Artificial Identification: A Novel Privacy Framework for Federated Learning Based on Blockchain

竹本志恩

May 9, 2025

INIAD

目次

書誌情報

- **題名**
  - Artificial Identification: A Novel Privacy Framework for Federated Learning Based on Blockchain
- **発表日**
  - 01 February 2023
- **著者**
  - Liwei Ouyang, Fei-Yue Wang, Yonglin Tian, Xiaofeng Jia, Hongwei Qi, and Ge Wang
- **論文誌名**
  - IEEE TRANSACTIONS ON COMPUTATIONAL SOCIAL SYSTEMS, VOL. 10, NO. 6, DECEMBER 2023

どんな研究?

- ブロックチェーンを適用する FL について
- プライバシーとセキュリティを従来より確保し
- 運用に必要なコストを削減する

ブロックチェーン

- 公正で安定した記録システム
- 改ざんや障害発生に強い
- 記録の複製を複数の参加者が保持する
- ・参加者が脱落してもシステムは動く
- ・取引記録が残る

スマートコントラクト

- 公正で安定した記録システム
- ブロックチェーン上で
- 事前に定めた内容で契約し, 取引を実行
- 条件を満たすと自動で実行
- 不正や改ざんを防ぎ, 効率的な取引を実現

目次

背景

- FL の課題
  - データ送受信に関する攻撃の危険性
  - 信頼性の低いノードによる問題の発生
- ブロックチェーンの役割
  - 安全なブリッジ
    - グローバルモデルのダウンロード
    - ローカルモデルのアップロード
  - インセンティブメカニズムによる参加意欲の向上
- 従来手法はプライバシの保護が不完全
- 二種のスマートコントラクトで改善

目次

手法の概要

- フレームワークを作成
- 2 種のブロックチェーンシステムを利用
  - Ethereum
  - inter-plenary file systems(IPFS)
- 2 種のモジュールから構成
  - a. private P2P identification
  - b. private FL

a. プライベート P2P 識別

- 識別スマートコントラクト (ISC) を使用
  - FL 参加者を直接共有しない
  - ISC を通じてやり取り
  - 各サーバは参加者のリストをローカルに保持
- 正しい参加者を識別しやり取り
- 匿名性とプライバシに役立つ

問題設定

- Blockchain Account
  - acc で示される
  - 公開鍵と秘密鍵のペア $\{pk_{acc}, sk_{acc}\}$ と紐付け
- Federated Members
  - F で示される
  - ISC で他メンバを識別
- Federal Account
  - acc_{FE} で示される
  - $\{pk_{FE}, sk_{FE}\}$ でやり取り
  - 全メンバで共有
  - ここに F がメッセージを送るとブロードキャストする

問題設定

- Trust list
  - ある参加者 F_i, F_j がいる
  - F_j のリスト中の F_i について
    - F_i が F_j の P2P 認証で許可されたということ
    - 相互に信頼
  - 最終的に $\mathrm{TrustList}_{FE}$ ができる？
    - これは各参加者がローカルに保持するということ？
    - 最終的なリストはブロックチェーンに参加する全ての F を網羅する？

問題設定

- Active list
  - 仮に $F_i \in \mathrm{ActiveList}_{F_j}$ のとき
    - F_i が F_j によって合意された $\mathrm{TrustList}_{FE}$ を所持
    - F_j が F_i を学習の参加者とみなしている
  - 各 $\mathrm{TrustList}$ に対応する $\mathrm{ActiveList}$ が存在
    - $\mathrm{TrustList}_{FE}$ に対応するのが $\mathrm{ActiveList}_{FE}$
  - 協調的な FL の参加者一覧

b. プライベート FL

- 協調学習スマートコントラクト (*CTSC*) で学習
- FL において, 以下の 4 段階を実行
  - 登録
  - 検証データ交換
  - 学習
  - 終了処理

b-1. 登録

- 事前に設定された登録時間中
  - デポジット D_r と $\mathrm{len}(\mathrm{ActiveList}_{FE})$ を *CTSC* に報告
- 一定時間経過後，*CTSC* が合意された $\mathrm{len}(\mathrm{ActiveList}_{FE})$ を算出
- 各 F_i は怪しい参加者を除外する準備
  - 「正しい $\mathrm{len}(\mathrm{ActiveList}_{FE})$ を報告したが $\mathrm{ActiveList}_{FE}$ に含まれていないアカウント」を検出
  - ローカルな $\mathrm{RejectList}_{F_i}$ に追加

b-2. 検証データ交換

- 検証用の VSet を作成
  - $F_j \in \mathrm{ActiveList}_{FE}$ からサンプリングしたデータ VSet_{F_j} を受領
  - ローカルで統合し，最終的に VSet を構成

b-3. 学習プロセス

a. **モデル検証**
   - 各 F_i は VSet 上で
     - ローカルモデル Model_{F_i} と検証性能 E_{F_i} を生成
     - $F_i \to acc_{FE}$:
       $\mathrm{Enc}\{\,\mathrm{Enc}\{\,\mathrm{Path}(\mathrm{Model}_{F_i}),\, E_{F_i},\, acc_{F_i}\}_{k_{\mathrm{Active}}}\}_{pk_{FE}}$
       を送信しつつ， 他モデルを検証
     - 偽モデル提出または 2 ラウンド未提出 を検出した F_j を $\mathrm{RejectList}_{F_i}$ に追加

b. **罰則・報酬判定**
   - $CTSC$ が全 $\mathrm{RejectList}_{F_i}$ を統合，拒否回数 RJ_{F_i} を集計
   - RJ_{F_i} が閾値超：公開の $\mathrm{PuniList}$ へ（Punishment）
   - 超えなければ，$\mathrm{SucList}_{F_i}$ に追加（Success）

c. **モデル統合**
   - $\mathrm{SucList}$ に含まれる各 Model_{F_i} を用いて，連合モデル Model_{FE} をローカルに融合

b-4. 終了処理

- 条件を満たすまで (規定のラウンド数など) 繰り返し
- 最終的に，*CTSC* が
  - PuniList の参加者のデポジットを没収
  - SucList の参加者へ仮想通貨で報酬を付与

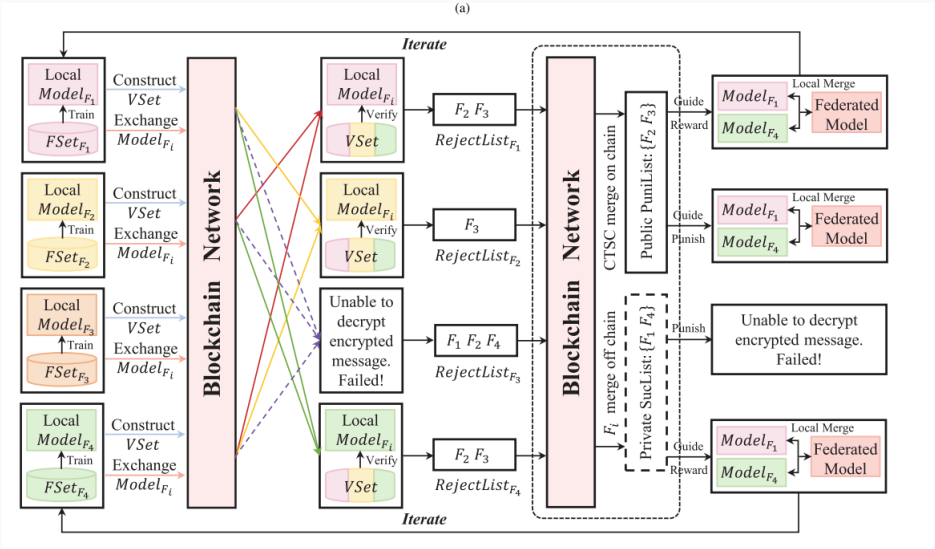
CTSC とセキュリティ

- CTSC：オンチェーン協調学習スマートコントラクト
- オンチェーン協調のセキュリティはスマートコントラクト上でメンバーリストに応じた関数呼び出し権限管理で担保
- 本稿では $\mathrm{ActiveList}_{FE}$ や $\mathrm{TrustList}_{FE}$ を公開・保存しない
- 各ステージの実行時間を厳格に設定し, 単一障害点の発生を防止

脅威モデル

1) フェデレーション外の誤呼び出し
   - 識別情報 $\{pk_{FE}, sk_{FE}, \dots\}$ は非公開
   - CTSC が公開かつ透明ゆえに，アウトサイダーの誤呼び出しが可能
   - 対策：登録時に正しい $\mathrm{len}(\mathrm{ActiveList}_{FE})$ を報告した者のみ $\mathrm{RejectList}_{F_i}$ 報告を許可
2) k_{Active} 非知悉者による妨害
   - (a) $\mathrm{len}(\mathrm{ActiveList}_{FE})$ は把握しても k_{Active} を知らない場合（図 1(a) の F_2, F_4 相当）
     - $\mathrm{RejectList}_{F_i}$ を構成できても復号不能
   - (b) 小規模協調下で総当たり攻撃により $\mathrm{TrustList}_{FE}$ や k_{Active} を解読
     - しかし $\mathrm{ActiveList}_{FE}$ に認識されなければ学習に寄与せず，最終モデルに影響しない

fig1


(a)
Iterate
Local $Model_{F_1}$
Train
$FSet_{F_1}$
Construct $VSet$
Exchange $Model_{F_i}$
Local $Model_{F_2}$
Train
$FSet_{F_2}$
Construct $VSet$
Exchange $Model_{F_i}$
Local $Model_{F_3}$
Train
$FSet_{F_3}$
Construct $VSet$
Exchange $Model_{F_i}$
Local $Model_{F_4}$
Train
$FSet_{F_4}$
Construct $VSet$
Exchange $Model_{F_i}$
Blockchain Network
Local $Model_{F_i}$
Verify
$VSet$
Local $Model_{F_i}$
Verify
$VSet$
Unable to decrypt encrypted message. Failed!
Local $Model_{F_i}$
Verify
$VSet$
F_2 F_3
$RejectList_{F_1}$
F_3
$RejectList_{F_2}$
F_1 F_2 F_4
$RejectList_{F_3}$
F_2 F_3
$RejectList_{F_4}$
Blockchain Network
CTSC merge on chain
Public PuniList: {F_2 F_3}
F_i merge off chain
Private SucList: {F_1 F_4}
Guide
Reward
Local Merge
$Model_{F_1}$
$Model_{F_4}$
Federated Model
Guide
Punish
Local Merge
$Model_{F_1}$
$Model_{F_4}$
Federated Model
Punish
Unable to decrypt encrypted message. Failed!
Guide
Reward
Local Merge
$Model_{F_1}$
$Model_{F_4}$
Federated Model
Iterate
(b)

罰則メカニズム

- 各イテレーションで $\mathrm{RejectList}_{F_i}$ の提出は 1 回に制限
- 全ての $F_i \in \mathrm{ActiveList}_{FE}$ に「正しい $\mathrm{len}(\mathrm{ActiveList}_{FE})$ を報告したが認識されていないメンバー」を監視し，$\mathrm{RejectList}_{F_i}$ に追加する義務
- $\mathrm{RejectList}_{F_i}$ ／ $\mathrm{PuniList}$ に含まれるのが悪意メンバーか正直メンバーかは区別困難
- 最終的に $\mathrm{PuniList}$ 登録者はデポジット全額没収
- 没収金 D_{Puni} は正直メンバーに均等分配

CTSC の主要な関数

- 登録：正しい $\mathrm{len}(\mathrm{ActiveList}_{FE})$ の報告
- VSet_{F_i} 交換：イベント発火＋モニタリング
- Model_{F_i} 交換：同上
- $\mathrm{RejectList}_{F_i}$ 報告：1 アカウント 1 回に制限
- デポジット引き出し：イテレーション終了後 1 回のみ
- 全関数は所定の呼び出し可能時間内にのみ実行可能
  |- Model_{FE} と資金の安全を保証

CTSC のまとめ

- プライベート FL 終了時，$\mathrm{ActiveList}_{FE}$ メンバーは理想的な Model_{FE} を取得
- 仮想通貨を通じて公平な報酬・罰則を実現

目次

得られた結果, 知見

- 提案手法はコラボレーションコストを削減しつつ
  - イーサリアム上での暗号通貨の支払額
  - 計算時間
- セキュリティやプライバシを担保

目次

テーマについて

- 何かしら連合学習に使えそう
- もし参考にするなら
  - IoT デバイスのデータを集約するサーバを用意
  - 各サーバが bc のブロックとなり,FL に参加
- 任意のノードを追加可能な設計にしたい
- bc でモデルのバージョン管理などできないか?
- 次読むなら
- Anton Wahrstatter et al. Openfl: A scalable and secure decentralized federated learning system on the ethereum blockchain. Internet of Things, 26:101174, 2024.

分かっていない/気になる点

- イーサリアムと IPFS をどう使い分けているか
  - 現状の認識
    - 前者がインセンティブや識別に利用?
    - 後者がファイル共有に強いらしいので, モデル交換?
- セキュリティとプライバシがどれくらい良いか
  - 類似手法と比較したい
  - そもそも bc を使わない FL とも比較したい
  - いまいち効果がピンときていない
  - 何かしら追試を行いたい
- 台帳上に保管されるデータはどれ?
  - モデルのパスくらい?

分かっていない/気になる点

- オフチェーンとオンチェーンとは何か
  - 前者が通常の FL
  - 後者が bc 上の FL?
  - bc 上の FL に参加できない $>$ ローカルで完結, という文脈?
- 結局どのように学習しているか
  - プライバシとセキュリティ担保の取り組みは一通り見た
  - P2P ネットワーク上でどのようにモデルを交換する?
  - 今の認識
    - 相互に信頼したリスト上の相手に逐次問い合わせ, 交換
    - 最終的に信頼されたノードのみ含まれるグローバルモデルが完成