Starlit: A Privacy-Preserving Vertical Federated Learning to Enhance Financial Fraud Detection*

Aydin Abadi**[1] Mohammad Naseri***[1] George Theodorakopoulos†[2]
Steven J. Murdoch‡[1] Sasi Kumar Murakonda§[3] Suzanne Weller¶[3]

[1] University College London
[2] Cardiff University
[3] Privitar

Abstract. Federated Learning (FL) is a data minimization machine learning approach that enables collaborative model training across diverse clients with local data while avoiding direct data exchange. Vertical Federated Learning (VFL), a key FL variant, finds applications in financial crime prevention. However, state-of-the-art VFL solutions for financial crime have notable drawbacks. They (1) lack formal security definitions and proofs, (2) assume that a financial institution treats a suspicious account only in a certain way, (3) involve $O(n^2)$ modular exponentiation where n is the total number of clients, (4) exclude an identity alignment phase from the implementation and evaluation, or (5) struggle to resist clients' dropouts. In this work, we introduce *Starlit*, a pioneering *scalable* privacy-preserving VFL mechanism designed to enhance financial fraud detection and overcome the aforementioned limitations. We have implemented *Starlit* and conducted a thorough performance analysis using synthetic data provided by 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

* Our solution, *Starlit*, has been cited by both the White House and UK Government websites [60,59].

** aydin.abadi@ucl.ac.uk
*** mohammad.naseri.19@ucl.ac.uk
† theodorakopoulosg@cardiff.ac.uk
‡ s.murdoch@ucl.ac.uk
§ sasi.murakonda@privitar.com
¶ sweller@informatica.com

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) [72] and secure Multi-party Computation (MPC) [74], along with their combination, are examples of mechanisms that allow parties to collaboratively analyze shared data while maintaining the privacy of their input data.[4]

FL is a machine learning framework where multiple parties collaboratively build machine learning models without revealing their sensitive input data to their counterparts [72,42]. FL can be categorised into three classes, based on how data is partitioned. Vertical Federated Learning (VFL) is one of the primary classes that has found applications in dealing with crime [5], developing financial risk models [19], or healthcare [44]. VFL refers to the FL setting where datasets distributed among different parties (e.g., banks) share the same samples (e.g., customers' names) while holding different features (e.g., customers' attributes such as name, address, and how they are perceived by a financial intuition).

Concurrently with this work, researchers have proposed VFL-based solutions in [5,46] to deal with financial crime. Nevertheless, as discussed in Section 2.2, they exhibit a subset of the following limitations: (1) they lack a formal security definition and proof, (2) they assume that a financial institution has already frozen suspicious customers' accounts (which limits the solutions' adoption), (3) they involve $O(n^2)$ computationally expensive modular exponentiation where n is the total number of financial institutions, (4) they assume the parties have already completed the identity alignment phase, hence excluding it from the implementation, performance evaluation, and security analysis, and (5) they fail to terminate successfully even if only one of the clients neglects to transmit its message, i.e., cannot resist clients' dropouts.

1.1 Our Contributions

In this work, we introduce *Starlit*, a novel scalable privacy-preserving vertical federated learning mechanism that helps enhance financial fraud detection.

By devising *Starlit*, we address all limitations of the state-of-the-art mechanisms. Specifically, we (1) provide a formal security definition of *Starlit* in the standard simulation-based paradigm and prove its security, (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 $O(n)$, (4) include all phases of *Starlit* in the implementation, performance evaluation, and security analysis, and (5) make *Starlit* resilient against dropouts of clients, e.g., financial institutions.

[4] The primary difference between FL and MPC is that in the former, parties jointly compute a global *model* while in the latter, they jointly compute a certain *function* of the inputs.

Starlit's implementation benefits from synthetic data (consisting of about four million rows) provided by a major organization that handles worldwide financial transactions. *Starlit* is the first solution that simultaneously offers the aforementioned features. We identify several applications of *Starlit* in the context of mitigating terrorism, aiding governance during pandemics, and improving malware identification.

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 Deferential Privacy to preserve the privacy of accounts' flags (that indicate whether an account is deemed suspicious). Moreover, based on our observation that each financial transaction is often accompanied by a random identifier, we allow a third-party feature collector to efficiently aggregate different clients' flags without being able to associate the flags with a specific customer.

A Summary of our Contributions. In this work, we achieve the following:

- Introduce *Starlit*, a novel privacy-preserving vertical federated learning mechanism designed to enhance financial fraud detection.
- Formally define and prove *Starlit*'s security using the simulation-based paradigm.
- Implement *Starlit*.
- Conduct a comprehensive evaluation of its performance.
- Demonstrate several real-world applications of *Starlit*.

1.2 Primary Goals and Setting

This paper focuses on a real-world scenario in which 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 entities $W = \{\mathrm{W}_1, ..., \mathrm{W}_m\}$, such as banks, eBay, and Amazon—aims to detect anomalous transactions.

In this case, often FSP holds a dataset containing transactions between the ordering account held by W_i and the beneficiary account held by W_j. Each entity in W maintains a dataset containing the customers' account information, including customers' details, their transaction history, and even local assessments of their known financial activities. While FSP is capable of training a machine learning model to detect anomalous transactions using its own data, it could enhance the analytics by considering the complementary data held by other entities concerning the customers involved in the transactions.

At a high level, the ultimate goal is to enable FSP to collaborate with other entities to *develop a model* that is much better than the one developed on FSP's data alone, to detect suspicious transactions and ultimately to deal with financial

[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`

fraud. However, a mechanism that offers the above feature must satisfy vital security and system constraints; namely, (i) the privacy of the parties' 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 partitioned data in which each FSP's transaction is associated with an ordering account residing at an entity, W_i, and a beneficiary account residing at another entity, W_j. Our proposed solution will enhance FSP's dataset with two primary types of features using the datasets of W_i and W_j:

- **Discrepancy Feature**: This feature will enhance FSP's data by reflecting whether there is a discrepancy between (i) the information (e.g., name and address) it holds about a certain customer c under investigation and (ii) the information held by the ordering entity W_i and held by the beneficiary entity W_j about the same customer. For each customer, this feature is represented by a pair of binary values $(b_{c,i}, b_{c,j})$, where $b_{c,i}$ and $b_{c,j}$ represents whether the information that FSP holds matches the one held by the ordering and beneficiary entities respectively.
- **Account Flag Feature**: This feature will enhance FSP's data by reflecting whether FSP and a bank considers a certain customer suspicious. This feature is based on a pair of binary private flags for a certain customer, where one flag is held by the ordering entity and the other one is held by the beneficiary entity. In general, banks often allocate flags to each customer's account for internal use. The value of this flag is set based on the customer'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 FSP'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 FSP to find out whether the data it holds about a certain (suspicious) customer matches the one held by a bank, we use PSI. Furthermore, to enhance FSP's data with the flag feature, each bank 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.

2 Related Work

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

- A centralized approach, where a bank either trains its model based on the history of financial transactions it holds, or a centralized party collects data in plaintext from different sources without considering the privacy of different parties. This approach is covered in Section 2.1.

- 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 2.2.
- Other none AI-based schemes, which is covered in Section 2.3.

2.1 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 neighbour, 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 analysed 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 [68,70]. 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 [32,17]. 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 [29] 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.

2.2 Distributed Approaches With Privacy Support

Generally, FL can be categorized into three classes [72], 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.

To the best of our knowledge, there is currently no FTL-based approach designed to address financial fraud. For the remainder of this section, our focus will be on schemes based on HFL and VFL

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 organisations 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.* [73] proposed an FL-based method using "Federated Averaging" [40] to train a model that can detect credit card fraud. Similar to the scheme in [56], this method falls under the HFL category.

VFL-Based Approaches. 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) the type of data utilized, as the scheme in [39] incorporates both social data and bank data, while *Starlit* relies on bank data and transaction data collected by the FSP, and (ii) *Starlit* operates in a multi-party setting, where various banks contribute their data, in contrast to the aforementioned scheme, which involves only two parties.

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 a VFL-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 in [5]. 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. 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. 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.

Simultaneously with our solution, another approach has been developed by Qiu *et al.* [46]. This alternative relies on neural networks and shares the same objective as the previously mentioned solution in [5] archives. 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 in [46]. 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.

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

2.3 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 [31,53]. 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 [20,1]. 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.

3 Informal Threat Model

Starlit involves three types of parties:

- Financial Service Provider (FSP). It facilitates financial transactions and payments between various entities. FSP often observes and maintains the financial messages, particularly transactions exchanged between the bank that sends a transaction and the bank that receives the transaction.
- Banks ($B_1, ..., B_n$). They are regular banks that provide financial services to their customers. It is possible that multiple banks share a common set of customers. The beneficiary bank refers to a bank to whom the payment is to be made. The ordering bank, on the other hand, is the bank that a customer

uses to issue a letter of credit. This bank is often referred to as the issuing bank because it issues the letter of credit on behalf of its client.
- Flag Collector (FC). It is a third-party server that aggregates some of the customer-related features held by different banks. 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 [25]. Hence, they follow the protocol's description without any deviation. However, they try to learn other party's private information, e.g., honest customers' data held by a bank should not be revealed to another bank.

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 a secure channels.

4 Preliminaries

4.1 Notations and assumptions

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 belonging 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 $S_\cap = S_1 \cap S_2, ..., \cap S_n$.

PSIs can be categorized into traditional and delegated types. In traditional PSIs, data owners interactively compute the results using their local data, whereas delegated PSIs utilize cloud computing for computation and/or storage while preserving the privacy of the computation inputs and outputs from the cloud. In this work, due to its efficiency, we employ a PSI from the former category and denote the concrete 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 [21]. Below, we restate it.

Table 1: Notation Table.

Symbol	Description
FSP	Financial Service Provider
FC	Feature Collector
B_i	A 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	Randomised 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*
$\mathcal{L}_{i+2}$	B_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

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 Randomised 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 $p_{00}/p_{01} \leq \epsilon$.

Randomised Response. In addition to the requirement of satisfying ϵ-LDP, Wang *et al.* [65] propose selecting the matrix parameters to maximise the prob-

ability of retaining the true value, i.e., to maximise $p_{00} + p_{11}$. This results in 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 [22]. 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 binarize the output.

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 some protection against inference attacks by FSP. Given a value of ϵ in LDP, 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 always assign an equal probability of converting a 0 to 1 and 1 to 0. Moreover, they need not be the optimal transformation matrices that provide maximum inference privacy, i.e., maximum protection against FSP'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), so human intuition would not be able to find them. More details about the game construction and solution are provided in Sections 7.

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/parties 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 General Procedure

Server:
Initialize global model θ
for each round $t = 1, 2, 3, ..., T$ **do**
    Broadcast θ to all participating devices
    Clients:
    for each client i (where $1 \leq i \leq n$) in parallel **do**
        Receive global model θ
        Compute local update g_i using local data
        Send g_i to the server
    Server:
    Aggregate local updates: $G_t = \sum_{i=1}^{n} g_i$
    Update global model: $\theta_{t+1} = \text{UpdateModel}(\theta_t, G_t)$

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 the 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:** In order 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, often (V)FL is an interactive process within which parties exchange messages. Thus, there is a possibility of a leakage to the participants of this process. 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 the corresponding 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.

4.5 Flower: A Federated Learning Implementation Platform

We implement *Starlit* within a FL framework, called *Flower*, that 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. On the other hand, 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 [41], FedOptim [47], or FedProx [37].

4.6 Security Model

In this paper, we use the simulation-based paradigm of secure multi-party computation [25] 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 security computes $\mathcal{F}$ in the presence of a static 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:*

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

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

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 (as 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.

In the (privacy-preserving) training phase, FC and FSP train the VFL model using the aligned data and an efficient VFL algorithm, called SecureBoost (discussed in Section 4.4).

Figure 1 depicts the interaction between parties in *Starlit*. As the figure indicates, FSP only transmits account numbers and message IDs to Banks and only transmits message IDs to FC. Furthermore, banks only transmit flags that are paired with these account numbers to FC. Neither the Banks nor FC can make useful inferences from these messages.

However, this may still leave the chance of an inference attack during model training/deployment. To address this issue, we use local differential privacy, where any flag values that leave the bank are obfuscated via a randomization

strategy. Note that this protection is an additional layer on top of what's already offered by the SecureBoost protocol, which only shares encrypted (aggregated) gradient information.

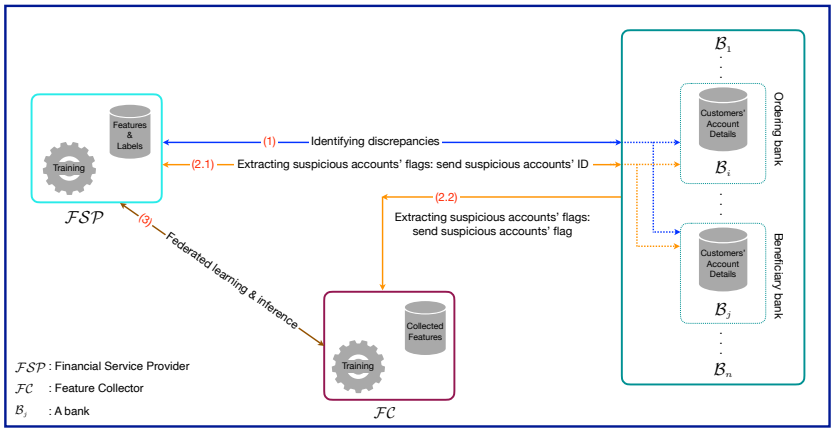


Fig. 1: Outline of the interactions between parties in *Starlit*. Phase 1 (feature extraction) includes steps 1 and 2 in the figure, while Phase 2 (training) involves step 3.

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 VFL schemes such as *Starlit*.

Celestial involves three types of parties, (i) a service provider FSP, (ii) a feature collector FC, and (iii) a set of parties $\{\mathrm{B}_1, ..., \mathrm{B}_n\}$ contributing their private inputs. Informally, *Celestial* allows FSP to generate a (global) model given its initial model and the inputs of $\mathrm{B}_1, ..., \mathrm{B}_n$. To achieve a high level of computational efficiency and scalability, in *Celestial*, we involve a third-party FC that assists FSP with computing the model (by interacting with B_is and retrieving the features they hold). The functionality $\mathcal{F}$ that *Celestial* computes takes an input initial model θ from FSP, a set S_i from every B_i, and no input from FC. It returns to FSP an updated model θ'. It returns nothing to the rest of the parties.[11] Hence, $\mathcal{F}$ can be formally defined as follows.

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

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

Since (1) FC interacts with $B_1, ..., B_n$ and collects some features from them and (2) FSP generates the model in collaboration with $B_1, ..., B_n$ and FC, there is a possibility of leakage to the participating parties. Depending on the protocol that realises $\mathcal{F}$ this leakage could contain different types of information. For instance, it could contain (a) each B_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 FSP, l_2 to FC, and leakage l_i to B_{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 realises $\mathcal{F}$, 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 $P \in \{FSP, B_1, ..., B_n, FC\}$, the following holds:*

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

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

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

where $1 \leq i \leq n$.

7 Flag Protection

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

7.1 The Game

The problem of finding a Privacy Mechanism (PM) that offers optimal flag privacy to a bank given the knowledge of the adversary (e.g., FSP or FC), is an instance of a Bayesian Stackelberg game. In a Stackelberg game the *leader*, in our case the bank, 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 FSP, plays next estimating the flag value, knowing the PM that the bank has committed to. It is a Bayesian game because FSP 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 bank 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 bank runs the PM $f(v'|v)$ to select a replacement value $v' \in \mathcal{V}$.
- **Step 2.** Having observed v', FSP selects an estimated flag value $\hat{v} \sim g(\hat{v}|v'), \hat{v} \in \mathcal{V}$. FSP knows the probability distribution $f(v'|v)$ used by the PM, and the bank'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 bank's privacy for this iteration of the game. This number represents FSP'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 both FSP and the bank. FSP tries to minimize the expected game outcome via its choice of g, while the bank 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 Optimisation Problem

We now explain how to construct 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.

FSP knows the function $f(v'|v)$ implemented by the PM, and thus can form a posterior distribution $\Pr(v|v')$ on the true flag value, conditional on the observation v'. Then, FSP 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 FSP may randomize among them. This randomisation is expressed through $g(\hat{v}|v')$, and in this case (10) would be rewritten as

$$\sum_{v,\hat{v}} \Pr(v|v')g(\hat{v}|v')d_p(\hat{v},v), \tag{11}$$

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 FSP, it holds that:

$$\Pr(v|v') = \frac{\Pr(v,v')}{\Pr(v')} = \frac{f(v'|v)\pi(v)}{\sum_v f(v'|v)\pi(v)}. \tag{12}$$

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

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

To facilitate computations, we define

$$x_{v'} \triangleq \min_{\hat{v}} \sum_{v} \pi(v)f(v'|v)d_p(\hat{v},v). \tag{14}$$

Incorporating $x_{v'}$ into Relation (13), the unconditional expected privacy of the bank can be rewritten as

$$\sum_{v'} x_{v'}, \tag{15}$$

which the bank aims to maximise by choosing $f(v'|v)$. The minimum operator makes the problem non-linear, which is undesirable, but Relation (14) can be transformed into a series of linear constraints:

$$x_{v'} \leq \sum_{v} \pi(v)f(v'|v)d_p(\hat{v},v), \forall \hat{v}. \tag{16}$$

Indeed, maximising the resulting value in Relation (15) under Relation(14) is equivalent to maximising Relation (15) under Relation (16). For every v', there must be some $\hat{v}$ for which Relation (16) holds as strict equality: Otherwise, we could increase one of the $x_{v'}$, so the value of Relation (15) would increase.

From Relations (15) and (16), the linear program for the bank is constructed by choosing $f(v'|v), x_{v'}, \forall v, v'$ to solve the following linear programming problem.

$$\textbf{Maximize} \sum_{v'} x_{v'} \tag{17}$$

subject to

$$x_{v'} - \sum_{v} \pi(v) f(v'|v) d_p(\hat{v}, v) \leq 0, \forall \hat{v}, v' \tag{18}$$

$$f(v'|v) \leq p^{\max}(v', v), \forall v, v' \tag{19}$$

$$\sum_{v'} f(v'|v) = 1, \forall v \tag{20}$$

$$f(v'|v) \geq 0, \forall v, v' \tag{21}$$

$$\frac{f(v'|v_1)}{f(v'|v_2)} \leq \exp(\epsilon), \forall v', v_1, v_2 \tag{22}$$

Constraints (20) and (21) reflect that $f(v'|v)$ is a probability distribution function for each v. Constraint (22) 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 enough 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 (19), 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 bank'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: A Privacy-Preserving Vertical Federated Learning to Enhance Financial Fraud Detection

8.1 Privacy-Preserving Feature Extraction Mechanisms

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

Finding Discrepancies Among Features. The analysis of the synthetic data indicates that for each transaction, FSP holds the following key string columns: (i) $customer_{name}$, (ii) $countryCity_{zipcode}$, and (iii) $street_{name}$ for both the ordering and beneficiary accounts. Each bank, for each account, holds various features including $customer_{name}$, $countryCity_{zipcode}$, $street_{name}$, and a flag.

Exploration of the provided synthetic dataset indicated that discrepancies in these columns between FSP and each bank could provide information about anomalous transactions. Various factors can cause discrepancies between FSP's data and a bank's data. For instance, a customer may have bank accounts with both sender and beneficiary banks; however, it has provided inconsistent details (e.g., about its address or occupation) to these banks. Furthermore, the transactions that FSP collects could have been tampered with by outsiders, e.g., [10,67]. Thus, including a feature that indicates discrepancies between a bank and FSP's data can improve models' accuracy.

To ensure that the information held by a bank aligns with the information held by FSP, while safeguarding the privacy of both entities, we use PSI. The use of PSI enables the identification of discrepancies in suspicious accounts, while protecting the privacy of non-suspicious accounts maintained by the bank.

The outcomes of the PSI, which indicate matches between distinct string columns of an account, are incorporated into the FL model as supplementary features. Specifically, FSP and each bank B_i participate in an instance of PSI, which takes as input a set of strings (defined above) from FSP and the bank. It returns to B_i the intersection containing the matching strings. B_i for each customer, adds a binary feature b to its customers' dataset (if it does not already exist). When a customer's detail is in the intersection, then it sets $b = 1$; otherwise, it sets $b = 0$. Figure 3 formally presents the above steps.

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 accounts), presented below.

Collecting Flags of Suspicious Accounts. The analysis of the provided synthetic data indicates that for each account that a bank holds, there is a flag indicating whether the bank considers the account suspicious. This type of flag provides additional information that aids in the detection of anomalies.

However, these flags are considered private information and cannot be shared directly with the FSP. To align the flags with the FSP's dataset without revealing them, we rely on the following observation and privacy-preserving idea.

- **Parties:** FSP and B_i.
- **Input:**
  - ◇ FSP's input is a set S_{FSP} of strings (taken from the dataset DS_{FSP} of the transactions), where each string has the form $account_{number}||customer_{name}||street_{name}||countryCity_{zipcode}$.
  - ◇ B_i's input is a set S_{B_i} of strings (from its dataset DS_{B_i} of all customers), where each string has the form $account_{number}||customer_{name}||street_{name}||countryCity_{zipcode}$.
- **Output:** Updated dataset DS_{B_i}.

---

1. FSP and B_i invoke an stance of PSI protocol: $\mathcal{PSI}(S_{\mathrm{FSP}}, S_{\mathrm{B}_i}) \rightarrow S_{\cap}$.
2. Given $S_{\cap}$, B_i parses each element of $S_{\cap}$ as $(account_{number}, customer_{name}, street_{name}, countryCity_{zipcode})$.
3. If binary feature b is not in DS_{B_i}, then B_i adds b to each account's feature.
4. B_i sets b as follows. For every $account_{number} \in DS_{\mathrm{B}_i}$:
   - Sets $b = 1$, when $account_{number} \in S_{\cap}$.
   - Sets $b = 0$, otherwise.
5. Return DS_{B_i}.

Fig. 3: PSI-based Feature Extraction to Identify Discrepancies.

The key observation is that each transaction, encompassing both ordering and beneficiary accounts, observed by FSP, is associated with a unique ID. This ID, being a random string, divulges no specific information about a customer's account. Therefore, if each bank associates each ID with a (couple of) binary flag(s) and sends these pairs to FC, FC cannot glean significant information about the customers' accounts linked to those IDs. Based on this observation, we rely on the following idea to extract the flags.

FSP sends transaction IDs and associated suspicious account numbers to the relevant banks. Note that FSP already knows which banks are involved in a transaction; therefore FSP only sends to each bank limited information about customers that have accounts with that bank. The banks then use their account information to pair each ID with the correct flag[12] and send this data to FC. When sending a flag for a suspicious account to FC, each bank also sends to FC the flag b that it generated in Figure 3 (to detect discrepancies).

Consequently, FC uses a set of triples (that includes an ID and two flags for each customer deemed suspicious) from the bank to create a dataset of flags, F. This dataset can then be used as the input data for the ML model.

The above private information retrieval mechanism is *highly computationally efficient*. However, this approach reveals certain information to the involved parties. Specifically:

[12] Note that given a transaction ID a bank can find a customer's account number and other related information about that account.

- Each bank gains knowledge of which of their accounts are active and the number of transactions associated with each account.[13]
- FC acquires information about which transaction IDs originate from each bank, enabling the calculation of the number of transactions between each pair of banks.

However, the privacy of sensitive information is still preserved. In particular:

- Each bank remains unaware of details about other participating banks or accounts held at other banks.
- FC cannot identify the accounts involved in a transaction; it only possesses information about the ID of the transaction and the two associated flags. Consequently, FC cannot glean any information about a specific account.

As evident during the feature extraction, each client (i.e., B_i) independently computes its message and sends it to FC without the need to coordinate with other clients. Consequently, 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 is unable to withstand clients' dropouts.

Extension. There is an alternative method for collecting flags, which involves employing an efficient threshold privacy-preserving voting mechanism introduced in [1]. This voting scheme enables the result recipient (e.g., FC or FSP) to ascertain whether, at the very least, a predefined threshold of the involved parties (e.g., banks) deems a particular customer suspicious. Importantly, this process does not disclose any additional information, such as individual votes or the count of 1s or 0s, beyond the final 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. Nevertheless, a more in-depth analysis is necessary to ensure that the system relying on the aforementioned voting scheme can withstand potential client dropouts.

For the sake of simplicity, we have presented a solution focused on a single flag per account. Nonetheless, it is important to note that this solution can be readily generalized to situations where multiple flags are linked to a single account. In this scenario involving multiple flags, when a bank receives an ID and the customer's name, it retrieves a vector of flags associated with that particular account. Following applying either DP or the aforementioned voting-based mechanism to the flags, the bank then transmits the resultant outcome to the FC.

8.2 Model Training and Inference

Following the feature extraction phase, FSP and FC jointly possess all the necessary data for training the anomaly detection model.

[13] Each bank already possesses this information.

FSP retains the features extracted from its data, while FC possesses features that involve comparing names and addresses of ordering and beneficiary accounts with information from banks, along with the flag values associated with these accounts (protected by DP).

This represents the VFL setup, where only one party (i.e., FSP) 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 [15,18,23,28,38,49,55,62,69,71,75]. We use the SecureBoost algorithm (discussed in Section 4.4 on page 11), which involves the exchange of encrypted (aggregate) gradients between FSP and FC during the training phase.

FSP 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], FSP coordinates with the FC to determine the split condition to be used.

It could be possible for FSP to deduce the flag values from the final prediction, indicating a noteworthy and potentially unavoidable trade-off between model interpretability and privacy. The FSP faces a dilemma: it can either elucidate the direct influence of the flag values on its prediction (a necessity for interpretability but lacking in flag value protection) or refrain from learning the precise effects of the flag values on the final prediction (favorable for privacy but detrimental to interpretability).

9 Security of Starlit

In this section, we first formally define the leakage that each party attains during the execution of *Starlit*. Subsequently, we formally state the security guarantee of *Starlit*.

Definition 3 (FSP–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_{B_i} be a dataset of customers held by each B_i, and v_i be each dataset's size, i.e., $v_i = |DS_{B_i}|$, 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 FSP and prm_2 is given by FC. $\mathcal{W}$ returns l_1 to FSP and l_2 to FC. Then, leakage to FSP 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 $s_i = |S_{B_i}|$, where S_{B_i} is a set of triples each of which has the form (ID, b, w), where ID represents a unique transaction random number, b is a binary flag indicating features' inconsistency (computed within the protocol in Figure 3), w is a binary flag indicating a suspicious account (computed in the procedure presented 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 FSP and prm_2 is given by FC. $\mathcal{W}$ returns l_1 to FSP 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 (Bank–Side Leakage). *Let $\mathcal{L}$ be the leakage function defined in Relation 6 and inp be the input of all parties. Moreover, let DS_{FSP} be the dataset of customers' transactions observed by FSP and DS_{B_i} be a dataset of all customers held by B_i. Also, let S_{FSP} be a set of pairs each of which has the form $(ID, customer_{name})$, where ID represents a unique transaction random number and $customer_{name}$ refers to a customer's name which has an account with B_i. Then, leakage to B_i is defined as:*

$$\mathcal{L}_{i+2}(inp) := \Big((DS_{FSP} \cap DS_{B_i}), |DS_{FSP}|, S_{FSP}\Big)$$

Theorem 1. *Let $\mathcal{F}$ be the functionality defined in Relation 5 (on page 15). 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-defferentially private and provides optimal flag privacy (w.r.t. Game presented in Figure 2), the SecureBoost and $\mathcal{PSI}$ are secure, then Starlit securely realises $\mathcal{F}$, w.r.t. Definition 2.*

We prove the above theorem in Appendix A.

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.

10.2 Dataset

Our experiment involves the utilization of two synthetic datasets:

- Dataset 1: A synthetic dataset that simulates transaction data obtained from the FSP's global payment network.

- Dataset 2: Synthetic customer/account metadata, inclusive of flags, derived from the partner banks of the 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 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, payment corridor, amount, and timestamps. 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) Timestamp: the time at which the individual transaction was initiated, (c) Sender: a bank sending the transaction, (d) Receiver: a bank receiving the transaction, and (e) OrderingAccount: an account identifier for the originating ordering entity. Appendix B provides detailed explanations of the fields contained within Database 1.

Outline of Dataset 2. The dataset comprises databases from various banks, encompassing information about their customers' accounts, including the flags associated with each account. Initially, the data was unpartitioned, with all the banks' information consolidated into a single table. This databset 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 C elaborates on the fields that Database 2 contains.

10.3 Implementation of PSI

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 relies on 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 $[2^9, 2^{19}]$. Table 2 summarises the run-time comparison between the two PSIs. We use the Flower adaptor to communicate between the PSI client (i.e., FSP) and the PSI server (i.e., a bank B). An instance of the PSI, for each account, takes as input the following string:

$$account_{number}||customer_{name}||street_{name}||countryCity_{zipcode}$$

The output of the PSI is received by the participating B.

Table 2: 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

10.4 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 successfully 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 FSP 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.

10.5 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 12). In *Starlit*, SecureBoost is executed jointly among FSP 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 utilises 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.

11 Empirical Results

We assessed *Starlit* using a synthetic dataset provided by a major messaging network provider to financial institutions. The dataset consists of about four million rows. Our evaluation of *Starlit* includes various perspectives:

- Privacy-utility trade-off, discussed in Section 11.1.
- Efficiency and scalability, explored in Section 11.2.
- The choice of concrete parameters, covered in Section 11.3.

Features Used. In this study, our focus does not lie on feature exploration or hyperparameter tuning to enhance model accuracy. Instead, we employ a straightforward approach, utilising 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 *centralised* model constructed at FSP. In this centralized

model, all data from banks is fully revealed without any privacy protection. The same set of features listed above is extracted, and 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 a 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 analysing the AUPRC that can be achieved at a given level of privacy, we modify the flag values that banks send using an appropriate privacy mechanism (i.e., 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 (averaged over 5 runs to account for the randomness of the privacy mechanism and the training process).

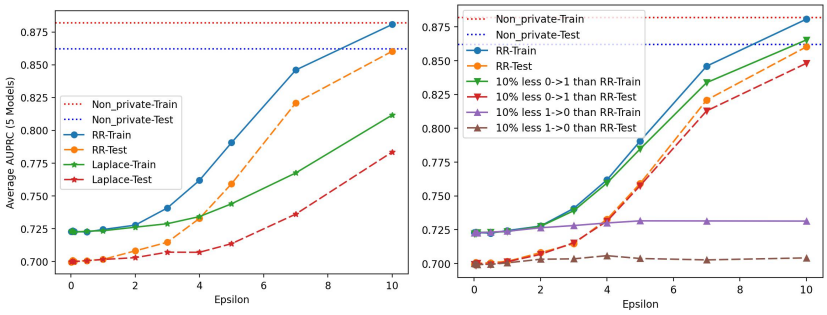
We chose XGBoost for the baseline and also in our experiments to understand the privacy-utility trade-off as it can be theoretically proved that it will achieve the same accuracy as our federated SecureBoost solution, given the same set of parameters and input data. SecureBoost does the same computation as XGBoost while constructing the trees albiet 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.

It is worth noting that in an alternative configuration of our solution, since the flag values are protected with local differential privacy, the banks could even agree to send all the features directly to FSP instead of FC, which implies FSP could directly train the model using standard XGBoost without the need for implementing SecureBoost.

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 Randomised Response (RR) and Laplace mechanism with post-processing for achieving Local Differential Privacy (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,



(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.

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 unequivocally 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. Our solution can be split into two main phases: feature extraction and training (or prediction). The SecureBoost training was configured with 10 trees, each with a depth of 3, a dataset sampling rate of 40%, and "Gradient-based One Side Sampling" (GOSS) sampling of 0.1. Efficiency results for this baseline are provided in Table 3. This baseline was employed to investigate the impact of various configurations on efficiency.

Table 3: Efficiency metrics for baseline performance investigation. In the table, H represents time in hours and GB refers to gigabyte.

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 analysed the *efficiency* of *Starlit* with different SecureBoost configurations. The evaluation's results are illustrated in Table 4. SecureBoost offers various options that can be employed to enhance efficiency in different settings. For instance, both direct sampling[14] and GOSS sampling offer 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. Furthermore, the integration of *Starlit* with FATE and Flower enables the splitting of large messages into chunks, facilitating more efficient processing.

Although *Starlit* uses a large number of Flower rounds, many of the last rounds are empty. This is required because FSP and FC decide together when training and prediction are finished but the Flower requires to pre-set a number of rounds. Similarly, messages need to go through the Flower's server; while a peer-to-peer communication protocol could yield a more efficient implementation. This has an impact on the network metrics.

Furthermore, *Starlit* is *highly scalable*. This is due to the *Starlit* design choice that keeps the model training phase independent of the number of banks. Instead, the training phase involves only FSP and FC. On the other hand, the computation complexity of the feature extraction phase scales linearly with the number of banks and the number of accounts that seem suspicious from FSP's perspective.

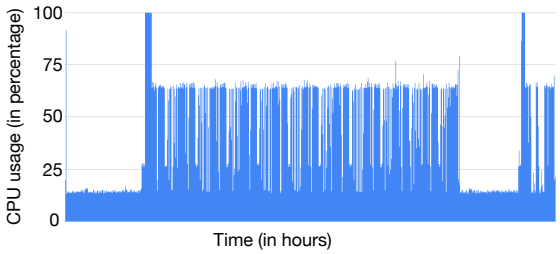


Fig. 5: CPU utilisation of *Starlit*. Peaks in the graph indicate times of high CPU activity, while valleys represent periods of lower activity. A steady line with little variation indicates a consistent level of CPU usage.

Starlit utilizes a high level of CPU. The main reason is that the underlying SecureBoost training is highly CPU intensive, this is due to the encryption required by the algorithm to encrypt the gradient. The CPU utilisation of *Starlit* is provided in Figure 5. 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.

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

Table 4: The efficiency of the overall solution 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

The time that takes to train or predict can be reduced by increasing parallelism or using hardware-specific libraries like the Intel Paillier Cryptosystem Library.

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 centralised XGBoost model with those used in the federated solution. Specifically for the number of trees, tree depth, and L2 regularisation parameter. The parameters selected are as presented in Table 5.

Table 5: The choice of parameters for centralised and federated settings.

Setting	Number of trees	Tree depth	L2 regularisation	ϵ
Centralised	10	5	0.1	–
Federated	10	5	0.1	10

11.4 Contrasting Starlit with the Baseline Framework.

In this section, we provide a brief comparison of *Starlit*'s performance with the baseline scenario, drawing insights from the information presented in Tables 3 and 4.

Starlit and the baseline archive 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 6. 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.

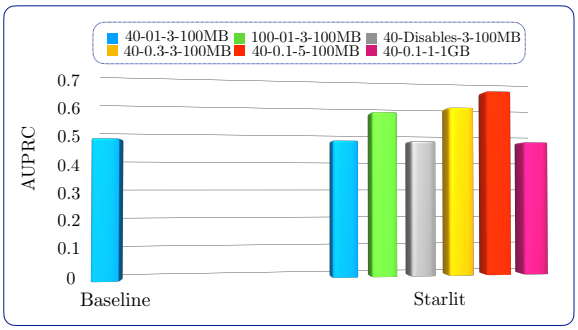
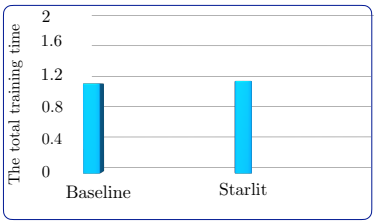


Fig. 6: Comparing the AUPRC of the baseline and different settings of *Starlit*. A bar's label is a concatenation of (1) direct sampling rate, (2) GOSS, (3) tree's depth, and (4) maximum message size.

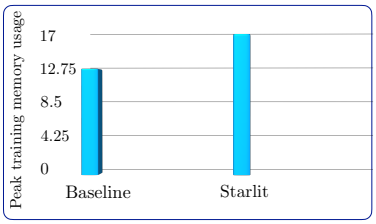
As illustrated in Table 4, *Starlit* achieves its highest AUPRC level (i.e., 0.652) when the tree's depth is set to 5. Remarkably, in this instance, *Starlit*'s AUPRC surpasses even the baseline setting (i.e., 0.652 versus 0.4715). Having identified the parameter that yields the highest AUPRC in *Starlit*, 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 7a 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 7b 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 7c illustrates, *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 7d demonstrates, under this metric, *Starlit* underperform the baseline by a factor of 1.3, i.e., 1316 versus 993.

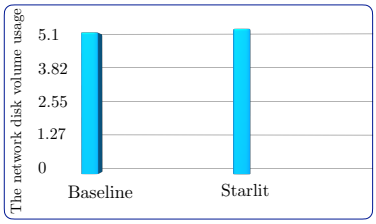
Hence, when the tree's depths in *Starlit* and baseline are set to 5 and 3 respectively, then *Starlit* can attain a superior AUPRC level compared to the baseline. However, in this setting, *Starlit* would impose approximately 1.3 times higher cost than the baseline does.



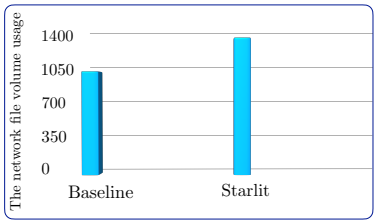
(a) Comparison between the total training time between the baseline and *Starlit*.



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



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



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

Fig. 7: In this figure, *Starlit*'s parameters are set as follows: sampling rate = 40, GOSS = 0.1, tree's depth = 5, and max message size = 100MB.

12 Further Applications of Starlit

Starlit is highly adaptable to many use cases involving a wide range of collaborative analysis tasks, outlined below.

12.1 Mitigating Terrorism

The effective response to the challenge of counterterrorism 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 [48,76] 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. This, in turn, strengthens the collective security efforts of the international community.

12.2 Digital Health

While digital healthcare offers numerous advantages, addressing one of its primary challenges, namely data privacy, is crucial [24]. *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 [30]. *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 Collaborative Detection of 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 [27], 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, the technology plays a crucial role in preventing fraud or misuse, ensuring that resources are allocated with fairness and appropriateness as top priorities.

12.4 Applications of Starlit's Components

Through the development of our solution, we have demonstrated privacy preserving creation of features that are a function of data held by multiple parties (e.g. the string matching features on account names and addresses held by FSP and the banks). 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 maximise 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 prioritise utility maximisation 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.

13 Conclusion and Future Work

In this work, we introduced *Starlit*, a scalable privacy-preserving vertical federated learning mechanism that helps enhance financial fraud detection. We formally defined and proven 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 have 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. During the implementation of *Starlit* we had to overcome several challenges to build the implementation upon existing open-source FL frameworks, i.e., FATE and Flower. In this work, we reported these challenges and the techniques we used to overcome them, with the hope that they can assist future researchers who need to develop similar systems.

In any secure FL, the output inevitably reveals certain information about the private inputs of the participating parties. This reality may dissuade some parties with sensitive and valuable inputs from engaging in the federated learning process, particularly when they lack interest in the resulting outcome. Therefore, future research could enhance *Starlit* by introducing a mechanism to reward parties that actively contribute to the analysis outcome. This approach has the potential to narrow the divide between the data market [26,34,61] and FL. Another intriguing avenue for future research involves fortifying the security of *Starlit* to withstand fully malicious parties.

References

1. Abadi, A., Murdoch, S.J.: Payment with dispute resolution: A protocol for reimbursing frauds victims. In: Proceedings of the 2023 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 Computing Surveys (CSUR) 55(3), 1–38 (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 36(6), 87–98 (2021)
19. Cheng, Y., Liu, Y., Chen, T., Yang, Q.: Federated learning for privacy-preserving AI. Commun. ACM (2020)

20. Confirmation of Payee Team: Confirmation of payee- response to consultation cp20/1 and decision on varying specific direction 10 (2020), https://t.ly/xiJQM
21. Duchi, J.C., Jordan, M.I., Wainwright, M.J.: Local privacy, data processing inequalities, and statistical minimax rates. arXiv:1302.3203 (2013)
22. Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating noise to sensitivity in private data analysis. In: Theory of Cryptography. pp. 265–284 (2006)
23. 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: Proceedings of the 30th ACM International Conference on Information & Knowledge Management. pp. 443–452 (2021)
24. 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)
25. Goldreich, O.: The Foundations of Cryptography - Volume 2, Basic Applications. Cambridge University Press (2004)
26. 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)
27. Government, U.: Benefit fraud (2023), https://www.gov.uk/benefit-fraud
28. 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)
29. Hassan, M.U., Rehmani, M.H., Chen, J.: Anomaly detection in blockchain networks: A comprehensive survey. IEEE Communications Surveys & Tutorials (2022)
30. 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
31. Jacomme, C., Kremer, S.: An extensive formal analysis of multi-factor authentication protocols. ACM Transactions on Privacy and Security (TOPS) (2021)
32. Jung, E., Le Tilly, M., Gehani, A., Ge, Y.: Data mining-based Ethereum fraud detection. In: IEEE International Conference on Blockchain (2019)
33. Kairouz, P., Oh, S., Viswanath, P.: Extremal mechanisms for local differential privacy. Advances in neural information processing systems 27 (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: Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II. Lecture Notes in Computer Science, vol. 8043, pp. 54–70. Springer (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: Proceedings of Machine Learning and Systems 2020, MLSys 2020, Austin, TX, USA, March 2-4, 2020. mlsys.org (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: Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA. Proceedings of Machine Learning Research, 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: Artificial intelligence and statistics. PMLR (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, pp. 31–56. Emerald Publishing Limited (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: Proceedings of the 2012 ACM conference on Computer and communications security. pp. 617–627 (2012)
53. Sinigaglia, F., Carbone, R., Costa, G., Zannone, N.: A survey on multi-factor authentication for online banking in the wild. Computers & Security 95, 101745 (2020)
54. Srokosz, M., Bobyk, A., Ksiezopolski, B., Wydra, M.: Machine-learning-based scoring system for antifraud cisirts in banking environment. Electronics 12(1), 251 (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 SIGKDD International Conference on Knowledge Discovery and Data Mining, San Jose, California, USA, August 12-15, 2007. 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, X., Du, S.: An optimized association rules mining framework and its application in chinese social insurance fund data auditing. Available at SSRN 4292653
69. 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)
70. Xiuguo, W., Shengyong, D.: An analysis on financial statement fraud detection for chinese listed companies using deep learning. IEEE Access 10, 22516–22532 (2022)
71. Xu, R., Baracaldo, N., Zhou, Y., Anwar, A., Joshi, J., Ludwig, H.: Fedv: Privacy-preserving federated learning over vertically partitioned data. In: Proceedings of the 14th ACM Workshop on Artificial Intelligence and Security. pp. 181–192 (2021)

72. Yang, Q., Liu, Y., Chen, T., Tong, Y.: Federated machine learning: Concept and applications. ACM Trans. Intell. Syst. Technol. (2019)
73. 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)
74. Yao, A.C.: Protocols for secure computations (extended abstract). In: 23rd Annual Symposium on Foundations of Computer Science (1982)
75. Zhang, C., Li, S., Xia, J., Wang, W., Yan, F., Liu, Y.: {BatchCrypt}: Efficient homomorphic encryption for {Cross-Silo} federated learning. In: 2020 USENIX annual technical conference (USENIX ATC 20). pp. 493–506 (2020)
76. Zurcher, A.: How trump, biden and clinton secret files cases compare (2023), `https://www.bbc.co.uk/news/world-us-canada-64230040`

A Proof of Security

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

A.1 Corrupt FSP

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

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

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

where prm_{FSP} includes the input parameters (e.g., the initial global model) of FSP to federated learning, DS_{FSP} is the dataset of customers' transactions observed and held by FSP, DS_{B_i} is a dataset of all customers held by B_i, and prm_{FC} includes the inputs parameters of FC to federated learning.

Furthermore, each $\mathsf{View}_{\text{FSP}}^{\mathcal{A},\mathcal{PSI}_i}$ refers to the FSP's real-model view during the execution of an instance of $\mathcal{PSI}$ with B_i, while $\mathsf{View}_{\text{FSP}}^{\mathcal{A},\mathcal{SB}}$ is FSP's real-model view during the execution of SecureBoost. The simulator $\mathsf{Sim}_{\text{FSP}}^{\mathcal{F},\mathcal{L}_1}$, which receives FSP's inputs (prm_{FSP}, DS_{FSP}) 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 $DS'_{\text{B}_1}, ..., DS'_{\text{B}_n}$, where the size of each DS'_{B_i} is v_i and the element of DS'_{B_i} are picked uniformly at random from the set elements' universe (for all i, $1 \leq i \leq n$).
3. for each B_i, extracts the FSP-side simulation of $\mathcal{PSI}$ from the simulator of $\mathcal{PSI}$ using input sets DS_{FSP} and DS'_{B_i}. Let $\mathsf{Sim}_{\text{FSP}}^{\mathcal{PSI}_i}$ 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 $\mathsf{Sim}_{\text{FSP}}^{\mathcal{PSI}_i}$ to the view.

4. extracts the FSP-side simulation of $\mathcal{SB}$ from the simulator of $\mathcal{SB}$. Let $\mathsf{Sim}_{\text{FSP}}^{\mathcal{SB}}$ denote this simulation. It appends $\mathsf{Sim}_{\text{FSP}}^{\mathcal{SB}}$ 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 SecureBoost ($\mathcal{SB}$) 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, $\mathsf{View}_{\text{FSP}}^{\mathcal{A},\mathcal{PSI}_i}$ and $\mathsf{Sim}_{\text{FSP}}^{\mathcal{PSI}_i}$ are computationally indistinguishable. Similarly, under the assumption that SecureBoost is simulatable, $\mathsf{View}_{\text{FSP}}^{\mathcal{A},\mathcal{SB}}$ and $\mathsf{Sim}_{\text{FSP}}^{\mathcal{SB}}$ are distinguishable. Thus, the real and ideal models are indistinguishable.

A.2 Corrupt FC

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

$$\mathsf{View}_{\text{FC}}^{\mathcal{A},Starlit}\Big(prm_{\text{FSP}}, DS_{\text{FSP}}, DS_{\text{B}_1}, ..., DS_{\text{B}_n}, prm_{\text{FC}}\Big) := \{S_{\text{B}_1}, ..., S_{\text{B}_n}, \mathsf{View}_{\text{FC}}^{\mathcal{A},\mathcal{SB}}\}$$

where S_{B_i} is a set of size s_i. It contains triples each of which has the form (ID_C, b_C, w_C) corresponding to a customer C, where ID_C represents a unique transaction random number, b_C is a differentially-private binary flag indicating features' inconsistency, w_C is a differentially-private binary flag indicating a suspicious account. The simulator $\mathsf{Sim}_{\text{FC}}^{\mathcal{F},\mathcal{L}_2}$, 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 $S'_{\text{B}_1}, ..., S'_{\text{B}_n}$, where each S'_{B_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 $S'_{\text{B}_1}, ..., S'_{\text{B}_n}$ to the view.
4. extracts the FC-side simulation of $\mathcal{SB}$ from the simulator of $\mathcal{SB}$. Let $\mathsf{Sim}_{\text{FC}}^{\mathcal{SB}}$ denote this simulation. It appends $\mathsf{Sim}_{\text{FC}}^{\mathcal{SB}}$ to the view. We assume the latter simulation can be produced, using the leakage $\mathcal{W}_2(prm_1, prm_2)$ and SecureBoost ($\mathcal{SB}$). It outputs the view.

Next, we argue that the two views are indistinguishable. In the real model, each ID_C is a uniformly random string, as is each ID' in the ideal model. Thus, they are indistinguishable. Also, the elements of each pair (b_C, w_C) 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_C, w_C) and (b', w') are differentially private and indistinguishable. Moreover, under the assumption that SecureBoost is simulatable, $\mathsf{View}_{\text{FC}}^{\mathcal{A},\mathcal{SB}}$ and $\mathsf{Sim}_{\text{FC}}^{\mathcal{SB}}$ are distinguishable. Hence, the real and ideal models are indistinguishable.

A.3 Corrupt Bank

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

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

$$\{(DS_{\mathrm{FSP}} \cap DS_{\mathrm{B}_i}), S_{\mathrm{FSP}}, \mathsf{View}_{\mathrm{B}_i}^{\mathcal{A},\mathcal{PSI}}\}$$

The simulator $\mathsf{Sim}_{\mathrm{B}_i}^{\mathcal{F},\mathcal{L}_{i+2}}$, which receives input DS_{B_i} and leakage $\mathcal{L}_{i+2}(inp) := \Big((DS_{\mathrm{FSP}} \cap DS_{\mathrm{B}_i}), |DS_{\mathrm{FSP}}|, S_{\mathrm{FSP}}\Big)$ takes the following steps.

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

Now, we are ready to demonstrate that the two views are indistinguishable. The intersection $DS_{\mathrm{FSP}} \cap DS_{\mathrm{B}_i}$ is identical in both views; therefore, they have identical distributions. Each ID in S_{FSP} is a uniformly random string, so is each ID' in $\bar{S}_{\mathrm{FSP}}$. As a result, they have identical distributions too. Also, each customer's name $customer_{name}$ in S_{FSP} and $customer'_{name}$ in $\bar{S}_{\mathrm{FSP}}$ have identical distributions, as they are both picked from S_{FSP} that includes some customers' of B_i who have made transactions. Due to the security of $\mathcal{PSI}$, $\mathsf{View}_{\mathrm{FSP}}^{\mathcal{A},\mathcal{PSI}_i}$ and $\mathsf{Sim}_{\mathrm{FSP}}^{\mathcal{PSI}_i}$ are computationally indistinguishable. We can conclude that the two views are computationally indistinguishable.

□

B 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.

C 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.