none AI-based) prevention mechanisms, such as Multi-Factor Authentication (M-FA) and Confirmation of Payee (CoP) schemes, outlined below.

For bank users to prove their identity to remote service providers or banks, they provide a piece of evidence, called an "authentication factor". Authentication factors can be based on (i) knowledge factors, e.g., PIN or password, (ii) possession factors, e.g., access card or physical hardware token, or (iii) inherent factors, e.g., fingerprint. Knowledge factors are still the most predominant factors used for authentication [13,43]. The knowledge factors themselves are not strong enough to adequately prevent impersonation [30,53]. M-FA methods that depend on more than one factor are more difficult to compromise than single-factor methods. Thus, in general, M-FA methods lower the chance of fraudsters who want to gain unauthorized access to users' online banking accounts.

Confirmation of Payee (CoP) is a name-checking service for UK-based payments [19,1]. It provides customers greater assurance that they are sending payments to the intended recipient, helping to avoid making accidental, misdirected payments to the wrong account holder, as well as providing another layer of protection in the fight against fraud, especially Authorised Push Payment fraud [7]. In short, CoP ensures that a money recipient's details (inserted by the money sender) match the record held by the recipient's bank.

B Proof of Security

Proof. We prove Theorem 1 by examining the scenario in which each party is corrupt, one at a time.

B.1 Corrupt Srv

In the real execution, the view of Srv is defined as follows:

$$\mathsf{View}^{\mathcal{A},\textit{Starlit}}_{\mathrm{Srv}}\Big(\underbrace{prm_{\mathrm{Srv}}, DS_{\mathrm{Srv}}}_{\text{Inputs of Srv}}, \underbrace{DS_{\mathrm{C}_1}, ..., DS_{\mathrm{C}_n}}_{\text{Inputs of } \mathrm{C}_1, ..., \mathrm{C}_n}, \underbrace{prm_{\mathrm{FC}}}_{\text{Input of FC}}\Big) :=$$

$$\{\mathsf{View}^{\mathcal{A},\mathcal{PSI}_1}_{\mathrm{Srv}}, ..., \mathsf{View}^{\mathcal{A},\mathcal{PSI}_n}_{\mathrm{Srv}}, \mathsf{View}^{\mathcal{A},\mathcal{SB}}_{\mathrm{Srv}}\}$$

where prm_{Srv} includes the input parameters (e.g., the initial global model) of Srv to federated learning, DS_{Srv} is the dataset held by Srv, DS_{C_i} is a dataset held by C_i, and prm_{FC} includes the inputs parameters of FC to federated learning.

Furthermore, each $\mathsf{View}^{\mathcal{A},\mathcal{PSI}_i}_{\mathrm{Srv}}$ refers to the Srv's real-model view during the execution of an instance of $\mathcal{PSI}$ with C_i, while $\mathsf{View}^{\mathcal{A},\mathcal{SB}}_{\mathrm{Srv}}$ is Srv's real-model view during the execution of SecureBoost. The simulator $\mathsf{Sim}^{\mathcal{F},\mathcal{L}_1}_{\mathrm{Srv}}$, which receives Srv's inputs $(prm_{\mathrm{Srv}}, DS_{\mathrm{Srv}})$ and the leakage $\mathcal{L}_1(inp) := \Big(v_1, ..., v_n, \mathcal{W}_1(prm_1, prm_2)\Big)$ operates as follows.

1. generates an empty view.
2. generates n sets $DS'_{\mathrm{C}_1}, ..., DS'_{\mathrm{C}_n}$, where the size of each DS'_{C_i} is v_i and the element of DS'_{C_i} are picked uniformly at random from the set elements' universe (for all i, $1 \leq i \leq n$).
3. for each C_i, extracts the Srv-side simulation of $\mathcal{PSI}$ from the simulator of $\mathcal{PSI}$ using input sets DS_{Srv} and DS'_{C_i}. Let $\mathsf{Sim}^{\mathcal{PSI}_i}_{\mathrm{Srv}}$ denote this simulation. Note that the latter simulation is guaranteed to exist because this specific PSI, i.e., $\mathcal{PSI}$, has been proven secure in [36]. It appends each $\mathsf{Sim}^{\mathcal{PSI}_i}_{\mathrm{Srv}}$ to the view.
4. extracts the Srv-side simulation of SecureBoost $\mathcal{SB}$ from the simulator of $\mathcal{SB}$. Let $\mathsf{Sim}^{\mathcal{SB}}_{\mathrm{Srv}}$ denote this simulation. It appends $\mathsf{Sim}^{\mathcal{SB}}_{\mathrm{Srv}}$ to the view. Note that we assume the latter simulation exists and can be produced, using the leakage $\mathcal{W}_1(prm_1, prm_2)$ and $\mathcal{SB}$ introduced in [18]. It outputs the view.

Now, we are prepared to demonstrate that the two views are indistinguishable. Since $\mathcal{PSI}$ has been proven secure, $\mathsf{View}^{\mathcal{A},\mathcal{PSI}_i}_{\mathrm{Srv}}$ and $\mathsf{Sim}^{\mathcal{PSI}_i}_{\mathrm{Srv}}$ are computationally indistinguishable. Similarly, under the assumption that SecureBoost is simulatable, $\mathsf{View}^{\mathcal{A},\mathcal{SB}}_{\mathrm{Srv}}$ and $\mathsf{Sim}^{\mathcal{SB}}_{\mathrm{Srv}}$ are distinguishable. Thus, the real and ideal models are indistinguishable.

B.2 Corrupt FC

In the real execution, the view of FC is defined as follows:

$$\mathsf{View}^{\mathcal{A},\textit{Starlit}}_{\mathrm{FC}}\Big(prm_{\mathrm{Srv}}, DS_{\mathrm{Srv}}, DS_{\mathrm{C}_1}, ..., DS_{\mathrm{C}_n}, prm_{\mathrm{FC}}\Big) :=$$

$$\{S_{\mathrm{C}_1}, ..., S_{\mathrm{C}_n}, \mathsf{View}^{\mathcal{A},\mathcal{SB}}_{\mathrm{FC}}\}$$

where S_{C_i} is a set of size s_i. It contains triples each of which has the form (ID_u, b_u, w_u) corresponding to a user U, where ID_u represents a random ID of a sample, b_u is a differentially-private binary flag indicating features' inconsistency, w_u is another differentially-private binary flag (indicating a suspicious user). The simulator $\mathsf{Sim}^{\mathcal{F},\mathcal{L}_2}_{\mathrm{FC}}$, which receives leakage $\mathcal{L}_2(inp) := \Big(s_1, ..., s_n, \mathcal{W}_2(prm_1, prm_2)\Big)$ operates as follows.

1. generates an empty view.
2. constructs n sets $S'_{\mathrm{C}_1}, ..., S'_{\mathrm{C}_n}$, where each S'_{C_i} has the following form.
   - contains s_i triples (ID', b', w').
   - each ID' is a string picked uniformly at random.
   - each b' has been constructed by picking a random binary value and then applying the local differential privacy to it.
   - each w' is generated the same way as b' is generated.
3. appends $S'_{\mathrm{C}_1}, ..., S'_{\mathrm{C}_n}$ to the view.

4. extracts the FC-side simulation of $\mathcal{SB}$ from the simulator of $\mathcal{SB}$. Let $\mathsf{Sim}^{\mathcal{SB}}_{\text{FC}}$ denote this simulation. It appends $\mathsf{Sim}^{\mathcal{SB}}_{\text{FC}}$ to the view. We assume the latter simulation can be produced, using the leakage $\mathcal{W}_2(prm_1, prm_2)$ and $\mathcal{SB}$. It outputs the view.

Next, we argue that the two views are indistinguishable. In the real model, each ID_u is a uniformly random string, as is each ID' in the ideal model. Thus, they are indistinguishable. Also, the elements of each pair (b_u, w_u) in the real model are the output of the differential privacy mechanism (DP), so are the elements of each pair (b', w'). Due to the security of DP, (b_u, w_u) and (b', w') are differentially private and indistinguishable. Moreover, under the assumption that SecureBoost is simulatable, $\mathsf{View}^{\mathcal{A},\mathcal{SB}}_{\text{FC}}$ and $\mathsf{Sim}^{\mathcal{SB}}_{\text{FC}}$ are distinguishable. Hence, the real and ideal models are indistinguishable.

B.3 Corrupt Client

In the real execution, the view of each C_i is defined as follows:

$$\mathsf{View}^{\mathcal{A},\mathit{Starlit}}_{\text{C}_i}\Big((prm_{\text{Srv}}, DS_{\text{Srv}}), DS_{\text{C}_1}, ..., DS_{\text{C}_n}, prm_{\text{FC}}\Big) :=$$

$$\{(DS_{\text{Srv}} \ \cap \ DS_{\text{C}_i}), S_{\text{Srv}}, \mathsf{View}^{\mathcal{A},\mathcal{PSI}}_{\text{C}_i}\}$$

The simulator $\mathsf{Sim}^{\mathcal{F},\mathcal{L}_{i+2}}_{\text{C}_i}$, which receives input DS_{C_i} and leakage $\mathcal{L}_{i+2}(inp) := \Big((DS_{\text{Srv}} \cap DS_{\text{C}_i}), |DS_{\text{Srv}}|, S_{\text{Srv}}\Big)$ operates as follows.

1. generates an empty view and appends $DS_{\text{Srv}} \ \cap \ DS_{\text{C}_i}$ to the view.
2. sets $z = |S_{\text{Srv}}|$. Then, it generates z pairs, each of which has the form (ID', feat'_u). In each pair, ID' is a uniformly random string (picked from the IDs' universe), and feat'_u is one of the user's unique features in S_{Srv}.
3. generates an empty set $\bar{S}_{\text{Srv}}$ and appends all the above z pairs (generated in step 2) to $\bar{S}_{\text{Srv}}$. It appends $\bar{S}_{\text{Srv}}$ to the view.
4. generates an empty set $\bar{DS}_{\text{Srv}}$ and then appends $DS_{\text{Srv}} \ \cap \ DS_{\text{C}_i}$ to $\bar{DS}_{\text{Srv}}$.
5. appends to $\bar{DS}_{\text{Srv}}$ a set of uniformly random elements (from the set elements' universe) such that the size of the resulting set $\bar{DS}_{\text{Srv}}$ is $|DS_{\text{Srv}}|$.
6. for each C_i, extracts the C_i-side simulation of $\mathcal{PSI}$ from the simulator of $\mathcal{PSI}$ using input sets $\bar{DS}_{\text{Srv}}$ and DS_{C_i}. Let $\mathsf{Sim}^{\mathcal{PSI}_i}_{\text{Srv}}$ denote this simulation. It appends each $\mathsf{Sim}^{\mathcal{PSI}_i}_{\text{Srv}}$ to the view. It outputs the view.

Now, we are ready to demonstrate that the two views are indistinguishable. The intersection $DS_{\text{Srv}} \cap DS_{\text{C}_i}$ is identical in both views; therefore, they have identical distributions. Each ID in S_{Srv} is a uniformly random string, so is each ID' in $\bar{S}_{\text{Srv}}$. As a result, they have identical distributions too. Also, each user's unique feature feat_c in S_{Srv} and feat'_c in $\bar{S}_{\text{Srv}}$ have identical distributions, as they are both picked from S_{Srv} that includes some users' of C_i. Due to the security of $\mathcal{PSI}$, $\mathsf{View}^{\mathcal{A},\mathcal{PSI}_i}_{\text{Srv}}$ and $\mathsf{Sim}^{\mathcal{PSI}_i}_{\text{Srv}}$ are computationally indistinguishable. We can conclude that the two views are computationally indistinguishable.

C Further Discussion on PSI Implementation

In this work, we initially focused on and implemented the RSA-based PSI proposed in [3], due to its simplicity that would lead to easier security analysis. However, after conducting a concrete run-time analysis we noticed that this protocol has a very high run time. Therefore, we adjusted and used the Python-based implementation of the PSI introduced in [36] which yields a much lower run-time than the one in [3]. This protocol mainly relies on efficient symmetric key primitives, such as Oblivious Pseudorandom Functions and Cuckoo hashing. The security of this PSI has been proven in a standard simulation-based (semi-honest) model and the related paper has been published at a top-tier conference. The efficiency of the PSI protocol mainly stems from the efficient "Oblivious Pseudorandom Function" (OPRF) that Kolesnikov *et al.* constructed which itself uses the Oblivious Transfer (OT) extension proposed earlier in [35].

To compare the two PSIs' runtime we conducted certain experiments, before developing the *Starlit*'s implementation. The experiments were carried out on a laptop with 8 cores, 2.4 GHZ i9 CPU and 64GB of memory. We did not take advantage of parallelization. Our experiments were carried out with each party having 2^n set elements and compared the run-time of the PSI in [3] with the one in [36]. We concluded that the PSI in [36] is around 10–11 times faster than the one in [3]. We conducted the experiments when each party's set's cardinality is in the range $[2^9, 2^{19}]$. Briefly, our evaluation indicates that the PSI's runtime increases from 0.84 to 367.93 seconds when the number of elements increases from 2^9 to 2^{19}. Table 4 summarizes the run-time comparison between the two PSIs. We use the Flower adapter to communicate between the PSI client (i.e., FSP) and the PSI server (i.e., a bank C). An instance of the PSI, for each account, takes as input the following string: $account_{number}||customer_{name}||street_{name}||countryCity_{zipcode}$

Table 4: The run-time comparison between the RSA-based PSI in [3] and our choice of PSI proposed in [36] (in sec.).

Protocol	Set Cardinality										
	2^9	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}
[3]	4.10	9.32	16.56	32.78	65.45	132.97	252.56	524.32	1059.49	-	-
[36]	0.84	1.18	1.84	3.23	6.06	11.63	23.75	45.80	99.09	183.79	367.93

D Fields of Dataset 1

The provided synthetic Dataset 1 contains the following fields:

- MessageId: Globally unique identifier within this dataset for individual transactions.
- UETR: The Unique End-to-end Transaction Reference—a 36-character string enabling traceability of all individual transactions associated with a single end-to-end transaction
- TransactionReference: Unique identifier for an individual transaction.
- Timestamp: Time at which the individual transaction was initiated.
- Sender: Institution (bank) initiating/sending the individual transaction.
- Receiver: Institution (bank) receiving the individual transaction.
- OrderingAccount: Account identifier for the originating ordering entity (individual or organization) for end-to-end transaction.
- OrderingName: Name for the originating ordering entity.
- OrderingStreet: Street address for the originating ordering entity.
- OrderingCountryCityZip: Remaining address details for the originating ordering entity.
- BeneficiaryAccount: Account identifier for the final beneficiary entity (individual or organization) for end-to-end transaction.
- BeneficiaryName: Name for the final beneficiary entity.
- BeneficiaryStreet: Street address for the final beneficiary entity.
- BeneficiaryCountryCityZip: Remaining address details for the final beneficiary entity.
- SettlementDate: Date the individual transaction was settled.
- SettlementCurrency: Currency used for transaction.
- SettlementAmount: Value of the transaction net of fees/transfer charges/forex.
- InstructedCurrency: Currency of the individual transaction as instructed to be paid by the Sender.
- InstructedAmount Value of the individual transaction as instructed to be paid by the Sender.
- Label: Boolean indicator of whether the transaction is anomalous or not. This is the target variable for the prediction task.

E Fields of Dataset 2

The provided synthetic Dataset 2 contains the following fields:

- Bank: Identifier for the bank.
- Account: Identifier for the account.
- Name: Name of the account.
- Street: Street address associated with the account.
- CountryCityZip: Remaining address details associated with the account.
- Flags: Enumerated data type indicating potential issues or special features that have been associated with an account.

F Challenges

To implement *Starlit*, we encountered a set of challenges. In this section, we briefly discuss these challenges and explain how we overcame them, with the hope that they can assist future researchers who need to develop similar systems.

F.1 The Challenge of Using Flower in Starlit's Implementation

The Flower clashed with our proposed architecture in which there was no central server controlling the process. In *Starlit*, FSP acts as a coordinator responsible for initiating the feature extraction and learning phases. Nevertheless, in Flower, a central server executes a series of training rounds, each comprising a sequence of steps that involve sending instructions to and receiving results from the clients in the system. Flower also demands the server's strategy to have precise knowledge of each client's designated actions in every round of communication.

This process complicates the ability to perform any pre-processing of data before entering the iterative training process, a necessity for us to complete our feature extraction phase.

To address these challenges, we developed a server strategy implementation that functions as a message router for the clients in the system. This design enabled us to preserve our initial concept of Srv acting as a coordinating entity, allowing the server strategy to remain indifferent to the specific actions required in each round of messages. This alteration significantly streamlined the server logic. Moreover, this change substantially enhances the prototype's extensibility and adaptability to additional use cases.

F.2 The Challenge of Using FATE in Starlit's Implementation

The implementation of *Starlit* uses SecureBoost implementation initially integrated within FATE, an open-sourced FL project (as discussed in Section 4.4, page 7). In *Starlit*, SecureBoost is executed jointly among Srv and FC. Since Flower does not incorporate FATE and the Flower API itself does not permit parties to communicate directly with each other, we have devoted significant effort to readjusting the implementation of *Starlit* to seamlessly integrate with the Flower API. We address this challenge by routing the messages that parties exchange through the Flower's server.

Moreover, the Flower's clients operate in a stateless manner. To seamlessly integrate with FATE without the necessity to snapshot the entire system for each Flower round, we developed a new implementation of the FATE API. This implementation utilizes Python multiprocessing queues for sending and receiving messages. By adopting this approach, we were able to instantiate longer-lived standalone FATE processes dedicated to training and prediction tasks. In this configuration, each Flower client acts as a proxy, facilitating the exchange of messages with the respective FATE process through Python multiprocessing queues.

G Starlit's CPU Utilization

Figure 7 presents the CPU utilization in *Starlit*. In the figure, we used 5 CPUs and did not use any CPU-specific library to run Paillier encryption, used as a subroutine in SecureBoost.

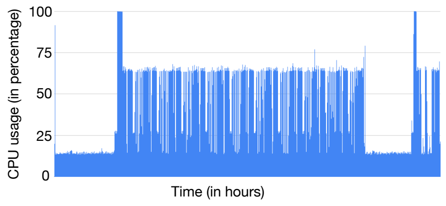


Fig. 7: CPU utilisation of *Starlit*. Peaks in the graph indicate times of high CPU activity, while valleys represent periods of lower activity. A steady line with little variation indicates a consistent level of CPU usage.

G.1 Applications of Starlit's Components

Through *Starlit*'s development, we demonstrated privacy preserving creation of features that are a function of data held by multiple parties (e.g. the string matching features on users' features held by Srv and the clients). This flexible feature creation is an essential building block of any federated learning problem.

We have additionally developed a *general-purpose* framework that can be used (in other contexts) for selecting optimal obfuscation mechanisms to safeguard flags (any categorical variables), aiming to maximize inference privacy while adhering to specified constraints on utility and local differential privacy. This framework offers flexibility in expressing utility constraints and can be adapted to prioritize utility maximization under specified constraints on inference privacy and guarantees of local differential privacy. Our modular design and integration between FATE and Flower enable not just SecureBoost, but also a broad range of FATE functions to be executed via Flower. Furthermore, the Flower adapter architecture we have built contributes to extending the use of Flower to vertically partitioned use cases.